**Tab 1**

Final Report

Course Title: COSC 4P02

Course Instructor: Naser Ezzati-Jivan

Due Date: April 27th, 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 (Kitan) 4**](#_gp3xs6113cmy)

[**2. Requirements (Russell) 4**](#_feftga4ns6uc)

[**3. Installation (Jerome + Ashu) 4**](#_7inu78v181vi)

[**3.1. Technical Manual 4**](#_xyu2iuw6461w)

[**3.2. User Manual 5**](#_79j6y99si18w)

[Accessing the Application 5](#_s7itrunxxhuc)

[Dashboard Overview 6](#_uau9on4maqid)

[Profile Summary 6](#_2grc5taooggh)

[Weekly Schedule 6](#_q1h5fsq2btl5)

[Quick Navigation 6](#_99t102apv1fn)

[Upcoming Deadlines 6](#_deyff76bnkxg)

[Campus Weather 6](#_pp11uqxhcmic)

[User Registration and Login 7](#_3z6obimu9qjc)

[Signing In 7](#_p8j981xb52bv)

[Creating an Account 8](#_pu80ciszdazu)

[Course Registration 8](#_rn3bgysugf7f)

[Suggested Courses 8](#_q56qtbmjhpr6)

[Search Courses 9](#_ka1cnlv9f05)

[My Courses 9](#_761uj3j9wnxv)

[Course Details 9](#_pab7ghrbzu5o)

[Managing Courses 10](#_l8yoizz9ym2f)

[Course Discussions 10](#_j892rv9136u3)

[Creating a Post 10](#_iymnb365deyu)

[Filtering Discussions 10](#_nfcgl5v72lqs)

[Course Reviews 11](#_o6wa9qlekekr)

[Submitting a Review 11](#_tcb60qjf1jbz)

[Filtering and Sorting Reviews 11](#_2ctxmrmf5evv)

[Academic Progress 11](#_kbwgl9stfueb)

[Privacy Protection 11](#_rvtbuwsswuwy)

[Degree Progress 12](#_y07roxmiiwz2)

[Projected Graduation 12](#_apm2dxi6362l)

[Program Requirements 12](#_v22n2zb6al8k)

[Your Profile 13](#_eakdawos3pck)

[Profile Overview 13](#_8rudr1pvonwe)

[Editing Your Profile 13](#_ep74r3yplvt0)

[Account Management 13](#_sve0vlsjr4f)

[Signing Out 13](#_36jp3y2iv0hu)

[**4. Testing (Ola + Avi) 14**](#_rfpweghuyy5r)

[**4.1. Tests 14**](#_vwvsy2do8vpv)

[4.1.1. Framework 14](#_659xdgwv3tom)

[4.1.2. What was Tested? 14](#_roaikibw3rir)

[4.1.3. Test Case Sample 15](#_jeh0zbutf6vw)

[4.1.4. Continuous Integration 15](#_jq3ng2n1ojs0)

[**4.2. Results 16**](#_7d7lttm7xg86)

[4.2.1. Pertinent screenshots: 16](#_7s93l97aveql)

[4.2.2. Overview of Test Results 19](#_e61blobqxdwc)

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

[**6. Team Contribution 21**](#_sfv6gl4xc3kx)

[**7. Release (Fatima) 22**](#_o5zndgrz8dh7)

[**7.1. Scrum meet 22**](#_42c4bkvgf8dx)

[**7.2. Reports 23**](#_8ku6xieh8cjv)

# 

# Introduction (Kitan)

This report outlines the collaborative development and testing of CourseMix, a web-based academic planning tool aimed at simplifying and enhancing the student experience. The application enables students to create accounts, select their program type (including co-op options), visualize their course requirements, generate automatic timetables, track graduation progress, and access valuable insights such as course reviews and failure rates to make more informed academic decisions. Our team has worked throughout the semester to design, build, and test the application using agile practices, continuous feedback, and modular development. We applied technologies such as React, Next.js, TypeScript, and Tailwind CSS, and used Jest and Testing Library for unit testing. Although we encountered challenges implementing continuous integration with GitHub Actions, we established a strong testing framework and developed a responsive, functional tool that meets its intended goals. Now at completion, the project reflects our full development cycle from ideation to delivery.

The report details each phase of this process, covering functional design and implementation (requirements, installation, and testing), quality assurance (test frameworks, results, and continuous integration), and supporting components like technical and user documentation. It also includes visual proof of development (pertinent screenshots), collaboration tracking (GitHub log activities), and reflective summaries (team contributions and release planning). Each section has been authored by team members who led their respective areas.

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

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

# Requirements (Russell)

A student is able to:

* create an account to save their academic preferences and progress
* select their program type (co-op, etc.)
* easily see and plan the courses needed to pass their registered program
* have a timetable automatically generated based on their selected courses, to visualize their schedule
* see reviews, failure rates, and other insights for each course to make informed decisions
* provide reviews and opinions regarding courses
* track how close they are to graduation (based on selected program and completed courses – a time estimate is calculated)

# Installation (Jerome + Ashu)

## Technical Manual

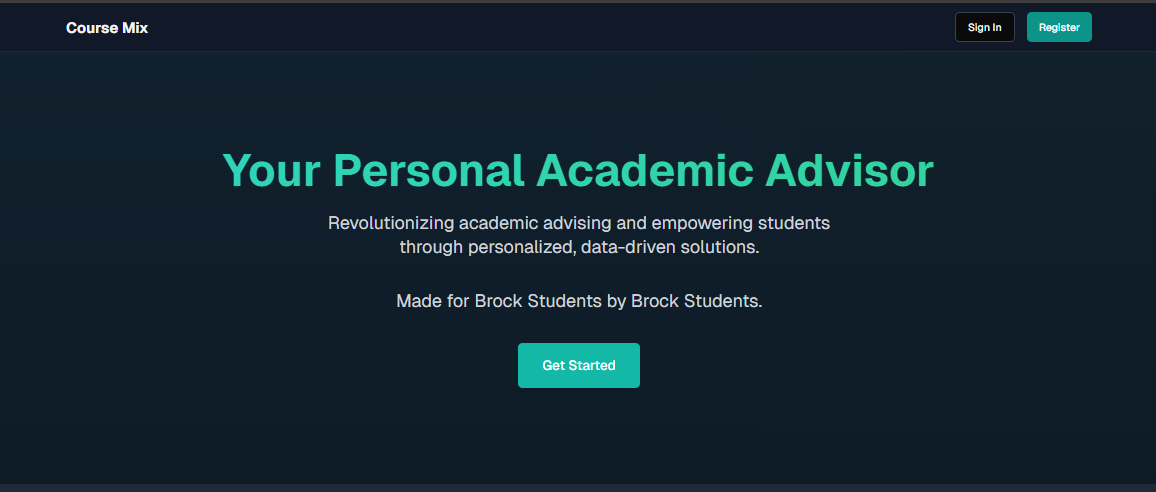
In order to install a local copy of our app, first clone the entire github repository by running “git clone <https://github.com/Avipatel1107/COSC4P02>”. Once the repository has been cloned locally, navigate to the directory of the app in your terminal. When you are in the COSC4P02 directory, run “cd coursemix” and here you will be in the directory of the entire application. From this folder, you will want to run npm install in order to install all packages and dependencies that our project relies on. After all packages are done being installed, you must create a “.env.local” file. This file will contain all the API keys necessary for our project. You will need the following: NEXT\_PUBLIC\_SUPABASE\_URL, NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY, SUPABASE\_SERVICE\_ROLE\_KEY, RESEND\_API\_KEY, GRADE\_ENCRYPTION\_SECRET.  
  
