**Tab 1**

Progress Report 1

Course Title: COSC 4P02

Course Instructor: Naser Ezzati-Jivan

Due Date: February 23rd, 2025

**Team Name: The Mixers**

Ashu Chauhan – 7001571

Avi Patel – 6741961

Fatima Abourida - 7119490

Jerome Uwaneme -7141270

Olaoluwa Akanji - 6908776

Oreoluwa Akanji - 6910483

Russell Salacup – 7177884

# 

# Table of Contents

[**Table of Contents 2**](#_chhozyeoym6q)

[**1. Introduction (Russell) 3**](#_gp3xs6113cmy)

[**2. Design (Jerome) 3**](#_ldqbi82ree6h)

[**3. Implementation (Avi, Ashu) 10**](#_ncfanq4b8zjg)

[3.1. Tech Stack 10](#_4aq9rkejk5m1)

[3.2. Supabase 10](#_btdgigje5otx)

[3.3. Resend 15](#_kbqnbihmzewf)

[3.4. Domain 15](#_9n9f807vl7aa)

[3.5. Vercel 15](#_c4zao3dd3mm2)

[**4. Release (Fatima) 16**](#_o5zndgrz8dh7)

[4.1. Scrum meet 16](#_xzqhjbcc1hku)

[4.2 First sprint 17](#_jx4qt1j49nan)

[4.3 Second sprint 17](#_k33rhxdxe701)

[4.4 Next steps 17](#_1om7zdstdhhp)

[**5. Problems/Discussions 17**](#_feftga4ns6uc)

[**6. GitHub Log Activities 20**](#_mbfhl2kvdgmr)

[**7. Team Contribution 20**](#_n7l7pmay7som)

# 

# Introduction (Russell)

By this time, Sprint 1 – out of the 4 we have planned – has been completed, and the project’s deliverable, the Course Mix web app, contains the following elements:

* Functional landing page
* Account / login system (and relevant backend structure)
* Student home page, with quick statistics and schedule
* Templates of several feature pages for:
  + Inputting enrolled courses
  + Inputting previous grades
  + Inputting and viewing reviews
  + Adjusting profile information
* Nav-bar and footer

The team Discord server has proven to be very useful over the past few weeks; aside from acting as a meeting group call, it also allows us to organize discussions by feature, share concept visuals, and vote on design choices provided by other team members.

Jira: <https://coursemix2025.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog>

Github: <https://github.com/Avipatel1107/COSC4P02>

# Design (Jerome)

**Figma and other design attempts**

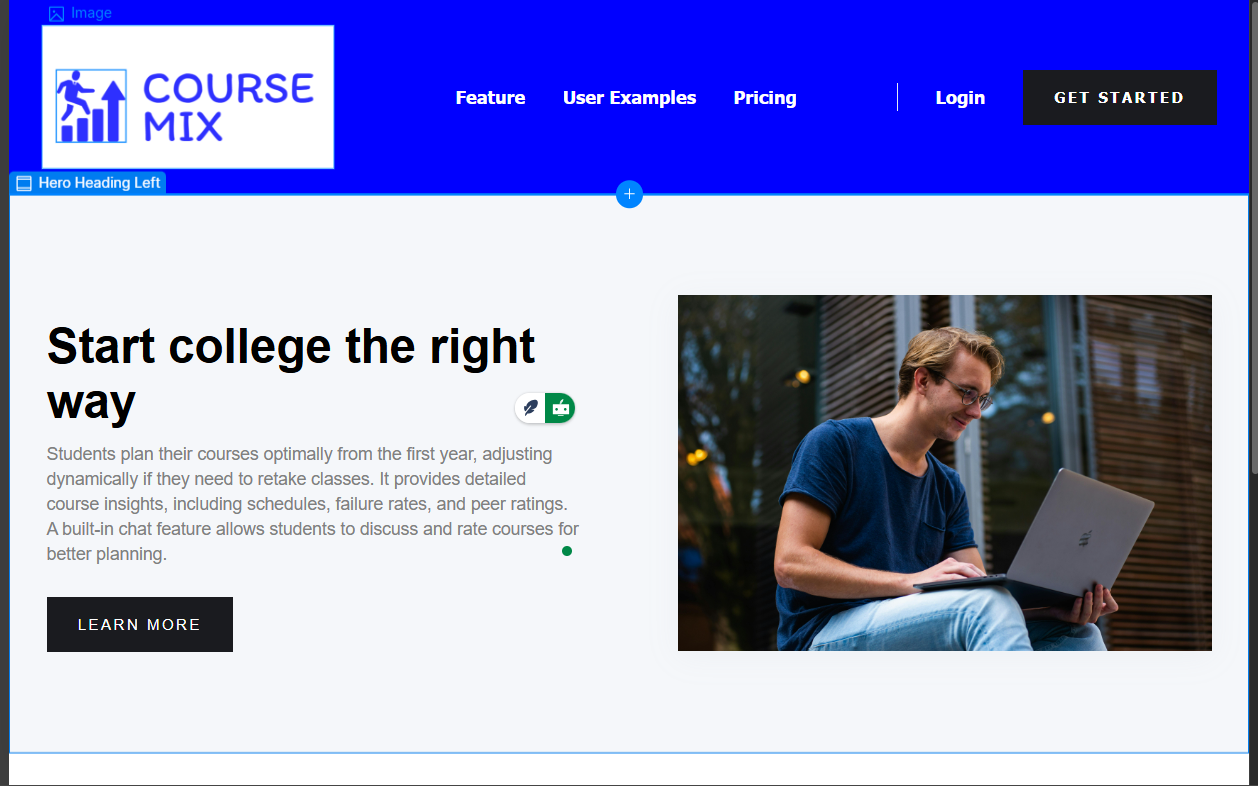
In designing the web application, we initially selected a Figma design template as the foundation for our UI/UX. We then proceeded to translate the design into code using our chosen tech stack, which is discussed in the implementation section.

