Pathways Mobile

Building a Mobile App for Students and Universities from the Ground Up

SCALE Learning Technologies

Mentors:

Tiffany Reiss, Osman Bakari, Jack Wharton, Jimmy Zheng, Christopher Nathman, Matthew Johnson, Ernest Spicer

Team LD:

Ganeshram Krishnamoorthy, Caleb Lee, Yihui Fang, Wenda Liu

TABLE OF CONTENTS

I. Introduction

This document serves as a culmination of Team LD's work on the SCALE *Pathways* mobile app. The purpose of this project is to provide students with a way to create and manage their academic schedules throughout their college journey. This document will provide extensive details on the project overview, team member bios, project requirements, and specifications. Finally, this document will provide details on the status of the *Pathways* app prototype. The team will share the current progress on the prototype as well as describe what the prototype looks like.

I.1. Project Introduction

Students choosing to pursue higher education face many difficult decisions. These include choosing the institution that best fits their needs, selecting the degree(s) they wish to pursue, and picking the individual classes that they will be taking each semester or quarter. These considerations are incredible pain points for students; it has been found that 32.9% of undergraduates are not able to complete their degree programs [1]. The factors that cause these students to drop out include attending an institution that does not match the individual's preferences, switching majors after a change of heart, and having too difficult of a course load. Students are overwhelmed by the incessant number of decisions regarding universities, degrees, and courses that they will have to make during their higher-education journey. SCALE *Pathways* will be a cross-platform mobile application that aims to address these issues. It will streamline the process of selecting a degree from an institution, tracking one's progress through time for their degree, and making adjustments to any hurdles faced. With these solutions, *Pathways* hopes to take a load off the shoulders of students around the country.

I.2. Background and Related Work

Team LD's primary contribution to *Pathways* will be implementing the mobile version of the existing robust course planning website prototype. It is built with the motivation that choosing the correct path in higher education is a critical issue. In recent times, this issue has been further exacerbated by the pandemic in which students have experienced significant reductions in learning efficacy. This has particularly harmed students who do not have access to adequate financial and social resources. An example of these unfortunate outcomes is the situation at QC, an urban college with a socially vulnerable student population located only three miles from the epicenter of New York City's COVID-19 outbreak in 2020. The pandemic reduced the freshman retention rate by 26% and altered the graduation plans of 30% of all QC students [3].

To make it easier for students to navigate through issues such as these, there are currently numerous apps available that track one's academic schedules. One of the most popular apps in this category is Class Timetable [4]. With its minimalist design, this app allows students to enter the courses they are taking and their accompanying tasks. It can also display its information in a week view. Another similar app is Schooly [5]. It features a clean user interface with Notes, To-Do's, and different themes. Like Class Timetable, Schooly lets the user manually enter their courses and their tasks. The last app we will discuss in this space is Class Schedule Planner & Tasks [6]. This app distinguishes itself in this space with extensive voice assistant shortcuts, schedule sharing, statistics, and many schedule viewing options.

The current solutions on the market lack the ability for learning institutions, such as universities, to enhance the schedule-tracking process for students. The primary purpose of each of the above-mentioned apps is to help students track their academic schedules, and yet, they all do so in a manner that does not allow the entity creating the schedules to participate. Learning institutions like universities would be incentivized to establish this communication link

as it would help attract students to their degree programs. This hole in the market is what SCALE *Pathways* Mobile hopes to fill, offering the unique advantages of being able to select standardized classes, academic schedules that are reactive to changes from both the learning institution and a student, and the ability to compare and contrast different schedules, and more.

I.3. Project Overview

With multiple fields of study in a university, it is intimidating for students to plan an optimal course schedule throughout their years in college. SCALE *Pathways* Mobile is an application that assists current and incoming students with their scheduling based on their major. On a basic level, students will be able to input their university, major, and courses. With this information, the application will automatically generate the optimal schedule for the student throughout the university. SCALE *Pathways* Mobile offers unique advantages in comparison to its current market competitors, such as being able to select standardized classes, academic schedules that are reactive to changes from both the learning institution and a student, and the ability to compare and contrast different schedules.

This schedule created by SCALE *Pathways* Mobile will be modular. Additionally, it can be revised or regenerated based on class availability, changes to graduation requirements, and if a student needs to retake a class. The schedule will be divided into separate terms to clearly show the student which classes to sign up for. Classes will also be categorized and highlighted (i.e., UCORE, Major Class, CAPSTONE, etc.). Classes can be added in bulk via CSV (.csv) files. The schedule can be saved or reverted based on each user. Previously saved schedules can also be accessed, as well as making multiple schedules for each student.

University admins (advisors) will be able to customize different courses as they see fit. Adding details such as course codes, credit hours, descriptions, and availability. Additionally, admins will be able to generate their own pathways to graduation based on specific majors.

This project is based upon an existing web application “SCALE *Pathways*”, however, the mobile version will be built from the ground up. Using the React Mobile framework, the application will be built to support both iOS and Android devices. In previous iterations of the web application, there was an implementation of graphs to build the schedule. We will be iterating on that code and improving the functionality to fit all major requirements. A validation algorithm was also used to check and maintain course dependencies to ensure a correct schedule. We will be iterating on top of the previous version to improve handling and a possible refactor.

SCALE *Pathways* Mobile must be built to be iterated upon, different classes and majors are always being added and removed. Major requirements can change from year to year, so it is essential that the code uses precise software engineering and object-oriented principles to maintain scalability.

An optional objective that Team LD would like to include is automated functionality with university systems. Currently, universities have to manually provide their course information. It would be easier to integrate more universities into SCALE *Pathways* if we implemented a wrapper or other software solution to automate the process by which their course and university information is uploaded onto the app. This objective will be difficult to scale as different universities use different legacy APIs with their scheduling software. However, implementing this feature with WSU would be our primary objective for this year.

