Software Design Documents

Feb. 10, 2022

Team CareerNet

Team Sponsors:

Dr. Andy Wang,

José R. Díaz Aquino

Team Mentor:

Han Peng

Team Members:

Carter Taylor,

McKenzie Clark,

Carmen Montalvo,

Ran Li

Version 2.0

Table of Contents

1.0 Introduction

Creating an environment where college students can achieve their goals after completing an undergraduate degree is a united objective for college deans. Currently, there is a lack of products that address this desired result, leaving a gap in the market for this project. One way to achieve that goal is to encourage external learning outside of the curriculum. External learning could take the form of a job, internship, or research related to a student's field of study. Observing students' external learning as a collective can prove to be difficult as there is currently little to no accessible data on these experiences.

CareerNet is a web application that scrapes student information from popular career-building websites and surveys and organizes the relevant data to be easily accessible by college deans so that they may understand the career growth of their students. Tracking a student’s career milestones is valuable to any university because it can give insight on not only the student but the development of the degree plan they are in. This allows a college to get a better grasp on what degree plans are succeeding and how they can improve upon them. This product applies to all colleges interested in creating a student success-focused university.

This project’s sponsors are Dr. Andy Wang and Mr. José Díaz Aquino. Dr. Wang is a dean and professor in the College of Engineering, Informatics, and Applied Sciences (CEIAS) at Northern Arizona University. Mr. Diaz Aquino is the career development program director of CEIAS at Northern Arizona University. These sponsors are responsible for looking at the success in and out of the classroom for more than 2500 students in 24 different undergraduate programs for CEAIS alone. Doing this is vital in gauging the success of the program, college, and university as a whole and allows them to make changes to the curriculum as needed. Together their goal is for all NAU students to have some form of external learning experience within their field of study by 2026. Achieving this goal places importance on helping students better themselves to accomplish their desired career path after graduation.

One of the most important aspects of this web application is implementing a secure user-friendly interface that allows admin and faculty to upload, retrieve, store, and view student or school data. With CareerNet being a web application, it needs to be compatible with the layout of the desktop being used and will be stored on an NAU server. The following section will give an overview of how all these features will be implemented.

2.0 Implementation Overview

The clients, Dr.Andy Wang and Mr.José R. Díaz Aquino, commissioned the team to build a web page-based product that tracks the success of students and alumni within a career field. It automates key data collecting and updating on a simple graphical interface that configures and manages the students' process while also displaying visualizations, e.g., graphs, tables, and timelines to review the evolving progress on each goal. A navigation bar at the top of the page allows the user to maneuver through all the functions of the web application. Structuring this product in this manner will provide simplicity and make every one of the features available to the customers at a glance.

2.1 Technologies

In order to implement an achievable product, the team selected some software and tools based on how useful they are in designing and implementing the final product.

- ReactJS - This is used in forming the front-end as it is a great way to build user interfaces components. It is used in creating dynamic and appealing web applications. React also allows the team to create reusable UI components and provides an easy-to-use front-end framework. Using ReactJs can integrate HTML, CSS, and JavaScript effortlessly.

- NodeJS(Express) - NodeJS with an Express framework is used as the back-end language. Because Express NodeJS can communicate with the Mysql database and front-end, it can perform all of the necessary functions and data updates, along with uploading CSV documents.

- MySQL - As the database language, the team decided to use MySQL because it supports multi-threading, makes full use of CPU resources, and supports multi-users. It optimizes the SQL query algorithm, effectively improves query speed, and provides management tools for organizing, checking, and optimizing database operations. MySQL can handle large databases with tens of millions of records which makes it a perfect language to use for storing, updating, and managing this product's data.

Based on these useful technologies, implementing the key functions of the CareerNet web application will prove to be successful using the correct architecture.

3.0 Architectural Overview

In order to implement the solution to the dean’s problem, the team has created the best architectural design for this web application. The project has many different connecting parts in order to accomplish all tasks requested by the user.

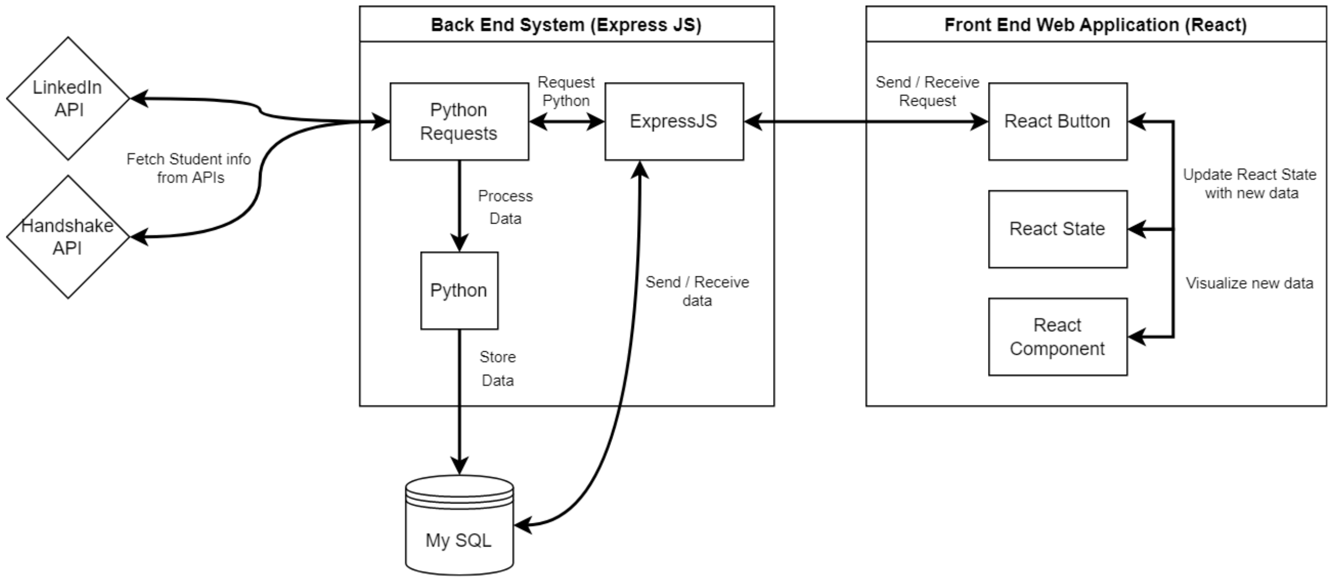


Figure 1: Communication flow between each component

Descriptions for each component are as follows:

- *Front End Web Application (ReactJS):* The interface the user interacts with in order to access features of the application. After the user is logged in, they will be able to access the features of the site. Each component of the front-end allows the user to communicate with the project's back-end system. Depending on which front-end component the user interacts with, they will retrieve/send different information to the back-end.

