Preparing the E-TRIALS Minimum Viable Product for Release

Improving Platform Usability, Security, & Infrastructure

March 19, 2022

Authors:
Edward Philippo
Matthew Spofford

Submitted to:
Dr. Neil T. Heffernan
Mr. Ryan Emberling

Worcester Polytechnic Institute
Worcester, MA

Abstract

Online learning has become vital during the COVID-19 pandemic, but it is unclear how teachers can best support online coursework. To address this, the E-TRIALS project (EdTech Research Infrastructure to Advance Learning Science) leverages the ASSISTments online learning service to provide a web application where anyone can build & deploy learning science research studies. Student projects in 2020 and 2021 kickstarted the current iteration of E-TRIALS and led to the hiring of a professional software development team. Since then, this project helped prepare the platform for public release to the research community by improving application functionality, usability, security, and infrastructure. With the assistance of this project, E-TRIALS was launched in February 2022.

For further questions or inquiries about this project, contact the project advisor using the email shown below.

Dr. Neil T. Heffernan: nth@wpi.edu

Acknowledgements

Our team would like to recognize and show our gratitude for everyone that has enabled this project. It would not have been possible without their help. Thank you to:

- Dr. Neil T. Heffernan for providing the opportunity to work with the ASSISTments Foundation and being supportive of us and our work throughout the entire project.
- Cristina Heffernan for providing historical insight on the purpose and importance of E-TRIALS and the ASSISTments in general, as well as providing invaluable feedback about platform improvements.
- Ryan Emberling for his incredible mentorship throughout the project. His constant patience, understanding, and empathy made engineering work a pleasure. He was willing to go out of his way at every turn, whether to provide support for a task or just to check in with us outside of the project. We are truly grateful for his efforts!
- Amina Alibasic, Jennifer Choi, Emir Fatusic, Aaron Haim, Hannah Pandolph, and Brian Rojas for being a wonderful development team to work with. We could not have asked for a more supportive and knowledgeable team.
- Angela Kao for being immensely helpful with managing our project's administrative logistics. Her help made scheduling and account management a breeze.

Table of Contents

Abstract i

Acknowledgements ii

Table of Contents iii

Chapter 1. Introduction 1

Chapter 2. Background 3

2.1 - What is ASSISTments? 3

2.2 - What is E-TRIALS? 3

2.3 - Using the Open Science Framework 4

2.4 - E-TRIALS Development History 5

Basis of E-TRIALS 5

User Research & Prototyping 5

Full-Stack Implementation 7

Chapter 3. Methods 11

3.1 - Overarching Objectives 11

Implement and Fix Basic Functionality 11

Improve Interface Usability 11

Ensure a Secure Product 12

Establish Robust Development Infrastructure 12

3.2 - Team Workflow 12

Chapter 4. Results 14

4. 1 - Improving Functionality 14

4. 2 - Build Page and General Usability Improvements 15

Build Page Overhaul 15

Assorted Improvements 17

4.3 - Back-End Security Fixes 18

Endpoints Authorized for Owners/Admins 18

User Role Permissions for Studies Phases 19

Enable Specific Endpoints for Local Environments 20

Remove OSF API Key from Version Control 21

4.4 - Development Infrastructure and TypeScript 22

Chapter 5. Discussion 24
5.1 - Significance of Our Work 24
5.2 - Future Work 25
5.3 - Final Thoughts 26

References 27

Appendix 28
Appendix A: Load Problem Data after OSF ID Validation in HomeOsfModal.vue 28
Appendix B: Updating OSF Project Based on Changes in Study 28
OSFManager.java 28
OSFManagerImpl.java 29
StudyManagerImpl.java 31
Appendix C: Disabling Removal Buttons for Admins 33
StudyContentEditor.vue 33
ViewChosenProblems.vue 33
Appendix D: Additions to User Permissions in StudyManager.java 34
Appendix E: Additions to User Permissions in StudyManagerImpl.java 34
Appendix F: Environment Specific Features in StudyManagerImplLocal.java 38
Appendix G: Removing OSF API Key from Back-end 39
Appendix H: Problem and Problem Set Types for TypeScript 40
Problems.ts 40
ProblemSet.ts 41

Chapter 1. Introduction

The number of students who rely on remote learning has skyrocketed since the start of the COVID-19 pandemic. News media has closely followed educators' decisions, and much of that covered has highlighted missteps: from college shutdowns to childcare woes. All of this has given new urgency to a previously overlooked question: "How do you improve online education?" Unfortunately, the complexity of education makes this question difficult to even approach (Koedinger, 2013). Learning science is a slow-but-steady field of research owing partially to the resources available. Current study design tools require researchers to create entire courses, and then recruit students to take them. Such costs make learning science research difficult for all and totally infeasible for many.

The ASSISTments Foundation tried to address this issue by adding research functionality to the existing content moderation tool (colloquially known as the "1.0 Builder"). Still in use today, this product has a very wide set of capabilities, ranging from problem set creation to study construction. However, this jack-of-all-trades approach has led to significant problems. The 1.0 Builder is hard to use and harder to learn, with an interface that is visually dated and functionally lacking. Operations that are conceptually simple (such as associating specific pieces of content with experimental conditions) are unclear and time consuming. Scattered menus make it difficult for new users to find specific actions, and any research functionality is clearly an afterthought.

Two previous projects have sought to solve these issues by developing a modern web application with usability as a central focus. Student work in *E-TRIALS* (Krichevsky, Spinelli, 2020) kickstarted the new product (named E-TRIALS) by designing and prototyping key pages in the study building process. The end of that project prompted hiring a professional development team to continue work, who were augmented a year later by the second student project. *Continuing the Development of E-TRIALS* (McCarthy, 2021) brought a series of interface improvements and laid the stage for user testing.

Despite this progress, at the start of this project E-TRIALS still had a lot to improve if the platform was going to move from a proof of concept phase to a minimum viable product. The software infrastructure needed improvements, only one type of study was supported, and it was not yet possible to deploy studies to students. The application also suffered from general usability problems, bugs, and missing functionality, and many user interfaces needed to be

updated to match revised designs. Evidently, there was a lot of work to be done for enabling E-TRIALS to become a professional enterprise-level application.

The goal of this project was to perform software engineering tasks to support the ASSISTments Foundation as they release the E-TRIALS minimum viable product, in order to improve the user experience for educational researchers. The team accomplished this goal via the four following objectives:

1. Implement and fix basic functionality
2. Improve interface usability
3. Ensure a secure product
4. Establish robust development infrastructure

Chapter 2. Background

2.1 - What is ASSISTments?

ASSISTments is a web-based tutoring platform that strives to make it easy for teachers to tailor the ideal education to each of their students. Teachers assign problems to their students, and then, based on the students' performances, receive reports highlighting how to best support each student as they move forward. ASSISTments has a substantial user base consisting of both students and instructors, and has experienced significant growth since the start of the COVID-19 pandemic.

ASSISTments supports a variety of question and assignment types. Questions range from conventional multiple-choice questions to free answers. Assignments are made up of questions, and can either be traditional (linear) problem sets or dynamic "skill builders" which require students to answer questions until they have reached a certain streak of correct answers in a row.

In addition, ASSISTments includes built-in functionality to support educational research. As one example, the platform is able to dynamically swap out explanations or hints (also known as "student supports") on any given problem. For example, two students could be assigned the exact same problem but receive different explanations of how to solve it. On a larger scale, a researcher could assign two explanations across a population of 200 students. The researcher could then compare the resulting grades and infer which explanation was more effective at helping the students learn. This mutually beneficial relationship gives researchers opportunities to conduct studies while also allowing ASSISTments to benefit from research results.

2.2 - What is E-TRIALS?

E-TRIALS (the Ed-Tech Research Infrastructure to Advance Learning Science) is a research platform that provides learning science researchers with an easy and low-cost way to create and deploy their own studies. It utilizes ASSISTments' research integration and established user base, which means researchers can spend less time working out the details of their study's deployment and more time developing the study itself.

Before this project began, E-TRIALS let users create what is known as a "Support Comparison" study, where they choose an existing problem set to serve as the basis for the study and then design two student supports (hints or explanations) for every problem in the problem

set. Deploying the study will evenly distribute the custom supports across a random sample of student users. When the study concludes, the researcher will receive student performance data broken down by which support the student received. The professional development team also planned to implement a new type of study named “Problem Varied”. Unlike the Support Comparison studies, which evenly varied supports among participants, Problem Varied studies would evenly vary problem sets among participants. This study type was implemented during the course of this project by the professional development team.

Beyond speeding up the research process, E-TRIALS’ streamlined deployment has the possibility to greatly increase accessibility of learning science research. Many researchers lack access to funding and participant populations: notably those from historically underfunded institutions, students, or those from small institutions. Their work is hampered–or even cut short–by the costs of traditional deployment methods. By making study deployment an easy, inexpensive affair, E-TRIALS provides research opportunities to those who have good ideas but lack the funding to act on them.