Another optional objective that LD would like to implement is “the ability to construct schedules that targets a specialization within a major, such as the AI/ML path for WSU’s

Computer Science B.S. Degree. This could be implemented by using tags, in which the app would create a specific course schedule for the student prioritizing courses with the corresponding tags if they wish to pursue the ML specification.

I.4. Client and Stakeholder Identification and Preferences

Our team has identified two clients, which are students and universities. Students are clients since our app aims to help students to build their schedules using the major that they have picked. If students are using the app, Team LD can gather data with their consent and improve the app. Universities can be a client if they choose to cooperate with the team. This is because the app would require full access to a university's API in order for the app to aid the students in creating a schedule for them. The stakeholders are Tiffany, Osman, Jack, Jimmy, Christopher, Matthew, and Ernest. This project requires the team to build an app from the ground up to aid students to build a course schedule based on their major, and the university they are attending. The stakeholder already has a website up and running with the features. The team is tasked with transferring the features to IOS and Android.

II. Team Members - Bios and Project Roles

Wenda Liu is a computer science major with an interest in machine learning and web applications. Wenda has prior experience in building front and back end for web applications. Wenda has a wide range of languages he can use like C/C++, C#, Python, and HTML. Wenda is responsible for team management.

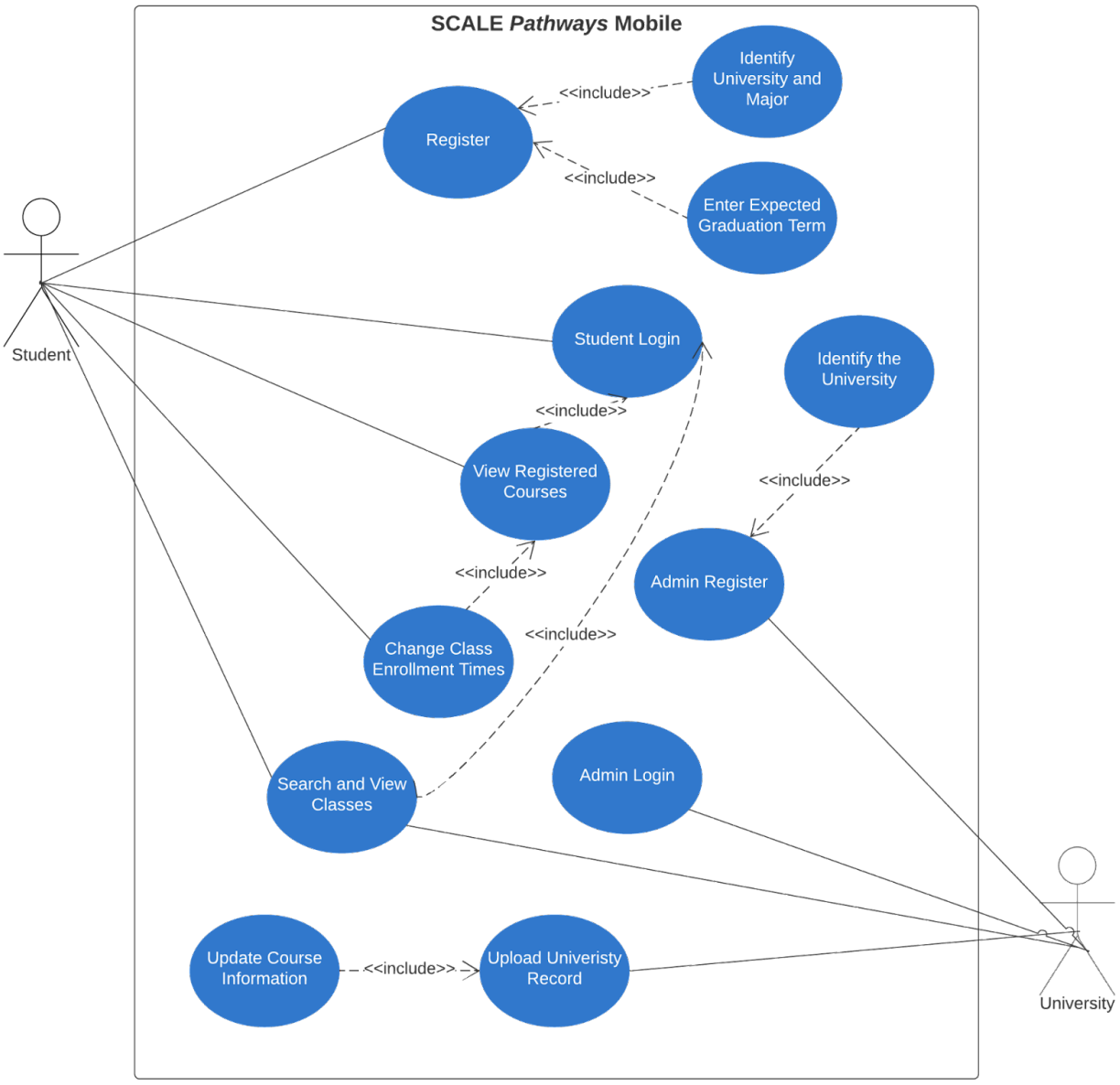
Caleb Lee is a senior computer science major that wants to go into the field of software engineering or cyber security. Caleb has experience with college-level cyber security competitions. Caleb has experience with C/C++, C#, Python, and Haskell.

Yihui Fang is a senior Computer Science student with an interest in software development. Yihui has held teaching assistant positions in Data Structure class, completed a Front-end Engineer, and participated in a deep learning-based vulnerability research program. Yihui has a good understanding of C/C++, Python, JavaScript, and Haskell.

Ganeshram Krishnamoorthy is a senior Computer Science B.S. student. Ganeshram has used JavaScript frameworks such as Ionic Framework to build native cross-platform applications for iOS and Android. Ganeshram's team utilized machine learning to rank second place in WSU's Digital Agathon 2020. Ganeshram is interested in machine learning and artificial intelligence and is proficient in C/C++, Python, and JavaScript. Ganeshram is the team leader.

