University of British Columbia Okanagan

COSC 499 – 201

Capstone Software Engineering Project

2023 Summer Term 1&2

Project Topic Number: #6

Title of project:

Gamified Coding Practice Platform

Assignment: Final Report

August 15, 2023

Team Members:

Joe Gaspari [Project Manager/Scrum Master]

Archita Gattani [Technical Lead]

Gyumin Moon [Integration Lead]

Jason Ramos [Software Manager]

Alrick Vincent [Client Liaison]

1. Project Name

Gamified Coding Practice platform

2. Abstract

This document serves as a comprehensive guide for the final report of the project. Its objective is to provide a clear understanding of the project's purpose, objectives, outcome and handover instructions. The document outlines the project's outcome by defining the specific purpose, objectives, requirements, design and testing. Additionally, the document identifies the project's work breakdown structure, burn-up chart, reflection and handover guide that will be utilized by the client to take the project further. Overall, this document plays a crucial role to successfully end this project and be taken to the next level by the client.

3. Project Statement

The Gamified Coding Practice Application project, undertaken by Learnification Technologies, aims to help students learn and practice coding skills. By leveraging the power of modern technologies, personalization, and gamification, the application intends to create an engaging and efficient learning experience for students.

The primary objective of the project is to develop a dynamic web application that encourages students to practice coding by providing them with a gamified environment. The application will let students choose their preferred programming language to focus on, allowing them to tailor their learning experience according to their individual needs and interests.

To achieve this, the project will utilize React JS, a powerful JavaScript library. It will form the foundation of the application's front-end, ensuring a seamless and user-friendly interface for students. The core functionality of the Gamified Coding Practice Application will be built around Open AI’s APIs. It will generate coding questions tailored to the selected programming language.

The gamified nature of the application will enhance student engagement by incorporating elements such as badges, leaderboards, and rewards. By making coding practice enjoyable and interactive, the application will encourage students to dedicate more time and effort to their learning journey.

4. Project Purpose

To leverage the power of artificial intelligence (AI) in revolutionizing the way students learn to code. The project will create an interactive and immersive learning environment that empowers students to develop essential coding skills with ease and confidence. By harnessing the capabilities of AI, the project will provide personalized learning tailored to each student's unique needs, abilities and learning pace.

To enhance the efficiency and effectiveness of coding education by providing instructors with a comprehensive and customizable set of questions. The goal is to equip educators with the tools to seamlessly align coding assessments with their course curriculum, ensuring students are challenged appropriately while reinforcing key concepts. The project will harness AI to automate the process of generating coding questions, taking into account the desired learning outcomes, difficulty levels, and programming languages. The project aims to save instructors valuable time and effort by offering a diverse range of high-quality questions that cater to the specific needs of their students.

5. Project Goals and Objectives

Project Goals:

- Develop a dynamic, gamified web application.
- Encourage students to practice coding skills.
- Provide a personalized learning experience.
- Utilize AI-assisted learning for generating coding questions.
- Foster continuous learning and improvement among users.
- Make coding practice enjoyable and engaging through gamification elements.
- Enhance student motivation and engagement in coding.
- Promote collaboration and competition among students.
- Improve coding proficiency and skills among users.
- Contribute to the advancement of education technology.

Project Objectives:

- Design and develop a user-friendly and visually appealing web application.
- Implement a selection mechanism for students to choose programming language and topics.
- Integrate AI technologies to generate coding questions based on language and topics selected by students.

- Develop an assessment system using AI to evaluate code submissions and provide feedback.
- Incorporate gamification elements such as achievements, leaderboards, and rewards.
- Ensure the application is accessible on various devices and browsers.
- Test and refine the application for usability, performance, and reliability.
- Gather user feedback to continuously improve and enhance the application.
- Monitor and analyze user engagement and progress within the application.
- Promote the application among educational institutions and coding communities.
- Provide documentation and support resources for users and administrators.
- Maintain data privacy and security of user information.
- Collaborate with AI technology providers for efficient integration and updates.
- Evaluate the impact and effectiveness of the application on students' coding skills.
- Stay updated with emerging technologies and educational trends to enhance the application's features and capabilities.

6. Project Requirements

a. Non-functional Requirements

Must Have:

1. The system must pass an accessibility audit (WCAG 2.1).
2. The system must be responsive for all non-mobile devices.
3. The system must have a seamless and straightforward user interface that minimizes user clicks or actions to achieve the goal.
4. The system must be properly documented for future developers through written documents and in-line code documentation.
5. The system must have a guide for teachers and students on how to use the platform.
6. The developers must create test cases before writing or committing new additional features or components.
7. The developers must be comfortable using HTML, CSS, JS, React, NodeJS, and SQL.

Should have:

1. The system should not store personally identifiable information except email.
2. The AI-generated questions should have less than a 5% user report rate.
3. The system should allow instructors the ability to view the confidence score of each of the questions generated for the student.

Could have:

1. The system could provide users with a dark theme option.
2. The system could be responsive for mobile devices.

Will not have:

1. The system will not support an adjustable modularized layout design to allow users to personalize the layout of different UI components.

b. Technical Requirements

Must Have:

1. The system must be developed using ReactJS for the front-end.
2. The system must be developed using NodeJS for the back-end.
3. The system must update asynchronously.
4. The system must integrate with an AI API to generate practice questions.The system must be dockerized for deployment.
5. The system must securely store user passwords through salting or hashing.
6. The system must have server-side security for user inputs.
7. The system's database must be a relational database.
8. The system must have validation for all user-input fields.
9. The system must make use of stable, popular libraries instead of in-house solutions.
10. The system must include server-side logging that captures all database queries and events that may lead to a failure.

Should have:

1. The system should indicate to users any errors found within the system.
2. The system should have client-side security for user inputs.

c. User Requirements

Must have:

1. All users must be able to register and login to the system.

2. All users must have the ability to change their password and reset it through the email linked to their account.
3. Learner users must be able to choose their programming language and topic and level of proficiency.
4. Learner users must have an alias or username.
5. Learners must be able to view their streak while accessing the platform.
6. Leaders must be able to see a timer when their streak is about to expire via the platform and/or email notification.
7. Instructor users must be able to create a class and share a join code.
8. Instructor users must be able to create a set of topics and customize the parameters such as proficiency level and question format for the to-be-generated questions.
9. Instructors must be able to create and keep track of multiple courses at the same time.

Should have:

1. All users should be able to delete their accounts.
2. All users should be able to edit their profile information.
3. All users should be able to view other public user's achievements through their dashboard.
4. Learner users should be able to emphasize chosen achievements through chosen badges and courses.
5. Learner users could have the option to opt into or out of public leaderboards.
6. Instructor users should be able to archive courses after the term has ended.

