

Goatconnect

Major Qualifying Project



Goatconnect

A Major Qualifying Project submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the Degree of Bachelor of Science

Submitted to:

Professor Rodica Neamtu

Worcester Polytechnic Institute

Submitted by:

Loren DiLoreto

Jacob Feiss

Harrison Kyriacou

Date:

March 1, 2023

This report represents the work of three WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the projects program at WPI, please see:<http://www.wpi.edu/Academics/Projects>.

Abstract

Student Athletes at WPI (Worcester Polytechnic Institute) generally establish great connections with their current teammates; thus, while transitioning out of college and searching for jobs and internships, connections with former athletes and alumni are great resources for finding opportunities for work. Currently, these connections are facilitated by the team's coaches. In general, alumni working for companies that have positions open reach out to the coach and the coach forwards the opportunity to their team. An application specifically designed for helping make connections between former and current WPI Athletes would help create a tighter community and empower our students to get access to employment opportunities. The goal of this MQP is to create an athlete-oriented web networking application named Goatconnect to help student athletes on the football team get access to employment opportunities.

Acknowledgements

The project was expertly guided by Professor Neamtu and Football Coach Chris Robertson, who provided invaluable support in various aspects of development. Galen Lipin and Antonio Bothelho-Filho also made significant contributions to the project by facilitating the seamless integration of WPI single sign-on with the application.

Table of Contents

List of Figures	4
1.0 Introduction	5
2.0 Background	7
2.1 Similar Services/Applications	7
2.2 NextJS	8
2.3 PostgreSQL	9
2.4 React	10
3.0 Methodology	12
3.1 Project Management Framework	12
3.2 User Interface Design	14
3.3 User Beta Tests	16
4.0 Implementation	18
4.1 Implementation Tools	18
4.2 Application Architecture	19
4.3 Application Security	21
4.4 Documentation	22
5.0 Results and Analysis	23
6.0 Conclusion and Future Work	27
Works Cited	35

List of Figures

Figure 1: Goatconnect Jira Board	14
Figure 2: Screenshot of UI Mockup for Signup Page	16
Figure 3: Screenshot of UI Mockup for Create Job Listing Page	17
Figure 4: Goatconnect Architecture Map	21
Figure 5: Ease of Use of the Application	24
Figure 6: Qualitative Results Analysis	25

1.0 Introduction

With the rapid advances of technology in recent years, software applications have become ubiquitous in various industries, including education and the workplace. Among these applications, some have been especially helpful for networking and recruiting in the professional space. The most popular of these services is LinkedIn, which offers a web and mobile app for establishing connections with professionals from any industry or geographic location. A crucial feature of LinkedIn and other similar platforms is the ability for recruiters to post job listings and sort through potential candidates.

Although these platforms are great in general, there are situations where people could benefit from closer settings, such as instantly knowing what school and state prospective hires are coming from. An example of one of these scenarios is communication between WPI athletes searching for a professional role, and former WPI athletes looking to hire candidates they can trust to join their company. Currently, communication between these parties is facilitated through coaches, typically by forwarding text messages or emails back and forth. As the alumni network for WPI athletic teams become larger, the role of the coach becomes increasingly labor-intensive and many opportunities are missed, despite the effort put forth from the coach.

The objective of this Major Qualifying Project is to conceptualize, design, and develop a mobile application that will serve as a robust community-building platform for current and former athletes of the WPI. By fostering direct communication among the members of this community, the app will streamline their networking efforts, promote meaningful connections, and reduce the dependence on intermediaries such as coaches.

Moreover, the app will empower WPI football players by providing them with an avenue to remain connected to the team beyond their time as student athletes. Through the platform's various features, users will have the opportunity to expand their employment opportunities, access mentorship, and engage in career development. Goatconnect will enable WPI athletes to form a thriving community for years to come.

2.0 Background

The MQP team conducted background research into similar services/applications, backend server frameworks, frontend User Interface frameworks, and database options. This research guided the development of Goatconnect by establishing a robust framework prior to embarking on the development phase. This section will describe each of these research areas.

2.1 Similar Services/Applications

Handshake is a large-scale recruiting and job search application used by employers and students. Employers use the application to find students that have great professional skill sets to make new hires and help grow their companies. Students use Handshake to apply for jobs and connect with employers. Handshake offers an online platform where Universities can sign up and give students access to log into Handshake using their school single sign-on systems, so no new login is needed [7].

Users are first creating an account that requires minimal information such their name, age, and major, while also being able to add optional information later such as their resume, a cover letter, and any transcripts they have.

Chat and video call features are crucial parts to the flow of Handshakes app as well. While the process of setting up a profile and applying to jobs is not that different from most other job sites, having a built-in chat and video chat feature increases the user experience of the app and makes it so users have no reason to go somewhere else. Handshake also has a basic mobile app with fewer features than the website, but gives users the same information anywhere and act immediately to take advantage of opportunities. Handshake also allows for colleges to create

events to hold career fairs and for recruiters to hold information sessions, giving students even more career opportunities.

Another job search app that is very popular is LinkedIn, which allows users to connect with others on a professional level. Like Handshake, LinkedIn gives users a professional platform to look for opportunities. LinkedIn's biggest advantage over other sites is the ability to network with alumni and professionals while looking for jobs and mentorship [7]. LinkedIn works to establish personal connections completely unaffiliated with universities. LinkedIn also has a mobile app allowing users to keep up with notifications and applications on the go. LinkedIn's features overlap with Handshake and many other job searching platforms, and gives users a familiar way to search for jobs.

2.2 NextJS

NextJS was used to create our application, it is a Javascript and React framework that provides features to increase the quality of development. The two main features that NextJS provides are a Javascript server framework and a React build and compiler. NextJS provides functionality that makes both of these processes easier and more efficient.