All of the Supabase keys can be generated in your own supabase projects settings, the Resend API key can also be generated through Resends platform, however the GRADE\_ENCRYPTION\_SECRET is the key which encrypts all the student grades with. Thus, this should be a random generated string of sufficient length, and must be kept private to ensure malicious users can’t access student grades. Once all of these keys are filled, you are now able to work on the project and develop it, you can view the app locally using “npm run dev”, from here you will be able to see the full application in your browser at the address localhost:3000 assuming the port 3000 is not already used

## User Manual

### Accessing the Application

The CourseMix website can be accessed by visiting:

<https://www.coursemix.ca/>.  
Upon visiting the site, you will land on the homepage, which highlights key application features, showcases student testimonials, and provides additional information in the footer section.



### Dashboard Overview

After logging in, users are welcomed with a personalized dashboard showing important academic information at a glance.

### 

### Profile Summary

* Displays your full name, student ID, and current program.
* Shows current term (e.g., Winter 2025), term progress, and reading week status.
* Provides an **academic overview**, including:
  + Current grade average.
  + Number of courses this term.
  + Total course completion status.

### Weekly Schedule

* A visual calendar of your registered courses.
* Classes are displayed by day and time, including course codes and instructor names.
* Empty slots reflect your available times for planning or self-study.

### 

### Quick Navigation

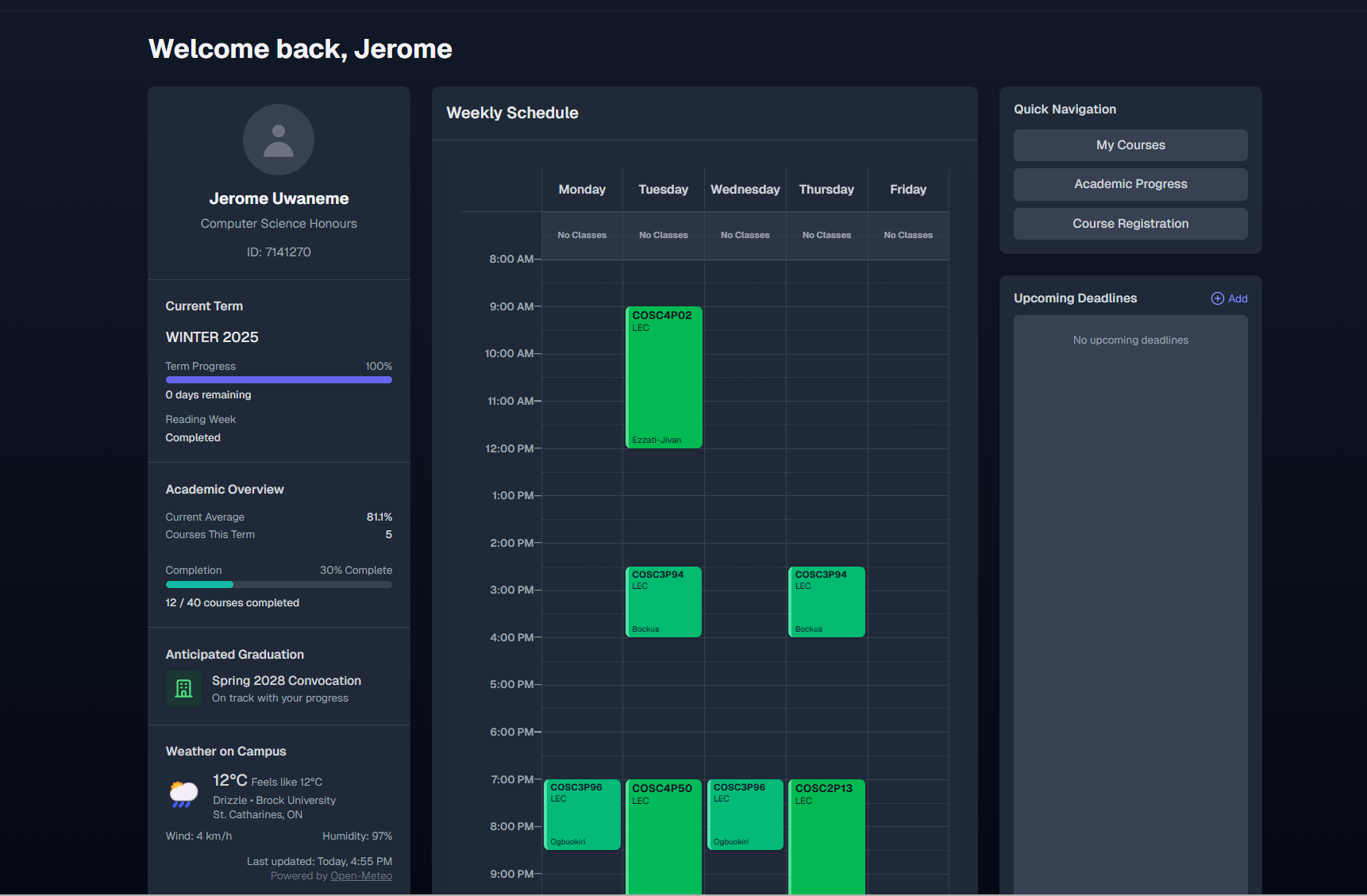
* Access core sections with one click:
  + **My Courses**
  + **Academic Progress**
  + **Course Registration**

### Upcoming Deadlines

* A panel for managing your tasks and deadlines.
* Click **Add** to input new assignments, tests, or reminders.

