**Tab 1**

Progress Report 2

Course Title: COSC 4P02

Course Instructor: Naser Ezzati-Jivan

Due Date: March 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) 5**](#_enjb7z76yxz8)

[3.1. Academic Progress Page 5](#_btdgigje5otx)

[3.1.1. Data Requirements and Safety 5](#_bmcb6wrq0a6h)

[3.1.2. Comprehensive Description of the Page 6](#_qb8u1za1ti5h)

[3.1.3. The projected graduation algorithm 6](#_mbgjeqx6vk3j)

[3.2. Suggested courses/timetable 7](#_tzvcjyr6rhit)

[3.2.1. What is it? 7](#_p9lz9hdb67kf)

[3.2.2. The Algorithm 7](#_bmujsklr3qnj)

[3.2.3. Electives 8](#_fh8qxfa1gv9d)

[3.2.4. Prerequisites 8](#_jhwcpwo6yidu)

[3.2.5. Original Idea vs Current Implementation 8](#_hi9xko4yd140)

[3.3. Future Proofing 9](#_vquarebnbyho)

[3.4. Supabase policy changes (Ola) 10](#_gwcrqmbox7lq)

[**4. Release (Fatima) 12**](#_o5zndgrz8dh7)

[4.1. Scrum meet 12](#_xzqhjbcc1hku)

[4.2 Current Sprint 12](#_jx4qt1j49nan)

[4.3 Course Mix Social Media Content Strategy (Ore) 13](#_yadphdgapsjo)

[4.4 Next steps (Ola) 14](#_4jm77r6ppjqg)

[**5. Problems/Discussions (Ola) 14**](#_feftga4ns6uc)

[**6. GitHub Log Activities (Ola) 17**](#_mbfhl2kvdgmr)

[**7. Team Contribution 18**](#_n7l7pmay7som)

# 

# Introduction (Russell)

By now, the app’s core functionality nears completion; only two Jira tasks on Sprint 3 are in progress. Miscellaneous features, stress / input testing, data population, and presentation remain as outstanding tasks.

Some pages have been expanded with additional functionality (for instance, what was once only a form for inputting grades to fuel suggestions now also prominently features the bar indicating progress towards graduation, and both the user and the site itself can mark courses as in progress, further dividing this infographic).

It remains to be seen how well the site will hold once users begin utilizing it, as well as just how many programs and courses it can track as it scales up. Ideally, the same page code will be used, and we only need to add rows to the back-end.

To avoid exposing test cases to end users, input testing will be done on separate branches, and will remain un-merged with main, as the main branch is automatically synced to the live site. **This may cause testers to remain hidden from the GitHub Contribution infographics.**

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

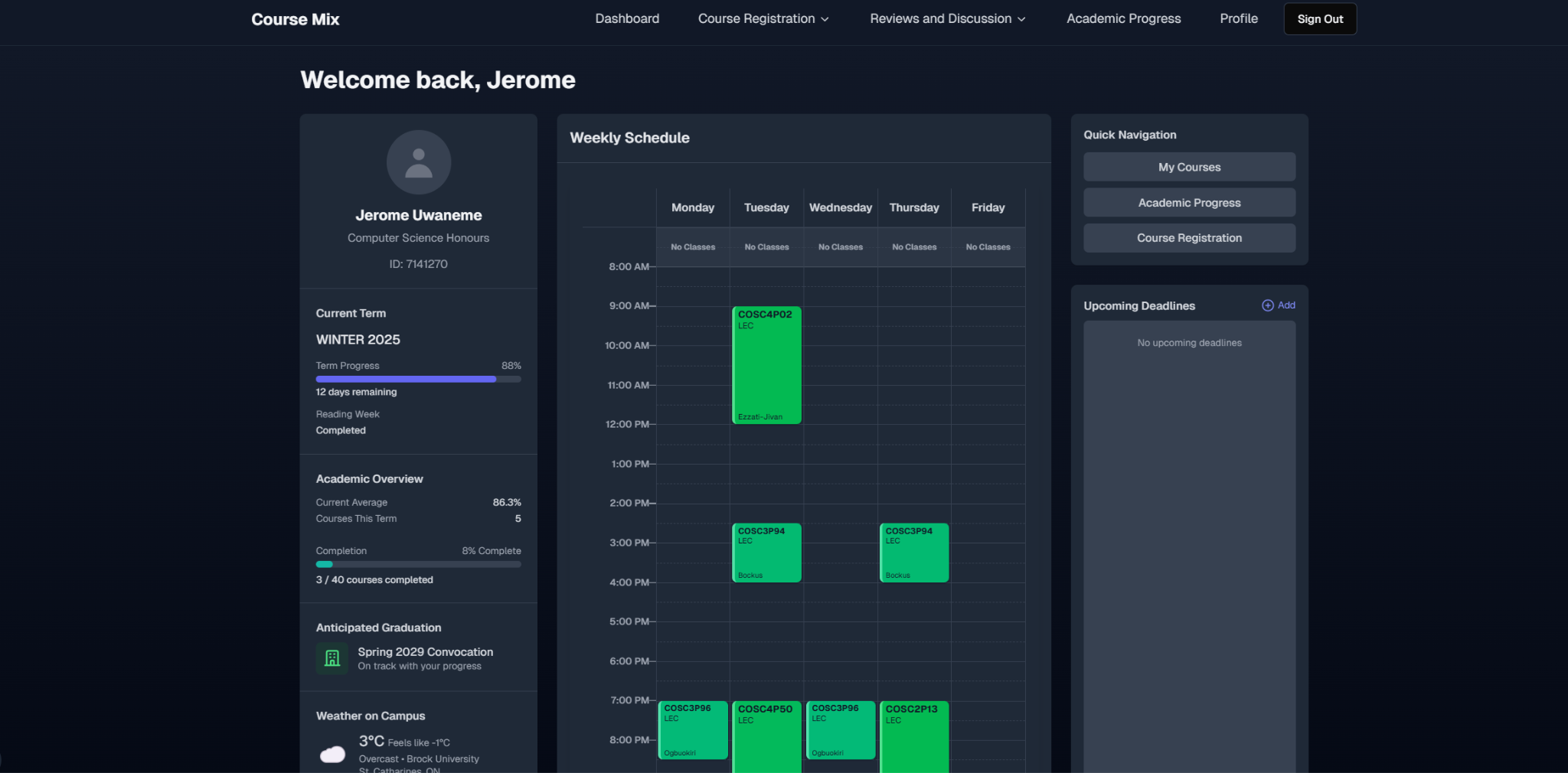
Github: <https://github.com/Avipatel1107/COSC4P02>