To improve the backend server of the application, NextJS applications can easily be deployed without needing to manually configure routes. The way vanilla javascript web servers work is having developers specify what data is sent to the client for each specific HTTP GET or POST request. NextJS abstracts this complexity away with 'pages' where the name of the React Javascript file is the route that will be served. So for example if someone want to serve a page on the route '/shop/checkout' you would make a folder called 'shop' and a file called 'checkout.jsx' inside that folder. This structure makes complex multi-page web applications much easier to develop.

NextJS makes building and compiling React files much easier and more efficient by automatically building and caching React builds and then sending those builds to the client when requested. NextJS also allows you to build pages at request time if the page depends on server side data. This makes NextJS very flexible in terms of different design patterns that could be implemented. We have the option to pre-compile a page, and then make API calls to the server to get the data, or we could get that data server side, build the page and then send it to the client. Both these options have their own pros and cons that are evaluated on a case by case basis.

The other main option we considered was ExpressJS, which is much more lightweight and another very popular backend framework. Express popularized a design technique called "middleware" where the HTTP request can be processed using these "middleware" functions that execute in series. The popularity of Express made it an obvious option, but we thought the functionality NextJS provides with building and routing made it a worthwhile choice.

2.3 PostgreSQL

Most software applications need a database to store information about a particular aspect of their service. The most popular option is to use a relational database. These types of databases store information in tables and allow you to relate data points to one another. This is a very intuitive method of storing information as the data used in most applications has many many possible relationships. Another benefit is that most relational databases use Standard Query Language, or SQL. This standardized language is the same across relational database management systems, so a developer can easily hop from one to another with little downtime.

One of the most popular options for relational databases is PostgreSQL, an open source object-relational database system that uses the SQL language. Its transactions feature ACID properties, which ensure data validity amidst various errors and technological mishaps. The

supported programming languages for PostgreSQL include Java, Python, C, C++, Go, Ruby, PHP, Perl, JavaScript, .NET, as well as some server-side procedural languages through extensions.

A relational database is the best option for our application since our data contains different account types that have multiple relations to each other. Using a relational database will also help us to plan out what data we will collect early, helping us determine the scope of the project. PostgreSQL is a great fit for our project due to its popularity and ample support. Because of its mass adoption, most software resources are compatible with PostgreSQL, which will be useful when we use various plugins and libraries to speed up the development process.

2.4 React

Developing a robust user interface can be a tiring task due to the repeated use of HTML elements and CSS styling along with connecting the interface with the backend. React – developed and maintained by Meta – is a library that introduces components into a codebase, grouping sections of elements and logic to take away some of the repetition usually needed while developing a user interface [6]. This allows for highly reusable parts of the user interface, saving developers time and headache while trying to replicate looks and functionality on multiple pages. You can download React using the instructions provided on the React docs [8].

React components all have at least one state, and can have many different changing states based on the functionality needed from that component [6]. States are changed through sending different props into the component through its parent. An example of something a state would be used for is changing the name displayed on a component based on the user that is logged into a system. The effect a changing state has on its component can be minimal to very drastic, the only limitation is the developer's imagination.

React hooks are a way to pass functions into a component from a parent class, and get the results of that call in the parent even though they are being called in the component [6]. Hooks, like props, expand the reusability of react components because developers can use one component and use multiple functions within that element. A practical use of hooks is using a component for a form that could both send information to the server when a user is trying to sign up while also being able to pull information from the server to check if a user is already in a database when they tr`

When there is a change passed into a component from its parent class, rather than reloading the entire component, React only reloads the specific parts that have changed since the last render call [6]. This is done to help render the application in the quickest and most efficient way possible, and the rendering process all happens without the page reloading, giving a transparent feel to the component updates.

React Native is a branch of React exclusively designed for developing a mobile UI. React is for web and mobile apps React Native is mainly for mobile app development, allowing you to build apps that render natively on IOS, Android, and also windows. React Native has specific functions unique from React allowing the application to have control of functionality from phones, including location, camera, and sharing options.

Component libraries are an extension of the basic component structured coding that React introduces. Component libraries offer unique components and styles pre existing components, simplifying development and saving on development time. Different component libraries are also developed for more specific use cases, for example a library like Ant Design is made for use in enterprise applications while Rebase is a lightweight library which means it is easier on computers to run fast – similar to the relationship between windows and linux.

3.0 Methodology

Before working on the codebase for our project, it was important to plan our goals. The number one priority was deciding on how the team would be managed and how to keep track of our progress along the way. We created a User Interface (UI) mock-up to envision the frontend of our application. Running user tests with current student athletes, coaches, and alumni to collect feedback was crucial to help iterations on the application to make sure the user experience was possible. This section will go more in depth on our methodologies mentioned above.

3.1 Project Management Framework

We decided to follow the Agile-Scrum Software Development Methodology (ASSDM), since each of us has experience working in industry with teams following Agile development it was an easy choice. ASSDM is also the most popular framework for software development. According to Freeform Broadcom, a technology company that conducted a global survey that asked almost 1300 senior staff about how their companies are using agile. They found that 88% of respondents at least somewhat use agile in their development teams [3]. Due to our experiences and the popularity of the project framework, Agile and Scrum were clear winners for us. We followed key concepts from the Agile Alliance [4] during our project. This meant we held daily “standups” to discuss each of our progress, divided work into functional increments called user stories, focused on iterative development, and regularly received stakeholder feedback.

Our daily stand ups lasted about 15 minutes and each of us would discuss what we did since our last meeting, what our plan was for the rest of the day, and if there were any impediments that we had that we needed help from the group on. Holding ourselves to meet

daily really helped our team stay on the same page and quickly work through problems before they became too big and scattered throughout the code base.

The next agile concept we followed was dividing our work into *user stories*. A user story is a feature that a user of the application would want to have. We then broke that user story up into tasks that would be needed to complete it. This kept our focus on the people that would be eventually using the application and forced us to focus on those features that would matter rather than get caught up in technical tasks that might not be important to the user. In order to keep track of all the tasks, we used the Atlassian Jira issue tracking tool [1].