III. Project Requirements

III.1. Use Cases



Name	Course Schedule
Description	First, a student needs to be able to register and identify their university. Next, they should be able to view and enroll in courses.
Actors	Student
Pre-Conditions	● The user must be logged in with a registered user account. ● The user must have identified their university

Flow of Events	1. The student selects their major from the list of offered degrees at their registered university. 2. The website populates the course table with the template schedule for that degree. 3. The student clicks on the course that interest them for more information 4. A pop-up box will be displayed containing course details, meeting location, and time. 5. The student can manually push back the course they choose to next semester. 6. As a result, classes in the following semesters that prerequisite classes would be pushed back too 7. The table's credit totals for each semester are updated to reflect the change.
Related Requirements	● The application must be able to populate the class table with a template schedule for every provided degree. ● Clicking on a class must direct the user to more information about that class. ● The user is able to manually move classes to the future semesters ● The application must be able to readjust the schedule to account for changes made by the user ● The class table must display the credit load for each semester
Post-Conditions	● The course table must be updated with the correct courses, with no relationship conflicts. For example, students can't take a course with its prerequisite class in the same semester.
Exceptions	None.

Name	Invalid Class Move
Description	Student attempts to move a class which might affect the graduation date
Actors	Student
Pre-Conditions	● The user must have been logged in with a registered user account. ● The user must have identified their university ● The user must select a degree and have a schedule loaded
Flow of Events	1. The student attempts to move the next semester's class to the previous semester 2. The application realizes that there is no way for the student to take that class 3. The application reverts the change. 4. The application displays an error message informing the student that the move is not possible.
Related	● The user is able to manually move classes between semesters.

Requirements	● The application must be able to readjust the schedule to account for changes made by the user. ● The application must be able to recognize invalid schedule changes.
Post-Conditions	● The course table is returned to the state it was in before the attempted change
Exceptions	None.

Name	Post and Update Course Enrollment
Description	A user with admin permissions for a given university uploads courses and record data for that university.
Actors	University Official
Pre-Conditions	● The user must be logged in with a registered admin account ● The user account must have admin permissions to at least one university The user must have already formatted their data into the desired CSV format.
Flow of Events	1. The user navigates to the profile page and selects the “Admin” tab. 2. The user clicks the “upload university record” button, and selects their preformatted CSV file. 3. The application shows a message verifying the success of the operation. 4. The course view page is refreshed to reflect the updated courses.
Related Requirements	● Allow the user to import course information from a standard file format: ● Provide a sample file to demonstrate the desired format ● Allow a university official to upload a CSV file of student academic records
Post-Conditions	● The new data is entered into the database.
Exceptions	None.

III.2. Functional Requirements

III.2.1. User information

Gathering User Information: the applications would require each student to use university and major information in order for the application to create and manage the student schedules.

Source: Primary stakeholders with SCALE originated this requirement. This requirement is necessary for the application to work.

Priority: priority level 0: essential and required

III.2.2. User Authentication

Authentication: the application must use university specific email to authenticate the student.

Source: Primary stakeholders with SCALE originated this requirement. This requirement is necessary for users to access their own applications.

Priority: Priority level 0:essential and required

III.2.3. Data Records

Storing the user data: The application will store the student data in a database and the user must be able to view and change them.

Source: Primary stakeholders with SCALE originated this requirement. This requirement is necessary for the application to view data.

Priority: Priority level 0:essential and required

III.3. Non-Functional Requirements

III.3.1. Easy to Use

This tool should be intuitive for its targeted demographic of college students. This is necessary for the continued use and proliferation of the SCALE *Pathways* mobile application.

III.3.2. Reliable

The application must be free of bugs and issues that prevent the application from functioning nominally.

III.3.3. Extensible

It must be simple for universities to integrate their APIs with SCALE *Pathways* mobile as this capability is integral to the application's functionality.

III.3.4. Maintainable

SCALE *Pathways* mobile must be built with robust design principles so that it is simple to maintain. Without this requirement, users will encounter negative downstream effects such as slow-to-respond pages and loss of data, leading them to cease their app usage.

III.3.5. Easy To Iterate Upon

In order to address user feedback and implement new features, SCALE *Pathways* mobile must be built in a manner that makes it easy to iterate upon.

IV. System Evolution

Software evolution is a big factor when the team was creating the design for this project. One fundamental assumption on which our project is based is that 'Students' and 'Universities'

are different actors who can complete separate actions. Another fundamental assumption is that the Student actor's functionality is dependent upon the University actor's provided information. For example, Students may only select from Universities that have registered, and only choose classes that Universities have provided. An assumption for the project's development is using a development framework that allows a single, unified code base for the iOS and Android versions of our application. Currently, a framework called React Mobile allows developers to code in a specific language and transfer that language to work on both IOS and Android. Another aspect the team needs to account for is making the project more flexible to be able to update and add new features on top of the current ones. With this, the system needs to be able to handle new attributes and be able to display the new attributes. Furthermore, the application needs to be easy to use for everyone, not just the engineers.

I. System Introduction

This section will provide extensive details on the solution approach, Architecture Design, Subsystem Decomposition, Data Design, and User Interface Design. Finally, this document will provide information on the status of the *Pathways* app prototype. The team will share the current progress on the prototype as well as describe what the prototype looks like.

II. System Overview

SCALE *Pathways* mobile's functionality and design will be based on the previous iterations of a SCALE *Pathways* desktop application. Previous teams have provided us with a solid foundation regarding Frontend design and Backend systems. We will use a cross-platform mobile architecture, React Native [7], to achieve improved workflow and simultaneous development to both platforms throughout the year.