### Campus Weather

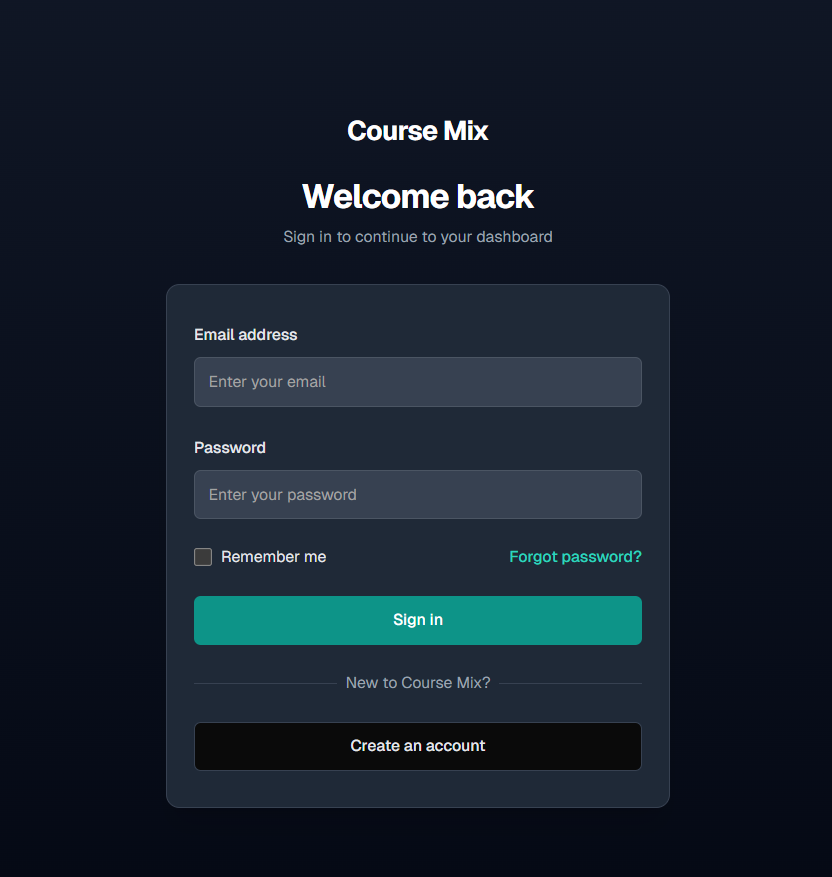
* Live weather updates specific to Brock University.
* Displays current temperature, wind, humidity, and forecast—powered by Open-Meteo.



### User Registration and Login

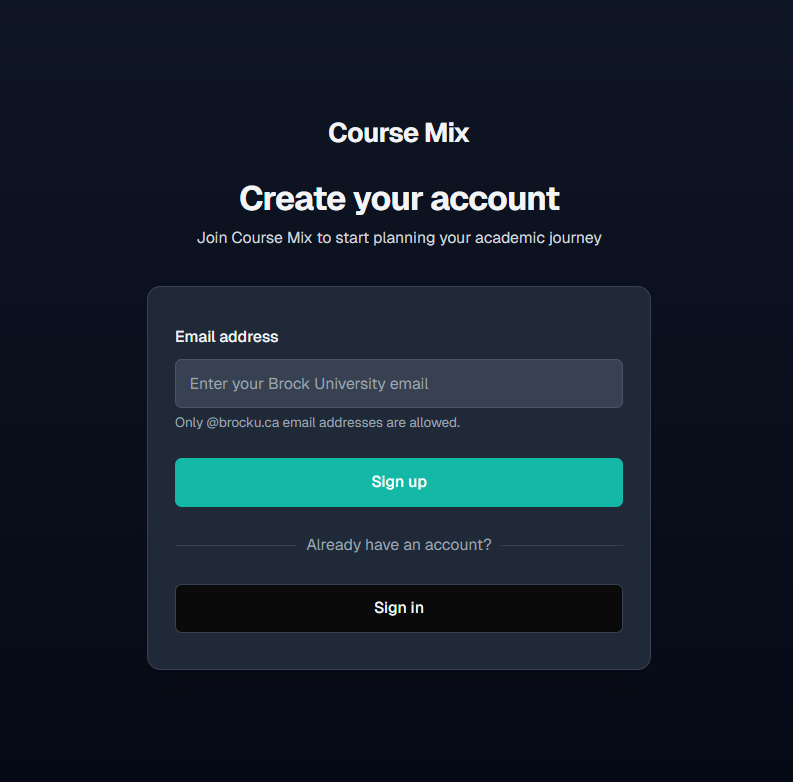
#### Signing In

To access your CourseMix dashboard:

1. Visit [**https://www.coursemix.ca**](https://www.coursemix.ca) and click on **Sign In**.
2. Enter your **email address** and **password** on the login screen.
3. Optionally, check **“Remember me”** to stay signed in.
4. Click **Sign In** to proceed.
5. If you forget your password, click **“Forgot password?”** to reset it via email.

#### Creating an Account

If you are new to CourseMix:

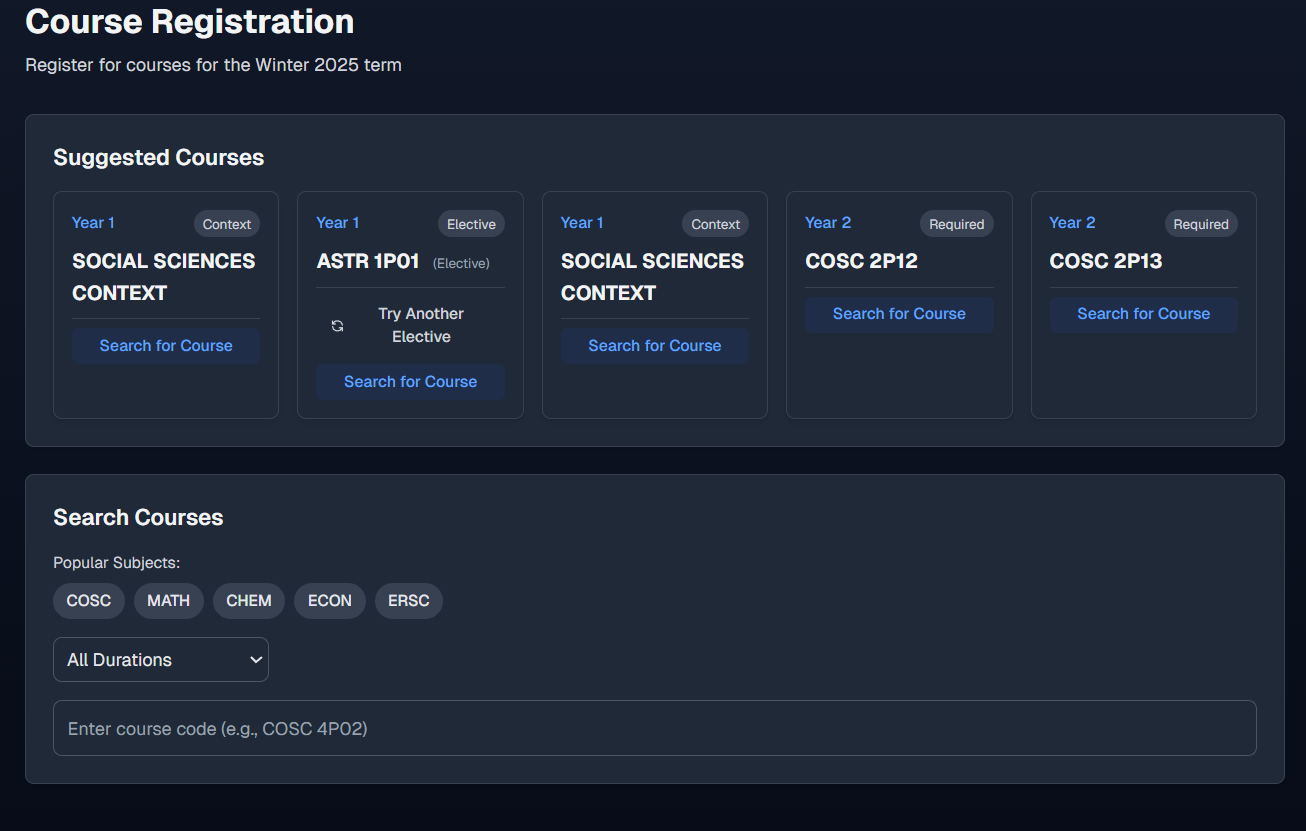
1. Click **Create an account** on the login page.
2. Enter your **Brock University email** (only @brocku.ca addresses are accepted).
3. Click **Sign up** to create your account.
4. After registration, check your email for any confirmation or onboarding messages.

### Course Registration

Once logged in, users can register for courses for an upcoming term using the **Course Registration** page. This section provides smart recommendations and flexible search features to make planning easy.

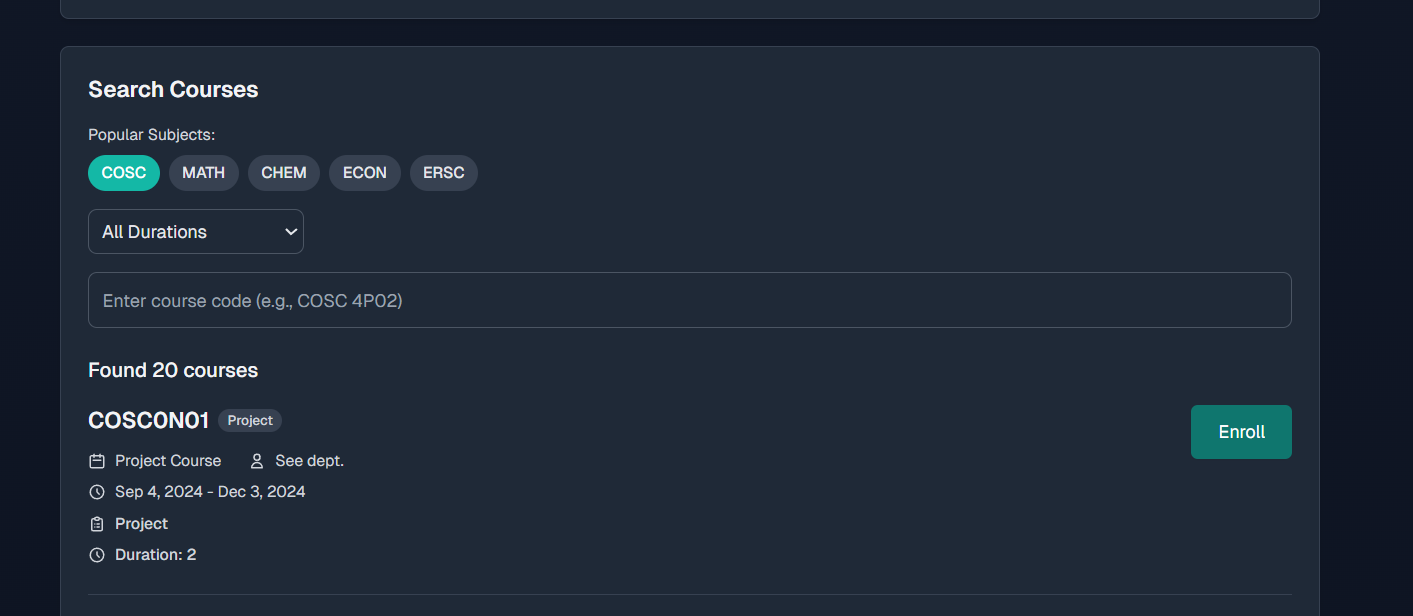
### Suggested Courses

* The app displays recommended courses based on your academic year, context requirements, and previously selected electives.
* Each course card shows:
  + **Year level**
  + **Type** (e.g., Required, Elective, Context)
  + A **Search for Course** button for further exploration.
* If you want a different elective suggestion, click the **refresh icon** to get another option.



### Search Courses

* Use the **Search Courses** section to manually look up any course:
  + Select from **popular subjects** (e.g., COSC, MATH, CHEM).
  + Filter by **duration** (e.g., All Durations).
  + Enter the **course code** (e.g., COSC 4P02) in the input box to search directly.



### My Courses

The **My Courses** page displays a personalized list of all courses a student is currently enrolled in for the selected term.

#### Course Details

Each course card includes:

* **Course Code** (e.g., COSC4P02)
* **Type**: Indicates if it's a lecture (LEC), lab, or seminar.
* **Instructor Name**: Who’s teaching the course?
* **Schedule**:
  + **Days** and **Time** the class takes place.