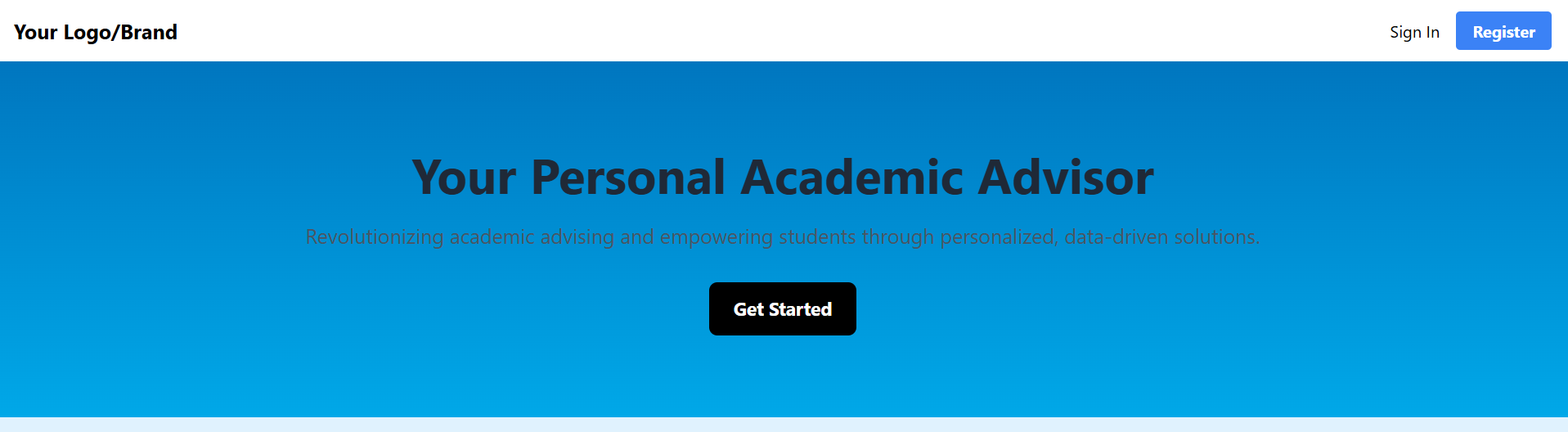
**Prototypes**

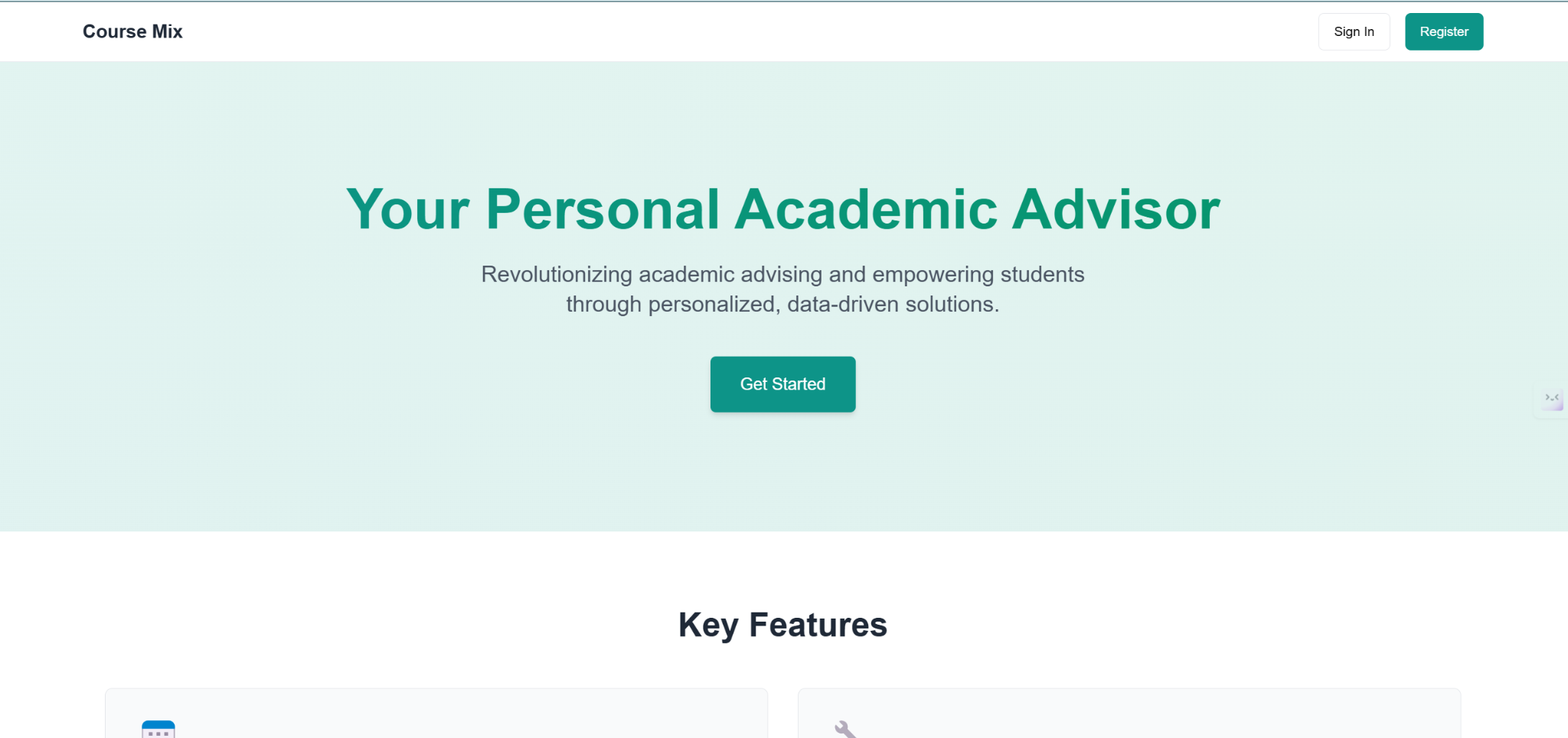
**Figma Design**



**Webflow attempt**



**First code implementation**  
  


Our goal was to balance usability and aesthetics, making the interface easy to navigate while incorporating vibrant elements to enhance engagement. We eventually settled on the design below:  


**Features and Page Implementation**

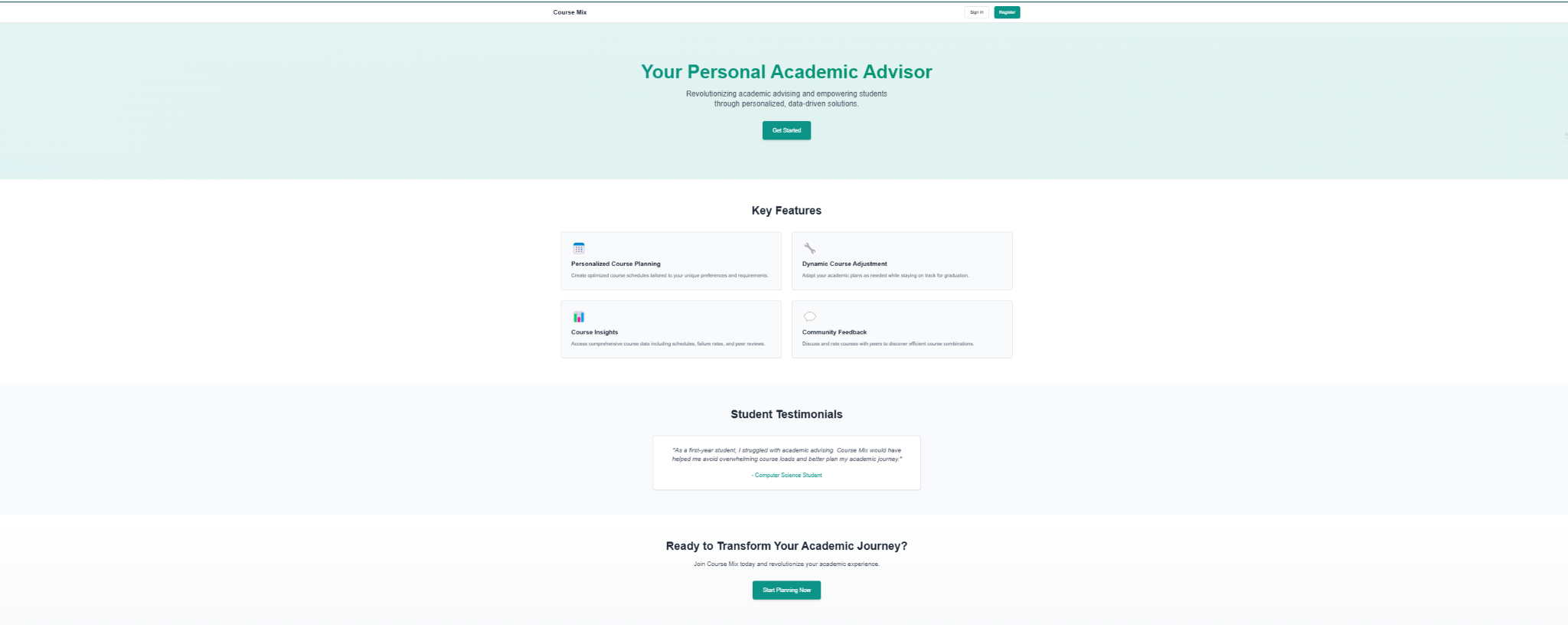
Landing Page

**Purpose**

The first page users see, providing an overview of the platform

**Functionality** Navigation menu to Sign In and Register Pages.  
 Dynamic animations or hero section for engagement.

**User Flow**

New users can explore features and register; existing users can log in.  
  


Authentication Pages (Login & Signup)