- *Back End System (ExpressJS)*: The back-end system will handle requests from the front-end system, performing logic to receive and send data with the front-end system. The back-end will act as a direct interface between the front-end and the data it needs from the database (retrieving or storing data from the front-end).
- *Database (MySQL):* The database will store the student information uploaded from the front-end system. The backend system stores and retrieves data directly from the database, returning the data to the front-end.

The flow of the application begins with the front-end being the user interface. As the user interacts with the front-end of the application, certain actions on the site will send requests to the back-end portion to either send or receive data. The back-end portion of the application is listening for these requests and will react accordingly based on what type of request the front-end action specified. Possible actions could be requesting student information from an entered student name, uploading new student data using a CSV to be stored in the database, or requesting existing student data to be enriched from LinkedIn and stored in the data. The back-end system receives these requests and will use the specifications from the front-end to then interact with the database. After interacting with the database, the back-end will return a response to the front-end allowing it to display the results.

4.0 Module and Interface Descriptions

After discussing desirable features with the client, the team has created the best approach to the situation. There are three overarching categories to depict the architecture of the system which are the front-end, back-end, and MySQL data management. Each of these is divided into subsections that are described in detail below.

4.1 Front End Web Application

The front-end web application is based on the main parts of data upload, data search, data visualization, and admin setting. This is also the most important part as it will provide the key functions of the web application. In this section, we will describe the services provided by each front-end component and how they fit into the architecture. Data upload includes data extraction from CSV and LinkedIn. Data search contains filters to filter data. Data visualization includes graphing of overall data and milestones for each student to present data more intuitively to the users. The UML in Figure 2 below diagram of all of the components for the front-end web application.

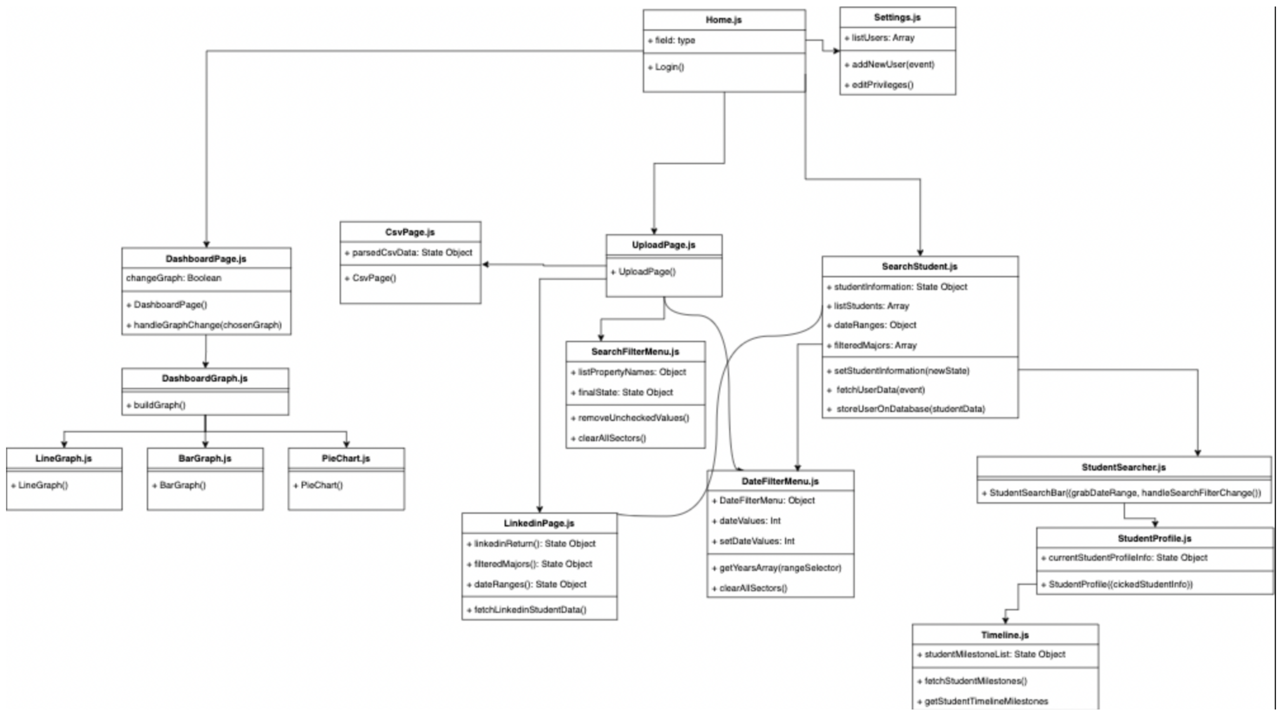

Home.js
+ field: type
+ Login()
Settings.js
+ listUsers: Array
+ addNewUser(event)
+ editPrivileges()
CsvPage.js
+ parsedCsvData: State Object
+ CsvPage()
UploadPage.js
+ UploadPage()
DashboardPage.js
changeGraph: Boolean
+ DashboardPage()
+ handleGraphChange(chosenGraph)
SearchStudent.js
+ studentInformation: State Object
+ listStudents: Array
+ dateRanges: Object
+ filteredMajors: Array
+ setStudentInformation(newState)
+ fetchUserData(event)
+ storeUserOnDatabase(studentData)
SearchFilterMenu.js
+ listPropertyNames: Object
+ finalState: State Object
+ removeUncheckedValues()
+ clearAllSectors()
DashboardGraph.js
+ buildGraph()
LineGraph.js
+ LineGraph()
BarGraph.js
+ BarGraph()
PieChart.js
+ PieChart()
StudentSearcher.js
+ StudentSearchBar({grabDateRange, handleSearchFilterChange()})
DateFilterMenu.js
+ DateFilterMenu: Object
+ dateValues: Int
+ setDateValues: Int
+ getYearsArray(rangeSelector)
+ clearAllSectors()
LinkedinPage.js
+ linkedinReturn(): State Object
+ filteredMajors(): State Object
+ dateRanges(): State Object
+ fetchLinkedinStudentData()
StudentProfile.js
+ currentStudentProfileInfo: State Object
+ StudentProfile({cickedStudentInfo})
Timeline.js
+ studentMilestoneList: State Object
+ fetchStudentMilestones()
+ getStudentTimelineMilestones

Figure 2: UML of the front-end components

4.1.1 Data Upload

For the front-end web application, data upload is the main process of getting the different data and presenting the raw data to the user. The web app will parse the data from CSV files to store data, and fetch data from LinkedIn to implement the function of fetching data.

