Final Report

Version 1.0
28 April 2021

Team DigiFolio
Dr. Andy Wang
Fabio Santos

Burbank, Logan(lead)
Braudaway, Jackson
Chen, Kailin
Marschel, Parker
Yang, Zhenyu

Table of Contents

1. Introduction

Landing a job right out of college can have a determining factor to whether or not someone has had industry or internship experience prior to graduation. This leaves students who do not have prior experience to search for a job after they have graduated or even forcing these graduates to look into other fields not related to their degree for work. This is not only a problem for students attending Northern Arizona University; it is also an underlying problem for millions of other graduates looking for work.

Currently, at Northern Arizona University, students' success is tracked through exit surveys before each student graduates. While this is semi effective in getting students to provide feedback before they graduate, this information is obtained too late in a students academic career to be effective in helping students find a job. Even with programs such as job fairs or Handshake in place, they are all optional for students to participate in. Our solution aims to change this and actually promote student success by tracking a student's progression at the beginning of their college career rather than at the end through the use of a web application that will display their career milestones and progress highlights. Through this, both faculty and students will be able to see how the student is doing at any point in their college career and be able to assist them in reaching their set goals.

Our sponsor for this project is Dr. Andy Wang, progressor and dean of the College of Engineering, Informatics, and Applied Science here at Northern Arizona University. With Dr. Wang being the dean of the college, he looks over thousands of students. He understands how well the college and its students are performing is not easy to do with only exit surveys. Our solution will be able to paint an accurate representation of this so that Dr. Wang can work to help students achieve goals and be more successful by the time graduation comes. Overall, with this tool, students will be able to track their progress towards their career goals.

2. Process Overview

Throughout the course of this project our team worked alongside our mentor Fabio Santos and our client Dr. Wang to understand the problems and needs of the current system in which student career success is tracked and measured. By doing this we gained a better understanding of how our solution should accommodate this problem and assisted us in every step of the software development life cycle from planning to design to implementation and then full deployment of our solution. As our application was under development we tried and tested a few different technologies along the way including fusion charts for our statistical graphing module, node.js for our application framework, mySQL for our backend database, and the NAU ITS server to host the application after its completed development. Now, prior to completing the application our group members took on individual roles and responsibilities while putting forth different procedures to organize the code base and discuss the project as a group. While we did meet with our mentor weekly and our client fairly often as needed, we also came together as a group every Friday outside of these meetings on Discord to go over upcoming tasks and responsibilities. Here we were able to share files and make decisions in regards to how and what to include with our solution as well as who would do what for different upcoming tasks in order to break those tasks into smaller pieces that could all fit together in the end like a puzzle. Additionally, meeting with each other allowed us to bounce ideas off one another and made sure we all had the same level of understanding for upcoming goals and tasks. In the end, communicating with our mentor, client, and each other provided us with ideas on how and what to include in our solution to meet our clients needs all the way through the planning, analysis, design, implementation, integration, and finally the deployment of our application.

3. Requirements

Our functional and non-function requirements were acquired through our communication with our client Dr. Wang. At the introduction of this project we gathered our initial requirements, which over the course of the development life cycle gradually changed. This change included the addition of more requirements and modification of others, which also correlated directly with our growing understanding of what our client was looking for in our application. As a result of this, our main functional and non-functional requirements emerged as the following:

Domain Requirement 1: User System

Our first domain level requirement involves implementing a user system that will allow for user accounts with role-based permissions. There will be three account types, with these types being Student, Faculty, and Alumni. Each of these types will have specific basic functional requirements. With that, each account type will have functional requirements:

FR1.1: A student can create a student account with student permissions.

- **1.1.1:** A student can log in and log out.
- **1.1.2:** A student can view and modify their own portfolio page.
- **1.1.3:** A student can decide what is visible on their page.
- **1.1.4:** A student can view public information on other student pages.
- **1.1.5:** A student can view the overall progress dashboard.

FR1.2: A faculty member can create a faculty account with faculty permissions.

- **1.2.1:** A faculty member can log in and log out.
- **1.2.2:** A faculty member can view and modify any student portfolio page.
- **1.2.3:** A faculty member can view the overall progress dashboard for individual students, as well as a summative progress dashboard for a program, department, and college.
- **1.2.4:** A faculty member can change a student account into an alumni account.

The second functional requirement is broken down into four smaller functionalities, and it is also a simple requirement. This is just like the first one, with the exception that it is for the permissions of a faculty member. The biggest difference here is that the faculty member is able to modify any student page, but is not able to select what pieces are public. Also, the dashboard will show faculty individual student progress toward her/his goals, as well as a summative progress for a program, a department, and a college. Finally, the faculty will also be able to modify a student account's status, allowing student accounts to become alumni accounts through a faculty member.

FR1.3: A student account will become an alumni account when the student has graduated.

- **1.3.1:** An alumni can log in and log out.
- **1.3.2:** An alumni can view their own portfolio page.
- **1.3.3:** An alumni can decide what is visible (public) on their page.
- **1.3.4:** An alumni can request to have their portfolio page updated.

The third functional requirement is broken down into four smaller functionalities, with this user type having the least functionality. Ultimately, this account is similar to a student account, with the ability to request page updates rather than being able to directly modify the page. This is so the information being added can be verified by an administrator.

Domain Requirement 2: Digital Portfolio

Our second domain level requirement involves implementing a digital portfolio page that will be built for each student account created. This page will be linked to a student account, and it will include information about their name, contact information, and career milestones.

FR2.1: The portfolio can be modified by the user.

- **2.1.1:** A user can add a career milestone to the page.
- **2.1.2:** A user can remove a career milestone from the page.
- **2.1.3:** A user can report a data problem to a system administrator.

The first functional requirement for the digital portfolio can be broken down into three smaller requirements. These requirements are focused on the milestones with the user portfolio which will be updated by that user over time.

FR2.2: The portfolio will display student career information.

- **2.2.1:** The student name will be displayed.
- **2.2.2:** The student career milestones will be displayed.

The second functional requirement for the digital portfolio is broken down into two smaller functionalities. The page will need to be able to build using the user information when they choose to request their page. This will involve displaying the student name, as well as the milestones connected to the student.

Domain Requirement 3: Goal Dashboard

Our third domain level requirement involves implementing a digital dashboard capable of displaying the current progress of the student population in achieving university-specified goals.

> **FR3.1:** The dashboard must calculate the percentage of students with an internship (including externship, Co-op, research project with experiential learning outcomes) for a unit (program, department, or college).

The first functional requirement focused on the dashboard is simple enough to not need any further breaking down. Ultimately, this will be a function that is done semi-automatically, with the dashboard module needing to be able to calculate how many students have earned an internship, externship, co-op, or research/experiential learning opportunities. As this will be displayed in multiple possible ways, the dashboard will achieve this by keeping a total count of students and consistently updating the number who have had this internship. From this, a graphing module with this calculated percentage will be displayed for a visual representation.

> **FR3.2:** The dashboard must calculate the percentage of students with a job offer for a unit (program, department, or college)

The second functional requirement focused on the dashboard is also not needing to be broken down further. This will be extremely similar to the previous challenge, with the only difference here involving what goal is being checked. In this case, the module will already be storing the total number of student accounts, and it will now store the number of students who have received a job offer. Then, it works like the previous requirement in how it will check and calculate the results.