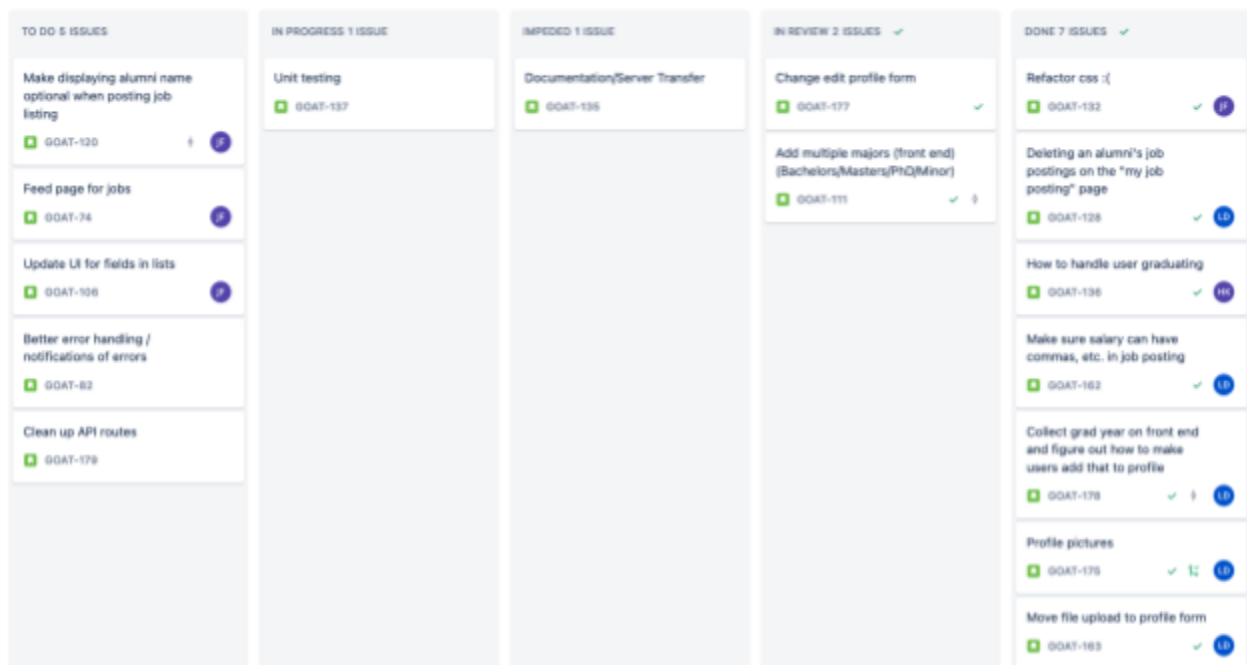


Figure 1: Goatconnect Jira Board

To develop iteratively, we continuously cycled through the same procedures in weeklong “sprints” that end with the application in a “finished” state. These procedures included planning out any user stories we would work on for the week, breaking them down to smaller tasks as

needed, assigning tasks to team members, implementing each task, and finally testing the final state so it can be considered “done”.

The final agile key concept we followed was regularly receiving stakeholder feedback. This technique is known as “prototyping” which the Interaction Design Foundation describes as the way for “design teams implement ideas into tangible forms from paper to digital.” [5] We considered the football coach, Coach Chris Robertson, to be our key stakeholder as he is familiar with the team and alumni, and has been serving as a liaison between them for many years. In order to keep him involved, we held weekly meetings with our advisor where we would invite Coach Robertson when there was a demo to show. When we received feedback from Coach Robertson, we would convert that feedback into a user story and incorporate it into our sprint backlog.

3.2 User Interface Design

Before beginning development, it is necessary to decide what types of components and front-end features will be needed. For planning purposes, we created a user interface mockup using an online tool called Figma that includes components like forms, lists, and a navigation bar, while also planning for the direction that the design and user experience should go in [9]. Figma is an online tool that allows you to easily mockup UIs using drag and drop shapes, and then link them using actions to mimic the control flow of an application [2]. From the interface mockup we separated the different sections into pages and components so we could develop more focused designs for each part of the mockup.