Could have:

1. Learner users could have personalized and customizable profile cards to emphasize favorite achievements.
2. Learner users could receive notifications about their progress and streak reminders.
3. Learner users could be alerted when class lessons are updated.
4. Learner users could be alerted about their course progress and streaks via notifications (either via email or via the web application).

d. Functional Requirements

Must Have:

1. The system must be accessible through a publicly available online domain.

2. The system must be able to accept user feedback on questions and reward feedback with bonus points and badges, which prompts system improvements.
3. The system must be able to prompt AI-generated questions based on a specified topic, level of proficiency, and programming language chosen by the user.
4. The system must be able to track and display user progress through course milestones, user points, achievement badges, and usage analytics.
5. The system must allow instructors the ability to view student and class analytics including students' progress in the assigned questions, how many they have completed, how many they have left, percent success rate, visual dashboard of progress, number of questions they have completed, their streak/points and the number of badges earned.
6. The system must support user login, logout, registration, and password retrieval via email linked to the account via email linked to the account.
7. The system must be able to indicate whether user-provided answers are correct or incorrect and provide meaningful AI-generated feedback with high confidence e.g. 95%.
8. The system must be able to validate whether an AI-generated question relates to the topic selected with a high degree of confidence (95%)
9. The system must support JavaScript, Python, and HTML/CSS.
10. The system must support the main user archetype: learners.
11. The system must include tooltips to guide users.
12. The system must be visually appealing to the users with a modern design.
13. The system should be able to give users AI-generated suggestions or hints when prompted by the user or a series of incorrect answers.
14. The system should have a public leaderboard categorized into topics and programming languages. This can be turned off, and the scores will be calculated by correct answers (weighted by question difficulty), milestones, streaks and badges.
15. The system should contain user dashboards for students showing course progress, badges earned, and other user achievements.
16. The system should support the main user archetype: Instructors.
17. The system should further support Java, C, C#, and C++ .
18. The system should have a class platform for instructors to create a set of topics and customize the parameters such as proficiency level and question format for the to-be-generated questions.
19. The system should allow learners to join classes through a join code similar to Kahoot platform.

Should have:

1. The system should easily accommodate additional programming languages for students to practice.

2. The system should have an FAQ page that includes questions/answers that may be popular.
3. The system should include terms of service and privacy policy information.

Could have:

1. The system could have an embedded IDE supporting autocorrection, linting, and semantic highlighting (which may give local variables distinct colors to improve code comprehensibility and structure).
2. The system could have an additional user archetype: administrators
3. The system could have an admin portal to view site usage analytics and (update/delete/add) users and classes.

7. Assumptions and constraints

Assumptions:

1. All potential users will have access to a stable internet connection to use the web-platform.
2. Users may have little to no knowledge of programming concepts prior to using the web-platform
3. The selected AI API will be running and fully functional throughout the project.
4. The developers will be able to program in the selected technology stack (React, NodeJS, and SQL).
5. Future expansion or maintenance of the system will not significantly affect the architecture of the system.
6. All stakeholders will be available at all times.
7. The main requirements for the project will be completed within the specified timeframe.

Constraints:

1. The project is under severe time constraints.
2. Some members of the development team will be away for disclosed amounts of time during the development time frame.
3. The project's front-end must be developed using React and associated libraries.
4. Limited user testing might cause disparities between client and user needs.
5. Budget constraints may limit the scope of the project, as well as the AI capabilities.
6. The development environment must be standardized in docker containers.
7. Depending on the AI API choice, requests might be limited.

8. Unit and integration tests must be passed before merging into the master branch.

8. Summary Budget

Technologies	Cost/Month	Total Cost	Actual Cost
Digital Ocean Hosting (Droplets)	$6	$18	$24
Open AI GPT-3.5 turbo ($0.002/750 words)	$20	$60	$6.54
Open AI GPT-4 ($0.03/750 words)	$200	$600	$0

9. Design

a. Actors

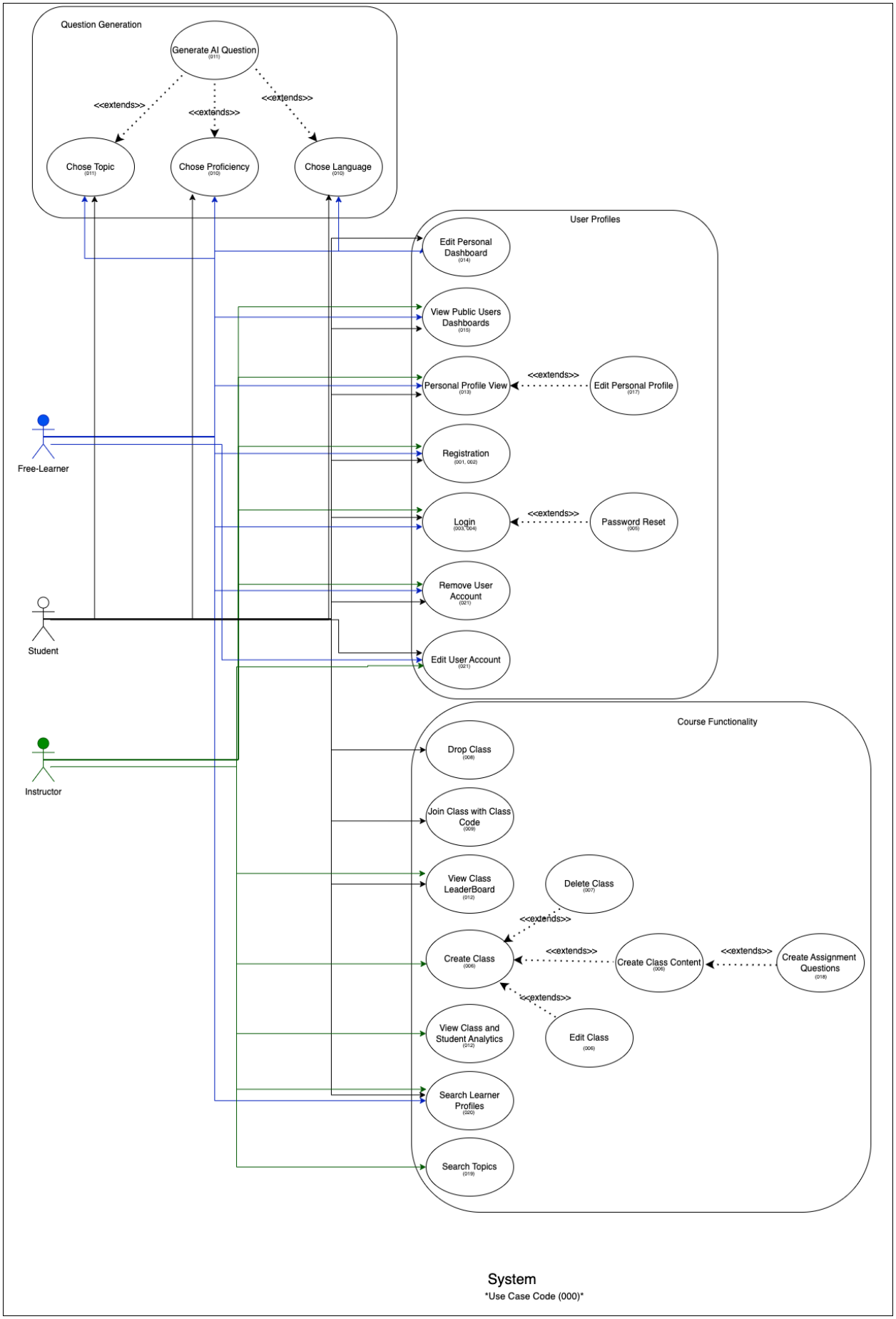
Self learners or free users are able to generate questions on any topic, language or proficiency level that they wish. They can complete streaks, gain points and view their statistics all at their own pace. Free learners will be able to become student learners if they are given a course code to register into a class.

Student users differ from self - learners in that a student user can join a class via a class code. Class codes will be generated by an instructor to allow students the ability to work on question sets that directly relate to the course content outlined by the professor. Student users start as free users, but can always return to a free learning state whenever they choose.

Instructors will differ the most from the two user groups above in that they will not be completing questions. Rather instructors will be allowed to generate classes within their main dashboards. They can generate class codes that can be sent to students to allow them to join classes. Instructors will be able to create assignments and have AI generate them a set of questions that directly relates to their course curriculum. Instructors will be able to view class analytics and more specifically individual student progress through their respective courses.