#### Managing Courses

* To **drop a course**, simply click the red **“Drop Course”** button on the corresponding card.
* To **add a new course**, click the **“Add Courses”** button in the top right, which will redirect you to the course registration page.

### Course Discussions

The **Course Discussions** section provides a space for students to ask questions, share insights, or talk about course-specific topics.

#### Creating a Post

* Use the **dropdown menu** at the top to select a course (e.g., COSC4P02).
* In the input field, type your message or question.
* Click the **Post** button to share it with others enrolled in the same course.

#### Filtering Discussions

* Use the **Filters panel** to search by:
  + **Keyword**
  + **Date**
  + **Time**
* Click **Clear Filters** to reset the filters and view all posts.  
    
    
  

### Course Reviews

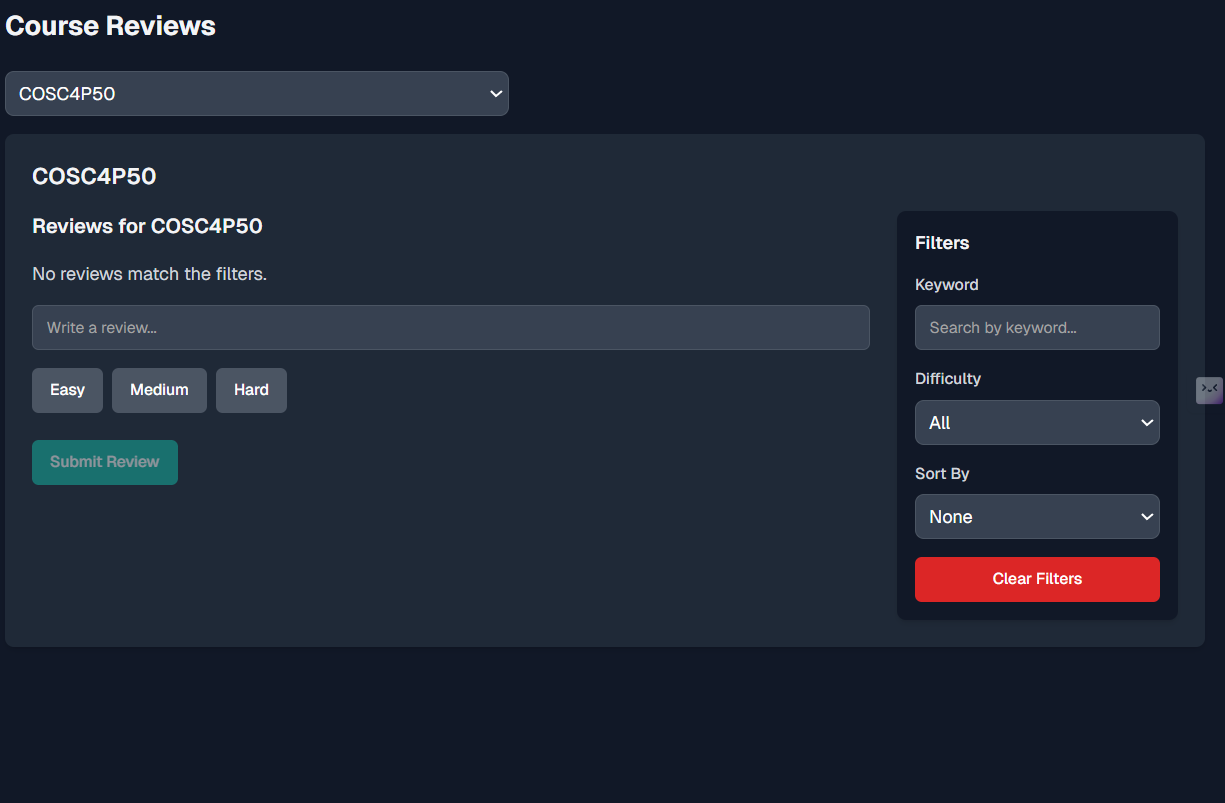
The **Course Reviews** section allows students to anonymously leave feedback on course difficulty and experience.

#### Submitting a Review

* Select a course from the **dropdown list** (e.g., COSC4P50).
* Write your review in the text box.
* Choose a **difficulty rating**: Easy, Medium, or Hard.
* Click **Submit Review** to post it.

#### Filtering and Sorting Reviews

* Use the filter panel to:
  + Search by **keyword**
  + Filter by **difficulty**
  + Sort by various options (e.g., most recent, most helpful)
* Click **Clear Filters** to remove all filters and view all reviews.



### Academic Progress

The **Academic Progress** section allows students to track their degree completion, grades, and estimated graduation timeline securely.

### Privacy Protection

All grade data is end-to-end encrypted using **AES-256**, ensuring that only the student can view their actual grades.

