Applying HCI Design Practices to the Development of the BrainEx User-Interface to facilitate fNIRS Research

Advisors:

Professor Erin Solovey and Professor Rodica Neamtu

By:
Kyra Bresnahan
Margaret Goodwin
Yihan Lin

Working collaboratively with:
Vandana Anand

A Major Qualifying Project
WORCESTER POLYTECHNIC INSTITUTE

Submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science.

August 2019 – March 2020

Authorship

This project was completed over the course of seven months of study. For five of those seven months, we worked as a team of four to complete our deliverables. A former version of this paper detailing this period of time submitted at Vandana Anand's completion of the project can be found through WPI's e-projects collection under the title "Applying HCI Design Practices to the Design of the BrainEx User-Interface to facilitate fNIRS Research." That paper includes a similar authorship page detailing the contributions that Vandana specifically contributed to. The rest of us (Kyra Bresnahan, Margaret Goodwin, and Yihan Lin) collaborated with Vandana in writing and editing all portions of this paper that are preserved from that past version. In addition, the three of us also collaborated on additional writing contained in this paper detailing our development process after Vandana's graduation. Our typical writing process involved one person writing the original version of a section, with all other members then suggesting edits. Therefore, the final product of these sections is work from all three of us.

Abstract

This project aims to develop a user-interface for BrainEx using HCI practices to enable fNIRS researchers to explore and analyze large datasets. The target users were identified through interviews with lab staff and developing user personas. Through iterative design, prototypes of increasing complexity and detail were designed, evaluated, and refined to satisfy user needs while fulfilling system requirements. The final application encompasses a user-friendly and tested interface that accomplishes the tool's most essential functionality.

Acknowledgements

Many thanks to Professor Erin Solovey and Professor Rodica Neamtu for their many hours of work advising the team. Also, thanks to all the user testing participants and developers of the BrainEx backend. All of their feedback throughout the process was invaluable for the success of this project. Lastly, thank you to all of our friends and family who supported us along the way.

Table of Contents

Table of Tables

Table of Figures

Executive Summary

BrainEx is a command-line Brain Computer Interface (BCI) application that allows researchers to find *k* best matches for time series sequences representing functional near-infrared spectroscopy (fNIRS) data (Dubey, et. al., 2019). The team iteratively prototyped and began implementing the frontend for the application using Human-Computer Interaction (HCI) methods and principles.

Human-Computer Interaction is a multi-disciplinary field that focuses on advancing user experience through methods such as iterative development and user-centered design (IDF, 2019a; Usability, 2017b; Mora, 2015). According to HCI principles, in order to maximize usability and utility of user interfaces, developers should strive for continuous and informative communication with the user in their application (Schneiderman, 2013). To create user-friendly applications, designers must gather user requirements and continuous feedback in the process known as iterative design. Iterative design, the process of creating prototypes of increasing detail and complexity while refining them based on feedback, allows developers to resolve problems early and make improvements quickly (Pidoco, n.d).

Brain Computer Interfaces are a newer concept in the realm of HCI. A BCI is an interface that allows computers to sense and collect brain signal data directly from the brain (Guger, et al., 2019). fNIRS is the use of near-infrared spectroscopy that allows researchers to measure blood hemoglobin levels to collect brain signal data (Grohol, 2017). It is relatively non-invasive and uses a portable cap and light sensor system. According to Tan and Nijholt (2010), many BCIs are often lacking in user-centered design is because the field of BCI "is just now coming out of its infancy". As fNIRS is a relatively new field, there is limited progress developing customizable and usable research tools that could widely apply to projects outside of the original developers' research scope. As a result, fNIRS researchers often develop their own tools for their own research, resulting in functionality being prioritized over usability (Anonymous lab researcher, personal interview, September 10, 2019).

The WPI HCI lab, led by Professor Erin Solovey, aims to conduct research on mind wandering and focus control using fNIRS data and various fNIRS research tools. Researchers in the lab perform data collection, preprocessing, and processing/data analysis. Each of these pieces of the overall lab workflow includes specific tools tailored to the task.

Recently, the HCI lab has begun to develop a new tool to join their current suite of tools. BrainEx is a data analysis tool for time series data that was developed to allow researchers in the WPI HCI Lab to efficiently explore the large amount of brain data collected from various experiments (Dubey, et. al., 2019). This tool allows users to find k best matches to a given time series sequence. The current BrainEx application has been designed to be a research-oriented tool that operates through the command line. In order to expand the user base and reduce the learning

effort, our team set the project goal to develop that interface through user-centered design and rapid prototyping.

Once the team's goal was decided, we accomplished the following objectives to complete the project:

1. Explored existing BCI tools by conducting usability analyses
2. Collected and analyzed user requirements to identify target users
3. Determined system requirements through design specification modeling and task analysis
4. Designed rapid prototypes using an iterative design strategy
5. Completed development of the basic pipeline of the proposed application
6. Gathered user feedback and planned for future development

In order to complete the first objective, the team rated multiple tools using predetermined usability parameters; the team also interviewed users in the lab that use the tools to find out their usability and utility. After completing this objective, our team had a better understanding of the current tools' strengths and weaknesses. Thus, we were able to better avoid the same flaws within their future design of the BrainEx interface.

The second objective required two parts. First, the team interviewed undergraduate and graduate lab staff to gather user-experience feedback about the current operation of the lab as well as collect user requirements for the BrainEx interface. From this, they were able to compile a set of user personas that reflect the current users within the lab to better have the user in mind when designing. The team also interviewed several developers of the BrainEx command line tool to gather specific details on its various functionalities and how a user-interface could best incorporate and transform these functionalities. The developer interviews gave the team the information they needed to create a simplified state diagram and conducted a task analysis based on potential user needs/tasks with BrainEx.

Once the team understood user needs, they created conceptual, semantic, syntactic, and lexical models outlining the functionalities and architecture of the BrainEx command line tool. These models allowed the team to fully understand the capability and limitations of the current BrainEx system. The models allowed the team to consider the systematic design of the application before designing the interface.

After completing the first three objectives, the team had enough understanding of the system to begin designing the interface. In order to adhere to the HCI principles of iterative and user-centered design, the team created four prototypes and received user feedback on each one.

1. First, to confirm that the team had understood user needs correctly, the team designed storyboards which outlined the prominent features of the application. The storyboards

were then presented to users for feedback. Based on the collected user feedback, the team corrected any major misunderstandings about the application and learned the importance of how different functionalities are communicated to the user and how each individual functionality should be presented.

2. Next, to establish the basic design structure of the prototype, the team designed a low-fidelity prototype on Balsamiq with interactions. The prototype was then presented to users for feedback. For this prototype, users were confused with the way user testing was conducted, leading to an improvement of testing style in future iterations. In terms of the prototype itself, users were overall lost at what to do and where to look for things. The team decided to focus on developing a more intuitive control flow to reduce confusion. The team also revised the user testing protocol to make the procedure more understandable.
3. To consolidate a more intuitive and navigable control flow of the prototype, the team designed a mid-fidelity prototype on Balsamiq with more interactions. The team also developed sitemaps to help with understanding during this iteration. Users were overall satisfied with the mid-fidelity prototype and indicated it was more intuitive than the previous iteration. More work could be done to reduce confusion risen from technical jargon or confusing names.
4. The team decided to pay attention to basic error handling, the flow within each page/screen, and including more visuals in the next prototype. To add visual elements (color palette, appearance of graphs, graph legends) to finalize the prototype design, the team designed mood boards and a high-fidelity prototype on AdobeXD for a more customizable design. The prototype was then presented to users for feedback.

The resulting high-fidelity prototype provides a concrete plan for the team's implementation of the interface. Users said that this prototype was easy to navigate when performing tasks. They completed their tasks quickly and were able to give more granular comments on the improvements to be made, such as clarification of language.

Next, the team compared different popular web development frameworks, such as React and AngularJS, and also looked at its compatibility with other features such as including visualizations with Javascript libraries like D3. During this period of time, Vandana also began the implementation of the application by programming a few interface pages in JavaScript.

Kyra, Margaret, and Yihan then continued the project by completing implementation. Kyra focused on creating a Python file that would allow information to be passed from the frontend to the backend and vice versa. Yihan focused on data visualization and Margaret focused on other frontend implementation.

Finally, we finished our project by soliciting additional user feedback. We then applied any minor user-experience changes suggested by users. We suggest that future developers in the

WPI HCI Lab continue to develop this application and add more useful features such as exploring both the raw and preprocessed data.

1. Introduction

BrainEx is a tool developed at WPI to facilitate Brain-Computer Interfaces (BCI) research, specifically in functional near-infrared spectroscopy (fNIRS) data (Dubey, et. al., 2019). The goal of the tool is to find the *k* best matches for a subsequence of time series data of their choosing. It improves upon earlier tools with similar goals by using distributed computing to cluster similar sequences. This allows for BrainEx to find matches faster by minimizing computations. The accuracy and efficiency of the tool have been tested using data collected from fNIRS experiments within WPI's HCI Lab. In these experiments, researchers gather concentration or focus data from the brain using an fNIRS cap.

While the BrainEx tool is effective in achieving its goals, it is currently only a command line tool. To increase its usability and accessibility to more researchers in the WPI HCI lab, this tool needs a graphic user-interface (GUI). However, tools developed for fNIRS research are often ad hoc and development is more focused on functionality rather than usability. Therefore, this team's project aims to facilitate the research of fNIRS at the WPI HCI Lab by developing an intuitive graphical user-interface for BrainEx using effective Human Computer Interaction (HCI) design practices as an improvement on current tools.

HCI has existed since the 1970s; researchers in this field study best practices in interactivity for people working with computers (IDF, 2019a). The goal is to make computer applications user-friendly by focusing on utility and usability. This means ensuring that applications are functional and easy to use. In developing an interface for BrainEx using user-centered design, the team will streamline the workflow of the lab by reducing the time needed to learn and retain memory of the function of the application.

The team met their goal of applying HCI design practices to BrainEx by identifying a methodology. The first objective of this methodology was to analyze the usability of existing BCI tools to identify gaps in understanding and user-experience. Then, the team identified and collected the necessary user requirements for the interface through completing user and system analysis. Next, the team outlined the system specifications and designed rapid prototypes of the BrainEx interface using an iterative design strategy. The team made sure to perform evaluations of the application among themselves and conduct user testing sessions with potential users to make sure prototypes are meeting the user's expectations throughout the design process. Finally, all members of the team contributed to the development of the project. By completing these objectives, the team hopes to show the benefits of applying HCI to developing research tools, improve the BCI pipeline and efficiency of BCI research at WPI, and identify more areas for development.

2. Background

2.1 An Introduction to Human-Computer Interaction

Human-computer interaction (HCI) refers to the study of how the relationship between humans and technology can advance user experience (IDF, 2019a). Since the 1970s, HCI has increasingly become a vital part of developing technology with the rise of personal computers making it necessary for technology to be more widely usable.

One goal of HCI is to identify a user-experience problem (e.g. accessibility issues caused by disabilities, complex processes, etc.) and solve the problem through user-centered design and iterative development (Carroll, 2012; Algrim, 2019). Feedback from users drives designs forward as designs are refined to fit users' needs in both functionality and usability. Through every step, designers and developers check their understanding of user-specified requirements until a fully realized product has been created. By following HCI guidelines and applying its concepts to development, people can create useful products that are easy, if not enjoyable to use.

2.1.1 The Principles of HCI

To ensure that the systems developed are well-designed and useable, Schneiderman developed eight important principles (Schneiderman, 2013). These principles include:

- **strive for consistency**: consistency in both actions and visuals (e.g. terminology, prompts, menus, etc.) should be maintained throughout the application, especially in similar situations;
- **enable frequent users to use shortcuts**: as a user becomes more familiar with an application, they will want to reduce the time spent performing actions by using various shortcuts;
- **offer informative feedback**: each action prompts some form of feedback from the system, correlating to the complexity and importance of the action,
- **design dialog to yield closure**: related actions should be consolidated into one package that offers the user some sense of accomplishment when each set is completed;
- **offer simple error handling**: when a user makes a serious error, the system should both detect it and offer a simple solution;
- **permit easy reversal of actions**: allow a user to undo a recently performed action to reduce anxiety if they make an error;
- **support internal locus of control**: design the system so that the user initiates the actions rather than the system so that they feel in control of the application; and

- **reduce short-term memory load**: keep displays simple, functionality consolidated, and distractions limited to not force the user to remember more than necessary.

In short, continuous and relevant communication between the user and the system is essential for usability. The workflow of an application should be as simplified as possible while still accomplishing a task as desired. No matter the specific needs of the user, these principles should be followed during the design process.

2.1.2 The User-centered and Iterative Design Process

The user-centered design process (UCD) ensures that common mistakes such as inefficient development practices, unmanaged risks, poor communication, etc. are avoided (Usability, 2017b). There are four basic activities in the initial design process (Usability, 2017a):

1. understanding the problem space,
2. identifying user requirements for a useful product,
3. creating interactive versions of the design(s), and
4. testing and evaluation of the design(s) with users.

Before tackling any problem, it is important to understand the environment in which the problem exists. Designers should consider who will use the product, what they will use it for, and how they will use it, also referred to as user requirements, through interviews, focus groups, surveys, and other methods. The user requirements can then be transformed into the initial design.

Those using UCD concepts create several successive designs, or prototypes, of their product that increase in detail and complexity until the final design is fully realized (Mora, 2015). A prototype is a powerful and effective way to quickly collect feedback on a design or product and they can take many forms (IDF, 2019b). The complexity ranges from simple, low-fidelity prototypes to high-fidelity ones with visuals and interactions (Solovey, 2019g). Prototyping is important because, in the initial data collection stage, feedback from target users is based on either existing products or a description of features that do not yet exist. Users share what they *might* think or do given their mental model and the information provided without a concrete example in front of them to which to react. While this is useful when starting to develop an initial prototype, it does not lead to a perfect product. With concrete examples, the user can demonstrate the usability of the design in real-time and save costly development time (Usability, 2017b).

Prototypes of how the product will look are created and tested to refine how elements are arranged and tasks are represented. An initial prototype is created and tested with users; any issues with the prototype (appearance, control flow, clarity, etc.) are recorded and analyzed. A

report of these findings, including prioritized Usability Aspect Reports (UARs) that detail critical issues (how many users experienced them, what kind of issue was it, the severity of the issue, etc.) and a summary of general findings can give valuable insight into what changes to make in the next iteration (Affairs, 2013). Before any actual implementation is done, developers *iterate* through this process to create the best design solution possible, which saves both time and money for both the developers and the stakeholders (Usability, 2017b). The next iteration is then refined to eliminate problems, and this process repeats until the product is ready (Pidoco, n.d.). These iterations must be created quickly, making wireframing and prototyping tools very useful. An illustration of this process can be seen in the figure below:

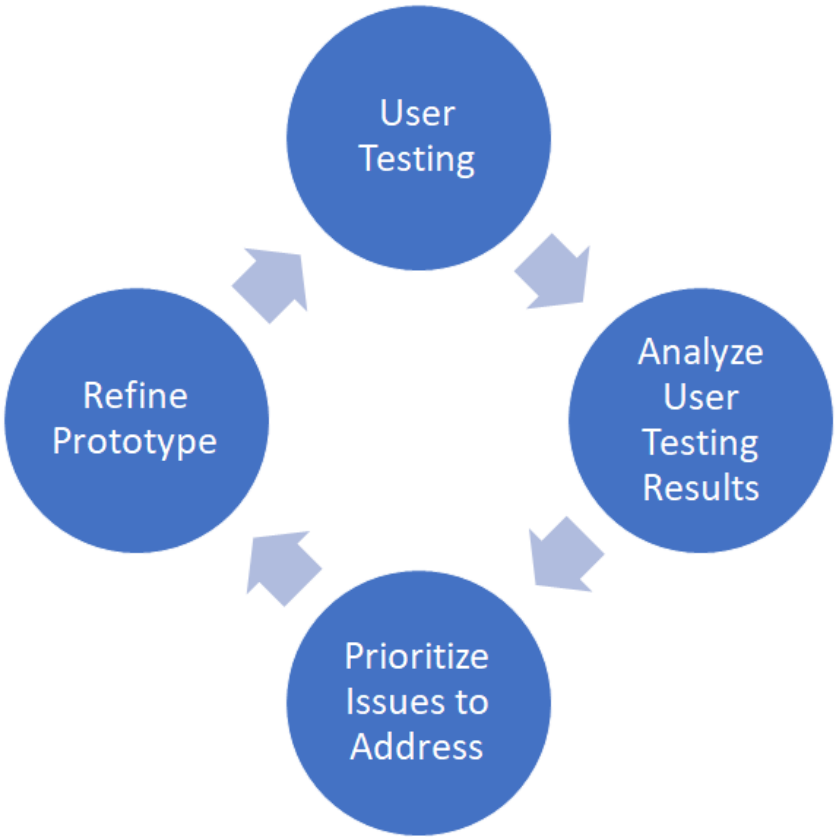


Figure 1: An illustration of the iterative design cycle through rapid prototyping. Adapted from Iterative Design by Pidoco. n.d. Retrieved from https://pidoco.com/en/help/ux/iterative-design

One of the most important reasons to use iterative design and prototyping is that it results in a much more usable application (Affairs, 2013). However, it also helps developers eliminate flaws in early stages of development that would otherwise be expensive to fix later. With constant user feedback throughout the development process, the product evolves according to the *user's* needs, thus resulting in the most useful and cost-effective solution.

2.2 Brain-Computer Interfaces

Brain-Computer Interfaces (BCI) are a newer space in the field of Human-Computer Interaction. BCIs have provided a new method for people to convey messages with brain data. These technologies collect real-time streams of brain data from people performing cognitive activities while a signal detecting device receives their brain data. According to Guger et al., (2019), four main components must exist in all BCIs:

1. sensors that can detect brain activity (most of which are non-invasive),
2. automated signal processing software that is used to identify brain activity,
3. an external device that provides feedback based on the processed signal, and
4. an operating environment that controls how the above three components interact with each other as well as the end-user.

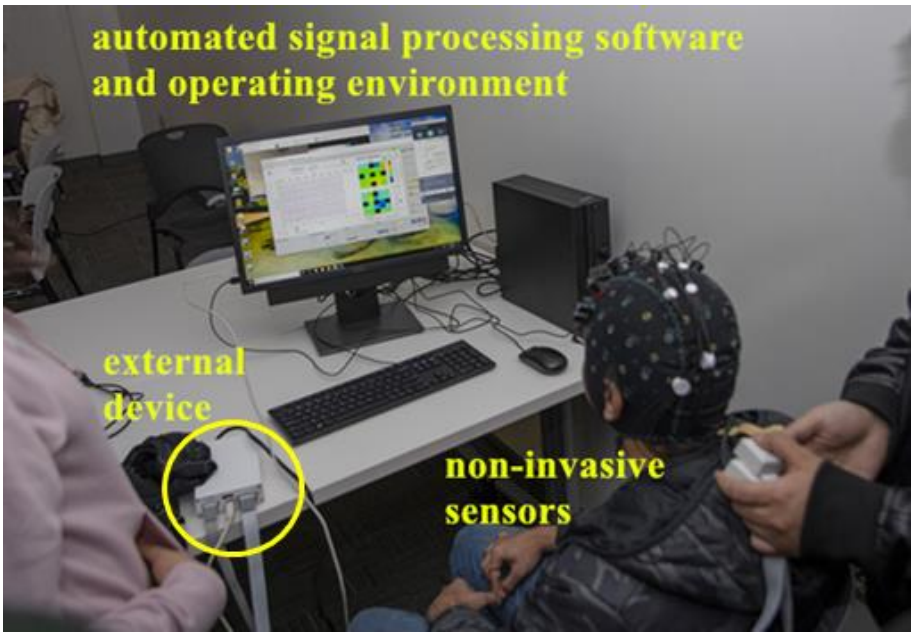


Figure 2: An example BCI annotated with the four main components

In the early stages of BCI research, most researchers were focused on BCIs that could facilitate communication for disabled people (Guger et al., 2019). In the past several years, BCI research has been extended to many new applications outside of the medical field, such as education (Brockington et al., 2018).

2.3 Functional Near-Infrared Spectroscopy (fNIRS)

Functional Near-Infrared Spectroscopy (fNIRS) is a type of functional neuroimaging technology that offers a non-invasive, safe, portable, and low-cost method of indirect and direct monitoring of brain activity. It allows researchers to collect brain data through a cognitive brain monitor and monitor blood flow and oxygen levels in the various parts of the brain by measuring changes in near-infrared light. It is a relatively new technique but has shown promising results in studies done to-date (Grohol, 2017).

During fNIRS experiments, users wear caps with sensors to monitor brain activity. The fNIRS sensor is attached to the user's cap, as shown in Figure 3 below, and can be monitored through a BCI that is either connected directly to a computer or a portable computing device that records the user's data as they engage in specific tasks. The advanced signal processing allows real-time brain data collection during the execution of the task (Grohol, 2017). Changes in brain activity are then measured by blood hemoglobin — the protein molecule in red blood cells that carries oxygen from the lungs to the body's tissues and returns carbon dioxide from the tissues back to the lungs — and oxygenation levels in particular brain regions. One of the important brain regions that is most commonly measured is the prefrontal cortex because it is the part of the brain that is responsible for planning complex cognitive behavior, personality expression, decision making, and moderating social behavior (Grohol, 2017). Depending on the researcher's preferences, collected fNIRS data is parsed and stored so it can be used for further research to test hypotheses on brain activity and workload (University of Connecticut, 2017).

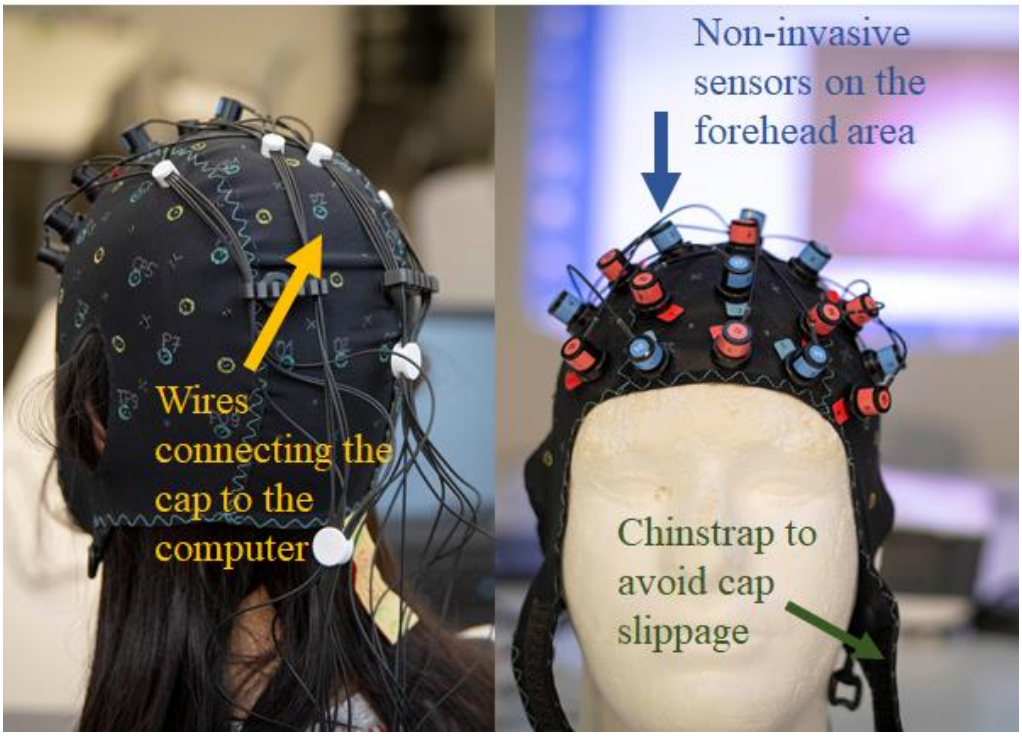


Figure 3: Pictures of an fNIRS cap, front and back, from the WPI HCI lab. Note the annotated features that allow the cap to collect data.

Some of the many reasons to use the fNIRS-BCI system are because it is safe, can produce accurate results, and is portable. The fNIRS cap that the users wear emits no more light into the user's brain than the amount of sunlight that human skin is exposed to while walking outside, making it largely harmless to the wearer (University of Connecticut, 2017). In addition, fNIRS can produce highly accurate results of brain data collection because it is more tolerant of errors such as the motion of the sensors on the cap (NASA, 2019). It is advantageous over other neuroimaging systems because it directly measures blood oxygenation levels (Tak & Ye, 2013). Moreover, fNIRS is portable as it can easily be taken anywhere and does not take up much space.

2.4 The WPI HCI Lab

The WPI HCI Lab, led by Professor Erin Solovey, defines one of its goals as conducting research to seek ways to classify cognitive states of mind wandering and focus control with fNIRS-based brain data. Students of various educational backgrounds and progress collaborate with Professor Solovey and other, sometimes interdisciplinary, professors to facilitate research and develop tools to assist in that research. The research is divided into three overarching stages: data collection (i.e. using the fNIRS brain cap and data collection tools), preprocessing (i.e. removing noise and truncating unneeded data), and processing (i.e. data exploration and analysis). The lab uses both open source fNIRS data analysis tools (both from external and

internal developers) and proprietary software developed by NIRx Medical Technologies to perform each stage. Figure 4 below illustrates the main three stages as well as the goals and tools that fall under each phase. Each stage is also annotated with known areas of improvement if there are any. This section will then give a brief overview of each tool listed, as well as any tools used in the past. More information about each of the tools as well as an analysis of their usability can be found in Chapter 3, as understanding the current solutions to a problem is crucial in developing a new one.

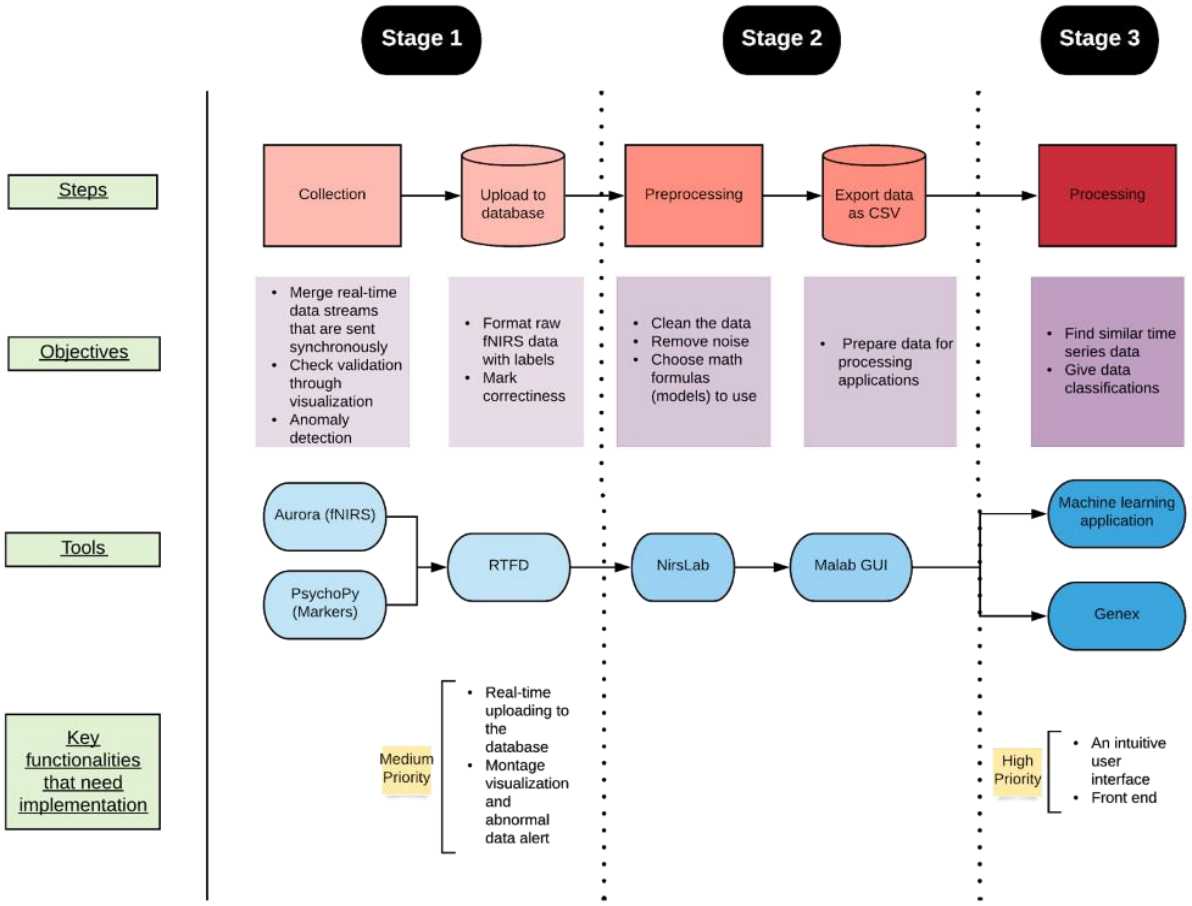


Figure 4: Diagram of the overall workflow in the WPI HCI Lab. Processing applications denoted in Stage 3 have not been fully developed at the time of this diagram's creation.

2.4.1 Data Collection Tools

Aurora (see Figure 5) is a tool designed to acquire fNIRS data. It is able to establish a wireless connection with the fNIRS device known as the NIRSport2. Users can create multiple configurations in Aurora, allowing various ways of measuring data with different regions of the brain (see Figure 5). It also provides basic functionalities like displaying montage (i.e.

visualization for monitoring the channel connections) and data plots to ensure a smooth data collection process.

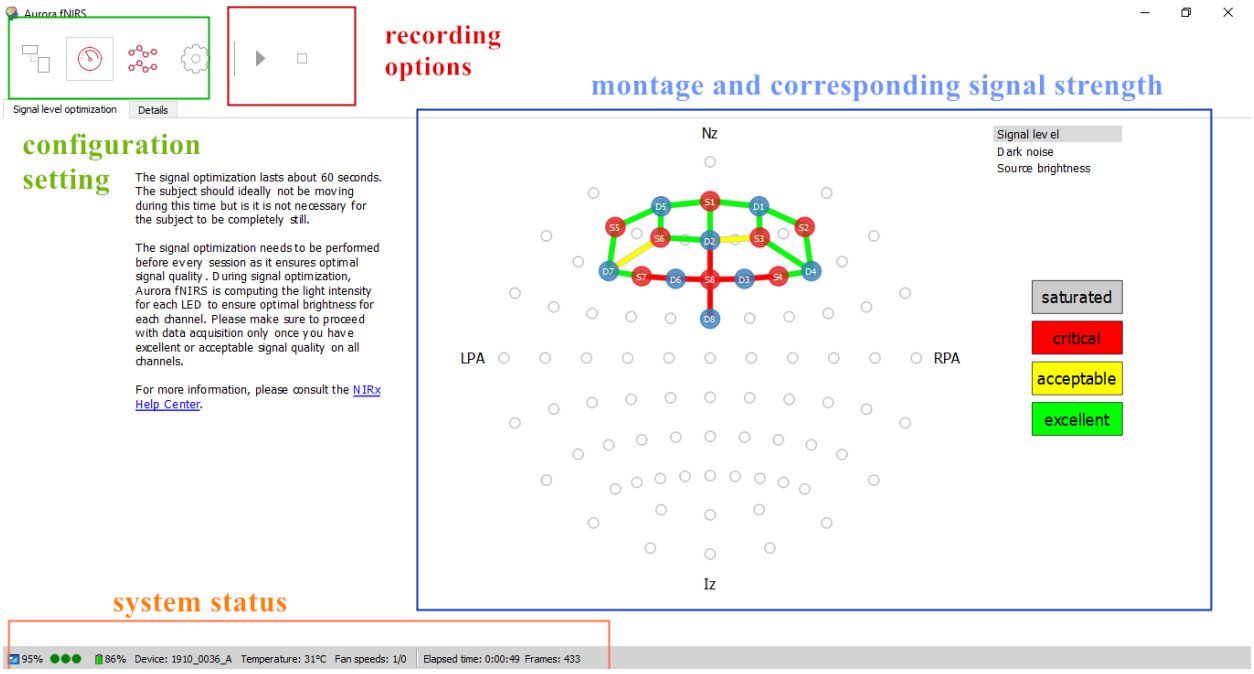


Figure 5: Signal Calibration screen from Aurora. November 17, 2019.

The Real-Time fNIRS Data-analysis (RTFD) tool (see Figure 6) was designed by WPI students working in the lab to facilitate the fNIRS data collection process in conjunction with Aurora. It parses the fNIRS brain data from Aurora into the CSV format simultaneously as the application receives it. As of right now, it provides a user interface for uploading the data to the WPI HCI lab server. The developers are hoping to incorporate visualizations of different channels into RTFD and some basic error handling prompts in the future.

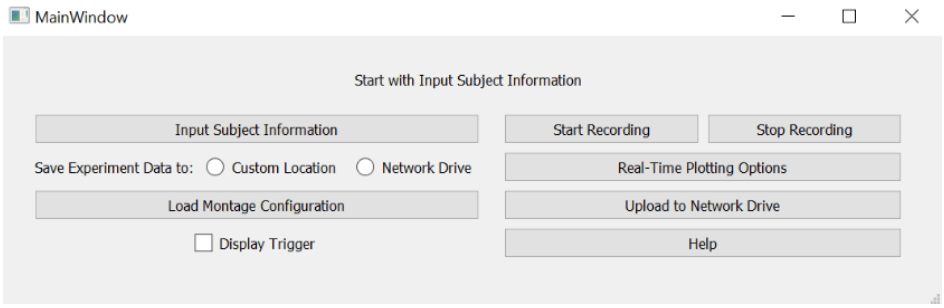


Figure 6: RTFD developed by WPI students working from the HCI Lab. November 17, 2019.

2.4.2 Data Preprocessing Tools

NirsLab (see Figure 7) is used as a preprocessing tool to prepare data so that additional operations such as machine learning algorithms can be applied to further analyze and draw

conclusions from the data. The application has many features for preprocessing data, but the lab primarily uses the truncate time series, check raw data, and apply frequency filter methods. The data analysis features are also used to view useful graphs and visualize the data.

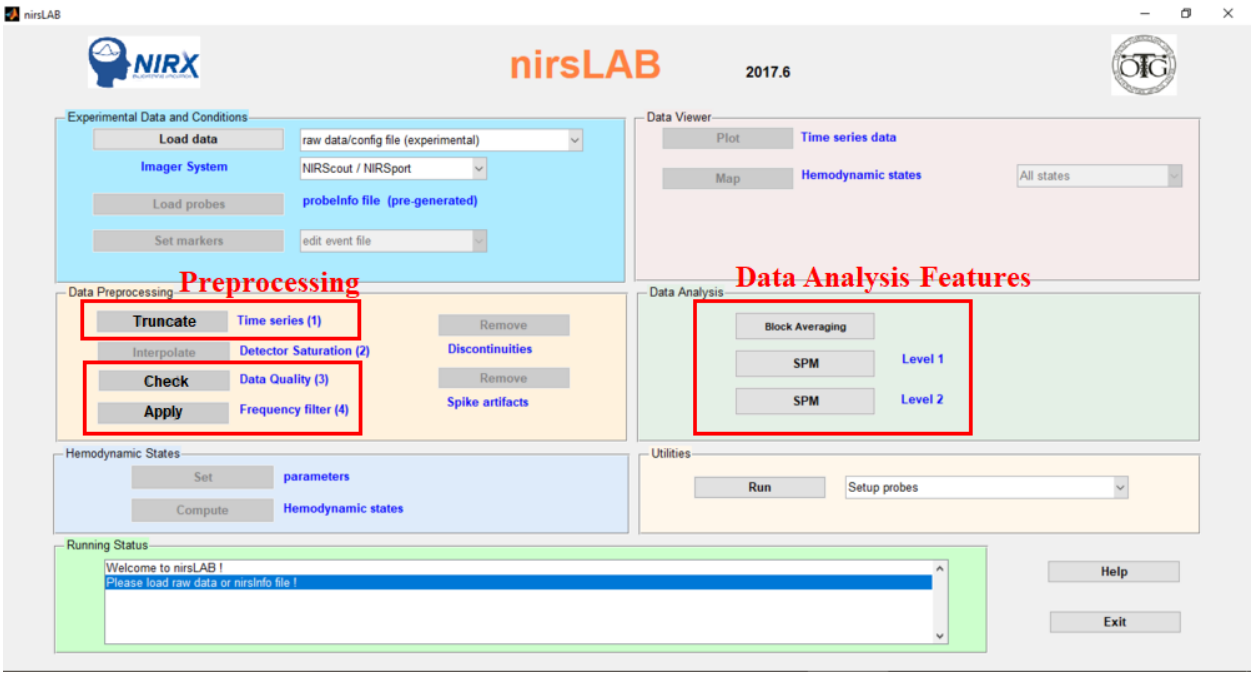


Figure 7: Screenshot from NirsLab with the truncate, check, apply, and data analysis functionalities highlighted. November 2019.

The **Matlab_GUI** (see Figure 8) is a graphical user interface developed in MATLAB by Professor Solovey's previous students at Drexel University that is used to preprocess collected experimental fNIRS data offline. The functionalities implemented in the GUI were designed to streamline analysis of fNIRS data by allowing users to visualize the whole time series, translate from raw data to de-oxy/oxy hemoglobin values, and view specific time intervals. It also allows users to export the data they are viewing in either CSV or *nirs format.

Figure 8: Screenshot of the Matlab GUI developed by Drexel students. Screenshot Retrieved November 20, 2019.

2.4.3 Data Processing/Analysis Tools

Homer2 (see Figure 9) is a Matlab-based application that has been around since the early 1990s (NITRC, 2019). According to the official documentation (Homer-fNIRS, n.d.), the software has been widely applied to fNIRS-based projects and has many processing methods that have been implemented to support various kinds of fNIRS-based research. Its primary purpose is to convert fNIRS data into maps of brain activation so the data can be viewed, analyzed and processed further down in the data handling pipeline (fNIRS Analysis, 2019). All the functions can also be executed at the script level, allowing for more flexibility.

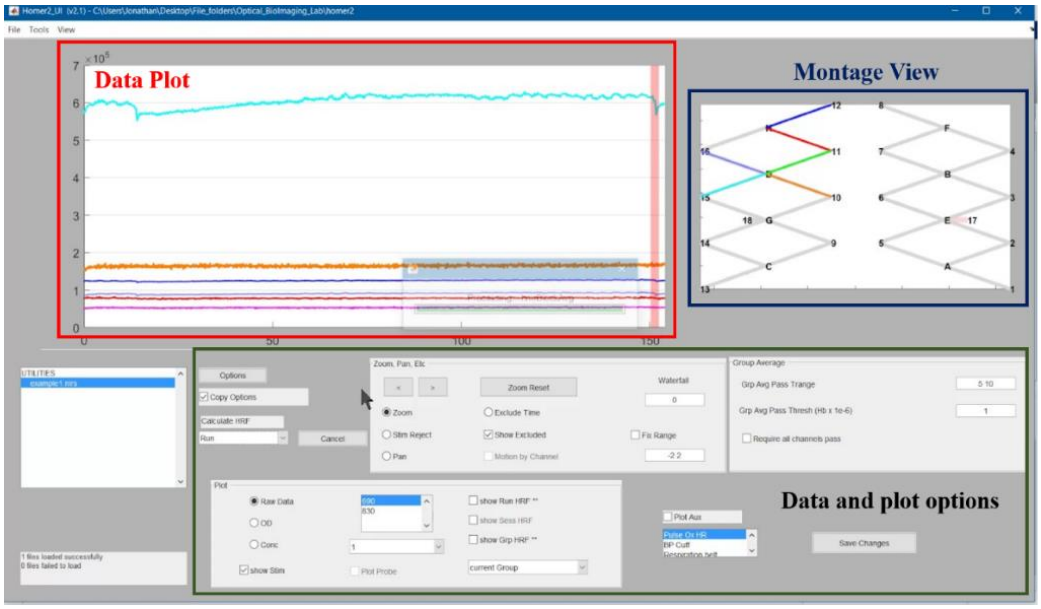


Figure 9: Homer2 screenshot. Adapted from Homer2, 2017. Retrieved from https://www.youtube.com/watch?v=MM4CB6K2Nec.