Administrators will not have direct access to the question generation process. Admin users are responsible for maintaining the platform as a whole. They will be able to generate institution codes that can be given to instructors to allow them the ability to register as an instructor. Administrators will be allowed to view the instructors and classes within an institution, while being able to maintain multiple institutions at one time.

b. Use-case descriptions & Scenarios


Question Generation
Generate AI Question
(011)
<<extends>>
<<extends>>
<<extends>>
Chose Topic
(011)
Chose Proficiency
(010)
Chose Language
(010)
User Profiles
Edit Personal Dashboard
(014)
View Public Users Dashboards
(015)
Personal Profile View
(013)
<<extends>>
Edit Personal Profile
(017)
Registration
(001, 002)
Login
(003, 004)
<<extends>>
Password Reset
(005)
Remove User Account
(021)
Edit User Account
(021)
Free-Learner
Student
Instructor
Course Functionality
Drop Class
(008)
Join Class with Class Code
(009)
View Class LeaderBoard
(012)
Delete Class
(007)
<<extends>>
Create Class
(006)
<<extends>>
Create Class Content
(006)
<<extends>>
Create Assignment Questions
(018)
<<extends>>
Edit Class
(006)
View Class and Student Analytics
(012)
Search Learner Profiles
(020)
Search Topics
(019)
System
Use Case Code (000)

(001) Student / Free Learner Registration:

Primary actor: Student

Preconditions: None

Postconditions: Student successfully registers an account in the system.

Main Scenario:

1. Student accesses the registration page.
2. Student provides the required information, such as name, email address, and password.
3. Student submits the registration form.
4. The system validates the provided information.
5. If the information is valid, the system creates a new student account.
6. Student is redirected to the login page and can now log in with their registered credentials.

Extensions:

- If the provided information is invalid or incomplete, the system displays an error message and prompts the student to provide valid information.

(002) Instructor Registration:

Primary actor: Instructor

Preconditions: None

Postconditions: Instructor successfully registers an account in the system.

Main Scenario:

1. Instructor accesses the registration page.
2. Instructor provides the required information, such as name, email address, and password.
3. Instructor submits the registration form.
4. The system validates the provided information.
5. If the information is valid, the system creates a new instructor account.

6. Instructors are redirected to the login page and can now log in with their registered credentials.

Extensions:

- If the provided information is invalid or incomplete, the system displays an error message and prompts the instructor to provide valid information.

(003) Student / Free Learner Login:

Primary actor: Student

Preconditions: Student has a registered account in the system.

Postconditions: Student successfully logs into the system.

Main Scenario:

1. Student accesses the login page.
2. Student enters their email address and password.
3. Student submits the login form.
4. The system validates the provided credentials.
5. If the credentials are valid, the system authenticates the student and grants access to their account.
6. Student is redirected to their personalized dashboard or the main application interface.

Extensions:

- If the provided credentials are incorrect or do not match any registered accounts, the system displays an error message and prompts the student to enter valid credentials.
- If the student's account is not yet activated, the system displays a message indicating that the account needs to be verified.

(004) Instructor Login:

Primary actor: Instructor

Preconditions: Instructor has a registered account in the system.

Postconditions: Instructor successfully logs into the system.

Main Scenario:

1. Instructor accesses the login page.
2. Instructor enters their email address and password.
3. Instructor submits the login form.
4. The system validates the provided credentials.
5. If the credentials are valid, the system authenticates the instructor and grants access to their account.
6. Instructor is redirected to their dashboard.

Extensions:

- If the provided credentials are incorrect or do not match any registered accounts, the system displays an error message and prompts the instructor to enter valid credentials.
- If the instructor's account is not yet activated, the system displays a message indicating that the account needs to be verified.

(005) Reset Password:

Primary actor: User/Student/Instructor

Preconditions: User has a registered account in the system.

Postconditions: User's password is successfully reset.

Main Scenario:

1. User accesses the "Forgot Password" page.
2. User enters their registered email address.
3. User submits the form to request a password reset.
4. The system verifies the provided email address.
5. If the email address is valid, the system sends a password reset link to the user's email.
6. User receives the password reset email and clicks on the reset link.
7. User is redirected to a password reset page.
8. User enters a new password and confirms it.
9. User submits the form to reset the password.
10. The system validates and updates the user's password.

11. User receives a confirmation message that their password has been successfully reset.
12. User can now log in with the new password.

Extensions:

- If the provided email address is not found in the system, the system displays an error message indicating that the email address is not registered.
- If the new password does not meet the system's password requirements, the system displays an error message and prompts the user to choose a valid password.

(006) Instructor Create/Manage Class Content:

Primary actor: Instructor

Preconditions: Instructor is logged into their account and has access to their dashboard.

Postconditions: Class content (questions, topics, students) is created or managed by the instructor.

Main Scenario:

1. Instructor accesses their dashboard.
2. Instructor selects the option to create new class content.
3. Instructor provides the necessary details, such as class name, topic, and questions.
4. Instructor submits the form to create the class content.
5. The system validates the provided information.
6. If the information is valid, the system creates the class content and associates it with the instructor's class.
7. Instructor can view and manage the created class content, such as adding or editing questions, topics, or students.

Extensions:

- If the provided information is incomplete or invalid, the system displays an error message and prompts the instructor to provide valid information.
- If there are any errors during the creation of class content, the system displays an error message and informs the instructor about the issue.

(007) Instructors Delete Class:

Primary actor: Instructor

Preconditions: Instructor is logged into their account and has access to their dashboard.

Postconditions: The specified class is successfully deleted from the system.

Main Scenario:

1. Instructor accesses their dashboard.
2. Instructor selects the option to delete a class.
3. Instructor chooses the class they want to delete from the available list.
4. Instructor confirms the action.
5. The system removes the specified class from the system and associated data, such as class content and enrolled students.
6. Instructor receives a confirmation message that the class has been successfully deleted.

Extensions:

- If the instructor does not have the necessary permissions to delete the class, the system displays an error message indicating the lack of authorization.

(008) Student Joining a Class Using Code:

Primary actor: Student

Preconditions: Student is logged into their account and has access to the class enrollment interface.

Postconditions: Student successfully joins the specified class using the provided class code.

Main Scenario:

1. Student accesses the class enrollment interface.
2. Student selects the option to join a class using a code.
3. Student enters the class code provided by the instructor.
4. Student submits the code.
5. The system validates the entered code.

6. If the code is valid and the class is open for enrollment, the system adds the student to the class.
7. Student receives a confirmation message that they have successfully joined the class.

Extensions:

- If the entered code is invalid or does not correspond to any available classes, the system displays an error message indicating an invalid code.
- If the class is not open for enrollment or the maximum capacity has been reached, the system displays a message indicating that the student cannot join the class at the moment.

(009) Learners Choosing a Language and Proficiency:

Primary actor: Student

Preconditions: Student is logged into their account and has access to language and proficiency selection options.

Postconditions: Student's chosen language and proficiency level are saved in their profile.

Main Scenario:

1. Student accesses the language and proficiency selection options.
2. Student chooses a programming language from the available options.
3. Student selects their proficiency level in the chosen language.
4. Student submits the language and proficiency selection.
5. The system saves the chosen language and proficiency level in the student's profile.
6. Student receives a confirmation message that their language and proficiency selection has been successfully saved.

Extensions:

- If the chosen programming language is not available in the system, the system displays an error message indicating an invalid language selection.
- If the chosen proficiency level is not within the valid range, the system displays an error message indicating an invalid proficiency selection.

(010) Learners Choosing a Topic and Getting an AI-Generated Question:

Primary actor: Student

Preconditions: Student is logged into their account and has access to topic selection and question generation options.