4.1.1.1 CSV

CSV files are able to be dragged and dropped into the UI. The web app will parse the data to store it in an array of objects (Figure 4), with each row being an object, and each column being a key in that object (Figure 3). Once the user clicks the upload button, a pop-up will state the success or failure of the upload.

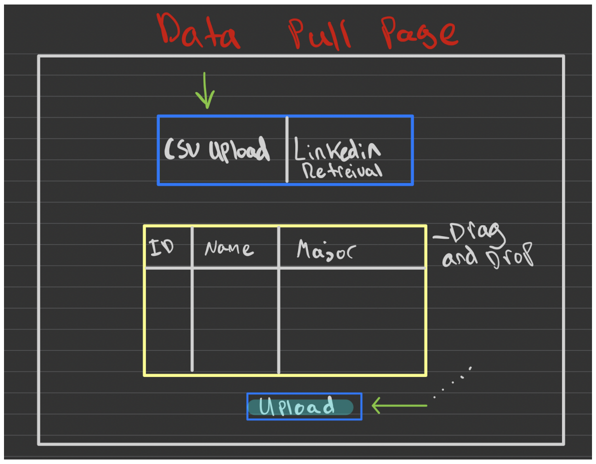


Figure 3: Sketch of Data Pull Page using CSV Upload

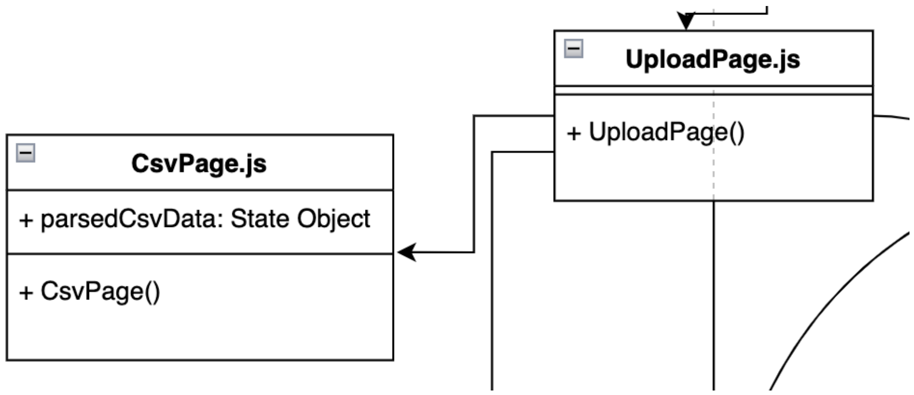


Figure 4: UML Diagram for CSV Upload

4.1.1.2 LinkedIn

Another functionality in this program is LinkedIn fetching which is displayed upon the upload page along with its applicable filters (Figure 6). Side menu filters are available to let users limit which student data they want to update from LinkedIn.

- Filter by Major: If filter by major is selected, a search bar pops up with all the different majors the user has access to.

They can check the majors to select which ones they are interested in fetching LinkedIn data for (Figure 5).

- Filter by Graduation Year: Users can give the graduation year range of the students they want to pull LinkedIn data from. If this is selected, LinkedIn data will be grabbed for the graduation range that was selected.
- Filter by last updated: Users can choose to pull LinkedIn data based on the last time the data for the student(s) was pulled. This function is primarily used to pull data for students that were added within the past day/week/year.

After fetching the data, the application will display a pop-up with the success of the retrieval or the failure to do so along with the name of the data that was not collected.