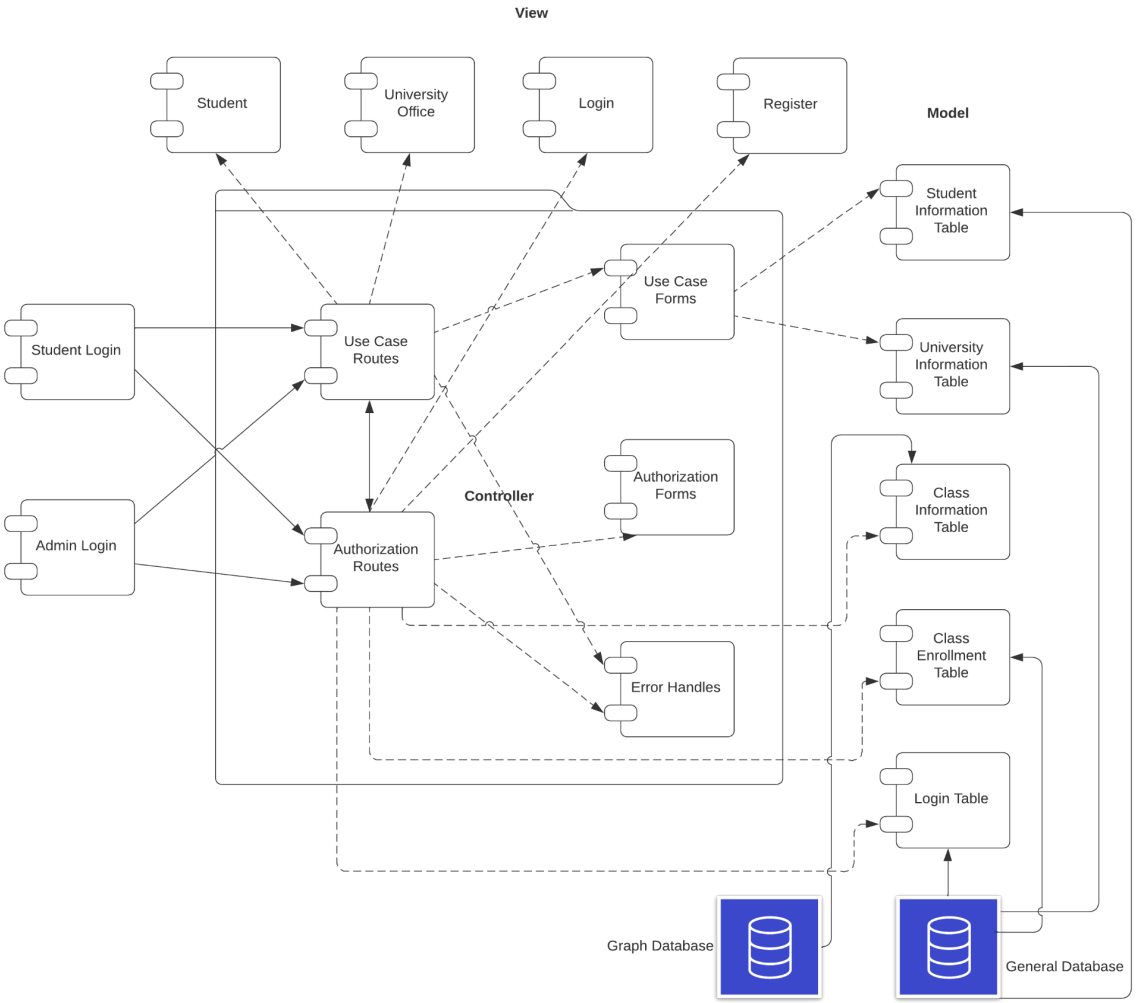
Team LD's project will primarily be focused on functionality for student users on iOS and Android platforms. Admin functionality will be a secondary goal once we have completed the student experience. Backend databases will remain relatively the same as the desktop application. Team LD's goal is to create a seamless iteration of the original *Pathways* application into a mobile format. SCALE *Pathways* Mobile will have all the features from the desktop app, including schedule generation, profile customization, graphing visualization, etc., in a mobile format. Team LD will be focused on the user experience first, making it much easier for students to get an auto-generated schedule as soon as all of their data is processed.

III. Architecture Design

III.1. Overview

Team LD has decided to use a cross-platform application architecture for this application [8]. This allows SCALE *Pathways* Mobile to reach iOS and Android markets with a single, unified codebase. This design decision has benefits and tradeoffs. As stated, the primary use of this approach only has to maintain a single version of the app rather than two distinct versions of the app for iOS and Android. The primary tradeoff is poor performance in high-performance and graphics-heavy applications, neither of which apply to SCALE *Pathways* Mobile. Thus, it is the most suitable choice for us. The mobile platform would contain a Frontend for user interaction and a Backend for requesting services. The team has included a diagram detailing how we would develop our application based on this diagram. As for this, we would provide a general idea of our diagram and go into more detail of its subsystem decomposition. The Frontend will assist with the initial decision that a user makes, which is to select whether they login or create a Student account or a University Admin account. After the user has selected an

account type the Frontend will communicate with the Backend via Use Case Routes or Authorization Routes. This component is important to determine the type of account the user has and what kind of access the user will be granted while using the application. Starting with the Use Case Routes, if the user is logged in as a Student but has no data within their account, they only have access to their student profile and a general landing page. While if the user is logged in with a University Admin account with no data, they only have access to their university office profile and University information table which is a general landing page. The Frontend would change dynamically depending on the information that the user provides. For the Authorization Routes, if the user is a student and they have data on their schedule, they would have access to a new landing page that would benefit them more. For example, when the Student joins a University, they would be greeted with a page displaying the University's degree programs rather than the general landing page they were shown previously. If this user joins a degree program, the Authorization Routes would detect this change and present them with a new landing page that shows their Class Enrollment. While on the University Admin side, if the account contains data with the user's identity and the correct permissions have been granted, the user would have access to a Class Information page and be able to perform changes. More details on these object components can be found in the Subsystem Decomposition section.