2.5 BrainEx

When learning the overall pipeline pictured in Figure 4, the team discovered that a WPI team was currently developing a command line tool to help with data analysis. BrainEx is a tool designed for similarity exploration of brain data for neuroadaptive technology (Dubey et al., 2019). It uses “different similarity distances for robust identification of similar patterns in the brain data during complex tasks”. In short, it finds the k best matches for a user supplied time series sequence.

While classifying continuous time series data has remained a challenge in neuroadaptive technology, BrainEx approaches this problem by using dynamic time warping to compute the similarity between sequences with different lengths and temporal alignments (Rakthanmanon et al., 2012). Common issues within large datasets such as computational overhead are solved by using a “process one, query many” approach to effectively reduce the data mining space. Using simple-to-compute pointwise distances including Euclidean, Manhattan, Chebyshev, etc, the resulting dataset is reduced in size which makes exploration of specific warped counterpart distances more efficient. The application uses the time warped versions of these distances to improve similarity calculations for time series data. The below screenshot of the provisional interface created by the development team shows the annotated features of Brainex (see Figure 10).

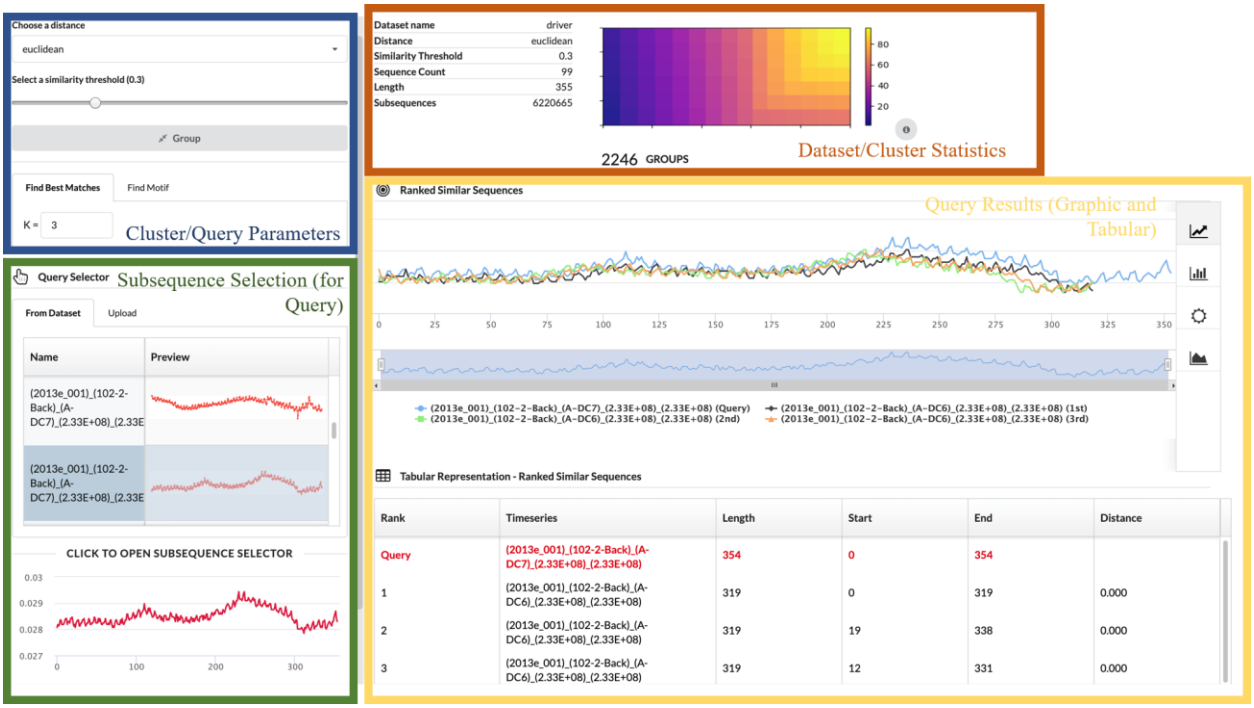


Figure 10: Screenshot from the BrainEx UI. Adapted from Dubey et al., 2019.

In a recent proceeding by Dubey et al. published in 2019, BrainEx showed promising evidence for supporting time domain data exploration to identify similar sequences of brain data. It is capable of performing robust identification of similar patterns in the brain data during complex tasks using different similarity distances. This will serve as the foundation for interactive systems allowing cognitive states and adapting system behaviors to be better classified in the future.

2.6 HCI and fNIRS

The fNIRS-based BCI tools are limited as they often cannot deliver exactly what the lab team wants to achieve. Many of these tools are developed by neuroscientists who have specific research needs that may not match up with another lab's needs (anonymous lab researcher, personal interview, September 10, 2019). Hence, sometimes the importance of making the tool intuitive and easy-to-use for novices is overlooked.

In addition, one of the reasons user-centered design is often lacking in BCI applications is because the field of BCI "is just now coming out of its infancy" (Tan and Nijholt, 2010). The emerging state of the field leaves very few resources in past research and existent tools for researchers. Professor Solovey suggests that her lab's practice of using a combination of off-the-shelf tools and custom-made tools is common practice across the field due to this gap (personal correspondence, August 28, 2019). As such, most research is currently focused on the

development of the tools themselves, leaving less time for creating robust interfaces for said tools (Tan and Nijholt, 2010).

For example, the original BrainEx user-interface was developed based on a list of required features and not necessarily focusing on usability. The tool, while useful for research, would benefit from an intuitive interface so that even novice lab staff would be able to learn it with minimal assistance. In addition, the development team expressed the need to have a new graphic user interface (GUI) for their improved backend since it is difficult for users of varied technical background to learn the command line tools to use the product (personal correspondence, August 28, 2019).

2.7 Project Objectives

Once the team decided to focus on creating a user interface for BrainEx, they created plans and timelines for the project. The overall goal of the project is to facilitate fNIRS research at the WPI HCI Lab and streamline workflow by developing an intuitive user-interface for BrainEx using HCI design practices. The team accomplished the following objectives to complete the project:

1. Explored existing BCI tools by conducting usability analyses
2. Collected and analyzed user requirements to identify target users
3. Determined system requirements through design specification modeling and task analysis
4. Designed rapid prototypes using an iterative design strategy
5. Completed development of the main pipeline of the proposed application
6. Gathered user feedback and planned for future development

These objectives are broken down into tasks and further detailed in the following chapters of this report.

3. Exploration of Existing Tools to Identify Gaps in the Current Approach to BCI Tools

3.1 Process

To explore the current approach to BCI tools, the team analyzed the existing tools used in the lab across all phases — Aurora, RTFD, nirsLAB, the Matlab GUI, and the original BrainEx interface (Although it was mentioned in the background, Homer2 was not analyzed because its use in the lab has declined) — based on the following parameters (Nielsen, 2012):

- Effectiveness: How good is a system at doing what it is supposed to do?
- Efficiency: Once users have learned the design, how quickly can they perform the task?
- Safety: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?
- Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?

Each category was given a rating on a scale of 1 to 5, with 1 being “not satisfied” and 5 being “completely satisfied”. The team also conducted a more in-depth analysis on each tool that highlights key features of the tools and their usability. The analysis required each team member to successfully install and use their designated tool based on the given documentation and consult with other lab staff when necessary. By identifying both the strengths and weaknesses of these tools, the team developed an improved understanding of the current usability of BCI tools and where they can be improved.

3.2 Outcome of Tool Analysis

The team first qualitatively analyzed and assessed each major feature within the tools before assigning a quantitative value to each category listed above. A synthesis of our findings and our major takeaways can be found below the individual analyses.

3.2.1 Aurora Tool Analysis

As introduced in the Background Section, Aurora is the primary fNIRS data collection tool used in the WPI HCI Lab. It performs data collection smoothly and provides error handling functionalities in case of signal loss. One of Aurora’s strengths is its visualizations of montage

and various channels (see Figure 5 from Section 2.4.1 for the visual representation of montage). Although it is rare to experience any data loss on Aurora, Aurora does offer warnings in case of unstable signals. Aurora seldom crashes; once it does, there is no way to recover from the data loss. One concerning disadvantage Aurora has is its limited capability of customization. It does not support data files to be exported in alternative formats, nor does it allow users to change where data files can be saved. The control flow also forces the users to traverse back and forth between screens in less intuitive ways that consume user time and memory. Additionally, there are unnecessary warning dialog windows that cannot be hidden once read. Nonetheless, Aurora is still a learnable tool: only a few commands need to be remembered in order for users to accomplish an ordinary task. The interface is not perfectly intuitive but only requires a small learning curve. Also, because the steps required to accomplish a typical task like collecting brain data for an experiment are minimal, Aurora makes it easy for users to reestablish proficiency after a long time of no use.

3.2.2 Real-time Data Streaming and Analysis (RTFD) Tool Analysis

RTFD was designed to assist with data collection which is normally done on Aurora by simultaneously writing the received fNIRS data in CSV files. RTFD is still a work in progress and the developers planned to develop more functionalities such as plotting and error prevention. As of right now, RTFD is capable of integrating the data streams from both PsychoPy, a package for neuroscientific research, and from Aurora. However, there are a few bugs which could be detrimental to the users, primarily due to the lack of error handling or prevention functionalities. For instance, if the user attempts to start recording fNIRS data before the connection with the fNIRS device is established, it can cause the program to freeze without prior warning, as shown in Figure 11.

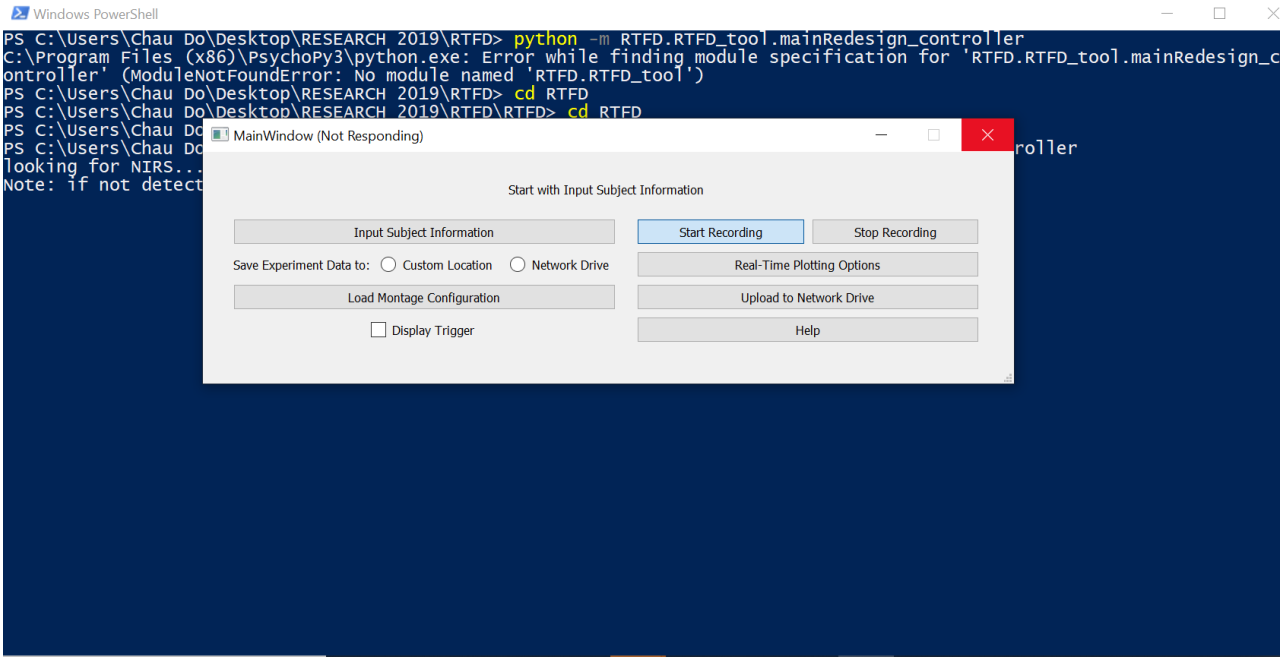


Figure 11: RTFD Not Responding screen

It has also been found that some computers are unable to detect fNIRS data through RTFD and the reason is still under investigation. If the recording process goes smoothly without freezing, there are only a few clicks required for the user to complete the recording task on this program. Users can then upload data files to the cloud within the program to streamline the workflow. As discussed before, RTFD is error-prone. Once the program window is frozen, all data points recorded are lost, and there is no way to recover the lost data. Aside from that, RTFD offers a shortcut for users to update file names with ease. It has a relatively simple user interface and most functions are self-explanatory. Because the interface is relatively straightforward, users typically do not have issues with memorizing the commands required to complete a typical recording task.

3.2.3 Matlab_GUI Tool Analysis

The Matlab_GUI was developed to assist in the data visualization and analysis of fNIRS brain data. It pulls the fNIRS data from a postgreSQL database and inputs it into the GUI. To run the application, the user must have the latest version of Matlab installed and several additional plug-ins that are not listed in the provided installation instructions. However, when the user attempts to run the application, which they can do through the command line, it will prompt the user to install the required plug-ins, as shown in Figure 12 The user will be unable to open the

application before individually installing the tools and restarting the Matlab software, preventing progress within their workflow.

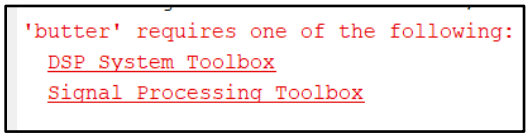


Figure 12: Error message received if required plug-ins not installed

The application takes an average of ten seconds to open to an empty plot and prompts the user to select a specific research subject. While this is happening, in the command window it shows the loading progress in the format shown in Figure 13. There is no indication of progress on the actual GUI screen. The user must switch between viewing the GUI and the command window in order to interact with the entirety of its functionality.

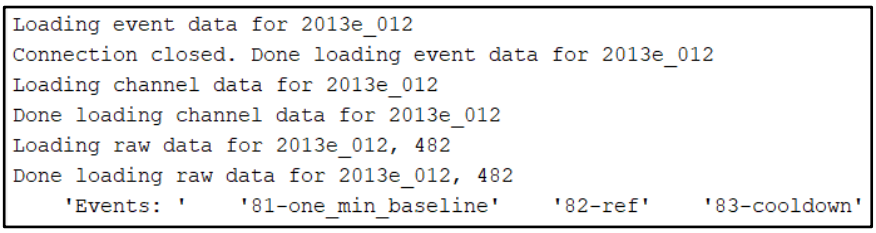


Figure 13: Progress indicator on command line

For the data visualizations, the user can only view time-series from one subject at a time, but they can view multiple channels and events for that subject. To select multiple channels and/or events, the user must use traditional keyboard shortcuts CTRL+Select and Shift+Select and all but the current selection disappears if the shortcuts are not used, as shown in Figure 14**.**

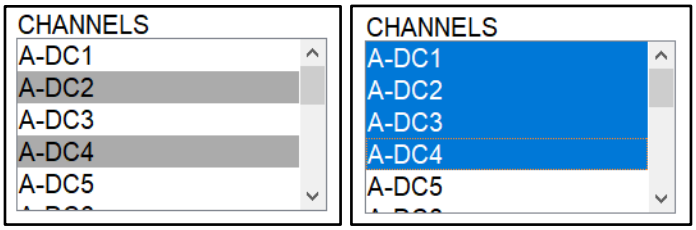


Figure 14: Channel selection

Whenever the user wants to view their selection on the plot, they must select the “Plot Data” button. This process can take anywhere from 3 to 60 seconds or more. When a different channel or event selection is made, the plot content is erased until “Plot Data” is pressed again. There are several plot manipulation options that appear when the user hovers their mouse over the graph such as panning and, brushing. The icons are small in proportion to the rest of the

application and faint in color against the white background of the GUI, which Figure 15 demonstrates.

Figure 15: Plot manipulation icons are faint

There is also a button that restarts the application that is grouped with other more commonly-used buttons such as “Export to Homer” and “Help”, shown below in Figure 16.

Figure 16: Other options are easier to find

This “Restart” button then closes the application after a few seconds and then reopens it with all previously selected options cleared. This could be detrimental to the user’s workflow if they were to select it by mistake when exporting the data.

The error messages are at the code-level, making it difficult to decipher what is wrong. For example, when trying to export to CSV, the following error message is shown in the command window as shown in Figure 17.

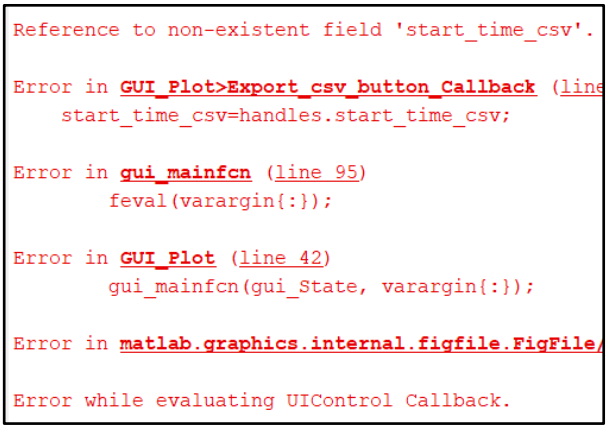


Figure 17: Errors are only shown at the code-level

The given error messages refer to specific variables and function names that are not exposed to the user through the GUI. There is no notification for the user within the GUI that an error has occurred and the user is unlikely to notice right away if they are not looking at the command window, nor would they be able to understand the error without a deep knowledge of the codebase.

Overall, the Matlab_GUI has sufficient functionality for its purpose, but the lack of intuitiveness within the application and the low-level error reporting detract from its usability.

3.2.4 BrainEx Tool Analysis

While there is a need for a new interface, there is a provisional interface for an older version of BrainEx that the team was able to analyze. The BrainEx tool's purpose is to aid the user in analysis of their previously collected time series data by helping them find the k best matches for their desired sequence (Dubey, et al. 2019). It accomplishes its goal by prompting the user for input on loading data on the upper left portion of the screen. Dropdown menus for this portion make selection easy, but some of the labeling relies on technical jargon. Once the data has been loaded, the user may choose a sequence to query by scrolling through different thumbnails of the data. The user may also zoom into a sequence to select smaller subsequences if desired. This is convenient because it is easy to see graphically how different subsequences related to the larger sequence, but is displayed in the only pop-out window in the application. The user may also input desired criteria for the query. Again, these criteria tend to overuse inaccessible terminology. In addition, the querying input section of the application is located in the lower left portion of the application. It is not delineated well from the data loading input section, which could cause users to confuse their purposes. Finally, data is displayed on the right side of the screen. After clustering, different statistics and a group density cluster map are shown on the results panel. The group density cluster map, shown in Figure X, is difficult to interpret as it is not in an easily recognizable format. However, the query results, as displayed graphically and tabularly, are easy to mentally parse. Overall, the application is a safe option as data loss and crashing are both rarities. However, there are some functionalities such as cluster exploration that would improve the application's ability to reach its goal. In addition, the unintuitive layout and overuse of complicated technical terms gives this application a steep learning curve.

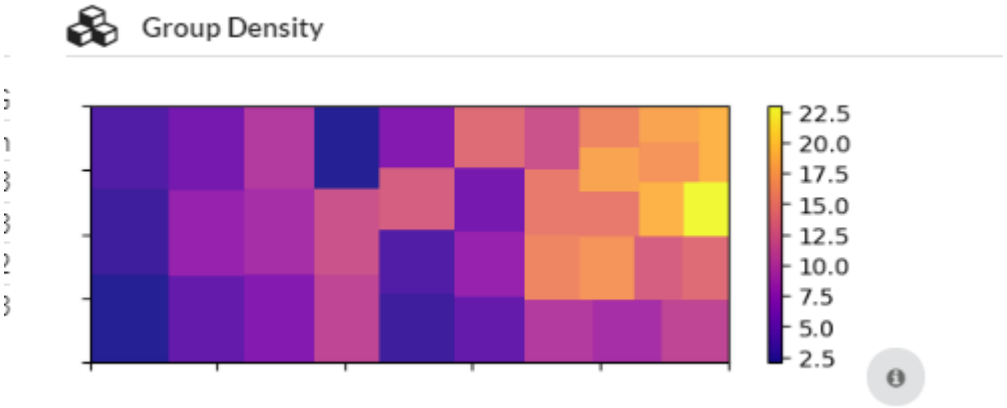


Figure 18: The Group Density mapping of clusters on the BrainEx UI is hard to parse as it is difficult to understand its presentation of data (Retrieved from BrainnEx November 4, 2019).

3.2.5 NirsLAB

NirsLAB is an fNIRS data analysis tool that is used to preprocess and analyze the data. It accomplishes this goal by providing the user with options such as truncating the time series data, being able to check the raw data, and applying filters to narrow down the focus. The users are also able to view visuals or graphs to see the results of the data. In this aspect, the tool is effective to the users, but the application is not customizable which makes it difficult to fit specific user needs. The application is efficient in that it is intuitive to the user to be able to clearly see all the labels for the features in the interface. However, some features in the interface are slow after a user clicks on it, which could be improved. Additionally, the steps to be taken to process the data are also labeled with numbers in the order in which they should be executed so the user knows how to immediately start handling the fNIRS data. When the user clicks on one of these steps out of order, there is error handling dialogue displayed as a warning to the user. However, sometimes the error messages are not meaningful or do not display. It is also rare for the application to crash, but if it does happen, the user can lose all their data and will have to restart. In terms of learnability, the application is easy to learn because there is only one procedure the user must follow in order for the functionality to work. However, there is limited or no documentation for solving technical issues which could waste a lot of the user's time. Regarding memorability, it is easy for the user to come back to the interface and know how to perform all the steps because there is only one path through. Overall, the application is easy to pick up and learn, but there could be more improvements to make the application customizable to the user's needs, time efficiency, and effective error handling dialogues.

3.2.6 Conclusion to Tool Analysis

Through their tool analysis described in Section 3.2.1 through Section 3.2.5, the team produced the following table of ratings on a scale of 1 to 5 with 1 being "not met" and 5 being "thoroughly met":

Table 1: Tool analysis ratings

	Effectiveness	**Efficiency**	**Safety**	**Learnability**	**Memorability**
RTFD	1	4	1	4.5	5
Aurora	4.5	3	4	4.5	4.5
BrainEx	3.5	4.5	5	1.5	2
Matlab_GUI	3	2	2	3.5	4.5
NirsLab	3.5	4	2	2	4.5

The team's main takeaways from analyzing the usability of these tools were the importance of communication, intuitive control flow, user-centered error handling, and clear and concise verbiage. Tools that had a high score in "Memory" were simple and straightforward, with relatively effective control flow. Tools that received a lower score in "Effectiveness" and "Safety" had poor or no error handling and data recovery. Those with low "Learnability" had inaccessible terminology and only partial documentation. Tools with low "Efficiency" were slow to respond and offered little feedback to the user and there was also more room for error due to either the application crashing or the error messages being at the developer level. In conclusion, to meet the goals of each evaluation category and improve upon current applications, the team aims to include user-centered error messages, continuous and useful feedback, a streamlined control flow, accessible language, and thorough documentation when developing their interface.

4. Collection and Analysis of User Requirements

In addition to understanding the current solutions in the problem space, the team wanted to understand what users would need from the system. Therefore, the team began collecting and analyzing user data. Through interviewing potential users, the team aimed to determine the likely end users, their technical experience, and their use of the current set of BCI tools. The team then analyzed both the user-specific and the task-specific data collected to determine the audience for the tool. Then, from the results, they determined the base technical requirements.

4.1 Conducted User Analysis on Lab Staff

The team conducted semi-structured interviews with nine members of the lab staff, with a mix of undergraduate and graduate students, to be able to formulate accurate user personas, or profiles. In addition, we interviewed one of Professor Solovey's former students with more familiarity with fNIRS system. Interviews were recorded with the user's consent (recordings will not persist beyond the length of the project) and we took detailed notes while keeping the user anonymous. In addition to creating user personas, the team analyzed user feedback on existing lab tools received during interviews.

4.1.1 Process

In order to gather requirements about features to implement in BrainEx, the team conducted interviews with three undergraduate students, three graduate/PhD students, and three developers (of varying levels of study) to acquire information about various fNIRS tools present in the HCI Lab. The team also collected information about target users such as their demographics, education level, and familiarity with the tools. In addition, the team analyzed their feedback on the use of current tools and found common areas of advantages, disadvantages, and improvements to apply to BrainEx.

To be able to approach the design of the BrainEx UI from the users' perspective, the team first determined who the target users are. Through consulting with the advisors, the team decided to treat the WPI HCI Lab staff (undergraduate, graduate, and PhD researchers) as target users since they regularly use BCI research tools such as BrainEx.

The team prepared an interview preamble that introduced themselves, described their overall goal, and what they hoped to gain from the interview to provide useful context for the user and a clear agenda for the interview. This document can be found in Appendix A.

The team also produced a set of general interview questions. These questions encompass many aspects of user experience including what tools they currently use and for what purpose,

their user-experience with the tools, and their expertise in using them. Depending on the user's role and domain of expertise in the lab, certain questions were omitted or improvised. Where applicable, the user was asked to demonstrate the tools they mention during the interview. In essence, the team hoped to have the following questions addressed:

- What is the demographic of the lab staff who are potential users of BrainEx in the future?
- What are their needs and wants of using the existing tool/tools?
- Are there any aspects of the tools that they find frustrating?
- What is their typical workflow when using the fNIRS data?
- What is the most useful/useless thing they found on this interface?
- What is their level of understanding of the tool(s)?

A full list of interview questions can be found in Appendix F.

4.1.2 Outcomes

After conducting the interviews, the team determined that the target users in the lab range from undergraduates to PhD students from many different backgrounds. The most common subject areas of expertise within the lab are Computer Science, Electrical & Computer Engineering, and Biomedical Engineering. Undergraduates typically facilitate the data collection process while graduates perform data analysis. Some undergraduate and PhD students are also involved in developing fNIRS-based tools for various phases of the lab workflow. Overall, all the students in the WPI HCI Lab facilitate fNIRS-based research in the lab and have experience with the tools used in the lab to collect, process, and analyze data from fNIRS-based BCI technologies.

In order to effectively communicate and sharpen design focus, the team developed fictional user personas that represent the target user. When the team completed the interviews, the information collected was synthesized into three distinct user persona groups with which to drive the designs: novice users, intermediate users, and advanced users. The novice user group represents the undergraduate researchers who typically facilitate data collection and some preprocessing; the intermediate user group represents the graduate/PhD researchers who usually perform more complicated procedures such as data analysis; and the advanced user group represents the developers of the backend. Developers are considered a stakeholder rather than a target user. While the developers may not necessarily use the application for its designed purpose as target users, their roles require constant interactions with the frontend to maintain the backend. Thus, their perspectives were considered in the design. The personas, as shown in Table 2, include demographics, educational background, and additional character details that helped evaluate user goals as they relate to the application. The interview results for lab staff were first

grouped by education and experience level. Commonly occurring themes (e.g. tool frustrations, what they liked about the tools, etc.) were then consolidated for each group. Lastly, the different perspectives of each of the users were included by prioritizing the most important issues that should be addressed in the development of BrainEx. The final user persona tables to summarize the information from conducting the interviews can be found in Appendix H.

*Table 2***:** *Template for persona information*

Name	Name of the user
Age	Age of the user
Education	Education level of the user
Title & responsibilities	Title and general duties user performed in the WPI HCI Lab
Goals & frustrations	The goals, useful features, and improvements the user mentioned about the tool they are demonstrating
Narrative	Background of the user in regard to their work and expertise with the tool
Quote	Quote the user mentioned about the tool

Through the interviews and user personas, the team decided that they will focus the UI on novice users. If novice users are able to use BrainEx, then it most likely will not be a problem for more experienced users to learn. From the user personas, the team was able to clearly identify the stakeholder group, and the novice, intermediate, and advanced expertise groups. The personas also helped the team determine which group to ask specific questions that may regard guidance, development feasibility, or more advanced options pertaining to the BrainEx UI.

Alongside these user personas, the team also noted common user needs from the interviews. Once the team identified the recurring needs, they consolidated and organized into 4 categories (see Table 3). They also identified the associated functionality of the user-interface they will design, which are explained in more detail in Section 4.2 and Appendix P.

Table 3: User needs for BrainEx tool alongside associated BrainEx functionality

User Need	**Functionality**
Data exploration	Cluster Explorer Dataset Explorer
Investigate data patterns	Query best matches for data sequences
Export database	Save database
Export best matches	Save query results

User Tool Analysis

The team's collection of data from target users included the overall task workflow of the research as well as the user-experience of the aforementioned tools. To add depth to their previous tool analysis, the team examined the interview data to extract the usability aspect of each tool based on the opinions of the lab's users. The usability aspects include the following:

- Effectiveness: How good is a system at doing what it is supposed to do?
- Efficiency: Once users have learned the design, how quickly can they perform the task?
- Safety: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?
- Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?

The tools that were analyzed include: RTFD, Aurora, NirsLab, Homer, Matlab GUI, and BrainEx. The findings were summarized below:

- **Effectiveness:** Overall, these tools achieve promising results and are effective in usability and transparency. However, there are significant improvements that could be added to each of the applications to increase the usability for users.

- **Efficiency:** Overall, the time it takes to run certain functionalities should be fast. Users also commented that visuals are crucial to their understanding of the data. This is especially advantageous for data analysis purposes because users can clearly see

which regions need to be investigated more through color coded lines or markers on the data points. In addition, users noted that there is a need to export data in a csv format. Alternatively, a recurring comment was that the software was not customizable, and a specific procedure had to be followed as it was not open source. Another concern was overcrowding of the UI with features or graphs while at the same time, deviation of the user by having to navigate multiple pages. The UI should be broken up into a reasonable amount of stages for better navigability and usability.

- **Safety:** Users make errors sometimes in the applications, but there is often no way of knowing what was done wrong. For example, if a user clicks on a series of buttons in the wrong order, there is no error message or way of notifying the end user about what is happening. This causes a lot of frustration, having to restart, or spending hours on reading about the feature. In addition, another recurring concern was that there is no proper documentation for the applications. Either the documentation for a tool is poor, not updated, or does not exist. It would be easier to have guidance within the application by hovering over a tooltip to learn more about a particular workflow or functionality.

- **Learnability:** The tools, with the exception of Homer and BrainEx, are easy to learn and are straightforward even when using for the first time. It does not take a long time to get acclimated with the features and functionality as there are clear labels to describe what each button does although there are areas for improvement. Homer and BrainEx have the same concerns in which they are not apt for non-technical people, as Homer requires programming knowledge, and both are difficult to learn for people who are not in the field of study or do not have prior experience or knowledge of fNIRS data procedures.

- **Memorability:** Overall, after using the tools and getting familiar with them, users reported that going back after a brief interval was not a problem for any tools besides Homer. For the other tools, they have to follow similar steps to collect and process data that are easy to learn. For Homer, on the other hand, users lost touch with the programming knowledge needed to process the data.

Analyzing the interview data gave the team a broader context to the usability of the lab's entire workflow. These findings also helped the team understand some common usability issues with current BCI tools from the users' perspective. Therefore, the team was able to sharpen their focus during the design phase to prevent the same issues.

4.2 Gathered system requirements of BrainEx

To understand better how the UI should implement the users' required tasks, the team sought to understand the technical framework and structure of the BrainEx API. The team met several times with the developers to go through how the application works and how to use it in order to extract the system requirements of the UI. The developers provided the team with several resources such as the system architecture diagram, tutorials, and development documentation for a deeper understanding of the API's implemented functionalities. In addition, the team used the BrainEx backend through the command line to cement the user workflow. To review and confirm these concepts, the team documented the commands and created a flowchart to illustrate the high-level overview of the BrainEx system as well as its specific capabilities. Doing so allowed the team to abstract the technical details so they could better focus on identifying all the user interactions within the interface.

Through meeting with the developers of the BrainEx backend, the team had the opportunity to see BrainEx's command line functionality. The developers demonstrated the process of creating a database from a CSV file, loading a database from memory, building the cluster groups, and querying the data. The main features of the BrainEx API include the following:

- Creating and configuring a SparkContext according to a machine's memory and cores for the distributed computing to take place in
- Parsing a CSV file of raw data into the desired data structure to prepare for preprocessing
- Loading a previously preprocessed dataset from a folder location
- Saving the preprocessed data into the correct format so it can be loaded again in the future
- Preprocessing the raw data by grouping similar time sequences into clusters
- Generating a list of query sequences for experimental use
- Querying into the preprocessed dataset with a query sequence and the desired parameters to find similar sequences

There are also several metadata features that can be used for exploring data beyond specific queries:

- Retrieving the total number of sequences in each cluster
- Retrieving the thumbnails of representative sequences
- Retrieving the sequences within a cluster using a given representative sequence

See Appendix P for more detailed information of the functionality of the BrainEx API. Figure 19 shows the general flow of events.

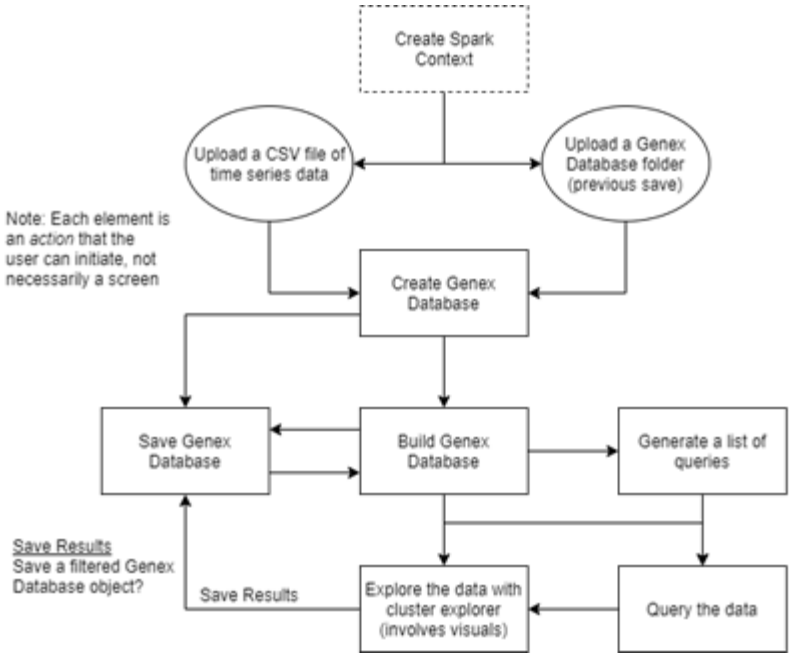


Figure 19: A flowchart explaining actions of the BrainEx application. It is mainly linear with some deviation for multiple options, leading the team to the idea of a prescribed flow for initial screens and tabbing for main screens.

5. Completion of System Design Specifications

After determining user and developer technical requirements, the team created several design specifications to start deciding how to combine the two into one cohesive system. The team developed a series of system models starting from a high-level understanding and gradually adding more technical details to determine the final requirements with respect to users' needs. These models are intended to help the overall design of the interface functionality focus on the end user.

The first model, the conceptual model (Jacob, 2018), describes the features of the application and the relationship to their functionality in terms of objects. Then, the semantic level design describes the functionalities in detail, including specific functions from the current API. Following this, the syntactic level design, also known as a state diagram, was made to establish the different states the UI will take as different functions are executed. Lastly, the lexical level design was used to give concrete definitions of each action executed in the syntactic level. These models are explained in detail in Subsections 5.1 through 5.2. Each model provided different types of insight into the desired organization of the final app that allowed for less confusion when prototyping

To conclude the design phase, the team provided the backend development team with these models to ensure that all design decisions on the user-side are implemented in the command line tool.

Throughout the development of the interface, the team's understanding of the requirements as presented in the models did not change drastically from the time the models were initially created. Therefore, while the models presented in these documents are not the original models created by the team, the changes are minimal enough that just presenting this version of the models accurately portrays the team's specification thought process.

5.1 Created the Conceptual Model

The purpose of the conceptual model is to help the designers understand the actions and objects necessary for users to operate the UI. It is also referred to as the "mental model" because it represents the users' perception of the UI. The deliverable representation of this model includes lists of the objects, operations, and their relationships in the UI. Taking the example of modeling a text editor, objects would include "characters, files, [and] paragraphs," relationships would include "files contain paragraphs contain characters," and operations would include "insert, delete, etc." (Solovey, 2016).

The team created their conceptual model as the first model in their design specification process. They used the research collected about the technical specifications and user needs to pull together a list of features and objects needed in the application.

Table 4 shows the initial conceptual model created by the team. The conceptual model was helpful to the team because it helped organize the team's thoughts on what objects, relationships, and operations from the BrainEx command line tool that the users would want to see implemented on the frontend. It also helped the team realize that while there are few objects, there is much data held within these objects, which will require careful management.

Table 4: The team's conceptual model helped organize the users' desired functionality in one place for reference during prototyping

Objects	**Relationships**	**Operations**
● SparkContext ● Dataset (Dataframe structure) ● Query object ● Query results	● The other objects reside within the SparkContext ● Query results are subsets of dataset ● Query result is generated from query object	● Load database ● Build database (this does the clustering) w/ build specifications (length of interest, type of distance, similarity threshold) ● Save database ● Generate query ● Query database ● Plot query results ● Explore clusters ● Explore dataset

5.2 Created the Semantic Model

The semantic model expands upon the conceptual model to unify the technical abilities of the backend API and user-specified functionalities. For each action, the team specified a function definition, the necessary parameters, any output/feedback, errors that could occur, and how those errors would be handled. This information is organized in table form. Each function is represented by its own table.

The semantic model created by the team is comprised of Tables 5 through 12. Each table represents a different function that users wish to see implemented in the frontend, as determined by the conceptual model. The creation of the semantic model was essential to the team's understanding of how the users should be able to interact with each backend function as they used the interface. Each table of the model ensures that the team has a plan for handling all aspects of each function before entering prototyping.

Table 5: Semantic model of loading the preprocessed dataset

Function	Load database
Parameters (implicit and explicit)	Either an existing DB file created in another session or a CSV file containing data, need a Spark Context before running
Description	Loads data into API for user manipulation
Feedback	Loading bar while file uploads if it is able to upload properly, then transition to next screen where user can select build options. If unable to upload for any reason, should produce an alert and stay on file selection screen.
Error Conditions	• Give error message if file does not exist and stay on file selection screen • Give error message if file is formatted incorrectly and return to file selection screen from loading bar

Table 6: Semantic model of preprocessing the raw dataset

Function	Build database
Parameters (implicit and explicit)	There must be a Spark Context and loaded dataset before running. Explicit parameters: Database/self, Similarity threshold, length of interest, distance type, verbosity (may be obscured from user)
Description	Groups and clusters sections of data of length specified by length of interest slice
Feedback	Loading bar while database builds if it is able to build properly, then transition to next screen where user can select querying and viewing options. If unable to build for any reason, should display an alert and stay on build screen
Error Conditions	• Limit user input to ensure length of interest valid (i.e. upper bound larger than lower bound and length found in data) • Limit user input to ensure similarity threshold is between 0 and 1 • Limit user input to ensure distance type valid

Table 7: Semantic model of saving the preprocessed dataset

Function	Save database
Parameters (implicit and explicit)	There must be a Spark Context and loaded dataset before running.
Description	Saves a database to the local folder to be able to reload later
Feedback	Alert to let user know if saved properly or unable to save
Error Conditions	• If unable to save, alert user

Table 8: Semantic model of generating a list of query sequences

Function	Generate query
Parameters (implicit and explicit)	CSV file listing desired queries (each query includes features and start/end lengths; will probably be abstracted by user by having user select a sequence and generating the CSV from that), number of features in said queries; must have Spark Context
Description	Generates an object that holds information about a desired query
Feedback	Alert when query has been generated or if it can not be generated
Error Conditions	• Give error message if CSV is invalid • Give error message if number of features does not match CSV

Table 9: Semantic model of finding similar sequences

Function	Query database
Parameters (implicit and explicit)	(built) Database/self, query object, number of best matches, boolean of whether to exclude representative sequence, overlap threshold; Spark Context must exist
Description	Generates an object that holds results for best matches based on a given query object
Feedback	Display query results if query can be computed, give error message if not
Error Conditions	• Give error message if query object is invalid • Limit user input to ensure best matches is a positive whole number • Limit user input to ensure overlap threshold is a number between 0 and 1

Table 10: Semantic model of plotting similar sequence results

Function	Plot query results
Parameters (implicit and explicit)	Query results, Spark Context must exist
Description	Displays desired data in a line graph
Feedback	Display line graph of given data
Error Conditions	• Give error message if query results does not exist

Table 11: Semantic model of exploring clusters

Function	Cluster explorer
Parameters (implicit and explicit)	(built) Database, Spark Context must exist
Description	Displays clusters in line graphs for exploration
Feedback	Display line graphs and statistics of clusters, allowing users to look through them
Error Conditions	• Give error message if database does not exist

Table 12: Semantic model of exploring the entire dataset

Function	Database explorer
Parameters (implicit and explicit)	(built) Database, Spark Context must exist
Description	Displays sequences in line graphs and provides statistics for exploration
Feedback	Display line graphs of sequences, allowing users to look through them
Error Conditions	• Give error message if database does not exist

5.3 Created the Syntactic Model

This model explains the processes the users will follow while utilizing the application. This is presented in the form of a state diagram. The model represents each state that the application may have (Solovey, 2016). These states are distinct representations of the data available to the user based on the operations they have completed. Each state is represented as a circle, bubble, square, etc. The connections between each state, represented as lines, are the operations that users may complete. The lines are labeled with the user action and system response. A one-sided line represents entering or leaving a state from outside the application. The team created a syntactic model for the entire system.