Postconditions: Student receives an AI-generated question based on their chosen topic.

Main Scenario:

1. Student accesses the topic selection options.
2. Student chooses a topic of interest from the available options.
3. Student submits the topic selection.
4. The system generates an AI-generated question based on the chosen topic.
5. The system presents the generated question to the student.
6. Student receives the AI-generated question for their chosen topic.

Extensions:

- If the chosen topic is not available in the system, the system displays an error message indicating an invalid topic selection.
- If there are any issues with the AI question generation process, the system displays an error message indicating the inability to generate a question.

(011) Student and Instructor View Class Leaderboard:

Primary actor: Student/Instructor

Preconditions: Student or Instructor is logged into their respective accounts and has access to the class leaderboard.

Postconditions: Student or Instructor successfully views the class leaderboard with relevant rankings.

Main Scenario:

1. Student/Instructor navigates to the class leaderboard section.
2. The system retrieves the necessary data to generate the class leaderboard.

3. The system ranks the students based on their performance metrics, such as points or completion rate.
4. The system displays the class leaderboard to the Student/Instructor, showing the rankings of students in the class.
5. Student/Instructor can view their own position on the leaderboard and the positions of other students.

Extensions:

- If there are no students enrolled in the class or no performance metrics available, the system displays a message indicating the absence of leaderboard data.

(012) View Own User Profile:

Primary actor: Student/Instructor

Preconditions: Student/Instructor is logged into their respective accounts and has access to their user profile.

Postconditions: Student/Instructor successfully views their own user profile information.

Main Scenario:

1. Student/Instructor accesses the user profile section.
2. The system retrieves the user profile data for the Student/Instructor.
3. The system displays the user profile information, including personal details, preferences, and achievements.
4. Student/Instructor can view and review their own user profile.

Extensions:

- If there is no user profile data available for the Student or Instructor, the system displays a message indicating the absence of profile information.
- If there are any restrictions or permissions applied to the user profile, the system displays the relevant information based on the Student or Instructor's access level.

(013) Instructor Can View Class and User Analytics:

Primary actor: Instructor

Preconditions: Instructor is logged into their account and has access to class and user analytics.

Postconditions: Instructor successfully views the analytics related to classes and users.

Main Scenario:

1. Instructor accesses the class and user analytics section.
2. Instructor selects the specific class or user for which they want to view analytics.
3. The system retrieves the necessary data and generates analytics based on the selected class or user.
4. The system presents the analytics to the Instructor, displaying relevant information such as performance metrics, progress, or engagement levels.
5. Instructor can analyze and interpret the analytics data to gain insights into class or user performance.

Extensions:

- If there are no available analytics data for the selected class or user, the system displays a message indicating the absence of analytics information.

(014) Edit User Profile:

Primary actor: Student or Instructor

Preconditions: Student or Instructor is logged into their respective accounts and has access to edit their user profile.

Postconditions: Student or Instructor successfully edits their own user profile by modifying personal information, preferences, or settings.

Main Scenario:

1. Student or Instructor accesses their user profile section.
2. The system presents the current user profile information and settings.
3. Student or Instructor selects the option to edit their user profile.
4. Student or Instructor makes changes to the personal information, preferences, or settings as desired.

5. Student or Instructor saves the changes made to their user profile.
6. The system validates and updates the user profile with the edited information and settings.
7. Student or Instructor receives a confirmation message that their user profile has been successfully edited.

Extensions:

- If there are any restrictions or permissions applied to editing the user profile, the system displays an error message indicating the lack of authorization.
- If there are limitations on the allowed modifications to certain fields or settings, the system displays a message indicating the available editing options.

(015) Instructor Generating Question:

Primary actor: Instructor

Preconditions: Instructor is logged into their account and has access to the question generation functionality.

Postconditions: Instructor successfully generates a coding question for a specific topic or programming language.

Main Scenario:

1. Instructor accesses the question generation interface.
2. Instructor selects the desired topic or programming language for the coding question.
3. Instructor specifies any additional parameters or requirements for the question, such as difficulty level or constraints.
4. Instructor initiates the question generation process.
5. The system utilizes the AI-based question generation algorithm to generate a coding question based on the specified parameters.
6. The system presents the generated question to the Instructor, displaying the problem statement, input/output requirements, and any additional instructions.
7. Instructor reviews the generated question and makes any necessary edits or modifications.

8. Instructor finalizes the question and saves it to the question bank or assigns it to a specific class or student.

Extensions:

- If the desired topic or programming language is not available in the system, the system displays an error message indicating an invalid selection.
- If there are any issues with the question generation process, such as generating an incomplete or invalid question, the system displays an error message indicating the inability to generate the question.

(016) Search Topics:

Primary actor: Student/Instructor

Preconditions: Student/Instructor is logged into their respective accounts and has access to the topic search functionality.

Postconditions: Student/Instructor successfully searches for specific coding topics.

Main Scenario:

1. Student/Instructor accesses the topic search interface.
2. Student/Instructor enters the desired keyword or topic in the search bar.
3. Student/Instructor initiates the search.
4. The system performs a search based on the entered keyword or topic, filtering the available coding topics.
5. The system presents the search results, displaying a list of coding topics that match or are related to the entered keyword.
6. Student/Instructor can browse and select the desired coding topic from the search results.

Extensions:

- If there are no coding topics available in the system or no matching results for the entered keyword, the system displays a message indicating the absence of search results.

(017) Search Learner Profiles:

Primary actor: Instructor

Preconditions: Instructor is logged into their account and has access to the learner profile search functionality.

Postconditions: Instructor successfully searches for and views learner profiles based on specific criteria or filters.

Main Scenario:

1. Instructor accesses the learner profile search interface.
2. Instructor specifies the desired search criteria or filters, such as programming language proficiency or completion rate.
3. Instructor initiates the search.
4. The system performs a search based on the specified criteria, filtering the available learner profiles.
5. The system presents the search results, displaying a list of learner profiles that match the specified criteria.
6. Instructor can browse and select a specific learner profile from the search results.
7. The system displays the selected learner profile, showing relevant information such as programming language proficiency, completion rate, or achievements.

Extensions:

- If there are no learner profiles available in the system or no matching results for the specified criteria, the system displays a message indicating the absence of search results.