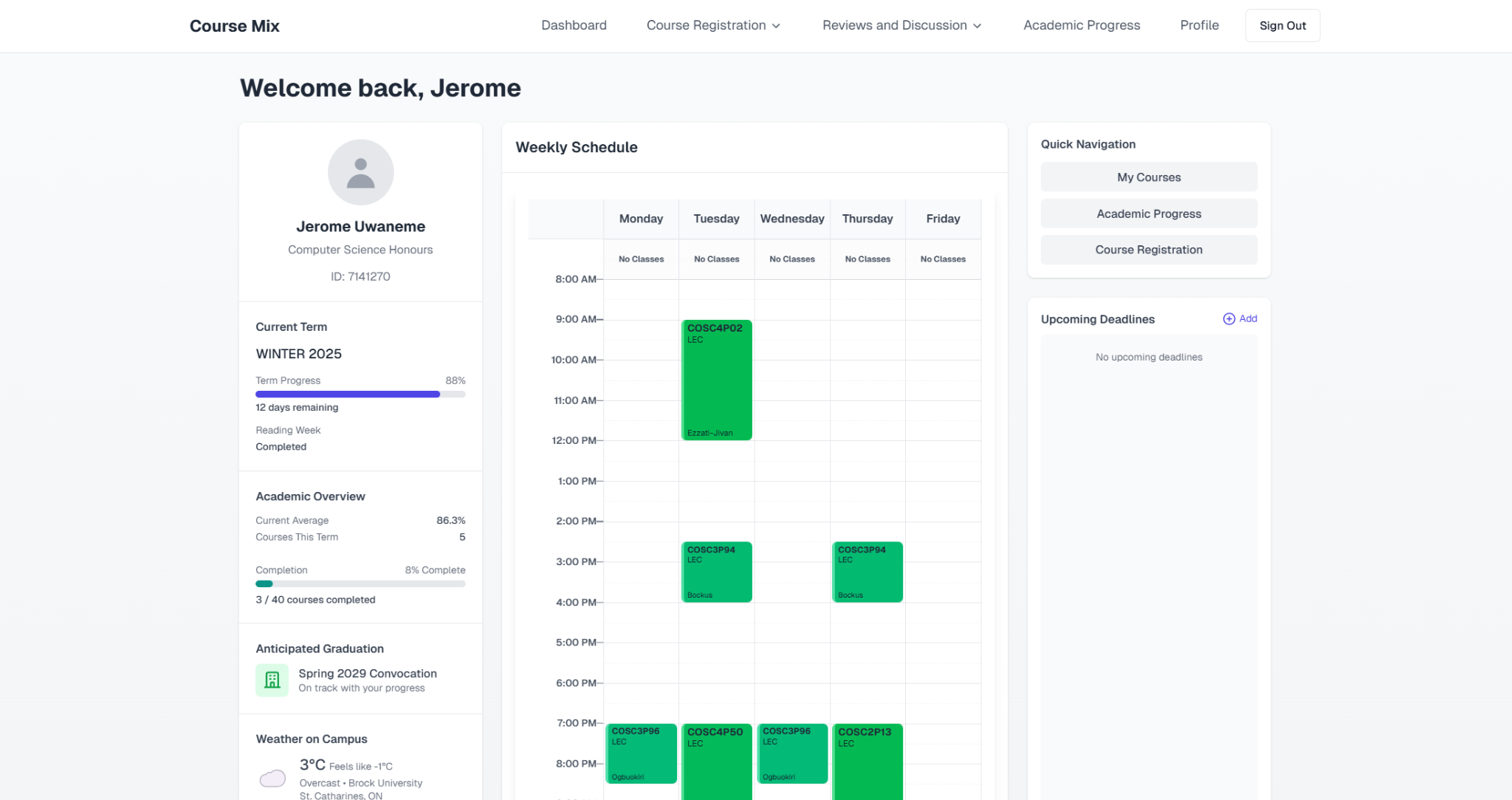
# Design (Jerome)

**Dark mode and light mode**

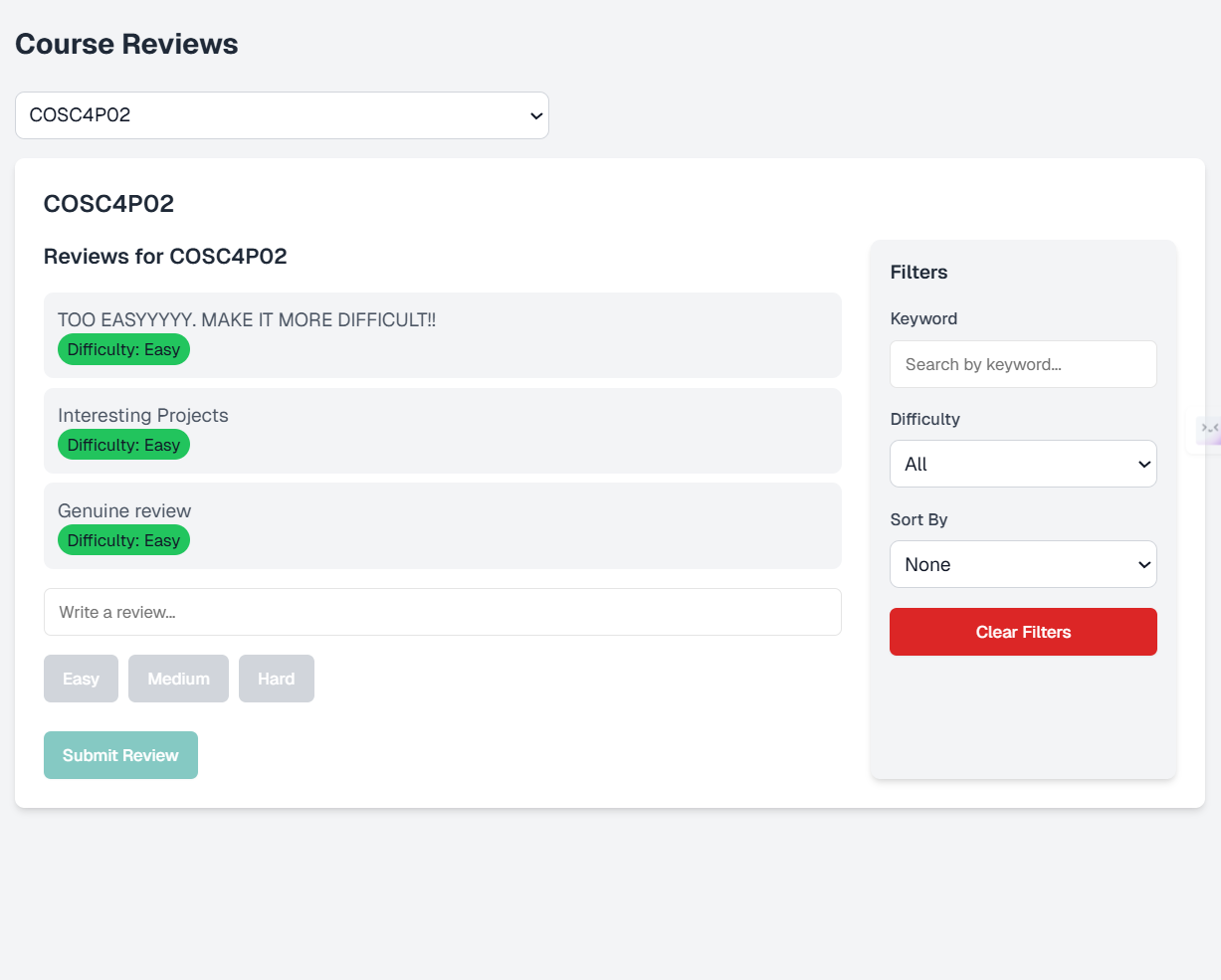
While maintaining the existing functionality, we have introduced a new feature that enhances user experience: Dark Mode and Light Mode. This addition allows users to customize the interface according to their visual preferences, reducing eye strain and improving readability in different lighting