> **FR3.3:** The dashboard must display each percentage in multiple chart forms
>> **3.3.1:** The percentage can be displayed as a pie chart.
>> **3.3.2:** The percentage can be displayed as a progress bar.

The third functional requirement focused on the dashboard is the only one needing to be broken down further, mainly to lay out the different formats that will be possible on the dashboard. We have decided to display these percentages in different graphical forms such as a bar graph, pie chart, pyramid graph, etc.The user will have the option to swap between these three different charts with a simple menu of choices.

Domain Requirement 4: Pulling from LinkedIn

Our fourth and last domain level requirement involves implementing a module capable of reading data from LinkedIn and analyzing what data fits into which category. This data will then be used to populate the student portfolio pages. This module must be able to pull information from LinkedIn, analyze and classify the data, and store the data in its correct place.

- **FR4.1:** The LinkedIn module can pull data from LinkedIn servers
  - **4.1.1:** A request can be sent to the LinkedIn servers using its API
  - **4.1.2:** The resulting package is checked to verify it is what was requested.
  - **4.1.3:** The package can be unpacked into temporary memory.

The first functional requirement focused on the LinkedIn module is broken up into three separate functionalities, with each being necessary to achieve the greater requirement of pulling data from LinkedIn. In order to do so, the module must be able to send a request to the LinkedIn server, verify the result that is sent, and unpack the result into temporary memory. These stored results will then be used for the other functional requirement, and this step will typically be done automatically, with the choice to scan for new data manually.

- **FR4.2:** The LinkedIn module can classify the data into categories.
  - **4.2.1:** Data can be classified as an identifying piece (ie. email)
  - **4.2.2:** Data can be classified as a career milestone

The second functional requirement focused on the LinkedIn module is broken up into just two smaller functionalities, with the main purpose being to classify the information pulled from LinkedIn into categories. These categories as we see it will include two important ones to take note of, with the first being any identifying information (LinkedIn API uses email), and the second being a career milestone. LinkedIn will likely not make it obvious what each career milestone we are pulling entails, so this module will check and place the data in the right spot once it is pulled.

Domain Requirement 5 (Stretch Goal): Job Posting Page

One domain level requirement not mentioned above is the Job Posting Page. This is because it is our main stretch goal, but unfortunately we were not able to allocate enough time to work on this after finally getting all of the functional requirements implemented. Although, the job posting page is not urgent to the functionality of the rest.

This page will be made to display job postings that are on Handshake, and this will be done using a tool created for the exact purpose. This mainly leads to only one functional requirement, and that is displaying data.

> **FR5.1:** The Job posting page can automatically display a list of job postings from Handshake, allowing links to quick access to the Handshake posting from their site.

This single functional requirement is all that will be required of the job posting page. It will be made as a gateway of sorts to specific pages on the Handshake servers. This will allow students to have a viewable list of job postings that they can easily access and apply for through the given route on Handshake.

Overall, these are the resulting requirements that have been obtained over the course of the development life cycle of our project that lead us to our solution. In the end, each of these pieces were brought together to form our web-based application.

4. Architecture and Implementation

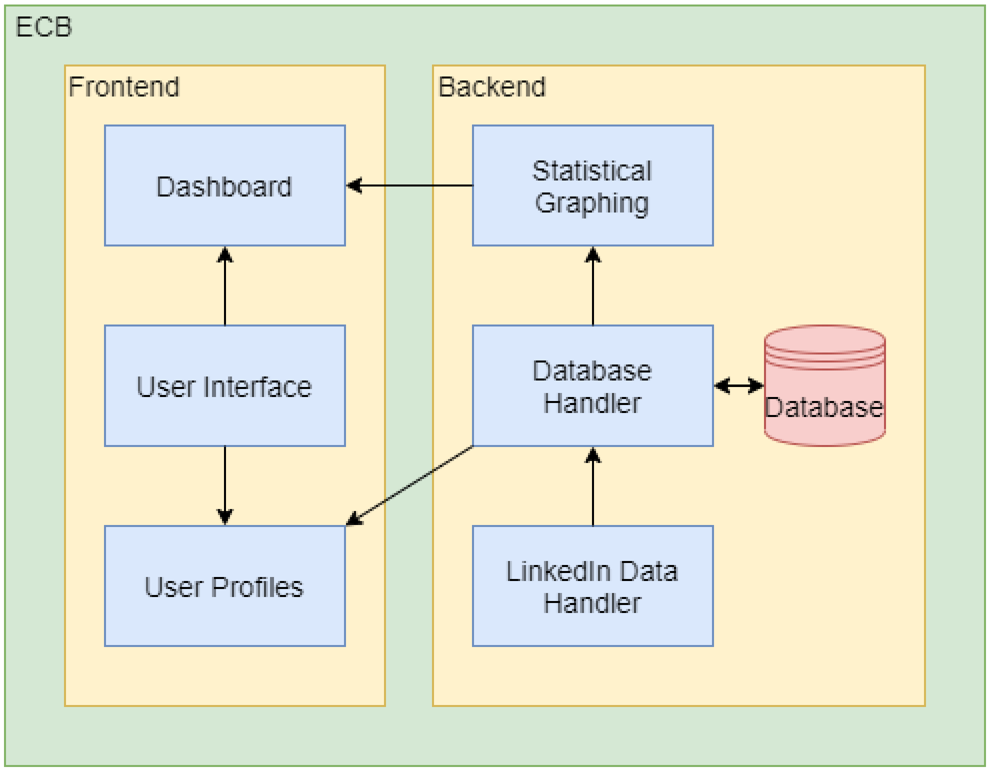

Figure 4.0.

Our architecture consists of several key major parts. As can be seen in Figure 4.0, we have the backend, the frontend, and the application as a whole. The frontend will consist of three main parts, the dashboard, the user interface, and the user profiles. The backend will also consist of three main parts with an added database to hold all the data of the application itself. The three main parts of the backend are the Statistical Graphing, the Database Handler, and the LinkedIn Data Handler. Overall, the entire application holds all of these components to create the application as a whole.

The front-end components are the objects that users interact with. The most important component that the user interacts with is the entire user interface itself, it will allow users to register with the site, create a profile based on their academic achievements, and navigate throughout the site to look at the overall success of every student. And next is the user profile. The user profile will allow users to create their own profile based on all their academic milestones, and retain their profile as a resume profile through NAU. Finally, there is a dashboard on the frontend, where all the student's milestones and accomplishments will be displayed in graphical form to see the overall accomplishments of the students.

The components that the backend consists of are where the developers connect all the components together into one big application that the users get to experience. First, there is the LinkedIn data handler which will scrape data from a LinkedIn account and move all the information into the database. Second, there is the statistical graphing component where it will take all the data and calculate specific goals and display them in a graph on the website. Lastly, there is the database handler and the database itself where all the information will be stored and organized to be accessible by any requests made from the website.