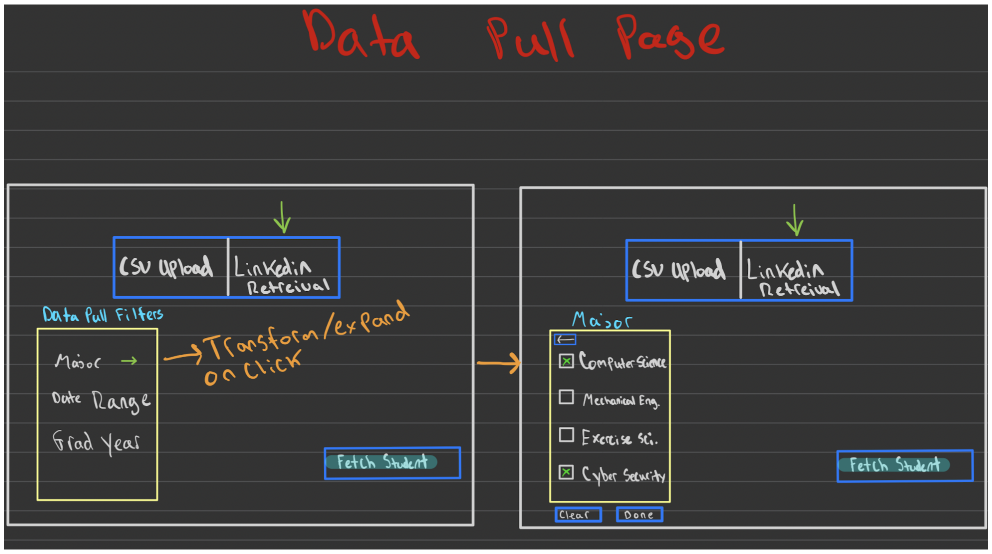


Figure 5: Sketch of Data Pull Page using LinkedIn Retrieval

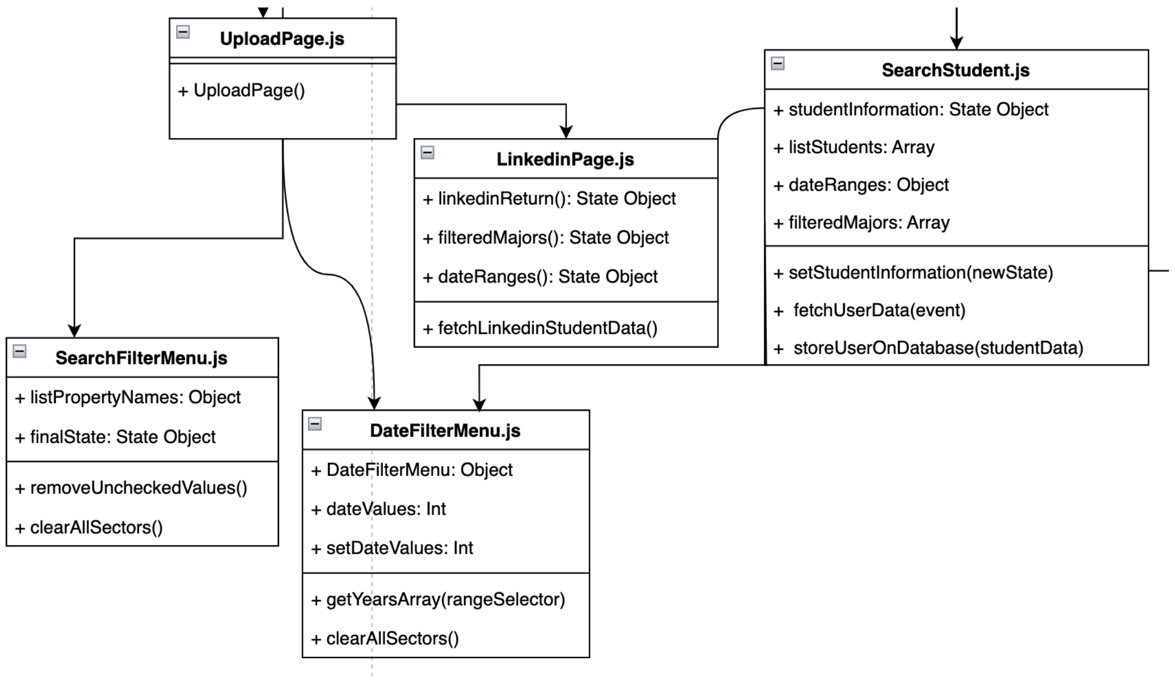

UploadPage.js
+ UploadPage()
LinkedinPage.js
+ linkedinReturn(): State Object
+ filteredMajors(): State Object
+ dateRanges(): State Object
+ fetchLinkedinStudentData()
SearchStudent.js
+ studentInformation: State Object
+ listStudents: Array
+ dateRanges: Object
+ filteredMajors: Array
+ setStudentInformation(newState)
+ fetchUserData(event)
+ storeUserOnDatabase(studentData)
SearchFilterMenu.js
+ listPropertyNames: Object
+ finalState: State Object
+ removeUncheckedValues()
+ clearAllSectors()
DateFilterMenu.js
+ DateFilterMenu: Object
+ dateValues: Int
+ setDateValues: Int
+ getYearsArray(rangeSelector)
+ clearAllSectors()

Figure 6: UML Diagram for LinkedIn fetching

4.1.2 Data Search

Another main function of the application is the data search. The goal of this feature is to allow the users to search for relevant data that they are looking for and record it. The web application uses filters to organize the data clearly.

4.1.2.1 Filters

Filter in data search will be the most important function to classify the data search. When users search for data on this web application, they first see a search bar where they can search by first and last name. If they enter only part of the name the database will display all the data corresponding to that entry. Access user options are then provided, where the access user can choose to view all career data for the student(s) of their choice. The UML diagram in Figure 7 shows the functionality of the data search.

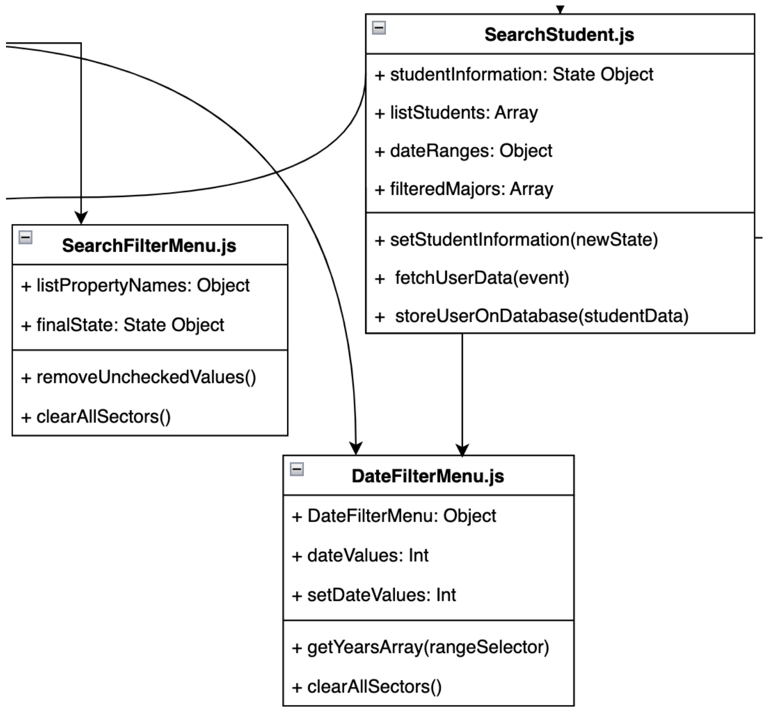


Figure 7: UML diagram of the data search

For the data search function, the team will extend a function called filter. The filters include filtering by major and graduation year. The Major Filters will show different majors, such as Computer Science, Mechanical Engineering, Applied Computer Science, Electrical Engineering, etc. (Figure 8). The users can select one or several majors, the database will match the student data of the same majors in the database, and then display the filtered data to the users. If user would like to reselect the majors they can simply click to uncheck the chosen major or click the clear button that will clear all the options they have checked. Date Filters is another choice filter, which is similar to that of the majors but with a dropdown showing the years the user can choose from (Figure 8). The user can select a year and the application will give career data for all students within that year.