Figure 20 represents the syntactic model created by the team. This diagram helped get the team starting to think about what different states or screens should exist in their interface. The

team noticed that there are fewer states than they expected to see based on their earlier ideas for interfaces.

Figure 20: The team's semantic model provides an order to the functions previously outlined and shows the states of the interface.

5.4 Created the Lexical Model

Finally, the team completed a lexical model based on the syntactic model. The purpose of the lexical model is to define each action from the syntactic model. This helped the team understand how the end-user will be completing each of these actions. The level of granularity for the lexical model should detail every step a user takes to complete the action (Solovey, 2016). For example, a lexical model of shutting down a Windows computer would be to click the Windows button, then click the power options button, then click the shutdown button.

The lexical model (Table 13) provided a way for the team to envision how the users will interact with the application. In creating their lexical model, the team was able to determine what interaction concepts and mechanisms the users would be able to use to make the frontend easy to follow and effective at its purpose.

Table 13: The lexical model for the application allows for an understanding of the actions in the application and how they are to be accomplished

Action	**Definition**
Window_Open	Click BrainEx icon
Load_File	Browse to desired database or CSV file → click load
Local_Copy_Database	Click save database
Build_Database	Enter similarity threshold on slider, enter length of interest in 2 textboxes, select distance type from dropdown menu → click build
Query_Database	Navigate to query tab → Enter number of best matches in textbox, check box of whether to exclude representative, Enter overlap threshold on slider → click query

6. Design of Prototypes Using Iterative Design Strategy

After completing models of the system and confirming their understanding of the application, the team completed several stages of prototyping, from ideation to interaction. Each stage was iterated upon until the objective for that stage was achieved. The team justified each design choice in an informal report, which was written along with each UI prototype. After each iteration, the team planned their next iteration by revisiting the result of their evaluation and listing the changes needed on the current system. The team noted features that were working well for the user as well as features that were not noticeable or needed more improvement. If there are multiple viable ways to design a section of the prototype, the team compared the trade-offs to each version of the screen and decided on which version to adopt by considering usability, desirability, and usefulness. Table 14 below shows the team's timeline for this process.

Table 14: A timeline of the 3 stages of design including purpose, goal, and evaluation.

Design Phase	**Purpose**	**Design Goals**	**Evaluation Method(s)**	**Estimated Time**
Iteration 1 - Ideation	Confirm fundamental concepts of system features and user needs	Storyboards	Self-evaluate using theoretical models (part of design specification), verify correctness and confirm user needs	1 week
Iteration 2 - Utility Refinement	Ground the basic prototype design structure	Low-fidelity prototype on Balsamiq (wireframes)	Heuristic evaluation and user testing	1 week
Iteration 3 - Usability Refinement	Solidify control flow and structure of system through rapid prototyping	Mid-fidelity prototype on Balsamiq (more interactive wireframes)	Heuristic evaluation, user testing and post user-testing questionnaire	1.5 weeks
Final Iteration - User Experience Refinement	Determine all visual elements of user interface to enhance usability	High-fidelity prototype on Invision (fully designed prototype)	Heuristic evaluation, user testing and post user-testing questionnaire	2 weeks

6.1 Iteration 1 - Initial Design Ideation with Storyboards

The first iteration was intended to help the team collect more input on functionality, understand the design concepts, and familiarize themselves with the fundamental structure of the application. The team used storyboards for the initial prototype because they allowed design concepts to be reviewed and validated by experts before receiving feedback from target users within the lab. By the end of this iteration, the team expected to be able to answer the following question: does the current plan for the application meet user needs and does the interface accurately reflect the system behind it?

6.1.1 Process

Design of Storyboards

The team first brainstormed all features that needed to be addressed in the UI on a whiteboard (see Figure 21) to solidify their understanding of user needs. Then, they created a storyboard for each individual feature of the envisioned system (e.g. uploading a file, zooming in on a graph, etc.) at a time to concentrate only on the essential interactions. Each storyboard featured text, captions, and users with only the essential details. Depending on the complexity of the feature, the team presented multiple versions to allow users to provide useful feedback on which they preferred and why.

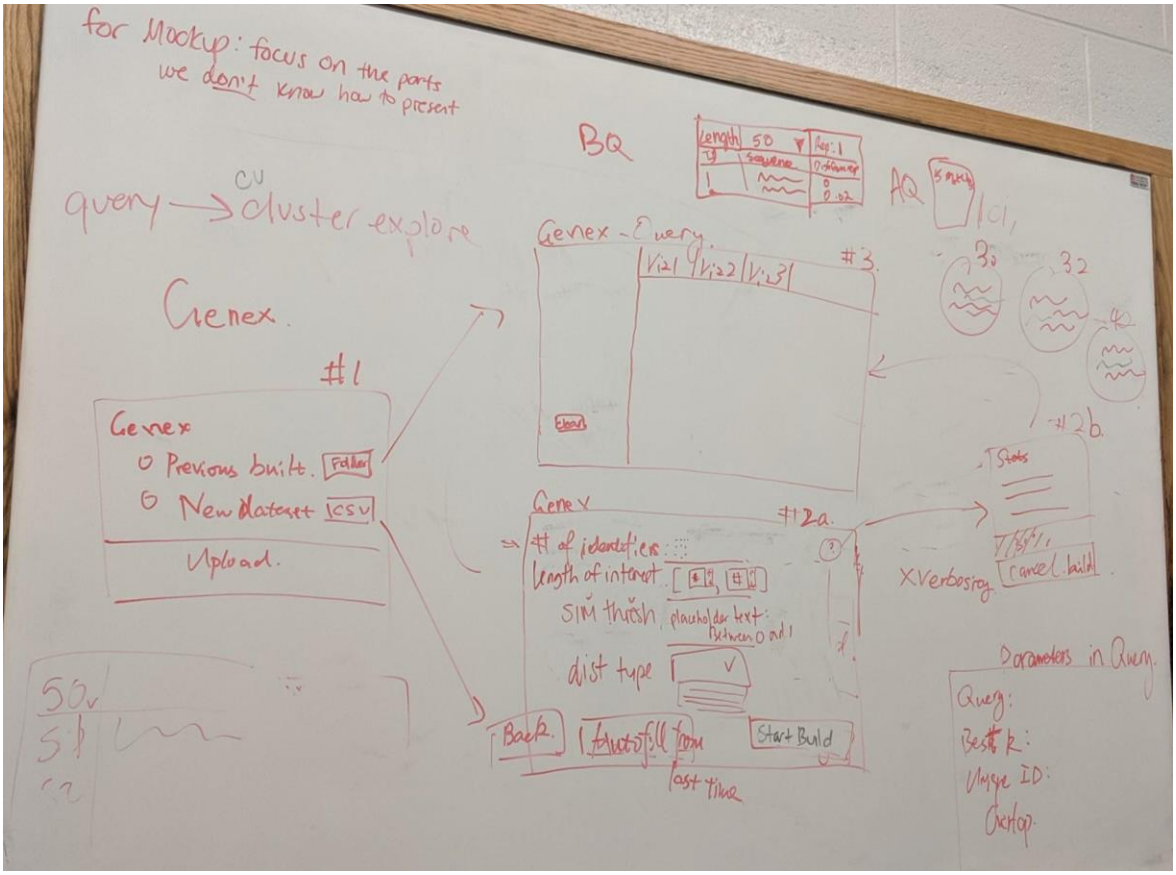


Figure 21: Brainstorming for the Storyboards

Evaluation of Storyboards

Once the team designed the storyboards, the correctness of each storyboard was verified through self-evaluation using the models previously developed in Chapter 5. The team first evaluated the storyboards themselves and ensured that they abide by the previously determined

design specifications. The BrainEx developers, who have more expertise in the domain, further evaluated the storyboards. The resulting storyboards can be found in Appendix M.

The team then collected feedback directly from target users by presenting potential design features to them in the form of storyboards. Storyboards were presented to the lab staff, and the team noted strong reactions to each one. For each storyboard, the team also asked open-ended questions to allow room for discussion. Exemplary questions included: what do you like about this feature? Do you find it useful? See Appendix B for a full evaluation protocol. The feedback received from users was incorporated into the storyboards, the result of which can be found in Appendix N. Next, the team presented the storyboards to the professors to receive critical feedback and eliminate any misconceptions about the system. Any significant changes to the team's understanding of the system (e.g. additional features, control flow within an individual component, etc.) were applied to the storyboards after gathering the feedback from the user testing sessions and advisors. Lastly, the team consolidated and analyzed the users' responses in order to incorporate any previously unconsidered user needs into the final storyboard design.

6.1.2 Outcomes

Evaluations

In this iteration, the users' feedback was instrumental in identifying the places where the placement of tasks both in the application needed improvement in order for users to be able to use the application effectively. Users' feedback showed there could be clearer language and a more simplified presentation of functionalities. Users also expressed a desire to know more about the shape and interaction with the data, such as where it was coming from and how exactly they could and should interact with it. Given the feedback, it became clear that clarity within the application in terms of control flow and nomenclature would resolve most of the issues. Users also expressed confusion in the representation of the data attributes in the storyboard as well as how users would access data in the application. Therefore, the team decided that these topics gathered from the testing the storyboards could become the focus of the low-fidelity prototype.

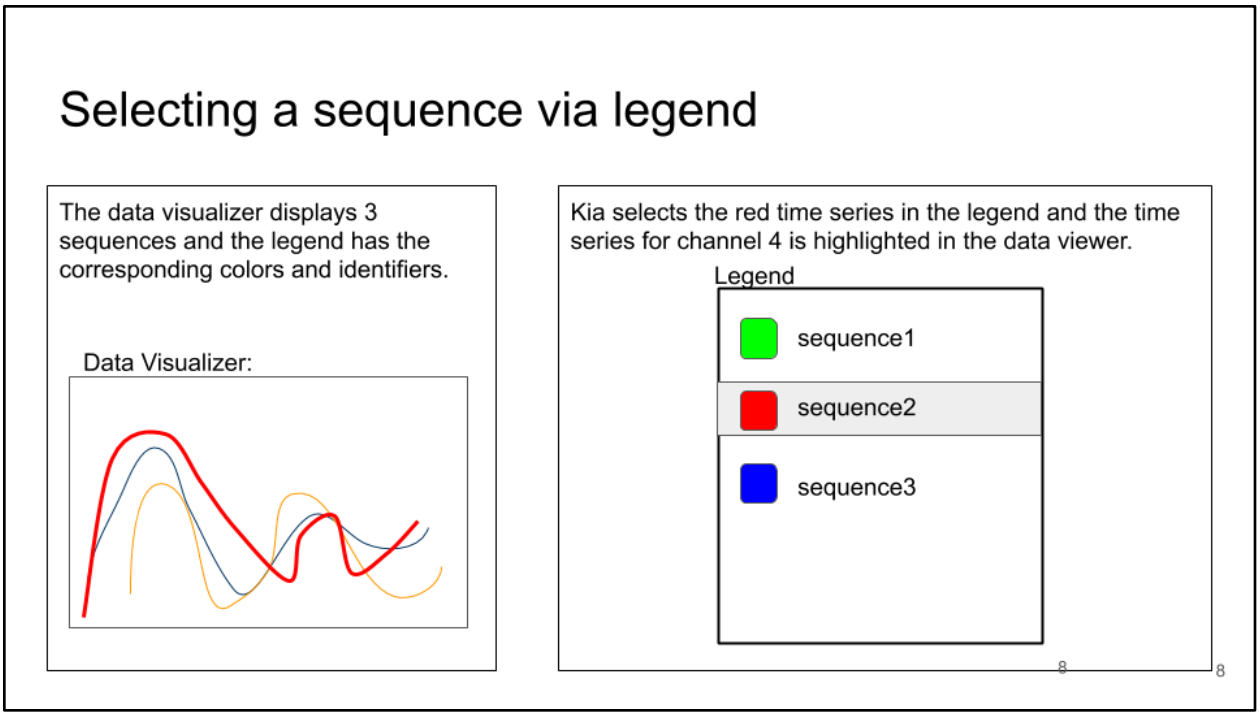


Figure 22: An example storyboard highlighting how to select a sequence using the Legend

Based on this round of feedback, the team discovered that the storyboards should be more compartmentalized and independent of each other, showing just a single functionality. The users mentioned that they were confused about the representation of the data as well as where it came from, such as from a file or user input. This feedback was then applied to refine the storyboards and solidify the team's understanding, and more details can be found in Appendix O. An example of one of the final Storyboards is shown in Figure 22. These storyboards became useful in Iteration 2 as a preliminary prototype on which to base the Balsamiq prototype.

Conclusion

Storyboards served as a useful resource for ensuring that all features and their individual components were incorporated into the following iterations. The feedback from users helped the team identify flaws with their testing methods, such as not presenting the goal of the application before beginning to ask questions. This helped the team fix wording for study methods in future iterations.

6.2 Iteration 2 – Utility refinement with low-fidelity prototype

With a clear plan for the individual components and functionality of the BrainEx system, the team began designing the base prototype of the application, keeping in mind the general control flow but focusing on the flow of each individual component. The low-fidelity prototype

was designed to establish the basic structure of the prototype screens based on the essential features that were highlighted from user feedback on the storyboards. The primary goal of this iteration was to answer the following question: Does this system solve user problems and meet user needs?

6.2.1 Process

User Task Analysis

After creating all the related models and storyboards, the team was able to analyze what users should be able to do with BrainEx. This step allowed the team to break down the workflow within BrainEx and decompose the core user task into a hierarchy of ordered subtasks, thus helping the team to brainstorm how the overall structure should be designed in the initial prototype.

Essentially, a typical user task would be to find similar sequences to the sequence(s) of their interest. In order to complete this task, users are expected to know whether they have a sequence that is ready for similarity search or not, whether they have a preprocessed dataset, and if not, what the preprocessing parameters will be. Preprocessing in this application shortens the amount of time spent looking for similarities as it pre-arranges similar sequences into clusters. The starting point of the task is marked by the selection of a dataset, either preprocessed or raw, for data analysis. Once they start preprocessing the dataset, users may know the status of the system by checking the progress of preprocessing. Users are also required to know the parameters input for preprocessing as well as similarity search. The end of preprocessing is marked by the system's capability to find similar sequences to the given sequence. Core components that can help users understand their tasks are visualizations and tables which can give a closer view of specific time series data. The completion of this task is indicated by the result of similarity search of the given sequence.

In summary, the user interface of BrainEx needs to fulfill two main tasks of the user: exploring data and finding similar subsequences in the given dataset. The bare minimum operation requirement follows: importing a preprocessed dataset → uploading a subsequence as the baseline → find similar subsequences in the given dataset. A more complicated task would require users to first preprocess a raw dataset, then explore data before being able to locate a subsequence of interest; and lastly, the user can find similar subsequences. Because the preprocessing stage alone can take up to a day, it is important for the system to display progress to the users so users can consider their own time constraints while working on the task. Thus, a diagram of user task hierarchy which illustrates the major steps was created to summarize the team's findings and highlight the key operations within the system (see Figure 23).

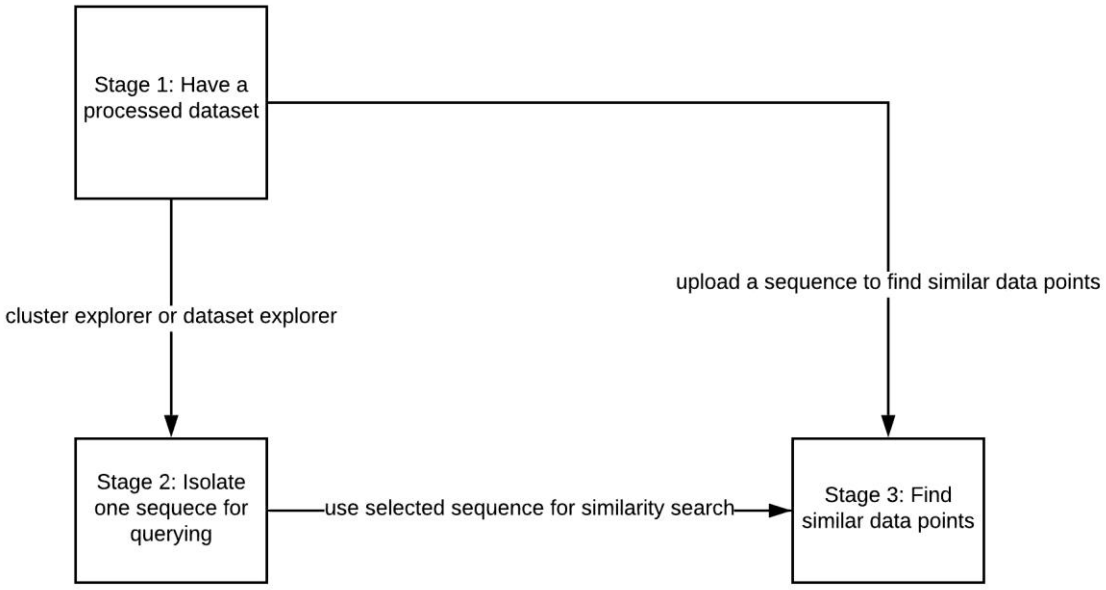


Figure 23: Diagram of User Task Hierarchy

Design of The Low-fidelity Prototype

After the team was able to ground their understanding on the user tasks, they started to brainstorm ideas on how to design the first interactive prototype. To minimize time and resources invested in developing the application, the team started with creating a low-fidelity prototype to communicate their design ideas to the target users and collect any user requirements that might not have come up in the interviews. The user-task related analysis above helped the team better form the structure of the prototype design.

First, the team brainstormed together and created sketches on whiteboards (see Figures 24 and 25) as the initial design prototype. The team focused on addressing the user task questions above as the priority in this design phase. Then the team transferred the sketches onto Balsamiq where they designed a wireframe with basic interactions for each feature. The team chose Balsamiq instead of other prototyping tools because it resembles paper sketches and is reliable, in the sense that it has consistently styled UIs. It is also flexible, allowing the team to make changes quickly using widgets and work on the project collaboratively (“Balsamiq”, n.d).

Figure 24: Whiteboard sketch of the Basic Cluster Explorer Page from the low-fidelity prototype

Figure 25: Whiteboard sketch of a more detailed Cluster Explorer Page from the low-fidelity prototype

The main priority in the design of the low fidelity prototype was the layout of each of the planned screens in the application. Based on the feedback received on storyboards and the understanding of the application gained through modeling, the team grouped similar actions together on separate screens. The main actions the team discovered were loading data, clustering data, exploring data (entire dataset or clusters), and querying. The structure of the prototype reflected the importance of these three actions.

In this prototype, the user must load a dataset, then cluster the dataset. After these prescribed actions, the user has more freedom to explore the entire dataset (see Figure 26) or explore the data per cluster. They may then select a sequence in one of the explorers or using an uploaded file to query for best matches (see Figure 27). Since the team needed to rapidly

prototype, Cluster Explorer was not the focus in the low-fidelity prototype; rather, the team spent most of their efforts on the design of the Dataset Explorer and Query Finder pages.

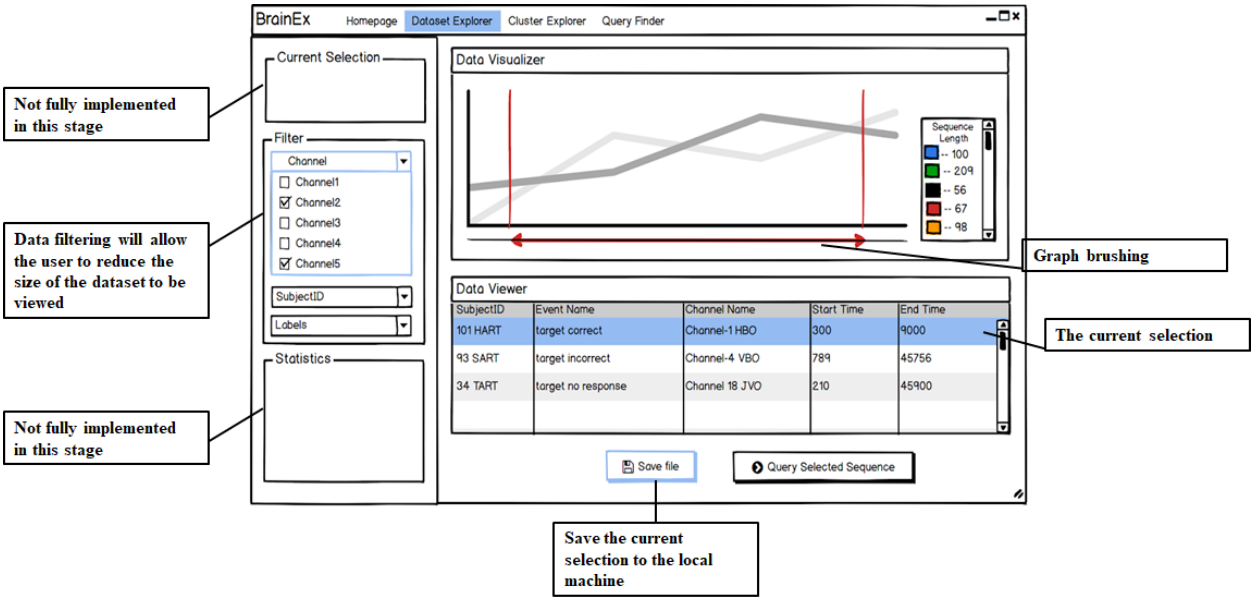


Figure 26: Dataset Explorer page from the low-fidelity prototype

Figure 27: Query Finder page from the low-fidelity prototype

Heuristic Evaluation

Before the prototype was presented in front of users, it was analyzed for compliance with guidelines. Heuristic evaluation can easily expose problems before any user testing without the need for identifying any specific tasks or activities. This also provided a shared language for any better solutions to be proposed.

Regarding guidelines, the team adopted applicable items from Nielson's heuristics model (Nielson, 1994), which encompasses the following aspects:

1. Visibility of system status: Is there appropriate feedback of what is going on in the system?
2. Match between system and the real world: Are any real-world metaphors and/or analogs used? If so, do they match how the real-world objects interact?
3. User control and freedom: Are users able to choose several/many paths of interacting? Are there exits for mistaken choices?
4. Error prevention: Does the interface attempt to minimize possible user errors?
5. Recognition rather than recall: Does the system fill in known info when possible?
6. Flexibility and efficiency of use: Are there special shortcuts that experts can use for efficiency? Can users record/tailor actions to suit their needs? (advanced)
7. Help users recognize, diagnose and recover from errors: Are error messages clear to user? Do they suggest solutions?
8. Help and documentation: Are there clear and concise labels? When needed, is help available?

Using Nielsen's ten principles for heuristic evaluations, each team member individually rated the low-fidelity prototype on eight different aspects (two of which were not applicable to the low-fidelity design).

User Testing

After the team performed the heuristic evaluations, the prototypes went through user testing. User testing served as an empirical evaluation method that allowed the team to observe the user interacting with the UI prototype. The team recruited the lab staff who were previously interviewed as test subjects since they are the target users. Testing was performed by giving users a specific task and asking them to execute it on the UI prototype without giving specific directions. The team asked the users to discuss everything they were thinking from the time the users saw the statement of the task to completion (i.e. "thinking aloud") and video recorded the screens of the user plus their voice, for reference later. The videos were not kept permanently. The team observed this process and noted any critical incidents that point to the user's success or

failure with respect to their tasks. These critical incidents include but are not limited to (Solovey, 2019):

- User does not succeed in achieving the goal within 5 minutes
- User tries several operations or the same operation over again, and then explicitly gives up
- User attempts to find three or more alternatives in order to achieve the goal
- User achieves the goal using a suboptimal approach that is not within the team's intention
- User expresses hesitation or other negative affect

The team's observations were documented in the format of the following Usability Aspect Report (UAR):

Table 15: Usability Aspect Report template (Solovey, 2019)

SubjectID	A code to anonymize the participant
Name	Succinct description of the incident
Evidence	Facts
Explanation	Interpretation of the evidence
Severity	Rating on a 1-4 scale where 4 means catastrophic and 0 means not a problem. Justify the severity with the following parameters: frequency (of the occurrence), impact, and persistence (whether it is an one-time error).
Solution (optional)	Possible fixes and tradeoffs
Relationship (optional)	Link to related reports

After organizing the user testing information in the UARs to capture incidents based on user feedback, the team analyzed the information by grouping similar incidents together and prioritizing the ones that had high severity ratings.

Users were asked a set of questions when the test was complete to solicit direct feedback. Questions were revised to reflect the result of heuristic evaluations so that user responses could be used later to address or help prioritize the concerns raised from the heuristic results. Below are examples of questions the team asked:

- Is the task confusing or too complicated?
- What are the system features that they feel are not useful and ones that are extremely useful?

Through observing the users who will navigate and interact with the system, the team hoped to discover details that might have been left out in their own perspectives.

6.2.2 Outcomes

By conducting both heuristic evaluations and user testing, the team was able to keep the design centered on their utility goals.

Heuristic Evaluation

Although done independently, team members gave relatively consistent ratings to each category of the heuristic evaluation overall. The team concluded that they should prioritize the categories of "help and documentation", "error prevention", as well as "error recovery", in descending order of prioritization. This means, in the next iteration, the team should aim for more intuitive label names in the design, as well as means to handle user errors. The team also would apply fixes to small issues that were identified during user testing. An aggregated table of ratings is included below.

Table 16: Aggregated results of heuristic evaluations for the low-fidelity prototype from iteration 2

Category	**Average Rating out of 5 (based on four team members' input)**
Visibility of system status	3.25
Match between system and the real world	3.5
User control and freedom	5
Error prevention	2
Recognition rather than recall	3.125
Flexibility and efficiency of use	3
Help users recognize, diagnose and recover from errors (Error recovery)	0.25
Help and documentation	1.25

User Testing

User tests were conducted with two target users who were more experienced with brain data analysis so the team could have a better understanding of whether the prototype would meet their needs. The complete user testing protocol can be found in Appendix C. Users were able to successfully complete most of the assigned user tasks but they had many troubles arriving at the solution. The average task completion time was more than five minutes, given that testing subjects were not given time to play around the prototype for enough time before user tasks were assigned. Researchers also had to step in multiple times to give more instructions on how to proceed because testing subjects kept experiencing critical incidents, which are documented in more detail in the Usability Aspect Reports in Appendix I. Overall, the team was recommended to develop a more intuitive and effective control flow for the prototype and offer more guidance within the application. The team also needed to fix the nomenclature inconsistencies to reduce any confusion. Current user testing protocol needed to be revised so users could have more time to get familiar with the prototype on their own before user tasks were assigned. In future user tests, giving instructions to the testing subjects should be avoided as much as possible. The team needed to prioritize the development of a more intuitive and effective control flow for the mid-fidelity prototype. The team also actively looked for better naming conventions to make the application more intuitive.

Conclusion

To make meaningful decisions to prioritize features to be tested in the next prototype, the team evaluated the importance of the design aspects brought up from both the heuristic evaluations and user testing. Their findings from these evaluations suggest that common design aspects the team should focus on included error handling, labels and guidance within the system, and most importantly, a more intuitive control flow. Therefore, it was the team's priority to develop a control flow that requires a minimum learning curve for novice users. Nonetheless, the goal of this design phase was met as the team was able to confirm that the system design was meeting user needs from user testing.

6.3 Iteration 3 - Usability refinement with mid-fidelity prototype

Once the team had the general structure of the application solidified, they focused on the overall control flow, refining the prototype to be intuitive and easier to understand. The primary goal of this iteration was to answer the following question: Can the user understand/navigate through the system without much external guidance?

6.3.1 Process

Design of The Mid-fidelity Prototype

As the design matured, the team moved towards linking all wireframe windows containing the various functionalities of the application together to ensure a smooth user-interaction control flow. The team started to develop more mid-fidelity prototypes that allowed more interactions and highlight key features. They then wrote up design justifications for each page in the prototype and created a sitemap (see Figure 28) that provided additional information about screens and their relationships to support the structure of the control flow.

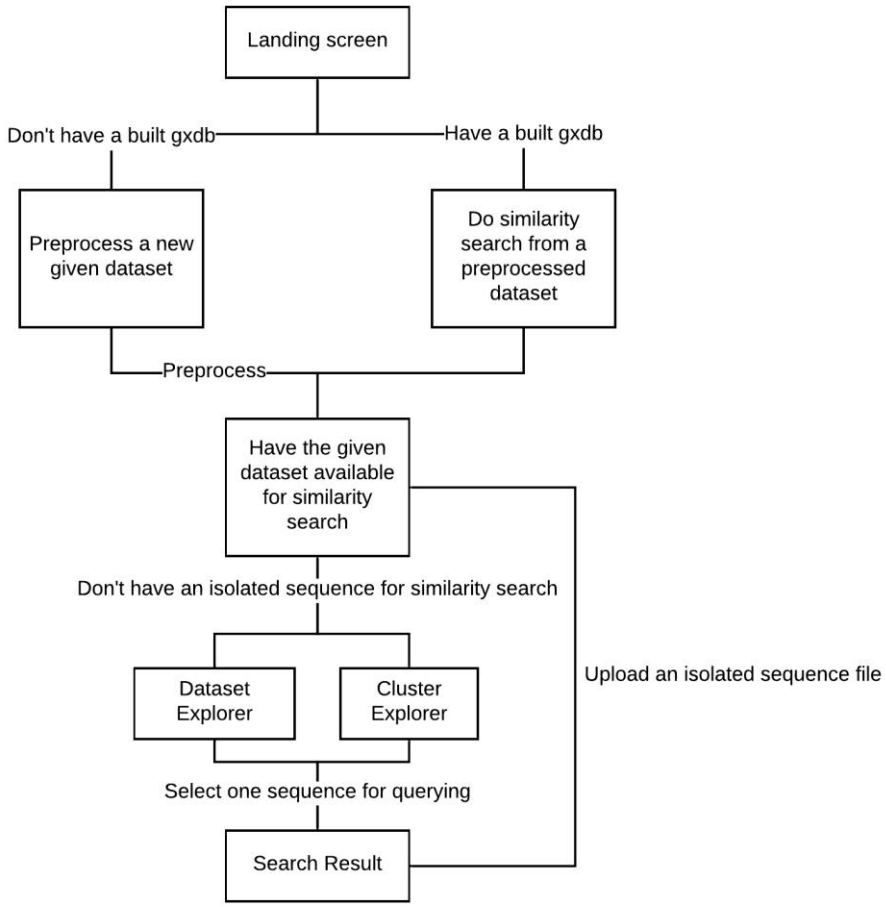


Figure 28: Site map of the mid-fidelity prototype

Following the user feedback from the low-fidelity prototype, the team decided to focus on the control flow of the application. Therefore, this was the focus of the team's mid-fidelity prototype design as the team used whiteboards to brainstorm more ideas (see Figures 29 and 30).

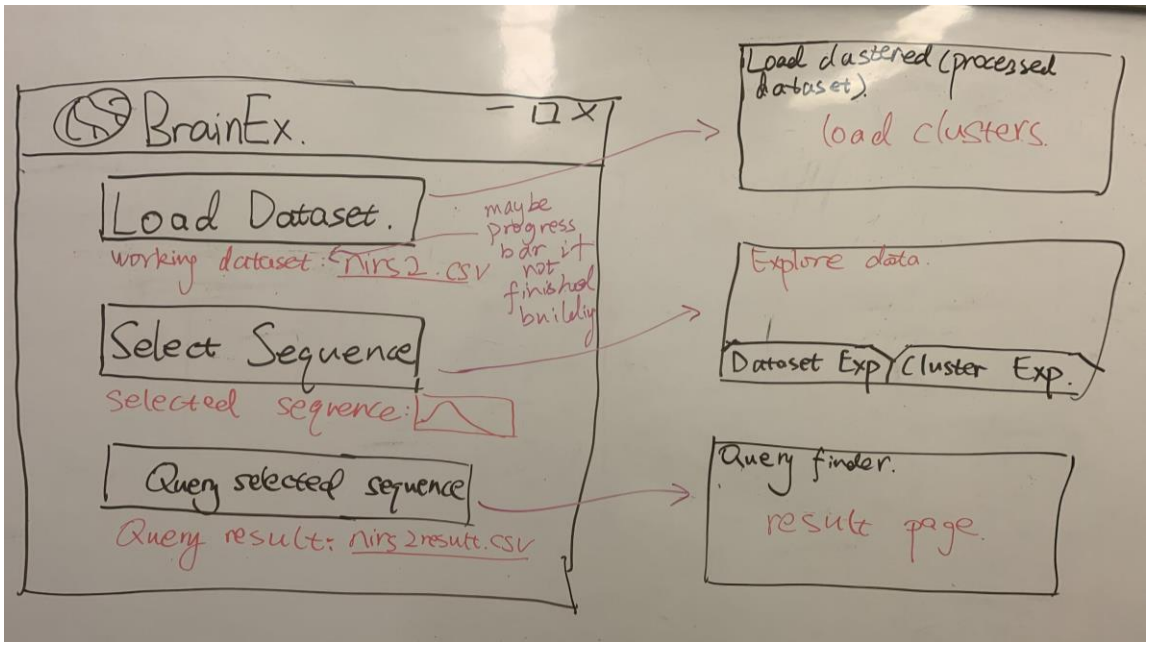


Figure 29: Whiteboard Sketch of the Homepage from Mid-fidelity Prototype Ver.2

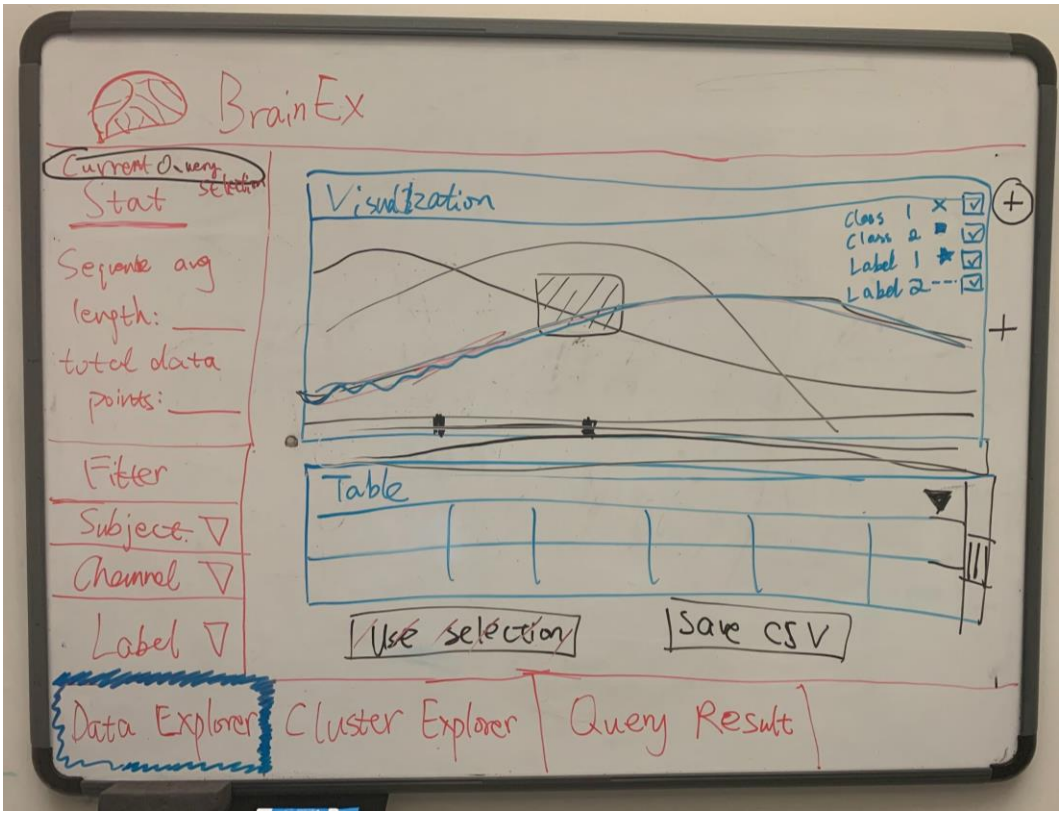


Figure 30: Data Explorer Page from Whiteboard Sketch of the Mid-fidelity Prototype Ver.1

In order to determine the most user-friendly control flow for the application, the team designed two different prototypes with different flows. One included the same flow as the low-fidelity prototype with specific updates based on feedback. This version of the mid-fidelity prototype begins with loading in the data and then leads to clustering the data. If the data does not need to be clustered or when data finishes clustering, the application moves to the preprocessing completion page shown in Figure 31. From here, the user can select the tab they would like to start on within the application, with the tabbed structure mirroring that of the low-fidelity prototype.

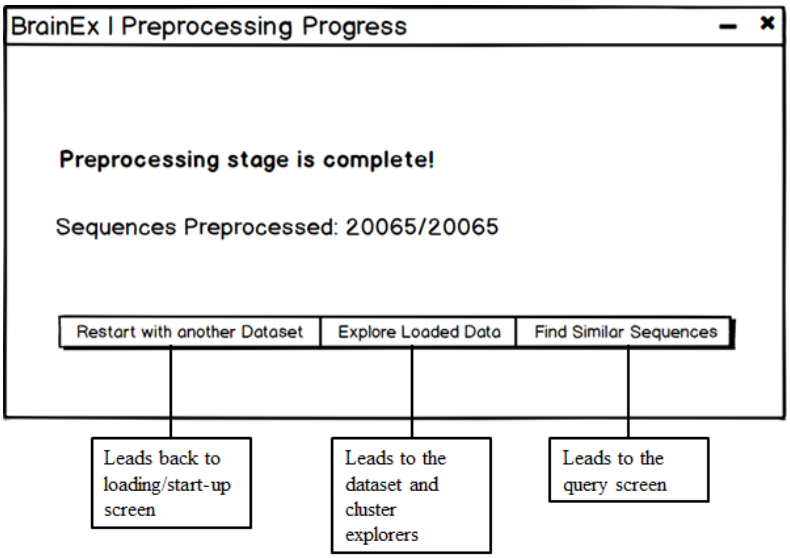


Figure 31: Preprocessing Completion Page from the Mid-Fidelity Prototype Ver.1

The other prototype included a central menu page instead of tabbed navigation (see Figure 32). This was menu page also replaced the data load landing page of the low-fidelity prototype. This page allowed the user to break up the tasks in their work as it required them to go back to the menu whenever they are switching tasks. While this may bring some clarity to the user, it doesn't allow for smooth transitions between tasks.