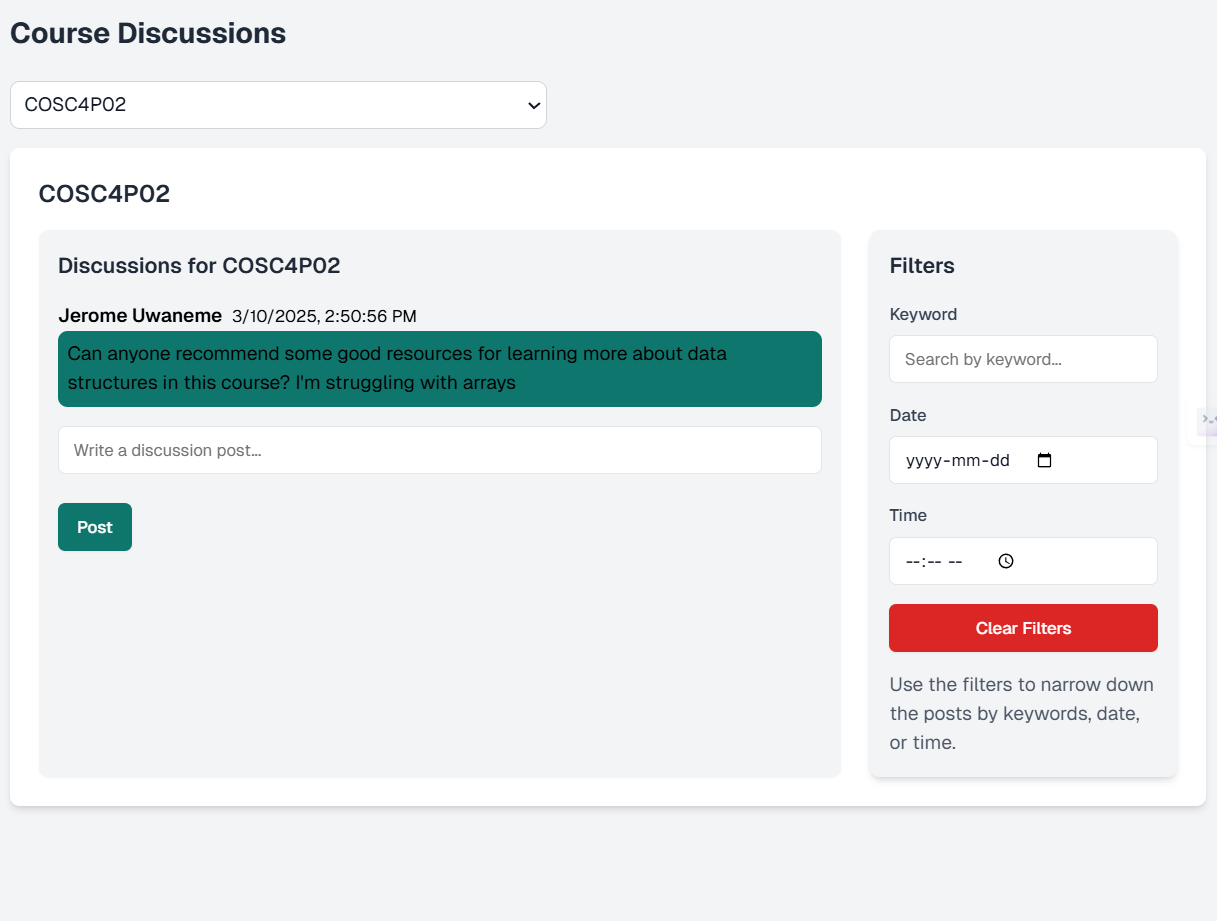
conditions.