**Purpose**

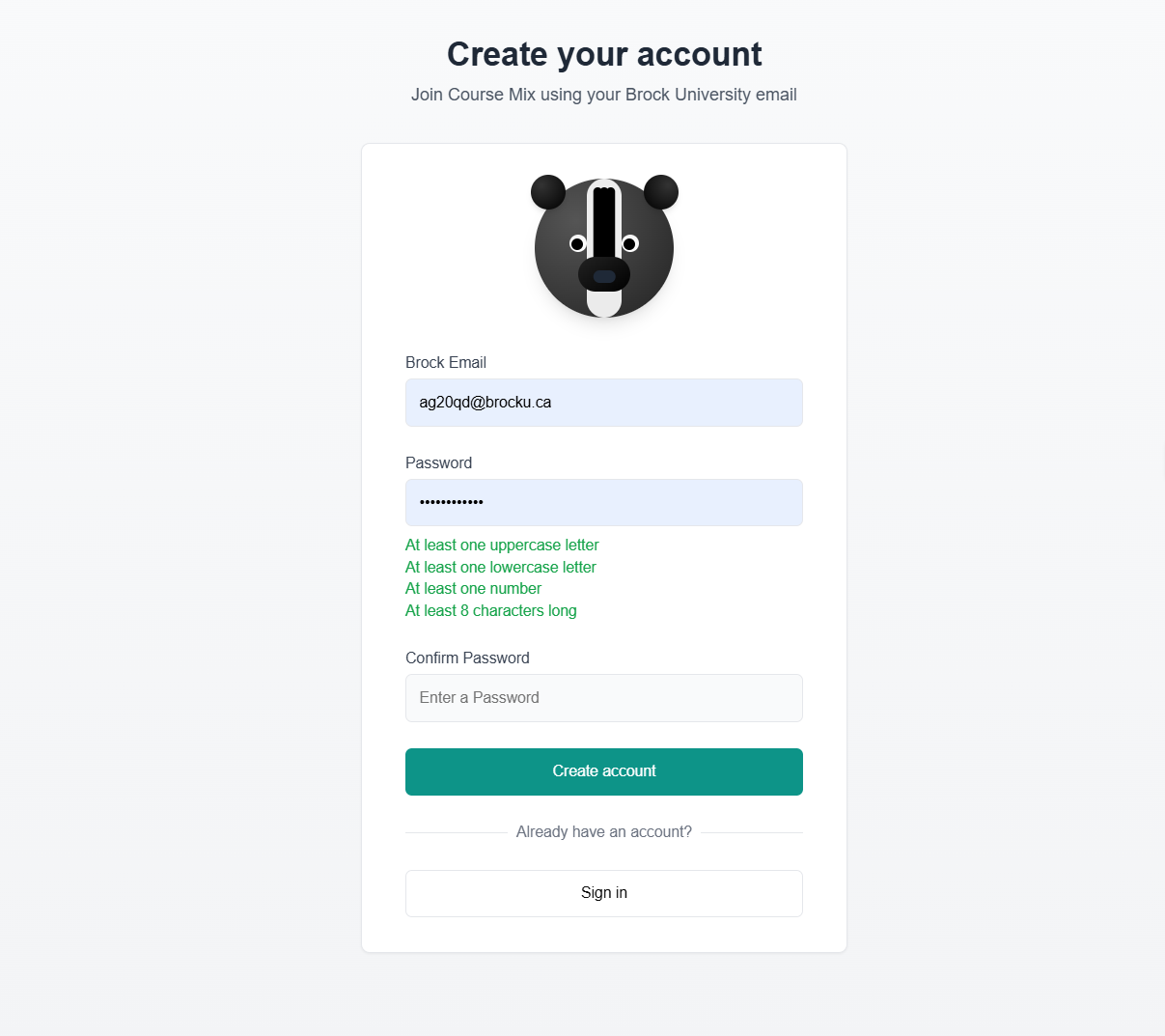
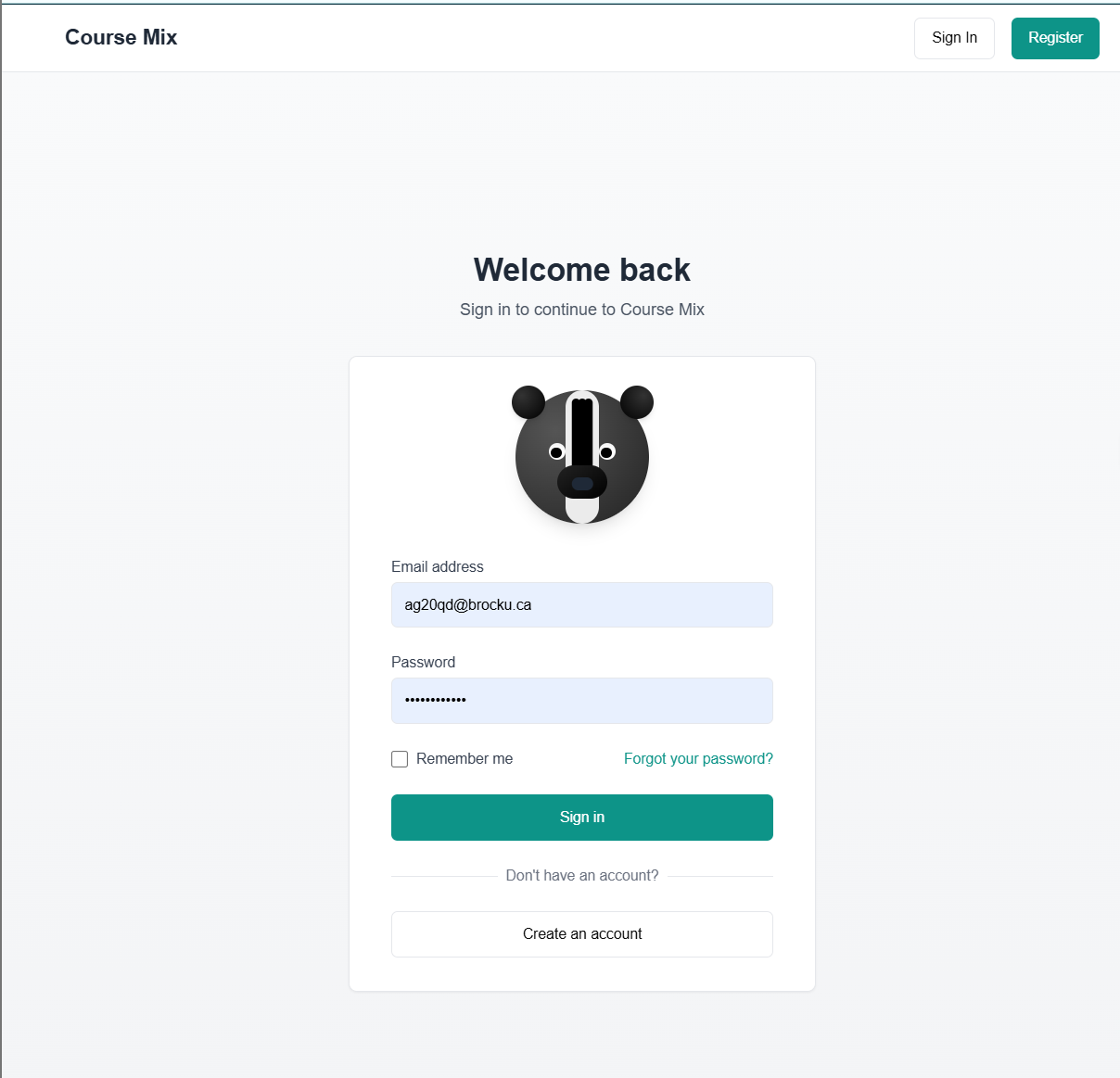
Allows users (students) to create accounts or log in securely.

**Functionality**  
 Secure Authentication (Supabase)

Password reset Feature

**User Flow**

Users enter credentials → verified by backend → redirected to dashboard.