2.3 - Using the Open Science Framework

With educational researchers being the primary users of the E-TRIALS platform, the E-TRIALS team decided it would be useful to provide researchers the capability for sharing their studies and research they develop. The Center for Open Science, a non-profit organization provides such a platform named the Open Science Framework, or OSF (Center for Open Science, n.d.). The purpose of the platform is to provide collaborative opportunities for sharing research openly in the scientific community. E-TRIALS decided to leverage this by integrating with OSF using their web-based application programming interface (API). The OSF API enables the E-TRIALS web application to post researchers' projects and studies they are currently developing. As a result, researchers need to create an OSF account and provide their corresponding ID before being given “researcher” privileges.

2.4 - E-TRIALS Development History

Basis of E-TRIALS

The development of the E-TRIALS platform initially began as a result of the original ASSISTments content building tool, known as the 1.0 Builder. While this tool was well equipped at modifying hints and explanations like the current E-TRIALS platform, it was also highly generalized for various tasks. The tool was intended not only for constructing studies, but also providing teachers the ability to create their own problem sets, enabling administrative content moderation, and even creating more advanced studies than what is currently capable on the E-TRIALS platform. However, as the 1.0 Builder accumulated more features, this multi-purpose capability became a flaw. Due to the builder's high complexity, even basic tasks in the 1.0 Builder were challenging for researchers to accomplish. As a result, researchers would often require assistance from ASSISTments team members to help with building their studies. This significantly impacted their ability to efficiently build the studies they desired. The ASSISTments team quickly recognized this problem, and began work on the E-TRIALS platform to alleviate these issues.

User Research & Prototyping

The initial design of the E-TRIALS began with a student Interactive Qualifying Project (IQP). This work was accomplished by Mr. Nicholas Krichevsky and Mr. Kamryn Spinelli advised by Dr. Neil Heffernan, Dr. Korinn Ostrow, and Mr. Ryan Emberling. Their work focused on redesigning the study building process (Krichevsky, Spinelli, 2020). To do this, they developed prototype software for building studies and researching user preferences.

To diversify the prototypes produced in this project, Mr. Krichevsky and Mr. Spinelli worked separately on their prototypes. Their goals were to represent and display the complex structure of a study more easily. They first constructed wireframe prototypes to display the study structure, shown in *Figure 1*. These graphics would help researchers visualize the flow of their study and understand how simple or complex it is. They also constructed prototypes for selecting, editing, and even creating problem sets for the study. Their prototypes also supported the pre-configuration of the study in order to provide further flexibility to researchers.

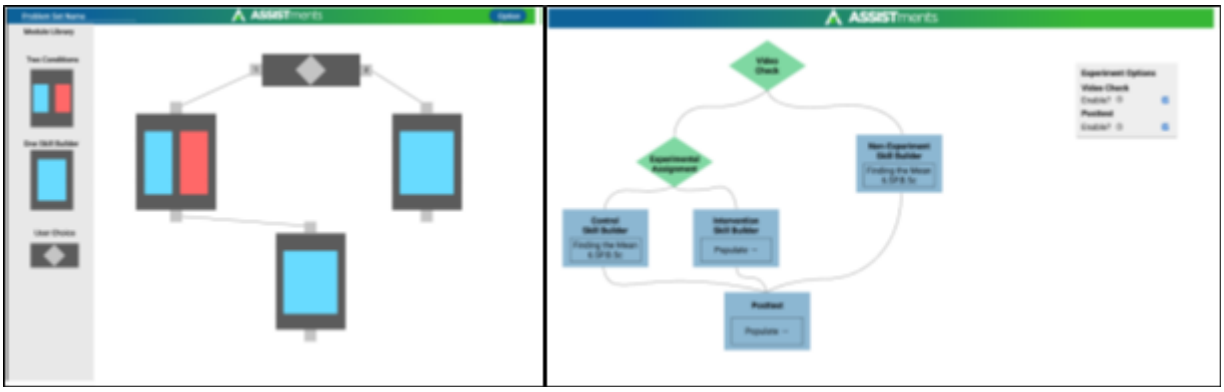

Figure 1: Two studies created in Mr. Krichevsky's prototype (left), and Mr. Spinelli's prototype (right).

Once these prototypes were constructed, the team organized user testing sessions within the ASSISTments organization to properly determine the strengths and weaknesses of their designs. Team members asked users to follow instructions to build a study using their interface. They observed that providing users with too much control over the study building process was overwhelming. This feedback was instrumental in constructing the finalized prototype.

The efforts of this project culminated in the construction of the final unified prototype, seen in *Figure 2*. This application was built using the Vue.js framework, and the Vuetify Material design framework. During the development, they also focused on testing and responsiveness of the app.

Figure 2: Homepage of the prototype E-TRIALS interface.

While the efforts of this project resulted in providing a finalized user interface for an improved study building experience, there was still remaining work to be done. This user interface still lacked essential functionality, as there was little to no back-end software supporting the project.

Full-Stack Implementation

E-TRIALS development continued in 2021 with an IQP by Mr. Timothy E. McCarthy and again advised by Dr. Neil Heffernan, Dr. Korinn Ostrow, and Mr. Ryan Emberling (McCarthy, 2021). During this time, several paid software engineers (such as Mr. Brian Rojas) joined the E-TRIALS development team to further improve the product. Their work improved the existing user interface and provided underlying back-end improvements that were vital for E-TRIALS to function. This project began evolving away from the prototype and towards a minimum viable product.

The first focus of this project was integrating the front-end web browser software with a server-side back-end. Initial iterations of the project had all study data stored within a web browser and provided no capabilities of retrieving study information beyond a local browsing

session. The integration of the front-end and back-end logic enabled users to have persistent data across multiple browsing sessions and multiple computers. This feature also would enable end-to-end testing improvements, as well as providing more interactive capabilities that were not as easily achievable beforehand.

Additionally, Mr. McCarthy and the rest of the E-TRIALS team improved the study building process by lowering the complexity of the study, enabling researchers to create their own study without the support of an E-TRIALS team member. This involved modularizing components of the study by allowing them to be toggled on and off. To implement this feature, they restructured the relationship between study data within the database, using the following Entity Relationship diagram shown in *Figure 3*. The team also implemented Table Definition Files (TDFs) to automatically generate the database tables and the corresponding code for accessing the new database tables. They also began heavily using the Java Spring framework to more easily implement web applications (VMWare, Inc., n.d.). These improvements increased development and maintenance time.

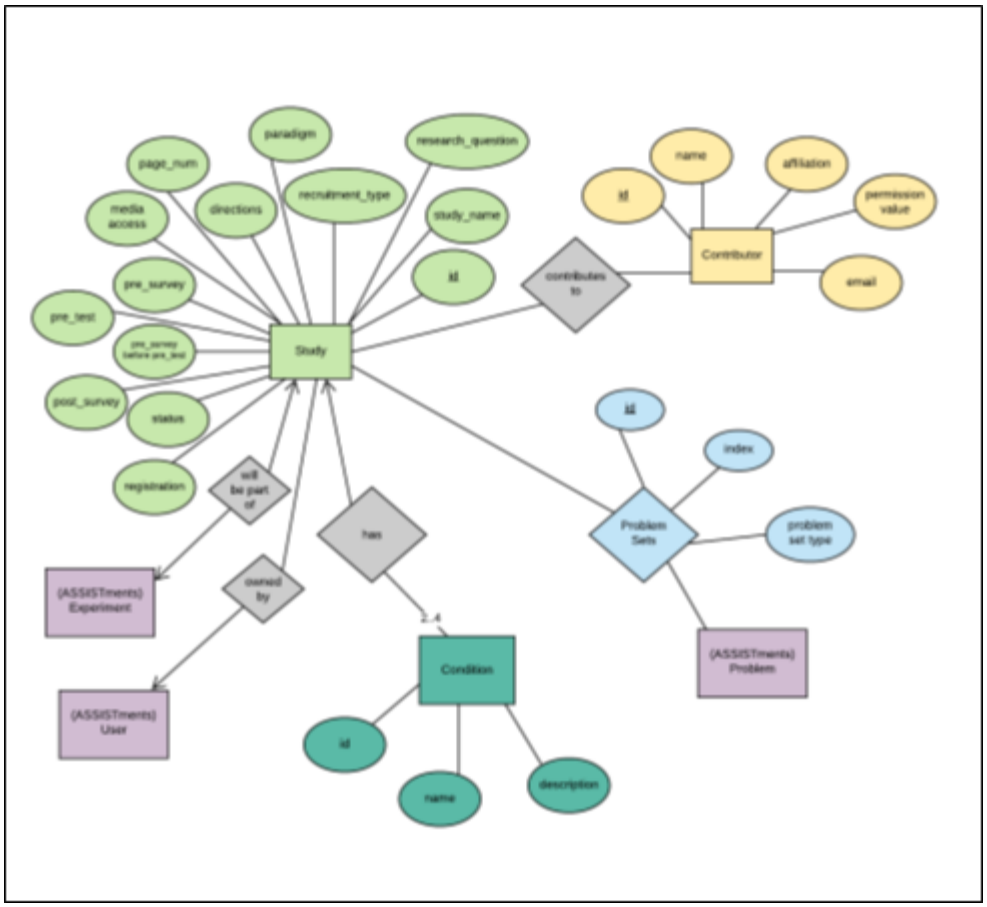

Figure 3: The Entity Relationship Diagram of the E-TRIALS tables

Lastly, Mr. McCarthy and the team set out to improve the problem set selection process, a critical piece of E-TRIALS' study builder. The main goal was to remove any level of uncertainty that could occur during the problem set selection phase by clarifying the interface. One major improvement for problem selection was adding functionality for filtering and selecting studies based on aggregate data. An example design of the problem set selection menu can be found below in *Figure 4*.