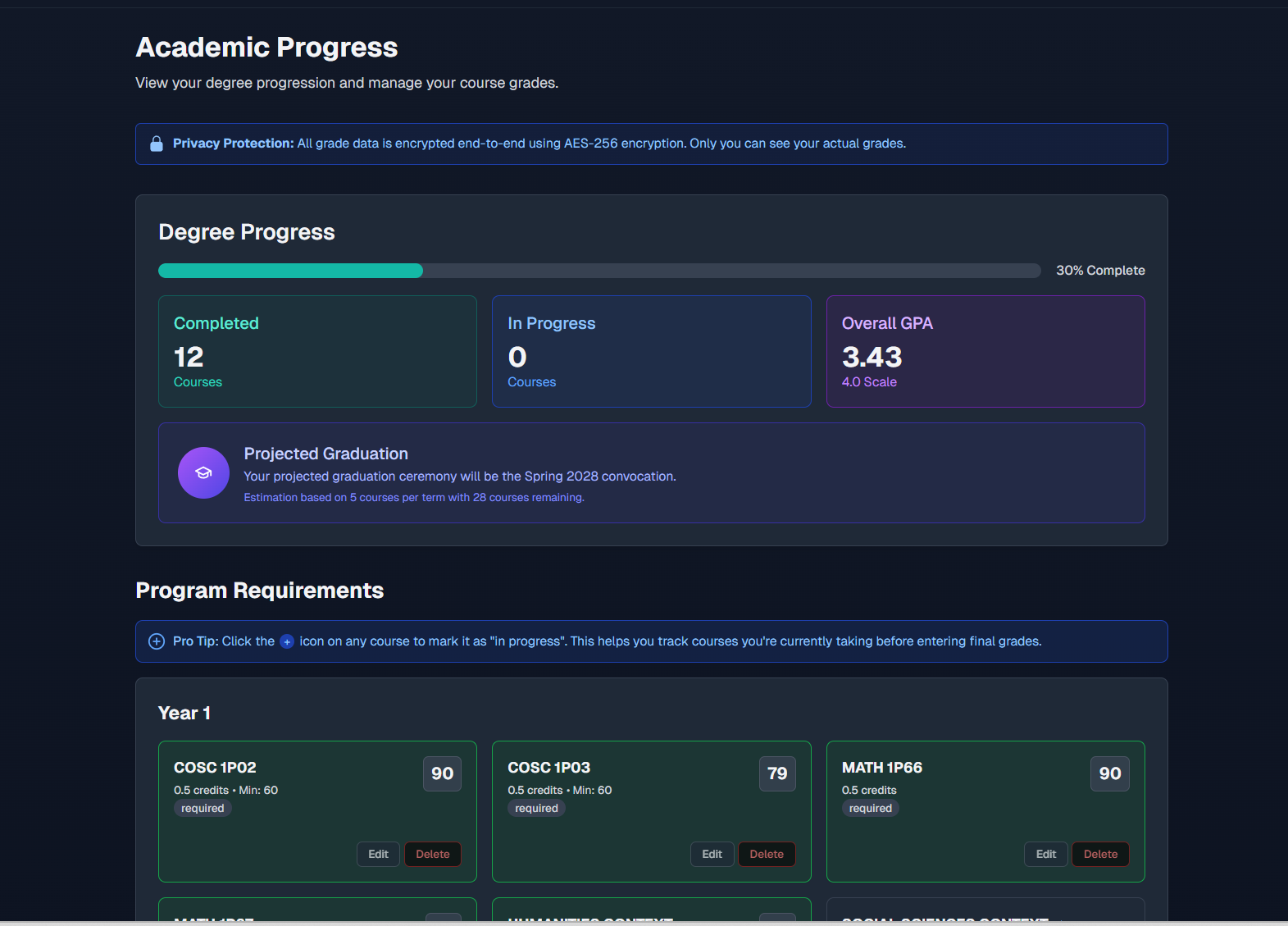
### Degree Progress

* Shows a visual completion bar based on your completed, in-progress, and remaining courses.
* Key stats include:
  + **Completed Courses**: Total number successfully finished.
  + **In Progress**: Courses currently underway.
  + **Overall GPA**: Displayed on a **4.0 scale**.

### Projected Graduation

* Estimates your graduation term based on your current pace (e.g., 5 courses per term).
* Shows how many courses are left to complete your degree.

### Program Requirements

* Displays required and elective courses by **year level** (e.g., Year 1).
* Each course card shows:
  + **Course Code & Credits**
  + **Minimum Pass Mark**
  + **Final Grade** (if completed)
* Buttons available:
  + **Edit**: Update the course grade or status.
  + **Delete**: Remove the course from your record.
* A tip reminds users that clicking the status icon will mark a course as *“In Progress.”*

### Your Profile

The **Profile** section lets users manage their personal and academic information with ease.

#### Profile Overview

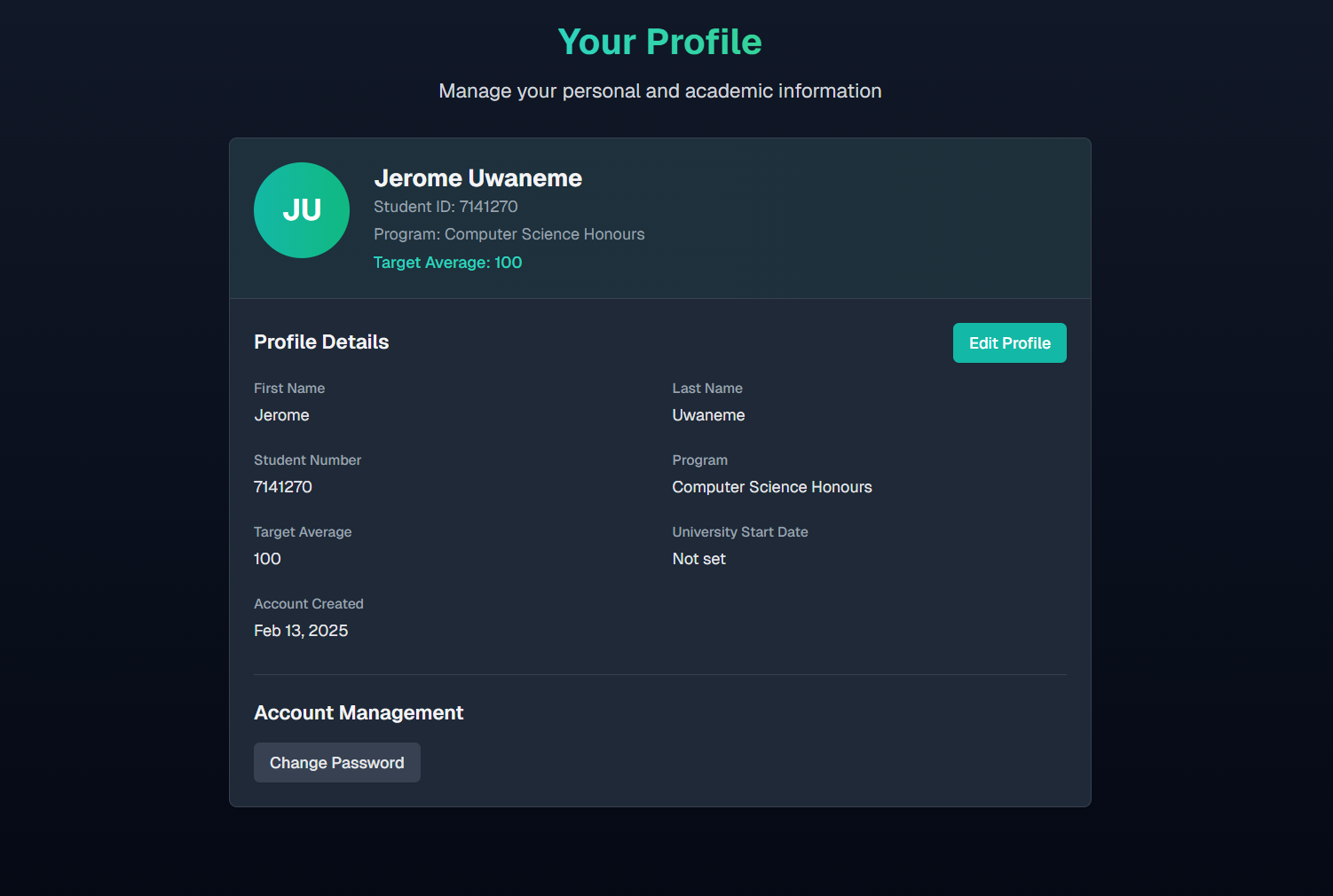
* Displays your **full name**, **student ID**, and **program**.
* Shows your **target average** for academic planning and motivation.
* Lists the **date your account was created** and optional details like **university start date**.