Student Dashboard

**Purpose**

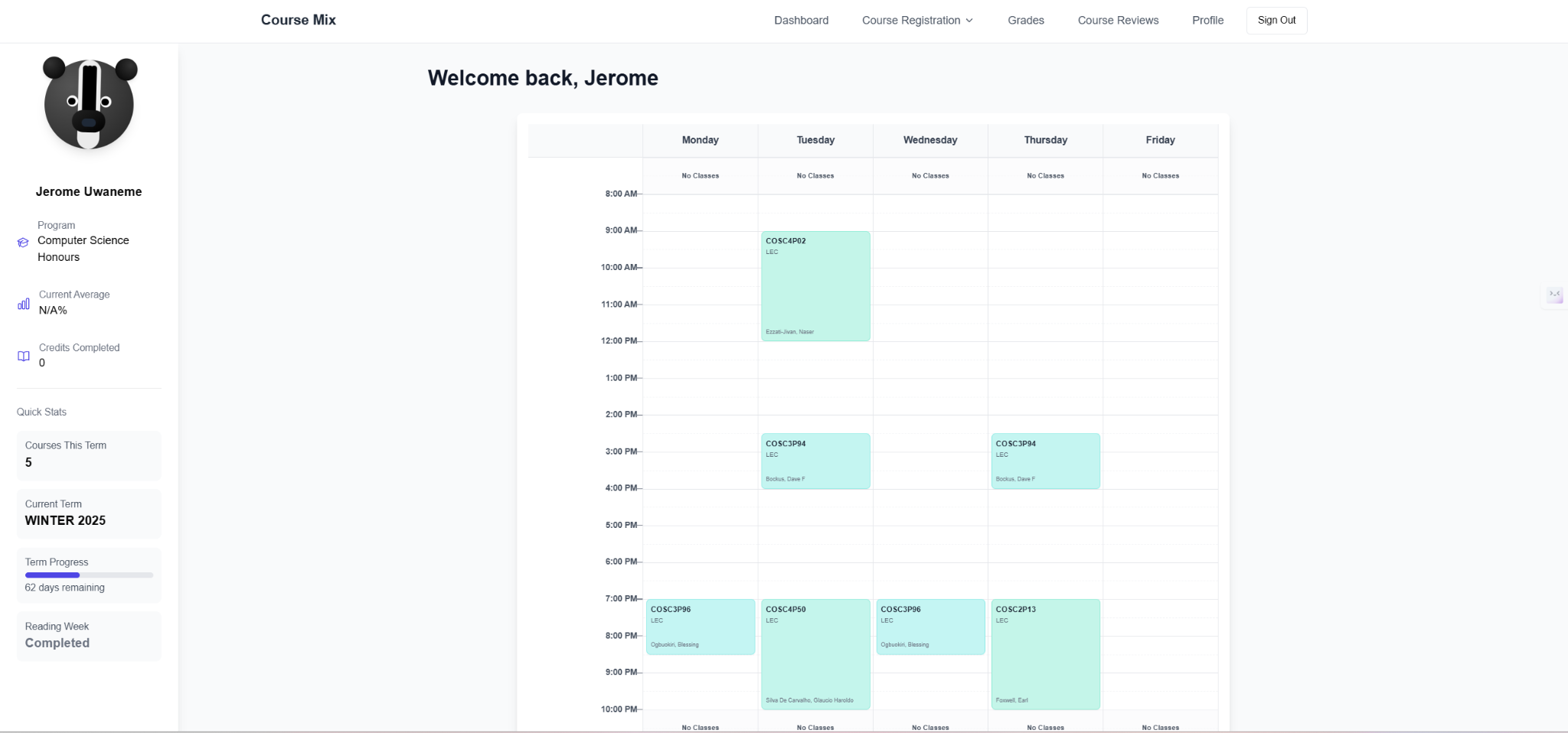
A central hub where students view their schedules and track program completion

**Functionality**

View class schedule and access other sections of the application

**User Flow**

Students log in → land on the dashboard → manage their tasks and schedule

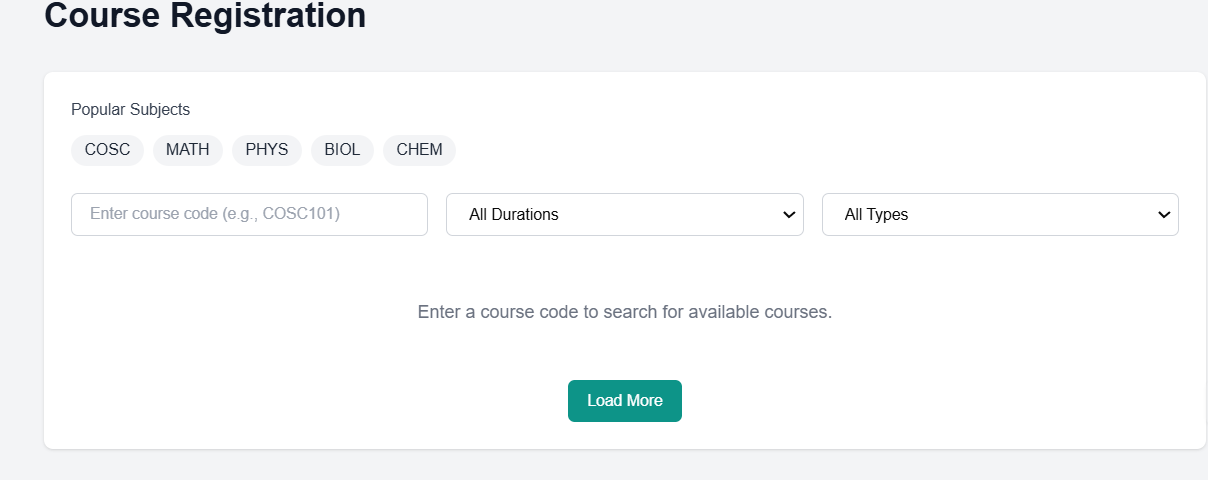
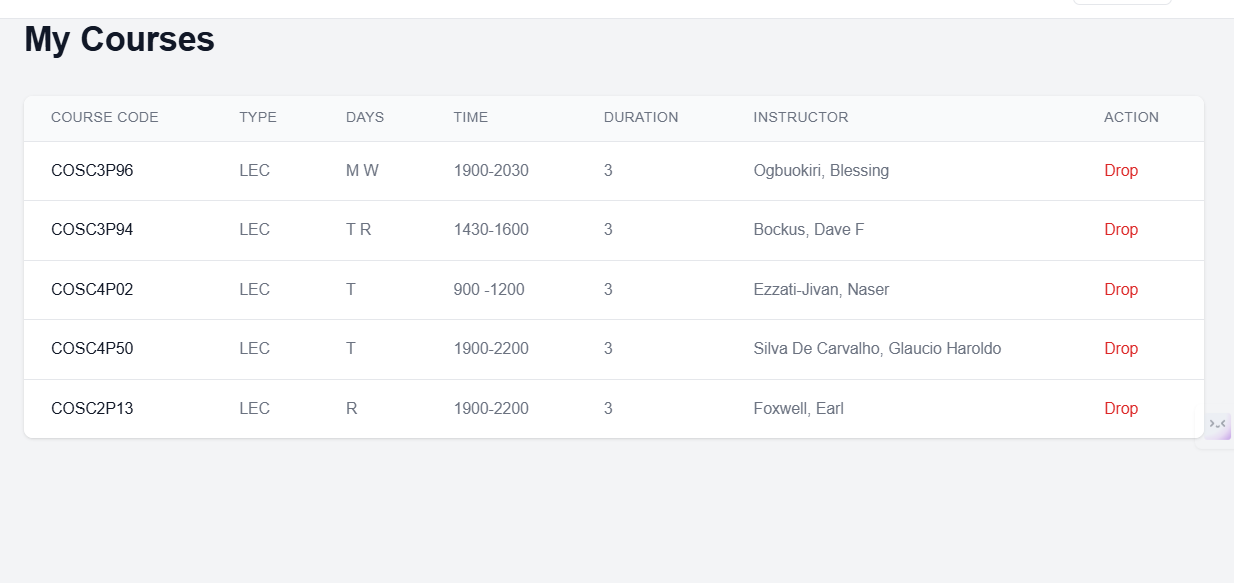


Course Registration Page (Register and My Courses)

**Purpose**Enables students to browse, select, and register for courses based on their academic requirements and availability.