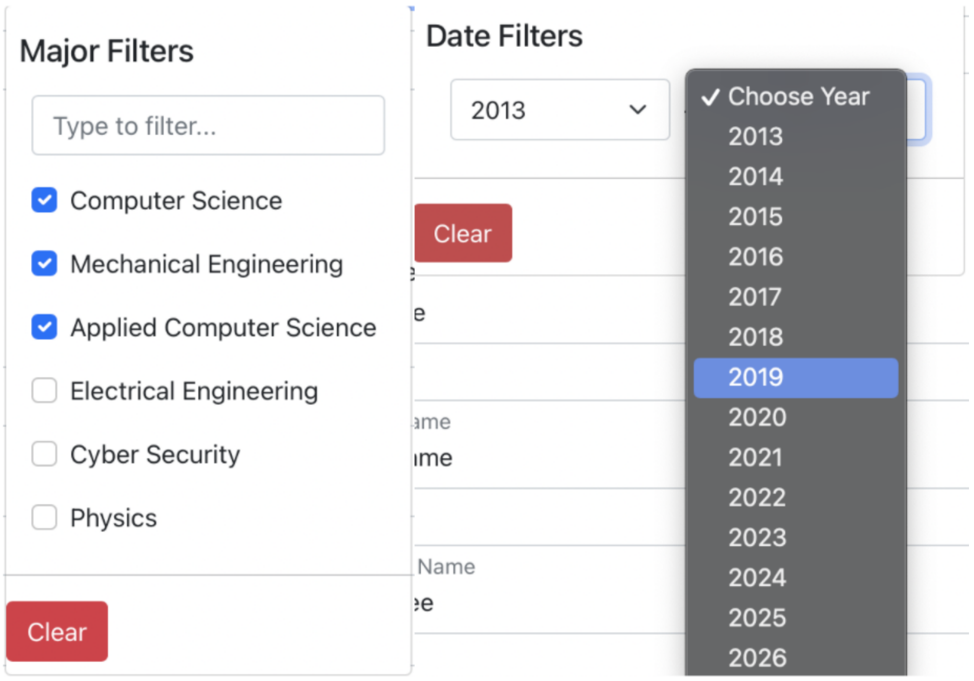


Figure 8: Model of data search filters

4.1.3 Data Visualization

Data visualization, as another major task within this web application, presents the data to the users in a more complete and clear way in two very intuitive ways: graphing the data as a whole and timelines of each student's milestones.

4.1.3.1 Graphing

CareerNet also implements a dashboard with customizable charts that include a line graph, bar graph, and pie chart that are changed through a dropdown box. There are two other filters on this page that allow the user to refine the data they would like to display upon the requested graph. One of these is selecting the majors the user would like to display on the chart while the other is what data within those majors they would like to view (Figure 9). The front-end will communicate these requests to the back-end in order to retrieve the filtered data before displaying the chart. The fonts, colors, and sizing are not customizable and default settings for this feature will be set by the team with the agreement of the clients. The functionality of this feature can be seen in Figure 10.

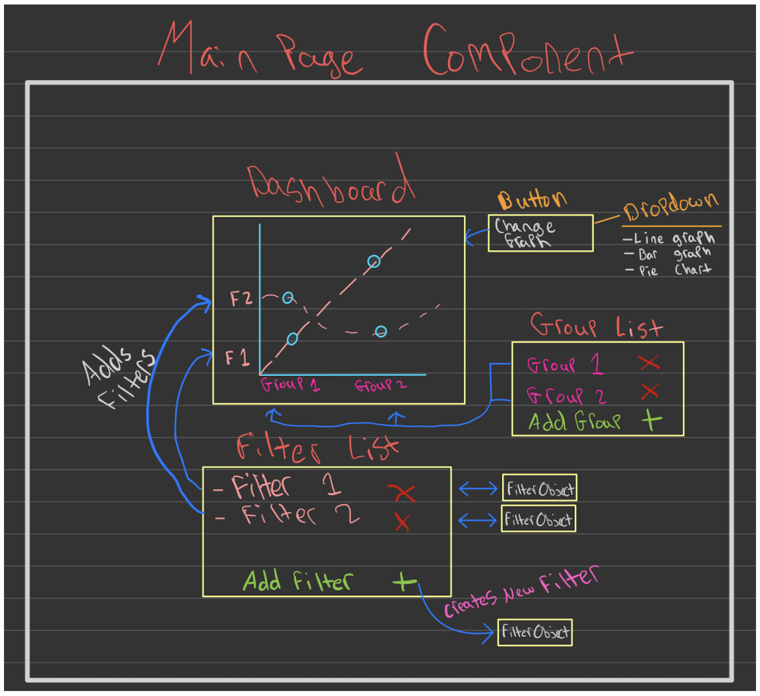


Figure 9: Sketch of the Main Page implementing the graph

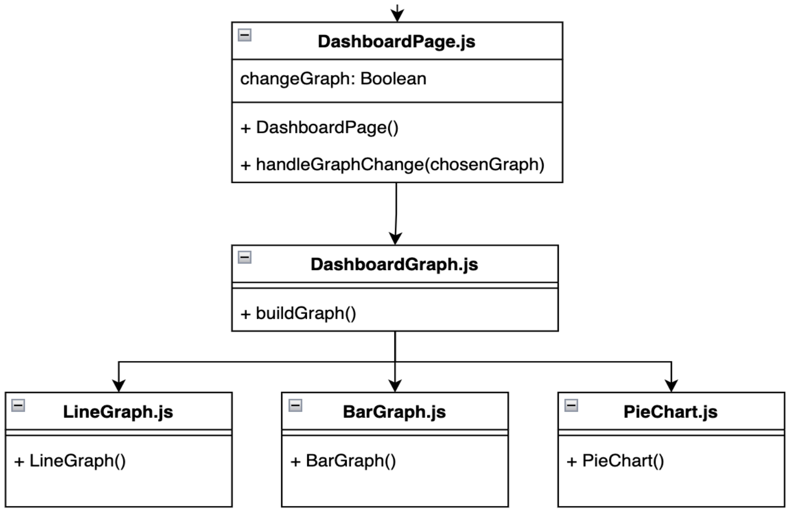


Figure 10: UML diagram for graphing data

4.1.3.2 Milestones

Along with showing graphs for the data as a whole, this web application will allow the users to view the milestones of individual students, i.e., their education, jobs, and external learning experiences. Once the student is searched there will be a view more button for each student that will redirect them to their respective milestones page. This page will display a timeline outlining all of the student's career experiences with a default vertical view and an optional horizontal view button. The process in which this is executed can be seen in Figure 11 below.