The overall architecture design will be a web two-tier. Web two-tier has only two tiers: the client application and the database. The client application, where the client can send requests to the server-side without a middleman having to interfere with any of the requests. The server side then handles the database where it can store information without having to have a third action to keep all the information. Our users will simply be able to create their profile which will go straight to our database and then the request will be sent back with their profile and any information they may have entered into the database from the frontend. Designing our architecture as a web two-tier makes it simple and easy to maintain.

4.1 Key Responsibilities and Features

4.1.1 User Profiles

User profiles have many requirements which involve implementing a user system that will allow for user accounts with role-based permissions. There will be three account types: Student, Faculty, and Alumni. Each of these types will have specific basic functional requirements. A student's use in this application will involve logging in and out, viewing portfolio pages, modifying their own page contents and visibility, and seeing the overall progress. The faculty account is just like the student account, with the exception that it is for the permission of a faculty member. The biggest difference here is that the faculty member is able to modify any student page, but is not able to select what pieces are public. Also, the dashboard will show faculty individual student progress toward her/his goals, as well as summative progress for a program, a department, and a college. Finally, the faculty will also be able to modify a student account's status, allowing student accounts to become alumni accounts through a faculty member. Ultimately, an alumni account is similar to a student account, with the ability to request page updates rather than being able to directly modify the page. This is so the information being added can be verified by an administrator. The profiles will talk back and forth between the database and the user's profiles to make sure data is up to date.

4.1.2 User Interface

The user interface is a combination of all the features including the dashboard, the user profiles, and the website itself. In this user interface a user is allowed to register and create a profile. This profile will then be able to have its own personal profile page which has many features that are included in the profile section. If a user is already registered, then they are able to simply log in. The user interface also includes the dashboard where overall goals will be displayed.

4.1.3 Dashboard

The dashboard part consists of a digital dashboard capable of displaying the current progress of the student population in achieving university-specified goals. For now, there are two different goals this dashboard will be capable of tracking: the total percentage of students with an internship (including externship, Co-op, research project with experiential learning outcomes), and the total percentage of students with a job offer. This dashboard will get its data from the database in the backend. It will then calculate anything based on the goals that are implemented on the website.

4.1.4 LinkedIn Data Handler

The LinkedIn data Handler will scrape data from a LinkedIn account, which will be used to populate the student portfolio pages. This module must be able to pull information from LinkedIn, analyze and classify the data, and store the data in its correct place. The main purpose of the scraping is to classify the information pulled from LinkedIn into categories. These categories as we see it will include two important ones to take note of, with the first being any identifying information, and the second being a career milestone. LinkedIn will likely not make it obvious what each career milestone we are pulling entails, so this module will check and place the data in the right spot once it is pulled. This LinkedIn scraping will be used from an HTML scraping tool to get the raw HTML from a user's connected LinkedIn profile which will then be placed inside the database that holds all information in an organized manner.

4.1.5 Statistical Graphing

The statistical graphing will be a function that is done semi-automatically, with the dashboard module needing to be able to calculate how many students have earned an internship, externship, co-op, or research/experiential learning opportunities. As this will be displayed in multiple possible ways, the dashboard will achieve this by keeping a total count of students and consistently updating the number who have had this internship. This way, the two values it needs are always available to display the results as a fraction or as a percentage. Also, the internship will be checked using a flag attached to the student, to allow for quicker calculation when running through all of the students.

The module will already be storing the total number of student accounts, and it will now store the number of students who have received a job offer. Then, it works like the previous requirement in how it will check and calculate the results.

4.1.6 Database Handler & Database

The database is where all information will be stored. The database will then communicate with its database handler who then distributes all the information to the designated areas such as the user profiles and the dashboard. The database is the most important part of the website because, without the database, there would be nothing to display about anything or anyone.

4.2 Module and Interface

In this section, we will go over the multiple modules and interfaces we will be using in order to implement our software product. These modules are the primary pieces to make the program work, and we will break each one down into the different parts and methods each module will need.

4.2.1 LinkedIn Data Scraping Module

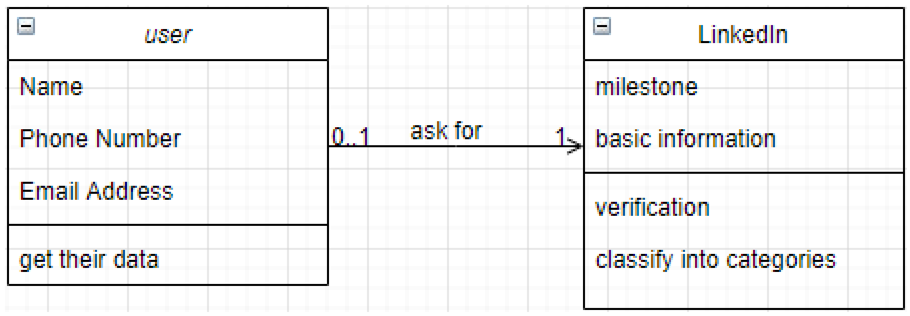

Figure 4.2.1

The LinkedIn module must be able to pull information from LinkedIn, analyze and classify the data, and store the data in its correct place. Therefore, the LinkedIn module has two main functional requirements:

- Pulling data from LinkedIn servers
- Classifying the data into categories

For the first requirement, once the users ask for data from their own interface, a request will be sent to the LinkedIn server using its API. Then it will verify the result that is sent, and unpack the result into temporary memory. These stored results will then be used for the other functional requirement, and this step will typically be done automatically, with the choice to scan for new data manually.

After classifying the data, which has been stored at the last step, into categories, these categories as we see it will include two important ones to take note of, with the first being any identifying information (LinkedIn API uses email), and the second being a career milestone. LinkedIn will likely not make it obvious what each career milestone we are pulling entails, so this module will check and place the data in the right spot once it is pulled.

The result that was pulled from LinkedIn successfully is the most important step of the whole module. We are trying to pull data from LinkedIn using different ways to make sure the success rate. Although we have a rough prototype, changes will be made within our modules to ensure that further development will be modular and reusable.

4.2.2 Statistical Graphing Module

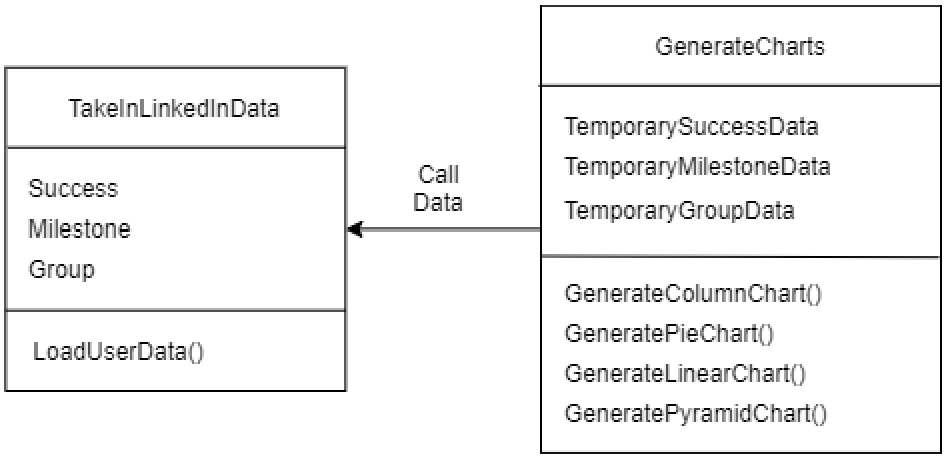

Figure 4.2.2

The statistical graphing module will be able to take in the data which is pulled from LinkedIn, analyze it, and then generate statistical graphs based on the milestones of the userbase. In order to achieve this purpose, this module will have two responsibilities:

- Taking in data that pulled from LinkedIn
- Generating statistical charts

The first responsibility will interact with the LinkedIn module. First, the LinkedIn module will pass some data it pulled, and the statistical graphing module will check if the data is valid. Then it will analyze it, and store the data in a temporary data structure for the convenience of calling and reading the information back.