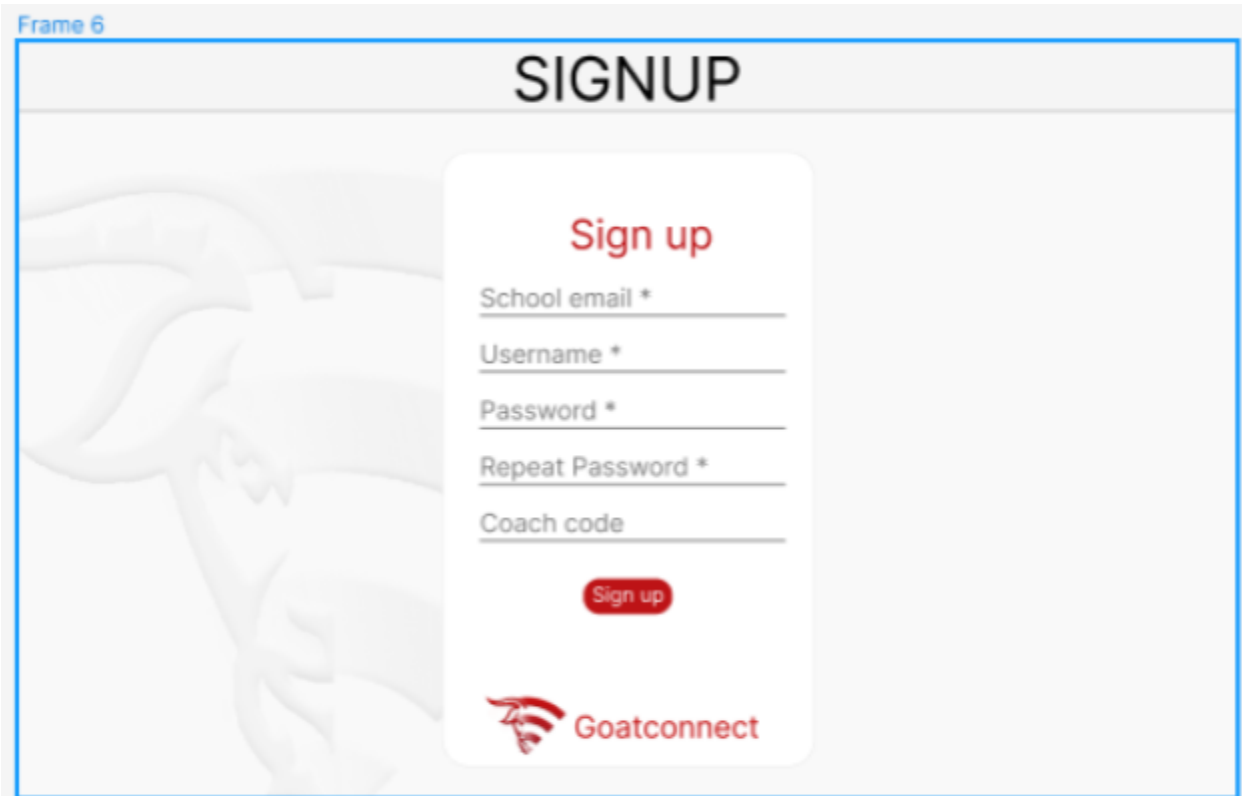
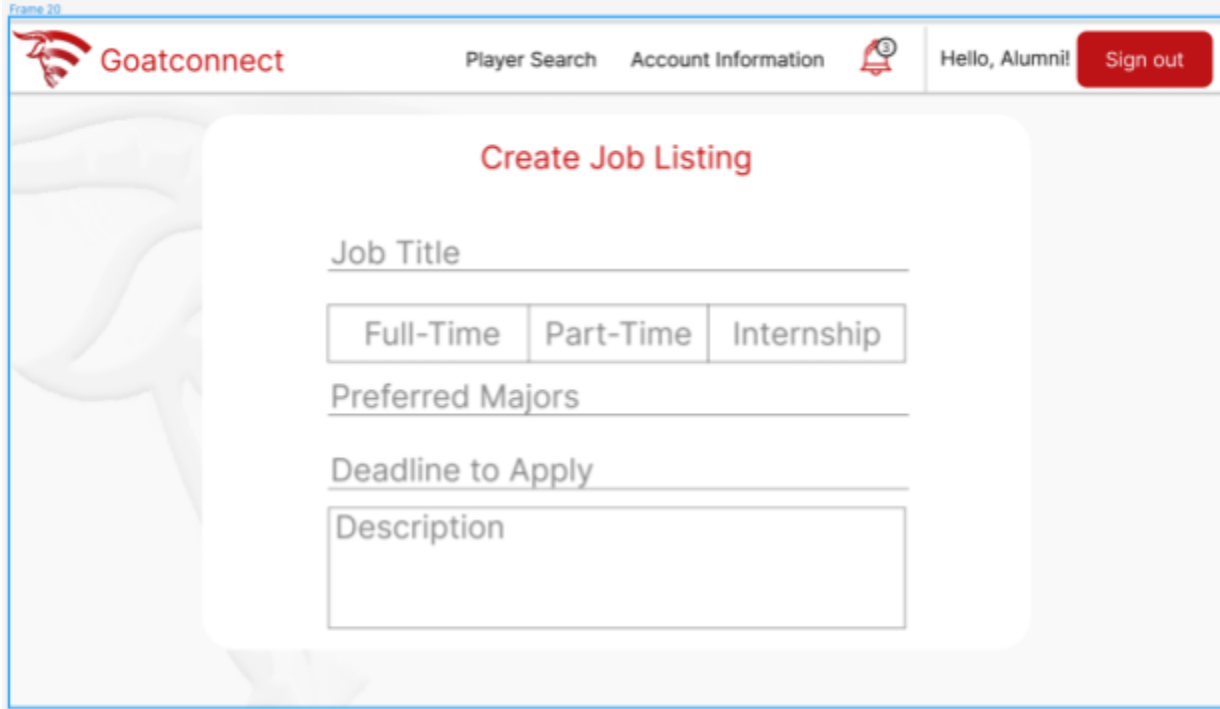


Figure 2: Screenshot of UI Mockup for Signup Page

After the initial design was made the UI went through multiple iterations as new features were added and discussions with our coach took place. As new features were added some elements of the UI proved to be not very user friendly, some of the corrections that were made were very small and some of them resulted in large overhauls to already existing UI elements. Discussions with our coach took place weekly, and since he will be one of the main users of our application it was important to our team that our coach had input on what would or would not be relevant to the use of the application. Features and smaller decisions on the UI were changed from iteration to iteration based on feedback our coach was giving us. If there was something in the UI that he was not able to figure out without direction from our team, it was deemed that it was not user friendly enough for release and altered within the next couple of iterations. This

constant user testing was a really helpful resource for the team to make sure the user experience is as good as it can be.



The screenshot shows a web application interface for creating a job listing. At the top, there is a navigation bar with the Goatconnect logo on the left, and links for 'Player Search' and 'Account Information' in the center. On the right, there is a notification bell icon, a greeting 'Hello, Alumni!', and a red 'Sign out' button. The main content area is a white card with the title 'Create Job Listing' in red. Below the title, there are several form fields: a text input for 'Job Title', a selection of three buttons labeled 'Full-Time', 'Part-Time', and 'Internship', a text input for 'Preferred Majors', a text input for 'Deadline to Apply', and a large text area for 'Description'.