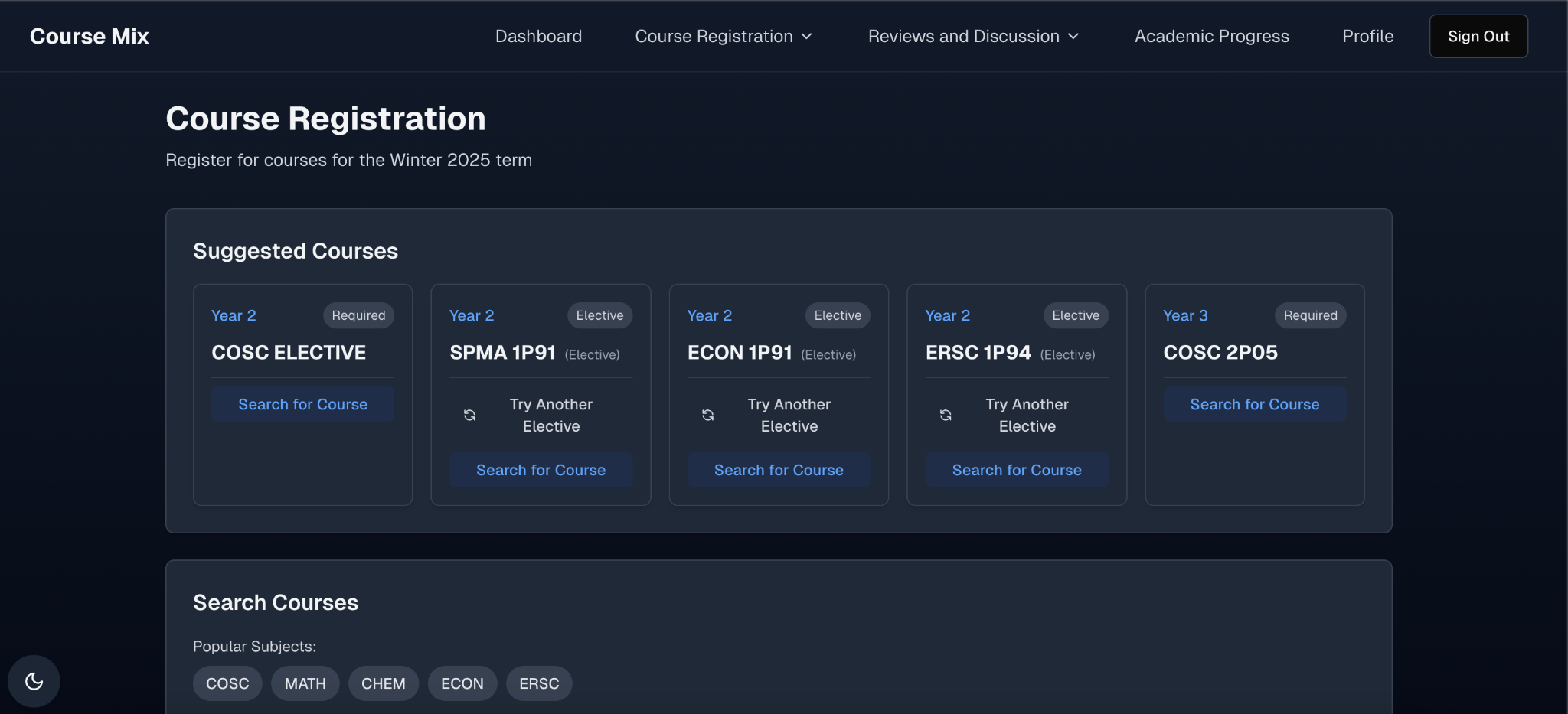
**Course Reviews and Discussions**

While keeping the old functionality we decided to integrate a new course discussions feature, making it easier for users to engage with each other and the course material. Furthermore, we implemented a filter system to easily allow users to navigate and find relevant reviews and discussions based on their preferences. The filter system includes options to sort by difficulty level, keywords, and recent activity, ensuring a more personalized and efficient experience.  
  
  




**Course Recommendations/Suggestions**

This was just a design choice we decided to implement for suggesting courses to students, an in depth explanation into the algorithm and implementation choices will be explained in section 3.2.



# Implementation (Avi, Ashu)

## Academic Progress Page

##### Data Requirements and Safety

Due to the nature of the academic progress page, we had to ensure that not only the correct data was being shown to users, but also that all the data was secure and private as the information being passed around is rather sensitive. The academic progress page populates based on a students program, which is changeable in their profile settings. Since the academic progress page stores sensitive data such as student grades, we had to ensure that this was secure so that us owners can’t easily just view student grades, and also when communications are made to the server and back, malicious users cannot intercept this information. To do this we encrypted the grades when they get sent to the database, and decrypted them to display them for the end user. The encryption key is a randomly generated string that no one has stored including us. This ensures that when grades are inputted into the database, we see a random string in the grades field, and not the integer value, as each row is associated with a specific user, and we would be able to see sensitive information on students.

##### Comprehensive Description of the Page

The academic progress page fetches the students profile, and using this information matches the program's id, which is a primary key in our database to the foreign key reference in our program\_requirements table. Inside of this table, we hold all of the requirements for each program that we have inputted. This design makes the software very scalable as adding new programs is simply just updating the database with new entries, and no refactoring of code is necessary. Inside the program requirements we have columns that say the course code, the type of course (elective, core course, context credit), program year, and status. These are the most necessary fields for the academic progress page as they are the fields that determine how the academic page is rendered. When a course is marked as in progress, the status is changed to “in progress”, and when a grade is entered it changes to “completed”. This column determines the color of the card. When a grade is entered, it gets added to a student\_grades table, and if this grade is not greater than the course's minimum grade (if there is one, this value could be null), then the course card will change its border to red to mark it as a failed course. Our code is very scalable and works for programs of all shapes and sizes, since the course cards are simply pulled from the table, and mapped out accordingly, this makes it so that our code likely needs little maintenance for the current layout of degrees, and rather just updates to the degree requirements if Brock decides to change these. A key feature that exists on this page is a projected graduation date tracker, which dynamically tracks the time left until graduation based on the courses completed by a user currently, and how long they have left. Another key feature is the GPA calculation, which is something that Brock currently does not offer to students, this is a great key feature that students can utilize as it is relatively standardized. We also offer a progress bar to see how far into your degree you are, so students continue to see their progress and stay motivated knowing that they are making progress and chipping away at their degree.

##### The projected graduation algorithm

The algorithm gathers the currently completed, and current in progress courses, and compares that to the total number of courses required in your degree. Then using this information, it figures out how many courses are required to complete your degree. The algorithm will assume that a student will always take 5 courses in fall and winter term, unless the user has less remaining. It then calculates how many terms are left to graduate based on the remaining courses, and from there it figures out when the student is likely to finish their studies. The algorithm does not factor in that students will take spring courses. This was an intentional design choice as after a fair bit of experimenting, we found that factoring in spring courses is very inconsistent, as a majority of programs do not actually depend on spring courses being taken to progress naturally in the degree, and are rather used as a way to get ahead in your degree. Using the studies completed projection, we then find the nearest convocation from this, which is either the fall or spring convocation, and that is what we display to the end user. Ultimately, this projection does have its drawbacks, and could be faulty, however we found that altering the algorithm would result in many more inconsistencies, and this was the most reliable algorithm we found. This is definitely something that can be improved upon potentially, however due to time constraints and attempting to prioritize bigger tasks, we made the choice that since this is simply a *projection* and is not to be taken as a guaranteed date, it is acceptable due to its somewhat dependable accuracy.

## Suggested courses/timetable

##### What is it?

As a core feature of our application, we always wanted to create this app with the intention of helping students progress through their program. To help them achieve this, we always wanted to have a feature that suggested courses for students to take. To achieve this, we added a suggestions panel on the course registration page. There are 5 courses displayed here that are suggested to you based on your current progress in your degree.

##### The Algorithm

To accurately display relevant courses, we gather the students' progress in their degree from their academic progress page, which is all mostly stored in the student\_grades table. From this information, we gather 5 courses based on the students earliest year courses left which are still not complete. If there exists 5 courses then these 5 are suggested (assuming pre-reqs are not required). In a simpler way, the algorithm goes to a user's grades, gathers them all, compares them with the program's requirements, goes through the first year courses, checks if all the requirements are completed, and goes through each year sequentially. The first 5 courses it finds, with priority to earliest year courses get suggested. If a course requires a certain prerequisite to be taken, then this course will be swapped for the next available course. Available meaning a course that is currently allowed for the user to take given that they have met the requirements for it. For example, hypothetically, let’s say that a student has completed every year's requirements and is in 4th year. Our suggestions will not suggest COSC 4P02 to a user until 4P01 has a grade entered into it, which in turn marks it as “completed”, this ensures that the suggestions stay relevant to each user's needs. Whenever we have an elective for a student, we manage to handle this in a unique way which will be discussed in a separate section, but this is a simple implementation to a complex problem.