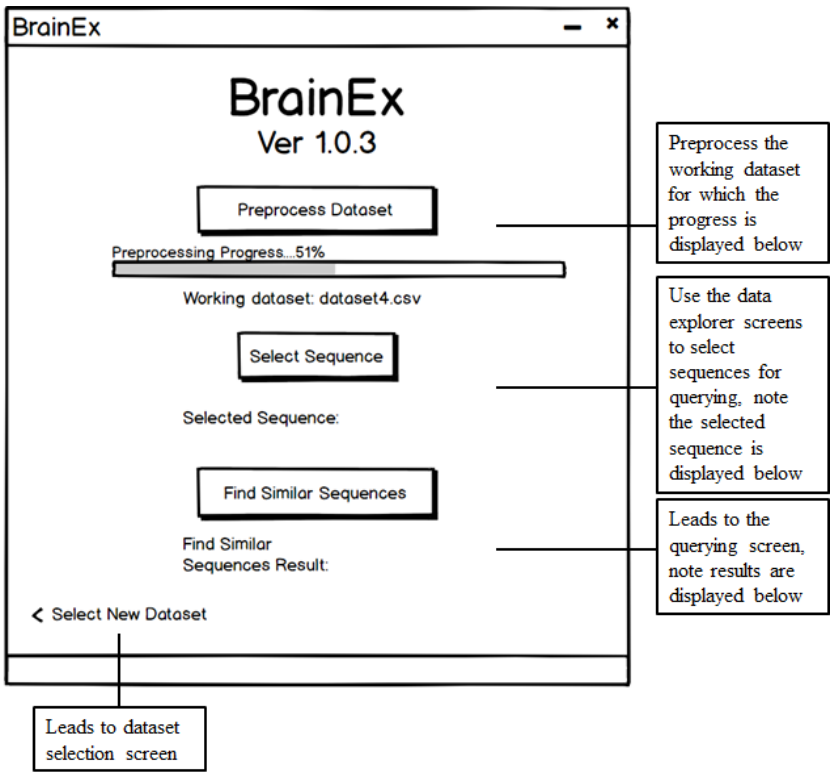


Figure 32: Homepage from the Mid-Fidelity Prototype Ver.2

After gaining feedback from two users and completing heuristic evaluations, the team combined the favored features of the two prototypes into one prototype that includes a central menu, but includes tabs between the two styles of exploration to enable easier switching between the two. The data exploration page, shown in Figure 33, was also improved by the implementation of features such as a back button and statistics, as well as clarified language.

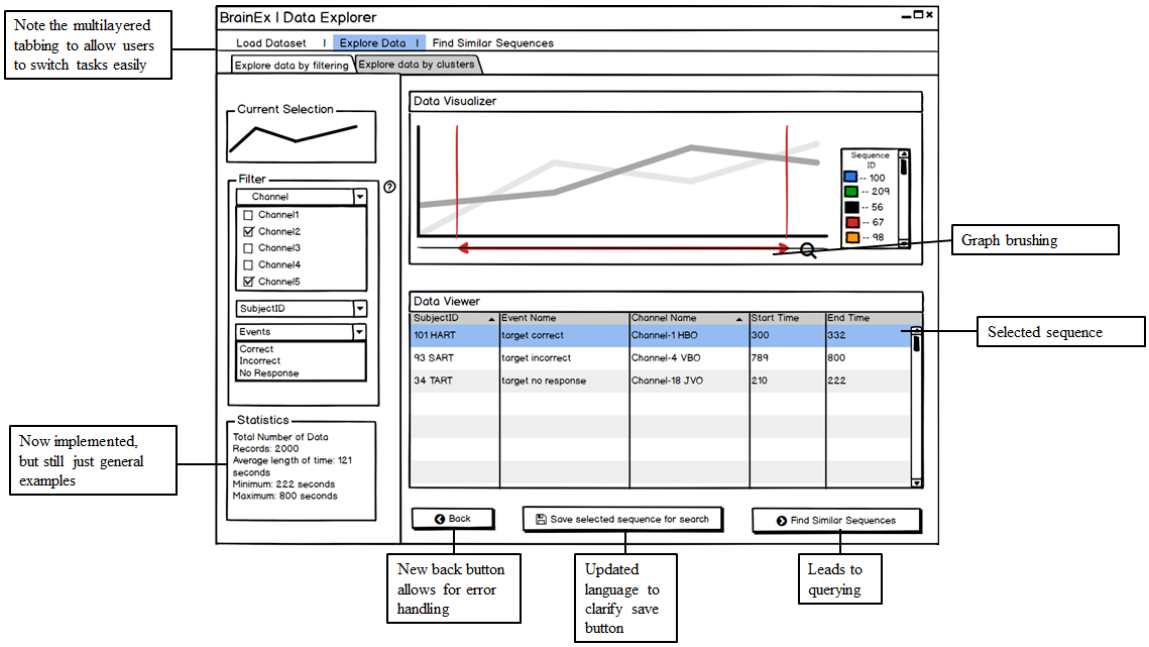


Figure 33: Explore data by filtering page from finalized mid-fidelity prototype

The team was able to spend more efforts on the clustered data page (see Figure 34) in the mid-fidelity prototype. A smooth transition between cluster selection through representative sequences and exploration of the sequences within the clusters was implemented.

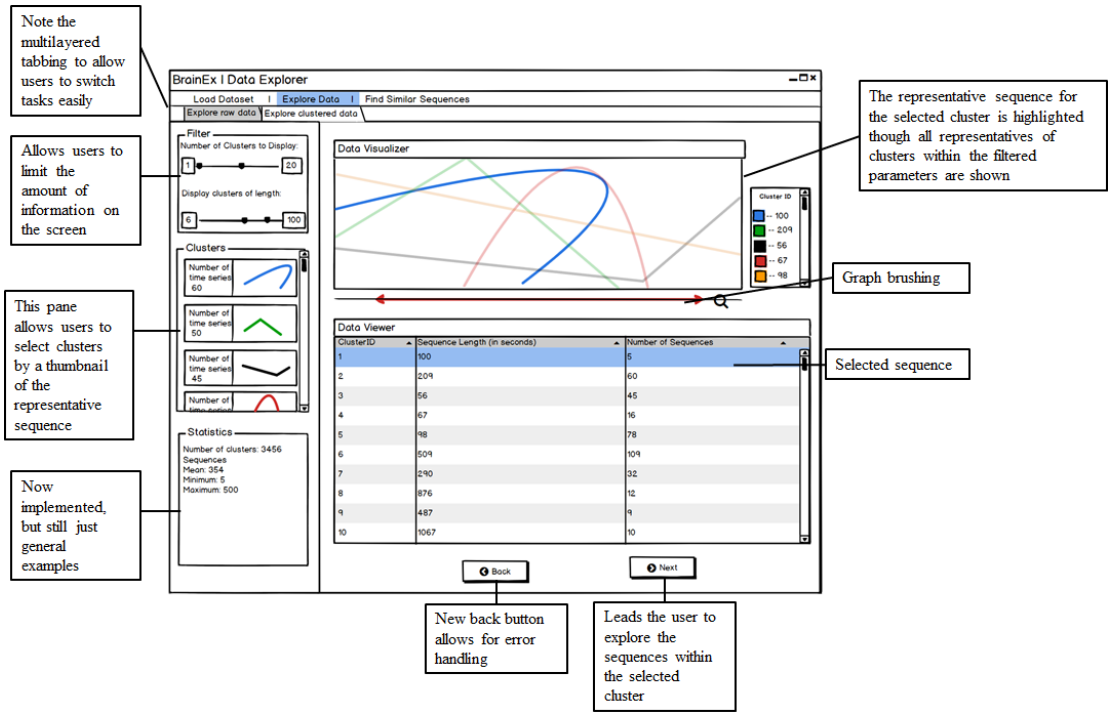


Figure 34: Explore clustered data page from finalized mid-fidelity prototype

Finally, for the mid-fidelity prototype, the query page also benefited from clarifying language (see Figure 35). Also, the accordion style menu was replaced with a numbered menu as users suggested this would be a welcomed clarification.

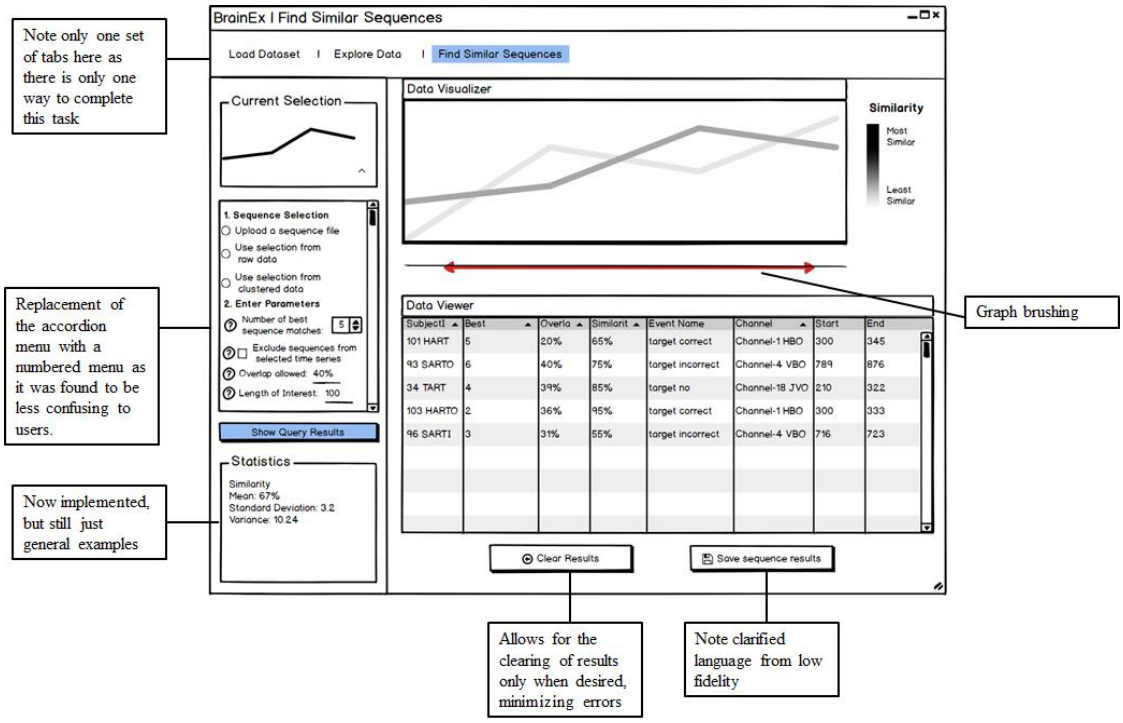


Figure 35: Find similar sequences page from finalized mid-fidelity prototype

Heuristic Evaluation

The team conducted heuristic evaluations after the prototype design on Balsamiq was completed. Besides the original set of questions laid out in Section 6.2.2, two more questions that were not applicable to the previous low-fidelity prototype were added to the heuristic evaluation in this iteration to be evaluated (Nielson, 1994):

1. Consistency and standards: Keep a consistent look and feel throughout the interface
2. Aesthetic and minimalist design: Maintain an aesthetic and minimalist layout without unneeded baggage (not applicable to low-fidelity prototype)

User Testing

Following the heuristic evaluation, the team conducted experiments by showing the user testing subjects two versions of the mid-fidelity prototype, with only the necessary difference in screens related to the control flow to gather rapid feedback. This step ensured that the control

flow could be settled as soon as possible in the early stage of the design before more details are added. Then, the team merged the best control flow features of the two designs and completed more user testing to ensure that the ultimate control flow was successful.

The team also sent out post-testing satisfaction questionnaires in the form of a system usability scale (SUS) to user testing subjects to assess how the mid-fidelity prototypes were perceived. The list of subjective questions from the questionnaire included:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

User responses were on a 1-5 scale with 1 being strongly disagree and 5 being strongly agree. The results were recorded and further analyzed as quantitative data to help the team better grasp their progress.

After user testing was conducted, four out of five participants voluntarily filled out the questionnaire (see Table 18).

6.3.2 Outcomes

Heuristic Evaluations

Using the predetermined heuristics, each team member individually rated the mid-fidelity prototype on ten different aspects along with a short explanation of their rating for each category. An aggregated table of ratings is included below.

Table 17: Aggregated results of heuristic evaluations for mid-fidelity prototype from iteration 3

Category	**Average Rating out of 5 (based on four team members' input)**
Visibility of system status	4.625
Match between system and the real world	4.625
User control and freedom	4.56
Error prevention	3.375
Recognition rather than recall	3.875
Flexibility and efficiency of use	1.75
Help users recognize, diagnose and recover from errors (Error Recovery)	1.25
Help and documentation	3.625
Consistency and standards	4.5
Aesthetic and minimalist design	3.875

Although done independently, team members have given relatively consistent rating to each category overall. The team came to the conclusion that they should prioritize the categories of "flexibility and efficiency of use", "error prevention," and "error recovery." Error handling is still worth a lot of attention. However, this prototype has improved in terms of clarity on the labels compared to the previous iteration. In contrast, flexibility and efficiency of use was sacrificed to offer a less confusing and more straightforward workflow as a tradeoff. This means, in the next iteration, the team should aim to achieve a more flexible system, in addition to exploring more error handling methods.

User Testing

The goal of testing the mid-fidelity prototype was initially to test two variations of control flow and compare and contrast the merits of each based on user testing feedback. A total of five testing subjects participated in the user testing session that were spread across two days (see Appendix J for complete UARs). On the first day, two testing subjects reacted to two different versions of the prototype. Once the team collected sufficient feedback on each, the two

prototypes were combined based on their strengths and adjusted for flow. On the first day of testing, users provided valuable insights to the team by suggesting a combination of the strengths of both versions could be combined into one. Based on the feedback from day one that suggested confusion at a distinct menu screen, the team then merged two versions of design into one and conducted three more testing sessions on the second day. A full testing protocol can be found in Appendix D. User responses of their preference between the two versions also provided valuable information to the team on how to improve the control flow (see Section 6.3.1 for more design details).

Users also completed the SUS, as mentioned previously. The four participating users agreed that the program did not require a large learning curve (see Table 18 below). The SUS results conveyed a positive message to the team, particularly addressing the team's fundamental design goal: the design was easily understandable without much external guidance. The prototype had achieved satisfying results based on user responses in terms of the control flow and the team should move on to adding more visual elements to the prototype. Although users rarely made any errors to trigger the error handling functionalities implemented on this prototype, users expressed their appreciation of error prevention handling functionalities once they were introduced to them. The team should still look for better nomenclature and start making the wireframe prototype into a fully designed and interactive prototype.

Table 18: Aggregated results of SUS for the mid-fidelity prototype

SUS Questions	**Average Rating (1 being strongly disagree and 5 being strongly agree)**
I would like to use this system frequently.	3.75
I found the system unnecessarily complex.	1.75
The system was easy to use.	3.5
I would need technical support to be able to use the system.	1.5
Various functions in this system were well integrated.	4
There was too much inconsistency in the system.	1.75
Most people would learn to use this system very quickly.	4.25
The system was cumbersome to use.	1.5
I felt confident using the system.	3.75
I needed to learn a lot of things before I could get going with this system.	2

Conclusion

The team met their design goal at this phase because based on the SUS and user testing results, users consented that the system was intuitive. The team should aim for developing more error handling functionalities while taking time constraints into account. On the other hand, the team should make a final decision on the naming convention of labels so there could be less confusion, although this aspect had significantly improved from the previous iteration. Most importantly, the team should move on to including all design elements in the full prototype which should showcase all the necessary features. Additionally, if time permits, the team should explore more error handling functionalities and investigate whether to increase user restrictions so the system could be made even more straightforward.

6.4 Iteration 4 - User experience refinement with high-fidelity prototype

Predicting an application's usefulness involves making sure the application has sufficient utility and usability. After refining the prototype to achieve utility and usability, the application was made to provide a more intuitive user experience with a consistent appearance and layout. The primary goal of this iteration was to answer the following question: Does this application have a cohesive design and is it pleasant to use? In the final design stage, the prototype was transformed into a prototype with fully designed visuals and maximized interactivity.

6.4.1 Process

Aesthetic Styles of Final Prototype Design

The team created moodboards using Niice (see Figures 36 and 37), a creative review platform specifically focused on helping teams build effective graphic designs for web applications ("Niice", n.d), to enable them to articulate the correct tone for the users' demographic. The moodboards helped map out the team's potential choices of font, color, and styles which the team can use when creating the final prototype, the steps of which are further detailed in Section 7.1.2.

Figure 36: Moodboard for High-Fidelity Prototype Ver.1

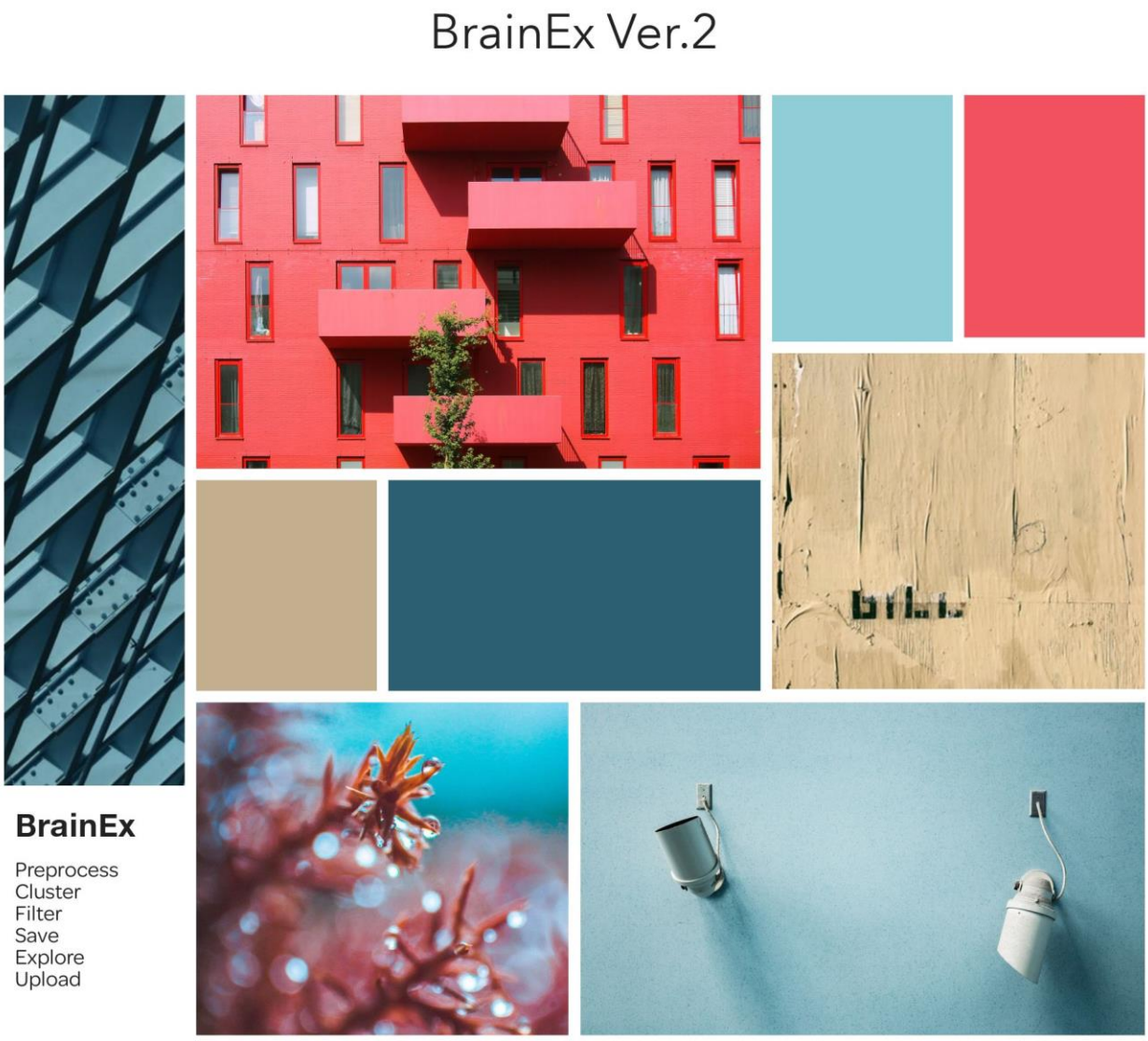


Figure 37: Moodboard for High-Fidelity Prototype Ver.2

The team collected feedback from five users in total and asked their preference on the two moodboards shown above. Four out of five users preferred Version 1; therefore, the team decided to adopt a blue color scheme and font choices of Open Sans (for texts) and Lora (for headers).

When designing the final layout of the UI, two important things the team considered were leveraging the target users' current mental models using metaphors and providing visual clues that suggest operations using affordances. The team referred to the following design principles ("7 Gestalt Principles of Visual Perception", 2019):

- Minimize visual interruptions (have panes on different screens structured similarly)
- Emphasize elements with the use of contrast (use different colors or font sizes to direct user attention)
- Direct user attention by manipulating their scanning patterns on the screen (place the boxes in the order of human scanning patterns)
- Group similar items together using proximity (place similar items more closely together)

By following these principles, the team was able to create a cohesive design theme with the overall appearance of the application. Screens for the finalized UI prototype were created in AdobeXD, a user-experience tool used to create designs of web pages ("AdobeXD", n.d), to allow more thoroughly designed screens. The team added a color scheme and font family to the designs and then added interactivity by uploading them to Invision, another user-experience tool used to add interaction and navigation between previously designed screens ("Invision", n.d). After the fully designed prototype was created, the team moved on to evaluation and testing for the finalized UI prototype.

Heuristic Evaluations

The team followed the same procedure to conduct heuristic evaluations as in Iteration 3, and the outcomes of which are further detailed in the next Section.

User Testing

For the final round of user testing, the team shifted their focus to the usability aspect to make sure the final product can make brain data analysis more productive, efficient, and less prone to errors. Using the user feedback from previous iterations, the team refined the final UI prototype and presented it to three members of the HCI lab staff to gather any final feedback. These members ranged in expertise from novice to advanced, bringing in a variety of viewpoints. The team also collected feedback from a high-fidelity specific SUS from all three subjects who participated in the user testing (see Table 20).

6.4.2 Outcomes

Description of the Final High-fidelity Prototype

To illustrate the overall control flow of the user-interface, the team created a site map that includes each major component of the prototype. Users may navigate back-and-forth between the screens shown in Figure 38 as needed.

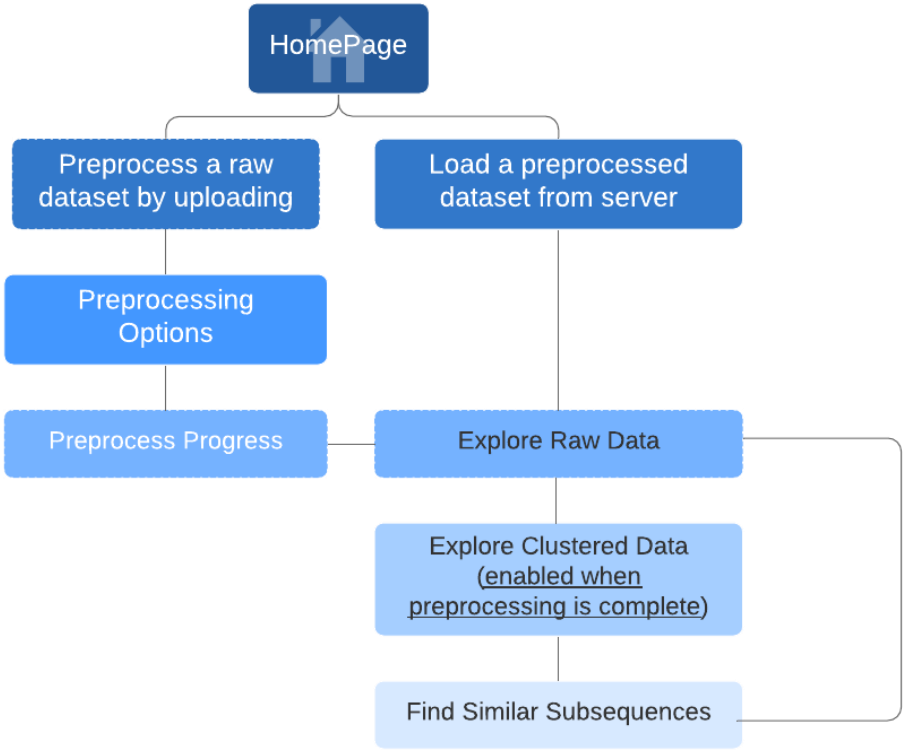


Figure 38: Site map for the final prototype

Users begin on the homepage, which allows two main operations: preprocessing raw datasets by uploading a CSV file (either by selecting from the server or uploading a local file to the server and then selecting) or loading preprocessed datasets (in gxdb format) from the server. Figure 39 below shows the key features of this process, including the server-side datasets, adding files to the server, and previewing a raw dataset.

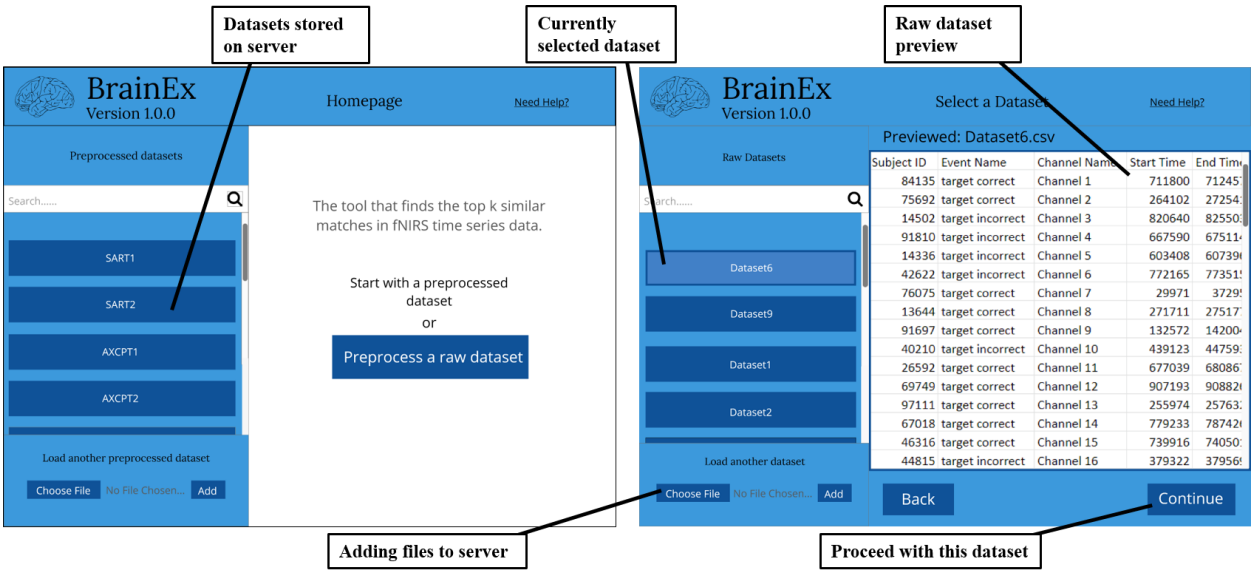


Figure 39: Select a preprocessed dataset and select a raw dataset screens, annotated with key features

The process of selecting a preprocessed dataset (above, left) is identical to selecting a raw dataset other than users previewing the contents of a raw dataset before proceeding to preprocessing. The previewed dataset is inset to indicate that it is currently selected. The user may also use the search bar above the listed datasets to filter by name, but this feature is not currently interactive within the prototype.

Once the user uploads an unpreprocessed dataset, the user is provided with a set of parameters with default values already selected (see Figure 40), allowing them to make adjustments before proceeding. Then, once the user initiates preprocessing, users are free to explore the raw data separately while the process is in progress. Once preprocessing is completed, users are able to explore clustered data.

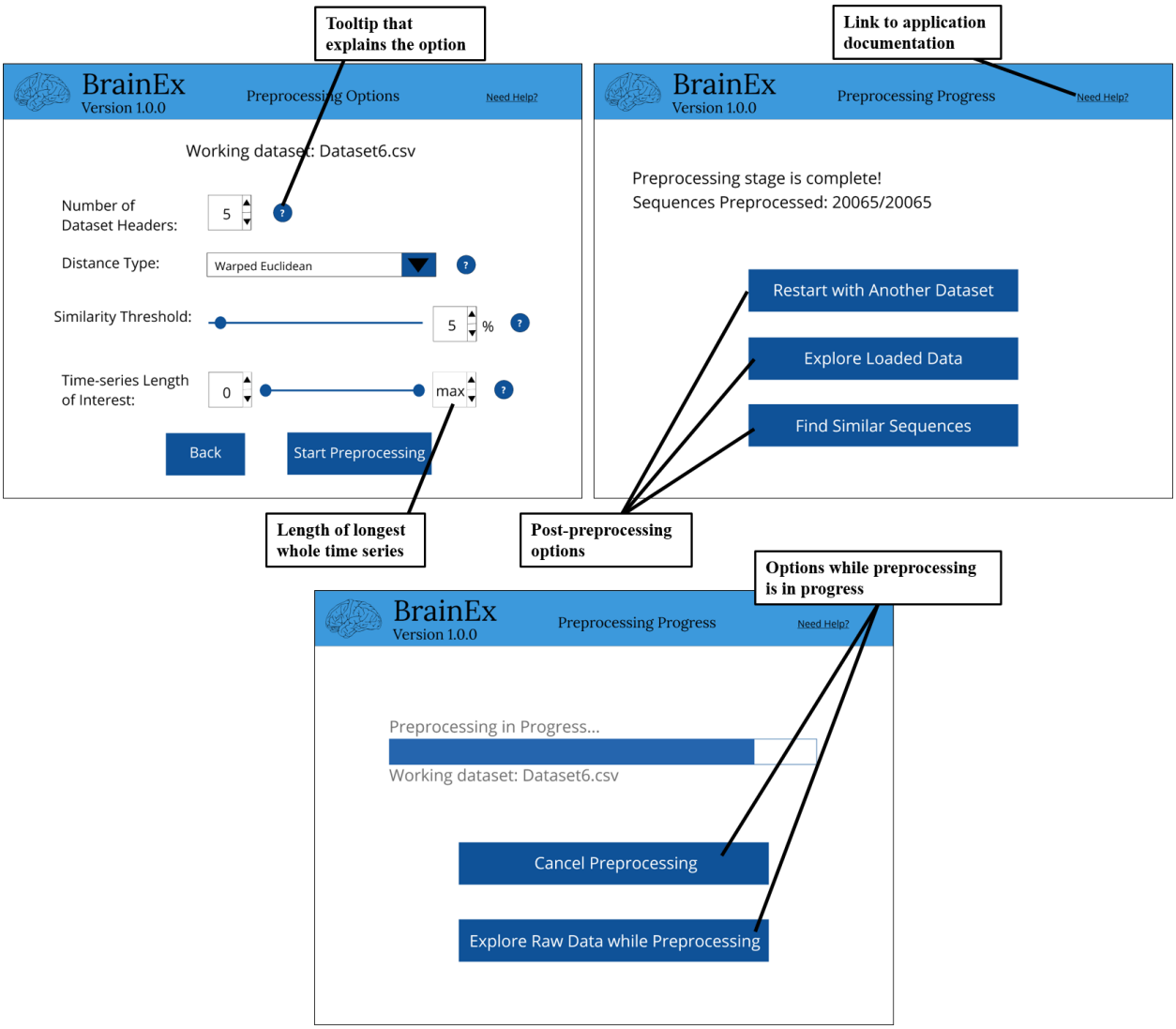


Figure 40: Preprocessing screens annotated with key features

As shown above (top left), each parameter also has a tooltip that the user can view if they need clarification on what a parameter is and how it affects the preprocessing. These tooltips can also be found throughout the application in areas we felt required more guidance based on user feedback. In the bottom screen shown in Figure 40, the user is provided with a progress bar indicating how far along the preprocessing is. While this is still in progress, the user can either cancel it entirely or view the original raw data.

Users are able to select a subsequence in order to find similar subsequences in the given dataset by:

- selecting a subsequence from raw data,
- selecting a subsequence from clustered data, and

- uploading a saved subsequence from a local drive.

Users may upload a pre-saved sequence or select one from the raw data even while the dataset is being preprocessed. However, they cannot find similar sequences to this selection until the preprocessing is complete. The current sequence selected, as well as the option ofuploading a sequence, can be found in the top left of the screen, as shown below in Figure 41. Located on the same panel, the save icon button allows the user to save their current selection to their local drive.

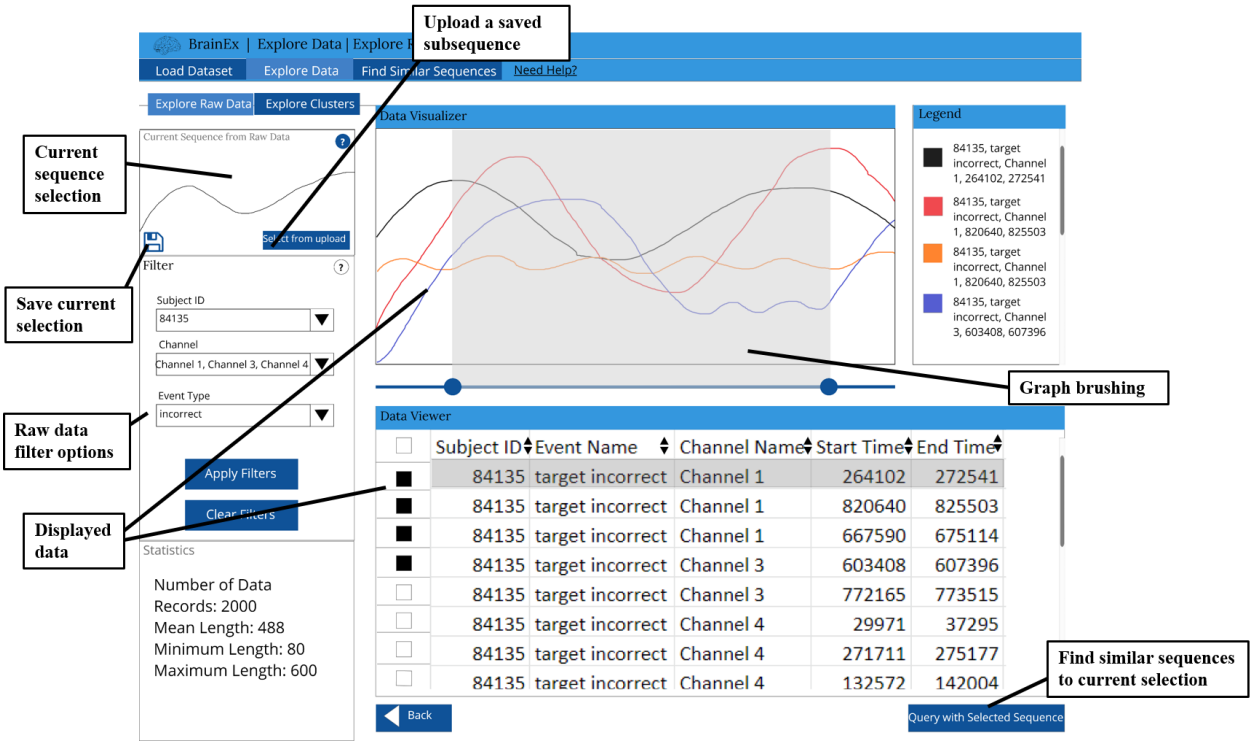


Figure 41: Explore raw data screen annotated with key features

When the user views the “Explore Raw Data” page, they are presented with a table of the whole time series within the dataset in the Data Viewer as well as dynamically populated filter options with various types of labels that together form the unique ID of the time series . The user can filter the data by any combination of filters and apply them to the Data Viewer content.

To view time series in the Data Visualizer, the user can select and deselect multiple items in the table, as well as in the legend (top right of Figure 41). If the user wants to take a closer look at a particular region of the time series, they may use the sliders below the visualizer or “brush” on the graph (click and drag on the graph to select the desired interval) and the graph will zoom into the desired region.

The current selection window will update with the users most recently selected time series. When the user wishes to find similar sequences with their selection from the raw data, they can select “query with selected sequence” to proceed to Find Similar Sequences.

Alternatively, a user may also select a time series from the Explore Clusters page. These sequences are the “exploded” time series created during preprocessing; they consist of subsequences of varying length of the original set of time series. The subsequences are grouped by similarity. They are represented by a single subsequence that is used in the querying process. In the initial Explore Clusters screen, shown below in Figure 42, the user is able to view just the clusters and a visual of their representative. In addition to the filter options on the top left that include the number of clusters shown and the length of interest, the user may also sort the Data Viewer to show the top largest clusters or the top longest clusters using the drop-down annotated below. The clusters also have an additional legend on the left-hand side that shows the cluster details alongside a thumbnail of their representative.

Figure 42: Explore clustered data screen annotated with key features

The currently selected cluster is indicated by the selected cluster being highlighted on left-side legend and in the Data Viewer. If the user wishes to view the contents of the cluster, they can do so by selecting the cluster in the legends, Data Visualizer, or Data Viewer and selecting “View Selected Cluster”. This will bring them to a nearly identical page, but the top left filter options are replaced with the current selection window and the Data Viewer contains the

sequence information of the contained sequences, similar to that of Explore Raw Data. If they just wish to search with the representative of the cluster, thought, they can select “Query with Selected Sequence”.

When on the Find Similar Sequences screen (see Figure 43), the user is presented first with empty Data Visualizer, Data Viewer, and Legend components. To find matches, the user must adjust the query parameters to their liking and initiate the query with “Start Query”. The user will then be provided with useful statistics about their results, including the average similarity value as well as the standard deviation of how similar the results are to the query. The Data Visualizer functionality remains the same across all components, but the Data Viewer in Find Similar Sequences lists the query sequence as well as the results in ranked order of similarity.

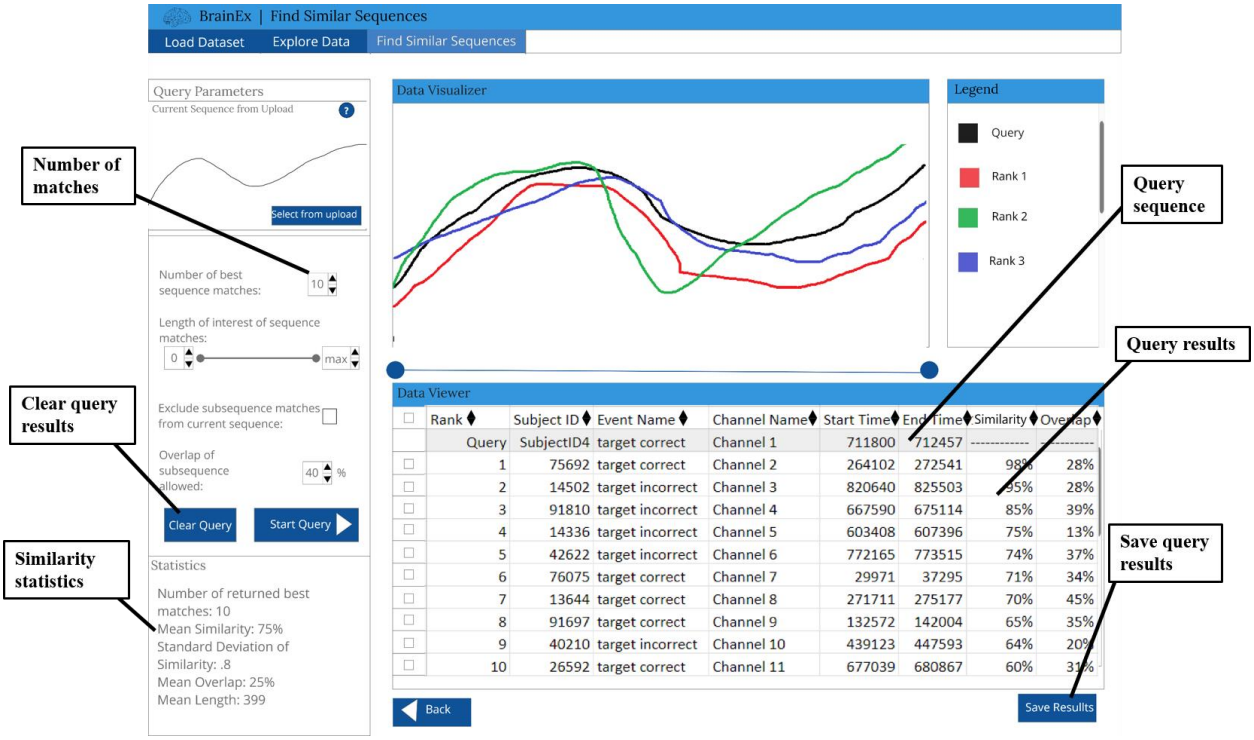


Figure 43: Find similar sequences screen annotated with key features

The user can save their query results to their local drive if they want to refer back to the results later. At the moment, reloading these results back into the application is not in our current scope, but the results will be formatted in such a way that they can be loaded into other tools for viewing fNIRS data.