Figure 3: Screenshot of UI Mockup for Create Job Listing Page

3.3 User Beta Tests

To collect feedback from actual users, we randomly selected players from the football team, randomly selected football alumni, and randomly selected football coaches to use the application for 30 minutes. We then asked them questions [Appendix B] to collect their opinions and feedback. This feedback would be recorded using handwritten notes and voice recordings when given consent [Appendix A]. After collecting this data, we will analyze it into overlapping themes and convert any widespread issues into our sprint backlog as user stories. To perform this test, we received approval from the IRB (Institutional Review Board).

After this first beta test, we refined the application into a production ready state and deploy it publicly. We will then grant access to another group of randomly selected players,

alumni, and coaches for an extended period. During this time, the users were allowed to use the application. At the end of this test, we will send out a survey [Appendix C] with the same questions we asked during the first test. We will similarly analyze the results of this survey and determine where the application improved, and if there are any other features that were requested.

4.0 Implementation

4.1 Implementation Tools

Github is a software development and version control hosting service based on the version control system Git [10]. We used Github for our source code and documentation, separated into two different private repositories. Our code was maintained in two primary branches, one for development and one for production. All new features were merged into development where they could receive adequate testing. Changes to the production branch would only come directly from development branch merges, typically at the end of a sprint or when a new feature was ready for a public release.

Jira is issue-tracking software typically used in Agile development environments [11]. Its issue tracking and task organization allowed us to seamlessly navigate our weekly sprints. We organized our tasks into stories small enough for one of us to complete inside an overarching epic, which typically lasted between 1-3 weeks. Any stories that were not important enough to make the current sprint were added to a backlog in which stories would be pulled from in the event of extra time at the end of a sprint. We decided stories' importance based on a priority ranking system. High-priority stories are ones that are needed for the basic functionality of the application, this includes high functionality features, bug fixes, etc. Medium-priority stories are generally wanted for the application, these are things like dark mode and the implementation of a video call api. Low-priority items are wants that most likely will not be completed during the duration of the project and will be left for the next team to continue our MQP if they want to.

Google Drive is a file storage and synchronization service. We used it as a central repository for all of our MQP documents aside from code documentation. This includes meeting

agendas, slideshows, and notes, as well as UI mockups and sketches. Due to its commonplace, it should be easy to pass our documents off to any future groups that are interested in continuing the project.

Visual Studio Code (VS Code) is a source code editor compatible with a variety of coding languages and frameworks. It contains plenty of features that accelerate the development process while maintaining the low profile of a code editor. The ability to connect to a remote host and automatically port forward proved useful for our team since our development server was hosted on premises at WPI. VS Code also contains extensions which provided development features tailored to each coding language that we used. We also took advantage of the Git integration in VS Code, which allowed us to code and manage the source versions all in one spot.

4.2 Application Architecture

To begin creating the application, we started by creating a NextJS “skeleton” application with the minimal amount of code needed to get it working. This allowed us to understand the structure and organization of a NextJS application. After having this starting point, we began building the home pages of the website and built the website outwards from there. Having the home page, we had to then have a way to log in and sign up for the application. The log in and sign up design patterns will be discussed in more detail in a later section. Next, we built out the player pages, which added job search functionality, and alumni pages, which added job creation functionality.

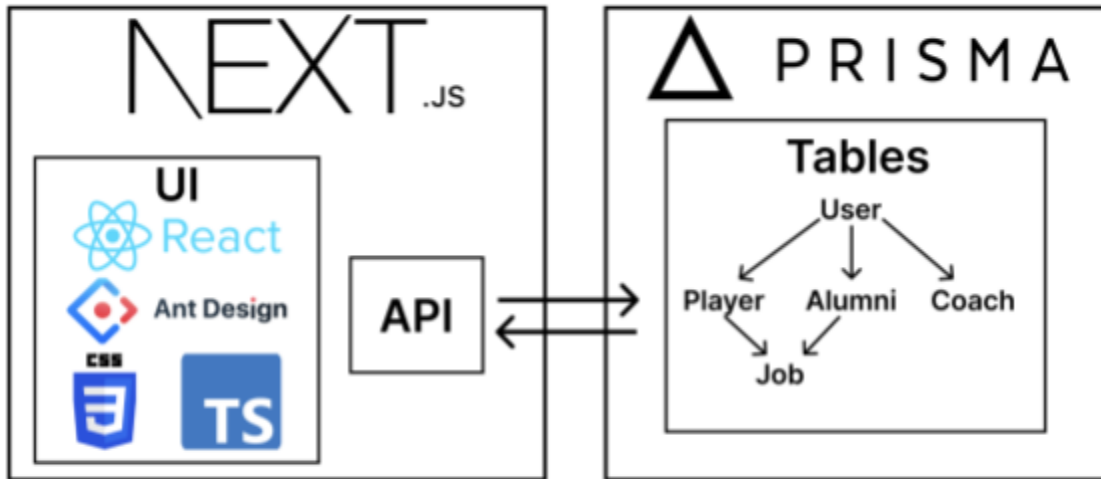


Figure 4: Goatconnect Architecture Map

The next thing we focused on was the “coaches” pages. These pages would be used by coaches to invite users onto the platform. This presented a few challenges as we needed to ensure the website would be secure and not just anyone can make an account and post potentially malicious job postings. These challenges along with our sign in/sign up patterns will be discussed in our **Application Security** section. Having these pages meant the “core” functionality of the application was complete. The next task would be to ensure all pages followed a uniform style and design.

We decided to use AntDesign as our component library early on in order to minimize the amount of time we needed to work on individual components. AntDesign is a library made for use with React that incorporates different components and other various design elements. Using AntD allowed us to have a starting point for our different design components, and gave us valuable pieces we were able to leverage throughout our design pattern throughout our application. We used the AntD docs to look at all the available components and assess which ones would be useful for our application. Most of the components used are a combination of

AntD components alongside specific assets and styles we made specifically for use in our application in order to get the best user experience.