##### Electives

Electives are a part of every program, and thus we had to come up with a unique way to deal with them. To handle these, we decided that any true ELECTIVE, aka not a course that is required to be a program course and above 3x00+ for example, we decided that we would title these “ELECTIVE” in our database to ensure consistency. When our algorithm spots an “ELECTIVE” course code, it replaces the text with a random option from a table in our database titled “elective\_suggestions”. This table is currently filled with courses that we find are typically taken by students at Brock as electives, and courses that students tend to enjoy and perform well academically. We wanted to do this, as the goal is to help students perform better academically, and thus suggesting courses that have proven track records of helping students to excel and buff up their grades and are a great way to help them achieve this goal.

##### Prerequisites

To handle prerequisites, we simply added a table called “prerequisites” and in here, we have a course code column, and a prereq column which also stores a course code. We simply perform a check in our code to see if this course code exists in the pre-req table, and what the pre-reqs for the course code are, and if there exists any, we check to see if they are completed or not. This solution is very scalable and maintainable as we simply just need to worry about data entry as opposed to code refactoring in theory. Currently, it is only scaled for COSC students just as a proof of concept but it can be scaled towards other programs at Brock.

##### Original Idea vs Current Implementation

Originally we had intended and hoped to suggest courses based on course difficulties, and use your current grades, and target average to help align the suggestions to your academic standing so that you could achieve your dream gpa/average. Unfortunately, we quickly realized that this plan was not achievable with the current dataset available to us. We simply had no way to accurately measure course difficulties, especially outside of Computer Science, and since we wanted to ensure that this application can be easily scalable up from CS, any implementation idea we had, would require far too much effort given the reward. Our current implementation is still personalized, and accurate, and is a cool feature, but it could be supercharged if we had the necessary data. As a clever trick, we discussed the possibility of using our Course Reviews feature, and course discussions as a way to actually gather this data. Ultimately, to get the data necessary to build out a feature like this, we would need a much higher user base, and this requires marketing the app to Brock students, and hopefully getting students to actually use it, and make use of the course reviews page. However, if we did get a large enough user base, we could in theory achieve the original goal by creating a pipeline that would use the reviews and discussions students post for courses to accurately determine course difficulties. From there we could use these newfound difficulties to revamp the entire suggestions system from being a simple algorithm to a much more complex suggestion network that takes in many more complex factors in determining the correct suggestions. Ultimately, I do not believe that we will be able to implement this feature by the end of this course, but whether or not we can actually get a user base after this course concludes, and collect the necessary data to build a more accurate suggestions feature is to be seen.

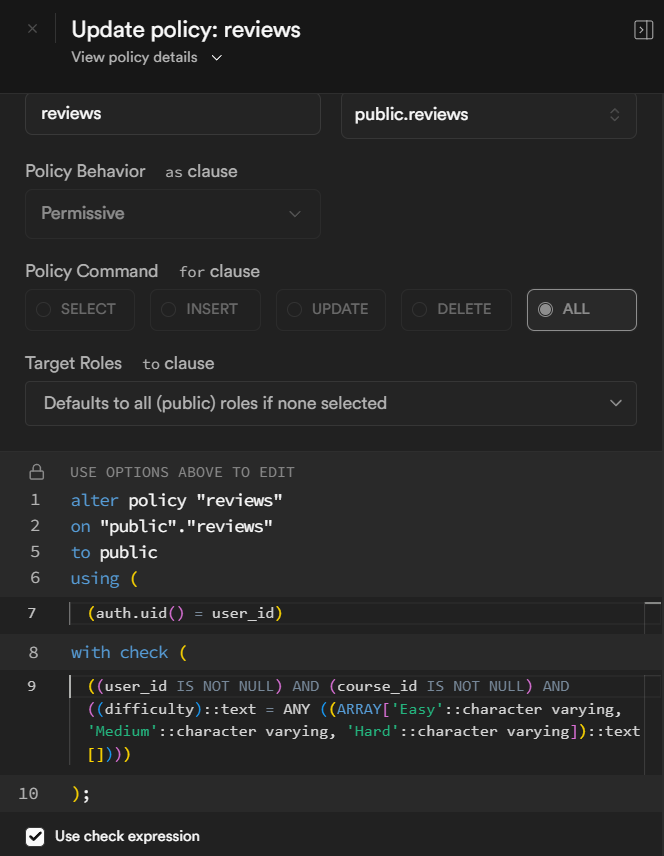
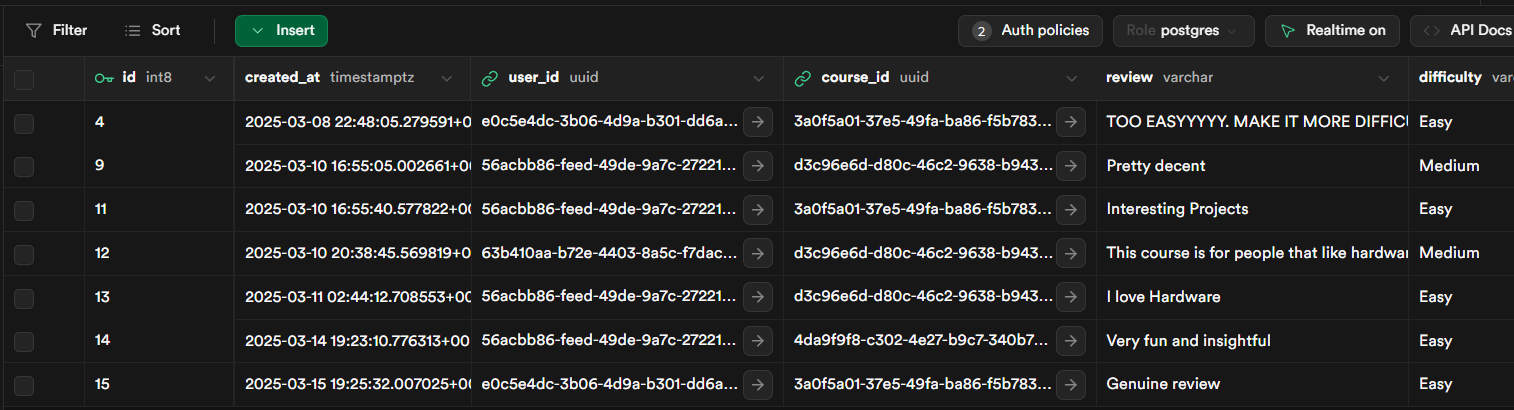
## Future Proofing