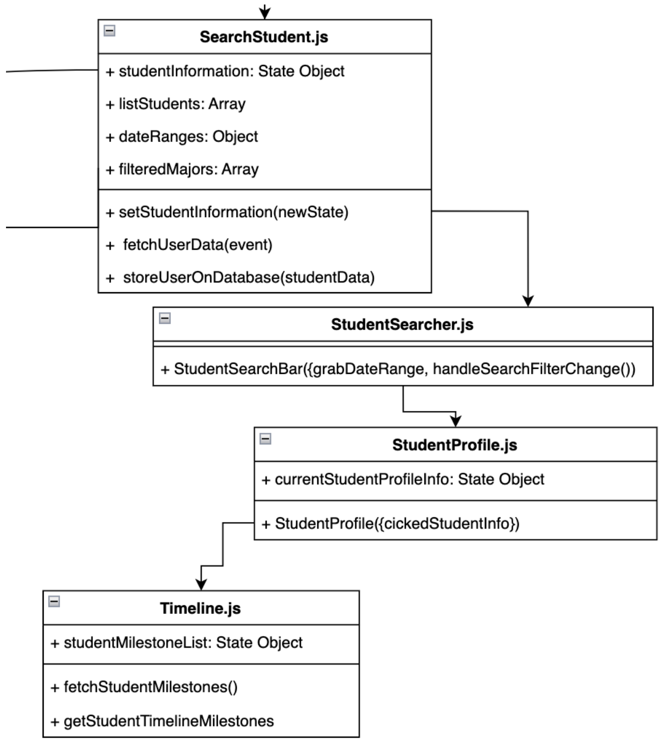

SearchStudent.js
+ studentInformation: State Object
+ listStudents: Array
+ dateRanges: Object
+ filteredMajors: Array
+ setStudentInformation(newState)
+ fetchUserData(event)
+ storeUserOnDatabase(studentData)
StudentSearcher.js
+ StudentSearchBar({grabDateRange, handleSearchFilterChange())
StudentProfile.js
+ currentStudentProfileInfo: State Object
+ StudentProfile({cickedStudentInfo})
Timeline.js
+ studentMilestoneList: State Object
+ fetchStudentMilestones()
+ getStudentTimelineMilestones

Figure 11: UML Diagram for showing Milestones

4.1.4 Admin Settings

Admin setting will provide this web application with the ability to grant privileges for special users. This function is very convenient for administrators to manage users.

4.1.4.1 Granting Privileges

An exclusive feature within this product is the admin settings page. This page will only be visible in the navigation bar to users with the admin role. On this page, the user has the option to add or edit an existing user(Figure 13). When adding a new user, the page will change and show a field in which the administrator will type in the new user's email and click the Add User button (Figure 12). A pop-up will show if the action was successful or if it failed.

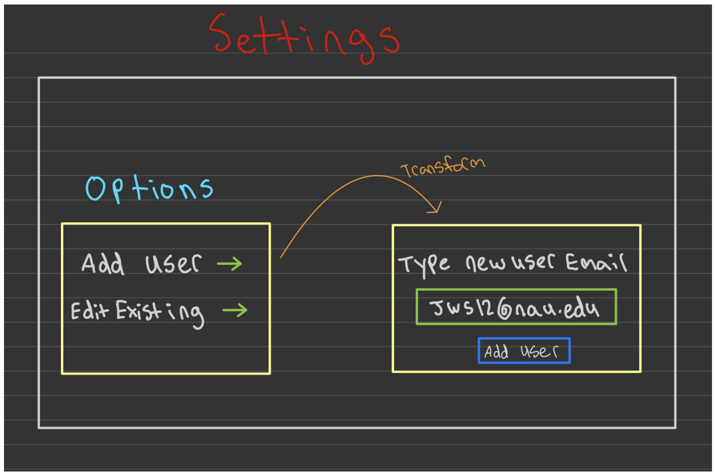


Figure 12: Sketch of Admin Settings using the Add user function

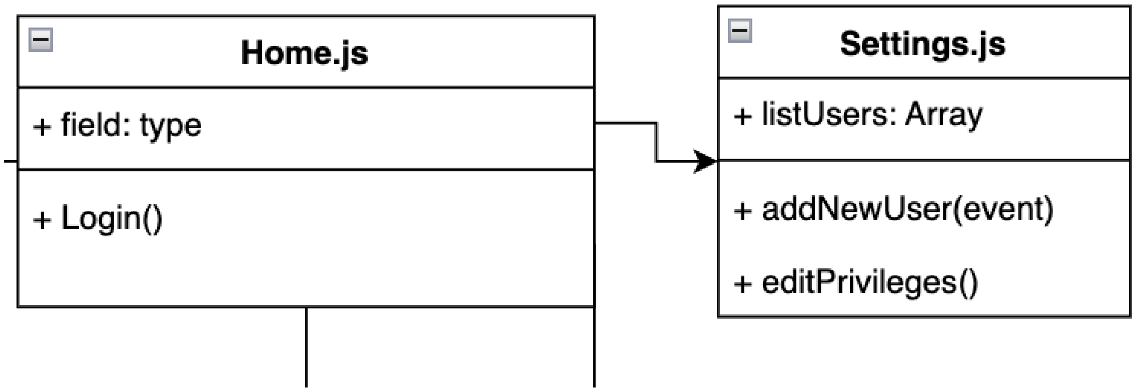


Figure 13: UML Diagram for Admin Settings

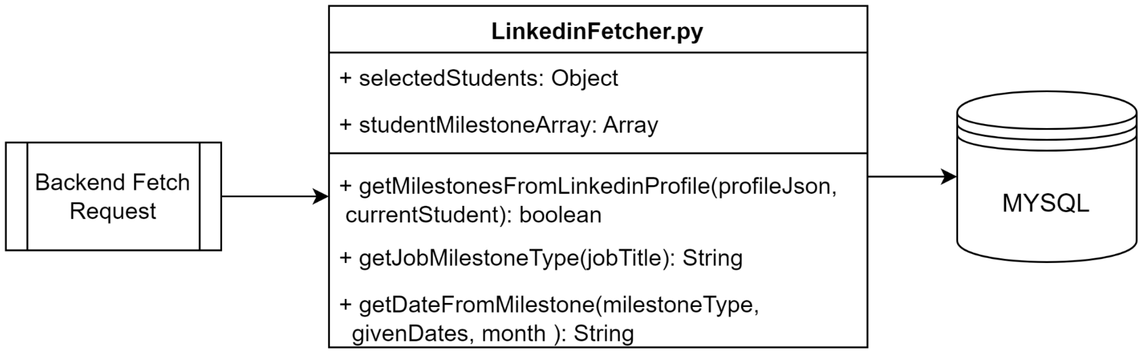
Editing an existing user will change the page similar to adding a new user where a field will be displayed in which the user will search for a current user with access to this system by using their email. Once the requested user is found, the user has the option to remove or add privilege by checking or unchecking boxes for each feature. They will then click the apply button to execute the changes and before this is done a pop-up will appear asking the administrator to confirm the changes they wish to make.

4.2 Back End Web Application

For the back-end web application, LinkedIn fetching is one of the main functions that grabs the data needed from the LinkedIn API and store it in the MySQL database. A CSV module also allows the users to upload CSV files to the back-end, which will eventually send the prepared CSV data to the database for storage.