After gathering and storing the data, the second responsibility will check which kind of charts that user wants to have, then call the corresponding charts' function, the charts will show the statistical information clearly, and make it easy to understand. After generating the charts, this module will interact with the user interface, and the charts will be shown in the Dashboard bar.

The TakeInLinkedInData entity has three variables: Success, Milestone, and group, and one function LoadUserData(). The function will get data from the LinkedIn module and pass them into two variables.

The GenerateCharts entity consists three variables: TemporarySuccessData, TemporaryMilestoneData, and TemporaryGroupData, and four functions GenerateColumnChart(), GeneratePieChart(), GenerateLinearChart(), and GeneratePyramidChart(). The data stored in two temporary variables is passed by the TakeInLinkedInData entity, and four functions will create corresponding statistical charts using the data in temporary variables.

4.2.3 User Interface

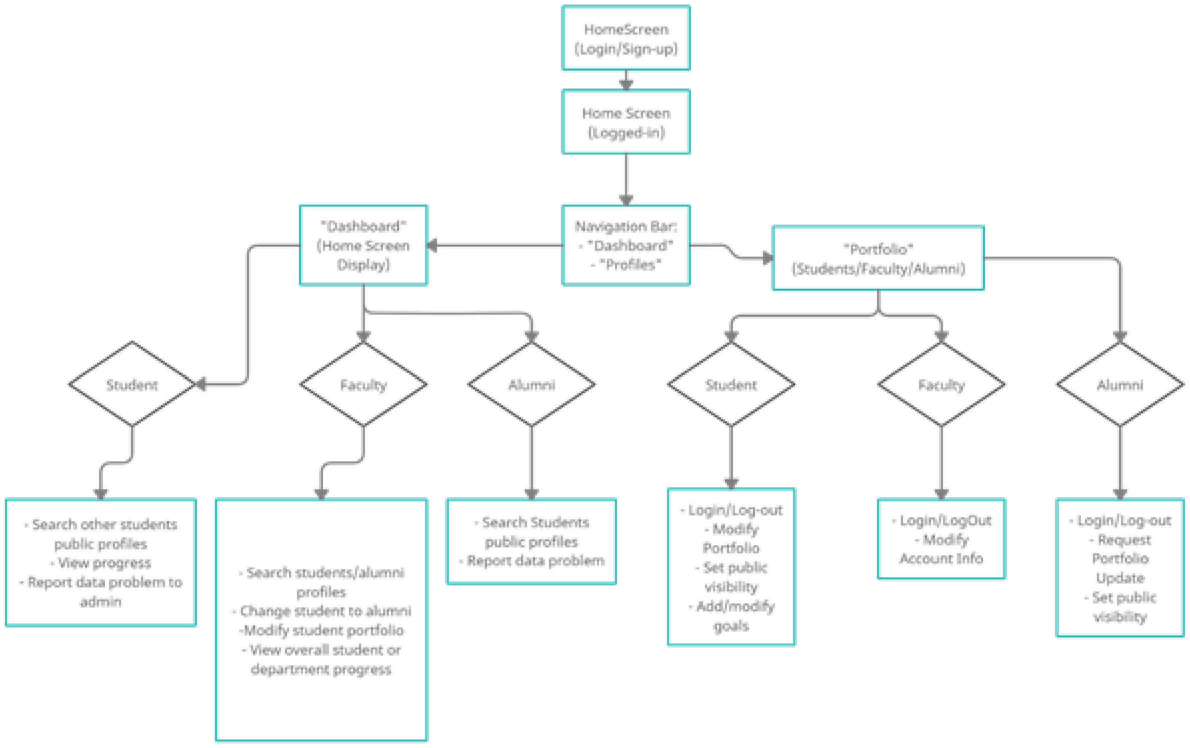
HomeScreen (Login/Sign-up)
Home Screen (Logged-in)
Navigation Bar: - "Dashboard" - "Profiles"
"Dashboard" (Home Screen Display)
"Portfolio" (Students/Faculty/Alumni)
Student
Faculty
Alumni
Student
Faculty
Alumni
- Search other students public profiles - View progress - Report data problem to admin
- Search students/alumni profiles - Change student to alumni -Modify student portfolio - View overall student or department progress
- Search Students public profiles - Report data problem
- Login/Log-out - Modify Portfolio - Set public visibility - Add/modify goals
- Login/LogOut - Modify Account Info
- Login/Log-out - Request Portfolio Update - Set public visibility

Figure 4.2.3

The user interface will be able to provide the user with all of the necessary operations they will need to do on the application from traversing through different public profiles to editing their own profiles, etc. With three different privileges on the application, the user interface will differ slightly for different users. However, it will remain the same for certain pages such as the home screen, which will prompt the user with a login or sign up option that will also be the same no matter who is logging in or signing up (i.e. students, faculty, alumni). As stated prior, once logged into the system this interface will provide different features depending on the user.

Student users, once logged in, will have a home screen (dashboard displayed) which will contain a navigation bar. The bar should have options such as “Dashboard” to return the user back to the home page, “Portfolio” that will take them to their own profile page, and a search bar to look up other students' public profiles. The portfolio page will allow a student to logout, modify their portfolio, set public visibility preferences to different features of their portfolio, and add/modify their goals. The dashboard will allow the student to view their own goals and progression toward them as well as being able to report an issue to administrators.

Alumni privileges will have the same options across the navigation bar as student users. However, upon entering into their own portfolio they will not have any goals to add or modify, rather they will be able to logout, set public visibility preferences to aspects of their profile, and request to have their portfolio updated. Going back to the dashboard, alumni will be able to search students and other alumni profiles as well as report a data problem to administrators.

Faculty will be the most different as these users will have the highest level of privileges on the system. Upon login, a faculty user will also have the same navigation bar. However, when searching alumni or students the faculty member will be allowed to make changes to their profiles including changing a graduating student’s privileges to that of alumni, or edit any other parts of a profile necessary. The dashboard for faculty members will also provide them with the option to view student profile progress or overall department progress. If the user goes to their profile page, they will be able to logout or modify their profile (username, password, etc.).

4.2.4 Database Handler Module

The database handler module will be implemented in order to easily handle reading and writing to the database our system will run off of. The primary purpose of this handler is to keep all database operations in one place, so if the database is ever worked on or modified, the changes will only be seen in one module. In order to achieve this purpose, the module will have the following responsibilities:

- Adding and removing information to the database
- Modifying information inside of the database

- Reading information from the database

These three responsibilities, while basic, will allow the rest of the modules to be sent information stored in the database, as well as send requests to change the database without having to worry about each module connecting directly to the database.