Figure 4: The content selection modals as designed in Figma.

This project pushed E-TRIALS closer towards the beginnings of the minimum viable product and accomplished many goals in terms of moving towards a full-stack implementation. However, by the time the project completed, the application was still unprepared to be released. Several interfaces were still a work in progress: most notably the build page and deploy page. Mr. McCarthy concluded his project by recommending the E-TRIALS add collaborative features and conduct more user testing.

Chapter 3. Methods

3.1 - Overarching Objectives

Implement and Fix Basic Functionality

In order for a proof of concept to become a viable product, it needs to correctly perform all of its essential functionality. E-TRIALS is no exception, and one of the original objectives of this project was to implement planned features. A working product was especially important to this project, as E-TRIALS' launch roughly coincided with the project's conclusion. While most functionality additions were handled by the product's professional developers, both team members also contributed over the course of the project.

Improve Interface Usability

As mentioned in the background, one of the primary visions of the E-TRIALS project is to make learning science research widely accessible. Low-cost project deployment, despite being the platform's main draw, is not the only factor. Another key aspect of accessibility is the usability of the software itself.

This project sought to both modernize the interface and avoid mistakes made by E-TRIALS' predecessor. The 1.0 Builder had several shortcomings that made it difficult to use and harder still to learn. The most notable of these was the organization of the application's interface, which in many places reflected its underlying implementation more than it did its users' expectations. As a result, the platform was difficult to understand for users without a deep knowledge of the underlying system.

As such, usability was a key concern for E-TRIALS. At the start of this project, many pages on the platform still used first-draft interface designs. This project redesigned key areas of the application to align with newer designs that incorporated feedback from early user testing. This included the study build page, a page that had been designated as especially critical being the place where researchers spend time writing out all of their student supports. This team improved the build page by making it easier to navigate within the page and by clearly indicating what the user has left to complete.

Ensure a Secure Product

Security was not originally a focus of this project. However, the team discovered early in the project that multiple study-related endpoints did not verify authorization for the authenticity of incoming requests. These endpoints included admin requests that, while not shown on the front-end unless the user was an admin, could be made by anyone regardless of the admin status. Another security issue included the OSF API key being hard-coded into the back-end codebase.

As a result, the team additionally worked to identify the extent of the vulnerabilities and comprehensively resolve them in a way that will prevent future vulnerabilities. Fortunately, there was no need to investigate for evidence of prior misuse, as at the time of fixing E-TRIALS had not been launched publicly.

Establish Robust Development Infrastructure

Another objective acquired over the course of the project was to improve the E-TRIALS team's development infrastructure. The project team members' roles as outsiders during the onboarding process made many existing issues readily apparent. While improving development infrastructure does not directly benefit the final product, the team thought it important for a variety of reasons. Better infrastructure makes it easier (and therefore faster) for future hires to join the team, it makes normal development efforts easier (and therefore faster), and it can ensure the product delivers with fewer bugs.

Strong typing is a feature of many modern programming languages and is valued due to its tendency to detect bugs before even running code (Cornell University, 2012). An example of one of these languages is TypeScript, a language that builds off JavaScript by adding static typing. As the E-TRIALS front-end currently uses JavaScript, it made sense to our team to transition to TypeScript in order to find current problems in the codebase and prevent issues in the future.

3.2 - Team Workflow

During this project, the E-TRIALS team was composed of Mr. Edward Philippo and Mr. Matthew Spofford working in concert with 7 full-time employees. This team managed the development process using the Agile Scrum methodology, and broke work into bi-weekly iterations known as "sprints." Every other Friday, the team conducted retrospective meetings to

discuss observed strengths and weaknesses from the last sprint. Then on the other Fridays, the team assigned the tasks for the upcoming sprint during a "sprint planning" meeting. These tasks, also known as "stories," were assigned by Ms. Hannah Pandolph (the product owner of E-TRIALS) and Mr. Emberling (the technical lead). To keep track of each task, the team used the project tracking software Jira. This tool enabled the team to organize tasks and keep track of who was working on it, its stage in development, details, comments, and related tasks. Tasks were also assigned "story points," numerical estimates of how many working days a task will take to complete. Following the Agile methodology, the team also gave status updates every day regarding our development work and had weekly "standup" meetings for more in-depth check-ins. However, due to the COVID-19 pandemic all of the status updates and meetings were accomplished online using Zoom and Slack. In order to more easily collaborate on the development process, the team used ASSISTments' existing GitHub organization, and E-TRIALS repositories, for easy maintaining and sharing code.

The E-TRIALS platform is split into two core components: the front-end and back-end. Both of these components were initially established through the previous efforts of Mr. Krichevsky, Mr. McCarthy, and Mr. Spinelli. The front-end, which provides the user interface of the application, is written using Vue.js, a "progressive framework" for building user interfaces for web browsers (Vue Team, n.d.). In order to deliver the user interface content to a user's web browser, the front-end relies upon a Node.js server for managing communication. The back-end server, which handles requests for data from the front-end, is written in Java using the Spring framework. The server can be run using the Tomcat servlet container and web server application. The back-end then obtains and stores user data using a PostgreSQL database. For the production, or enterprise, deployment of the software, the front-end and back-end servers were hosted using WPI's infrastructure, as well as recent infrastructure provided by Amazon Web Services.

Chapter 4. Results

4. 1 - Improving Functionality

During E-TRIALS' first round of user testing, researchers observed that they were unable to view any problem sets to add to their study. The problem set data never loaded for these new users; however, developers with established accounts had no issues loading problem set data. After some investigation, team members discovered that when a user provided their OSF ID during account creation to validate themselves as researchers, the front-end never attempted to load the problem set data afterwards. This meant that researchers would only be loading problem data once they have logged in for a second time. By simply loading the study data immediately after validating a researcher's OSF ID, the issue was resolved. *Appendix A* gives details on how this fix was implemented.

Related to OSF, researchers also found that if they modified their study that they created through E-TRIALS, the changes would not be reflected on their corresponding OSF project. This meant that their study title and research question would be completely dependent on what the researcher initially input when creating the study. This was a significant drawback, as information on OSF was now inconsistent. To resolve this, E-TRIALS simply needed to send these changes using the OSF API whenever these fields were modified. For more information on the implementation details, refer to *Appendix B.*

In order to facilitate content moderation on the platform, the administrative user role was added. These users would have access to all available studies regardless of the stage of development. Unfortunately, since these admin users view these studies in the exact same manner the researchers would, the hint and problem set removal buttons were still accessible. This enabled admins to be able remove data from a researcher's study, effectively eroding the integrity of a researcher's study. This behavior was clearly undesired, as it is vital to ASSISTments for upholding quality research. To resolve this issue, the removal buttons were disabled for admin users. More details on the implementation can be found in *Appendix C*.

4. 2 - Build Page and General Usability Improvements

Build Page Overhaul

The biggest usability change was restructuring the build page. The original version had two main issues: it was difficult for the user to see at a glance which problem part they were working on, and it was not possible for the user to discern which problem parts they had finished and which needed more work. This original interface is shown in *Figure 5*.

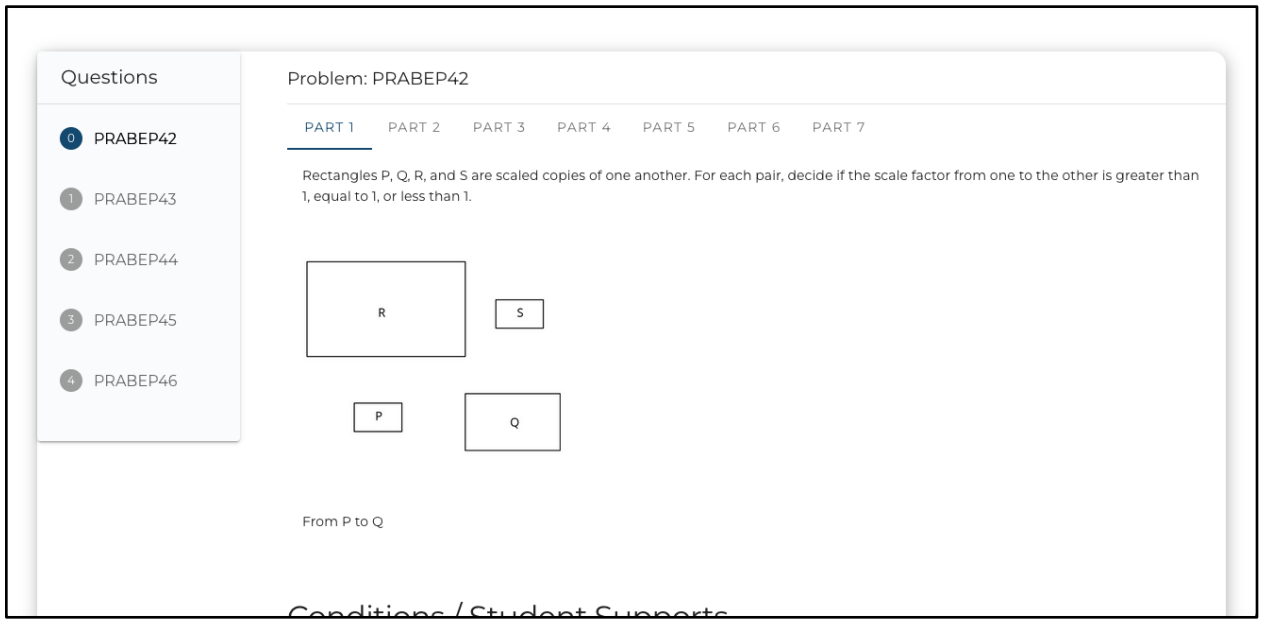

Figure 5: the build page before this project's improvements

This project addressed these issues by making several changes to the left-side stepper. First, the problem part tabs (shown across the top as PART 1, PART 2, etc.) were removed and incorporated as sub-steppers within the problem stepper. Secondly, the stepper icons were changed so that instead of showing the index of each item, they show whether a given item is complete, partially complete, or not started. Finally, there were various small improvements like consistently using alphabetic problem part identifiers (e.g. "Part A") and flattening the page instead of using a card display to be consistent with the rest of the application. The build page after these changes is on display in *Figure 6*.