Cascading Style Sheets (css) were used in combination with the component library to give the app a more unique look while also making sure that the app looks good in different circumstances. Work with css needed to be done to make sure that different window sizes and scales worked while having readable text and usable components. Css was also used to slightly alter components from AntD to make them fit in with the design of the application. In order to persist our application data, we then had to set up our backend database. To accomplish this, we used a PostgreSQL database and the Prisma NodeJS library.

Prisma is an open source object-relational mapping library. It features Prisma Client, which is a type-safe query builder for Node.js. This allowed us to generate complex PostgreSQL queries in a fast-paced development environment. We also used Prisma Migrate to update our database schema while maintaining existing data. Prisma Migrate also allowed us to keep a version control for our database schema, making it easy to revert to previous versions in the event of a mistake. Another tool we used was Prisma Studio, which provided a GUI for parsing and updating database records. We were able to populate our database with large amounts of realistic test data using the open source library Faker.

4.3 Application Security

A major challenge we faced was how to handle the security of the application. We examined two options; to handle username/password log in ourselves, or if we could utilize WPI's Single Sign On (SSO) infrastructure to authenticate users. SSO is a protocol where user authentication is handled by a trusted third party. In our case, we would be relying on WPI (who is relying on Microsoft) to handle user credentials like usernames and passwords. Using SSO

means our application database would not need to store and secure any sensitive user information.

We settled on supporting both sign on options to give users whatever experience would be easiest for them. In order to ensure the safety of user credentials, no user passwords are stored in our database in plaintext. Instead, we utilize the best practice of hashing user passwords with a random string called a “salt”. Then, when a user tries to sign in, we hash the password they enter with the salt and check if the result matches what we stored in the database. This takes some stress off of us to protect a user's sensitive information.

Utilizing SSO meant that we needed approval from WPI’s Chief Information Security Officer (CISO) and other information security administrators. This process included several meetings where we went over how we would use the data provided to us by WPI. We also collected the necessary URLs and cryptographic keys from the SSO admins that are required by the SSO protocol.

An even more secure method of authentication would be to solely use WPIs SSO infrastructure. Forcing users to use WPI SSO would be preferable from a security perspective, but would limit access to users who have access to a WPI account. This was a major issue for us since we are expecting alumni who may not have an active WPI account anymore to use the application. Giving users the choice is the best option we have in this scenario.

4.4 Documentation

The team engaged in a great effort to refactor and document and wrote the source code readable enough for a future team to pick up where we left off. This really takes the application to the next level and makes it more polished and long lasting. We are very excited that Goatconnect is live and already having users sign up and participate!

5.0 Results and Analysis

Our development team held a beta testing session for Goatconnect in November of 2022 with over 40 current football players at WPI. The session was an opportunity for the MQP team to gather feedback on the product's performance and user experience. Results from the survey given out to the focus group regarding our application showed high praise for its ease of use. All Participants but one, deemed it extremely intuitive, which is an important consideration given that the focus group represents our main target audience. These positive results indicate that the application was headed in the right direction, which allowed our team to make further improvements to the structure already in place for the rest of the project's duration.

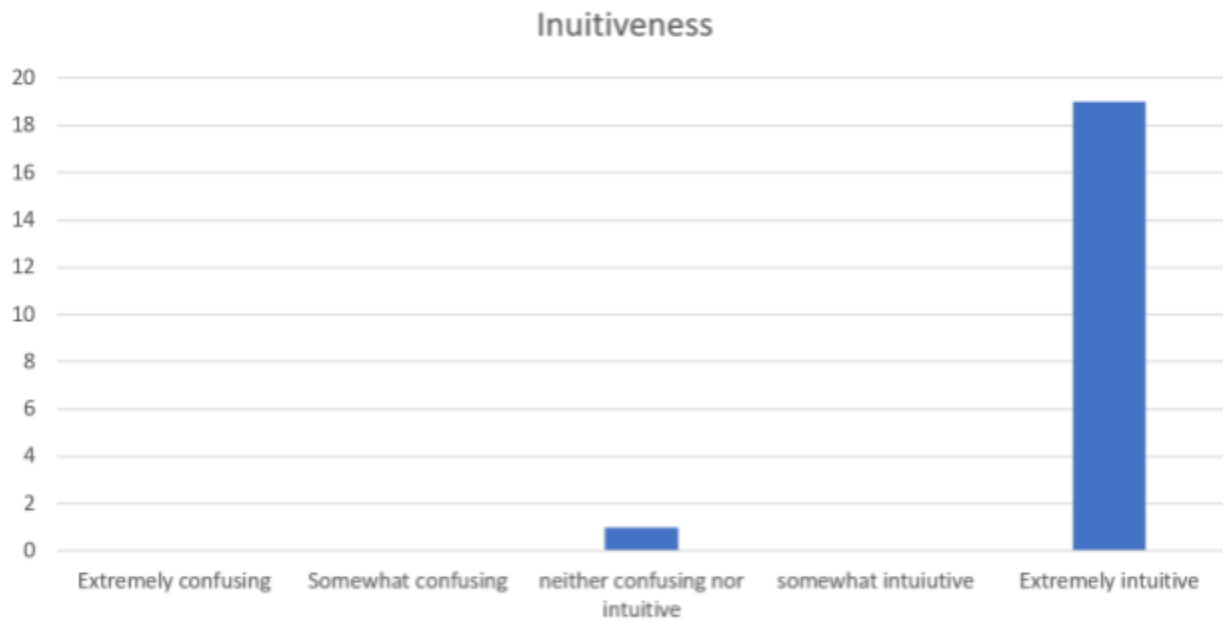


Figure 5: Ease of Use of the Application

During the discussion, the players shared their thoughts on what they liked about the app and suggested possible improvements that should be made. As shown in *Figure 6*, a majority of the athletes that participated in the beta test had suggestions for features, and less suggestions for

UI and functionality. While the amount of suggestions for features outweighed the amount of suggestions for UI and functionality, it was still important for our team to make sure we handled the possible functionality, user interface, and user experience improvements that could be made before going live with the application. Thankfully the beta test allowed for our team to become aware of some problems that were vital for us to address, even though there was a smaller amount of suggestions in that area.