III.2. Subsystem Decomposition

I.1.1. Frontend

a) Description

The Frontend is responsible for giving the user a way to interact with our API. The Frontend is mobile-based which requires JavaScript and JSX to make each mobile page dynamic. The mobile page styling will be built upon JSX.

b) Concepts and Algorithms Generated

The Frontend will be written in JavaScript or TypeScript, which is a superset of JavaScript, but rendered with native code through React Native. In this framework, rendering logic is coupled with UI logic, and they are handled through components. This is done because this simplifies the Frontend design process [9]. React Native has wrapped many existing elements on Android and iOS into components, and any further elements that we create can be wrapped as well. No business logic of SCALE *Pathways* Mobile will be implemented on the Frontend, as the Frontend runs on each user's personal device.

c) Interface Description

The Frontend interface is running through API calls to the Backend. The API will use GET and POST to receive and display the data.

Services Provided:

1. **User Interface**

The user interface will allow the user to enter their information into the Frontend where then the Frontend will communicate with the Backend.

Services Required:

1. **Backend**

The Backend will be used to make the Frontend a dynamic application. All application data displayed to the user through the Frontend will be stored in the Backend.

I.1.2. Backend

d) Description

Since our project exists within the SCALE *Pathways* application, we will continue to use Python and the Flask web framework to develop the Backend. However, to increase productivity in the long term, we will be utilizing the SQLAlchemy [10] and py2neo [11] object relational/graph mappers. Utilizing these tools will reduce the amount of code required to translate between native Python objects and data within our databases. We will also request access to WSU's API, which would give us an extensive list of classes, help identify class prerequisites, generate class schedules, and more.

e) Concepts and Algorithms Generated

The mobile app application will only serve API requests, the database will only handle user registration information, generate student's four years course plan and validate schedule changes.

f) Interface Description

The Backend integrates with all parts of the application, including the Frontend, MySQL database and Neo4j graph database.

Services Provided:

The business rules of how the user interacts with the data stored in each database are contained in the database subsystem.

Services Required:

1. **Neo4j Database**

   Neo4j will be used to generate the graph database, it will show the dependencies for each degree plan offered by the university office.

2. **Frontend**

   The mobile-based Frontend will consider the user interface and guarantee a pleasant user experience.

IV. Data Design

The previous Pathways team, called Team Artifact, had implemented a SQL database designed to store university data and user data. They also implemented a graph database to store information about classes and the relationships they have with one another. Our team will be continuing to utilize this database design, with both the SQL database and the graph database. Our team will be including one new schema to the Artifacts SQL database. It is called the User Data schema and will contain the user's email, name, major, and bio.

Course Schema Updates:

- Instruction Team Table
  - Represents a university semester, quarter, or term
  - References a university ID
  - Identified by a unique ID
- Course Offering Table
  - Represents a specific offering of a general course
  - References an instruction term ID and a course ID
  - Identified by a unique ID

Degree Schema Updates:

- University Degree Table
  - Represents a specific degree earnable at a university (BS or BA)
  - References a university ID, major ID, and a course ID
  - Identified by a unique ID

Schedule Schema Updates:

- Schedule Table
  - Represents a schedule saved by a user
  - References a user ID
  - Identified by a unique ID
- Schedule Slot Table
  - Represents a course offering that is saved to a schedule
  - References a schedule ID and a course offering ID
  - Identified by a unique ID

Student Record Schema Updates:

- Letter Grade Table
  - Represents a grade on a specific university’s grading scale
    - Enables support multiple grading scales

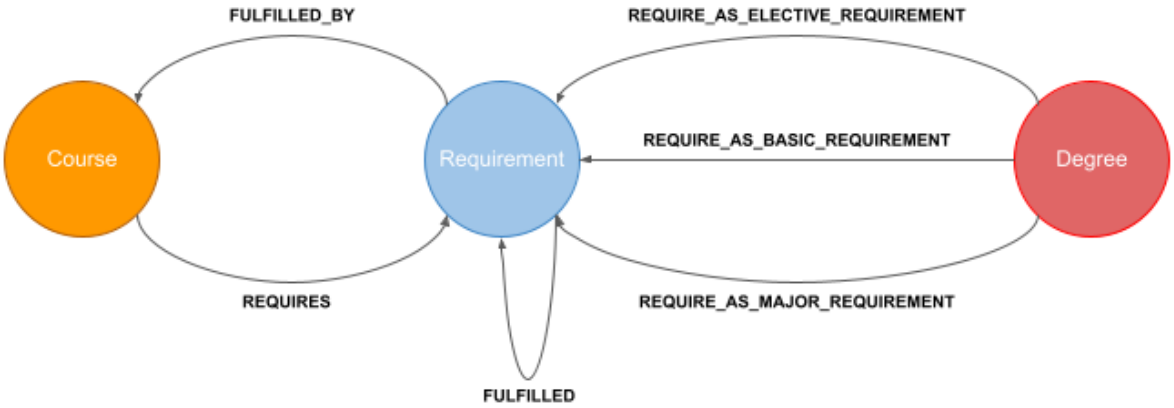
    - References a university ID
    - Identified by a unique ID
- Course Record Table
    - Represents a record of a student's grade in a past course
    - references a course offering ID, a user ID, and a letter grade ID
    - Identified by the unique course offering ID and user ID pairing

User Data Schema Updates:

- Email
    - Represents an email saved by a user
- Username
    - Represents a name saved by the user
- AccountType
    - Represents the user's type of account
- Bio
    - Represents a bio saved by the user
- Majors
    - Represents majors saved by the user

The Neo4j Graph Database

A graph database consists of nodes and directed relationships, each of which has a label to determine the "type", and attributes to store information. While Neo4j does not strictly enforce a schema, having a consistent structure for our data makes it much easier to retrieve the information we need.



This schema consists of three different types of nodes and six different types of relationships that they can have between them. With these relationships, we will be able to perform all the required logic to generate schedules, and auto-adjust schedules based on the department of courses.