#### Editing Your Profile

* Click the **Edit Profile** button to update any of the following:
  + First/Last Name
  + Target Average
  + University Start Date

### Account Management

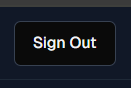
You can change your password securely by clicking **Change Password** in the Account Management panel.



### Signing Out

To end your session securely, click the **Sign Out** button, typically located in the top-right or bottom corner of the application interface.

* Clicking this will log you out of your CourseMix account and return you to the homepage or login screen.
* Always sign out, especially when using shared or public devices, to protect your academic and personal data.



# Testing (Ola + Avi)

## Tests

### Framework

We used **Jest** to test the core logic and functionality of our application. It runs test files in a dedicated \_tests\_ folder and compares actual outputs against expected results, enabling us to quickly identify and resolve issues. We wrote tests for things like helper functions and context logic, and Jest made it easy to isolate those pieces and make sure they work on their own. It also supports mocking, so we could simulate external dependencies without relying on real data. This allowed us to test edge cases and unexpected inputs like invalid dates, missing values, or unusual user behaviour that might break the app. By simulating these scenarios in a controlled test environment, we ensured our code responded properly, either by returning acceptable default values or displaying clear error messages.

### What was Tested?

The \_\_tests\_\_ folder in the coursemix directory contains a structured approach to testing the CourseMix application, focusing on key areas like user authentication, API integration, component rendering, and performance. These tests are designed to be both thorough and straightforward, ensuring robust coverage of edge cases and real-world scenarios. For example, the sign-up functionality is rigorously tested for various conditions, including invalid email inputs, empty fields, already registered emails, and both successful and unsuccessful API responses. These tests simulate real-world behaviour by mocking API endpoints, such as the verification email service, to ensure the application handles different scenarios as expected, including server and network errors.

In addition to functionality, the tests include performance validation to ensure critical features, such as form submissions and API calls executed efficiently without delays. The tests also validate error-handling mechanisms, ensuring appropriate feedback messages are displayed to users during failures or unexpected behaviours. This approach enhances the application's usability by addressing both functional and non-functional aspects comprehensively.

By covering UI rendering, form validation, error handling, API integration, and performance, these tests provide a decent foundation for maintaining the application's quality and reliability. The structured design of the tests ensures that new features can be added without breaking existing functionality. This testing framework not only ensures that the application is robust and user-friendly but also lays the groundwork for scalable and maintainable development in the future.

### Test Case Sample

[Test.pdf](https://drive.google.com/open?id=17Pdkx9C-KfDYkZyz9NFif7WjwM8D7ACU)

If the pdf does not work here is the link for reference:<https://github.com/Avipatel1107/COSC4P02/blob/main/coursemix/__tests__/app/auth-pages/sign-up/page.test.tsx>

This file above is one of the first test cases we’ve done, it contains unit tests for the SignUp page. The SignUp page is responsible for user registration, specifically validating and processing email addresses with a strict focus on ensuring only @brocku.ca email domains are accepted. These tests were implemented using the Jest testing framework alongside @testing-library/react to simulate user interactions and validate the expected behavior of the SignUp form. The tests include rendering verification to ensure all essential UI elements, such as input fields, buttons, and instructional text, are correctly displayed. Validation tests cover scenarios like empty input fields, invalid email formats, domain typos, and successful submission of valid @brocku.ca email addresses, including handling uppercase domains. The framework makes use of mocking for several instances, for example, the fetch API is mocked to simulate network requests and responses, and the next/navigation module is mocked to verify navigation behavior. Error handling is also thoroughly tested, ensuring proper messages are displayed for invalid input, API errors, and network failures. Additionally, the tests also confirm that the form displays a loading state during submission and that the submit button is disabled to prevent duplicate requests and crashing of the webpage. Overall, this file ensures the SignUp page's functionality is robust, adheres to validation rules, and provides appropriate feedback for various user interactions and edge cases.

### Continuous Integration

We attempted to implement Continuous Integration (CI) using GitHub Actions by setting up a custom workflow to automatically test and validate our code on each push and also run daily to ensure our code was working as intended. However, we faced challenges in configuring the correct file structure, which prevented the workflow from running as expected. Despite our efforts, the paths and dependencies didn’t align properly with the active/live environment and it was unable to parse through the pages as intended. If it were successfully implemented, CI would have streamlined our development and testing process by catching errors early when code is pushed, ensuring consistent code quality, and reducing manual testing efforts which would speed up collaboration and deployment.

## Results

### Pertinent screenshots:

### 

### 

### Overview of Test Results

### 

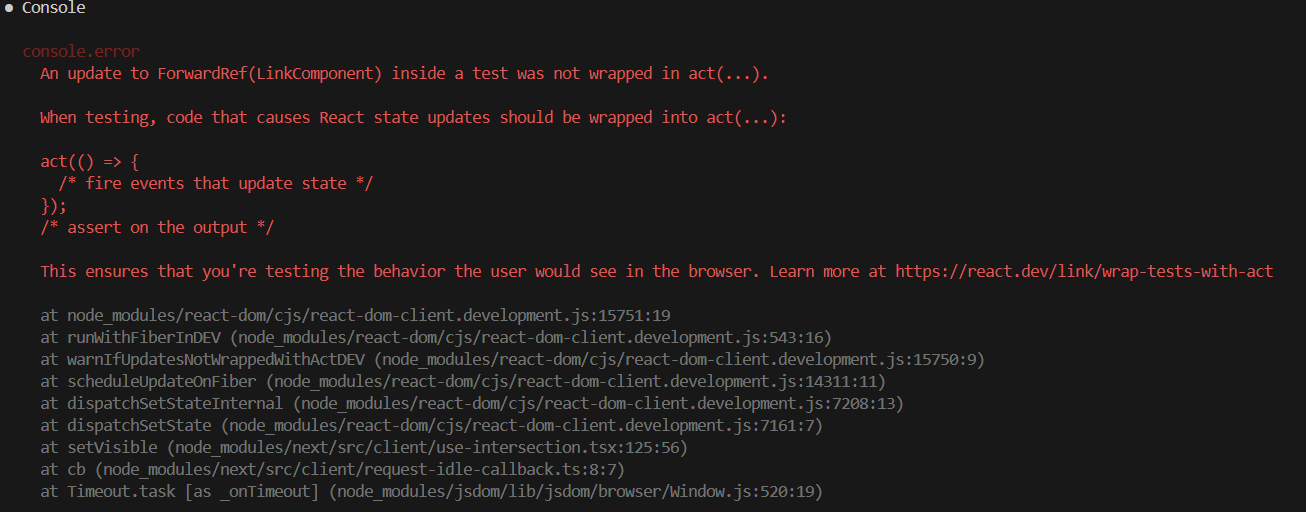
Out of 6 test suites, 5 passed and 1 failed. Notably, all 33 individual test cases across these suites were successful.