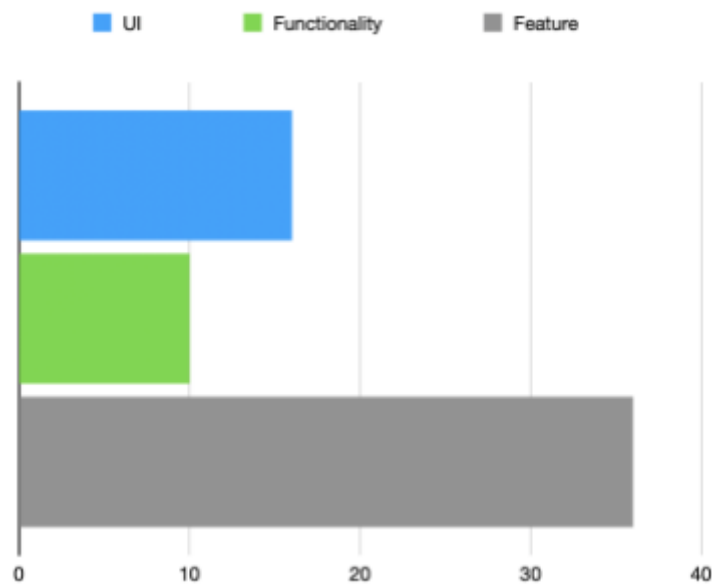


Figure 6: Qualitative Results Analysis

The players really wanted to see more information presented on job postings. Some of the information that they suggested to be added were job types (such as internship, contract, full-time, or part-time), locations of jobs, and specific majors that are required for the job. The ability to filter jobs by the data mentioned previously was also a popular request. They also appreciated the idea of a feed page that would show jobs that match their skills based on their profile. The ability to add a job to a favorites list, and the option to upload a resume or other files into their account were also suggestions that were popular in the user study.

All of the suggestions listed above were things that our team decided to include in the current iteration of the application. We added job types and majors to job listings to allow for alumni to specify what they are looking for, and allows athletes to sort by these things while looking for a job. The favorites list was at first implemented as a quick apply functionality but was altered to serve our applications needs in a more specific way, and allows users to save jobs that they are interested in. A feed page was also created for Alumni and for Athletes to show recent activity such as newly posted jobs and recent applications.

Overall, the beta testing session was a valuable opportunity for the development team to gather feedback and insights from its target users, and it will be used to improve the product before its official launch.

We also gained valuable feedback from our interview with Coach Robertson, Head Football Coach. Throughout our interview we had Coach Robertson walk through the application with a given set of instructions, and without giving him any direction on how to complete a task. Coach Robertson was able to get through the entire application without many problems, and afterwards he gave us a short list of things he would like to see improved in the application for ease of use. As one of the main users, he suggested that it would be easier and more efficient for coaches if there were batch invites to the app instead of sending out invitations one by one. Another suggestion mentioned was being able to edit users in the database from a coaches end. Coach Robertson also gave the team direction on what other features would be useful to him in the application and what things would not be used, such as social media feed and making announcements to his players.

Throughout the alumni onboarding process our team was able to connect with thirteen alumni users and get direct feedback on features and bugs from them. Alumni were able to point

us in the direction of what fields should or should not be required, and also were very quick to let us know when they found a bug, giving our team a much higher number of testers. Working with alumni inspired us to add the “give feedback” button, which allows users to send feedback directly to our team instead of emailing informally.

6.0 Conclusion and Future Work

When we started this project, the development team had the goal of creating a production ready and fully capable application for current student athletes and alumni to connect and find jobs or internships. The application officially went live on February 1st, 2023 and was open to current alumni of the WPI Football program to begin signing up and posting real jobs for the organizations that they are affiliated with. In order for Goatconnect to truly be “production ready” the development team was required to surpass the previous quality of work in every element of the development process. The user interface had to be smooth to interact with and intuitive enough for people with no technical background or interaction with the team to use. This was a requirement that we have never had to consider in our classes full of other computer science students. The application had to be responsive with quick loading times and error catching/notifications for users who may have inputted the wrong item into a form. To ensure the necessary level of security, the application must implement proper authentication and access control, utilizing only trusted and reliable libraries, frameworks, and protocols. All of these considerations ensured the CIA Triad [12] of confidentiality, integrity and availability for Goatconnect’s users.

There were also some other suggestions that were brought up during the beta that the team decided to add as future work. These suggestions recommended were current news related to the football team and a chat feature. Ways to improve the user experience could also include allowing alumni to make announcements to users about events their companies are hosting,

notifications to users, and the ability to mass send emails to users about patch notes or any other announcements.

Even though the application is now available to the team and alumni, there were many features and quality improvements that we had hoped to implement by the end of our projects. We suggest that if a future MQP team at WPI continues this project they consider these future work proposals for their project:

- Integrate Goatconnect with other applications
 - Handshake for easy profile creation and links to jobs
 - Outlook for easier inviting of users with email suggestions
- Current news of the team to keep alumni informed
- Ways for alumni to connect with each other in a non-professional setting just to keep in touch
- Chat feature for quick conversations between alumni and student athletes
- Alumni/Coach announcements that goes to players feeds and notifies users following that company/team
- More permanent hosting and services
 - Currently, the application is hosted on temporary WPI Linux servers. Ideally, the application would be hosted with the WPI athletics website
 - Currently, free tier AWS is used for email but it eventually will need to be paid. Goatconnect will need a more permanent email service option.
 - The database and file storage of Goatconnect is limited. As the application grows, there will hopefully be a service that would dynamically scale as the data grows
- Editing/deleting users by coaches

Appendix A: Evaluation Informed Consent Form

Informed Consent Agreement for Participation in a Research Study

This Informed Consent Form will be presented through a Qualtrics Survey

Investigators:

Loren Diloreto (SI, lddiloreto@wpi.edu), Jacob Feiss (SI, jhfeiss@wpi.edu), Harrison Kyriacou (SI, hmkyriacou@wpi.edu), Rodica Neamtu (PI, rneamtu@wpi.edu)

Title of Research Study:

Goatconnect - Developing a student-athlete specific networking application

Introduction

You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of the study:

The purpose of our usability study is to evaluate the user experience and usability of particular components in a student-athlete specific networking application.