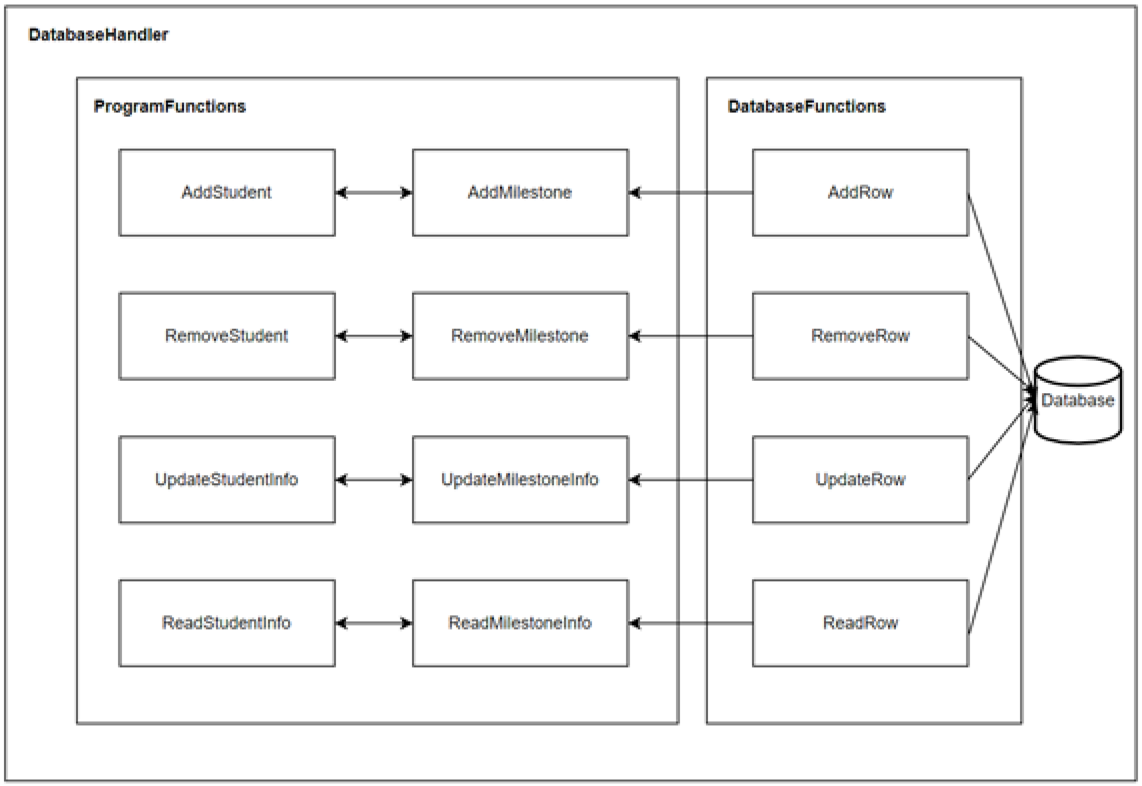

Figure 4.2.4.1

As can be seen in Figure 4.2.4.1, the three responsibilities are then broken down into pieces of two different submodules within the database handler. These submodules are the Program Functions, which will include the public functions the other modules will have access to. The other module is the Database Functions, which will be internally accessible to the database handler, with the only other connection to the module being the database itself.

The Program Functions will primarily focus on the two different entities we will be tracking in the database: students and milestones. For each of these, the functions accessible to the other modules are to add, remove, update, and read a row of information from the table. Each of these will be directly related to a private function in the Database Functions submodule, as both students and milestones will use similar functions to one another. In this regard, the Program Functions will invoke the Database Functions to actually connect with the database.

The Database Functions will consist of an AddRow, RemoveRow, UpdateRow, and ReadRow. The AddRow function will return a boolean value, with True being success, and False meaning the row was unable to be added. The RemoveRow function will return the row being removed stored in a custom data structure based on the entity being removed. The UpdateRow function will update a row, and return a boolean once again to show the success of the operation. Finally, the ReadRow function will return the row being requested in a data structure based on the entity being read.

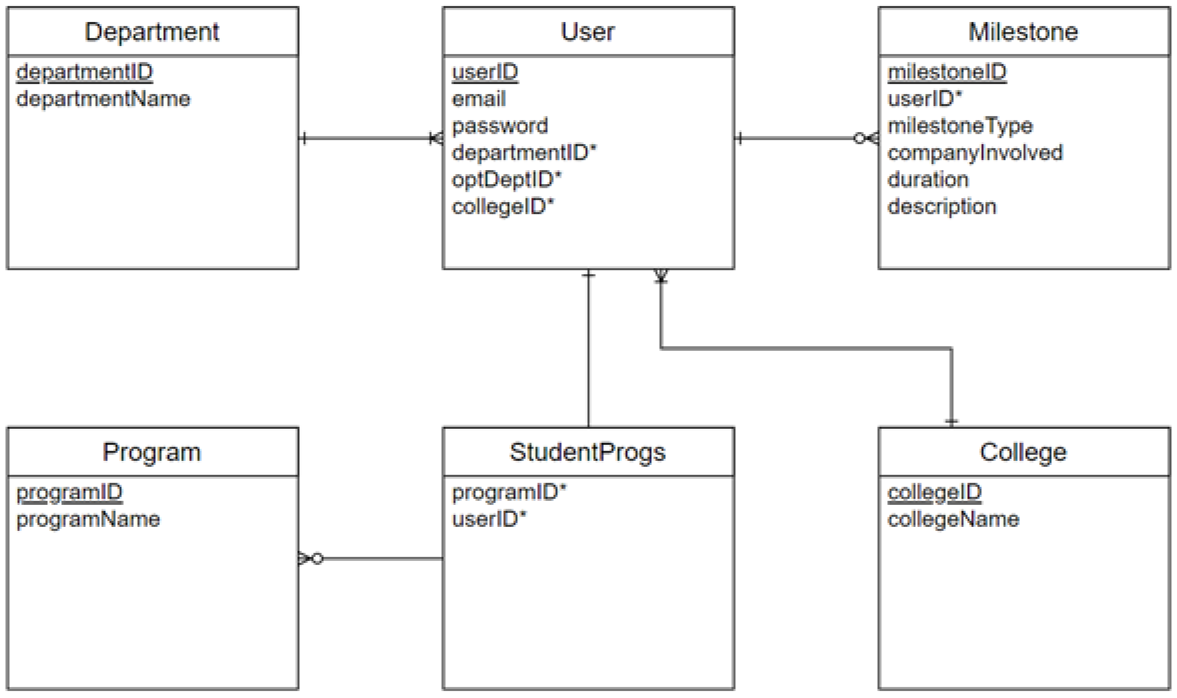

Figure 4.2.4.2

Figure 4.2.4.2 illustrates the different entities we expect to handle within our database. These are the pieces that will hold the information we expect to use for our statistical analysis, as well as the student profile page displays. The user will be the primary entity everything else is centered around, as this is a tool based around user information. In the future, we do expect that companies themselves can be added as entities to allow for better filtering tools, but for now, companies will remain in the milestone information that they apply to.