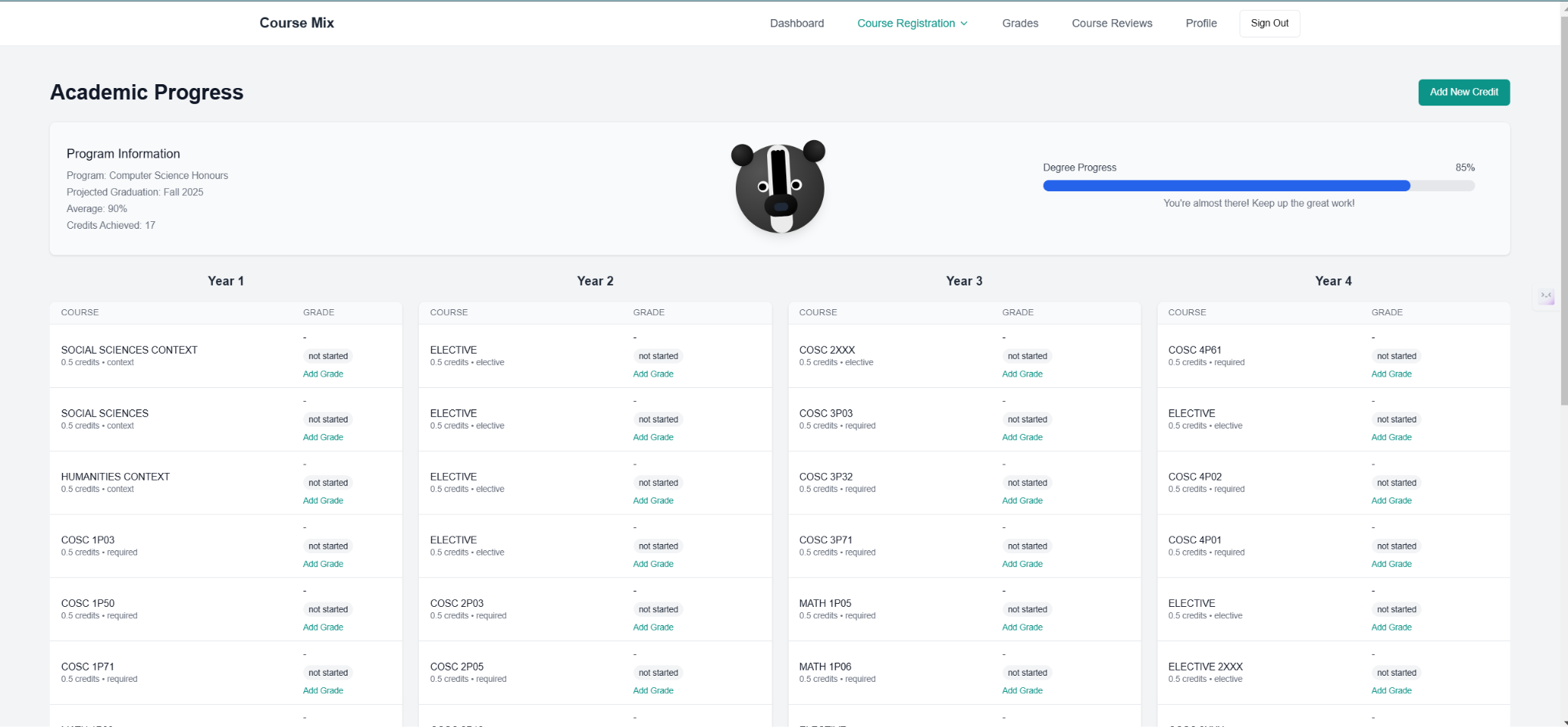
Functionality

* Displays available courses with details (course code, instructor, schedule, prerequisites).
* Search and filter options for easier course selection.
* Allows students to add or remove courses from their registration list.
* Drop courses

**User Flow**Navigate to the course registration page → Browse or search for available courses. → Student reviews and confirms selection. → Displayed on the My Courses page   
  
  
  
  
Grades Page

**Purpose**Enables students to post and review grades of courses taken giving an overview of academic performance.  
  
**Functionality**Displays grades for completed courses.

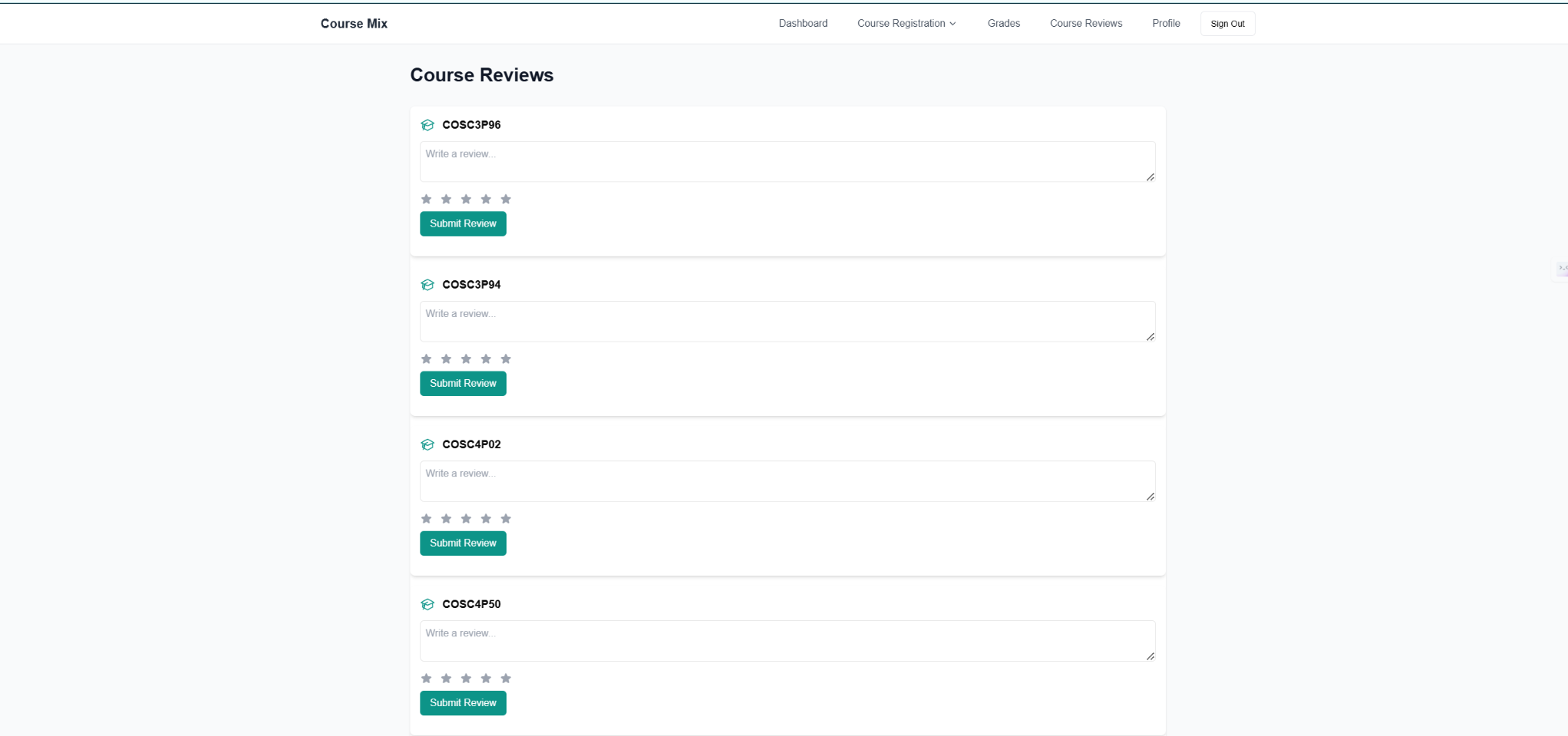
**User Flow**Student navigates to the Grades page → System retrieves and displays grades.



Course Reviews Page

**Purpose**Allows students to provide and read reviews about courses they have taken, helping others make informed decisions.

**Functionality**Students can submit reviews and rate only the courses they are enrolled in  
Displays average ratings and student feedback.  
  
**User Flow**

Student enrolls in course, towards semester’s end or learning period end, reviews are made accessible  
  
  


Profile Page  
  
**Purpose**

Allows students to view and manage their personal and academic details.