c. Diagrams

i. DFD

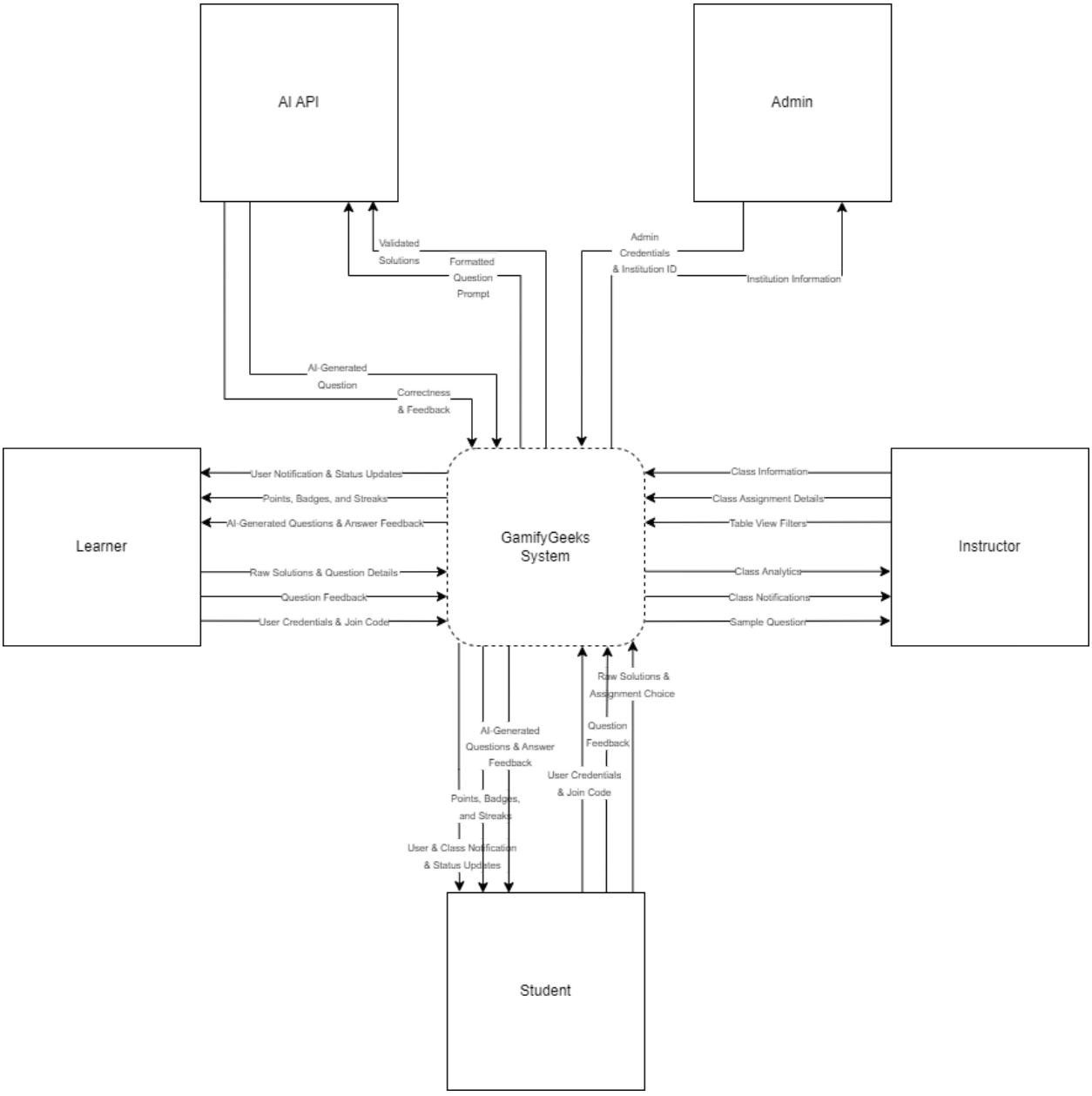


Figure 9.1: Data Flow Diagram Level 0

This **level 0** data flow diagram shows the basic overview of the whole system as a single boundary. Additionally, this illustrates the core pieces of data being exchanged between the system and 5 external entities being 4 user groups and 1 API.

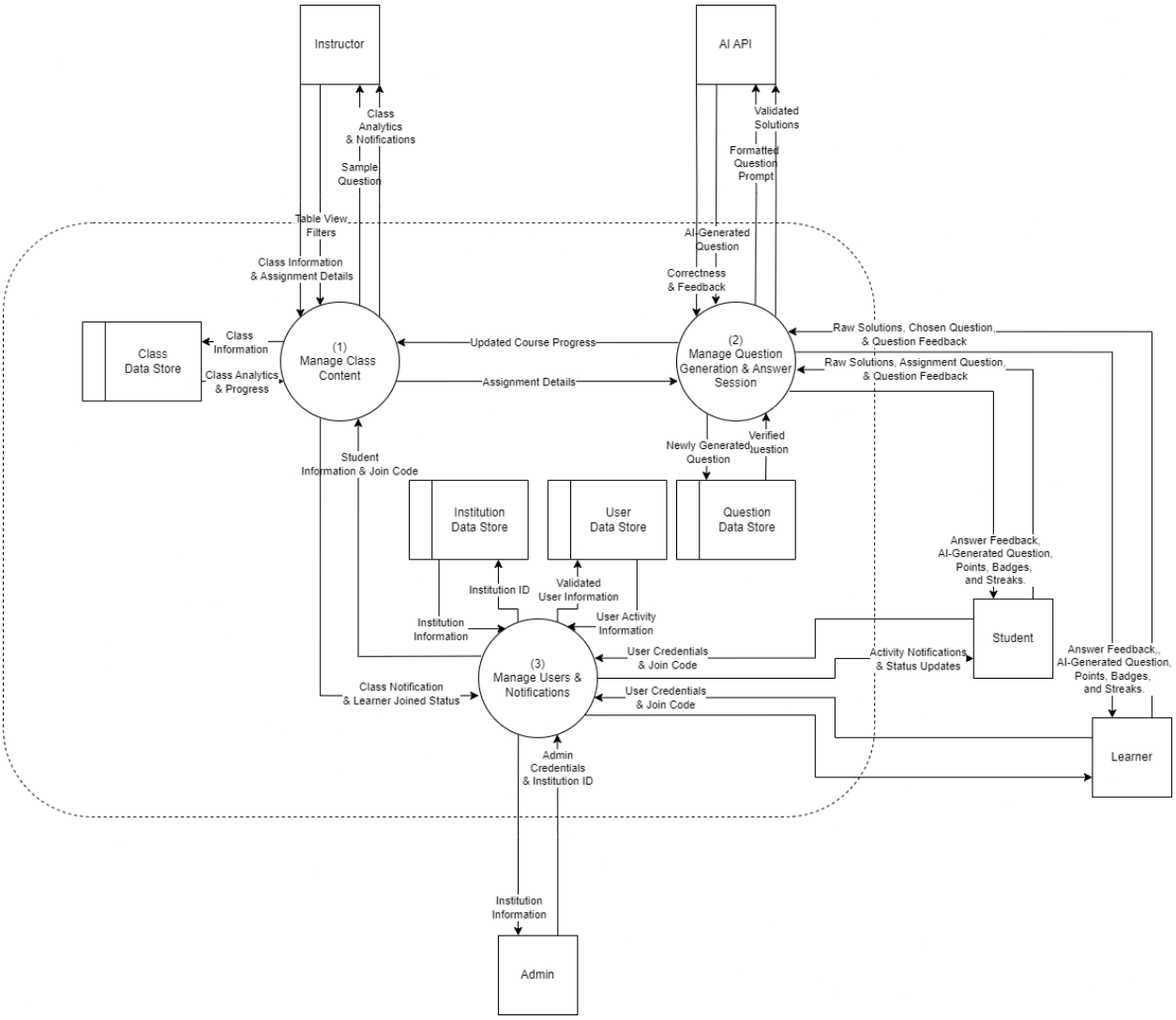


Figure 9.2: Data Flow Diagram Level 1

This **level 1** data flow diagram shows the more detailed breakdown of the system from the level 0 diagram into the 3 core processes of the system. In addition to this, the diagram also shows the relationship between those processes and the appropriate data stores.