5. Testing

For this project, we have implemented integration and unit tests, and currently implementing usability testing. Unit testing would test every part of our code to check whether its behavior meets expectations and that no unforeseen problems have occurred. Our integration testing examines how the individual components of the project work together to produce the required functionality. In our example, this applies to how LinkedIn data is passed through the HTML file to the database. Usability testing evaluates the user-friendliness of our applications by asking people who are not familiar with our projects to browse our website and complete various tasks. This exposes any problems with the user interface and other website components. Usability testing is very important to our project because it ultimately simulates how our end user will interact with our product. Considering the importance of this point, our test plan focuses more on usability testing.

5.1 Unit Testing

Unit testing is done to ensure that the components and modules of the application are working properly and returning the desired functionality. Making sure that each component functions correctly on its own is essential when it comes to combining each unit together in order to have an application that works as needed. By performing specific unit tests to different functions and operational procedures, any issues can be resolved before the application moves forward, thus providing a cleaner system. For our application, we plan on testing the individual functions and procedures belonging to each of the different main modules of our application to make sure they work as they are intended to. By doing this we will be able to gain a clear understanding of the behavior of the individual units that make up our application and fix issues that arise ranging from inappropriate handling of different types of input to problems that could crash the application. For this project, the individual modules that will be tested are the sign-in module, user search module, user profile module, dashboard module, and our database. Each of these modules must work as they should by themselves before they can work together to form the overall functionality of the application as a whole.

5.1.1 User Searching

In our application, this component is meant to allow a user to look up others who are pursuing degrees in the same or related fields. Entering the name or email of a person should provide a list of those who are registered users and allow the individual making the search to view their profile. The person making this search should not have the privilege to change or modify that user's profile but rather only view what is contained on it. To test this unit, we first plan on providing the search form with random inputs including characters such as '?', '/', '..', etc. and different phrases. Also, because this module needs to access the database, we need to provide different inputs that may allow a user to access parts of the database that they should not be able to. More specifically, this would be done by manipulating the SQL queries that parse through this database with different inputs such as “admin’--”, or “‘OR 1=1’--” which if not handled correctly could give the user admin privileges to the system or trick the query into retrieving all information because ‘1=1’ results to true and the query would no longer treat the rest of itself after because ‘--’ is a comment indicator. It is little things like these that could breach the confidentiality, integrity, and even the availability of our system.

5.1.2 User Profile

The user profile module of our application is meant to provide the user with a detailed representation of their goals and milestones. Here the user should be able to add/modify things on their profile to keep it updated and store personal information as well as information that will be viewable to other people when they use the search module. Our plan for testing this unit will include setting visibility permissions and making sure others who search for that user cannot see the elements set to private and only can see those set as public. In addition, we will be implementing testing to make sure invalid input(s) does not get added as a milestone to a user's profile. We will also be testing to make sure that every milestone an individual has on their profile is displayed as well. In order to do this we will be providing bad random input such as words composed of a random sequence of special characters, i.e. “?//[{@&”, etc. We will also create users with several milestones to ensure that parsing through a user's milestones collects each one. We also will need to test and make sure if a user has 2 of the same milestones that they are not repeated on their profile. Overall, our unit testing for the users profile will consist of creating different test users, where some of these users have multiple milestones, and where some try to input invalid data to add to their profile. By doing this we will be able to guarantee that a users page cannot have data that does not make sense and also that whatever information they want to display is displayed to others and sensitive information is only viewable by that user.

5.1.3 User Login/Register

Within the user interface module, we must test that users can register and log in. As a user comes to the home screen, they will be prompted to Log In using the top button. Once they are here, they will have the option of continuing with logging in or registering. We will need to ensure that when the user is logging in, the email field will check to be an email. If something else is passed in, we have to reject the attempt at signing in before it ever involves our database lookup. The password, however, can be any string, so that will not be checked for any format. The other primary testing for the login, at least, is to ensure it finds an account that exists, and this can be tested by passing in known emails and unknown emails and seeing the known ones give different results.

From the login page, there will be an option to register. This page is similar looking to the login page, except that it is much more important for testing. While the login page is basic in its testable pieces, the registration page must test the information that is put in to ensure our database remains consistent. This includes checking that an email is put in, which can be tested by sending in strings lacking the ending for an email string. The next step is to ensure that the password field is being matched to a repeat field for it, as we will not be displaying the password on the screen. This allows the user to ensure they are not mistyping the password and lets us confirm that is actually what they want the password to be. Considering the rest of the options are selections rather than raw string inputs, the rest of the fields do not need to be tested for any kind of format. If the email and password are not tested, though, our database could end up with all kinds of inconsistencies.

5.1.4 User Dashboard

Student users, once logged in, will have a home screen (dashboard displayed) which will contain a navigation bar. The bar should have options such as “Dashboard” to return the user back to the home page, “Portfolio” that will take them to their own profile page, and a search bar to look up other students' public profiles. Going back to Dashboard, once the users click the button ’Dashboard’, the dashboard will allow the student to view their own goals and progression toward them as well as being able to report an issue to administrators. Furthermore, he is able to search other alumni profiles as well as report a data problem to administrators through the buttons on the page. Except for the student users, faculty will be the most different as these users will have the highest level of privileges on the system. Upon login, a faculty user will also have the same navigation bar. However, when searching alumni or students the faculty member will be allowed to make changes to their profiles including changing a graduating student’s privileges to that of alumni, or edit any other parts of a profile necessary. The dashboard for faculty members will also provide them with the option to view student profile progress or overall department progress. If the user goes to their profile page, they will be able to logout or modify their profile (username, password, etc.).

5.1.5 LinkedIn Data Handler

The LinkedIn data handler module will be a core module to the system, as it is what is providing the data for the rest of the system to use. This data will typically be scraped from student LinkedIn profiles, and when involving web scraping, there is a chance of running into errors. For this reason, the LinkedIn data handler must be tested at each of its basic functions to ensure everything remains consistent. This ultimately means that first, the URL that is passed in as the user page must be checked to ensure that it is a user profile, and not some other site. From there, it must pull the career experience data from LinkedIn and ensure that the data is actually present before continuing. Otherwise, the system may try processing null data due to an error in the scraping, whether it be due to a bad URL given or a lack of experience. Finally, the system must test that the data that has been pulled can be classified into our database categories, Company and Milestone Type. Once all of this has been checked, the system will then be allowed to insert the data into our database. This will help keep inconsistent scraping from causing errors and inconsistencies, as well as ensure the LinkedIn module is working properly.

With all of the pieces of our unit testing laid out, we feel that we have a good plan for handling each part of our program. These tests will only ensure that the individual modules are working properly on their own, however. In order to ensure the modules work well with one another, we move on to talk about integration testing.

5.2 Integration Testing

Integration testing is connecting all the components together to make one system that works together. Once the components are connected together, tests can be made to check if they are acting together as expected. The main focus of the integration testing is to check the communications between the user's inputs from the frontend, backend, and database. These tests will be able to show the interactions between each module as well as show our expected results from each test.