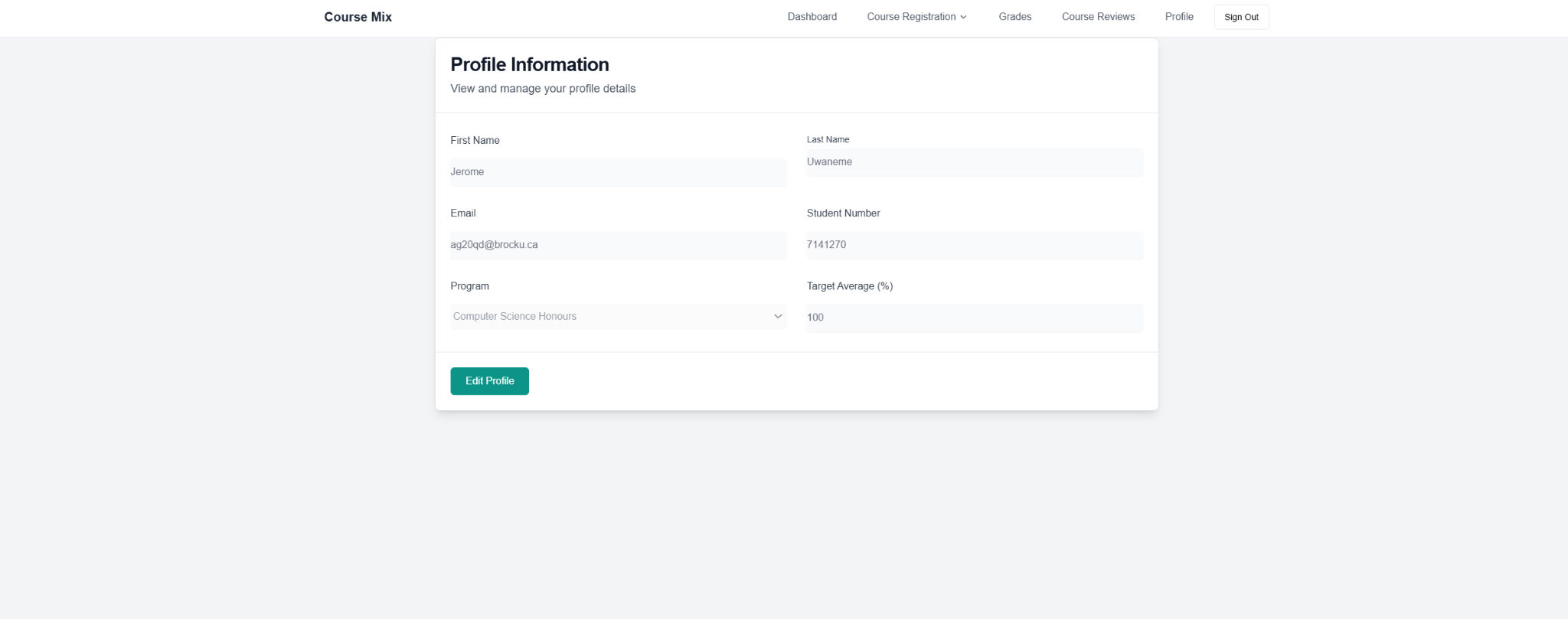
**Functionality**

Displays student information (name, email, student number, program).

Shows target average for academic goals.

Option to edit profile details.

**User Flow**

Student navigates to Profile page ➝ System displays personal and academic details ➝ Student clicks "Edit Profile" ➝ Student updates information ➝ Changes are saved and reflected in the system.  
  
  


# Implementation (Avi, Ashu)

## Tech Stack

* + 1. React, Next.js, Supabase

For this project, we decided to use React, Next.js, and Supabase as our core technology stack. React allows us to create a dynamic and responsive user interface, ensuring a smooth experience for students interacting with the platform. By using Next.js, we took advantage of server-side rendering and static site generation, which not only improves performance but also enhances search engine optimization (if required). Deploying the site using Next.js required minimal configuration, allowing us to focus more on feature development rather than backend setup.

Supabase serves as our backend solution, providing authentication, a PostgreSQL database, and real-time updates. This allows us to efficiently manage and store student data while keeping the system efficient and scalable. Since Supabase handles much of the backend functionality, we write less code, reducing maintenance and improving overall performance. Additionally, this setup ensures cross-platform support, allowing students to access the platform from any device.

* + 1. Tailwind CSS

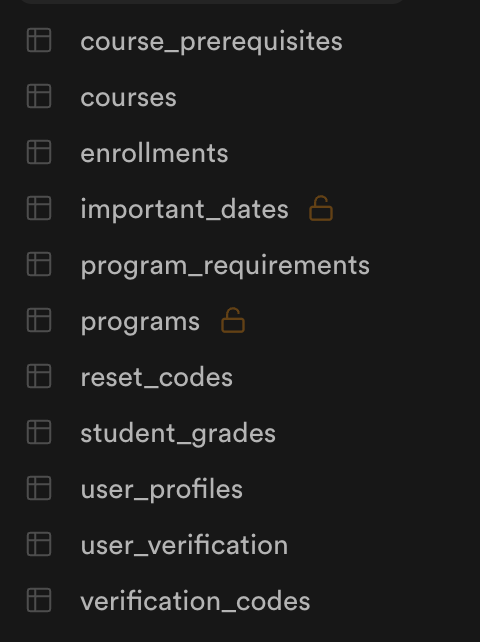
To design the interface and enhance the overall user experience, we are using Tailwind CSS for styling. Tailwind is a utility-first CSS framework that allows us to rapidly build responsive and visually appealing components without writing extensive custom CSS. It provides a consistent and modern design, making it easy to maintain and providing a uniform look across the platform.

Tailwind simplifies our development process by offering pre-built utility classes that reduce the need for writing traditional CSS rules. This makes our codebase cleaner and more manageable while ensuring optimal performance. By using Tailwind, we avoid unnecessary clutter in our stylesheets and HTML files because we mostly do inline CSS rather than having a new file for each component, leading to more efficient development.

## Supabase

* + 1. Tables

Currently, we have 11 tables in our database, which can be seen in the screenshot below.



Each table (except “important dates” and “programs”) has Row-Level Security which essentially means that the table has specific rules set for it so only specific users can access rows that they are associated with.

Our course\_prerequisites table contains a link between a course, and all of its prerequisites.

Our courses table contains every course at Brock for the current school year, this table will have to be repopulated yearly through our script. Enrollments table links a user with a course to store what courses the user is enrolled in.

Important dates are to manage when fall, winter, and spring/summer terms start and end, as well as reading weeks for each term. Having this information in a table makes it easy to update yearly as opposed to hard coding in the values.

Program Requirements is a table for managing the requirements for a degree, this table is used to populate the grades page on our website which essentially is similar to a degree progress excel sheet that is provided to students by our academic advisor.

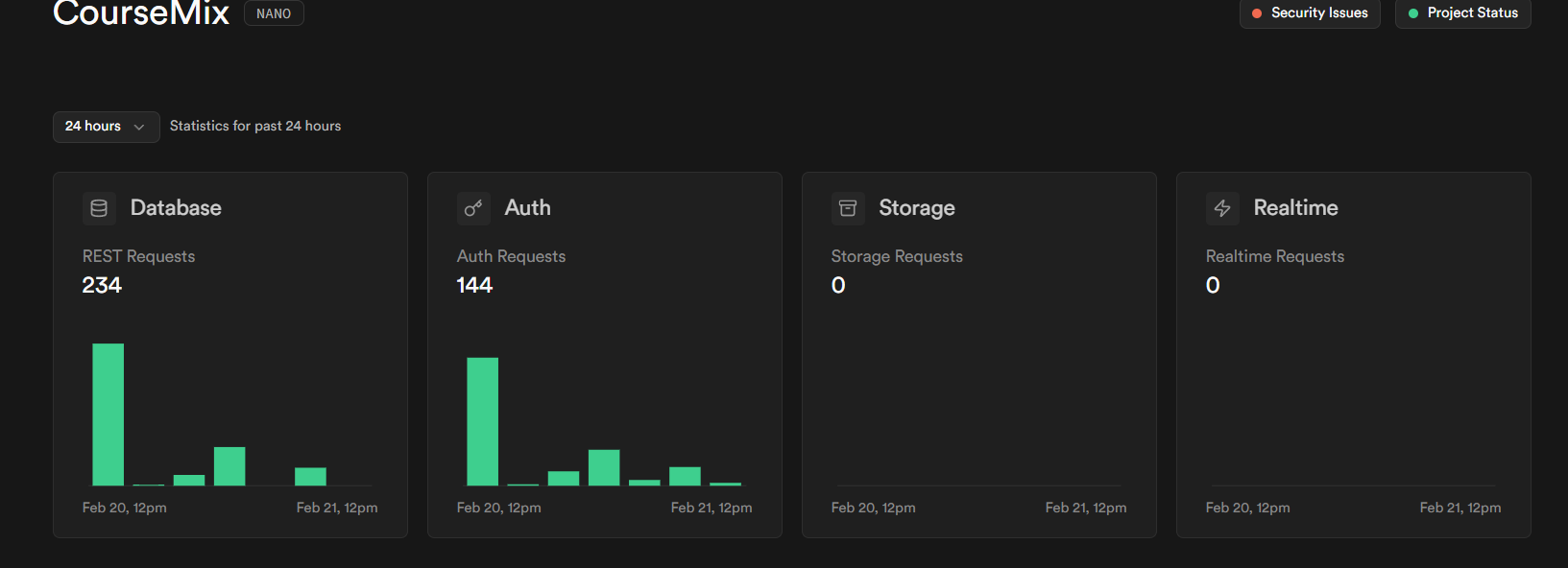
Programs holds all the programs offered at Brock.