4.2.1 LinkedIn Fetching

One main function of the application is the ability to fetch data from LinkedIn to enrich the existing student data with updated information of their career experiences while they are currently enrolled and after they graduate. After the initial student data is supplied through CSV upload with the students' name, ID, major, and graduation year, the data can be leveraged to fetch even more data from LinkedIn. The operation can be initiated from the DataUpload section of the front-end system and is detailed in the diagram below:



Backend Fetch Request
LinkedinFetcher.py
+ selectedStudents: Object
+ studentMilestoneArray: Array
+ getMilestonesFromLinkedinProfile(profileJson, currentStudent): boolean
+ getJobMilestoneType(jobTitle): String
+ getDateFromMilestone(milestoneType, givenDates, month): String
MYSQL

Figure 14: LinkedIn Fetching Module

As seen in *Figure 14*, after a request is made to fetch LinkedIn data from the front-end system, the express backend route uses Python to attempt to fetch LinkedIn data. The Python file grabs the necessary student data from the database before reaching out to LinkedIn to collect new milestone information regarding the student. If a profile is found matching the student, important milestones will be parsed from their LinkedIn profile and then uploaded and associated with the student's data in the database.

4.2.2 CSV

This application’s CSV module uploads CSV formatted documents to the system's back-end. The module then organizes the document’s student data to its appropriate storage space, while also checking the data for incorrect formatting or invalid data. If an issue occurs, the system returns an error message to the user, specifying what went wrong. The CSV module is a large component of the application's functionality, allowing the user to upload new students to the database as well as update student data via CSV files.

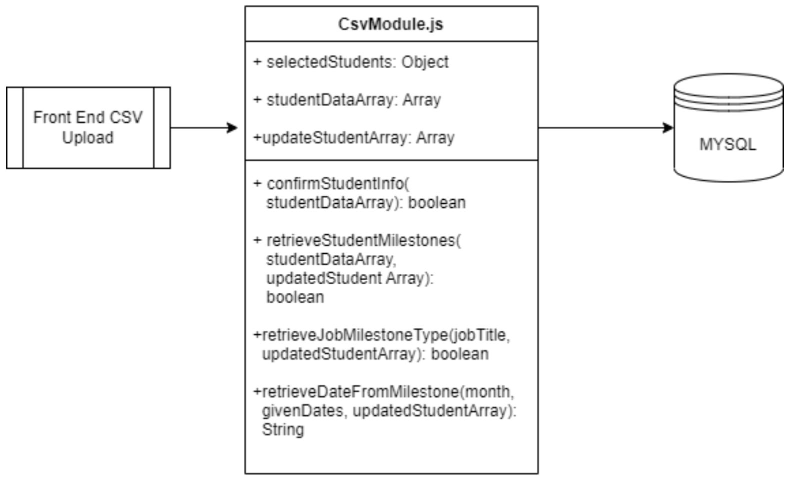

Front End CSV Upload
CsvModule.js
+ selectedStudents: Object
+ studentDataArray: Array
+updateStudentArray: Array
+ confirmStudentInfo(studentDataArray): boolean
+ retrieveStudentMilestones(studentDataArray, updatedStudent Array): boolean
+retrieveJobMilestoneType(jobTitle, updatedStudentArray): boolean
+retrieveDateFromMilestone(month, givenDates, updatedStudentArray): String
MYSQL

Figure 15: CSV Module

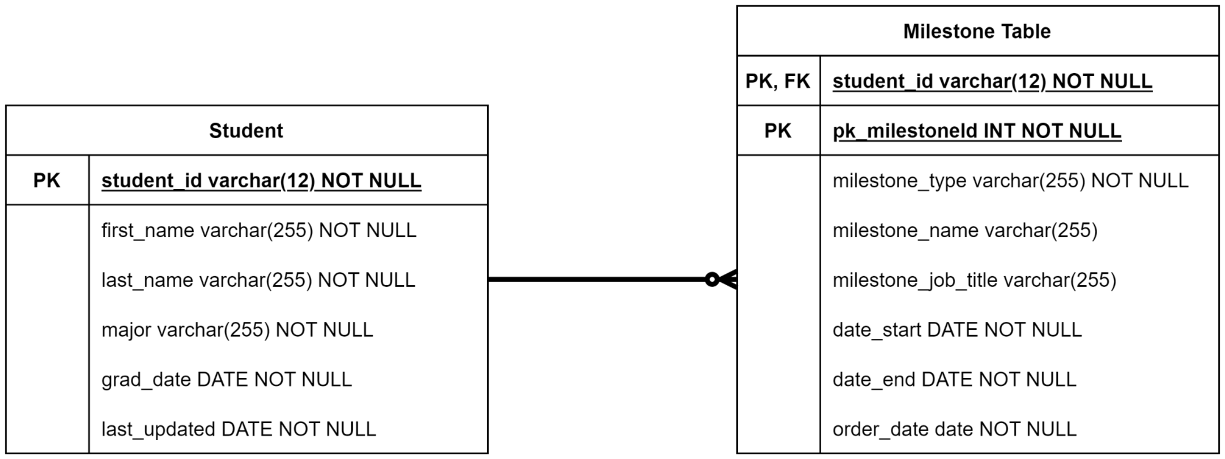
As seen in Figure 15, the CSV module is accessed when a CSV document is uploaded in the front-end interface. The use of this component then triggers the back-end system, which is written in Express Javascript, to analyze the uploaded data. Three main classes are utilized in this section, selectedStudents, studentDataArray, and updatedStudentArray. selectedStudents is an object that is a list of students that are accounted for in the back-end. studentDataArray is an array created as a result of the CSV file being uploaded. updatedStudentArray is a modified studentDataArray that, once prepared, gets sent to the MySQL server for storage.

4.3 MySql Data Management

The team created a milestone schema as one of the methods of MySql data management. In subsection 4.3.1, how student milestones will be designed within the MySql system is discussed in further detail.

4.3.1 Milestone Schema

A student can be added to the system through CSV using their first name, last name, major, and graduation date. For each student, they will only have one unique entry under the student table. Each student can have many different milestones associated with their student table. This can be seen in the diagram below.



Student
PK
student_id varchar(12) NOT NULL
first_name varchar(255) NOT NULL
last_name varchar(255) NOT NULL
major varchar(255) NOT NULL
grad_date DATE NOT NULL
last_updated DATE NOT NULL
Milestone Table
PK, FK
student_id varchar(12) NOT NULL
PK
pk_milestoneId INT NOT NULL
milestone_type varchar(255) NOT NULL
milestone_name varchar(255)
milestone_job_title varchar(255)
date_start DATE NOT NULL
date_end DATE NOT NULL
order_date date NOT NULL

Figure 16: Milestone Table Format

After the student is uploaded to the system, milestone entries can begin to be linked to the student. This process is initiated from the front-end system, where the user can click a button to reach out to LinkedIn and the process will use the student's LinkedIn profile to create milestones for the student. Milestones could be full-time jobs, internships, schooling, etc, and each separate milestone creates a new entry directly linked with the student's ID.