Every choice we have made has always emphasized scalability and sustainability. Ultimately, software changes rapidly, and we can only control what we can, however, we wanted to ensure that many core features would not require constant maintenance and tweaks. Our current implementation is rather well-built, but some areas could be much more optimized, and potential performance gains could be done by optimizing the procedure in which we calculate and do things. But currently, our web app will function, and can easily be scaled to meet the needs of all Brock students. We really only have to worry about entering accurate information into our database, and our codebase will handle the rest without much worry. There are very specific cases where our codebase will not handle it so gracefully (for example programs that are 6+ years long), however, this is not a standard case. And is very specific and most users should not worry. Since all of the calculations are performed in our code base, and the data required for these calculations, such as the projected graduation, prerequisites, and course suggestions, is all from information from our database, we must simply ensure that our data is up to date, accurate, and reliable. We currently have a script that we can set up on github action to automatically run once a year in the summer so that the courses will refresh their times, professors etc, to match the current school years scheduling. The only issue is that program requirements, prerequisites, and programs all require manual user entry. We simply could not find a reliable way to automate this, but given that these are so specific, and often do not change *much,* it may have been for the best. If there is demand, it does not take much time or effort to add a program, and the app will scale very easily to allow for users of this program to access the correct and relevant information. We have created a script that allows us to input program requirements via the command line, and this increased our efficiency in adding program requirements drastically, as it cut down the time/program from on average of 30 minutes to around ~10 minutes/program. Thus, the app is built to scale easily and meet the needs of **ALL Brock University students**, and with more data entries, we can easily accomplish coverage for the entire university.

## Supabase policy changes (Ola)

Updated the Supabase policy for the review page to align it with the front-end implementation:

* Public Access: Allows authenticated users to insert or modify only reviews associated with their account (auth.uid() = user\_id).
* Data Validation: Ensures user\_id and course\_id fields are not null, and that the difficulty field strictly matches one of the predefined options: 'Easy', 'Medium', or 'Hard'.

**Supabase snapshots**:  
 

# Release (Fatima)

## Scrum meet

* January 12: Finalized our project proposal
* 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
* 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 Progress report 2. Divided tasks.
* March 21: Went over progress.

## Third Sprint

We proved scalability potential by incorporating other programs to our site. This demonstrated our platform's capability to handle expanded offerings in the future. The previously noted bugs related to browser cookies have been resolved, ensuring a smoother user experience during session management.

We've also created and integrated our social media profiles directly to our website. This helps our marketing campaign and adds more legitimacy to our platform.

We have 1 remaining task, which is the degree audit.

## Currently Working On

We are aiming to complete the degree audit by next week. Most of the necessary information is already accessible on the website, however, we are working on developing a feature to allow users to generate a PDF of their degree audit This will allow students to easily share their academic progress, as well as access all the information needed on 1 page. Since currently students use a degree audit worksheet for their progress, including a similar worksheet would create a sense of familiarity to the students, hence a smoother transfer.

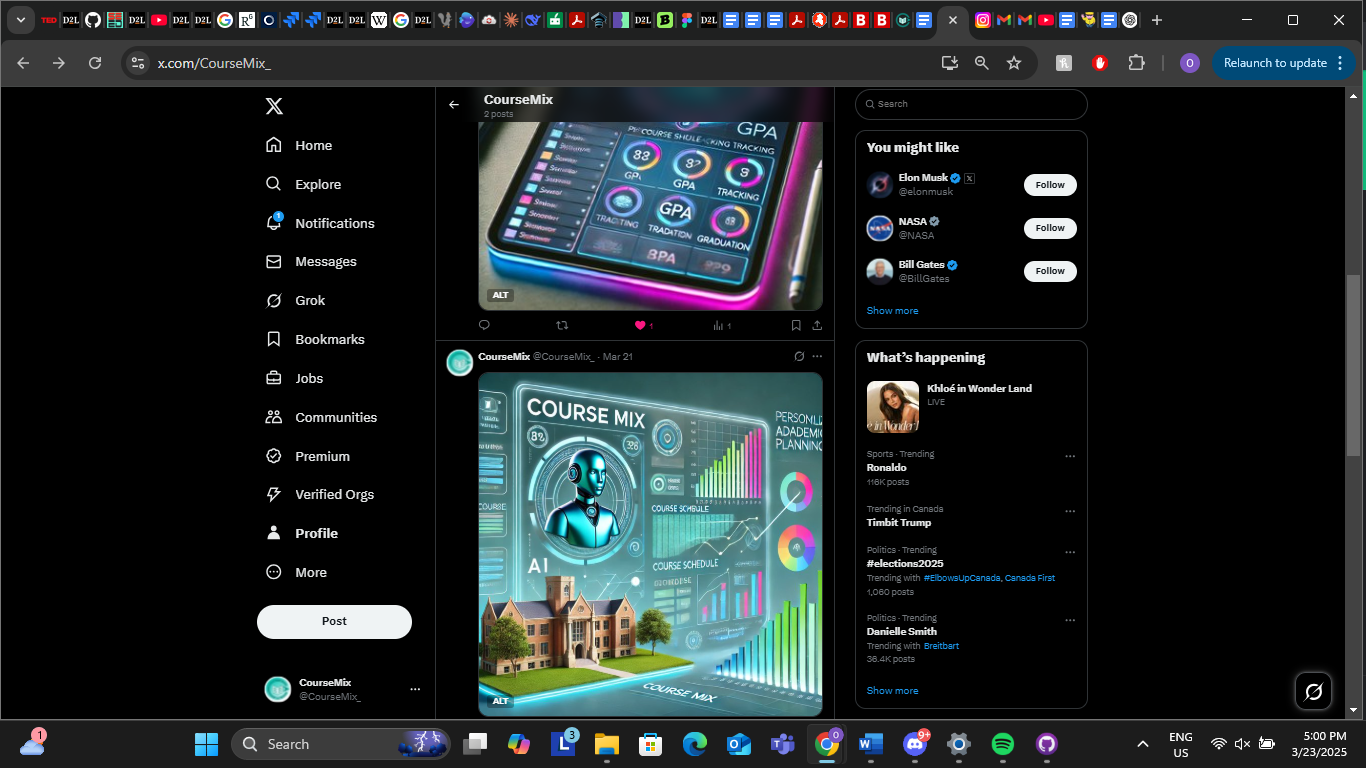
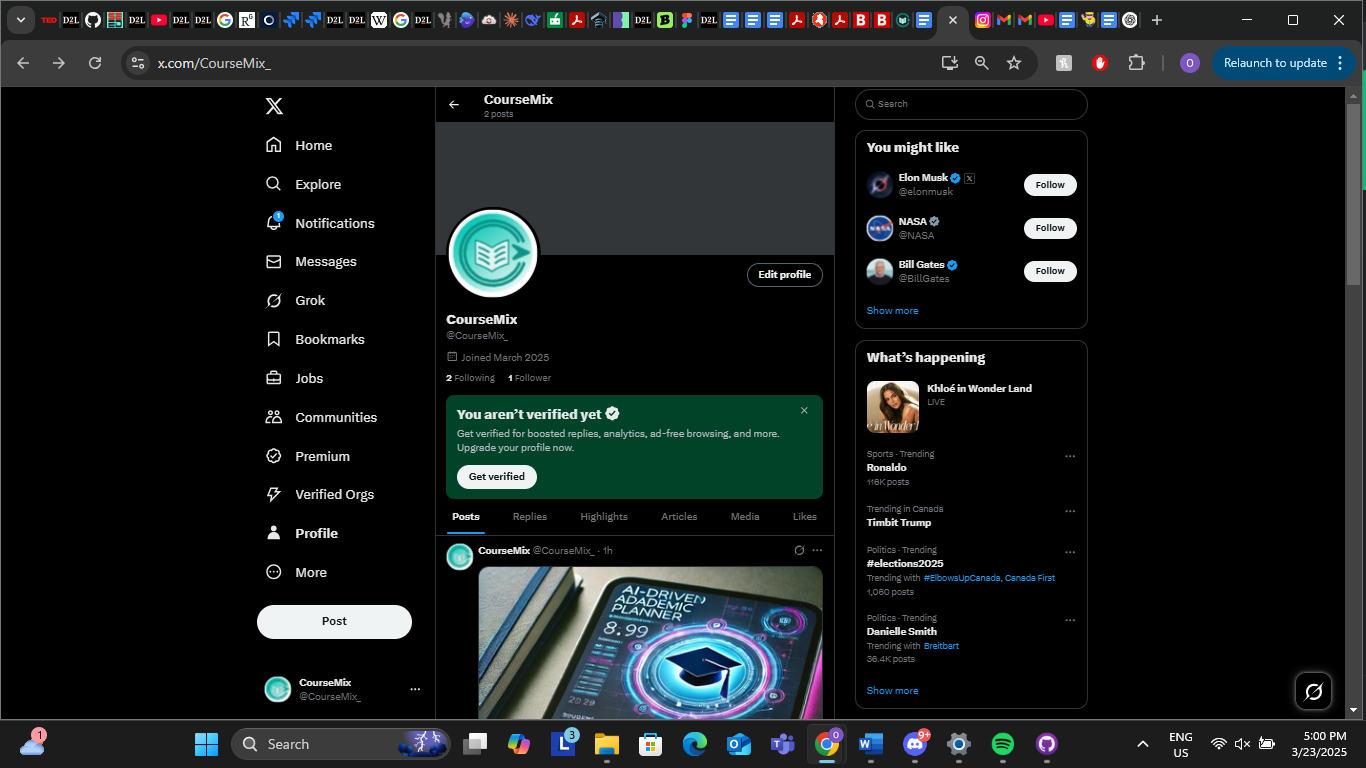
## Next Steps (Ola)