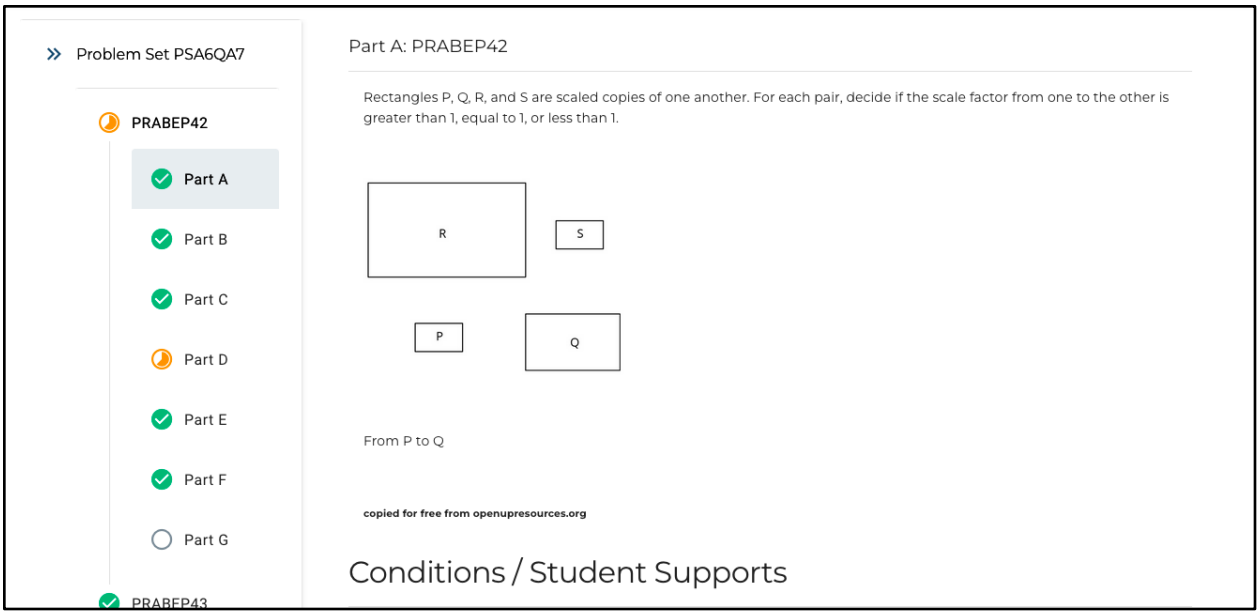

Figure 6: the build page featuring improvements from this project

The stepper redesign makes it clear what problem parts are being edited (the highlighted section), and the user can also clearly see which problem parts they have already completed and which need to be finished. In addition, the stepper collapses all problems except the active one, preventing it from taking up too much space while still providing all information the user needs. *Figure 7* shows the stepper with various selected problems.

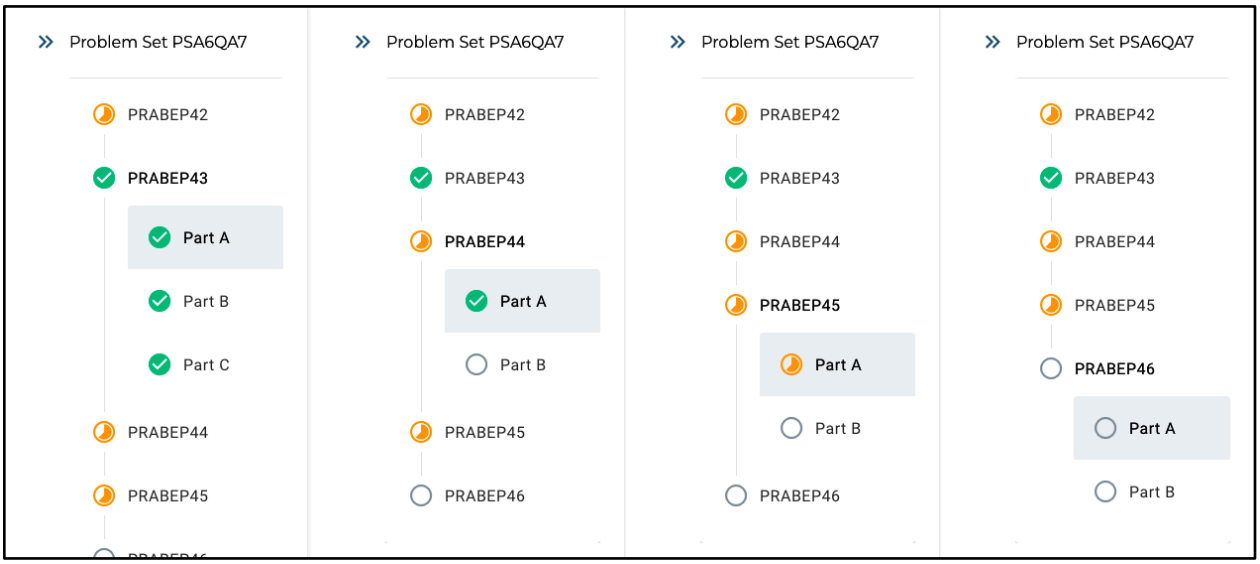

Figure 7: viewing different problems with the problem set stepper

Assorted Improvements

In addition to the build page, the team also contributed many assorted usability improvements across the entire application. These changes are listed individually below.

When signing in for the first time, a modal asks users for their OSF ID in order to link their E-TRIALS account with their OSF account. To improve this process, the OSF sign-in modal was changed to also accept a user's OSF URL (for example, `osf.io/abcde`) in addition to their OSF ID (which would be just `abcde`). While the instructions in *Figure 8* clearly state to to use an id, this should help users who inadvertently paste their URL instead.

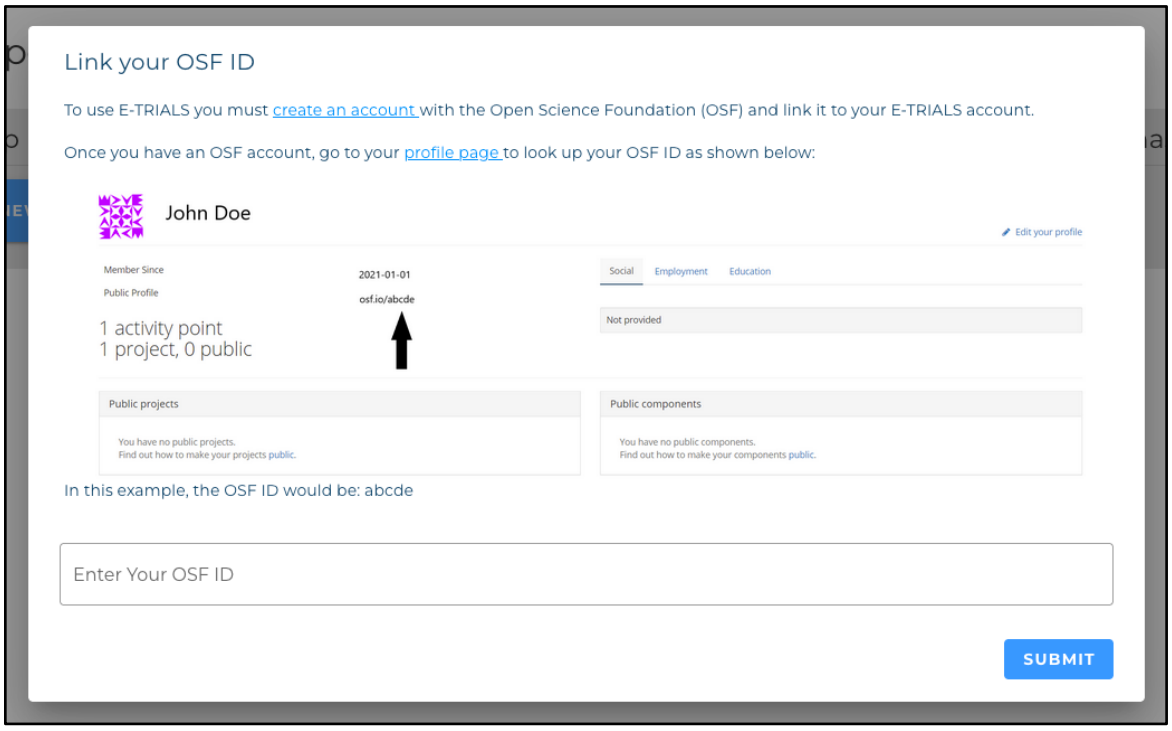

Figure 8: the OSF sign-in modal that accepts either an ID or a URL

When on the study page, a general feeling of bugginess and unresponsiveness was resolved by making the app wait to render page contents and the study stepper until after the study content has loaded. This prevents the user from seeing fields that start with missing information and have that information slowly pop into the screen as it loads. Instead, everything loads in the background and then it appears to the user all at once.