Heuristic Evaluation

The team completed heuristic evaluations individually and all the results were aggregated in Table 19. As shown, error prevention and error recovery had the lowest rating among all the

categories that were rated. This was expected by the team because there were nearly no error handling functionalities implemented in the final prototype, due to time constraints as well as the technical limitations of Invision. However, this will be made entirely feasible once the team is able to move on to the implementation phase and this aspect has been considered as necessary in the team's implementation goals, which will be introduced further in the Discussion Chapter. The aesthetic style could be further polished if the team was given more time to clean up the design but the prototype was able to maintain a simplistic design.

Table 19: Aggregated results of heuristic evaluations for mid-fidelity prototype from final iteration

Category	**Average Rating out of 5 (based on four team members' input)**
Visibility of system status	4.125
Match between system and the real world	4.25
User control and freedom	4.625
Error prevention	3.375
Recognition rather than recall	4.25
Flexibility and efficiency of use	3.875
Help users recognize, diagnose and recover from errors (Error recovery)	3
Help and documentation	4.5
Consistency and standards	4.375
Aesthetic and minimalist design	4.125

User Testing

User testing was performed with three users, one of which was from the more experienced user group and the other two were from the novice user group (see Appendix E for full user testing protocol). This helped the team to gather feedback from more perspectives and understand the learning curve of the prototype required for different target user groups. The team proposed an additional set of questions that were related to the overall look of the user interface and the team received an average score of four out of five on the overall look of the UI. The

success rates of user testing was almost a hundred percent and the average task completion time was less than 3 minutes, though the team acknowledged that testing subjects had had previous experience from the mid-fidelity prototype user testing to become more familiar with the application.

Upon comparing the SUS collected for the high-fidelity prototype with the one for mid-fidelity prototype, all but one category received a better score. Specifically, users did not think various functions in this prototype were as well integrated. This might be due to the fact that the interactive prototype on Invision contained some improper linking between the pages, thus giving the users a less pleasant navigation experience. This concluded the design phase and allowed the team to transition to the next step in the project, implementing the finalized UI and integrating it with the backend.

Table 20: Aggregated results of SUS for the high-fidelity prototype

SUS Questions	**Average Rating (1 being strongly disagree and 5 being strongly agree)**
I would like to use this system frequently.	4.6667
I found the system unnecessarily complex.	1.3333
The system was easy to use.	4
I would need technical support to be able to use the system.	1.3333
Various functions in this system were well integrated.	3.6667
There was too much inconsistency in the system.	2
Most people would learn to use this system very quickly.	4.6667
The system was cumbersome to use.	1
I felt confident using the system.	4.333
I needed to learn a lot of things before I could get going with this system.	2

6.4.4 Conclusion

The high-fidelity prototype was designed to finalize the overall aesthetics of the prototype, while addressing inconsistent nomenclature issues. Overall, users expressed that there has been a great improvement since the mid-fidelity prototype and users were not experiencing major difficulties completing the assigned tasks, which were documented in more detail in Appendix K. At the end, the team decided to apply necessary changes to the prototype, i.e., the flaws that might lead to the misunderstanding of the system structure and make the visual representation more consistent. For instance, a mistake was made in the high-fidelity prototype draft where users would not be able to select a subsequence until preprocessing executes to full completion; such mistakes were highly prioritized by the team and fixed instantly after it was found. Changes that would require longer input time were documented to be applied for future development plan (see Section 9.3 for more details).

7. Implementation of the User-Interface

Implementation of the BrainEx user-interface was started while some members of the team were still completing prototyping since Vandana needed to complete the project early. Therefore, Vandana completed Phase 1 of implementation. We defined phase 1 as the selection of the tools we would eventually use to implement the entire interface and the creation of the framework for the interface. All other implementation work was part of Phase 2, completed by all other members of the team.

7.1 Phase 1: Selection of implementation tools and creation of interface framework

During the last design phase, the high-fidelity prototype, one of the team members, Vandana, started the implementation phase of the project. This phase was continued by the three other team members once the final design decisions had been made and the final prototype has been evaluated.

The goal of this phase is to ensure that the user-interface designs created in the previous steps can be translated into a fully integrated system as well as ensure that an initial framework and functionality is implemented so that the rest of the team can easily pick up where the implementation was left off.

7.1.1. Selection of implementation tools

The team's understanding of the command line code and user needs informed their selection of tools and languages used to build the frontend. In particular, Vandana pooled knowledge of relevant frontend tools, such as AngularJS and React, by researching online and holding discussions with the WPI HCI Lab members and development team. All the team members then evaluated each tool based on a table. The table details how the tools/languages would interact with technical and user needs. Each technology has a row and each need has a column. Vandana rated each technology on a scale of 1-5 based on how she thought it would fill the need, then explained why. The team then selected their tools based on which tools show the most promise in covering all needs. The tables may be revisited as needed if gaps in coverage are found during implementation, although this is not expected since all needs should be discovered in past analysis.

Table 21: Technology selection justification

Technology	Easy to learn	Integrate with other technologies	Interesting features
JavaScript/HTML/ CSS These programming languages go hand in hand to help developers create well designed, functional user interfaces. HTML ("HTML: Hypertext Markup Language", 2019) and CSS ("CSS: Cascading Style Sheets", 2019) is used to structure content and present the appearance of the content, respectively. JavaScript ("JavaScript, 2019) is a high-level, object-oriented programming language used to provide functionality to the content on the screen.	**4** These three programming languages are easy to pick up and start implementing right away and have lots of documentation online.	**4.5** These technologies can be integrated with mostly all web development languages and tools such as Angular, React, D3, Bootstrap, and Express.	**4** There are many content structuring and presentation templates that provide an interesting way to portray a web page.
AngularJS AngularJS ("AngularJS", 2019) is a JavaScript-based open-source front-end web framework mainly maintained by Google, a community of individuals, and corporations to address common challenges of	**3** Angular is a little difficult to set up in the beginning but easy to adapt and learn.	**3** Angular integrates well with HTML/CSS, but not with visualization technologies like D3.	**4** Angular has a very good framework for structuring code and is neatly organized.

developing web applications.			
React React (“ReactJS”, 2019) is a JavaScript library for building user interfaces and is optimal for fetching and storing rapidly changing data. It is maintained by Facebook and a community of individual developers and companies.	**4** With React, there is lots of documentation online and it is very widely used. In fact, during conversations with the backend development team, many of the members have used React so they can be contacted for any questions. React is also easy to use and setup.	**4** React is easy to integrate with most technologies and can work well with D3.	**4** Users can get started with creating applications right away and React has a nice framework.
D3 D3 (Bostock, 2019) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers.	**3** D3 has many components involved so it is difficult to learn initially. However, there are lots of examples and documentation to learn from online.	**3** D3 can only be integrated with limited technologies and React is one of them.	**4** The graphs/visualizations made in D3 are useful and colorful. D3 provides interesting features built into visuals such as zooming and panning.
Bootstrap Bootstrap (“Bootstrap”, 2019) is an open-source CSS framework for creating responsive, front-end web pages. It contains CSS- and JavaScript-based design templates for typography, forms,	**5** Bootstrap is widely used and is one of the best and consistent CSS templates.	**5** Bootstrap easily integrates with technologies such as Angular and React.	**5** Bootstrap provides lots of interesting styles and components to pick from. It gives any application a unique aesthetic.

buttons, navigation and other interface components.			
Express/Axios/Node.js Express (“Express”, 2019) is an open-source web application and API framework for Node.js (“Node.js”, 2019), which is a JavaScript runtime environment that executes JavaScript code outside of a browser. Axios (“Axios”, 2018) is used to retrieve data from the client side, or frontend of the application and process the data in the server.	**4** These technologies are widely used to develop servers to communicate between the backend and client side of a web application. They are easy to install and integrate into a project.	**4** The technologies easily integrate with most web development application languages such as Angular, React, D3, Bootstrap, and Javascript/HTML/CSS	**4** There are interesting ways to develop the server and many features that add value, such as data protection, to the web application

From Table 21, the team decided that between React and Angular, React would be the best tool to use to implement the frontend of BrainEx because it scored highly in terms of effectiveness, learnability, and integration with other tools. The team will also utilize D3 for visualizations and graphs depicting the data since it works well with React. Bootstrap, HTML, CSS, and JavaScript will be used because they are both widely used and have interesting built-in frontend designs. Express, Axios, and Node.js will be used to develop the server to help communicate functions and logic between the backend and the client side of the BrainEx web application.

7.1.2 Frontend and Server Framework

Once the team decided the technologies they would like to use to program the UI, Vandana began to implement it. To start off the development process, Vandana created a set of initial tasks for development in regards to the server and frontend framework. Vandana first set

up the project environment, ensuring that the technologies selected for implementation, such as React, JavaScript, HTML CSS, Express, Axios, Node.js, and Bootstrap were configured properly.

For the GUI, she first focused on creating the page components and basic templates for each of the screens in the high fidelity mockup design. She then established the routers and clickable buttons on the pages to navigate between the different screens. After this, she focused on implementing the design of the BrainEx homepage (see Figure 44) using React, HTML, CSS, and Bootstrap to make sure it matched the designs and aesthetics of the high-fidelity prototype.

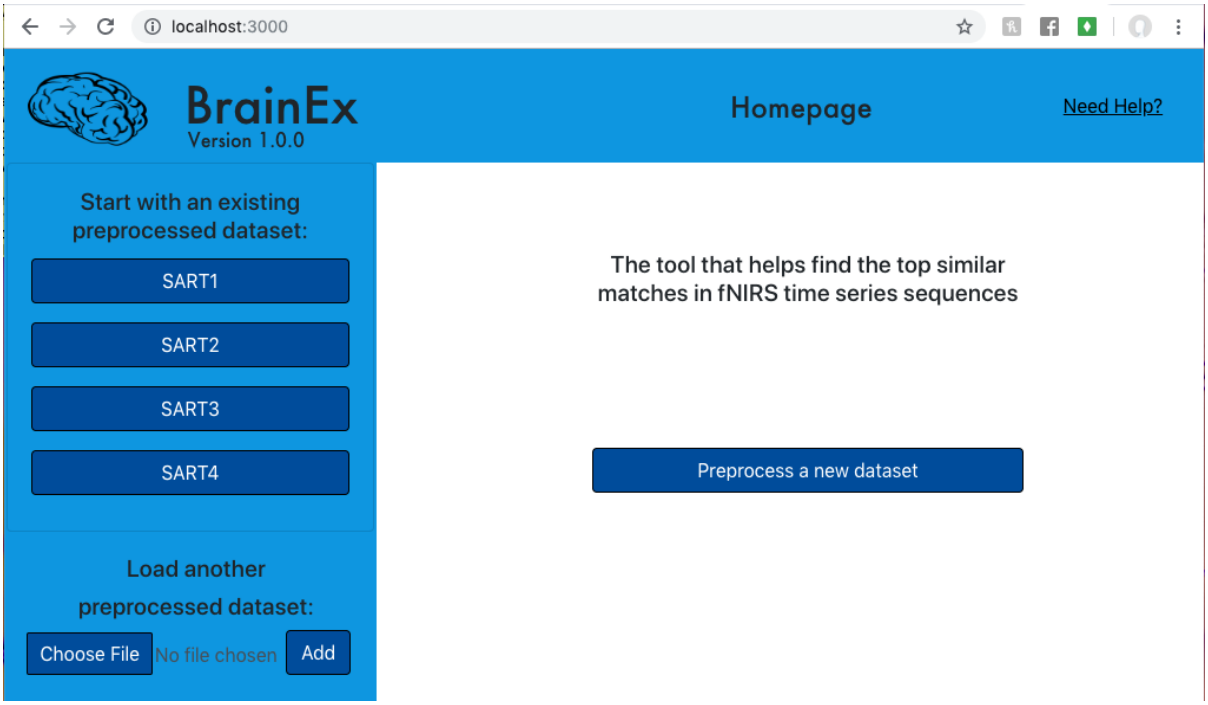


Figure 44: Screenshot of the original implemented homepage of BrainEx

Next, Vandana also added the header bar that contains the BrainEx title, logo, version, and screen title to the rest of the pages of the application. She also started creating a more structured and detailed layout of the CSV data viewer page and preprocessing options page (see Figure 45).

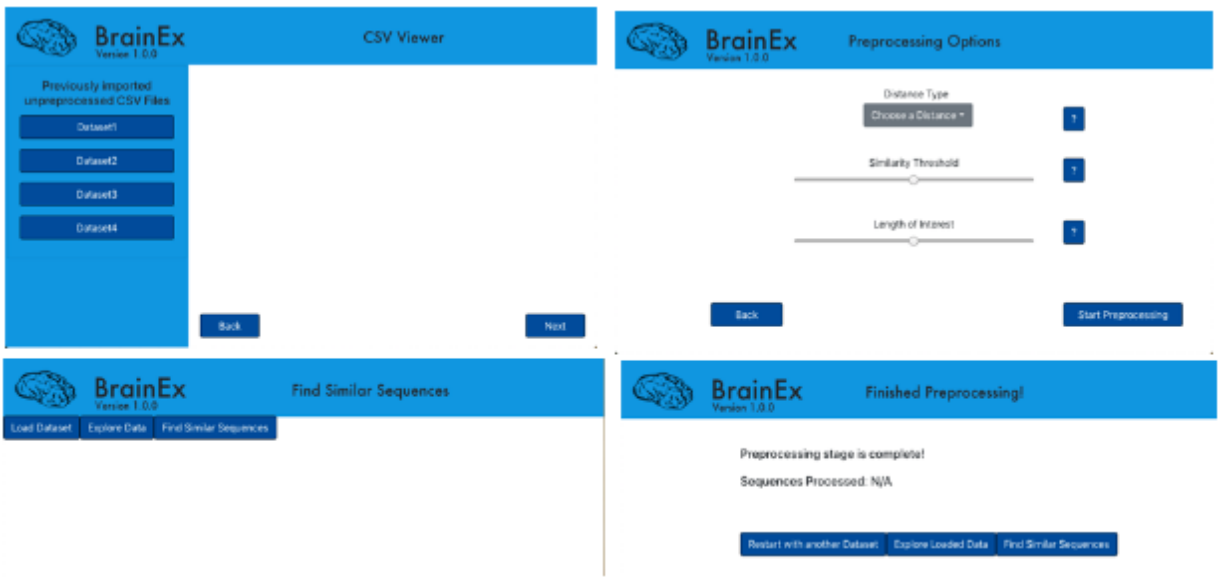


Figure 45: Screenshot of the other Phase 1 implemented pages of BrainEx

Then, Vandana added the BrainEx command line tool that the development team had been working on in the UI project. She first met with the developers to discuss the functionality and capabilities of the backend. She then ran through it using the command line to create the gxdb object that contains information about the inputted time series sequence's clusters and query results. This process will make it easier for the rest of the team to retrieve functions from the backend to apply and process the user inputs in the UI. Vandana also started working on the functionality to choose time series csv files from the user's local files on their device to the homepage of the UI. For this functionality, she first consulted a tutorial (Hamedani, 2019) to learn about setting up and implementing the Express and Node.js server in addition to setting up Axios in the frontend to upload files. She then wrote the server logic and frontend data-retrieving code to ensure there is communication between the server and the client side of the interface. This allows the application to retrieve multiple files that the user uploads to the UI and store the files in a folder on the server in order to access them when the user wants to preprocess the data. Vandana also implemented an error checking functionality that only allows the user to upload files with the ".csv" extension.

In addition to implementation, Vandana created a BrainEx tutorial document, Appendix L, that highlights how to run the application locally, how to use the application and run through the functionality at its current state, and an explanation of the project structure along with details of the contents of each file and folder. She also highlighted the future work in Section 8.3 that goes through the tasks that she has already done as well as the tasks she couldn't get to due to lack of time.

Overall, creating the initial framework to navigate between different pages of the application, implementation of the design of the homepage, CSV data viewer, and preprocessing options, and functionality to upload files of BrainEx will help build a foundation for the rest of the team members to immediately pick up the development process.

7.2 Phase 2: Implementation of Query Pipeline

When we began implementation, we knew we would have to break up the app into smaller pieces so that each person would have a set number of tasks to work on. We also knew that development often doesn't go according to plan, so we needed a plan that could be flexible. We wanted a full, working product by the end of the project, even if we couldn't implement every planned feature. Therefore, each person working on the project at this point was assigned a focus area based on their skill sets and learning interests. Kyra is knowledgeable in Python and wanted to learn more about how BrainEx handles data, so she focused on the Python code to wrap the BrainEx library. Margaret has experience in React and wants more practice programming frontend code, so she focused on the general frontend structure and capabilities. Finally, Yihan is also interested in the frontend and specifically has an interest in design, so she focused on representing the data processed by BrainEx.

In addition, we needed to order tasks to ensure we had a working product by the end of the time allotted for the project. Therefore, we decided to focus on implementing all tasks related to completing a query before implementing the dataset explorer or cluster explorer. This includes preprocessing, uploading a query sequence, and running a query. We believe this is the most important pipeline because it is the core functionality of the BrainEx interface.

To make sure we were completing tasks, we used the Trello online tool. Trello allowed us to make "cards" for each of our tasks that we could move around between categories on a board depending on their development status. It allowed us to see how many tasks we had created, were working on, or had finished. In addition, we met at least three times a week (if not daily) outside of our meeting time with our advisors to have a standup meeting in which each of us could discuss our progress and any challenges we may have had. Therefore, we were all knowledgeable in the progress of the app and could help another team member if they had a problem. In addition, we used a Git repository hosted on GitHub for version control of our code. Each of us had a separate branch so we could develop without breaking the current working version of the code. Once we developed to the point that an important feature was implemented and worked properly, we merged the code to master so that the master branch always had a working version of the application.

The final application allows end users to query for similar sequences in a CSV time series dataset of their choice. Due to the time constraints of the project and obstacles encountered, dataset

explorer and cluster explorer were not implemented, but we provide a plan for doing so. The specific implementation processes for different focus areas in the app are detailed below.

7.2.1 Python Server

Since the backend library for the interface is written in Python, we had to have some way for our frontend to call Python functions. The easiest method we found was to write a server in Python using a library called Flask. This library allows for RESTful API methods to be created for Python code. A RESTful API allows for HTTP "requests" to be made to a URL for a function to be completed. An HTTP "response" is returned that either confirms the proper execution of the function or provides an error message. Both requests and responses may pass data between the calling application and the API. In order to develop these methods, we looked back to the models we made prior to prototyping. Since we were focusing on getting the query pipeline to work first, we decided to implement the necessary functions to complete a query successfully using our interface. As we continued to work on the frontend, we could then use the Axios JavaScript library to make HTTP Post requests to these functions to complete the connected task on the backend. For example, when the user inputs the preprocessing arguments and clicks the "Start Preprocessing" button, the frontend uses Axios to send an HTTP Post request to the backend with the form information included, so that they can be used as arguments for preprocessing using the backend.

As we continued to develop the frontend, we also found a few more necessary functions that were not included in the models. For example, we have panels on the "Select preprocessed" and "Select raw dataset" pages that display names of files that have already been input into the application so that the user can query with those files without having to re-upload. It is not something we anticipated, but those file names must be provided by the backend. While we are implementing our application as a local application, if this were to be transferred to a central server setup one day, it would be especially imperative that the backend delivered the names of the functions instead of the frontend retrieving them itself, as they will be separated even more. Therefore, we wrote two functions: one that returns the list of preprocessed file names and one that returns the list of raw CSV files that are already 'uploaded' to the backend. The full list of functions and a description of what they do can be found in Appendix Q.

7.2.2 Frontend Design

We chose React, a Model-View-Controller (MVC) JavaScript framework, to develop the frontend because of its lightweight performance and compatibility with full stack technologies based on our evaluation prior to development. We found React to be extremely useful in development due to several of its features. Utilizing reusable components, React is able to keep a copy of the Document Object Model (DOM), also known as the virtual DOM to only render elements that need to change based on received data, thus optimizing the performance dynamic displays. React utilizes the DOM lifecycle and state values in order to achieve its lightweight

performance. Each component can have a state with one or more variables that contain some initial value, that can then be referenced throughout the JavaScript and HTML code. When a state value is changed, React re-renders only the component that uses the state value.

When approaching the front-end this term, it was important that the code was compartmentalized and design in such a way that allowed minor changes without compromising the appearance of the application. This involved using more dynamic CSS as well as following proper React design principles. The original code written in the previous term did not exhibit these qualities, and required significant refactoring to allow for proper scaling.

Alongside this technology, we also decided to use Material UI, a component library specifically made for React and React hooks, to achieve a more unified and consistent application design.

7.2.3 Visualization

Visualization of sequences, including the sequence thumbnails in the Ranked sequence matches table, the query result sequence plot, and the query sequence line chart, were all done through Recharts, despite our evaluation of D3 as best for our purposes prior to development. Recharts is a data visualization library native to React that is built on top of SVG elements with a lightweight dependency on D3 modules. It does not offer as much customizability as D3 does, but it allows simple visualizations to be quickly built on the fly and does not conflict with the React lifecycle.

Recharts offers many examples and detailed tutorials online; it is easier to adapt compared to D3. It also offers built-in API functions for tooltips, legends, axis labels, and brushing. We did not include tooltips in the design, however, due to performance concerns. The actual drawing functions are only re-rendered when the state variables change, which are primarily linked to the data being plotted on the chart.

With Recharts, the only remaining challenge pertained to data parsing. The way the time-series data points are formatted in the datasets did not match the way Recharts parses data, making it was necessary to reshape the line data points as well as assign a unique color to each row of data. Our steps to achieve the mapping is described in the section below.

7.2.4 Data Parsing

A key issue with having different technologies for the frontend and backend of the application is passing the data between the two. As data is primarily passed through HTTP requests in the BrainEx UI, it is passed in the JSON format. The JSON format is a string that formats data through a series of key-value pairs. Therefore, both the frontend and backend developers became

very familiar with encoding and decoding these JSONs to get the necessary information across. Oftentimes, the frontend would need the data in a specific shape in order to display it properly, or the backend would need arguments that were specific data types. We often used the Python library pandas to achieve this. This library is very responsive and allows for easy manipulation of data in a ‘data table’ format. Since the data we are working with has originated in CSV files and is only being fed through JSONs, it is in the proper format to be processed with padas without extra parsing. Using pandas, it only takes one Python command to remove a column of data or change the shape of the output JSON.

After the data is fed into the frontend from the backend, this data is then communicated using React’s props across multiple components. On the “Ranked Matching Sequence” table, colors in hex format and a column of checkbox values, which determine whether a sequence is being shown on the plot, are appended to the rest of the data before it is sent to the sequence plot as a JSON response.

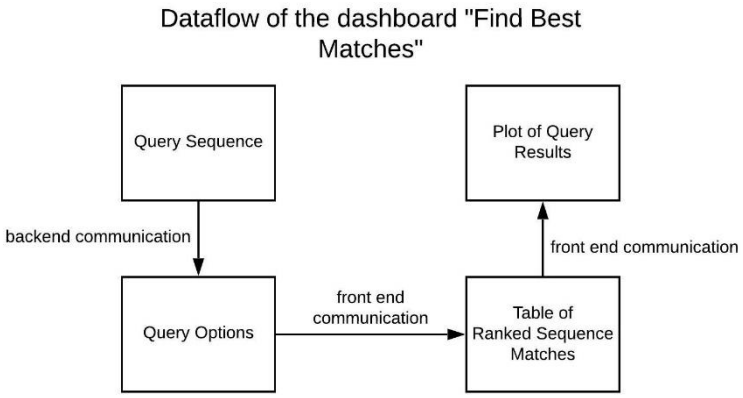


Figure 46: Diagram of the query data flow in the application

Then, another layer of data parsing is required as the data is passed from the “Ranked Sequence Matches” table to the “Query Results” plot. As the “Query Results” plot only requires data points with the proper ID as the header, more parsing is done with JavaScript on the frontend. Color codes are separated out as an array of strings, before they are mapped on the chart to fill in the lines on the sequence plot.

8. Results

The results of our design and development phases include our deliverable of a web application that provides a full, robust, tested pipeline for finding similar sequences within a time series dataset. It is tailored to users within the WPI HCI Lab working with data from fNIRS BCIs. We verified this by completing final testing with users within the lab.

8.1 User Testing

8.1.1 Goal

We aimed to collect a final round of user testing feedback so users could help highlight key changes needed in order to release the BrainEx UI as a final application. We hoped to use the feedback to prioritize the essential parts of our development plans to polish the final product.

8.1.2 Process

We completed a total of 3 user testing sessions, one with a graduate student who is very experienced in the lab and two with undergraduates who have a programming background but have less background in the fNIRS data analysis realm. One of the testing sessions focused on the application setup process while the other two emphasized on the potential improvement of the application design more.

We asked users to download the required packages with instructions from our documentation and provided them with guidance whenever there was a need. Then we asked the user to preprocess a new dataset and query using the default parameters. The users were asked to think aloud while they were interacting with the application.

At the end of each user testing, each participant completed an online system usability scale questionnaire (see detailed questions in Table 22 below) identical to those completed during the prototyping phase.

8.1.3 Outcomes

Most users experienced various levels of difficulty in installing all software requirements for the application. It was also made clear that our instructions were not as clear as possible and assumptions of the users' software environment on their personal machines were made without careful thinking. This process helped us acknowledge the details we missed in our documentation.

Throughout the testing sessions, we were able to identify many flaws in our application. Since we were still polishing the application at the time of user testing, many of the concerns raised

by the users were within our expectation. For instance, the font was too small to see clearly on some pages. In addition, necessary error prevention functionalities were not present in our system. Nonetheless, we were still able to collect valuable insight such as the suggestion to implement a button for saving query results, as well as making the distinction between dataset selection pages for preprocessing and raw clearer and more explicit. We applied most of the feedback right away to improve the design of the application as most of them only required small fixes from the frontend. For instance, all the testing subjects indicated that the font size on all the pages should be made larger and button colors should be made more visible, and we quickly adopted those changes where we saw fit. User feedback also tremendously helped us shape our documentation, since we were able to identify many details that we overlooked from developers' point of view.

Overall, we could see from the SUS results (see Table 22 below) that users were mostly satisfied with our application and did not experience much difficulty using the system. They felt the system was intuitive and had a consistent look throughout all the pages. The only major difference compared to the previous SUS result during prototyping was that users felt that they required more technical support to use the system. This also emphasized the importance of documentation, and we were able to produce a thorough documentation/tutorial on the usage of the application (see Appendix R).

Table 22: Aggregated results of SUS for the final product

SUS Questions	**Average Rating (1 being strongly disagree and 5 being strongly agree)**
I would like to use this system frequently.	4
I found the system unnecessarily complex.	1
The system was easy to use.	4
I would need technical support to be able to use the system.	3
Various functions in this system were well integrated.	4.5
There was too much inconsistency in the system.	1.5
Most people would learn to use this system very quickly.	4

The system was cumbersome to use.	2
I felt confident using the system.	4.5
I needed to learn a lot of things before I could get going with this system.	1.5

8.2 Description of the Final Application

Currently, the web application is broken into 2 main parts, with the first part being preprocessing and the second part being querying. There exist two main routes for preprocessing depending on whether the user already has a preprocessed dataset or would like to preprocess a raw dataset. Then once preprocessing is done, querying functionality is consistent across the two use cases.

If the user's goal is to query with a dataset that has never been preprocessed before, they need to first click on the "Preprocess A New Dataset" button on the homepage, then using the panel on the left, the user needs to upload a CSV file to preprocess by clicking on the "Choose Files" button on the lower left hand corner of the page.

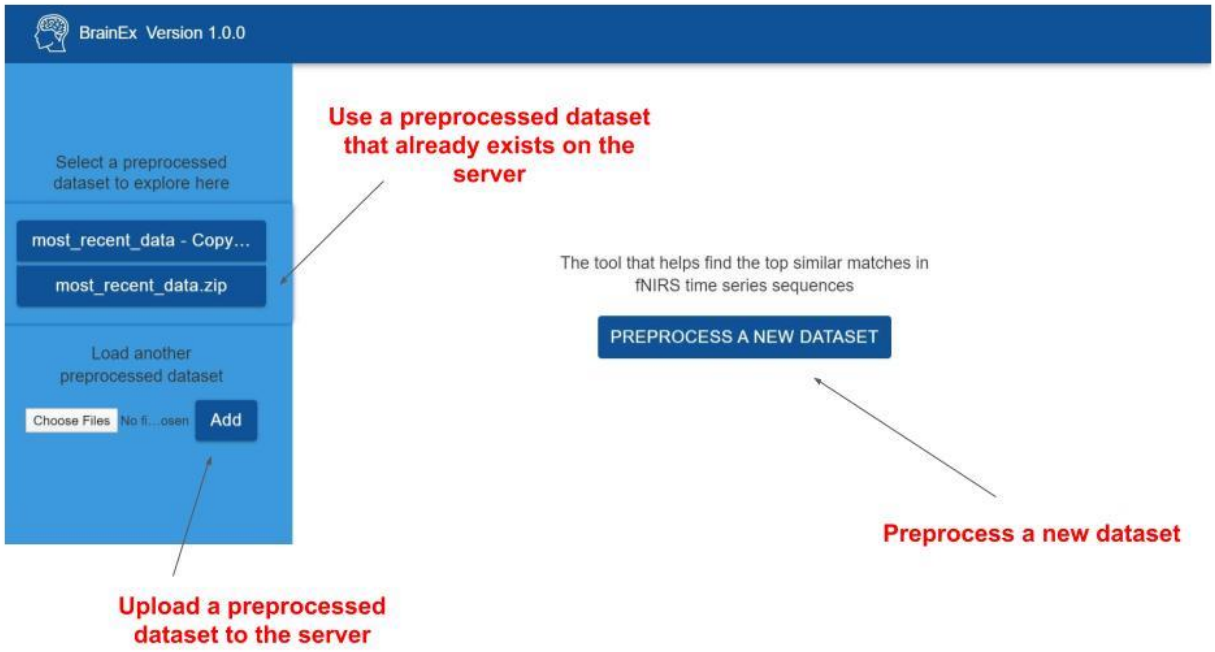


Figure 47: Home Page of the BrainEx UI

Once a file is selected on windows explorer, users can preview the dataset they have chosen by clicking on the “add” button near the “Choose Files” button.

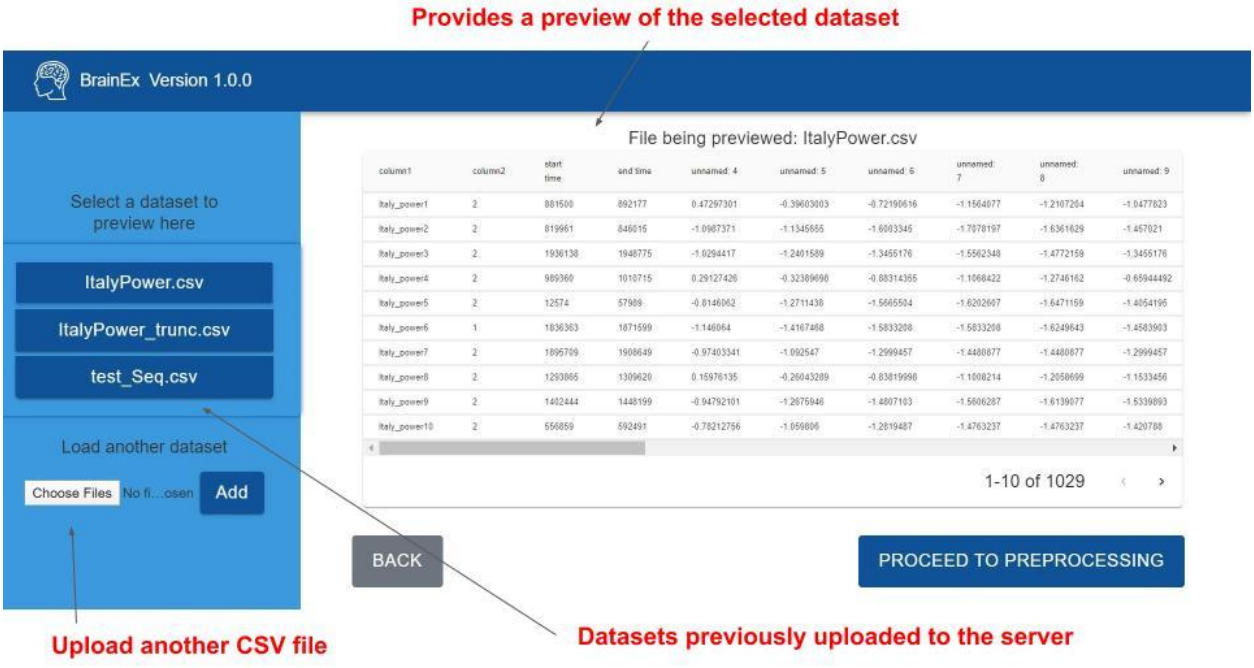


Figure 48: Select New CSV Screen of BrainEx UI

Once users verify that the file selected is correct, they can proceed to preprocess this dataset by clicking on the “Proceed to Preprocess” button on the lower right-hand corner. This will take users to the “Preprocessing Options” page, where users can fill in parameters for preprocessing. The user has the option to either use Python’s native multithreading processing or Apache Spark. Spark often runs faster but can be complicated to install locally depending on the host operating system. Therefore, if the user selects that they wish to use Spark, the program will check to make sure it is installed correctly.

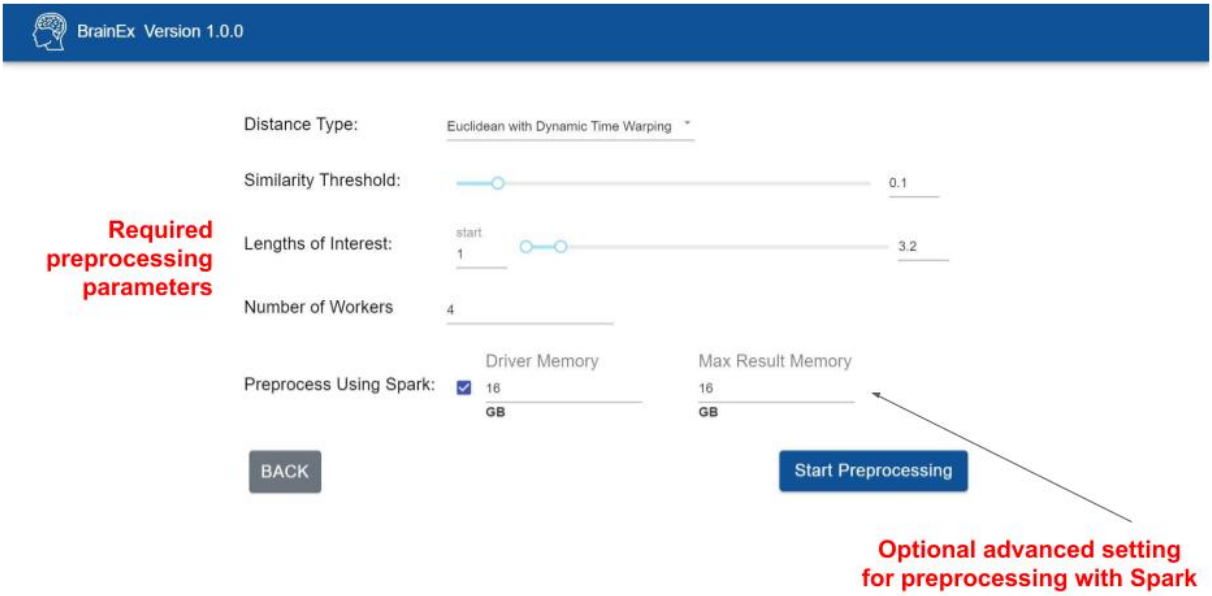


Figure 49: Build/Preprocessing Options Screen of BrainEx UI

Clicking on the "Start Preprocessing" button will take the user to the "Preprocessing Progress" page where an ongoing indeterminate linear progress bar is shown in the middle of the page to indicate that the preprocessing step has not been finished yet.

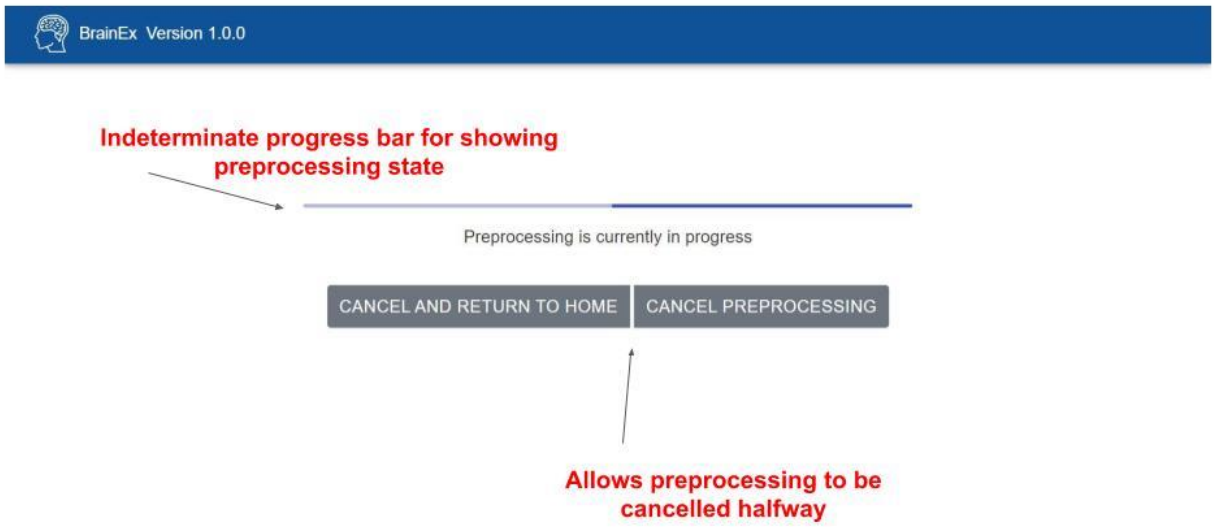


Figure 50: Build/Preprocessing Progress Screen of BrainEx UI, Incomplete

Once preprocessing is finished, users have the choices to either 1) cancel and go back to home or 2) query using the preprocessing dataset.

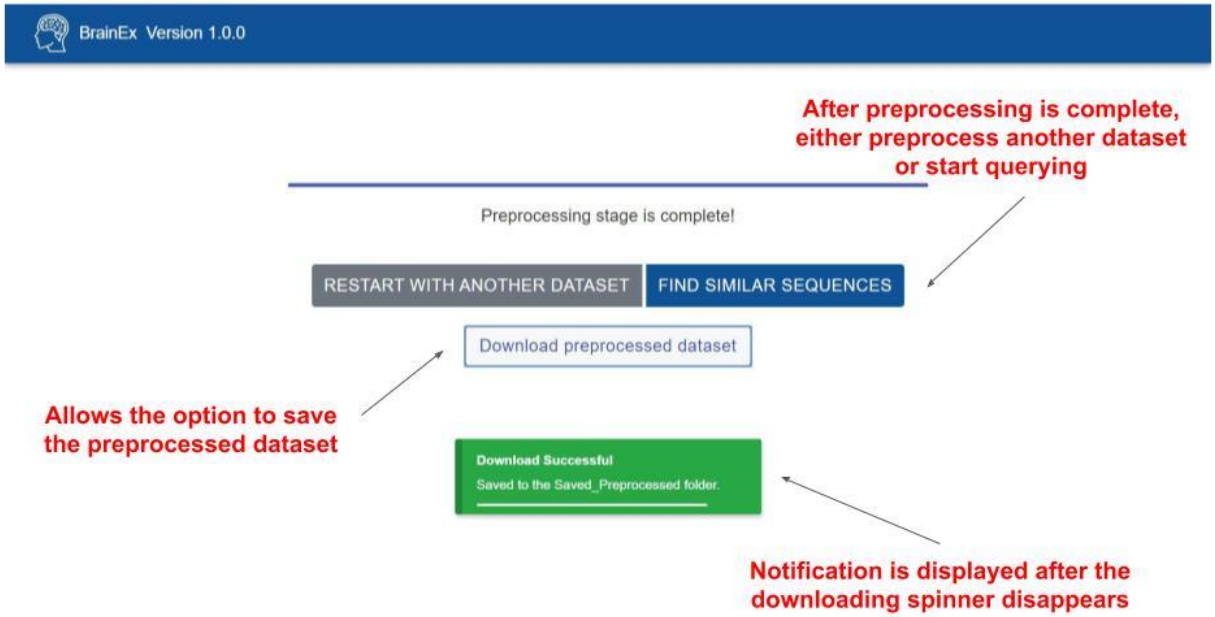


Figure 51: Build/Preprocessing Progress Screen of BrainEx UI, Complete

The user also has the option to first “Download Preprocessed Dataset” before taking any of the aforementioned actions. After the spinner icon stops spinning, a notification popup will appear at the bottom of the page, showing that the downloaded preprocessed dataset is saved as a zipped file in the folder *Saved_Preprocessed* by default. If the user decides not to save the dataset, they can proceed straight to the “Find Best Matches” dashboard.