* + **Video Launch**: We are also planning to create a product launch video to showcase the capabilities and benefits of the Course Mix platform. It will spotlight the essential features and functionalities of the web app. This approach significantly benefits visual learners and users who prefer guided walkthroughs, substantially enhancing user experience, engagement and understanding. A script will be prepared, and an in-person team meeting will be scheduled for some Tuesday to film the necessary scenes. This video will aim to effectively communicate the value proposition of Course Mix to potential users and stakeholders.
  + **Large-scale Testing**:
    - **User Testing**: We plan to conduct extensive beta testing involving friends, family, and selected interested participants. Their valuable insights on ease of use, accessibility, areas for improvement, and potential bugs will ensure that Course Mix meets user expectations and delivers an optimal experience.
    - **Automated Testing**: We are currently designing and implementing comprehensive automated test cases, specifically targeting input-handling sections such as user forms, course discussions and reviews. This structured approach ensures the robustness, resilience, security and reliability of the application's core functionalities. Additionally, we are reviewing and refactoring redundant code within our Next.js framework and database structures to improve (overall) application performance, maintainability, and scalability

## Course Mix Social Media Content Strategy (Ore)

## 

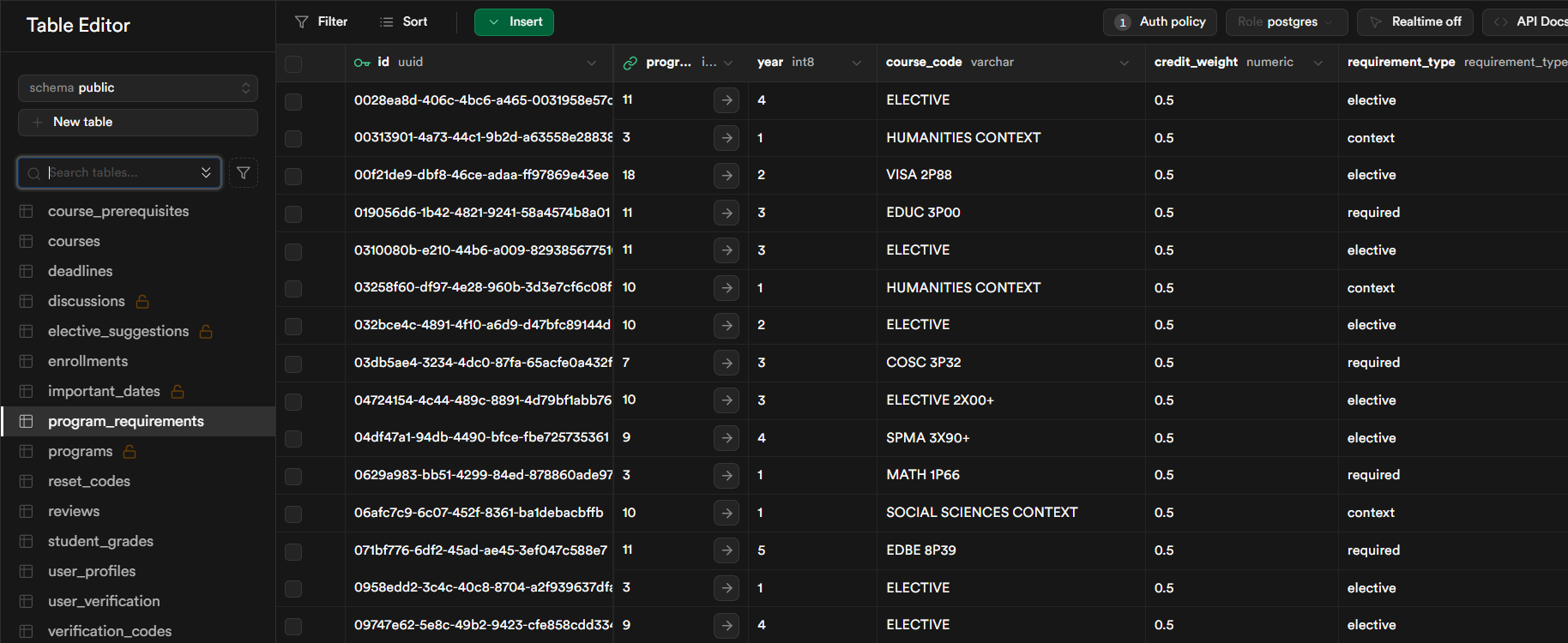
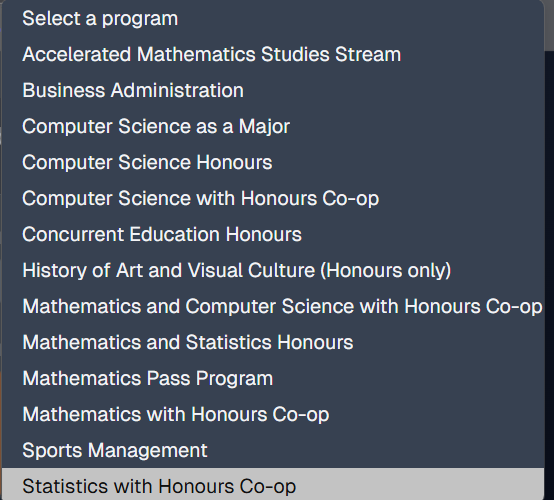
Instagram Screenshot