The existing study content editor was tweaked to enable a math editor so it can construct mathematical symbols. As such, researchers can now write hints and explanations with mathematical notation more easily.

The problem set selection modal was adjusted to remove the `isSkillbuilder` column. Instead, problem sets are now filtered by whether or not they are skillbuilders depending on the paradigm of the current study. For Mastery Learning studies, only skillbuilder problem sets are shown, but for Linear Complete All studies, skillbuilder problems are excluded.

In the admin view, the deploy buttons for each study were changed to be disabled unless the study in question is in the "Awaiting Approval" state. To keep the study deployment process apparent, the buttons are still visible. However, they cannot be clicked, so administrators no longer have the option to deploy studies that are not ready to be deployed.

The study stepper was changed to visually match the rest of the application and to more clearly reflect a user's progress in the study. The new stepper (shown in *Figure 9*) has icons instead of numbers, and also reflects the application's new page structure.

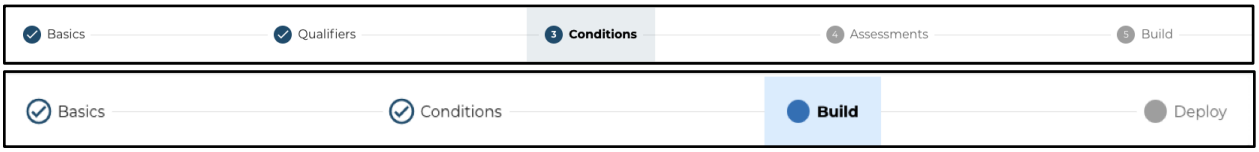

Figure 9: old study stepper (top) contrasted with new study stepper (bottom).

Finally, the study preview diagram was removed from the study user interface. Designed and implemented by previous projects, this diagram was originally implemented to visualize the structure of complicated study formats. However, the E-TRIALS team concluded that the diagram preview was no longer necessary due to the simplicity of the study types currently available on the platform.

4.3 - Back-End Security Fixes

Endpoints Authorized for Owners/Admins

With users having the ability to create and own specific studies that are created on the E-TRIALS platform, this enables a significant amount of convenience for users. The front-end enabled users to access these studies directly through a corresponding URL. However, during development Mr. Emberling observed a flaw in the implementation where the back-end handled these requests. He found that other users, regardless of ownership status, were still able to access another user's study through these URLs. This was a major security issue because the back-end was unintentionally enabling users to access and modify studies they did not own. This would

provide malicious attackers with the opportunity to change or delete information with E-TRAILS studies.

It was decided that the best approach would be to modify the back-end such that it rejected requests for viewing, modifying, and removing from studies if the user did not have study ownership. Administrative users would also be given access to specific back-end endpoints involved in viewing study content. This meant that administrative users would not have the authorization for modifying studies. However, this conflicted with the feature in the following section.

User Role Permissions for Studies Phases

After discovering the authorization issue for administrative and study-owning users, Ms. Pandolph decided it would be useful to dynamically change study authorization depending on the study's current phase of development. Studies have the following phases:

- Development - The researcher is still working on the study, and not ready for review.
- Awaiting Approval - The researcher has considered their study content ready for deployment, and needs review from administrative users.
- Deployment - The study has been made available to students, and data is being collected.
- Disseminate - The data has been collected, and is ready for researchers to analyze.

Due to these various phases of development, Ms. Pandolph determined that the user experience could be improved by limiting the type of action that could be conducted on a study, depending on the phase. These action types include viewing, editing, and removing from a study. An example of this was to revoke study owners' editing and removal capabilities once the study has been deployed. This would be the opposite for administrative users, in which they would keep their editing rights for deployed studies. This would indirectly require study owners to receive approval from administration to make modifications to their study while in deployment. The entire set of action rights for each study phase are described in *Table 1*.

Study Phase	**Study Owner Actions**	**Administrator Actions**
Development	View, Edit, Remove	View, Edit
Awaiting Approval	View, Edit	View, Edit
Deployment	View	View, Edit
Disseminate	View	View

Table 1: Authorized actions for study owners and administrators based on the study phase.

In order to implement this functionality, the first step was examining where the back-end handles data requests for studies. Where each web-request endpoint was being handled, some requests were involved in interacting with data that was indirectly related to a study. This included accessing study conditions, and contributor information. These endpoints were also not designed for differentiating the type of actions (e.g. viewing, editing, and removing) a request was conducting. Each request would also require an additional step for validating the authorization for a specific endpoint. To accomplish all of this, representations for data relationships to the study, and the action types were created. A new routine was also defined for validating permissions. All of the details regarding these additions can be found in *Appendix D*.

For the implementation of the permission validation routine, administrator and owner roles would need to be represented in the software. This representation would also be maintaining a list of possible study actions that a role could conduct for each phase of a study. This enabled the authorization routine to properly determine if the action a user is conducting is valid. The routine would examine the roles that a user held, and then "pool" together all of the valid actions for each role at the current study phase. Then, if the action the user was conducting occurred in the pool, then the action was considered authorized. *Appendix E* contains additional technical detail regarding how this was implemented.

Enable Specific Endpoints for Local Environments

During development and testing of features, there would be many occasions in which the tester or developer would create multiple studies on their E-TRIALS account in order to validate specific functionality. As a result, this would leave testers with multiple unnecessary studies authored by them that would never be used. In order to fix this, specific web-request endpoints

were developed on the back-end for conducting more efficient testing. These endpoints provided additional features including deleting all studies owned by a user, as well as truncating study data (which has since been deprecated). This enabled developers and testers to clean up and remove any studies that were no longer needed.

While these endpoints were useful in enabling a more efficient testing process, these powerful features were now easily accessible and vulnerable to potential attackers on the internet. To combat this security issue, Mr. Emberling decided that these web endpoints should only be accessible when a developer or tester is locally running the back-end on their own devices. This would prevent unwanted attackers from accessing these endpoints through the internet.

To fix this issue properly, the easiest solution was to rely upon the Spring framework's profile feature. Spring provides developers the opportunity to enable specific components or features depending on what profile it is configured in. E-TRIALS back-end implementation of Spring requires that the production server for the back-end use a "production" profile. When a local version of the back-end is run locally on an individual's device, the back-end uses the "development" profile. Since E-TRIALS is configured to differentiate between the local and production environments, these testing endpoints can simply be enabled only for the "development" profile. Conversely, this would disable any of these testing functionality on the production E-TRIALS server. For the implementation details, refer to *Appendix F*.

Remove OSF API Key from Version Control

Early in the development process, E-TRIALS integrated with OSF using their API. However, the only means of using the OSF API was by creating an account on their platform. Users of that account would then be able to generate an API "key", essentially a one time password, that would be associated with their account. This API key would then need to be continually sent to OSF whenever E-TRIALS is sending web-requests, in order to act for that account owner. As a result of needing to repeatedly use the API key, one developer eventually wrote their key directly into the back-end code for easier use, and therefore saved it within version control on E-TRIALS' GitHub repository.

While it was useful saving the OSF API key within the code, this resulted in several drawbacks. One concern was that by using one API key for everything, every developer and the

production server would be using the exact same OSF account which did not directly belong to them. This also was a significant security flaw, because the API key for this account was shared with anyone who had access to the back-end codebase. This meant that anyone, whether it be another E-TRIALS developer or a malicious attacker, could directly use someone else's API key for different purposes with OSF. To repair this issue, the OSF API key was removed from the back-end directly. Relying upon the Java Spring framework and reconfiguring the Tomcat server, developers could then specify their own API keys which would be saved outside of the codebase. This provided a best of both worlds situation, in which developers could repeatedly reuse their API key, while also being able to keep it separate from the E-TRIALS codebase. More details on this implementation can be found in *Appendix G*.

4.4 - Development Infrastructure and TypeScript

With the E-TRIALS platform becoming significantly more complex since it started, and with multiple types of data being handled at a time, the front-end codebase needed a change in its underlying infrastructure. Since the front-end was written in JavaScript, this became particularly confusing over time due to it being a weakly-typed language. As a result, this meant that objects did not have to meet a particular type or shape in order to be passed to functions. This also meant that because types were always unclear, it was impossible to determine what properties and methods an object had attached to it. This confusion impacts development efficiency, and makes it more difficult to implement new features.