**Key Observations**:

* The failure occurred at the suite level, not within individual test cases.
* This indicates an issue with setup or teardown logic (e.g., beforeAll/afterAll hooks) or the environment configuration.

**Probable Causes**:

* **Misconfigured Environment**: Potential issues include missing mocks, incorrect environment variables, or resource leaks such as unclosed connections.
* **Unhandled Asynchronous Operations**: Errors may stem from asynchronous logic in cleanup processes or unresolved promises occurring outside the main test executions. For example, an afterAll hook could throw an error after all tests have completed, causing the suite to be marked as failed—even though all test cases passed successfully.

**Conclusion**: Although all test cases succeeded, the suite-level failure suggests issues related to infrastructure or test lifecycle processes. This poses the question: Are there known shortcomings in the setup or teardown procedures that could explain the failure? If so, the failure may be considered expected. If not, it warrants further investigation to prevent false negatives (or misleading failures) in future test runs. This issue is indubitably resolvable by auditing the hooks and asynchronous logic in the failing suite and reviewing error logs for specific details. Correspondingly:  
  
**Explanation of the Error**: This warning commonly occurs in React testing when a state update (such as one triggered by rendering, an event, or an asynchronous action) happens outside the act() utility. The act function simulates user interactions and ensures that all updates are flushed before assertions are made. Notably, test runners like Jest may not wait for these asynchronous operations to complete, resulting in React issuing a warning about an unhandled state update. In this case, the warning was triggered by a delayed state change involving the Link component.

While this doesn’t affect the correctness of the test outcomes—all tests passed—the warning and subsequent suite failure can be resolved by explicitly wrapping asynchronous actions or event triggers in act(...). However, since it doesn’t impact actual test results, we considered it **non-critical.**

# GitHub Log Activities (Ola)

*The Contributions of line commits for Avi may seem high because the gitignore file, for some reason, did not ignore the node\_modules folder, and it was pushed out with other code, but was quickly reverted back*hi


# Team Contribution

| **Team Member** | **Contribution** |
| --- | --- |
| **Everyone** | **For the Project:** planning through Jira, source code through GitHub (see Section 11 above for details), data entry through Supabase, concept diagrams, user testing |
| Ashu Chauhan | **Report:** Installation Technical Menu  **Presentation:** Architecture, Implementation, AI Use, Projected Graduation Algorithm, Limitations. |
| Avi Patel | **Report:** Tests  **Presentation:** SE Activities (Dashboard, Course Registration, Database Table Generation) |
| Fatima Abourida | **Report:** format skeleton, release  **Presentation:** format skeleton, introduction, team procedure, edits |
| Jerome Uwaneme | **Report**: User manual  **Presentation**: Course Reviews and Discussions Overview (Features and design) |
| Olaoluwa Akanji | **For the report**: GitHub Contributions, Test Results Analysis and Interpretation.  **For the presentation**: Oversight of Software Engineering Practices (agile scrum methodology and project management frameworks) and detailed test structure and implementation.  . |
| Oreoluwa Akanji | **Report:** Introduction  **Presentation:** Marketing: Social Media |
| Russell Salacup | **Report:** Requirements, layout consistency  **Presentation:** Estimation of Contributions |

*Table 1.*

# 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 16 (with TA): presented our idea.
* January 19: Finalized our Release planning document
* January 21: Discuss project architecture.
* January 24: Decided on project architecture
* January 28 (with TA): 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
* February 25 (with TA): our live version is bugged, switching tabs affects proper data rendering. Branden mentioned the importance of incorporating automated testing and detailed contributions outside GitHub.
* March 4: Discuss GitHub issues, assign all issues, and manually add programs to the database
* March 11 (with TA): Decided to pause manual addition of programs, Brendan mentioned to include testing throughout, not after.
* March 18: Discussed the Progress Report 2. Divided tasks.
* March 21: Went over progress
* March 25 (with TA): scrapped notifications. Collaboration metrics were emphasized.
* April 4: planning to divide presentation, however, still needed few tasks to be done first, so we decided to circle back next meeting.
* April 8 (with TA): tasks left are to work on test cases and work on the presentation.
* April 17: Divided presentation slide topics
* April 20: 2 practice rounds through the presentation.

## Reports

* [MINUTES FOLDER](https://drive.google.com/drive/folders/11yFFUmQwHvp1TcY0EvnWwmKmt-sys-Jx?usp=sharing)
* [Project Proposal](https://docs.google.com/document/d/1mlFtKhi-uaHOAJ8gJ_abDU2cnqQrddc8mWpRGbxl44Q/edit?usp=sharing)
* [Release Planning](https://docs.google.com/document/d/1YpU2WBj83A-_kdDWmC0WMJbP9OyNrPpKmVMAxThPhms/edit?usp=sharing)
* [Progress Report 1](https://docs.google.com/document/d/1f9w36iE6i620YcarOsVI9698Ylgtka-dPipjvXvDyOg/edit?usp=sharing)
* [Progress Report 2](https://docs.google.com/document/d/14zg-I6Nm_B5Nj7CgJPvxiDT2AgiD4Y7dsk9kPXQhbDk/edit?usp=sharing)
* [Final Presentation](https://docs.google.com/presentation/d/1GFndxw64icFveDGsOOXXczzs4u_fQxHmAIcEEt80X9E/edit?usp=sharing)
* [Product Launch Video](https://drive.google.com/file/d/16bSpluelEuQlNApfzUamUWOQuHP_azfK/view?usp=sharing)