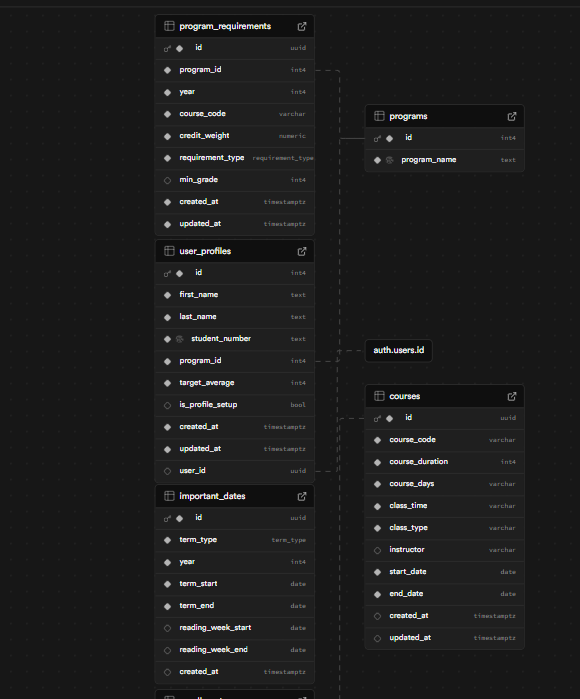
Reset codes are for password reset codes. In this table, when a user creates a request to reset their password, a code is generated in this table along with a created at timestamp, and an expiry timestamp, and a boolean value for if the code is used or not. This code is sent out to the user's email, and when the code is used, it is marked as true, and then the code is no longer usable.

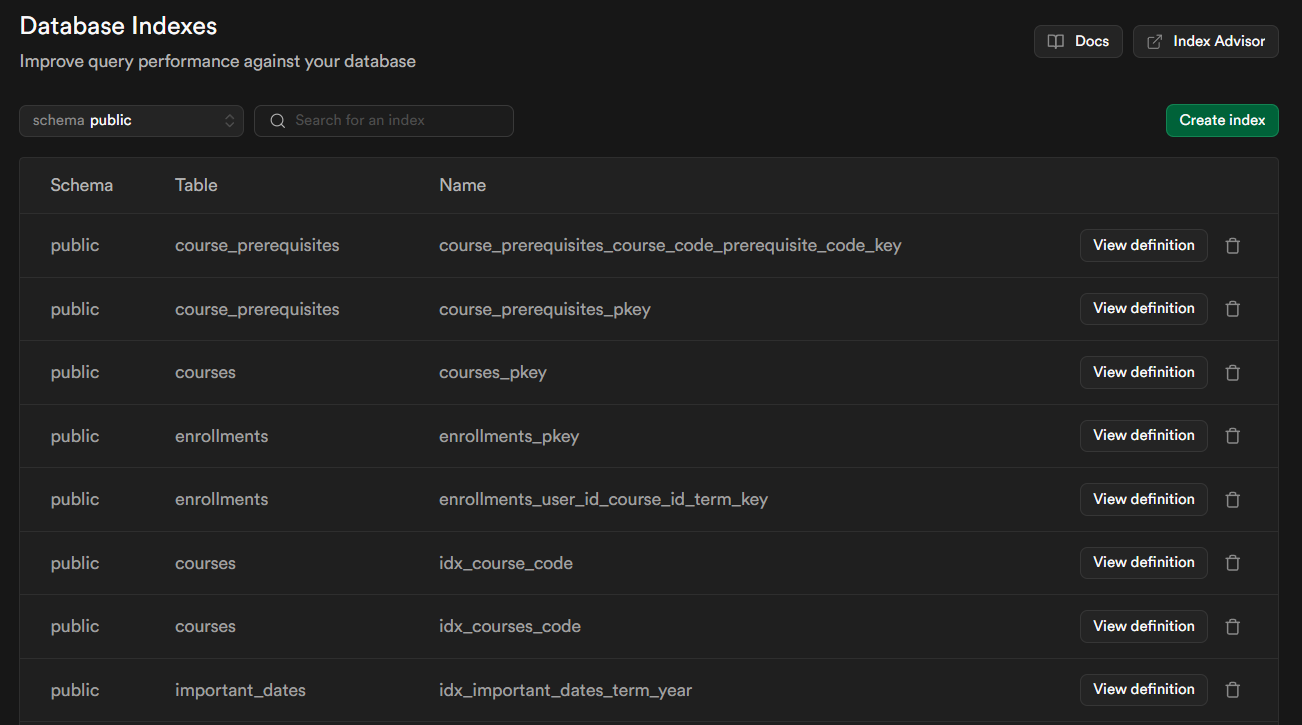
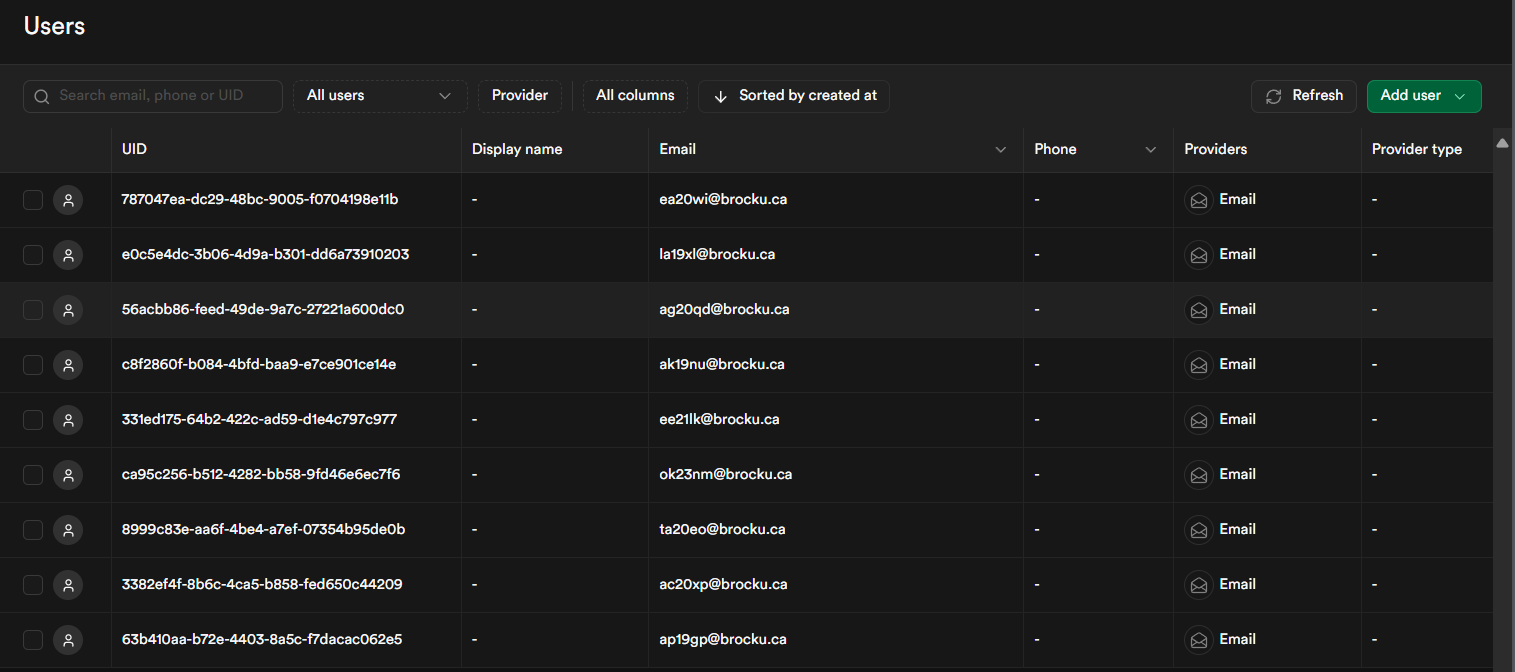
Student Grades is for handling the grades for each student in their courses.

User profiles is for managing the profile settings of users, such as their email, target gpa, name, student number, etc.

User verification is a table that simply tracks if a user's account has been verified or not; if it hasn’t, the user will not be able to sign in until the account is in fact verified. Verification codes are similar to reset codes, this is the table used to send out codes for new users to verify themselves with.  
**Supplementary screenshots**:







* + 1. Supabase Verification vs Our own workaround

In supabase, there is an auth table that holds all your users. Currently we are using Supabase free tier, and thus it is not made to handle production level requests. This became an issue when we were trying to test the built in supabase verification email system as it was very buggy. Sometimes it would not send the email until much later, and when it did send the email, the link inside of this email did not work, thus we decided to engineer our own solution.