To combat this confusion and improve the front-end infrastructure, effort has begun the long process of migrating to TypeScript (Microsoft Corporation, n.d.). This new language is strongly-typed and built directly upon JavaScript, providing static type checking built right into the language itself. Its goal is to provide similar benefits as other statically typed languages such as Java, which is used for the back-end software. Using this language would enable a clearer relationship between the back-end, and front-end data structures. One such example of this is the various data structures that were created on the front-end for handling problems and problem sets. With multiple unique data transfer objects, as well as transformed variants of these structures, it makes it difficult for developers to determine which structure might be in use in a specific front-end state or function. TypeScript removes this confusion entirely by explicitly defining what properties these structures contain. While this language adoption is important for

enabling a more successful developer experience, this implementation is still a work in progress, and will be completed post-project. For more details on how these new problem types were implemented, see *Appendix H*.

Chapter 5. Discussion

This project made several significant contributions to the E-TRIALS platform. These included broadly applicable bug fixes and additional functionality, usability improvements to the build page and across the application, authorization restrictions and local-testing-only endpoints, and implementing development best practices such as moving to TypeScript.

5.1 - Significance of Our Work

With E-TRIALS being a first of its kind educational research platform, publicly releasing this tool was a huge step for enhancing the educational research community. In order for the platform to reach the same level of quality and usability as an enterprise application, this project heavily focused on improving functionality. Resolving issues such as data not loading properly, or users having access to functions that weren't intended, are necessary for a clean and effective user experience. Without these functional improvements and bug fixes of general issues, the E-TRIALS platform would not have been released with the same degree of quality.

On top of resolving bug fixes, improving usability was also necessary to release the application. By completing the implementation of the redesigned study build page, this greatly enhanced the user experience. With the new layout, users are now easily able to determine what they were editing on the page, and also more clearly determine what a user still needs to finish completing for their student supports. This also made the build page more consistent with the rest of the platform design, offering more appealing visuals. In addition to the build page, improving the overall clarity and responsiveness of the E-TRIALS application was equally as important for the release of the application. These general improvements enabled E-TRIALS to have a more professional appearance, and enabled users to more easily understand how to use the platform.

While improving usability and appearance is important, security is vital for publicly releasing an application. As E-TRIALS has become more complex over time and is storing more data, maintaining privacy and securing user data is critical. Without secure systems in place, malicious individuals could misuse, modify, or corrupt user data without the necessary permissions. By implementing security fixes for E-TRIALS' web-endpoints, and confirming a user's authorization, this tightens the application security significantly. By being able to keep information safer, researchers will feel more comfortable using the application.

Finally, the transition of E-TRIALS to an enterprise level application also similarly requires that the application infrastructure improve. The team made efforts on this front to transition the front-end software towards the strongly-typed language TypeScript, which includes static typing. Many issues were encountered in the past as a result of a lack of static typing, which significantly hindered development time. Certain bugs more easily appeared as a result of simple typos or old code that would have been easily noticed with this change in infrastructure. While this change in infrastructure does not affect the end-user's directly, this is a significant quality of life improvement for the development team.

5.2 - Future Work

While E-TRIALS has just launched its minimum viable product, there is still a lot of work to do before it can be used at a larger scale. This work is split into the following main focuses: improving the usability and functionality of the product, expanding the product to allow other types of research, user recruitment, and platform infrastructure.

Improving the functionality and usability of the platform is key to its long-term success. By polishing functionality and usability, this will provide users with a more effective user experience, and in turn keep users confidence in the platform. Adding new features to the platform will also help to strengthen the platform, as well as entice more users. This includes implementing features such as previewing study supports as a student, problem set filtering, best so far conditions, and common wrong answers. In addition, formal user testing and informal post-deployment feedback would be a great resource to improve user interfaces and ensure the platform is both intuitive for new users to pick up and efficient for returning users to work with. This would also heavily rely upon strengthening user recruitment. By bringing more users into E-TRIALS, this enables missing features that have not been discovered to be brought to light. New users will also provide the development team more opportunities to polish functionalities or issues that had previously gone unnoticed.

Another next step would be to expand the capabilities of the platform. Currently, E-TRIALS only supports Support Comparison and Problem Varied studies. By broadening these offerings to include other study paradigms, E-TRIALS would be more helpful to its current users and could attract additional users whose needs are not currently met. While likely more

impactful than routine improvements, an undertaking of this nature would require careful planning and design to ensure it would operate well with the existing system.

Finally, improving the overall software infrastructure is important for scaling up the platform. For example, a more robust testing infrastructure would greatly benefit E-TRIALS by better detecting bugs, which would in turn make fixing them easier and cause the end product to deliver with fewer technical problems. Another infrastructure improvement includes continuing the process of migrating the front-end to TypeScript. Enabling static typing will also for a smoother developer experience, and prevent the platform from being less error prone.

5.3 - Final Thoughts

In addition to our contributions to the E-TRIALS platform, both team members are proud to report this project was a valuable learning experience. We learned new skills and honed existing ones in a practical setting, all the while knowing our changes would eventually make a difference to the product's users. We are particularly glad to have worked on a serious product as part of an established development team: these factors made this project a fantastic insight into the world of professional software development.

All of this–both the work and the personal development–would not have been possible without constant and rapid help from Mr. Emberling, starting on day one and going far beyond his obligations of engineering work. We also wish to specially thank Mr. Rojas and Ms. Alibasic for their support and review over the last few months. From engineers to engineers–thank you!

E-TRIALS is in good shape and in good hands as it launches its MVP, and we are thankful to have had the opportunity to contribute. We look forward to hearing about its future success!

References

Center for Open Science. (n.d.). *The Open Science Framework*. Retrieved February 26, 2022, from https://www.cos.io/products/osf

Cornell University. (2012, Spring). *Safety and strong typing*. Strong versus Weak Typing. https://www.cs.cornell.edu/courses/cs1130/2012sp/1130selfpaced/module1/module1part4/strongtyping.html

Koedinger, K. R., Booth, J. L., & Klahr, D. (2013). *Instructional Complexity and the Science to Constrain It*. Science, 342(6161), 935–937. https://doi.org/10.1126/science.1238056

Krichevsky, N. J., & Spinelli, K. P. (2020). *E-TRIALS: Developing a Web Application For Educational Research*. Worcester Polytechnic Institute; Digital WPI. https://digital.wpi.edu/show/wh246v51c

McCarthy, T. E. (2021). *Continuing the Development of E-TRIALS*. Worcester Polytechnic Institute; Digital WPI. https://digital.wpi.edu/show/n583xx730

Microsoft Corporation. (n.d.). *Handbook—The TypeScript Handbook*. Retrieved March 1, 2022, from https://www.typescriptlang.org/docs/handbook/intro.html

VMware, Inc. (n.d.). *Why Spring?* Retrieved January 30, 2022, from https://spring.io/why-spring

Vue Team. (n.d.). *Introduction—Vue.js*. Retrieved January 30, 2022, from https://vuejs.org/v2/guide/

Appendix

Appendix A: Load Problem Data after OSF ID Validation in HomeOsfModal.vue

```
…
passOsfId() {
        // This strips the ID of anything before the last forward slash
        const delim = '/';
        [this.submittedOsfId] =
                    this.submittedOsfId.split(delim).slice(-1);

        this.loadingSubmitted = true;
        this.$store.dispatch('checkOsfId', this.submittedOsfId)
               .then(() => {
                      this.$store.dispatch('getContent');

                      this.loadingSubmitted = false;
                      // show the modal with invalid state
                      this.isValidOsfId = true;
               })
               .catch(() => {
                      this.showSnack = true;
                      this.isValidOsfId = false;

                      this.loadingSubmitted = false;
               });
 },
…
```

Appendix B: Updating OSF Project Based on Changes in Study

OSFManager.java

```
…
import org.assistments.domain.exception.NotFoundException;
…
import org.assistments.etrials.domain.Study;
…
import org.assistments.etrials.web.StudyController.UpdateStudyRequestBody;
…
import org.json.JSONException;

public interface OSFManager
{
  // OSF Integration V2
 …
  public boolean updateProject(UpdateStudyRequestBody requestData, Study currStudy)
      throws JSONException, NotFoundException;
```

```
}
```