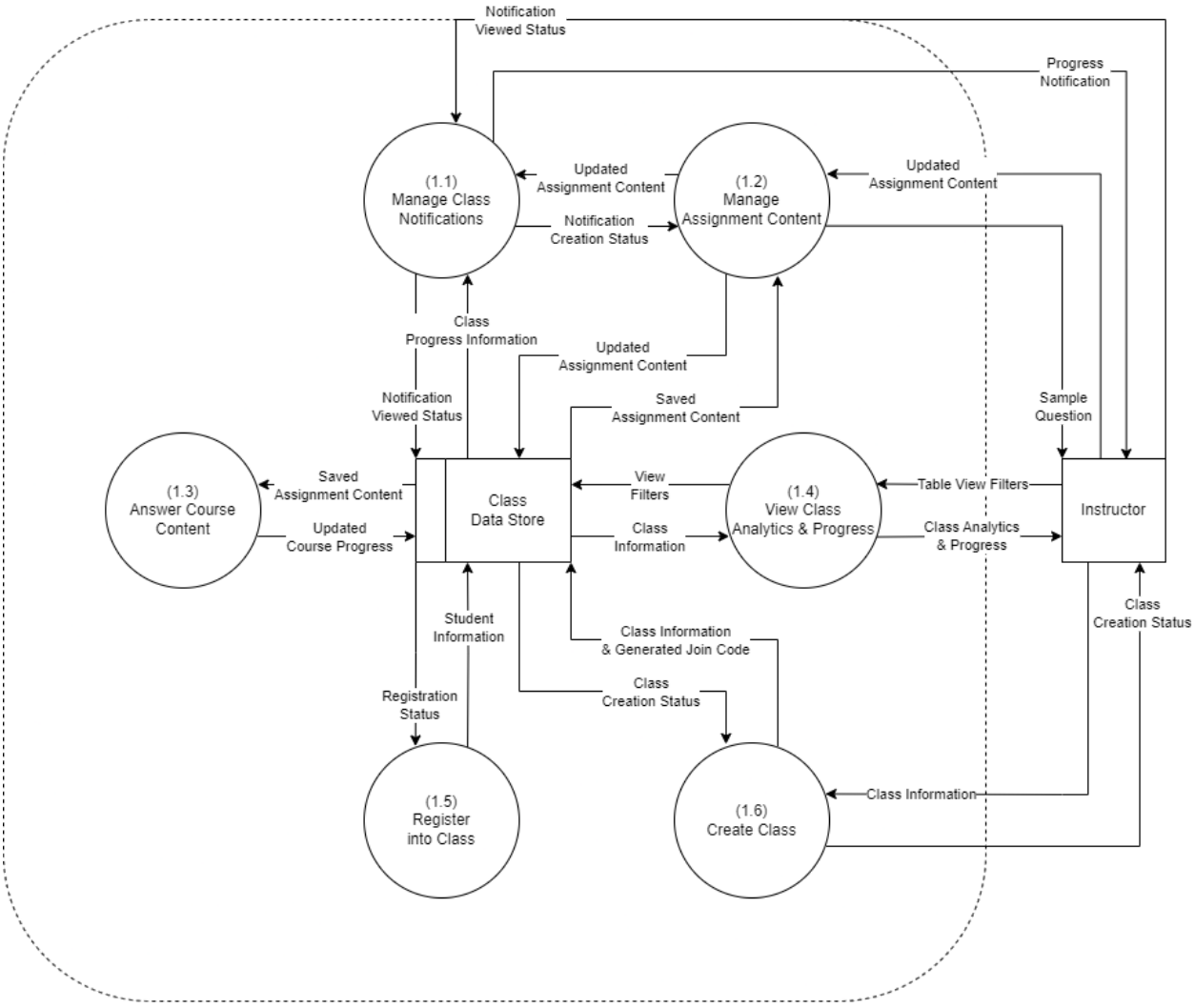


Figure 9.3: Data Flow Diagram Level 2 - Class Process

The first **level 2** data flow diagram expands on process (1) “Manage Class Content” and establishes its sub-processes.

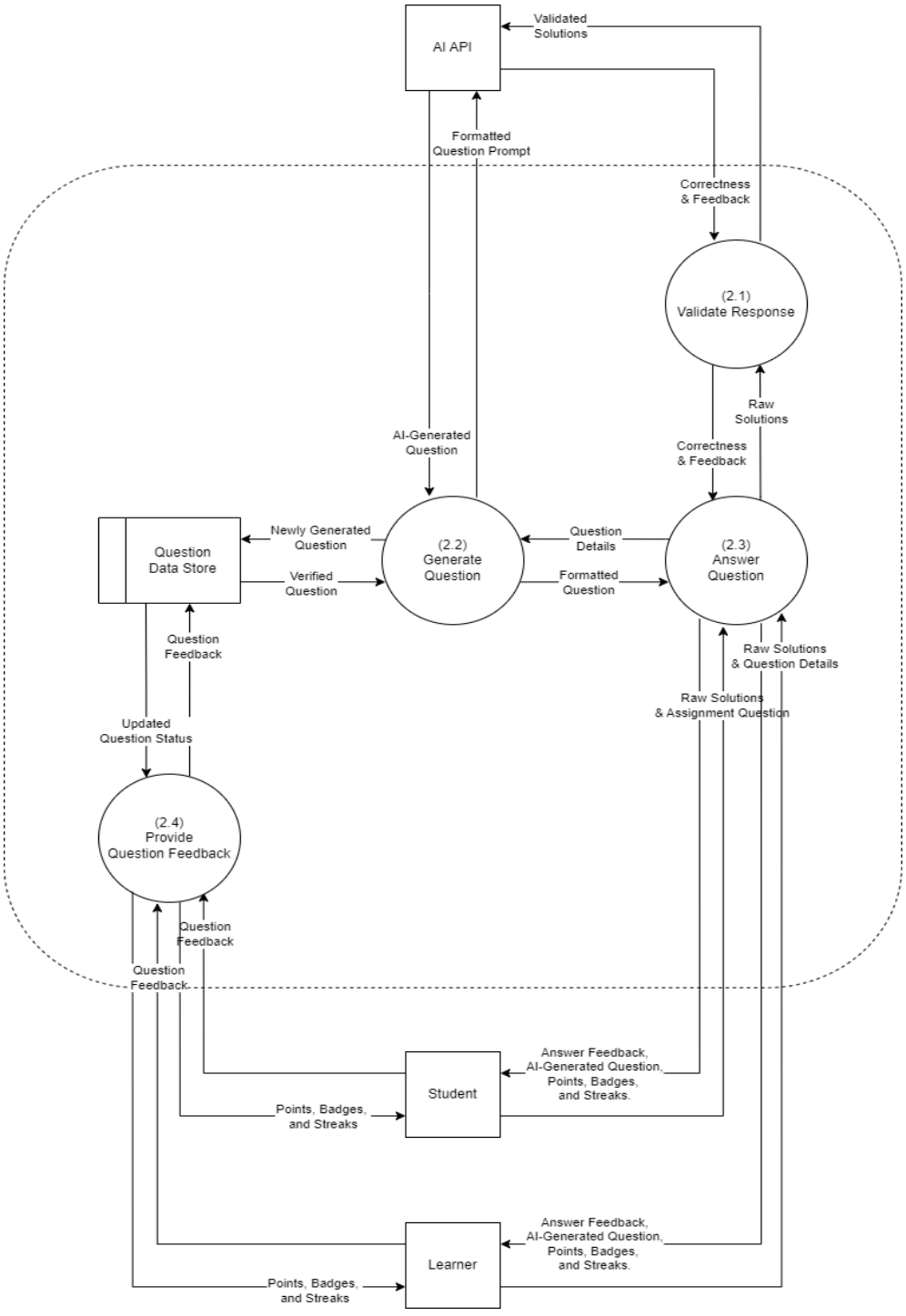


Figure 9.4: Data Flow Diagram Level 2 - Question Process

The second **level 2** data flow diagram expands on process (2) “Manage Question Generation & Answer Session” and establishes its sub-processes.