On the other hand, If the user already has a preprocessed dataset, they can upload it on the homepage via the “Choose Files” and “Add” buttons to upload a preprocessed dataset to the server.

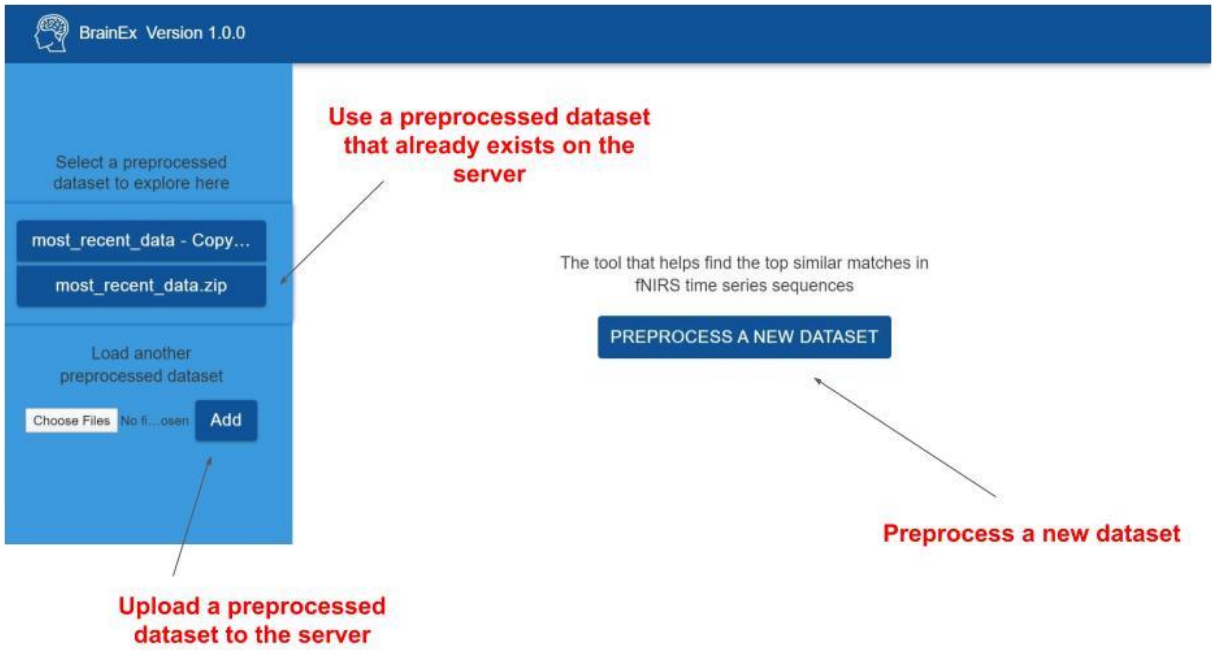


Figure 52: Home Page of BrainEx UI

This will open up a popup window, asking users to confirm their query options. Once confirmed, the user is going to be directed straight to the “Find Best Matches” dashboard.

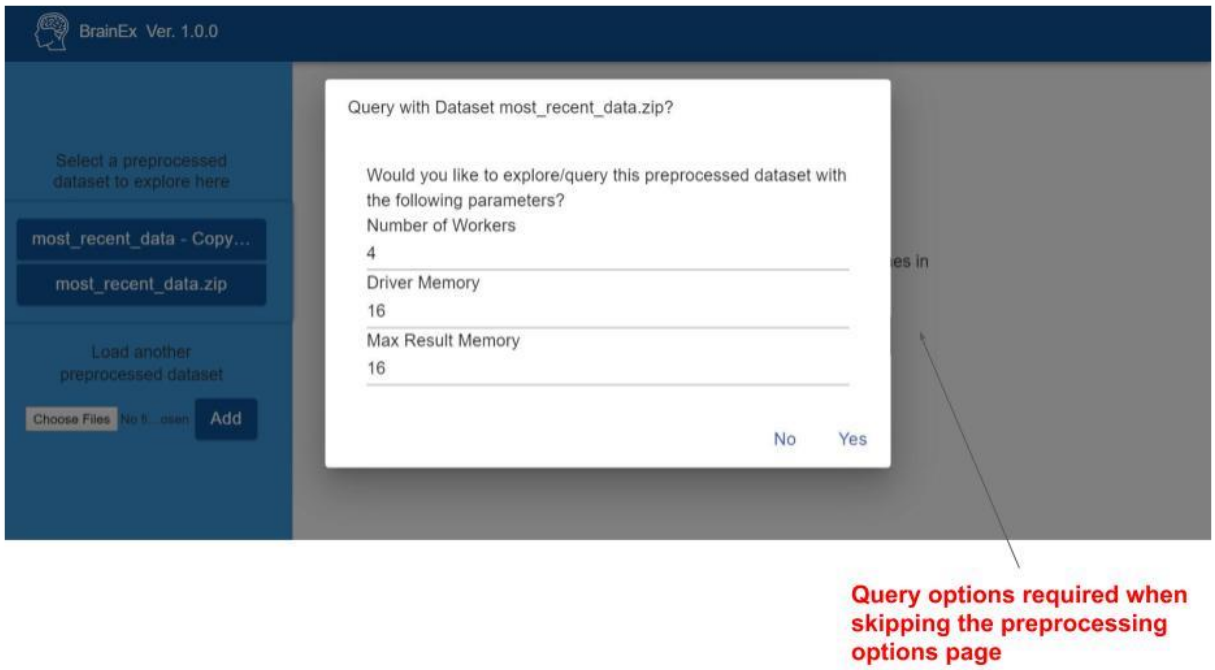


Figure 53: Home Page of BrainEx UI with Query Popup

On the "Find Best Matches" dashboard, the user needs to first upload a query in "Query Sequence" on the top left corner in order to find any similar sequences. Then, they should fill out the "Query Options" form below the "Query Sequence" section of the screen. The default values have been filled in so the user can directly click on "Start Query" if no parameter change is required. A spinner icon should appear on the right side of the page, indicating that querying is in progress, and the "Query Results" plot as well as "Ranked Sequence Matches" table are not ready for viewing yet. Once the spinner icon disappears from the page, the user should be able to view the "Query Results" sequence plot and "Ranked Sequence Matches" table.

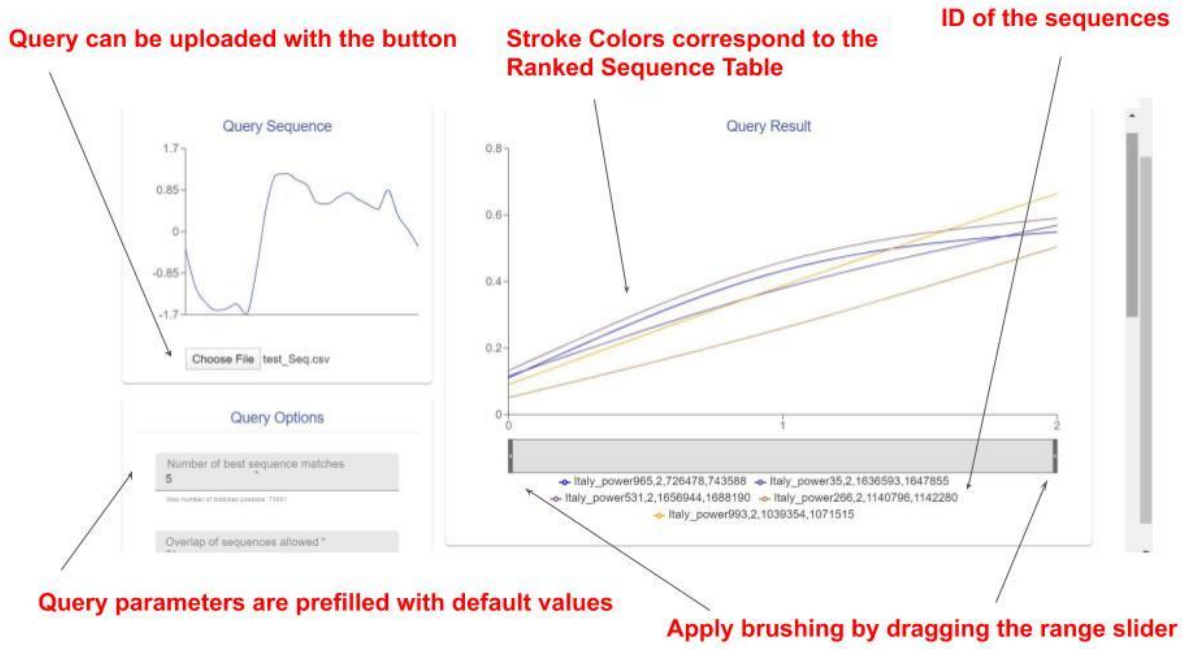


Figure 54: Query Dashboard

The user can toggle the show/hide column in the table on individual rows, as well as select/deselect all the rows. The visualization should be live updated as the checkbox fields change in the "Ranked Sequence Matches" table. The "Query Results" plot also enables brushing so lines can be zoomed in by using the range slider at the bottom of the "Query Results" plot. The user can also view the statistics on the bottom left corner of the dashboard.

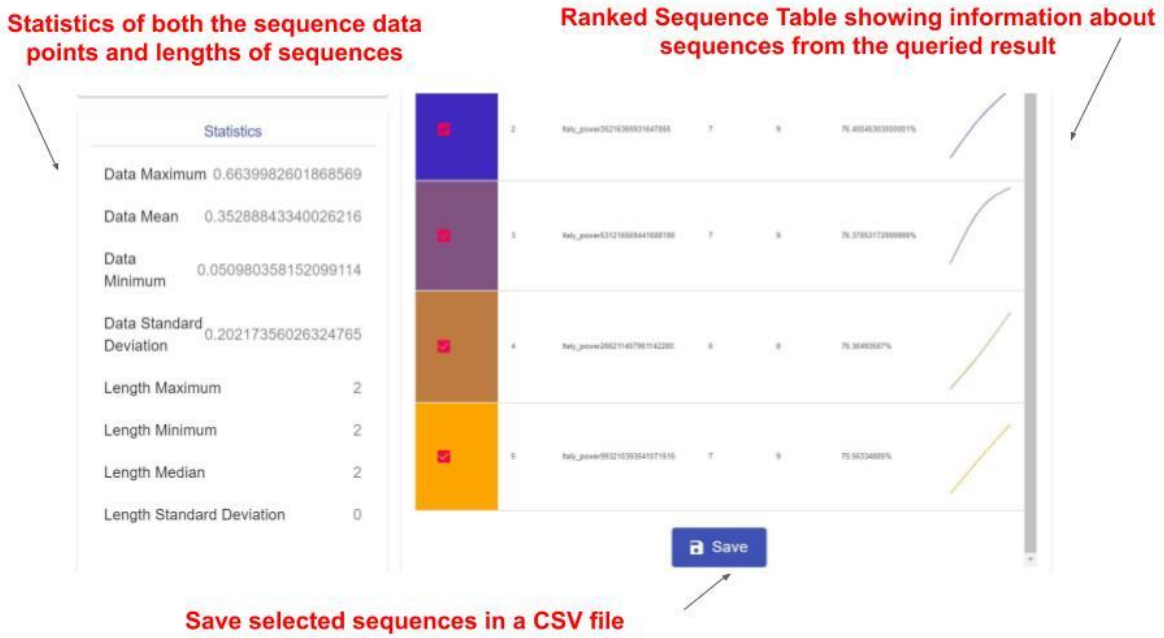


Figure 55: Query Results

If the user decides to switch to query with a different set of parameters or a different query sequence, they can always update their selections on the application by uploading a new query in the “Query Sequence” window or modifying the query parameters in the “Query Options” form.

For more in-depth information about our final product, please see our final README guide in Appendix R.

9. Discussion

While we are confident in our final product, we did of course come across some obstacles that affected our development. In addition, we have some suggestions for future improvements to the application and other features that can be added to the application to improve users' experience.

9.1 Obstacles and Limitations

In completing this project, we came across some obstacles. Many of these obstacles came from the technology that we utilized to make this project. In order to get the most robust frontend possible, we used React for JavaScript. However, the package we were given for the backend of our application was written in Python. While there are many successful applications that combine these two technologies, they are not made to interface easily. In addition, the backend for this application was actively developed while we were working on our project, so we had to make sure to adapt to the changes quickly so they would affect our project as little as possible.

9.1.1 Frontend Obstacles and Limitations

We started out developing the visualizations with D3.js, a JavaScript-based library for rendering visualizations using DOM elements as D3 is friendly with large datasets and supports intricate interactivity and animation. Using web standards such as SVG and canvas, D3 is capable of rendering creative and interactive interactions while maintaining extreme customizability. Implementing visualizations using D3.js later became one of the most challenging parts of development because both React and D3 wanted to control the DOM. Resources online on how to build React with D3 together, particularly for complex interactive charts, are seriously lacking. In our case, our dataset already required a lot of parsing, as well as dynamic population, which made rendering the chart in D3 more difficult given the dataset had an atypical shape. Furthermore, mapping individual colors to each line also created technical challenges. D3 requires data to be loaded at first, and then bound with the created SVG elements. This created conflicts with React's lifecycle; furthermore, it caused any changes, i.e, color mapping in our case, that must take place on the DOM hard to be made seamless.

In the end, we decided to switch to using Recharts, a D3-based library that does not collide with the use of React lifecycles. Although Recharts renders the visualization slightly slower than D3 does, we believe this is sufficient for our use case, which for the most part, is displaying only selected matching sequences. We expect that users would not attempt to view a very large number of matching sequences at a time, so Recharts is able to achieve the effect that we desire for now. However, if a dataset explorer page were to be developed next switching to another library such as D3 should be considered as D3 is better able to handle larger datasets.

9.1.2 Python Server Limitations

As all data sent between the frontend and backend were served using HTTP using Flask in the Python code and Axios in the JavaScript code, there were some transmission limitations. For example, in loading the files that are currently uploaded from the backend, we realized that we could not load the contents of every file to the frontend, because it was hard to get Flask to handle multiple files being uploaded at once. However, since the file is never processed on the frontend (it is just sent back to the backend when it is selected), we realized that the only information that needed to be passed was the filename so users could recognize and select it. The backend could then load that file for processing by name.

Another major limitation is that the backend includes a loading bar for preprocessing a dataset using Spark, but there is no way to transmit that information to the frontend since a RESTful API doesn't allow for multiple responses from one request. Therefore, we implemented an indeterminate loading bar for preprocessing so that the user can be sure that preprocessing is still happening. The loading bar begins to display on the frontend when the request is sent and stops moving when a response is received.

9.1.3 Backend Obstacles and Limitations

It was helpful in many aspects to be developing a frontend for a backend that was in active development and testing. If we were having issues with any of the code, it was easy to contact the developers. They could explain how we were calling functions incorrectly or even implement new features (i.e. preprocessing using Python native multiprocessing wasn't available until after we told the developer the issues we were having with our own Spark installations). However, while the application was frozen for most of our development, there was some refactoring between our prototyping and our implementation that led to some features we were expecting to exist to not be implemented. Sometimes, we were able to write our own code to work around this, such as uploading a query sequence. In other cases, such as lengths of interest for querying and preprocessing, we had to remove functionalities designed during the prototyping stage to better align with the backend code.

9.2 Areas of Improvement

In every project, there are always improvements to be made and this project is no exception. While the BrainEx application is attractive, intuitive, and usable, we suggest multiple improvements both to its aesthetics and functionalities.

9.2.1 Design

Because we used Material UI (MUI) as the main framework, modifying the stylings of the component proved challenging. We relied on the "!important" tag at several places on the CSS stylesheets in order to override MUI's defaults, which is not always the best practice in CSS. The best way to style MUI components is to use them within a functional component rather than a class component, but in some cases it was necessary to convert components from functional to class in order for passing props to work. Due to limitations with Material UI, we were unable to make the UI as responsive as possible in the given amount of time.

There are also some design choices that, if time permitted, we would have refactored. For example, on the Home page the preprocessed files are located on the left and a button to go to the "Select a new dataset" page is on the right. The "Select a new dataset" page looks nearly identical to the previous page. This is not ideal because the user believes that they are still on the same page or do not clearly see the difference between them. One solution to this, we believe, is to host both contents on the same page and simply have tabs labeled something along the lines of "preprocessed" and "raw".

9.2.2 System Status Feedback

In terms of system feedback to the user, we were unable to implement all that we wanted to due to time constraints and other setbacks. While we were able to apply the most prominent and useful after hearing back from users, not all possible errors have a corresponding client-facing notice. For example, if a file is not found, there is no notification displayed to the user. Also, when an action is successful (e.g. file is successfully uploaded), there is no notification sent to the user. Ideally, we would want any action made by the user to have some sort of feedback whether in the form of a notification or some other visual indication.

9.2.3 Query Sequence Upload Mechanism

Currently, the mechanism for selecting a sequence to query is not incredibly intuitive. In order to find similar sequences, the user must copy an entire row from the same CSV or preprocessed dataset they are looking at into a separate CSV. It must be a whole row since it requires all the same number of features as the dataset and the Python server is only expecting one row. If there were not the same number of features, then the features could not be converted to an identifier for the sequence, which allows for the "Exclude subsequence matches from current sequence" attribute to be applied when querying.

One way to improve this mechanism would be to allow the user to query with sequences from other datasets. The user should specify then whether the sequence was from the same dataset

as is being queried from. This will allow the system to determine whether the “exclude same sequence” is relevant.

Another way to improve this mechanism would be to allow users to select the data sequence graphically, since this is easier to understand than copying and pasting. This would be possible if the dataset and/or cluster explorers were implemented.

9.2.4 Error Handling/Prevention Methods

Although we tried to add as many error prevention/handling functionalities as possible, there exist some known methods that we did not implement due to time constraint. Namely, the “Start Query” button on the “Find Best Matches” dashboard should be disabled before a query is uploaded.

In regard to the parameters filled in the “Query Options” form, the frontend is able to display the maximum number of matches available right below the input field, and the text field of number of matches will turn red if the input number is greater than the maximum. However, there is nothing against inputting a number that is greater than the maximum number of matches available; the user is still able to submit the form which can trigger errors.

Additionally, users should avoid refreshing the page as much as possible. If they start preprocessing after refreshing the preprocessing options page, the form data will get cleared and throw an error. Similar data saved in the forms can be cleared after refreshing the page because the application is using web services to pass data back and forth.

9.3 Future Work

Finally, there are some extensions to this project that were either not implemented due to time constraints or being out of the scope of our project that we would like to propose to future developers in the WPI HCI lab.

9.3.1 Implementation of Data Explorer Pages

In the final prototype, we have designed the pages for “Dataset Explorer” as well as “Cluster Explorer.” However, both pages were not implemented at the time of this report due to time constraints. Although the final implemented application could handle the baseline functionality of finding similar sequences by querying, users are currently required to select and save a query outside of the application in order to find similar sequence matches. If given more time, developing the Dataset Explorer would be a priority as it could enable users to save a query so they could upload it on the “Find Best Matches” dashboard. Moreover, the implementation of

these pages would allow users to stay within the application after selecting a sequence as they could pass their desired sequence from an explorer to the querying page.

9.3.2 Performance Improvement

The application could become slower when the server has been running for a long time, because memory is not freed during this process. Although the preprocessing processes are shut down properly after each preprocessing cancellation/completion step, inevitably the application can lag over time. A temporary fix can be restarting the server, but implementing some kind of performance improving methods would be desirable in the long run.

9.3.3 Hosting on the Cloud

While the application is currently hosted on the user's local machine, the application has been built with the idea that it would be hosted on a cloud server in the future.

This could introduce several issues: for example, Spark can only allow one session running at a time. Data security would also become an issue as each user must have their own account in order to authenticate and keep their own data safe on the cloud.

However, there are also numerous benefits to hosting an application like BrainEx on the cloud. It will allow for users to be able to access the application and their data from any machine, which is often important for lab researchers as their equipment may change from location to location. In addition, it would allow for larger amounts of data to be stored and used with the application than are being used currently. Depending on the server hardware that is being used, the speed of the application could also be increased severely. Due to a combination of these factors, it is in fact, imperative that the application eventually be served from the cloud as the size of datasets and magnitude of research increase. We would highly recommend implementing some sort of database structure if BrainEx were to become a cloud application as well, since all data is currently saved in a simple file structure.

9.3.4 Updates to BrainEx API

At the time of our development, the backend developer of the BrainEx API refactored the query parameters. Therefore, lengths of interest, which was an old feature from the pre-refactored Genex API are not included in our "Query Options" form. If further updates to the Genex API are made available then an updated "genex" folder must be added to the BrainEx to replace the old version. The only required folder is the "genex" folder within the "Genex" repository. Currently, the best way to do this is to clone only the release branch of the Genex project and paste the "genex" folder into the BrainEx root project directory.

10. Conclusion

In completing this project, the team has completed a major learning experience and created a tool that could be used and expanded upon in the WPI HCI Lab for the foreseeable future. Throughout our project, we have learned many great ideas and techniques for improving user-experience. We were able to apply most, if not all, of these ideas to our application to ensure that this is a tool users could actually integrate into their daily research tasks. In addition to our thorough user documentation, our work on preserving this knowledge in our prototypes and paper ensure that future developers will be able to continue to use our philosophies.

Bibliography

7 Gestalt Principles of Visual Perception. (2019, April 10). Retrieved October 6, 2019, from UserTesting Blog website: https://www.usertesting.com/blog/gestalt-principles/

AdobeXD. (n.d.). Retrieved December 11, 2019, from https://www.adobe.com/products/xd.html?sdid=12B9F15S&mv=Search&ef_id=CjwKCAiAxMLvBRBNEiwAKhr-nE3ARjUlxfF8ZGRnj1zwz7EjkKjTvrae80Yw101LEC2eRYDvnxgwhoCqn0QAvD_BwE:G:s&s_kwcid=AL!3085!3!315233774139!e!!g!!adobexdadobexdadobexd.

Affairs, A. S. for P. (2013, October 9). Reporting Usability Test Results. Retrieved November 23, 2019, from /how-to-and-tools/methods/reporting-usability-test-results.html

Algrim. (2019). How HCI (Human-Computer Interaction) Has Evolved Alongside Technology. Retrieved September 30, 2019, from Algrim.co website: https://www.algrim.co/posts/51-how-hci-human-computer-interaction-has-evolved-alongside-technology

AngularJS. (n.d.). Retrieved December 11, 2019, from https://angularjs.org/.

Axios. (2018, April 4). Retrieved December 11, 2019, from https://flaviocopes.com/axios/.

Balsamiq. (n.d.). Retrieved December 11, 2019, from https://balsamiq.com/company

Bootstrap. (n.d.). Retrieved December 11, 2019, from https://getbootstrap.com/.

Bostock, M. D3.js. Retrieved December 11, 2019, from https://d3js.org/.

Brockington, G., Balardin, J. B., Morais, G. A. Z., Malheiros, A., Lent, R., Moura, L. M., & Sato, J. R. (2018). From the Laboratory to the Classroom: The Potential of Functional Near-Infrared Spectroscopy in Educational Neuroscience. *Frontiers in Psychology*, *9*. https://doi.org/10.3389/fpsyg.2018.01840

Carroll, J. M. (2012, October 12). Human Computer Interaction—Brief intro. In *The Encyclopedia of Human-Computer Interaction* (2nd ed.). Retrieved from https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro

CSS: Cascading Style Sheets. (n.d.). Retrieved December 11, 2019, from https://developer.mozilla.org/en-US/docs/Web/CSS.

Dubey, J., Sumaria, M., Oktay, E., Li, Y., Li, Z., Neamtu, R., & Solovey, E. (2019). Towards neuroadaptive technology using time warped distances for similarity exploration of brain data. *Proc. of Neuroadaptive Technology Conference*. Presented at the NAT 2019, Liverpool UK.

Express. (n.d.). Retrieved December 11, 2019, from https://expressjs.com/.

FNIRS Analysis. (2019). Retrieved September 29, 2019, from NIRx Medical Technologies website: https://nirx.net/fnirs-analysis

Grohol, J. M., & read, P. D. L. updated: 19 J. 2019~ L. than a minute. (2017, May 17). What is Functional Near-Infrared Spectroscopy? Retrieved November 23, 2019, from //psychcentral.com/lib/what-is-functional-optical-brain-imaging/

Guger, C., Allison, B. Z., & Mrachacz-Kersting, N. (2019). *Brain-Computer Interface Research: A State-of-the-Art Summary 7 BT - Brain-Computer Interface Research: A State-of-the-Art Summary 7* (C. Guger, N. Mrachacz-Kersting, & B. Z. Allison, Eds.). https://doi.org/10.1007/978-3-030-05668-1_1

Hamedani, M. (2019, January 14). React file upload: proper and easy way, with NodeJS! Retrieved December 11, 2019, from https://programmingwithmosh.com/javascript/react-file-upload-proper-server-side-nodejs-easy/

Homer-fNIRS. (n.d.). Homer2 Documentation. Retrieved March 5, 2020 from https://homer-fnirs.org/documentation/

How to improve your UX designs with Task Analysis | Interaction Design Foundation. (n.d.). Retrieved November 26, 2019, from https://www.interaction-design.org/literature/article/task-analysis-a-ux-designer-s-best-friend

HTML: Hypertext Markup Language. (n.d.). Retrieved December 11, 2019, from https://developer.mozilla.org/en-US/docs/Web/HTML

IDF. (2019a, March). A Brief History of Human-Computer Interaction. Retrieved September 30, 2019, from The Interaction Design Foundation website: https://www.interaction-design.org/literature/article/a-brief-history-of-human-computer-interaction

IDF. (2019b, September 1). Design iteration brings powerful results. So, do it again designer! Retrieved September 30, 2019, from The Interaction Design Foundation website: https://www.interaction-design.org/literature/article/design-iteration-brings-powerful-results-so-do-it-again-designer

Invision. (n.d.). Retrieved December 11, 2019, from https://www.invisionapp.com/about.

Jacob, Rob. (2018, January 6). Templates and Examples for User Interface Specifications. Retrieved October 6, 2019, from https://www.cs.tufts.edu/~jacob/171/templates.html

JavaScript. (n.d.). Retrieved December 11, 2019, from https://developer.mozilla.org/en-US/docs/Web/JavaScript.

NASA. (2019, November 22). Functional Near-Infrared Spectroscopy (fNIRS) Cognitive Brain Monitor. Retrieved November 23, 2019, from Nasa Technology Transfer Program website: https://technology.nasa.gov/patent/LEW-TOPS-84

Nielson, J. (1994, April 24). 10 Usability Heuristics for User Interface Design. Retrieved November 25, 2019, from Nielsen Norman Group website: https://www.nngroup.com/articles/ten-usability-heuristics/

Nielsen, Jakob. (2012, January 3). Retrieved October 6, 2019, from Nielsen Norman Group website: https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Niice. (n.d.). Retrieved December 11, 2019, from https://niice.co/.

NITRC. (2019). Homer2: Tool/Resource Info. Retrieved November 23, 2019, from https://www.nitrc.org/projects/homer2

Node.js. (n.d.). Retrieved December 11, 2019, from https://nodejs.org/en/about/.

ReactJS. (n.d.). Retrieved December 11, 2019, from https://reactjs.org/

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., & Keogh, E. (2012). Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 262–270. https://doi.org/10.1145/2339530.2339576

Schneiderman, B. (2013, September 12). Shneiderman's "Eight Golden Rules of Interface Design." Retrieved October 1, 2019, from Design Principles FTW website: https://www.designprinciplesftw.com/collections/shneidermans-eight-golden-rules-of-interface-design

Solovey, E. (2016, October). *CS338: Design Lifecycle Part 2*. PDF.

Solovey, E. (2019a, January). *Design & Think Alouds*. Google Slides.

Solovey, E. (2019b, January). *Prototyping*. PDF. Retrieved from https://drive.google.com/open?id=1RSC6FG32X596JTssdoHwiChlHuqHfsNm

Solovey, E. (2019c, February). *Ideation (2)*. Google Slides.

Solovey, E. (2019d, February). *Communication (1)*. Google Slides.

Solovey, E. (2019e, February). *Communication (2)*. Google Slides.

Solovey, E. (2019f, March). *Good Design*. Google Slides.

Solovey, E. (2019g, April). *Interface Evaluation*. Google Slides.

Tan, D., & Nijholt, A. (2010). Brain-Computer Interfaces and Human-Computer Interaction. In D. S. Tan & A. Nijholt (Eds.), *Brain-Computer Interfaces: Applying our Minds to Human-Computer Interaction* (pp. 3–19). https://doi.org/10.1007/978-1-84996-272-8_1

Usability. (2017a, April 3). User-Centered Design Basics. Retrieved September 30, 2019, from https://www.usability.gov/what-and-why/what-and-why/user-centered-design.html

Usability. (2017b, December 1). Benefits of User-Centered Design. Retrieved September 30, 2019, from https://www.usability.gov/what-and-why/benefits-of-ucd.html

Appendix A: Interview Preamble

Intro:

Good morning/afternoon; we are Vandana Anand, Kyra Bresnahan, Maggie Goodwin, and Yihan Lin. We are four seniors at WPI working on the development of a brain computing interface (BCI) for the functional near-infrared spectroscopy machine, also known as fNIRS. We are currently collecting qualitative data from users who have worked with fNIRS in the past, in order to identify potential areas of improvement in the user experience of existing BCI tools. Your input will be a valuable resource that can assist us in discovering the underlying user needs and future development opportunities.

Requirements:

This will only take about 30 minutes to an hour of your time and we will take notes along this process. Please understand that your participation is completely voluntary; you do NOT have to answer anything that you do not want to and the session can end at any time that you wish.

General Questions:

May we have your permission to record your audio for reviewing purposes?

May we have your permission to record your demo of using the tool for reviewing purposes? We will not be capturing your face during this process.

May we have your permission to quote this conversation in our final report?

If yes, may we have your permission to quote the conversation under your personal name? If not, you will be quoted anonymously to keep your information confidential.

Would you like a copy of our writing after we finish analyzing the qualitative data from this interview?

May we have your permission to take photos for our report?

Appendix B: Storyboard Evaluation Questions

Goal: Ensure the team's understanding of the backend and frontend requirements is accurate. Gather initial implementation and user requirements.

Procedure: Discussion format, gather a group of users and ask about implementing certain features. Trigger the group to talk about the pros and cons of how to implement said features.

Go to your target population.
Focus groups (small groups of people)
Present them with many alternate storyboards
Make ones you think they'll react strongly to
For each storyboard, ask a discussion question. Let the users talk, follow up.

Questions:
What did they like? Hate?
Did they have any strong reactions? Any surprising ones?
How does what they said indicate underlying needs they might have?
How does what they said indicate potential design opportunities?
Did they have any suggestions?

1. Is this feature useful? User inputs a graph and if that is found in the first channel but not the rest of the channel, we don't want to store this result. If this result is found in a majority of channels (# specified by user), we want to keep this information
2. Is this feature useful? Should we be able to identify 2 data shapes given by the user in the graph (not implemented in backend yet and can only search by 1 data shape currently)

3. Is this feature useful? Do we want to filter the correct, incorrect, and no response before building (before seeing the cluster explorer) to save more time or build everything in the beginning and then filter based on subject response in cluster explorer?

Appendix C: User Testing Protocol for Low-fidelity Prototype

Goal: Include user interaction and control flow to guide the user through the application and gather more information about the team's understanding of BrainEx and feedback on design decisions for the features.

Procedure: Check and observe whether the user is able to perform the following tasks when they are given the application and told to freely explore and figure out features of the application by themselves. Ask questions about user experience and usability at the end of the testing session. Specifically address if there is a problem, any recommendations for that problem, and assess if anyone else complained about this problem.

Tasks with necessary instructions:

1. Generate clusters from a new dataset
2. Upload a new dataset
3. Select a dataset and proceed to build the clusters
4. Pick the appropriate parameters and start clustering
5. NOTE: Loading page will automatically take you to the finished loading page once it is done (not a task)
6. Pick the page to view all the data
7. Filter the data based on the channel, subject, and label
8. Pick the region you want to view on the graph
9. Go to the cluster explorer
10. Filter the number of clusters to view
11. Filter the sequence length to view in each cluster
12. Click on the cluster and its representative to find it in the data visualizer
13. Click on next

14. NOTE: The cluster and its representative is still highlighted on this page. The clusters data points are listed in a table (not a task)
15. Filter the number of sequences in this cluster to view
16. Filter the sequence length to view in each cluster
17. Proceed to the Query finder
18. Pick use selection from cluster explorer
19. Enter the appropriate parameters
20. Show the query results
21. Pick the specific region to view in the data visualizer
22. Click on save file
23. Navigate back to the Home page

Questions:

1. Where do you think you would need more guidance in terms of using the feature? What seems unclear?
2. What parts of the app do you think would be useful and would definitely like to use?
3. On cluster explorer,
    a. How do we decide how many representatives to display to the user (maybe ask to input a number from 1-10) and on what basis (show representatives of the top 10 clusters with most data points)?
    b. If we have 30,000 clusters, how do we filter them down to approx 20? Should we ask for user input or another screen?
    c. Should number or length be displayed as the identifier in the view data shapes area?
4. On dataset explorer,
    a. How would you want to filter the data if there is so many to look at?
5. On query result page,
    a. What do you think is the most important information you would need to see on this page?
6. What statistics would you like us to see displayed?

Appendix D: User Testing Protocol for Mid-fidelity Prototype

Good morning/afternoon, we are Vandana Anand, Kyra Bresnahan, Maggie Goodwin, and Sylvia Lin. We are four seniors currently designing a user-interface for the time-series querying application, BrainEx. We are currently conducting user testing to receive feedback on the layout and basic functionality of the design. We will present you with our current iteration of the implemented front-end and several tasks to perform. Please speak your thoughts aloud whenever possible to provide any additional insight into your experience. Once you have completed all the tasks, there will be a series of reflective questions about the experience. You may opt-out of this test at any time. Your input will be a valuable resource that can assist us in improving the usability of our application.

Tasks:

1. For the already preprocessed dataset ‘SART2’, find a similar sequence by uploading a sequence file called ‘queryseq1.csv’
2. Preprocess a new dataset called “dataset2.csv” and explore that dataset while preprocessing. Then after preprocessing is done, select the sequence that has the subject ID of ‘101HART’ and use it to find similar sequences.
3. Explore data by clusters and, from the cluster that has 5 sequences, sort the sequences in the order of channel name. Then select the sequence that has the channel number 18JVO and save it

Questions:

4. Do you feel like the following aspects are lacking in the prototype: flexibility and efficiency of use(shortcuts), Error prevention and Error recovery?
5. How to best visualize data in dataset explorer?
6. How does the control flow feel to you?
7. Which version do you prefer between the two?

Appendix E: User Testing Protocol for High-fidelity Prototype

Good morning/afternoon, we are Vandana Anand, Kyra Bresnahan, Maggie Goodwin, and Sylvia Lin. We are four seniors currently designing a user-interface for the time-series querying application, BrainEx. We are currently conducting user testing to receive feedback on the layout and basic functionality of the design. We will present you with our current iteration of the implemented front-end and several tasks to perform. Please speak your thoughts aloud whenever possible to provide any additional insight into your experience. Once you have completed all the tasks, there will be a series of reflective questions about the experience. You may opt-out of this test at any time. Your input will be a valuable resource that can assist us in improving the usability of our application.

Tasks:

1. Preprocess a raw dataset named dataset_6
    a. Explore while preprocessing
    b. save selected subsequence
    c. Find similar subsequences by uploading a sequence
2. Explore clusters after preprocessing is completed

Questions:

3. On a scale of 1-5, please rate the learning curve of this tool, with 1 being extremely easy to learn and 5 being extremely difficult to comprehend.
4. On a scale of 1-5, please rate your satisfaction of this tool based on the overall UI look alone, with 1 being extremely consistent and usable and 5 being extremely difficult to use and sloppy.

Appendix F: Research Questions

1. What is our scope/what should we tackle? - BrainEx (identify problem)
    1. What are the lab's current tools and processes?
    2. What direction is the lab moving in?
    3. What technologies are being developed in the lab that require an interface?
    4. Why does a particular technology deserve our attention?
2. What would a user-interface for BrainEx have to do/look like? (requirements gathering)
    1. What have they used in the past? (old version of BrainEx)
    2. What do the developers of BrainEx need from an interface?
    3. What do the end-users of BrainEx need from an interface?
    4. What backend support does BrainEx offer?
    5. How does BrainEx work?
    6. User needs and wants from the lab staff
    7. Why would people use BrainEx?
3. How do we use HCI principles to design a UI for BrainEx? ("final" step/put it all together)
    1. What are the HCI principles? (refer to background)
    2. How do we ensure that we make a good design?
    3. What methods are we gonna use to design a prototype?

Appendix G: Interview Questions

1. How long have you been working with the fNIRS brain data and what do you use it for?
2. What is your typical workflow when using the fNIRS machine? Please include both customized tools that you/your lab may have built on your own as well as the pre-existing ones.
3. If you/your lab have built your own tools, why did you build it and what do you use it for?
4. If there is more than one, please let us know which tool you like the most and which tool you like the least. Why so?
5. How long have you been using and how often do you use this set of BCI tools?
6. Why did you/your lab choose/use these particular BCI tools?
7. Are there any aspects of the tools that you find frustrating? How so?
8. Have you ever felt at any point that certain functionalities you need are missing from this interface?
9. In the case of technical difficulties, how do you usually resolve them? Do you just google the problems you encounter? Search in their official documentation? Email the support team?
10. What is the most useful thing you found on this interface? Can you point to us things that you feel like you have never found a use of on this interface?
11. What do you find to be the most useful/helpful way of visualizing the fNIRS data?
12. Please rate your level of understanding of the tool(s) (1 being you have no idea and 7 being expert).
13. On a scale of 1 to 7 (1 being expert on first try and 7 being it took years to master), please rate the learning curve of the tool(s).
14. How long did it take you to get acclimated to them? What level of technical expertise do you think a person would need to use this tool?
15. Please take some time to envision the most ideal workflow of a BCI tool, from your personal point of view. Feel free to draw a diagram to elaborate.

16. Where do you usually save the data after it's collected/processed? Any issues with data loss?

Appendix H: User Personas

*Table X***:** *Persona Information (Undergraduates)*

Persona Result 1	
Name	Joe
Education	Freshman majoring in Biology
Titles & Responsibilities	Student worker in the lab training to facilitate fNIRS based research and use various BCI tools for data collection.
Goals & Frustrations	As a new student who just joined the lab, Joe feels that it is easy to pick up the procedures and workflow in the lab tools with some guidance. However, it is hard for him to learn specific features in the data collection tools needed to conduct fNIRS research without a prior background. He feels that there is a lack of documentation and is very difficult to troubleshoot technical issues himself.
Narrative	Joe is a freshman at WPI majoring in Biology. He is spending his first semester working in the HCI lab, primarily to conduct experiments with users and collect fNIRS data.
Quote	“It’s important to have a tool that is easy to learn and does not require any technical expertise”

Persona Result 2	
Name	Sheila
Education	Junior majoring in CS
Titles & Responsibilities	Student worker in the lab performing fNIRS research by using data collection tools and performing minimal data preprocessing.
Goals & Frustrations	Overall, Sheila feels that the tools she has used are straightforward and easy to learn. She especially likes the fact that there is no technical experience needed to begin using them. However, she feels that it would be ideal to have one UI to follow all the steps of the workflow because it gets confusing to navigate through multiple pages. The data collection tool has some useful data processing functions as well as the capability to remove discontinuities and pieces of data. The visualizations are very

	useful since they are customizable, but it would be ideal to limit the number of graphs to make it easier to use and visualize. Moreover, the data collection tools are not customizable itself, so Sheila is required to follow a specific procedure in the application and cannot adapt the tool based on her needs. In addition, it would be more convenient to run multiple applications on one machine rather than several. Also, all the tools lack documentation for technical issues. Because of this, Sheila finds that many features are still unknown and have never been explored. The tools were a little confusing to learn initially because there are no tooltips to guide her within the application. In terms of solving technical difficulties, error cases are hard to troubleshoot and the system freezes if a data stream cannot be detected.
Narrative	Julia is a junior at WPI double majoring in CS and BME. They have spent the summer of 2019 working in the HCI lab, primarily to conduct experiments with users to collect fNIRS data .
Quote	“It’s important to have a tool that is easy to learn, straightforward, and has meaningful visuals.”