5.2.1 Account Login

Users on this website will be able to create their own accounts in the system. Once an account is created the user will be able to login with an email address and password that will be kept in the database. This part of the system is able to keep track of who is logged in as well as all the information the user enters while logged in. Overall, the registration of a user is the starting point into the rest of the system and connections between the rest of the modules.

Integration Test	**Description**	**Expected Result**
User Registration	A user can register an account	- A user will be able to enter all information needed to create an account - A user's information is then added and entered into the database and will be able to login with Email and Password

User Login	A user can login to their account that that registered	- A user can login with their Email and Password - A session will be created where the browser keeps track of who is logged in - A user then has their own profile page they can add personal information to
User Logout	A user can logout of their profile	- A user can logout of their profile they logged into and will have to sign in again to access their account - The session created in the browser for the user will end after signing out

5.2.2 Dashboard

Both registered and non-registered users are able to access the dashboard module of the website. Anyone is able to choose the fields that are incorporated into creating the graphs. These graphs will display information from the database that is requested by a user. Expected results are that the information from the database will be displayed correctly in the graph requested.

Integration Test	**Description**	**Expected Result**
Pie Chart Display	A user can request a Pie Chart	- The information requested will be taken from the Database and displayed as a Pie Chart
Bar Graph Display	A user can request a Bar Graph	- The information requested will be taken from the Database and displayed as a Bar Graph

Pyramid Chart Display	A user can request a Pyramid Chart	- The information requested will be taken from the Database and displayed as a Pyramid Chart

5.2.3 Profiles

Inside each user's profile, a user will be able to add, edit and delete a milestone. A milestone is entered by adding what type of job it is as well as the name of the company. The milestone will then be put into the database after the user has entered all the information and hit submit. The expectation of our system is to correctly input each individual user's milestones. Another expectation is also to correctly display each user's milestones on their profile page. The communication between the modules and the database is extremely important and is expected to work smoothly throughout the system.

Integration Test	**Description**	**Expected Result**
Student User Adds a Milestone to their Profile Page	A student can add a new milestone to their experiences in their profile page	- New milestone is added to the user's profile page - The correct milestone is retrieved from the database and is viewable by the user
Student User Edits a Milestone in their Profile Page	A student can edit an existing milestone in their experiences in their profile page	- Existing milestone is updated in the database with the new information the user has entered in each field - New information is then updated on their profile page for the user to see

Student User Deletes a Milestone in their Profile Page	A student can delete an existing milestone in their experiences on their profile page	- Existing milestone is deleted in the database - The deleted milestone is no longer visible on the user page because it no longer exists in the database
Student User Changes the Visibility of their Profile Page	A student can change the visibility of their own profile from PUBLIC to PRIVATE and vice versa	- The visibility is changed in the database after a button is clicked - The current visibility of the user's profile will be seen from their page

5.2.4 Search

The search module is able to search the entire database for a keyword that displays any public profiles and displays their pages in a table. The expected response is that private profiles are not visible and that only public profiles are displayed. A user is then able to view any public information that a user has allowed to be public on their own page. Overall, the search module has to communicate with the database to correctly display information that has been allowed to be viewed publicly.

Integration Test	**Description**	**Expected Result**
User Search for Profiles	A user is able to search for other profiles	- A list of PUBLIC profiles will appear in the table that matches the searched criteria - All PRIVATE profiles will not be shown to Student Users

5.2.5 LinkedIn

The LinkedIn module is able to search a user's LinkedIn profile and populate our database with some data from the LinkedIn profile. Once this has been done, the milestones need to be checked for consistency of data to ensure our database module can handle inserting the data.

Integration Test	**Description**	**Expected Result**
Insert Pulled Data into Database	A piece of data can be formatted correctly to be inserted into the database	- A new milestone will be seen on the user profile when scraped - The milestone will fit into one of the four type categories

5.3 Usability Testing

Through usability testing, we can understand how actual users will interact with our web application and what obstacles we may encounter that we did not account for. Usability testing is done by recruiting volunteers who do not know anything about our website work and asking them to browse our website to perform specific tasks without following the guidance of the user manual we provide. Due to our team do not find any volunteers right now, our client, Dr. Wang, is going to be the volunteer and help us finish the usability testing.

We are going to use Zoom to perform our usability testing, and the testing will be recorded so that we could return and watch it again when needed, however, all members will present and take notes during the meeting. Before arranging the meeting, the user manuals will be sent to Dr. Wang, so that he has time to view and understand the actor he will play. The team leader will explain the process at the beginning of the meeting, and then all team members will mute their audio so that we do not interfere with the testing process and can better understand the first-time user's application experience.

From the usability testing, we will be able to gather important feedback to make our website more intuitive for new users. Taking this user input into account, we can see flaws and bugs that have never been seen before, so we can improve our product as a whole.

6. Project Timeline

Our basic overall project timeline started back in August of 2020, when we were introduced to our project and the client. Over the next couple of months we had weekly meetings to establish the project requirements with our client and fully understand their wants/needs. Once we knew more about the project we began to research how to develop our web application. After researching we began to develop a plan in order to create efficient solutions for our clients problems. We executed this plan by deciding on the best technologies to handle the needs of our requirements and designing our program architecture. Once these blueprints were created and we had a good proof of concept we began implementation. We had several issues come up during this phase that hindered our progress but we found ways to work around them and offered alternative solutions to the problem. The next step after completing our minimum viable product demo was to test and get feedback then improve the software accordingly. Finally, we presented our finished product to our client and they approved of what we had created. Although they were happy with the result there are still some minor things still left to do such as transferring server admin privileges but these things will be handled immediately. The main phases of our project is shown in the figure below

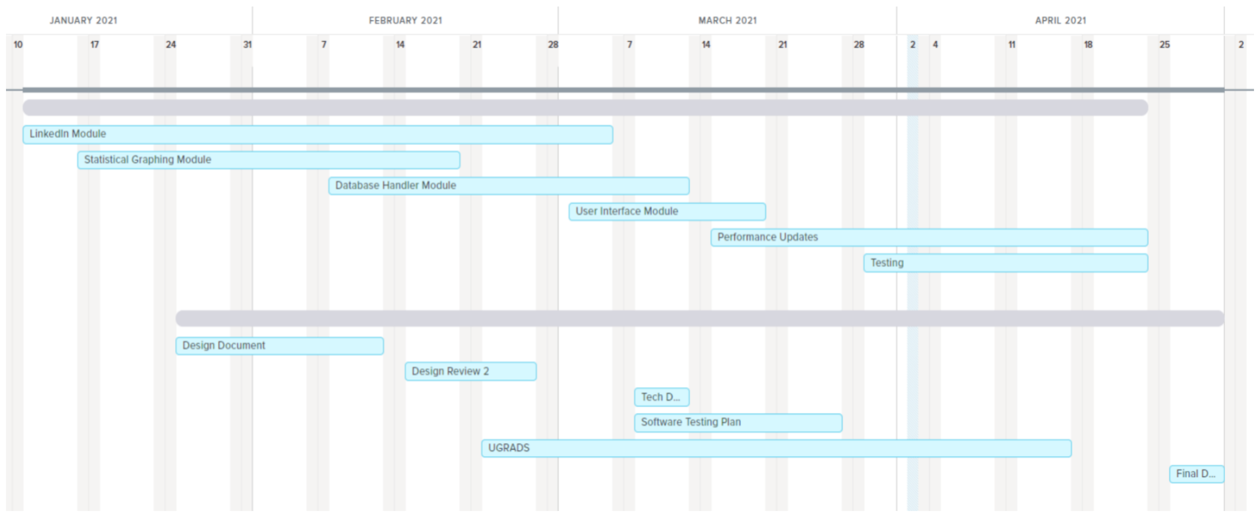

Figure 6.1

7. Future Work

If a 2.0 version of our web application were to be created, we really think that adding additional modules would be an excellent feature. Since time was limited and we had to build our web application from scratch, we were only able to create 2 modules. If another team worked on this project in the years to come, they would not only be provided with an already set up database and web application, but they would also have a template to how everything should look. This would allow them to add on to the project right away, rather than worry about how to integrate everything and make it functional because all of that stuff would be done already. Adding more modules can also allow students to work on even more topics that are discussed in class, which could increase their proficiency rate.