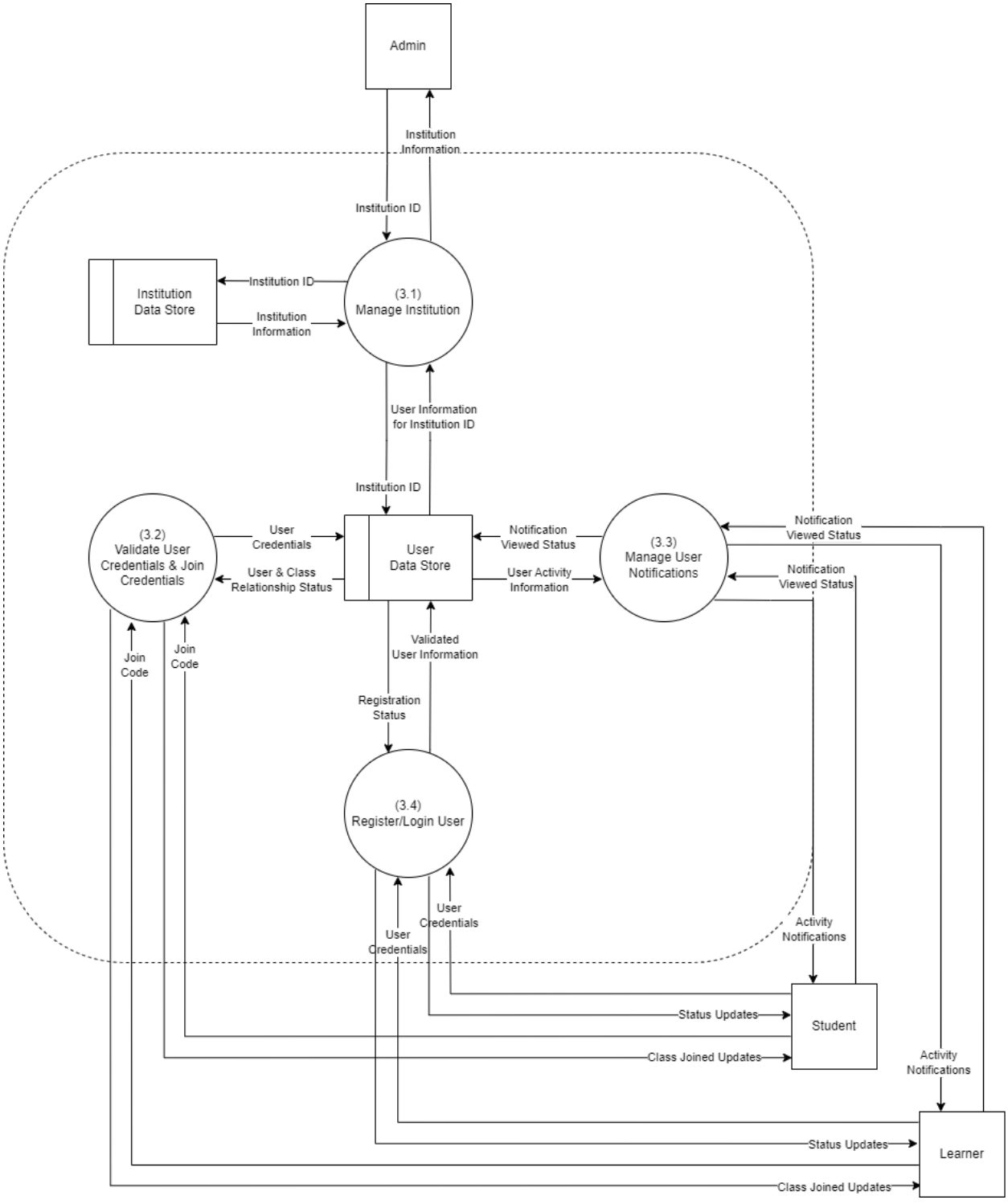


Figure 9.5: Data Flow Diagram Level 2 - User Process

The third **level 2** data flow diagram expands on process (3) “Manage Users & Notifications” and establishes its sub-processes.

ii. ER Diagram

This Entity-Relation (ER) diagram visually captures the structural design of the GamifyGeeks system, it's entities, and their relationships. The diagram facilitates a comprehensive understanding of the database's organiziation and interconnection, laying a strong foundation for efficient data management and informed decision making.