*Table X***:** *Persona Information (Graduates)*

Persona Result 3	
Name	Julia
Education	PhD student with a concentration in Bioinformatics
Titles & Responsibilities	Works in the lab with fNIRS brain data and conducts a cognitive study on learning in the prefrontal cortex using AX-CPT to detect different cognitive states in users. They also work on signal processing and developing algorithms to perform time series data analysis. They perform data collection and analysis with Aurora, RTFD, NirsLAB, MATLAB GUI as well as Homer and have a vast amount of expertise in these tools.
Goals & Frustrations	Overall, Julia feels that the data collection and data analysis tools are very available and widely used. They are easy to learn for the most part, except for one of the data analysis tools, as there is no coding or technical knowledge needed. In addition, she thinks the visualizations of the data are helpful because there are line graphs clearly showing the spikes and areas that need further investigation. The plotting and mapping functions for data analysis are visually appealing such as the color coordinated graphs and there is the ability to easily mark specific points on the chart as well as

	remove noise. However, Julia feels that it would be helpful to view only relevant (less than 4) charts at once instead of all together to increase usability and decrease overcrowdedness of the features. Also, the data analysis tools have one interface that performs all the functionalities, but Julia feels that this is not efficient because it slows down her productivity in the application. Instead, it would be more useful to break it up into different stages and ensure that the transition is smooth so that processing data can be streamlined. Another concern she had are that the tools are not open source, thus not customizable, and only function when following a specific procedure in the application. This makes it hard for Julia to adapt the tool based on her needs and the data. She also feels that the tools are not very intuitive, have bad documentation as it is poor or doesn't exist, and contain lots of pre-processing steps. A lot of her time is wasted by trying to learn and understand what a feature does or emailing the companies that make the tool to solve technical issues. She believes it would be useful to have guidance within the application for the workflow and specific features in the form of question mark icons. Especially for the data analysis tools, Julia felt she was less experienced with using them after taking a break from using the tool. Ideally, Julia thinks an application would be more usable, navigable, and understandable in a user's point of view if dialog boxes are shown for error handling when a user performs a wrong action on the UI, more information, such as markers, are shown on graphs, after the data collection step the data is exported in the format the user wants, such as .csv, to go into a data analysis software for pattern/anomaly detection, machine learning is used to tell users to look at an area after the data preprocessing step and these regions are highlighted for the user to investigate more closely for analysis, and the data collection as well as the data processing steps are separate from each other.
Narrative	Julia went to school for Computational Biology & Bioinformatics as well as CS and is completing their PhD at WPI with Professor Solovey. She has spent a year and a half working in the HCI lab, collecting and primarily analyzing fNIRS data by performing experiments with users.
Quote	"Clear, concise, and easy to access documentation as well as a clean, navigable website without overcrowding features are very important to aid in a user's ability to immediately understand an application"

*Table X***:** *Stakeholder Persona Information (Developers)*

Stakeholder Result

Name	Troy
Education	PhD with a concentration in CS
Titles & Responsibilities	Working in the lab to develop an improved backend, including the database and API, for BrainEx and improve functionalities in processing brain data while also designing the architecture of projects, running experiments, and testing BCI tools.
Goals & Frustrations	Troy explained that BrainEx was an old data analysis tool to perform analysis on brain data coming from a user during an experiment. The backend is implemented using PySpark and the UI is implemented with fairly new technologies including React, JavaScript, and Redux. He feels that BrainEx is easy to use for someone who has experience and understands the features, but not for a HCI-oriented professional who is just using the tool for example. He expressed that the new UI should be made easier to use because people who did not come from a technical background tend to have a bit of a learning curve using BrainEx without the documentation. In addition, the biggest problems with BrainEx was that it was taking a very long time to run and had many missing features, which is why a new version would be created to utilize distributed computing for better efficiency and contain additional functionality.
Narrative	Troy is completing his PhD at WPI with Professor Solovey. He has been researching and working in the lab on many BCI related projects, specifically on the project in regards to implementing an improved backend for BrainEx.
Quote	“Understanding the functions behind features of a UI can help designers and developers improve the usability and intuitiveness of the UI for the user” “Gathering and incorporating the user requirements is an essential part of delivering a viable UI” “Extensive documentation should be given for any UI so that any user, both technical and non-technical, can quickly become acclimated to it”

Appendix I: Usability Aspect Reports for Low-fidelity Prototype

Usability Aspect Report	
Subject ID	1, 2
Name	Confusion about where data was coming from in different pages of the application
Evidence	"Where do the csv files on the left side of the window come from? Are they from the local machine, server, application, etc...?" "On dataset explorer -- what is this table? Is it csv file content? If so, which one is selected?" "What does filter mean -- what is it showing in the visualizer on dataset explorer?"
Explanation	1) The user was confused as to where the csv files displayed on the left side of the homepage screen, labeled as SART1,2,3 datasets, had actually come from. They were not sure if they were already uploaded and did not click on them. 2) The user was confused as to where the data viewer contents on the dataset explorer page was coming from and did not know which sequence was selected in order to investigate further. 3) The user was not sure what the filter option on the dataset explorer page and what it would be showing in the data visualizer graph.

Severity	3, frequent as user encountered this problem several times, persistent because it is important to know where the files and table data come from in order to understand the application, high impact because it made the user confused about features of the application.
Solution (optional)	Improve the wording for the uploading files feature to make it more intuitive The user would rather see a blank screen when he/she first arrives to the page so he/she knows that nothing is currently selected. And when it is selected and being displayed, indicate which one it is. Start with nothing in the preview window in selection of csv files.
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 2
Name	The options the user has to pick are not clear
Evidence	“What are the best choices for the build options screen -- is the one already selected in similarity threshold the recommended default?” “What does ‘k best matches’ mean?”

	"Is altering this slider altering the length of clusters displayed or length of sequences within cluster displayed?"
Explanation	1) The user was confused as to whether the default options on the build screen, such as similarity threshold, distance type, and length of interest, are the recommended default or not 2) The user did not understand at first what "k best matches" referred to. He/she questioned if it referred to the shape, length, or channel. 3) On the cluster explorer screen, the "range of sequence length" option is confusing. All sequences within a cluster being the same length doesn't make sense to the user. This makes he/she think of individual sequences and not the whole clusters.
Severity	3, very frequent as the problem occurred many times, somewhat persistent because was with multiple options on different screens, medium impact as it affects the user's ability to select options in order to proceed
Solution (optional)	Have a help button on the side of each option to explain any confusion the user may have Label or have a description of the "k best matches" option
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 2
Name	Before making a selection to view data, the display pages were confusing
Evidence	“I am not sure why data is shown even though I haven’t selected anything.”
Explanation	The user was confused when he/she navigated to the CSV file viewer page and there was already a dataset displayed before he/she clicked on anything so it was not clear if it was already selected to be processed. In addition, when the user went to the explorer pages, there was data displayed before he/she selected or filtered any data.
Severity	3, it was a frequent problem with the user, not very persistent as this is a preference, medium impact because it can affect user experience
Solution (optional)	Current selection box should not have a selection shown if user has not selected anything Don’t display results in query finder unless there is something selected/queried
Relationships (optional)	N/A

Usability Aspect Report

Subject ID	1, 2
Name	Graphs and visuals were not intuitive
Evidence	“The colors and shapes are not consistent when I’m navigating to another page”
Explanation	The user though the graphs were not intuitive as it looked like the selected cluster was just added into the data visualizer and the other lines are not similar in the graph. Moreover, some lines in the cluster explorer graph were similar to other shapes on other graphs in different screens. It was not clear that the application was highlighting the shape he/she had selected.
Severity	2.5, frequent as user encountered the problem multiple times, persistent as it should be applied on all screens that have graphs, low impact because it helps to perfect the prototypes
Solution (optional)	The graphs should reflect what the user is actually selecting. The colors of the same shapes should be consistent. When showing users a chart of visual, make sure it is worded so that a non technical user can understand
Relationships (optional)	N/A

Usability Aspect Report

Subject ID	1, 2
Name	There are naming inconsistencies within the application
Evidence	“The number of clusters does not equal the range of the clusters”
Explanation	1) On cluster explorer, the cluster size range is not meant to show how many clusters to show. The user cannot change the length of a cluster as that is a data point, so saying that the user is selecting the range of the sequence length for all clusters would more clear. In addition, the wording for “Previously clustered datasets” and “CSV files” are changed between the homepage and the dataset explorer. Also, when the user goes to upload a file and return to the dataset explorer, the names of the datasets on the left change from “SART#” to “Dataset#”.
Severity	1.5, frequent as the problem occurred a couple of times, persistent because user is confused about the functionality, low impact because it is a preference
Solution (optional)	Keep names and screens more consistent or make them more clear as to what they are and where they come from on each screen.
Relationships (optional)	N/A

Usability Aspect Report

Subject ID	1, 2
Name	The control flow of the application is not clear
Evidence	“The flow and transitions of the application is not very clear.”
Explanation	1) The dataset explorer should not come after cluster explorer and the user would like to view what the data looks like before clustering the data. Also, the user thought the workflow was over at cluster explorer and was confused as to why he/she would want to have another query if he/she already had one. 2) The user did not know where to click or how to start in the query finder page. For example, it is not clear that entering parameters is the next step. 3) The user ended up circling between the dataset explorer, cluster explorer, and query finder which did not make sense so he/she went back to the homepage without meaning to go there. The transitions between screens eventually became clear, but were not intuitive in the beginning. 4) The tabs, that are intended to help the user navigate between the 3 screen mentioned above, do not make the control flow clear to the user. Buttons would be better in this case.
Severity	4, very frequent as it happened many times, persistent as it is important to have a control flow that is understandable and doesn’t deviate the user from the task, high impact as it could highly affect a user’s

	ability to perform functionalities in the application.
Solution (optional)	The exact flow of the application should be made clear to the user. Make sure to consider all the paths a user could take in the site. Automatically go to the next screen when the build process finishes.
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 2
Name	Guidance in certain places within the application is needed
Evidence	Facts
Explanation	1) The user was unsure where the number 200 came from on the loading clusters screen when it says the sequences have been processed 2) The user wanted more guidance for the build and clustering options, such as length of interest and what it means. 3) The user wanted to make more clear in the application that clustering is done on all the channels 4) The user wanted a more clear description of the purpose of each screen, such as dataset or cluster explorer.

Severity	2.5, frequent as the concern was brought up several times when testing the application, persistent, because the user was confused about what some features were doing which hindered their progress in achieving the task at hand, medium impact as guidance and documentation has been an overall concern in the lab.
Solution (optional)	Show what the numbers in the application mean Highlight the purpose of each option, especially in the build options screen, in terms that the user can understand and get rid of the default values
Relationships (optional)	N/A

Appendix J: Usability Aspect Reports for Mid-fidelity Prototype

Usability Aspect Report	
Subject ID	1,3,2,4,5
Name	Confusion with the control flow when navigating between pages
Evidence	Went to find similar sequence page and asked "didn't I already upload a file?" Clicked on Dataset explorer on find best sequences screen and said "they have no idea what they're doing" Clicked on find similar sequences on dataset explorer and asked "why am I back to this [find similar sequences] screen"
Explanation	1) The user kept going back and forth between the dataset and cluster explorer pages as well as the find similar sequences page. They were confused about the application and why they are getting back to the same page over again. After a lot of trial and error in navigating between pages, the user got to the find similar sequences page. For example, the user accidently went back to load dataset when he/she didn't necessarily mean to and lost pre-existing dataset he/she was working with 2) The user wasn't sure how to select a sequence and subsequently find the similar sequence on the find similar sequences page. Took the user a bit to find the "upload sequence file" button. The user thought there

	would be a filter bar to filter the data but didn't realize that clicking on the filter selection page or option was they way to start the process 3) Dataset explorer by filtering was hard to understand since the user remembered seeing it in the data viewer, which isn't filtering. Also have a back button to go back to the homepage on the dataset explorer screen. Right now the user is inclined to go back to the dialogue box screen with the progress bar. 4)The user was confused when clicking "upload" and the "next" button on uploading a new dataset in dataset viewer and then wasn't sure what next step was 5)The user was confused as to why he/she is exploring data while preprocessing happens and didn't know what exactly he/she was exploring.The user didn't know what preprocessing was doing and wasn't sure how to go back and check the progress of preprocessing right away
Severity	3.5, frequent because this problem occurred multiple times, persistent because control flow is very important as that is the way users can transition from screens to achieve their goal in the app, medium impact because it made the user very confused.
Solution (optional)	Improve the control flow and make it more intuitive, trade off is having enough time to do this It should be more direct to find a sequence from the query screen And it would be more helpful to have more ways to navigate.

Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 3, 4, 5
Name	Aesthetics and layout of the screens can be more appealing and consistent
Evidence	“I wouldn’t have noticed that I can sort by channels on the data table” “The data can be more consistent across screens”
Explanation	1) The user is not sure why scroll bar doesn’t work 2) On the BrainEx homepage having the file selection on the home screen is misleading because users will think they can select it and preprocess it again. Make the preexisting file selection a dialogue window. 3) It is not clear how to sort by channels in the data tables 4) Didn’t notice the tabs at the bottom the first time while using it 5) The Hart 101 subject ID in the data table in the find similar sequences screen should have 100% similarity 6) Should not be able to change sequence length on the second cluster explorer screen. Grey it out or just get rid of it, which would be more intuitive

	7) Underline sequences to make it like a header and make all the blue outlined save buttons into black. 8) Looking at twenty clusters at once would be enough for the user to look at on the cluster explorer page 9) The user got confused and couldn't tell which cluster had 5 sequences right away
Severity	1.5, frequent as the concern came up many times to improve the screen, not very persistent or impactful as these are opinions and help make the application more pleasing to use
Solution (optional)	Move the tabs leading to the different explorer screens to the top of the application instead of having them at the bottom
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	2, 4, 5
Name	Naming and labels in the application can be more consistent
Evidence	"Would the name of the button be start preprocessing or just preprocess?"
Explanation	1) The user thought "preprocessing" meant cleaning the data, meaning he/she would not want to bother looking at it. Is the naming convention start processing or preprocessing?

	2) Instead of load dataset, Home would be more intuitive 3) Keep it consistent as on screen says “processing dataset4” when it is actually processing dataset2
Severity	1, frequent as the problem came up several times, not very persistent as it is a preference to heighten intuitiveness, low impact
Solution (optional)	
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 4
Name	Error prevention should be added so that the user doesn’t make costly mistakes within the application
Evidence	“It would be helpful to handle possible user errors in the application to avoid catastrophic accidents”
Explanation	1) Add a “Is this the selection you want” dialog box on find similar sequences page after clicking on “upload sequence file”. Also add “You’re about to erase your upload sequence, are you sure you want to go back?” on the find similar sequences page. When the user clicks on “select a sequence”, the order of the option change so it would be helpful to

	have a back button in case the user clicks on something wrong. 2) Anytime the user selects a sequence, cache the data and save it as a temp file even if he/she doesn't click on the "save selected sequence" button
Severity	2.5, frequent as it was brought up in a few areas of the application, persistent because it would be helpful to keep the user from crashing the application and provides better user experience, medium impact as it would prevent accidental errors
Solution (optional)	Add a pop up box to and have user confirm when switching screens
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 5
Name	The graphs and visuals can be more appealing and intuitive
Evidence	"What is the big white area for in the find similar sequences page?"
Explanation	1) The user feels it is a little confusing in the find similar sequence page what goes in the big white space meant for the graphs and data tables before selecting any data to find similar data matches. 2) The minimum number of sequences to be displayed in the graphs should be 5 and the maximum should be 10 or 15. Anything more

	than that and there would be no more space or colors. 3) Add a magnifying glass on the slider to make it more intuitive and make it more interactive 4) Use the subject ID in the legend for the graphs on all the screens
Severity	2, not very frequent as it was not brought up as much, persistent because the visuals and graphs can convey important information to the user, low impact as it is preferences
Solution (optional)	Could have an empty visual or table before the user enters any data For the magnifying glass, add a + on the magnifying glass to represent zooming in and one for - to represent zooming out.
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	5
Name	More guidance in the application is needed
Evidence	“It would be nice to have a tutorial in the application before having to use it”
Explanation	The user believes it would be more intuitive and useful to have a tutorial in the application itself to get more accustomed to the tool
Severity	1, not very frequent or persistent as this is a nice-to-have feature, low impact

Solution (optional)	Can have a short tutorial when the user first opens app
Relationships (optional)	N/A

Appendix K: Usability Aspect Reports for High-fidelity Prototype

Usability Aspect Report	
Subject ID	1, 4
Name	Confusion with the control flow when navigating between pages
Evidence	“It would be useful to have all buttons in the application clickable to simulate a real UI” “Clicking on the BrainEx logo should take me back to the homepage”
Explanation	1) The save subsequence button is not mapped to the correct screen 2) Allow the user to save before preprocessing is done! Also, the user should be able to save query results 3)For the cluster explorer screen, make the data table reflect the user’s selection of looking at the top 5 and bottom 5 clusters 4) Make the progress bar when preprocessing the data pop out as dialog box so the user can keep an eye on it

	5) Make the “return to homepage” button clickable. The user hopes in the implemented application that loading a dataset would take them back to the home screen. 6) Add ability to be able to click on the BrainEx logo to go back to the homepage 7) The user thought that after filtering, there would be a button to click to apply the filters to the data 8) The user had to click around a lot in the application in order to find functional buttons because not all buttons were functional on a given screen 9) Change the upload file dialog to upload, which is a copy of save right now, and add a sequence to be selected
Severity	2, frequent as the problem occurred often, but not very persistent as these are nice-to-have features, low impact
Solution (optional)	Make sure the save button is clickable in file explorer
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	3
Name	More guidance in the application is needed
Evidence	“What is the find similar sequences page, how did I get here?”

Explanation	1) User needed a little bit of guidance to explore the data during preprocessing 2) User was a little confused on the find similar sequences page and went back to the tabs
Severity	2, not very frequent, persistent as it is important for the application to be intuitive, medium impact
Solution (optional)	Add tooltips and clear descriptions
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 4
Name	Naming and labels in the application can be more consistent
Evidence	“The recent and server files are confusing, why are those there and have I used this before?”
Explanation	1) User had many questions about the “Recent” files and was confused. They were not sure if they or another person had used the dataset before 2) The event names, labeled as target correct and incorrect, are confusing to the user
Severity	1, not very frequent or persistent as these are nice-to-have features, low impact

Solution (optional)	Remove recent files and all server files
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1
Name	The graphs and visuals can be more intuitive
Evidence	“It makes sense for the legend to be located on the left of the graphs”
Explanation	1) The left side of the explorer screens is sufficient as a legend and it makes sense to have it on the left. The user believes the legend would apply to what is in the data viewer
Severity	1, not very frequent or persistent as this is a nice-to-have feature, low impact
Solution (optional)	Place legend on the left of the graphs on all the screens to make it consistent
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 4

Name	Understanding user input options could be more intuitive
Evidence	“What are dataset headers?”
Explanation	1) The user was confused as to what dataset headers mean and whether he/she needed to previously know how many headers were in the dataset or if he/she can limit the amount of headers 2) The user was confused as to why there is no recommended length. 3) User was confused as to what distance type is 4) The current selection on the find similar sequences page is confusing
Severity	2, frequent as the problem occurred more than once, persistent because it is important to make the application intuitive, medium impact
Solution (optional)	Add a recommended length default parameter and make sure to start at 1 and not 0
Relationships (optional)	N/A

Usability Aspect Report	
Subject ID	1, 4
Name	Aesthetics and layout of the screens can be more appealing and consistent
Evidence	“Aesthetics are very important to the appeal of the application”

Explanation	1) Buttons are a little big on CSV Data Viewer screen 2) Move question mark help button to the top right and move save away from the “select from upload” button 3) The user pointed out missing lines on some boxes 4) Adjust the legend size so that labels don’t get split into two lines 5) Orange and red colors have the same end time in the data 6) Goes from “current selection” to “current sequence from…”. The user says this is not needed and just should have the current selection 7) There should be better contrast and clearer distinction between close buttons 8) Data viewer contents should be adjusted eventually so that they match better with the overall aesthetic 9) Adjust screen size. Invision does fit to width but not fit to height. The user had to scroll to view the entire screen and this makes it harder to find and click things in the UI 10) The user didn’t understand that the save button was different from the “select from upload” button because they were right next to each other 11) Switch the cancel and explore locations button. The user hovered over cancel first and didn’t want to misclick
Severity	1, not very frequent or persistent as this is a nice-to-have feature, low impact
Solution (optional)	

Relationships (optional)	N/A

Appendix L: BrainEx User-Interface Tutorial

BrainEx is the tool that helps find the top similar matches in fNIRS time series data

- Fork the project on Github from vanand23's (Vandana) repository: https://github.com/vanand23/BrainEx-UI
- Clone or download the project from Github
- Open the project in the Pycharm IDE
- Open the terminal and run the command: **npm install** to install all the project modules and dependencies
- To run the application in Pycharm, open two terminals. One for running the frontend and one for running the backend. Make sure you are in the "brainex" folder (the command is: cd brainex)
    - Starting frontend command: **npm start**
        - After running the frontend, the message in the terminal will say "Compiled successfully. Server is now running on localhost:3000"
    - Starting backend command: **node server.js**

        - After running the backend, the message in the terminal will say “App is running on port 8000”
- Once the application automatically opens up on the website (“localhost:3000”), you will see the homepage of BrainEx
- To upload an already preprocessed dataset, click on “Choose File” and select multiple CSV files. You can also choose a single CSV file. Note that if you try to choose any other file type, it will not work and you will encounter an error.
    - You will see that the “No file chosen” will change to the name of the file you chose
- After choosing the file, click on “Add”
- Once you do this, go to the project directory and under the folder named “PreprocessedDataFiles”, you will see that the files that you chose from your local directory will be added to the server of the website
- To start preprocessing a new dataset, click on the button named “Preprocess a new dataset”
- You can then click on the navigation buttons in the rest of the screens labeled as “back”, “next”, etc to get from one screen to another

Project Directory

BrainEx-V1

- The “test_db” folder is the database for BrainEx
- “ItalyPower.csv” contains an example of the fNIRS time series sequences
- “Test.py” is the file to run through BrainEx as the command line tool. Use this script to copy and paste sections of the code into the python console and go through the database functionalities
- brainex - folder where all the frontend code is stored and the directory to be in in order to run the BrainEx application
    - “Node_modules” are the node modules and project dependencies that are used by the project. This is added by typing the command: npm install in the terminal
    - “PreprocessedDataFiles” is the folder that contains all the user file uploads from the homepage
    - The “public” folder contains index.html. Index.html links to index.js (and it’s stylesheet called index.css) which is linked to App.js
    - “Server.js” contains the express server code and the logic to do the file uploading functionality
    - “Src” is the main folder with all the ReactJs code

- App.js is the main file of the application that executes the homepage as well as all the routers and navigation pages in the site. App.css is this file's stylesheet
- The BrainEx image logo is located here
- The "components" folder has the rendering of all the other pages in the application. If a new page needs to be created, create and name the page in this folder, go to App.js, and create a Router link to the page you are creating.

Appendix M: Storyboard Version 1

BrainEx

Goal: Given an fNIRS dataset, find the top k most similar sequences

Features:

- Load data file into Database
- Select options such as similarity threshold, number of best matches, type of distance, length of interest
- Unsupervised - Display cluster explorer
- Supervised - Display query sequence plot
- Write results to CSV file

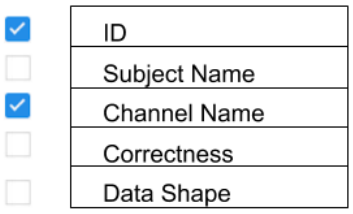
Load file into DB and Select Parameters

Kia will upload a CSV time series data file into the UI

After uploading, Kia will select the data file she wants to analyze from list of already uploaded csv files

Load file into DB and Select Parameters (continued)

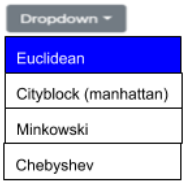
After picking the data file, Kia can filter the data and pick the areas she wants to view closely by choosing one or more of the column names in the dataset



Kia will then select the parameters needed to analyze the csv data

similarity threshold:

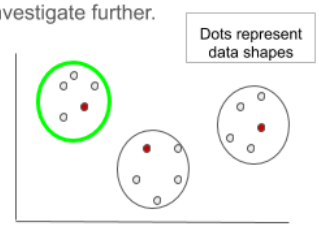
type of distance:



length of interest:

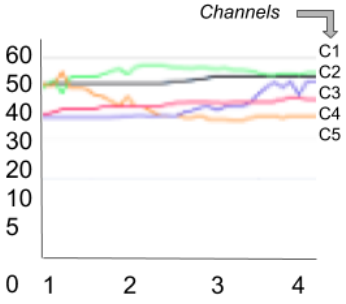
Display Cluster Explorer

Kia is not sure what exactly she wants to query based on the selected data. She can view the following regions to get an idea of how similar the data are in particular clusters, grouped by the length of the time series data within the cluster, and then select a cluster to investigate further.



Kia chooses to view the cluster selected in green

Kia can view a cluster that looks very unique and needs more attention. She can view a graph of the different points in the selected cluster.



Kia can also view the list of shapes of the graphs in a table which she can then pick and choose to investigate more closely.

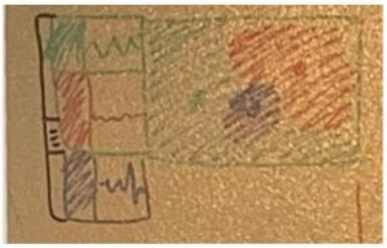
Result: Distance b/w 2 shapes

Kia chooses to investigate the data shape selected in green

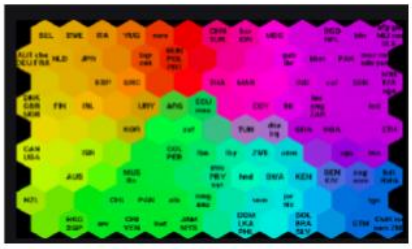
-> Another Option for Viewing Cluster Regions

Show a color coded table of the representatives and show a graph highlighting the clusters beside it

Representatives Cluster Graph



Cluster Graph with more detail:

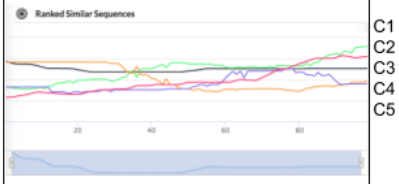


Display Query Sequence Plot

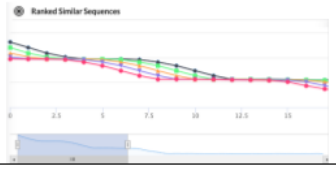
Kia can choose an additional parameter which is the number of best matches to find in the data set

number of best matches:

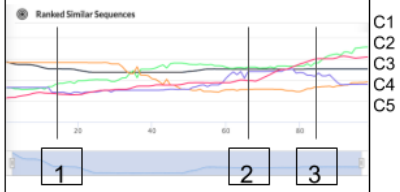
Kia can view the whole query sequence plot.



Kia can then use brushing to analyze the data.



Kia can also give classifications of "correctness" based on the markers and shapes of time-series data and identify the subjects that gave those answers



1 - Correct
2 - Incorrect
3 - No Response

Appendix N: Storyboard Version 2

BrainEx

Goal: Given an fNIRS dataset and a query sequence, find the top k most similar sequences to the given query.

Stage 1 Storyboards Goal: This is an early stage idea. We want to mainly get feedback on the query sequence and cluster explorer concepts

Features:

- Load data file into Database
- Select options such as similarity threshold, number of best matches, type of distance, length of interest
- Unsupervised - Display cluster explorer
- Supervised - Display query sequence plot
- Write results to CSV file

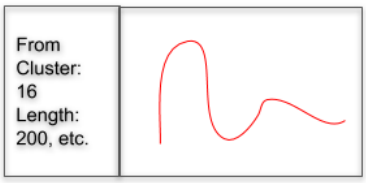
Scenario 1: Querying within the dataset

The user can select a sequence from within the loaded dataset they are working with in either the cluster explorer or dataset explorer and find k best matches to that sequence within that same dataset.

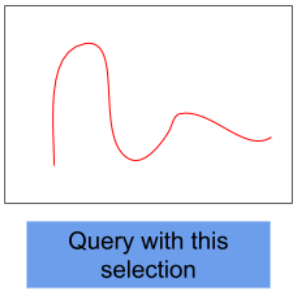
Scenario 1 - querying from within dataset

Kia has been looking through her data in the dataset explorer tab and found an interesting spike in one of the brain signals. She wants to see if there are any other spikes like this elsewhere in her data.

First, she makes sure the data is selected in the dataset explorer.



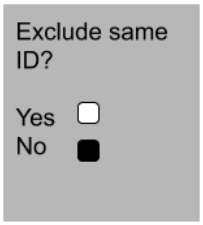
Then, she presses the "Query with this selection" found below the data selection panel.



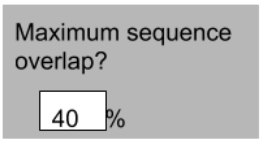
The UI brings her to the query tab.

Scenario 1- querying from within dataset ctd.

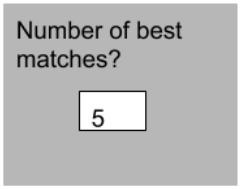
On the query tab, she is asked if she would like to exclude sequences with the same identifier. She says no.



She is then asked what the maximum amount of each sequence in the results overlapping must be.



Finally, she provides the number of best matches she would like to examine.



Scenario 1.5- querying from within dataset (excluding same id) ctd.

On the query tab, she is asked if she would like to exclude sequences with the same identifier. She says yes.

Exclude same ID?

Yes ■
No ☐

Finally, she provides the number of best matches she would like to examine.

Number of best matches?

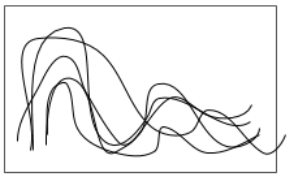
5

Scenario 1- querying from within dataset ctd.

Once all her selections are made, the “Query” button becomes active and she presses the button.

Query

This populates the two results panels. One panel shows line graphs of the 5 sequences. The other panel displays a statistic table including the distances from the query sequence (how similar) and the identifiers of the matches.



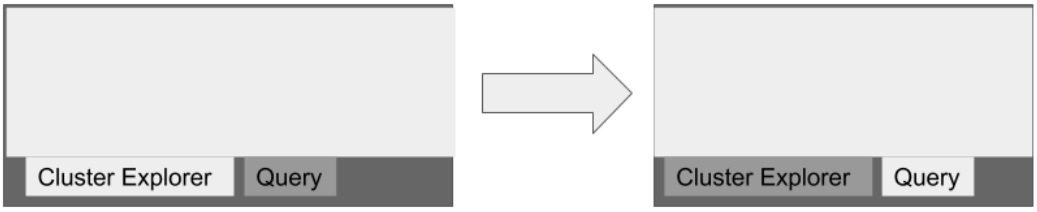
Match ID	Distance
1	33 9
2	02 10
...	

Scenario 2 Storyboard - querying by sequence file

Kia has a sequence she has isolated from one dataset and saved in a CSV file. **1.**

She wants to see if she can find a match in another dataset that she has already clustered in Brainex.

She switches from the cluster explorer tab to the query tab to do so. **2.**

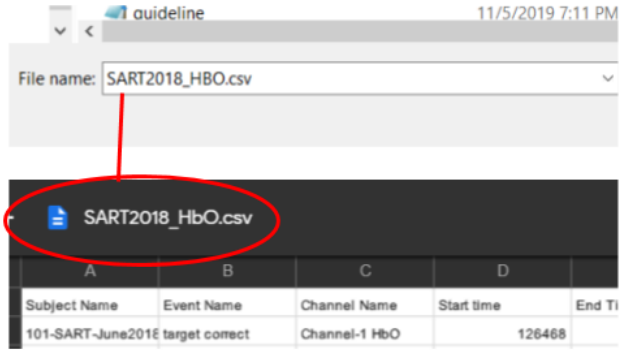


Scenario 2 - querying by sequence file ctd.

There, she selects that she wants to query by loading.

Load a sequence

Her file explorer pops up and she selects the CSV file with the sequence.



Scenario 2 - querying by sequence file ctd.

Then, she inputs the number of best matches she wishes to see.

Number of best matches?

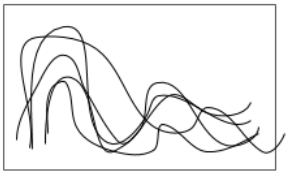
5

Then, she presses the “Query” button.

Query

Scenario 2 - querying by sequence file

This populates the two results panels. One panel shows line graphs of the 5 sequences. The other panel displays a statistic table including the distances from the query sequence (how similar) and the identifiers of the matches.



Match	ID	Distance
1	33	9
2	02	10
...		

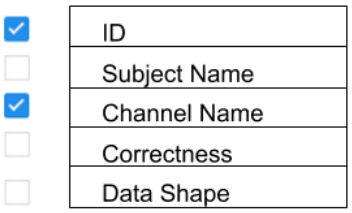
Load file into DB and Select Parameters

Kia will upload a CSV time series data file into the UI

After uploading, Kia will select the data file she wants to analyze from list of already uploaded csv files

Load file into DB and Select Parameters (continued)

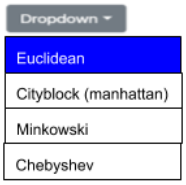
After picking the data file, Kia can filter the data and pick the areas she wants to view closely by choosing one or more of the column names in the dataset



Kia will then select the parameters needed to analyze the csv data

similarity threshold:

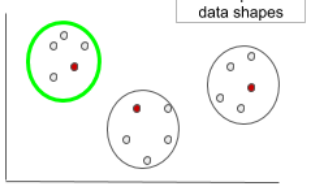
type of distance:



length of interest:

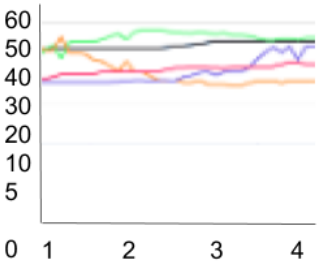
Display Cluster Explorer

Kia is not sure what exactly she wants to query based on the selected data. She can view the following regions to get an idea of how similar the data are in particular clusters, grouped by the length of the time series data within the cluster, and then select a cluster to investigate further.



Kia chooses to view the cluster selected in green

Kia can view a cluster that looks very unique and needs more attention. She can view a graph of the different points in the selected cluster.



Kia can also view the list of shapes of the graphs in a table which she can then pick and choose to investigate more closely.

Result: Distance b/w 2 shapes

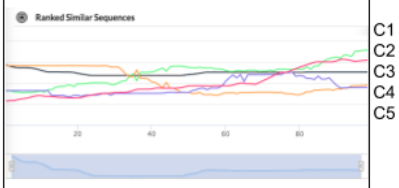
Kia chooses to investigate the data shape selected in green

Display Query Sequence Plot

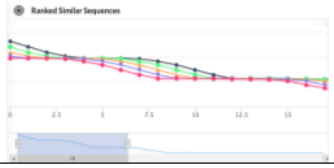
Kia can choose an additional parameter which is the number of best matches to find in the data set

number of best matches:

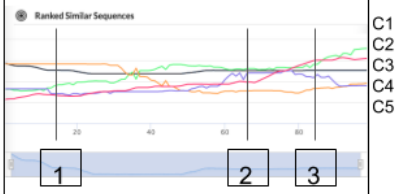
Kia can view the whole query sequence plot.



Kia can then use brushing to analyze the data.



Kia can also give classifications of "correctness" based on the markers and shapes of time-series data and identify the subjects that gave those answers



1 - Correct
2 - Incorrect
3 - No Response

Download Results File

After investigating the data, Kia can download a CSV file with all the analyzed data: the best matches from a query, or, one or more clusters and their representative

Appendix O: Storyboard Version 3

BrainEx Storyboards

By Maggie, Sylvia, Vandana, Kyra

The Goal of BrainEx

The goal of this application is, given a CSV file of fNIRS time series sequences, to find the top *k* most similar matches within the given dataset to an individual query sequence.

Features/Stories:

- Loading CSV file into BrainEx
- Clustering the loaded dataset
- **Exploring the dataset by labels**
- **Exploring the dataset by cluster**
- Choosing a query sequence within the dataset
- Querying with external time series sequence
- Querying with a time series sequence
- Toggle visibility of matches
- Zoom into a region of Query result graph
- Exporting the clustered dataset

In the next several slides are rough sketches of the features of this user interface in narrative form. They are in no way final versions of the application and any feedback, no matter how small, is welcome.

2

Loading and selecting files into BrainEx

Kia will upload a CSV time series data file into the UI

After uploading, Kia will select the data file she wants to analyze from list of already uploaded files and click continue.

3

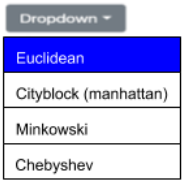
Clustering the loaded dataset

When Kia has an uploaded dataset, she then enters the parameters for clustering the data.

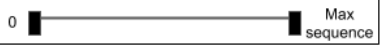
similarity threshold:

0.1 0.2 Custom: ____

type of distance:



length of interest:



When Kia has entered all the parameters, the "Start Clustering" becomes active. Kia clicks the button to cluster her dataset based on the given parameters.

Kia is then shown the clustering is in progress and is presented with the options to either cancel or explore the raw data.

4

Filtering raw data in “Explore Raw Data”

Kia wants to view the brain signal data from subject “101AB_01” to see how many answers they got correct. She selects the appropriate filter options in the filter window

Subject ID:

101AB_01 ▼

Channel:

All channels ▼

Event Label:

Target correct ▼

The data viewer populates with the “target correct” time series data for the given subject.

Data Viewer:

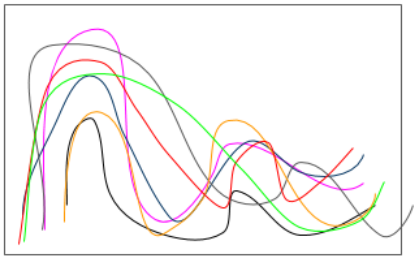
- 101AB_01 | Channel 1 | target correct | (thumbnail)
- 503GF_11 | Channel 5 | target correct | (thumbnail)
- 113DD_98 | Channel 3 | target correct | (thumbnail)
- 101AB_01 | Channel 4 | target correct | (thumbnail)
- 101AB_01 | Channel 5 | target correct | (thumbnail)
- 101AB_01 | Channel 6 | target correct | (thumbnail)
- 101AB_01 | Channel 7 | target correct | (thumbnail)

5

Toggling visibility of sequences on graph

The data visualizer is displaying the first 10 channels listed in the data viewer. Kia wants to compare the time series data between Channel 1, 4, and 15, but not any inbetween.

Data Visualizer:



She deselects the channels she doesn’t want that are already displayed, and selects channel 15 in the data viewer.

Data Viewer:

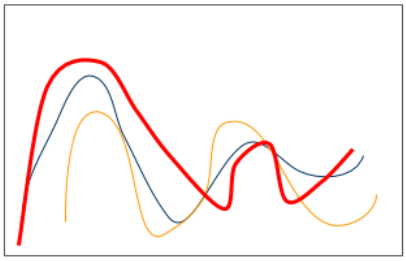
- 101AB_01 | Channel 1
- 101AB_01 | Channel 2
- 101AB_01 | Channel 3
- 101AB_01 | Channel 4
- 101AB_01 | Channel 5
- ...
- 101AB_01 | Channel 15

6

Selecting a sequence via data viewer

The data visualizer displays the time series data for channel 1, 4, and 15 for subject 101AB_01 where the event was “target correct”.