OSFManagerImpl.java

```
…
import org.apache.http.client.methods.HttpPatch;
…
import org.assistments.etrials.web.StudyController.UpdateStudyRequestBody;
…

@Component
public class OSFManagerImpl implements OSFManager
{
  …

 /*
  Function to update our Project on OSF
  @param UpdateStudyRequestBody consists of the studyId, the field name to be
updated, and the new value
  @param Study used to get the study id from the currently built study in E-TRIALS
  @return boolean, stating whether it was successful in updating the project
   */
  @Override
  public boolean updateProject(UpdateStudyRequestBody requestData, Study currStudy)
      throws JSONException,NotFoundException
  {
    // Obtain OSF project information corresponding to the given study
    String osfProjectID = this.studyProjectDao.findObject(

StudyProjectDao.Field.ETRIALS_STUDY_ID.getQueryTerm(currStudy.getId())).getOsfProject
Id();

    // If an OSF project ID has not been assigned, then it cannot be updated
    if (osfProjectID == null)
    {
      return false;
    }

    // Create our JSON Body
    JSONObject body = new JSONObject();
    JSONObject data = new JSONObject();
    JSONObject attributes = new JSONObject();

    data.put("type", "nodes");
    data.put("id", osfProjectID);

    switch (requestData.getFieldname())
    {
      case "studyName":
        attributes.put("title", requestData.getValue());
        break;
      case "researchQuestion":
        attributes.put("description", requestData.getValue());
        break;
```

```
    default:
      return false;
  }

  data.put("attributes", attributes);
  body.put("data", data);

  // Create the URL
  List<String> endpoints = new ArrayList<>();
  endpoints.add("nodes");
  endpoints.add(osfProjectID);

  String finalURL = urlBuilder(url, endpoints);

  // PATCH DATA
  try
  {
    HttpResponse response = callPATCH(finalURL, body.toString());
    if (response.getStatusLine().getStatusCode() == 403)
    {
      return false;
    }
    else
    {
      String responseString = new String();
      HttpEntity entity = response.getEntity();

      if (entity != null)
      {
        responseString = EntityUtils.toString(entity);
      }

      // Determine if updating data failed, and output errors that occurred
      JSONObject returnedJson = new JSONObject(responseString);
      if (returnedJson.has("errors"))
      {
        return false;
      }

      return true;
    }
  }
  catch (Exception e)
  {
    // Do nothing with the exception
  }

  return false;
}
…
 /*
Patch Rest function to make life easier
@param String: The URL of our PATCH function
@param Body: The data we are updating, reference the OSF Dev Guide
 */
```

```
  private HttpResponse callPATCH(String url, String body)
    throws InterruptedException, ClientProtocolException, IOException, Exception
  {
    HttpPatch httpPatch = new HttpPatch(url);
    StringEntity entity = new StringEntity(body);
    httpPatch.setEntity(entity);
    httpPatch.setHeader("Authorization", token);
    httpPatch.setHeader("Content-type", "application/json");
    Exception failure = null;

    int retryCount = 0;
    do
    {
      try
      {
        CloseableHttpResponse httpResponse = httpclient.execute(httpPatch);
        return httpResponse;

      }
      catch (Exception timeoutException)
      {
        ++retryCount;
        failure = timeoutException;
        continue;
      }
    }
    while (retryCount < 1);
    throw failure;
  }
  …
}
```

StudyManagerImpl.java

```
…

@Component
public class StudyManagerImpl implements StudyManager
{
  …

 /**
   * Updates study in the database
   *
   * @param  requestdata       consists of the studyId, the field name to be updated,
and the new value
   *
   * @return                   the updated study
   *
   * @throws NotFoundException thrown if studyId does not exist
   */
  @Override
 public Study updateStudy(UpdateStudyRequestBody requestdata) throws
NotFoundException
```

```
  {
    Study study = new Study();
    study = this.findStudyById(requestdata.getStudyId());

    switch (requestdata.getFieldname())
    {
      case "studyName":
        study.setStudyName((String) requestdata.getValue());
        osfMgr.updateProject(requestdata, study);
        break;
      case "researchQuestion":
        study.setResearchQuestion((String) requestdata.getValue());
        osfMgr.updateProject(requestdata, study);
        break;
      case "pageNumber":
        study.setPageNumber((int) requestdata.getValue());
        break;
      case "recruitmentTypeNumber":
        study.setRecruitmentTypeNumber((int) requestdata.getValue());
        break;
      case "experimentalParadigmNumber":
        study.setExperimentalParadigmNumber((int) requestdata.getValue());
        break;
      case "mediaAccess":
        study.setMediaAccess((boolean) requestdata.getValue());
        break;
      case "directions":
        study.setDirections((boolean) requestdata.getValue());
        break;
      case "postTest":
        study.setPostTest((boolean) requestdata.getValue());
        break;
      case "preTest":
        study.setPreTest((boolean) requestdata.getValue());
        break;
      case "preSurvey":
        study.setPreSurvey((boolean) requestdata.getValue());
        break;
      case "postSurvey":
        study.setPostSurvey((boolean) requestdata.getValue());
        break;
      case "preSurveyBeforePreTest":
        study.setPreSurveyBeforePreTest((boolean) requestdata.getValue());
        break;
      case "registration":
        study.setRegistration((int) requestdata.getValue());
        break;
      case "status":
        study.setStatus((int) requestdata.getValue());
        break;
      default:
        System.out
          .println("failed on updateStudyField, can't find attribute
".concat(requestdata.getValue().toString()));
        break;
```

```
    }
    studyDao.update(study.getId(), study);

    return study;
  }
}
```

Appendix C: Disabling Removal Buttons for Admins

StudyContentEditor.vue

```
…
       <v-btn
              v-if="isAdmin"
              id="remove-hint-button"
              text
              color="secondary"
              @click="removeHint()"
       >
              - Remove Hint
       </v-btn>
…
import auth from '../store/modules/auth';

export default {
       name: 'StudyContentEditor',
       …
       computed: {
              …
              isAdmin() {
                     return auth.state.isAdmin;
              },
       }
       …
}
```

ViewChosenProblems.vue

```
…
       <v-btn
              v-if="isAdmin"
              contained
              color="secondary"
              @click="removeSelectedProblems"

       >
              Remove Problem Set
       </v-btn>
…
import auth from '../store/modules/auth';

export default {
```

```
        name: ‘ViewChosenProblems’,
        …
        computed: {
                …
                isAdmin() {
                        return auth.state.isAdmin;
                },
        }
        …
}
```

Appendix D: Additions to User Permissions in StudyManager.java

```
public interface StudyManager
{
  /**
   * The relationships an Id could have with a study.
   */
  public static enum IdStudyRelationship
  {
       CONDITION,
       CONTRIBUTOR,
       STUDY
  }

  /**
   * Actions that could be conducted on a study.
   */
  public static enum StudyAction
  {
       EDIT,
       REMOVE,
       VIEW
  }

  …

  public void authorizeStudyAction(int id,
                                   IdStudyRelationship idRelation,
                                   StudyAction operation)
       throws AuthorizationException,
            NotFoundException,
            IllegalStateException;
}
```