5.0 Implementation Plan

Due to the architecture and features, the timeline for this project relies on scheduled and coordinated implementation from each of the members of Team CareerNet. The schedule is separated into four overarching modules, each with specific subsections detailing what needs to be completed. Because this is a time-sensitive project, multiple modules will be worked on concurrently by different team members. All fundamental pieces of this application will be completed by March 14th, 2022. This is to allow time for testing and refinement before the final product is released.

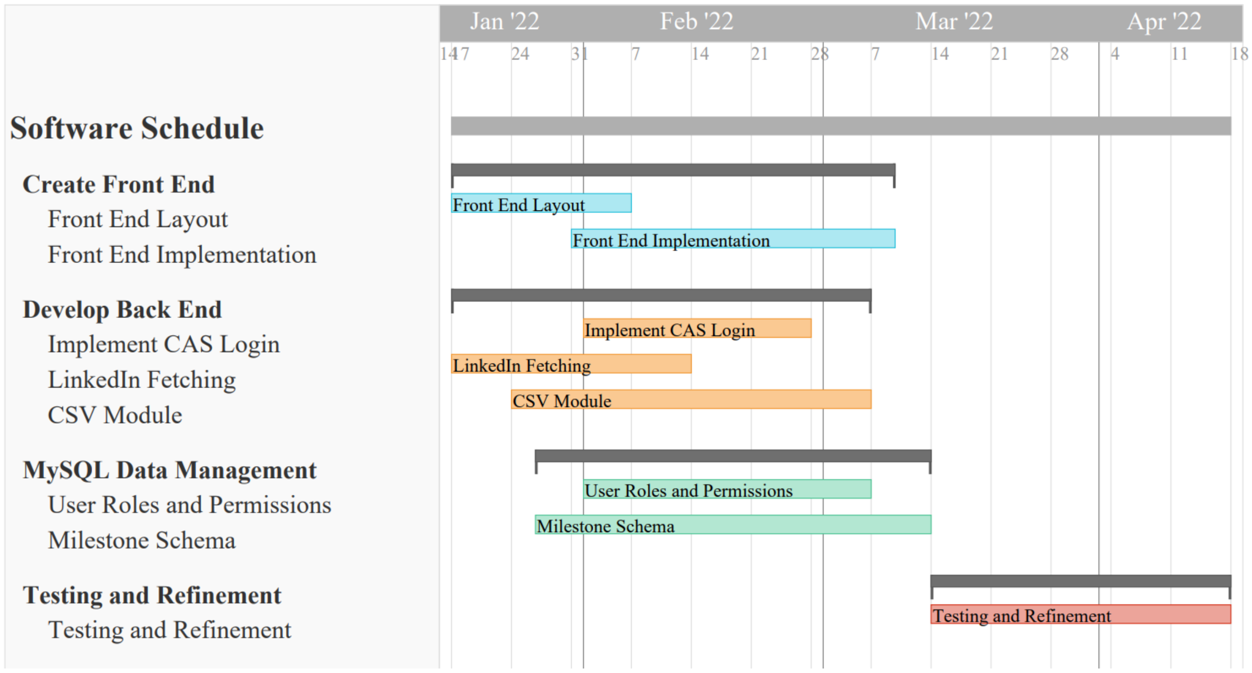


Figure 17: CareerNet Software Development Gantt Chart

5.1 Create Front End

The creation of the front-end for this application can be split into two parts: layout planning and the actual front-end implementation.

Layout planning structures each page of the application creating a “blueprint” of how a page will look, as well as what elements will be included and available to the user. Layout planning will be carried out by Carmen Montalvo and Ran Li, beginning January 17th and ending on February 7th. Front-end implementation implements the pre-planned layouts programmatically, allowing the user to utilize the functionality of the application. This will be performed jointly by Carter Taylor and McKenzie Clark, beginning January 31st and ending on March 11th.

5.2 Develop Back End

The development of the back end of this application will allow for the basic functions of the application to work correctly, instances of this include implementing a CSV module and LinkedIn fetching. LinkedIn data fetching is the first section to be worked on in this module and allows users to retrieve scraped student data from LinkedIn’s website. This section will be carried out by Carter Taylor, beginning January 17th and ending February 14th. Creating a working CSV module allows for the user to upload and organize CSV information to the application’s database. This will be executed by McKenzie Clark, beginning January 24th and ending March 7th.

5.3 MySQL Data Management

The implementation of the application’s MySQL data management system can be summarized in two sections, milestone schemas, and user roles and permissions. The purpose of milestone schemas is to lay out how each student’s milestones will be stored in the database. This will be executed by Carter Taylor beginning January 28th and completed by

March 14th. In the final data management section, user roles and permissions will separate admin profiles from other profiles, allowing administrators to give or take away certain privileges of other users. This section will be delegated by all members of the team, starting on January 31st and ending on March 7th.

5.4 Testing and Refinement

Application testing and refinement will occur after March 14th up to the project's deadline. This module will be executed by all members of Team CareerNet to ensure a well-tuned final product. Testing includes creating a document that outlines a plan with three major testing points, integration, unit, and usability.

6.0 Conclusion

College deans, like Dr. Andy Wang, aim to create an environment in which students are able to achieve their career goals, not only while they are earning their undergraduate degree but also after graduation. They, along with those in the career development director position, such as Mr.José R. Díaz Aquino, do this by encouraging students to get some form of external learning outside of the classroom, whether that be a job, internship, or research within their field of study. However, observing the students' experiences as a whole can prove to be difficult as the data pool is limited and not easily accessible.

CareerNet is a web application that scrapes student information from popular career-building websites and surveys and organizes the relevant data to be easily accessible by college deans so that they may understand the career growth of their students. This web application will implement a front-end using ReactJS, a back-end using ExpressJS, and a database using MySQL. The front-end allows the user to make requests like uploading, searching, and visualizing data as well as granting privileges to other users which is a private component for administrators. LinkedIn fetching and CSV uploading will be executed in the back-end of this app, while the database will store and organize all this information. To carry out this project, the team mapped out the most fitting schedule to design, implement, test, and refine the program.

As of right now, the team has been able to implement a user-friendly interface showcasing the requested functions from the clients Dr. Wang and Mr. Díaz Aquino. Currently, the development of the back-end as far as the design and implementation is being worked on. Once this is done, the team will work on managing the data received from LinkedIn and CSV files from Handshake and the clients.

All of these components will come together to create an efficient yet appealing web application for users, like deans, to utilize in order to gain information to improve colleges and help students achieve the milestones needed to obtain career success.