Node Types

- Course

    - This will store the course ID, and be how we primarily retrieve class information
    - Attributes: like title, course number, and prefix
- Requirement
    - This represents a prerequisite requirement, degree requirement, or elective requirement relationships
    - Attributes: name
- Degree
    - This represents a specific degree offered by the university, for the purpose of associating courses as required or part of a template
    - Attributes: name, track

V. User Interface Design

Team LD has created a Figma prototype for the application that the team envisions. Figma allows the team to create a semi-functional application prototype which can be interacted with in a similar manner to a real application. In the Figma prototype, we have created a loading page when a user has clicked on the application. Next, after the loading is done, it will bring the user to a login page where they could either enter their profile information to log in, or create a new account. If the user does not have an account, they can create a Student account or a University Admin account. For account creation, a screen is displayed which allows the user to enter their information. Let us first presume the user creates a Student account. The Student account will be greeted with a landing page upon signing in. The content of the landing page depends upon the user's state. If this is the first time the user has signed in, it is likely that the user is not already enrolled in a university. Thus, the landing page will display universities in a list view, which the Student user will be able to interact with. They will be able to find more information on the university by clicking on its UI element, such as the cost of attendance and a description. Once a Student user has joined a university, the landing page will show the degree programs available in their university in a list view. By interacting with a degree's UI element, the user will be shown information such as a course outline. Once the student selects a degree program, the landing page will display the user's schedule. Each iteration of the landing page and all other pages will contain an options icon in the top left corner. When clicked, this will open a pop-up providing the user with quick, easy-to-access links to each page in the mobile application such as Schedule, Degrees, Universities, Classes, and Account Settings. Another thing the user would be able to interact with is their profile page, where the user can update their personal information like name, email, and major. Now, let us presume that the user creates a University account. Upon sign-in, the user will be shown a landing page that will allow the user to perform Degree Management actions. The screen will display a list of the degrees that the University Admin user is in charge of. From this screen, they can search for degrees or classes. When they select one of these degrees, the page will then display a list of classes that belong to this degree. When a class is selected, the user is then able to edit its details. The user can also add more classes or degrees. Since this is the first stage of the prototype there are areas the team can refine and change after some testing with actual usage from users. Below are some brief descriptions for our current screens, which are displayed in the **Appendices** section.

Figure 1

From left to right, we have displayed the loading screen, the sign-in page, and the menu. The loading screen will be shown immediately after the user enters the app while content is being processed. The sign-in page will be displayed if the user has not already logged on with an account. After the user has signed in, it will no longer display that page and instead display

their landing page, unless the user has signed out. Finally, the menu UI element will be shown when the user clicks on the three horizontal bar icons in the top left after signing in. It will allow the user to navigate to different screens.

Figure 2

These are the landing pages for account creation, account editing and university selection. Once the user completes filling out all their information they will be taken directly to the university landing page to start the process of generating their schedule. Any edits that the user needs to make to their account can be made on the second landing page and is accessed through the menu icon (indicated by the three lines on the top left of each landing page).

Figure 3

The first image displayed is what the University account Sign-up page could look like. It asks the user to enter the following fields: the user's first name, last name, university email, password, and their university. The next image displays the University Admin's account settings. Here, they will be able to update their information should they choose to. The final image displays how a University Admin user would be able to filter classes and edit them.

Figure 4

From left to right, the screens shown are the reset password page, university details expansion, and degree selection landing page. The first screen demonstrates how the user will be able to reset their password, and this page will only be accessible through the login page or in the account editing option. The next screen shows what the app will look like when a Student user selects a certain university. The details will include: average tuition/cost, biography of university, location, and other important information that each university would like to add. After the Student user selects a university, a user will be able to select a certain degree that is offered at the university. The degrees are depicted in a list view. This is what is shown in the last screen.

Figure 5

From left to right, we have the Degree Management screen for University Admin accounts, the Course Management screen for University Admin accounts, and a screen that allows Student users to inspect classes in their degree. The first screen allows University Admin users to view and select any of the degrees that they manage. The next screen displays what the University Admin user sees when they select a degree to manage. They are shown a list of classes that must be completed in that degree program, which they can select to edit. The final screen displays So whenever students enroll or switch classes, the program will use the degree management list Admin added for reference. Course management is for admin to update all the prerequisite classes. Both students and admin can view and access the courses list for each major.

Figure 6

The first screen shows the landing page a Student user will see when they have selected a university and enrolled in a degree program. The Student user can see and analyze their auto generated schedule, as well as make changes to it. When a user taps on one of the classes, the bubble will expand to show more details about the class. This is what is shown in the second screen.

V. Glossary

API: Application Programming Interface

CSV: Comma Separated Values

UCORE: University Common Requirements

iOS: iPhone Operating System

QC: Queens College

AI: Artificial Intelligence

ML: Machine Learning

BA: Bachelor of Arts

BS: Bachelor of Science

API: Application Programming Interface

JSX: JavaScript XML

XML: Extensible Markup Language

UI: User Interface

VI. References

[1] M. Hanson and F. Checked, “College dropout rate [2022]: By year + demographics,” Education Data Initiative, 21-Jul-2022. [Online]. Available: https://educationdata.org/college-dropout-rates#:~:text=College%20dropout%20rates%20indicate%20that,up%20to%2040%25%20drop%20out. [Accessed: 21-Sep-2022].

[2] M. Kuhfeld, J. Soland, K. Lewis, and E. Morton, “The pandemic has had devastating impacts on learning. what will it take to help students catch up?,” *Brookings*, 03-Mar-2022. [Online]. Available: https://www.brookings.edu/blog/brown-center-chalkboard/2022/03/03/the-pandemic-has-had-devastating-impacts-on-learning-what-will-it-take-to-help-students-catch-up/. [Accessed: 30-Sep-2022].

[3] N. Rodríguez-Planas, “Hitting where it hurts most: Covid-19 and low-income urban college students,” *Economics of education review*, 28-Jan-2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8797148/#:~:text=Furthermore%2C%20the%20pandemic%20reduced%20freshman,expected%20household%20income%20of%2064%25. [Accessed: 30-Sep-2022].