Appendix E: Additions to User Permissions in StudyManagerImpl.java

```
@Component
public class StudyManagerImpl implements StudyManager
{

  @Autowired
  private ConditionManager condMger;
```

```
  @Autowired
  private ContributorManager contMger;
  …
  // Defines the authorized study actions for admin users depending on the following
study phases
  private static final StudyAction[] ADMIN_DEVELOPMENT_ACTIONS = new StudyAction[]
{StudyAction.VIEW, StudyAction.EDIT};

  private static final StudyAction[] ADMIN_APPROVAL_ACTIONS = new StudyAction[]
{StudyAction.VIEW, StudyAction.EDIT};

  private static final StudyAction[] ADMIN_ONGOING_ACTIONS = new StudyAction[]
{StudyAction.VIEW, StudyAction.EDIT};

  private static final StudyAction[] ADMIN_COMPLETE_ACTIONS = new StudyAction[]
{StudyAction.VIEW};

  private static final StudyAction[] OWNER_DEVELOPMENT_ACTIONS =
       {StudyAction.VIEW, StudyAction.EDIT, StudyAction.REMOVE};

  private static final StudyAction[] OWNER_APPROVAL_ACTIONS = new StudyAction[]
{StudyAction.VIEW, StudyAction.EDIT};

  private static final StudyAction[] OWNER_ONGOING_ACTIONS = new StudyAction[]
{StudyAction.VIEW};

  private static final StudyAction[] OWNER_COMPLETE_ACTIONS = new StudyAction[]
{StudyAction.VIEW};

  /**
   * Each value corresponds to a possible authorization role (admin, study owner,
etc.). It is defined with a list
   * for each status state of the study (in development, awaiting approval, etc.).
That array is made up of an array of
   * {@code StudyAction}'s, which are the valid actions (view, edit, delete) that can
be conducted by that role at the
   * given study state.
   * <br/>
   * This means that for a given role, that type of user will only be authorized for
a specific type of
   * action depending on what the current study state is. This validation is
performed by {@code authorizeAction}.
   */
  private static enum RoleAuthorization
  {
       ADMIN(ADMIN_DEVELOPMENT_ACTIONS, ADMIN_APPROVAL_ACTIONS,
ADMIN_ONGOING_ACTIONS, ADMIN_COMPLETE_ACTIONS),

       OWNER(OWNER_DEVELOPMENT_ACTIONS, OWNER_APPROVAL_ACTIONS,
OWNER_ONGOING_ACTIONS, OWNER_COMPLETE_ACTIONS);

       /**
       * Defines the authorized operations for all study statuses of the
authorization level.
```

```
	*/
	private final StudyAction[][] validStatusOperations;

	/**
	* Constructs a role's authorized operations. The row index corresponds to the
status level
	* of a study, and each row contains the authorized operations for that status.
	*
	* @param validStatusOperations Defines the authorized operations for study
statuses.
	*/
	RoleAuthorization(StudyAction[]... validStatusOperations)
	{
		this.validStatusOperations = validStatusOperations;
}

/**
 * Using the given study, determine if the current role is authorized to conduct the
given action.
 *
 * @param  studyStatus           Corresponds to the study status.
 * @param  action                The action that is trying to be conducted on a
study.
 *
 * @throws AuthorizationException Thrown if the user is not authorized for the given
operation.
 * @throws IllegalStateException  Thrown if the study status value is invalid.
 */
public static void authorizeAction(Set<RoleAuthorization> roles, int studyStatus,
StudyAction action)
	throws AuthorizationException, IllegalStateException
  {
	// For each role being checked, keep track of all of the valid operations
	final Set<StudyAction> validOperations = new HashSet<>();
	for (final RoleAuthorization authorization : roles)
	{
			try
			{
			// Obtain the authorized operations for the given
			// study status
			List<StudyAction> operationsList =
				Arrays.asList(authorization.validStatusOperations[studySta
				tus]);
			validOperations.addAll(operationsList);
			}
			catch (ArrayIndexOutOfBoundsException e)
			{
			// Only occurs if the study status provided was larger
			// than the validStatusOperations array
			throw new IllegalStateException(
				"Invalid study status given and cannot be authorized.");
			}
	}

	// Verify that the given operation is authorized for the given
```

```
        // study status
        if (validOperations.contains(action))
        {
                return;
        }

        // Authorized operation was not found for the given phase
        throw new AuthorizationException(
                "User does not have access to this study operation.");
        }
  }

  /**
   * Confirms the authorization of a study by determining whether the current user is
authorized
   * to conduct the given operation. If the user is not authorized for the operation,
an
   * {@link AuthorizationException} is thrown. This is dependent on whether or not
user is an administrative user/owns
   * the study, as well as the current status of the study. See {@link
RoleAuthorization} for more details
   * on which operations a role is authorized for depending on the status.
   * <br/>
   * The study is obtained through the given Id, and how it is related to the desired
study.
   * <br/>
   * E.g.
   * A contributor Id is given. The corresponding study Id is then obtained from the
contributor,
   * allowing the authorization to be processed.
   *
   * @param  id                  Id related to the study.
   * @param  idRelation          Defines the relationship the study has with the
given Id.
   * @param  action              Defines the action that is trying to be conducted on
a study.
   *
   * @throws AuthorizationException Thrown if the user is not authorized for the
given operation.
   * @throws NotFoundException   Thrown if the given Id or related study is not
found.
   * @throws IllegalStateException  Thrown if provided Id relationship or study
status value is invalid.
   */
  @Override
  public void authorizeStudyAction(int id, IdStudyRelationship idRelation,
StudyAction action)
        throws AuthorizationException, NotFoundException, IllegalStateException
  {
        // Using the given id-to-study relationship, obtain the desired study
        Study currStudy = null;
        switch (idRelation)
        {
        case CONDITION:
        Condition currCond = condMger.findConditionById(id);
```

```
        currStudy = findStudyById(currCond.getStudyId());
        break;
        case CONTRIBUTOR:
        Contributor currContrib = contMger.findContributorById(id);
        currStudy = findStudyById(currContrib.getStudyId());
        break;
        case STUDY:
        currStudy = findStudyById(id);
        break;
        default:
        throw new IllegalStateException("Invalid Id relationship to the study was
given.");
        }

        // If possible, setup the necessary role authorization depending if the user
is an admin/owner
        final Set<RoleAuthorization> roles = new HashSet<>();

        // Confirm whether or not the current user has admin capability
        if (AuthenticationHolder.hasRole(GlobalRoleType.PARTNER_ADMIN))
        {
                roles.add(RoleAuthorization.ADMIN);
        }

        // Determine if the current user owns the given study
        String ownerXref = currStudy.getOwnerXref();
        String userXref = AuthenticationHolder.getCurrentUserXrefStr();
        if (ownerXref.equals(userXref))
        {
                roles.add(RoleAuthorization.OWNER);
        }

        // The user is not authorized if no valid study role is detected
        if (roles.isEmpty())
        {
                throw new AuthorizationException("User does not have access to this
                study.");
        }

        // Determine if the admin/owner of the study is authorized to do
        // the given operation
        RoleAuthorization.authorizeAction(roles,
                                                currStudy.getStatus(),
                                                action);
  }
  …
}
```

Appendix F: Environment Specific Features in StudyManagerImplLocal.java

```
package org.assistments.etrials.manager;

import java.util.List;
import org.assistments.domain.exception.NotFoundException;
```

```
import org.assistments.etrials.dao.ConditionDao;
import org.assistments.etrials.dao.ContributorDao;
import org.assistments.etrials.dao.StudyDao;
import org.assistments.etrials.domain.Study;
import org.assistments.service.security.authentication.core.AuthenticationHolder;
import org.springframework.context.annotation.Primary;
import org.springframework.context.annotation.Profile;
import org.springframework.stereotype.Component;

@Component
@Primary
@Profile("development")
public class StudyManagerImplLocal extends StudyManagerImpl
{

  /**
   * Deletes all studies associated with the logged in user via study xref.
   *
   * @throws NotFoundException thrown if there are no studies associated with user,
or if for each study, there are no
   *                           contributors or conditions associated with that
study
   */
  @Override
  public void deleteStudiesByUser() throws NotFoundException
  {
       List<Study> deletableStudies =
                    studyDao.findAllObjects(
                           StudyDao.Field.OWNER_XREF.getQueryTerm(
                                  AuthenticationHolder
                                         .getCurrentUserXrefStr()));
       for (Study study : deletableStudies)
       {
              condDao.delete(ConditionDao.Field.STUDY_ID.getQueryTerm(study.getId()));
              contDao.delete(ContributorDao.Field.STUDY_ID.getQueryTerm(study.getId())
              );
              studyDao.deleteById(study.getId());
       }
  }

  @Override
  public void truncateStudy()
  {
       // jStudyDao.truncateStudy();
  }
}
```

Appendix G: Removing OSF API Key from Back-end

```
package org.assistments.etrials.manager;

…
import javax.annotation.PostConstruct;
…
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class OSFManagerImpl implements OSFManager
{

  …

  // // Our Private OSF Token
  @Value("${osf.token:#{null}}")
  private String token;
  // // The Parent OSF project ID
  private String mainProjectID = "p2y9n";

  /**
   * Used for validating that application properties have been correctly mapped to
their corresponding field.
   * (e.g.: validate ${osf.token} is mapped to <code>token</code>)
   */
  @PostConstruct
  private void validateProperties()
  {
    if (token == null)
    {
      throw new IllegalStateException(
          "It looks like your local application properties file has not defined:
         osf.token");
    }

    token = "Bearer " + token.trim();
  }
```

Appendix H: Problem and Problem Set Types for TypeScript

Problems.ts

```
export interface Problem {
       id: number;
       xref: string;
       body: string; // FIXME: Attribution text included at the moment?
       type: ProblemType;
       assistmentId: number;
       answers?: Array<Answer>;
       parentProblemSetIds?: Array<number>;
       curriculumIds: Array<number>;
       skillIds?: Array<number>;
       position: number; // int 1-n determines partLetter
       /* Custom Attributes */
       assistmentPosition?: number;
       partLetter: string | null;
  }
```

```
  export interface Answer {
       value: string;
       isCorrect: boolean;
  }

  export enum ProblemType {
       CHOOSE_ONE = 'CHOOSE_ONE',
       CHOOSE_N = 'CHOOSE_N',
       RANK = 'RANK',
       FILL_IN = 'FILL_IN',
       OLD_ALGEBRA = 'OLD_ALGEBRA',
       OPEN_RESPONSE = 'OPEN_RESPONSE',
       RACKET = 'RACKET',
       FILL_IN_IGNORE_CASE = 'FILL_IN_IGNORE_CASE',
       IFRAME = 'IFRAME',
       NUMERIC = 'NUMERIC',
       NUMERIC_EXPRESSION = 'NUMERIC_EXPRESSION',
       EXACT_FRACTION = 'EXACT_FRACTION',
       ALGEBRA = 'ALGEBRA',
  }

export type ProblemList = Problem[]

export interface AssignmentIDProblemGroup {
    [assistmentID: number]: Problem[]
}
```

ProblemSet.ts

```
export interface ProblemSetUsageData {
    dbid: number,
    folderPath: string,
    isSkillBuilder: boolean,
    lessonType: string,
    problemCount: number,
    problemSetId: number,
    problemSetName: string,
    studentsCompletedApril: number
    studentsCompletedAugust: number
    studentsCompletedDecember: number
    studentsCompletedFebruary: number
    studentsCompletedJanuary: number
    studentsCompletedJuly: number
    studentsCompletedJune: number
    studentsCompletedMarch: number
    studentsCompletedMay: number
    studentsCompletedNovember: number
    studentsCompletedOctober: number
    studentsCompletedSeptember: number
    xref: string
}

/**
```

```
 * Type representing a problem set (with its metadata)
 */
 export interface WIPProblemSetStudy {
    dbid: number
    studyBlockType: string
    studyId: number;
    problemSetId: number;
    studyBlockTypeId: number;
    position: number;
}

/**
* Type representing a problem set (with its metadata)
 */
 export interface ProblemSet {
    id?: any;
    name: string;
    grade?: any;
    numProblems: number;
    avgCorrectLastYear: number;
    studentsLastYear: number;
    hintsLastYear: number;
    commonCoreStandard?: any;
    averageToMastery: number;
}
```