Moreover, we plan to expand our application to all departments of NAU. Although the task may seem difficult, once done it will greatly improve NAU students' chances of employment. Our members would like to see everyone who uses our application get the job offers they want.

8. Conclusion

The main problem that we face is being able to account for the highlights of students' careers. When a student gets an internship or a job, NAU wants to highlight that fact and encourage every student to pursue anything that will help them get a job in the future. Our client, Dr. Wang, wants to set a goal for NAU Engineering students that one hundred percent of students will have had at least one internship or experience in the degree they're pursuing by 2025.

Currently, NAU has no way of tracking students' progress, we don't know who has had internships, or if any student is even benefitting from their learning. Our team aims to give students a place where they can store the milestones of their time here at NAU. NAU currently has exit surveys in place that ask students if they have gotten a job offer since graduation. There are a couple of problems with this, one being is that students might not have had a job offer one to two months after graduating. Another problem with this is that students might not always fill out the survey.

Our web application that we have made will make these processes a lot easier. Instead of exit surveys, NAU will have a place to see exactly what their students have achieved, and how it has affected their career for the future. This will both save time in not having to worry about every student filling out an exit survey as well as already answer some questions that students might not be able to answer in a survey. Our client could potentially put this into place for all colleges of NAU. NAU would then be able to show how many students have gotten jobs out of college. The future of this web application that our team has made could have endless possibilities for our client and how he plans to use it.

Our team has had a lot of fun working on this project together and we are excited to see it come to its last couple phases of development. We hope to see some improvement for this website in the future and are excited to see how our client will use it once it's in production. Lastly, we as a team have learned so much through this capstone class. We have all learned how to work as a team as well as fulfill a client's needs. We have been able to grasp some of what goes on in the real world as well as understanding how a software development process takes place. Capstone was easily the most realistic class here at Northern Arizona University.

Appendix A

Hardware

The two main programmers on our team developed on different systems but both coded in the same IDE. Both programmers used Visual Studio Code as a work environment for our web application. One programmer uses a Macbook Pro with a 3.5 GHz Dual-Core Intel Core i7 processor with 16 GB of memory. The other main programmer works on a desktop with a 3.6 GHz Quad-Core Intel i7 processor with 32 GB of memory. There is no need for a high powered machine to be able to run a web application or be able to program one either. Any basic computer should be able to run everything we use in this project.

Toolchain

Our team used Visual Studio Code for our work environment. VS Code makes it easy to run applications as well as see the folders you are working in. VS Code also has lots of plugins you can add to make programming in some languages easier. Inside our project we used Node.js for our backend. We installed some plugins to make running our program and finding files easier such as Express, Cors, dotenv, msSql and puppeteer. Express is a simple web framework that is easy to use and provides a lot of features to use in a web application. Cors is a node.js package that runs as a connection middleware for our project. Dotenv is a tool that allows us to put connection names and passwords in a separate file that then is easily accessible by our project. MSSQL allows us to connect to an SQL server through our backend. Finally, puppeteer is what we use to scrape LinkedIn to grab raw HTML that we then parse through to get the information we need from the user's profile. Puppeteer is the primary tool for our application's scraping function. These tools run together to allow our project to run smoothly and efficiently.

Puppeteer Scraping:

The puppeteer Node.js module is an important tool to understand. Ultimately, it will open a web browser in the backend and navigate different pages, and return the HTML of the page it navigates to. With this, if you want to expand it to scrape from other pieces of a LinkedIn profile, you can look at the HTML of the page that is scraped and parse it in a similar fashion to our LinkedIn scraping tools. A simple way to do this with the code already provided is to log the HTML that is returned before we parse through it, and find the section you want to read from and parse that out as well.

Setup

Running the web application on a local machine:

Step 1: Open up VS Code and open your project folder

Step 2: Install all the plugins for Node.js (Express, MySQL, Cors, dotenv, puppeteer) using npm install

Step 3: Make sure the connections to the LinkedIn account and MySQL server are proper connections

Step 4: Run the command “npm start app.js” where app.js has the startup for the backend server

Step 5: Open up your files to see how your website acts

Running the web application through the NAU servers:

Step 1: SSH into webapps.ac.nau.edu using your NAU credentials

This uses the command:
```
ssh <nau_id>@webapps.ac.nau.edu
```

Step 2: SSH into the egrcn instance at the localhost

This uses the command:
```
ssh egrcn@localhost
```

Step 3: Inside of the Server directory, find the server container name

This uses the command:
```
podman ps -a
```

Step 4: Start and stop the container to start and stop the site

This uses the command:
```
podman start <container-name>     OR
podman stop <container-name>
```

The server should now be up at https://ac.nau.edu/egrcn/

Production Cycle

When making changes to the code and wanting to deploy the changes, there is a simple process to follow within the server after you have SSH’d into the egrcn instance as seen in the Setup section above.

Once the code changes have been made, the files must first be moved to the server. You can make changes within the server using command line editors, but this is only useful for quick fixes. To start moving the files from the local machine to the server, you must first SCP the files into the server. They you must SSH into the webapps server, and SCP the files into the egrcn instance once there. This can be done using the following string of commands:

```
scp -r /directory/to/send <nau_id>@webapps.ac.nau.edu:DirectoryName
```

```
ssh <nau_id>@webapps.ac.nau.edu

scp -r DirectoryName egrcn@localhost:DirectoryName

ssh egrcn@localhost
```

Now the files will be in the server from the local machine. From here, you will use the Dockerfile that is already made to build the Docker container. While in the server directory, you will build the server container image. Then, once the image is built, you will build the container and run it. This is done in two commands, and they are:

```
podman build -t <image_name> .

podman run -d --name <container_name> -p 9007:9007 <image_name>
```

Using these two commands will start the server up with the files you had run.

It is important to note that only one container can and should be running at a time, as this is what is handling requests to the web server at the app domain. Therefore, it is important to check if any containers are running before running the new one. A list of all containers can be found using:

```
podman ps -a
```

Some useful container commands are listed here:

podman rm <container_name>	= removes a container
podman rmi <image_name>	= removes an image
podman start <container_name>	= starts a container
podman stop <container_name>	= stops a container