In our verification codes table, we store the users account information such as email and password, the password is hashed before it ever gets sent into our database, so we can not see users passwords at all in here thus users can rest assured their data is secure and safe. We also store the code that gets sent out to the email, and a boolean value called “used”. In simple terms, when the verification code gets used, the value in our table is updated to true, and when that value is set to true, the account is instantly added automatically into the auth table in supabase, thus resulting in the account being created. This workaround proved to work 100% of the time, unlike the built in supabase verification, and thus we decided to stick with it.

## Resend

* + 1. OTP verification emails

We integrated the Resend’s REST API into the project to handle OTP verification emails and password reset codes efficiently. When a user registers or requests a password reset, our backend generates a secure OTP or reset code and sends it via Resend's email service. The API allows us to send emails quickly with a high delivery rate,using this service and not one made by us reduces the chances of messages going to spam. On the frontend, users receive a prompt to check their email for the code, which they then enter to verify their identity.

Additionally, Supabase integrates with Resend, which allowed us to store and manage verification and reset codes in dedicated database tables. This setup ensures that authentication attempts are properly logged and can be validated securely. By leveraging both Resend for email delivery and Supabase for data storage, we created a secure, and efficient authentication system without the need of managing/creating our own email infrastructure.

## Domain

* + 1. Own Domain GoDaddy

The project is hosted on a custom domain that we purchased through GoDaddy, giving it a professional and recognizable online presence. After securing the domain, we configured the necessary DNS settings to point to our hosting provider, ensuring reliable access to the site. The project is fully live, allowing users to register, log in, and interact with all its features in real time. Although not all functionalities are implemented, user can at atleast register, signin and enroll for courses as well as dynamically view their schedules.

## Vercel

* + 1. GitHub

The project is hosted on Vercel, which provides easy deployment and automatic updates whenever we push code to the main branch on GitHub. Vercel integrates directly with our repository, so every time we update the main branch, it automatically builds and deploys the latest changes without requiring manual deployment. This ensures that the live site is always up-to-date with the latest features and fixes.

Additionally, Vercel gives us deployment previews when we create pull requests on feature branches in GitHub. This means we can test changes in a real environment before merging them into the main branch. These previews allow us to see if the build will succeed or fail ahead of time, helping us catch errors early and maintain a stable production environment.

* + 1. GoDaddy DNS Configuration

Once we configured the DNS settings on GoDaddy, we connected our domain with Vercel by adding the domain to our Vercel project dashboard. Vercel then provided us with the necessary DNS records to update on GoDaddy, such as CNAME and A records, to route traffic to the Vercel-hosted project. This integration ensures that when users visit our custom domain, they are directed to the live site hosted on Vercel. The DNS verification process also confirms the domain’s ownership and it also allows Vercel to generate a SSL certificate to ensure the site is marked as secure(https), completing these steps for the site allows it to be accessed under our personalized domain name.

3.5.3 Security Benefit  
 Restricting access to only .brocku.ca emails adds an extra layer of security, helping to prevent DDoS attacks. Additionally, Resend (code verification feature) enhances security by requiring users to input a verification code to complete the login process. This means that simply having a Brock email isn't enough—you must also verify it to gain access.

# Release (Fatima)

## Scrum meet

* + January 8: Discussed our project and divided tasks for our project proposal
  + January 12: Finalized our project proposal
  + January 14: User Stories Brainstorm, and divided tasks for the release planning document
  + January 19: Finalized our Release planning document
  + January 21: Discuss project architecture.
  + January 24: Decided on project architecture
  + January 28: Divide tasks for sprint 1. We decided to create a landing page and a few logo designs before starting on the tasks.
  + January 31: Voted on design, and further divided sprint 1 tasks.
  + February 4: Went over tasks, and discussed any challenges and questions that we have. Went over what was pushed to GitHub.
  + February 7: Went over progress, and prepared for our meeting on February 11
  + February 11(with TA): presented our website. Brenden was not able to break it. Professor Naser strongly suggested the use of AI.
  + February 18: Divided tasks for the release planning document
  + February 21: Finalized the release planning document

## 4.2 First sprint

* + We finished the first Sprint of the Course Mix project. The user can now create their account and profile. They can select their program status, such as Honours or Coop with Honours. However, we do not have the registration page hooked up to the Coop with Honours status. The user can also save their login credentials after setting up their account.

## 4.3 Second sprint

* + We are halfway through Sprint 2, and as a result, we are ahead of our schedule. The user can easily identify course conflicts, as they are prompted with a warning message upon registering for the conflicting course. The user has the ability to manually adjust the prerequisite schedule, as they can select any course they want and any section. The timetable is automatically generated based on selected courses so the user can visualize their schedule. The user also has access to all Brock University courses, including the course code, course type, day and time it’s offered, duration, and the instructor.

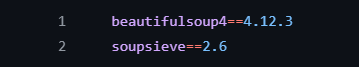
## 4.4 Next steps

* + We are hoping to complete Sprint 2 by Friday, February 28. We need to work on displaying the courses needed to pass for all the programs. We need to work on the backend for the graduation progress bar, seeing past grades, and credits earned. We have a few bugs related to cookies, we are hoping to tackle them too. We will continue with our scheduled meetings and depending on our project completion velocity, we may consider adding another sprint.

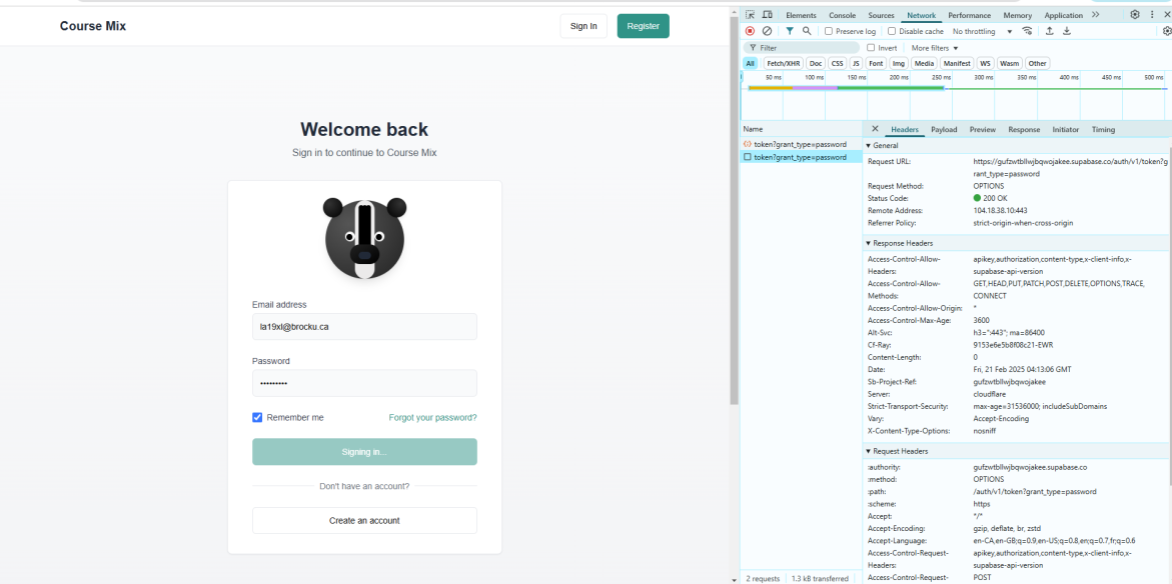
# Problems/Discussions

* We had significant concerns about web scraping. Initially, we implemented it using Beautiful Soup, but the process proved to be quite complex.

**Screenshots**:  


**Requirements.txt**:  


* **Solution**: Discovered a similar API and reached out to the creators for assistance.
* **Issue**: Accidental commits to main (Almost nuked the project)  
  **Solution**: omplemented a branch protection feature that requires two approvals and displays deployment status before merging into the main branch.
* **Issue**: Investigating and Resolving the Infinite Login Loop  
  **Current Findings**: The infinite loading issue appears to occur only when Supabase has been inactive for a while. Further investigation is ongoing.



# GitHub Log Activities



# Team Contribution

| **Team Member** | **Contribution** |
| --- | --- |
| Everyone | Testing |
| Ashu Chauhan | Implementation |
| Avi Patel | Implementation |
| Fatima Abourida | Design template and Release |
| Jerome Uwaneme | Design |
| Olaoluwa Akanji | Implementation, Problems/Discussion (web scraper piece, Research to resolve infinite login) |
| Oreoluwa Akanji | Logo/Theme Design |
| Russell Salacup | Branding Design, Proofreading Reports |

# 