Data Visualizer:



Kia selects the red time series in the data visualizer and the time series for channel 4 is highlighted in the data viewer.

Data Viewer:

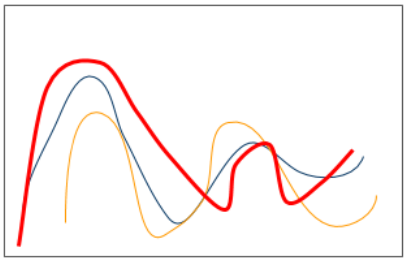
- 101AB_01 | Channel 1
- 101AB_01 | Channel 2
- 101AB_01 | Channel 3
- **101AB_01 | Channel 4**
- 101AB_01 | Channel 5
- ...
- 101AB_01 | Channel 15

7 7

Selecting a sequence via legend

The data visualizer displays 3 sequences and the legend has the corresponding colors and identifiers.

Data Visualizer:



Kia selects the red time series in the legend and the time series for channel 4 is highlighted in the data viewer.

Legend

- sequence1
- sequence2
- sequence3

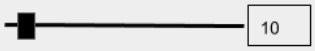
8 8

Filtering clusters in “Explore clusters”

Kia wants to view the first 10 clusters of from length 50 to 55 in the cluster explorer so she adjusts the filter parameters in the filter window

Filter:

Number of clusters to display:



Display clusters of length:

Kia can now view these clusters in the data viewer and the data visualizer, similar to the dataset data visualizer.

Data Viewer:

ID | Length | size (# of sequences) | shape
1 | 50 | 100 | (thumbnail)
2 | 50 | 85 | (thumbnail)
3 | 51 | 117 | (thumbnail)
4 | 51 | 44 | (thumbnail)
5 | 51 | 200 | (thumbnail)
6 | 52 | 15 | (thumbnail)
7 | 53 | 64 | (thumbnail)

9

Choosing a cluster from the data viewer - p1

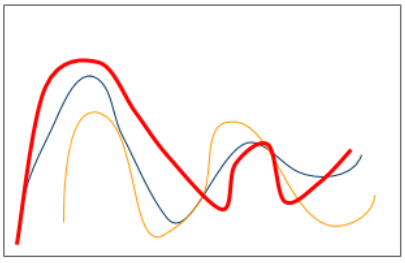
Kyra selects cluster 5 in the data viewer.

Data Viewer:

ID | Length | size (# of sequences) | shape
1 | 50 | 100 | (thumbnail)
2 | 50 | 85 | (thumbnail)
3 | 51 | 117 | (thumbnail)
4 | 51 | 44 | (thumbnail)
5 | 51 | 200 | (thumbnail)
6 | 52 | 15 | (thumbnail)
7 | 53 | 64 | (thumbnail)

The cluster is highlighted in the data visualizer. She decides she wants to view the contents of this cluster and selects “view cluster” on the screen.

Data Visualizer:



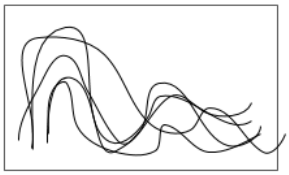
View Cluster

10

Choosing a cluster from the data viewer - p2

Kia can now see the contents of her selected cluster in the data visualizer and data viewer.

Data visualizer



Data Viewer:

- 101AB_01 | Channel 1 | target correct | (thumbnail)
- 503GF_11 | Channel 5 | target correct | (thumbnail)
- 113DD_98 | Channel 3 | target correct | (thumbnail)

11

Choosing a query sequence within the dataset - p1

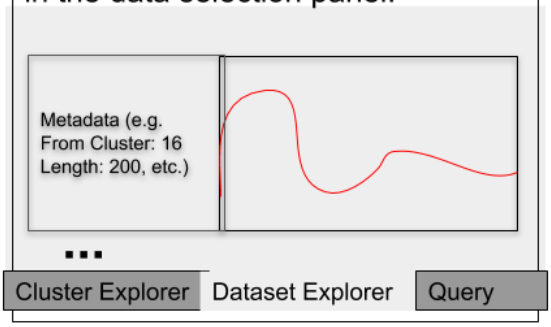
The user can select a sequence from within the loaded dataset they are working with in either the cluster explorer or dataset explorer with which to query the dataset

The Story:
Kia has been looking through her data in the dataset explorer tab and found an interesting spike in one of the brain signals. She wants to see if there are any similar spikes like this elsewhere in her data.

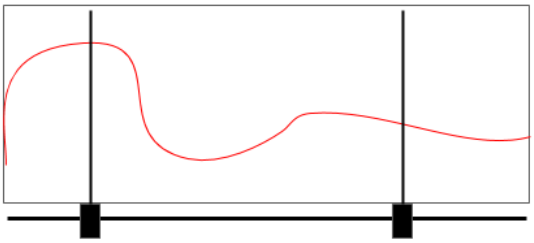
12

Choosing a query sequence within the dataset - p2

First, Kia selects the sequence she wants to query within the dataset explorer tab and it appears in the data selection panel. 1.



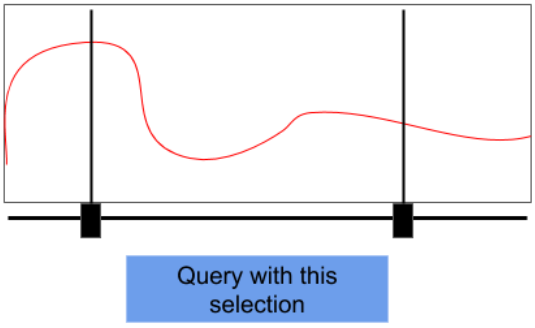
Then, Kia wants to isolate just the interval she is interested in the sequence using the slider in the data selection panel on the left hand side. 2.



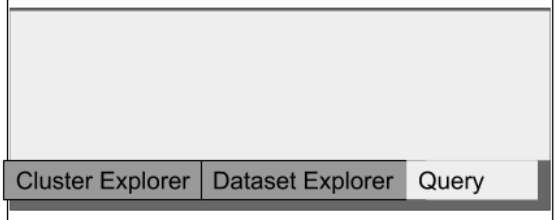
13

Choosing a query sequence within the dataset - p3

Now that she has her query sequence selected and isolated, she can hit the "Query with this selection" button in the data selection panel



Kia will then be brought to the query tab.



The story continues at slide 12

14

Loading an external query sequence - p1

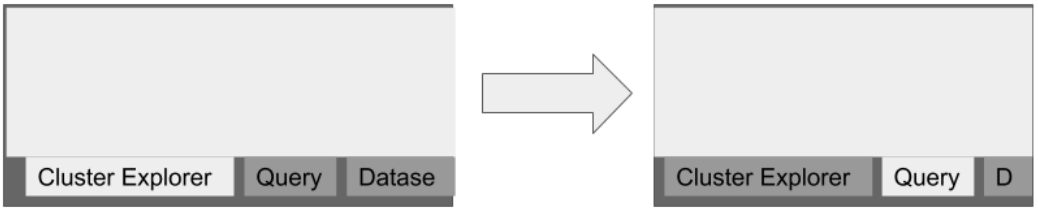
Kia has a sequence she has isolated from one dataset and saved in a CSV file. **1.**

SART2018_HbO.csv

	A	B	C	D	E	F
	Subject Name	Event Name	Channel Name	Start time	End Time	
	101-SART-June2018	target correct	Channel-1 HbO	126468	167986	0.1186

She wants to see if she can find a match in another dataset that she has already clustered in Brainex.

She switches to the query tab to upload her query sequence in the data selection panel. **2.**



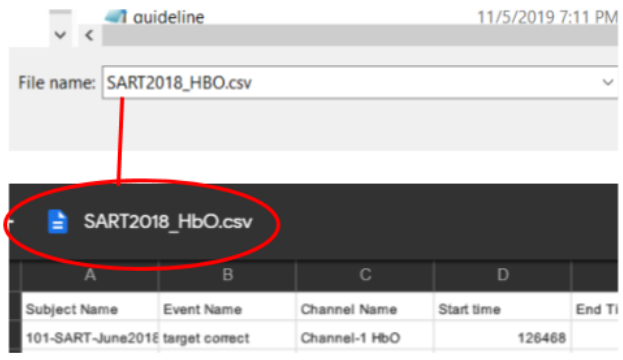
15

Loading an external query sequence - p2

There, she selects that she wants to query by loading.

Load a sequence

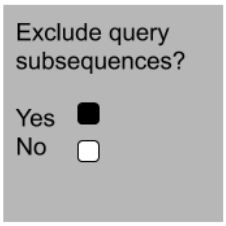
Her file explorer pops up and she selects the CSV file with the sequence.



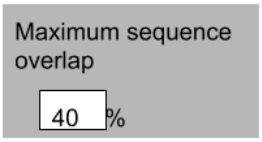
16

Querying with a time series sequence - p1

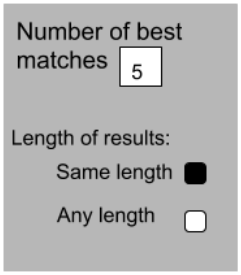
With her query sequence selected, on the left-hand side of the query tab, she is asked if she would like to exclude subsequences from her query sequence. She says yes



She then adjusts the overlap percentage so that the resulting sequences overlap no more than 40%



She provides the number of best matches she would like to examine and says she wants all results to be of the same length as the query sequence.

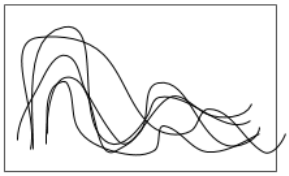


17

Querying with a time series sequence - p2

Once all her selections are made, the “Query” button becomes active and she presses the button.

This populates the results on the right-hand side of the screen: a table and a line graph. The graph shows the 5 matches overlayed with the query sequence. The table displays the rank of each match, its distance from the query sequence (how similar) and the data identifiers.

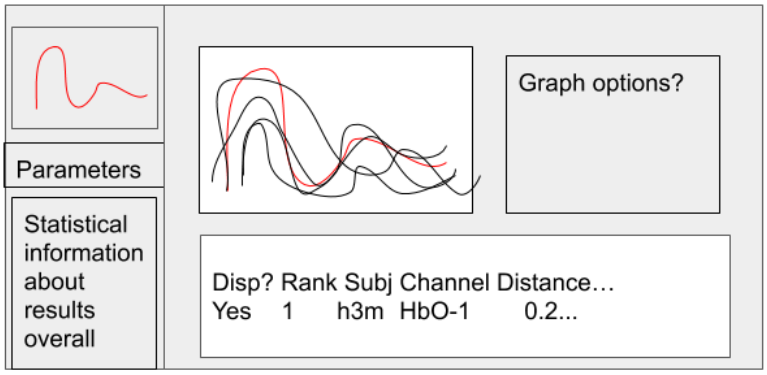


Match	ID	Distance
1	33	9
2	02	10
...		

18

Querying with a time series sequence - p3

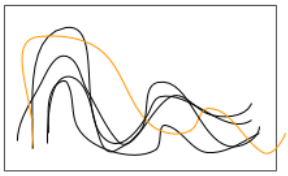
Kia can now view her query in the data selection panel and her query results on the same screen.



19

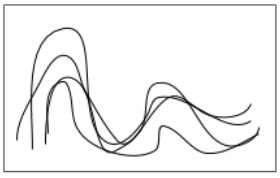
Toggle visibility of sequences/clusters using table

Kia examines her matches and notices that the 5th ranked match has a distance of more than 0.3. She selects it in the table and it highlights in the graph



Disp?	Rank	Subj	Chan	Dist
■	4	h3m	HbO-1	0.09
■	5	f8j	HbO-3	0.4

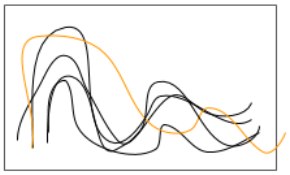
She wants to disregard this match so she deselects it in the table and it disappears from the graph.



Disp?	Rank	Subj	Chan	Dist
■	4	h3m	HbO-1	0.09
☐	5	f8j	HbO-3	0.4

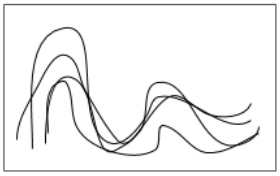
Toggle visibility of sequences/clusters using graph

Kia examines her matches and notices that one sequence looks very different from the others. She selects it in the graph and it highlights in the table



Disp?	Rank	Subj	Chan	Dist
■	4	h3m	HbO-1	0.09
■	5	f8j	HbO-3	0.4

She wants to disregard this match so she deselects it in the table and it disappears from the graph.

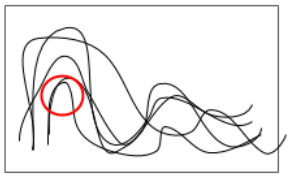


Disp?	Rank	Subj	Chan	Dist
■	4	h3m	HbO-1	0.09
☐	5	f8j	HbO-3	0.4

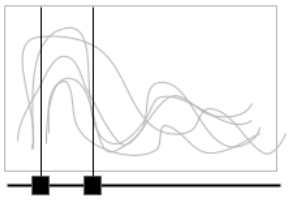
21

Zoom into a region of the data visualizer - p1

Kia notices a region in her results graph that she wants to look more closely at.



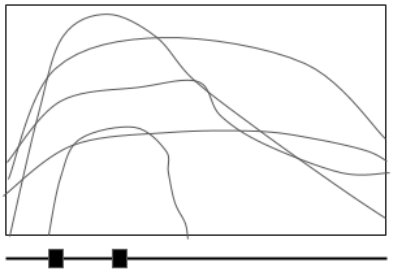
She selects the highlights the interval of the region she wants to enlarge using the sliders below.



22

Zoom into a region of the data visualizer - p2

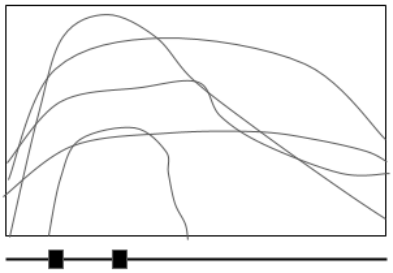
Kia can now view the zoomed in area of the graph. She can adjust the zoom or undo it altogether using the same slider.



23

Zoom into a region of the data visualizer - p2

Kia can now view the zoomed in area of the graph. She can adjust the zoom or undo it altogether using the same slider.



23

Exporting the clustered dataset

When Kia is done using the application she wants to come back to her results later. She downloads her results as a CSV file containing the query sequence and match info.

Download as CSV

24

Appendix P: BrainEx API Tutorials

By/Courtesy of Ziheng (Leo) Li, BrainEx Team

Genex Tutorial

This tutorial is about General Exploration System that implements DTW (Dynamic Time Warping) for exploration of time series data. Genex uses Spark as distributed computing engine.

Spark Context

Spark Context is the entry point to the services of Spark execution engine. It is used to extablish a connection to Spark execution environment and access Spark services.

Creating Spark Context

```
SparkContext(conf = conf)
```

Here conf is the Spark config options such as number of cores and memory.

Example:

```
num_cores = 32
conf = SparkConf(). setMaster("local[" + str(num_cores) + "]"). \
        setAppName("Genex").set('spark.driver.memory', '64G'). \
        set('spark.driver.maxResultSize', '64G')
sc = SparkContext(conf = conf)
```

Here 32 cores and 64 GB of memory is used.

Genex Database

Genex database (gxdb) is a core object in Genex. It stores original time series provided by user and some meta-data.

Creating Genex Database

Genex database can be created in two ways:

1. from_csv

It creates gxdb object from given csv file.

```
from_csv(file_name, feature_num, sc)
```

Here file_name is the source file, feature_num is the number of features in the dataset and sc is the given SparkContext. It returns a gxdb object from given csv data file.

Example:

```
from genex.database import genex_database as gxdb
mydb = from_csv('ECGFiveDays.csv', feature_num=2)
```

In ECGFiveDays dataset there are two features (Sequence Name and Label).

2. from_db

It is a load back function that returns previously saved gxdb object from its saved path.

```
from_db(path, sc)
```

Here path is the path where gxdb object is saved and sc is the SparkContext. The returned gxdb object resides in the given SparkContext.

Example:

```
mydb = from_db('/Genex/db/G_ECG_gxdb', sc)
```

/Genex/db/G_ECG_gxdb is the path where gxdb is saved.

Saving Genex Database

genex_database.save is used to save a gxdb instance locally which can be fetched and restored in future

```
genex_database.save(folder_name)
```

folder_name stores all the essential fields of gxdb instance

Example:

```
mydb.save('G_ECG_gxdb')
```

Grouping and clustering time series

genex_database.build is used for grouping and clustering time series based on the given similarity threshold and distance type.

```
genex_database.build(similarity_threshold, dist_type, loi, verbose)
```

Here similarity_threshold is the upper bound of similarity value between two time series (value is between 0 and 1), dist_type is the type of distance used for similarity calculation (values can be 'eu', 'ma', 'cb'), loi is the length of interest (grouping and clustering will be done only on the sequences of length within given loi), verbose is for printing cluster logs (value can be 0 or 1).

Example:

```
mydb.build(similarity_threshold = 0.1, dist_type = 'eu', loi = slice(110,135), verbose = 1)
```

Here grouping and clustering will be done only the sequences of length 110 to 135.

Generating the query

generate_query is used to generate the query set from given csv file.

```
generate_query(file_name, feature_num)
```

Here file_name is the source file and feature_num is the number of features in the dataset.

Example:

```
query_set = generate_query(file_name = 'ECG_Queries_set.csv', feature_num = 2)
```

Format of ECG_Queries_set.csv is as follows:

Feature 1	**Feature 2**	**Start**	**End**
ECG_1	Label 2	0	135
ECG_2	Label 1	0	110

Querying

genex_database.query is used to find k similar matches for given query sequence.

```
genex_database.query(query, best_k, exclude_same_id, overlap)
```

Here query is the query sequence, best_k is the number of best matches to retrieve, exclude_same_id defines whether to exclude the query sequence when finding k best matches, overlap is a float between 0 and 1 which defines how much the sequences in the query result can overlap (if overlap is 0.5 then no two sequences in the query result must overlap more than 50%)

Example:

```
query_result = mydb.query(query = q, best_k = 5, exclude_same_id = True, overlap = 0.5 )
```

Meta-data Information Guideline

Preparation

cluster_info_dict This field is a dict type data structure that stores the total number of each cluster,

which is distinguished from others according to the length of the representative sequence

Here is an example:

mydb.cluster_info_dict = {112:250, 113:240, 114:230, 111:260, 115:220}

Code Example:

```
for k,v in mydb.cluster_info_dict.items():
    print('Length of cluster: ' + str(k) +'->'+'\n number of cluster of the length.')
```

thumbnail_dict Based on the name of this field, it is also a dict type data structure.

Unlike the above one, this field aggregates the result of the clustering operation into a series of key-value pairs form.

The key is still the length of the representative sequence, and the value is a list of all the representatives that have that length value.

Here is an example: mydb.thumbnail_dict = {112:[repre1, repre2,. . . .], 113:[repre1, repre2,. . .], 114:[repre1, repre2. . .]. . . .}

Code Example:

```
test_seq = mydb.thumbnail_dict.get(112)[0]
cluster = mydb.get_cluster(test_seq)
```

get_cluster() This is a public method for the gxdb class, which will return a list of sequences inside a cluster,

of which is represented by the input sequence.

If there is no such representative, it will raise a exception that asks for a new input.

```
get_cluster(repre: Sequence) -> list
```

Code Example:

```
cluster = mydb.get_cluster(test_seq) # cluster = [seq1, seq2, seq3,.....]
```

Appendix Q: Function Guide

Name	URL	Description
getRawNames	/rawNames	Retrieves the names of the raw CSV files already uploaded to the application
getProNames	/proNames	Retrieves the names of the preprocessed database files already uploaded to the application
getStoreCSV	/getCSV	Uploads a raw CSV file to the application
getStoreDB	/getDB	Uploads a preprocessed database file to the application
setFileRaw	/setFileRaw	Sets the raw CSV file to be preprocessed and queried
setFilePro	/setFilePro	Sets the preprocessed database file to be queried
saveFilePro	/saveFilePro	Save a recently preprocessed file in a place accessible by the user
checkSpark	/checkSpark	Check if the user's Spark installation works
build	/build	Preprocess the dataset to the specifications
uploadSequence	/uploadSequence	Upload the CSV file with the sequence to be queried
complete_query	/query	Query the dataset with the uploaded sequence and given arguments
save	/saveDataOutput	Saves the query results to a CSV

stop	/restart	Shuts down the current Genex engine so another one may be made

Appendix R: BrainEx GUI Documentation

A guide to installing and using the BrainEx application

Use this application in full screen mode for best results.

Table of Contents:

Installation

Note: This app was developed on Windows 10 OS, but additional directions for MacOS and Linux are supplied when possible. Any operating system-specific additions are welcome.

Required software

It is necessary for the machine that BrainEx is running on to have the following software installed and configured:

1. Python 3.6 or 3.7 (Python 3.8 has some known issues with packages used by this application and the backend requires 3.6 or greater) with pip included
    a. Add the path to the python.exe and the Scripts folder to PATH
        i. For more information on how to add environment variables on Windows, Linux and MacOS, refer to this link
        ii. Note: for environment variables to take effect, the command window must be opened after they are set.

To check if Python is properly installed, or to see what version of python is on your current machine, open a new command prompt and execute the following command:

- Windows and Linux: `python --version`
- MacOS: `python -version`

For more information on how to verify your python installation/version, refer to this link.

2. Node.js (npm is required to run the React app)
    a. To check if node is properly installed execute the command `node --version` in a new terminal.
3. Microsoft Visual C++ Build Tools (install with default options)
4. Java 8 (required for Spark)
    a. To check if java is properly installed (and that you have the right version), execute the command `java -version`.
5. If you are on Windows, you will need additional software to unzip the Spark installation. We recommend 7zip as it is free and can handle .tgz unzips.
6. Spark/Pyspark (Please note that while PySpark is available for Pip install, environmental variables must be set to get the install to work properly)
    a. Windows installation and configuration
    b. MacOS installation
    c. Linux installation
    d. To verify that Spark is installed correctly, open a command terminal and execute the command `pyspark --version`.

If Spark is unable to be installed, you will still be able to run the functionalities of the app, though not to their best ability.

Installing the application

Cloning/Downloading Github repository

If you wish to use an IDE, we recommend installing Pycharm (Community Edition should be sufficient) with the default options and install the "JavaScript and TypeScript" bundled plugin. On startup of the IDE, you can select "Checkout from Version Control" and clone the Github repository this way. If there are any remote changes on the gitHub repository you wish to retrieve, check out how to sync with a remote repository on Pycharm for instructions on how to use the IDE for version control.

If you are not using an IDE or simply prefer using git through the command line, open a command prompt and navigate to the location you wish to install the application. Then, execute the following command:

`git clone https://github.com/krbrez/BrainEx-UI.git`

If there are any remote changes to the application, just execute `git pull origin` in the command line from within the project directory.

Alternatively, you can open this github link and download the repo as a zip file. Then, simply extract the containing files to the desired installation location. To update the application in this case would require re-downloading the application and completing the following installation instructions again.

Please note that the Github repository for the BrainEx UI includes a frozen version of the Genex API. If any major changes are made to Genex, the version in this repository will not be updated automatically.

Installing dependencies

Though it is not required, if you wish to use a virtual environment (directions for Pycharm) for this project (especially if you have multiple versions of Python installed), you will want to configure the Python interpreter with the version of Python that you want prior to installing any dependencies. If you are not using an IDE and still wish to do this through the command line, try referring to this link.

Otherwise, as long as the `--user` option is included in the pip installation command, packages should install without additional permission required.

If you are running this on a Windows operating system, you can simply execute the following command: `run_all.bat` and it will open two command prompts and simultaneously install npm and python dependencies and run both the React app and the Flask server.

Otherwise, execute the following commands approximate commands on your machine:

- *To set up the backend:*
    - `venv/scripts/activate` (only if using virtual environment)
    - `python -m pip install --user requirements.txt`
    - `cd BrainEx/backend` (when you are in the BrainEx-UI root directory)
    - `set FLASK_APP=functions.py`
    - `flask run`
        - This may need to be replaced by 'python -m flask run' if your Python Path is not configured properly.
- *To set up the frontend (in a separate terminal/command window):*
    - `cd BrainEx` (when you are in the BrainEx-UI root directory)
    - `npm install`
    - `npm start`

Once the development server is started and compiles, it should open a window in your default browser. If it does not automatically open, using the browser of your choice navigate to `localhost:3000`. If you wish, you can create a shell script for your non-Windows operating system, add it to the root directory and use that when you want to come back to the application.

Additionally, if you *are* using a virtual environment, the genex folder, while included in the root directory of the repo, will also be needed to be added to site-packages within the lib folder of the virtual environment.

Installation Troubleshooting

- If pip is not found, execute this command in the root directory (/BrainEx-UI): `python get-pip.py`
- If upon running the backend you receive any "module not found" errors (that are not genex) just execute `python -m pip --user install <package name>` in the root directory.
- If any problem persists with node modules, try deleting the `package-json.lock` file and node_modules folder and execute `npm install` again.

About the Application

Overview

BrainEx finds the *k* most similar sequence matches from within a dataset using a given individual query sequence. The advantage of this application is that querying time is reduced due to the similarity grouping done during preprocessing. Currently the application is capable of the following use cases:

- Uploading a raw dataset and/or choosing a raw dataset, preprocessing it for use, and querying with a sequence of the same number of identifiers (i.e. SubjectID, EventName and SubjectID, ChannelNum). The general path of this use case would be Home > Select a new dataset > Preprocessing options > Preprocessing progress menu > Find Similar Sequences.

- Choosing a preprocessed dataset and proceeding straight to querying, and using a query sequence with the same number of identifiers. The general path of this use case would be Home > Find Similar Sequences.

General File Structure

- BrainEx-UI
    - BrainEx
        - Backend: contains files relevant to the backend
            - tests
                - Test_functions.py: functions used to test some of the Python backend
            - Uploads
                - Preprocessed: contains all uploaded preprocessed files. These will appear on the Home page
                - Raw: contains all raw files. These will appear on Select a new dataset.
            - Functions.py: the Python backend that receives http requests from the frontend and parses them to work with BrainEx functionality
        - Node_modules: when `npm install` is executed, this folder will be created. It will contain all installed packages.
        - Src: contains files relevant to the React front end
            - Components: folder containing the individual components of this app. The main components are described below in Application tour
                - Preprocessing: contains files relevant to the preprocessing pipeline
                - Singletons: contains files that should only exist as once instance across the app (i.e. header and navigation bar)
            - Data: contains default data as well as dummy data used to develop this app

                - Stylesheets: contains any styling for the app. If a component does not have a dedicated stylesheet, the parent stylesheet is what is being applied (e.g. App.css also applies to any component within the app as it is the highest parent, but child CSS will override it if they exist)
                - App.js: the highest class/parent of the application. Contains all relevant Router info for navigation to each page
                - Brain.svg: the app logo
                - Index.js: file that servers the main App.js
                - serviceWorker.js: a file that can allow the React frontend to work offline and load faster. See the file contents for more detail. This file will only be used if the function called in index.js is changed from unregister() to register()
                - README.md: a readme file provided by React that details available scripts
                - Test.py: more test functions for testing backend functionality
            - Package.json: contains package dependencies to run React frontend. This is what is installed when `npm install` is executed. Executing `npm install --save <package>` will add the package to this file as well as install it
    - Example_files: contains example CSV files for raw datasets as well as query sequence CSV files
    - genex: folder containing the Genex package/API (when Genex is available as a pip package, use that instead)
    - Saved_Preprocessed: folder containing the saved preprocessed datasets in zip file format
    - Saved_Results: folder containing saved query results in a parsable format. Currently not outputted as a CSV file
    - Venv (if you are using one for this project): contains virtual environment information such as scripts and installed packages
        - Lib
            - Site-packages: this is where you would paste the genex package if you were using a virtual environment
    - .gitignore: contains the names of files and folders that should not be pushed to the repo and therefore not tracked by git.
    - brainex.bat: script to start the front-end React app
    - LICENSE: MIT license
    - README.md: the file you are currently viewing
    - requirements.txt: names of required Python packages to be able to run the backend and Genex
    - run_all.bat: script to run both the front-end and backend in two separate terminals
    - server.bat: script to run the Python backend

Application tour

Home

This is the initial/main page of the app. It has 3 main elements.

- A user can upload a preprocessed dataset created by this application, which comes in the form of a compressed ("zip") file that contains the necessary information for the application to use. The zip file must be created by this application or in the exact format to be of valid use.
- If the user clicks on a preprocessed file from the left side of the screen, a dialog will appear asking the user if they wish to proceed with this dataset, and to verify the Spark context parameters (number of workers, driver memory, max result memory). The user will then be brought to the "Find Similar Sequences" page upon confirmation of the parameters.
- On the right hand side of the screen, there is a brief description of the app and a button to instead choose a new, raw dataset. This will bring the user to the next screen, "Select a new dataset"

Select a new dataset

This page is for selecting a new, raw dataset rather than a preprocessed one. Users may also upload new datasets to their local copy of the application.

Listed below are the requirements of new datasets:

1. The dataset must be a CSV file of time series data.
2. The first row of the dataset must contain the headers/identifiers of the dataset
3. Any column header for a column containing the time series data points must be blank
4. The dataset should have at least 1 header/identifier.

When the user selects a dataset from the left side of the screen, a preview of the file contents will be displayed on the right side, including headers. The unlabeled time series data are given the nickname "unnamed #". Upon clicking a dataset, the "Proceed to Preprocessing" button will activate and the user will be able to go to the next screen.

Preprocessing options

The purpose of this page is to select the parameters with which the backend should preprocess your raw dataset. The screen begins with the recommended defaults displayed (Distance type: Euclidean with Dynamic Time Warping (DTW), Similarity Threshold: 0.1, Length of Interest: entire dataset, and preprocess without using Spark).

The parameters for the preprocessing stage are as follows:

Distance Type

This option is Euclidean with Dynamic Time Warping (DTW) by default.

This is the distance type used for similarity calculations. The choices are Euclidean with DTW), Manhattan with DTW, Minkowski with DTW, and Chebyshev with DTW. These options really only differ in how the similarity/distance is calculated, so the appropriateness is up to user preference.

Similarity Threshold

This option is set to 0.1 by default.

This parameter determines the upper bound of the similarity value between two time series. It is on a scale of [0:1]. The closer the threshold is to 0, the more similar the matches must be to be grouped together (and vice versa). For example, if the user sets the similarity threshold to 0, each sequence would be in a cluster/group consisting of only itself. If the user were to set the threshold to 1, then one large group of sequences for each specific length would be created.

Lengths of interest

This option is set to all lengths by default.

Lengths of interest refers to the interval of sequence lengths into which the user wishes to query. Grouping/clustering will be done only on the sequences of length within the given interval. All outside sequence lengths will not be included and therefore will not be queried. To achieve the "all lengths" setting again, just set the interval to the lowest and highest values allowed by the slider/input.

Number of Workers

This option is 4 by default and is required for preprocessing with *and* without Spark.

Number of workers refers to the number of cores that will be used to run the preprocessing algorithm. Please do not exceed the number of cores on your machine. You can check the number of cores on your machine through the performance tab on your task manager.

Preprocess using Spark

This option is deselected by default.

When this option is selected (to select it simply check the checkbox), additional options will appear to the right for memory allocation. If the Spark installation is found to be invalid, an error will occur and the options will not appear. These parameters, Driver Memory and Max Result Memory (both 16 GB by default), are used to configure the Spark Context.

When this option is not selected, the dataset will be preprocessed using Python Native Multiprocessing. Using this option may cause preprocessing and querying to take longer than if using Spark for larger datasets, but for smaller datasets the difference is negligible.

Preprocessing progress menu

During preprocessing

This page will show an indeterminate progress bar while the dataset is being preprocessed. The user will also be presented with the options to cancel preprocessing and return to the previous screen (preprocessing options) or cancel preprocessing and return to the home page. If you are preprocessing while using Spark, the backend terminal will show the progress of the Spark tasks/jobs. It may hang for some time before starting Stage 0 but then it will progress quite steadily.

After preprocessing is complete

Once preprocessing is complete, the screen will show a filled progress bar and the user will be presented with the option to restart with another dataset (and return to home), to proceed to find similar sequences, and/or to download the preprocessed dataset to the local root directory in a folder called "Saved_Preprocessed". The dataset will be saved as a zip file containing the necessary information for the application to use in the future.

Note: It is important to go through the proper channels if you wish to restart with another dataset, especially if you are using Spark. In order to correctly end your Spark session, you may select either cancel button, the "restart with another dataset" button, as well as the "Home" button on the "Find Similar Sequences" page. The browser "back" button is not sufficient in this case.

Find Similar Sequences

This page contains the main functionality of this application, finding the *k* best matches to a given query sequence.

Query Options

There are three available parameters for querying into the given dataset:

- Query sequence: a full row of data (identifiers, data points, etc.) copied from a dataset into another CSV file. This does not include any headers. The sequence must have the same number of identifiers as the dataset into which you are querying.
- Number of best sequence matches: the number of sequence matches the backend will return. This value must be equal to or less than the number of sequences within the preprocessed dataset. However, this does not guarantee that there will be matches close to the given query sequence. The max number of matches allowed for the particular dataset

is displayed below this field, and depends on the number of subsequences collected during preprocessing.
- Overlap: This value determines how much the sequence matches will overlap with each other. For example, if you set the overlap value to 40, then no two matches in the query result will overlap by more than 40%.
- Exclude subsequence matches from current sequence: This checkbox indicates whether or not any sequences with the same sequence id (a concatenation/tuple of the identifier values) will be considered in the match ranking process. This field is most relevant when the query sequence comes from within the chosen dataset.
- *Lengths of interest: This feature is not currently implemented in the Genex package. In theory, you would be able to query for specific sequence lengths within the lengths of interest that was originally provided before preprocessing.*

To start the querying process, the user must have a query sequence selected and valid input for the above options and then select "Start Query". The query may take some time depending on how the dataset was preprocessed and the *k* number of matches requested but when it returns the graph, table, and stats will populate with the results. To clear the results, the user can click "Clear Options", which will clear the currently selected options, not including the query sequence.

Query Results

This portion of the page displays the query results in both graph and table format. The "Ranked Sequence Matches" table includes the following information:

- Show: checkboxes to toggle an individual sequence's visibility in the graph. There is also a select all/deselect all checkbox in the header cell. With the current implementation, if the user were to uncheck all sequences individually, then the most recently displayed sequence would still appear on the graph. This column is color coded using a gradient scale between blue and orange with blue being the most similar match and orange being the least similar match. The colors correspond to the sequence's line color in the graph.
- Rank: the sequence's match rank (#1 being the best match)
- Sequence ID: a unique sequence identifier consisting of the concatenated row identifiers
- Start Time, End Time: The start and end time of the sequences as points in the data set
- Similarity: The percentage of similarity between this match and the query sequence. The closer to 100% the more similar the match is.
- Thumbnail: A color-coded thumbnail of the sequence showing the user its general shape. The color corresponds to its color in the graph.

The "Query Results" plot includes the sequences toggled to be shown in the graph, where the user can view and use the brushing tool below the plot to zoom into a specific interval of the graph. Simply click and drag each side to the desired area and click again to release. There is also a legend that displays the color and sequence IDof each displayed line. This legend will possibly become legible as more and more matches are requested.

Known limitations

1. We recommend using this application in fullscreen mode for the best results and experience.
2. Refreshing the page may cause the application to lose track of some relevant variables. If you have refreshed by mistake, return to the beginning of the pipeline through the proper channels (not through the browser back button).
3. If a change is made to the Genex package that changes names of functions, parameters, and/or return values, this application will require refactoring in order to function as expected.

Using the application

Once the dependencies for the application are properly installed (see Installation), the application is ready to be used. The user can execute the "run_all.bat" Windows batch script in order to start both the backend server and the BrainEx front end ("server.bat" and "brainex.bat", respectively), and the default browser should open to "localhost:3000". If the browser does not automatically open, typing this into any browser that supports JavaScriptwill work. The application will open up to the Home page and files from the path `BrainEx-UI/BrainEx/backend/uploads/preprocessed' (this is the path for uploaded preprocessed datasets) should populate on the left side `.

Use case tutorial: Querying with a raw dataset

To use a new, raw dataset, select "Preprocess with a new dataset" on the Home page. The following page, "Select a new dataset", will show the files stored in the folder `BrainEx-UI/BrainEx/backend/uploads/raw`. If you have a dataset on your local machine you wish to upload to the application, you can do so using the "Choose File" button below the file list. The file should follow the format outlined in Select a new dataset in order to be valid.

Selecting a file from the list will activate the "Proceed to Preprocessing" button and cause a paginated table of the file's contents to populate to the right. There may be some delay, during which "No Data" will be displayed in its place. When the preview appears, it will show 10 rows at a time, and the rest can be viewed by paging through using the table controls.

When you have selected the dataset you want to preprocess, clicking "Proceed to Preprocessing" will bring you to the Preprocessing options page. Here, you can use the default selections or enter your own. See Preprocessing Options for more detail on each option. If you select "Preprocess using Spark", the application will verify that Spark/Pyspark is properly installed. If it isn't, you will be notified. Otherwise, additional Spark/Pyspark parameters will appear.

Once you are satisfied with the selected preprocessing options, click “Start Preprocessing”, and you will be brought to the Progress page. While preprocessing is happening, the only options presented are to cancel and return to either Preprocessing options or the Home page. This will properly end the current Spark/Pyspark session so that another one may be created. A notification will appear when this happens successfully. Otherwise, once preprocessing is complete the screen will change to have the options to restart with another dataset (which will also properly end the current Spark/Pyspark session), to find similar sequences, and/or to download the preprocessed dataset.

Clicking on “Download the preprocessed dataset” will cause a spinning progress circle to appear while the download is in progress. Once the download is complete, the circle will disappear and a notification will appear indicating that it has been saved in the “Saved_Preprocessed” folder. This folder is located in the root directory of the project and contains compressed (“zip”) files of preprocessed datasets.

To query into this preprocessed dataset, select “Find similar sequences” to proceed to that page. Here, you will see querying parameters available on the left side, as well as an unpopulated statistics window, empty table, and empty graph. To select a query sequence, choose the file from your local machine. The query sequence file must contain only a single row of data taken from the same dataset or a dataset with the same number of identifiers. See Query Options for more information.

After entering the number of matches and overlap values, or using the defaults, clicking “Start Query” will initiate the querying process. While the matches are being found, a loading overlay will appear over the graph. On completion of the query, the table and graph will populate with all the matches. To toggle the visibility of each sequence on the graph, select/deselect the row in the table. To save the query results into a CSV file, click on the “Save Query Results” button at the bottom of the table. This will create a CSV file and store it in the “Saved_Results” folder of the project root directory.

Use case tutorial: Querying with a preprocessed dataset

This use case begins from the Home page as well. However, instead of selecting the “Preprocess a new dataset” button, you must select a zip file listed on the left side of the screen. Similar to using a new dataset, you can upload a preprocessed dataset to use. This dataset must be the same format as or come from the Saved_Preprocessed folder mentioned in the previous section. When a file is selected, a dialog modal will appear in the center of the page asking if you would like to query using this dataset. It will also prompt you to review/enter the necessary parameters to create the Spark context.

Clicking “yes” on this modal will bring you straight to the “Find Similar Sequences” page. Here, you will see querying parameters available on the left side, as well as an unpopulated statistics window, empty table, and empty graph. To select a query sequence, choose the file from your local machine. The query sequence file must contain only a single row of data taken from the same dataset or a dataset with the same number of identifiers. See Query Options for more information.

After entering the number of matches and overlap values, or using the defaults, clicking “Start Query” will initiate the querying process. While the matches are being found, a loading overlay will appear over the graph. On completion of the query, the table and graph will populate with all the matches. To toggle the visibility of each sequence on the graph, select/deselect the row in the table. To save the query results into a CSV file, click on the “Save Query Results” button at the bottom of the table. This will create a CSV file and store it in the “Saved_Results” folder of the project root directory.