[4] C. T. LLC, “Class timetable - schedule app,” *App Store*, 30-Mar-2011. [Online]. Available: https://apps.apple.com/us/app/class-timetable-schedule-app/id425121147?platform=iphone. [Accessed: 30-Sep-2022].

[5] A. Inc., “Schooly: School Planner,” *App Store*, 25-Feb-2020. [Online]. Available: https://apps.apple.com/us/app/schooly-school-planner/id1500232555. [Accessed: 30-Sep-2022].

[6] D. Kravcov, “Class schedule planner & tasks,” *App Store*, 13-Oct-2017. [Online]. Available: https://apps.apple.com/us/app/class-schedule-planner-tasks/id1278473923?platform=iphone. [Accessed: 30-Sep-2022].

[7] “Introduction · REACT NATIVE,” React Native RSS, 06-Sep-2022. [Online]. Available: https://reactnative.dev/docs/getting-started. [Accessed: 05-Oct-2022].

[8] R. Ranjan, “The Mobile App Architecture Guide for 2022,” Insights - Web and Mobile Development Services and Solutions, 03-Oct-2022. [Online]. Available: https://www.netsolutions.com/insights/mobile-app-architecture-guide/#cross-platform-application-architecture. [Accessed: 06-Oct-2022].

[9] Introducing JSX. React. (n.d.). Retrieved October 5, 2022, from https://reactjs.org/docs/introducing-jsx.html

[10] “The Python SQL Toolkit and Object Relational Mapper,” SQLAlchemy. [Online]. Available: https://www.sqlalchemy.org/. [Accessed: 05-Oct-2022].

[11] “The py2neo handbook,” The Py2neo Handbook - py2neo 2021.1. [Online]. Available: https://py2neo.org/2021.1/. [Accessed: 07-Oct-2022].