Procedures to be followed:

As part of the survey you will be asked to navigate to a website through a provided link. Through an accompanying Qualtrics survey, investigators for this research study will present you with predefined tasks to complete. Your interactions with the website will be logged. The Qualtrics survey will also ask you questions about your experience to complete after the tasks are done. This user study will take around 10-15 minutes.

Record keeping and confidentiality:

Your interactions with the app will be logged. Your survey responses will be recorded through Qualtrics. Records of your participation in this study will be held confidential. Any publication or presentation of the data will not identify you. Sensitive information will not be recorded in this study.

Your participation in this research is voluntary.

By clicking “Yes, I do consent.” below you are consenting to participate in this research. Or by clicking “No, I do not consent.” below you are not consenting to participate in this research.

The participant will then check a box to confirm that they consent to participate in the study or that they do not consent to participate in the study. The survey will then direct the participant if they have consented, to enter their email address, area of study, and select their phone, frequency of phone use, primary purpose of phone use. Otherwise, if they do not consent the survey will end

Appendix B: Evaluation Protocol

User Study Protocol

This user study will be used to evaluate the effectiveness of newly implemented features of a prototype web application for networking between current and former student athletes.

Recruited participants will receive an email including a link to the website, as well as a link to a corresponding Qualtrics survey. Within the survey, participants will be asked to perform predefined tasks and respond to questions. Following the execution of all of the tasks, the participants will be asked questions regarding their experience using the application. The survey responses will be recorded through Qualtrics. Additionally, the participants' interactions with the app will be logged to assess timing and application performance. The following are potential tasks that a participant testing the player functionality will be asked to do:

1. From the landing page, navigate to the Sign Up page
2. Create a player account using the form on screen
3. Log in using the information you just provided
4. Complete your account by filling out the necessary information
5. Navigate to the Job Listings page
6. Find any job listing with the word "software" in the job title and record the job title of the listing
7. Find any job listing for the company "GoatDev" and record the job title of the listing

The following are potential tasks that a participant testing the alumni functionality will be asked to do:

1. From the landing page, navigate to the Sign Up page
2. Create an alumni account using the form on screen
3. Log in using the information you just provided
4. Complete your account by filling out the necessary information
5. Navigate to the Players page
6. Find any player with the last name "Daniels" and record their email address
7. Find any player who is a chemical engineering major and record their full name
8. Navigate to the Create Job Listing page
9. Create a job listing by filling out the form
10. Navigate to the My Job Listings page
11. Find the job listing you just created

The following are potential question areas to be covered by the survey:

1. On a scale from 1 to 5, with 5 being the most difficult, how difficult was it to find an eligible job listing?
2. On a scale from 1 to 5, with 5 being the most difficult, how difficult was it to find a player with the correct last name?
3. On a scale from 1 to 5, with 5 being the most difficult, how difficult was it to find a player with the correct major?
4. On a scale from 1 to 5, with 5 being the most difficult, how difficult was it to create a job listing?
5. Is there anything that particularly appealed to your experience?
6. Is there anything that particularly hindered your experience?
7. Please provide any questions, concerns or suggestions you may have pertaining to this component and your experience with it.

Works Cited

[1] Atlassian. “Jira: Issue & Project Tracking Software.” Atlassian. Atlassian. Accessed October 11, 2022. <https://www.atlassian.com/software/jira>.

[2] Figma. Figma. Accessed October 11, 2022. <https://www.figma.com/>.

[3] “How Agile and DevOps Enable Digital Readiness and Transformation.” Freeform Dynamics, February 2018.

[4] “What Is Agile Software Development?” Agile Alliance. Agile Alliamnce, May 26, 2022. <https://www.agilealliance.org/agile101/>.

[5] “What Is Prototyping?” The Interaction Design Foundation. The Interaction Design Foundation. Accessed October 11, 2022.
<https://www.interaction-design.org/literature/topics/prototyping>.

[6] Baer, Eric. “What React Is and Why It Matters.” *O'Reilly Online Learning*, O'Reilly Media, Inc.,
<https://www.oreilly.com/library/view/what-react-is/9781491996744/ch01.html>.

[7] Mehrotra, Pranob. “Linkedin vs Handshake: The Ultimate Job Search Portal for University Students.” *Guiding Tech*, 3 Feb. 2022,
<https://www.guidingtech.com/linkedin-vs-handshake/>.

[8] “React Docs.” *React*, <https://react-cn.github.io/react/downloads.html>.

[9] “Free Design Tool for Websites, Graphic Design and More.” *Figma*,
<https://www.figma.com/design/>.

[10] “GitHub Features: The Right Tools for the Job.” *GitHub*,

<https://github.com/features>.

[11] Atlassian. “Jira: Issue & Project Tracking Software.” *Atlassian*,

https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=136973856930&campaign=18440774103&creative=639487383004&device=c&keyword=jira&matchtype=e&network=g&placement=&ds_kids=p73335831609&ds_e=GOOGLE&ds_eid=700000001558501&ds_e1=GOOGLE&gclid=Cj0KCQiAiJSeBhCCARIsAHnAzT8UEKyTxd8GZNIMDcuHMN0OHkZ-zHENfiNMVmk3_flivmDOeCQqgxMaAjzmEALw_wcB&gclsrc=aw.ds.

[12] Fortinet. n.d. “What is the CIA Triad and Why is it important?” Fortinet. Accessed

February 3, 2023. <https://www.fortinet.com/resources/cyberglossary/cia-triad>.