  
   
 X (Twitter) Screenshots

To increase awareness and engagement for Course Mix, we have initiated a targeted social media campaign on Instagram and X (formerly Twitter). Leveraging ChatGPT to aid in content creation, we have carefully crafted posts that highlight the app’s features, benefits, and unique AI-driven academic planning capabilities. Thus far, two posts have been published, and we are actively experimenting with different posting schedules to identify the optimal times for reach and engagement.

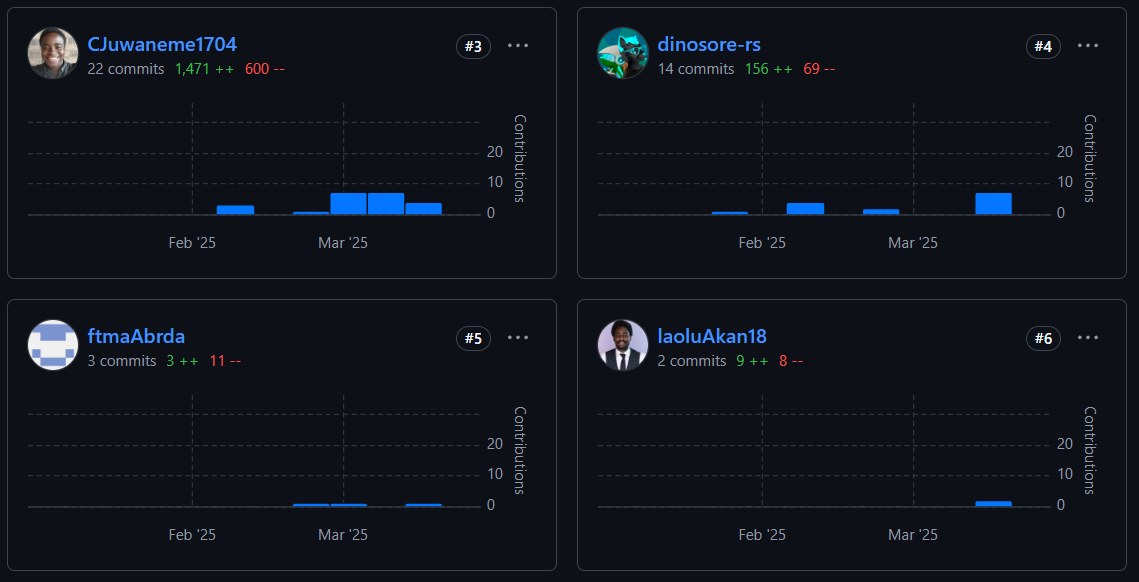
As part of this experimentation, we monitor post performance closely, tracking likes, comments, shares, and other relevant engagement metrics. By spacing out the posting times, we can better understand audience behavior and tailor our future content strategy accordingly. Moving forward, we plan to increase the frequency of posts, expand into new content formats (e.g., short videos), and maintain a steady presence on both Instagram and X. .

# Problems/Discussions (Ola)

* **Additional Program Requirements**: We discontinued this task because scalability was already demonstrated through the successful input of data for approximately 19 out of the 92 programs listed in Brock's undergraduate calendar. Since most Brock courses were already included, we prioritized our time on implementing other essential functionalities and conducting thorough testing. However, we can scale up further as required.  
    
  **Pertinent Screenshots**:  
  **Supabase**:  
    
    
  **Prod snapshot**:  
  
* **Push Notifications**: We decided against implementing notifications because Course Mix is a web application, making push notifications unfeasible. While email notifications were considered as an alternative, we ultimately concluded this feature was out of scope due to ethical concerns regarding spam.
* **Failure Rates and Policy Discussion**: We discontinued the failure rate analysis due to privacy considerations—grades are encrypted via a randomized key in the database to maintain user confidentiality, which prevents us from accessing or viewing this sensitive data. Instead, we developed a course review framework within Course Mix that effectively captures similar insights. This system includes a user-friendly filtering option, highlighting course difficulty rankings without compromising privacy.

# GitHub Log Activities (Ola)





# Team Contribution

| **Team Member** | **Contribution** |
| --- | --- |
| Everyone | Testing |
| Ashu Chauhan | Implementation of course suggestions/ graduation tracker/ prerequisite check |
| Avi Patel | Implementation of course suggestions/ graduation tracker/ prerequisite check |
| Fatima Abourida | Create and upload the meeting minutes to GitHub. Created GitHub issues for all the remaining tasks for the Sprints. Manually added programs to Supabase. Created a LinkedIn profile. Linked all the social media profiles to the website. |
| Jerome Uwaneme | Implementation of course review/discussions and filtration system |
| Olaoluwa Akanji | Implementation of course review/discussions and filtration system, Privacy policy refinement |
| Oreoluwa Akanji | Database design analysis and implementation |
| Russell Salacup | Design and Development for “Academic Progress” (formerly “Student Grades”) |

# 