VIII. Appendices

Figure 1

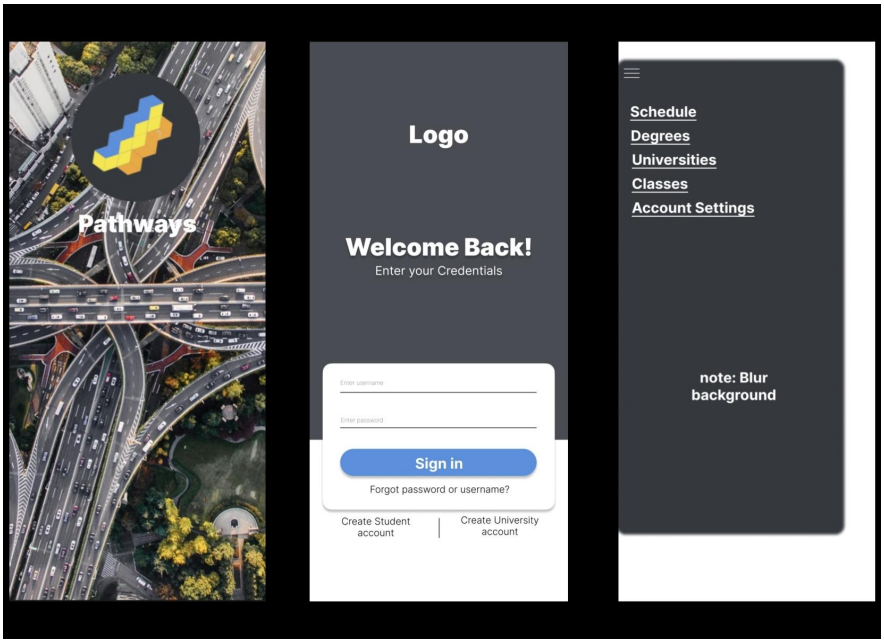


Figure 2

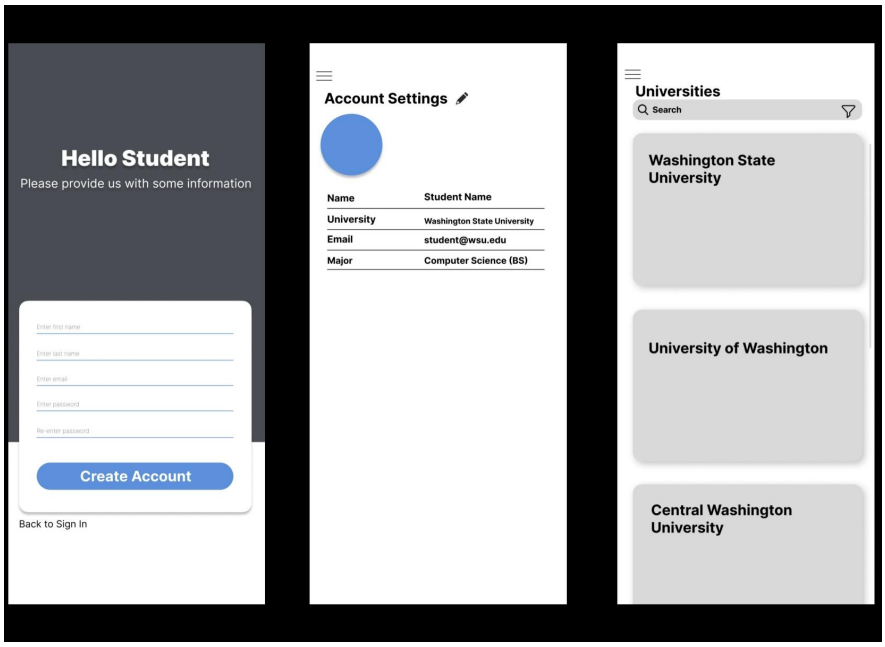


Figure 3

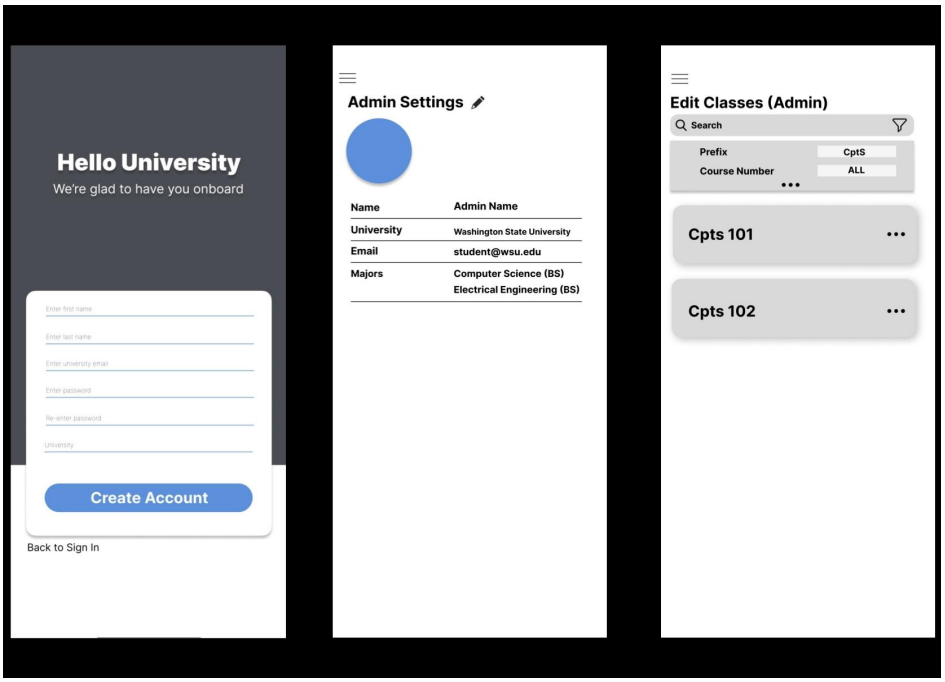


Figure 4

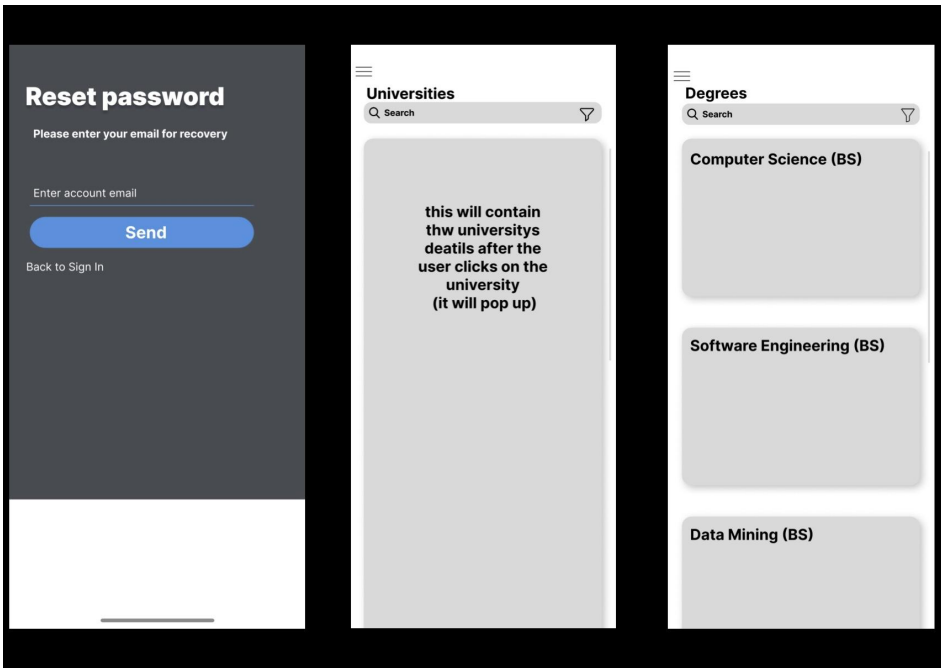


Figure 5

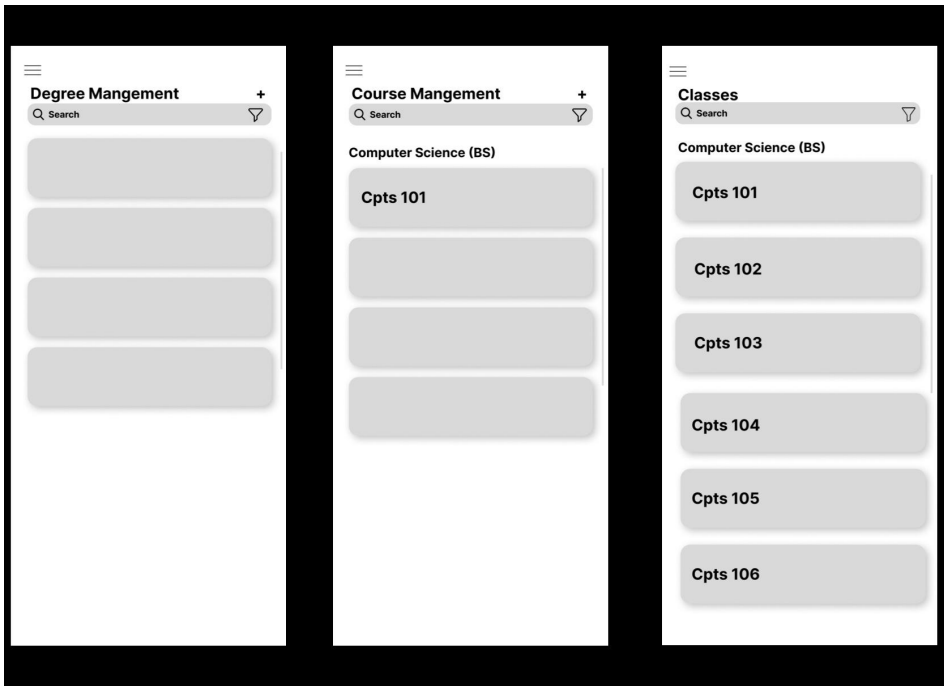


Figure 6