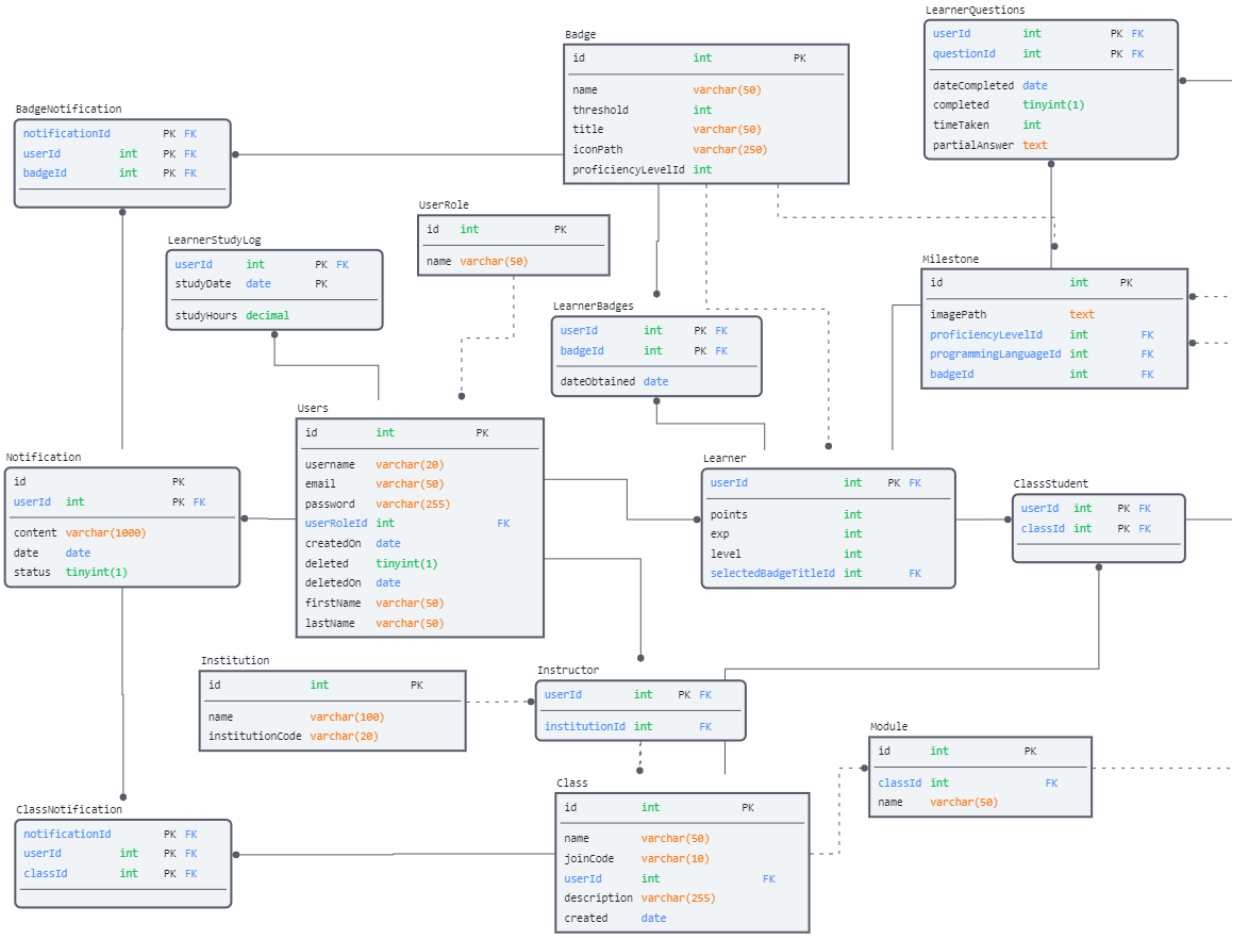


Figure 9.6: Section 1 of ER Diagram for GamifyGeeks

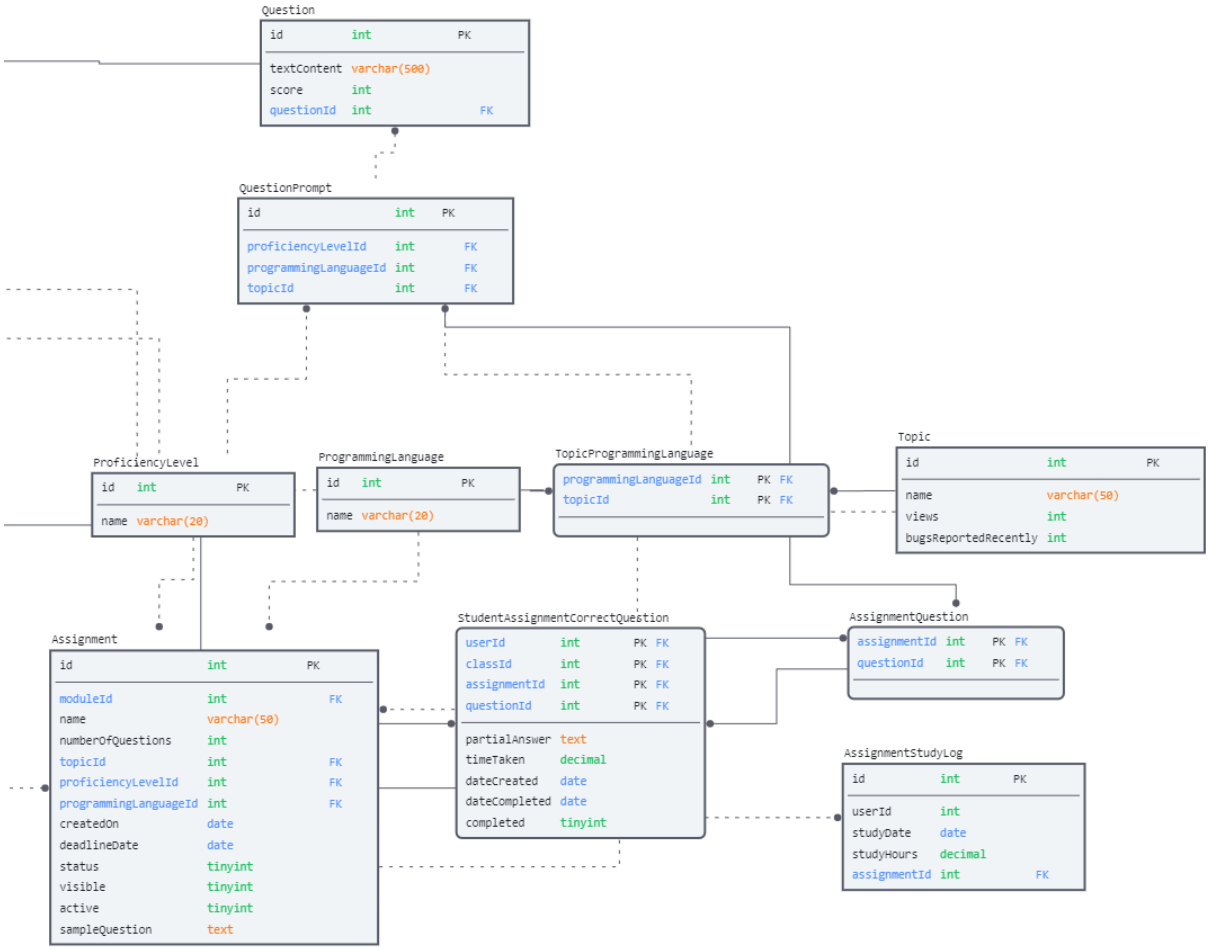


Figure 9.7: Section 2 of ER Diagram for GamifyGeeks

iii. Architecture Diagram

Our system follows a MVVM architecture pattern as it provides a set of predefined subsystems as well as their responsibilities. The MVVM pattern has four main components: the View, the View Controller, the View Model, and the Model depicted in the following diagram.

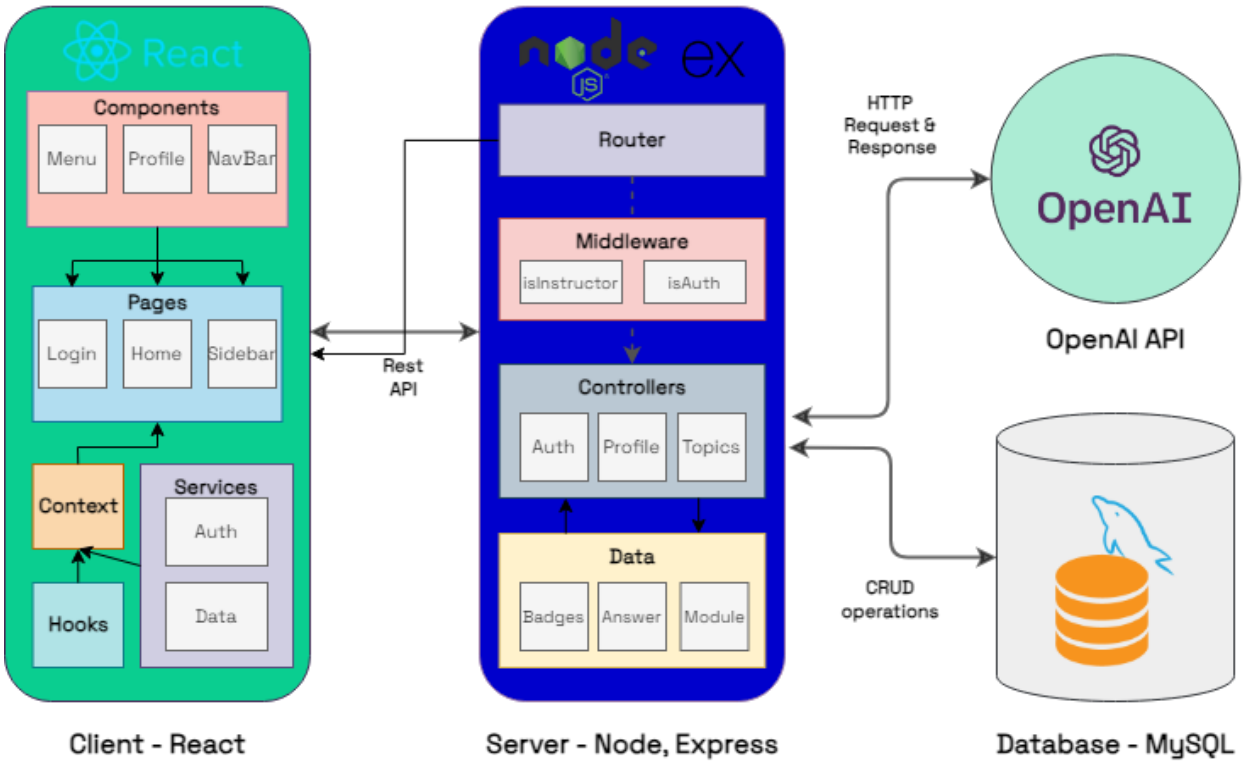


Figure 9.8: Architecture diagram for GamifyGeeks

iv. State Diagram

The state diagram shows how the user/instructor/student will go through the website. It shows the flow of events that will take place when someone accesses the website.

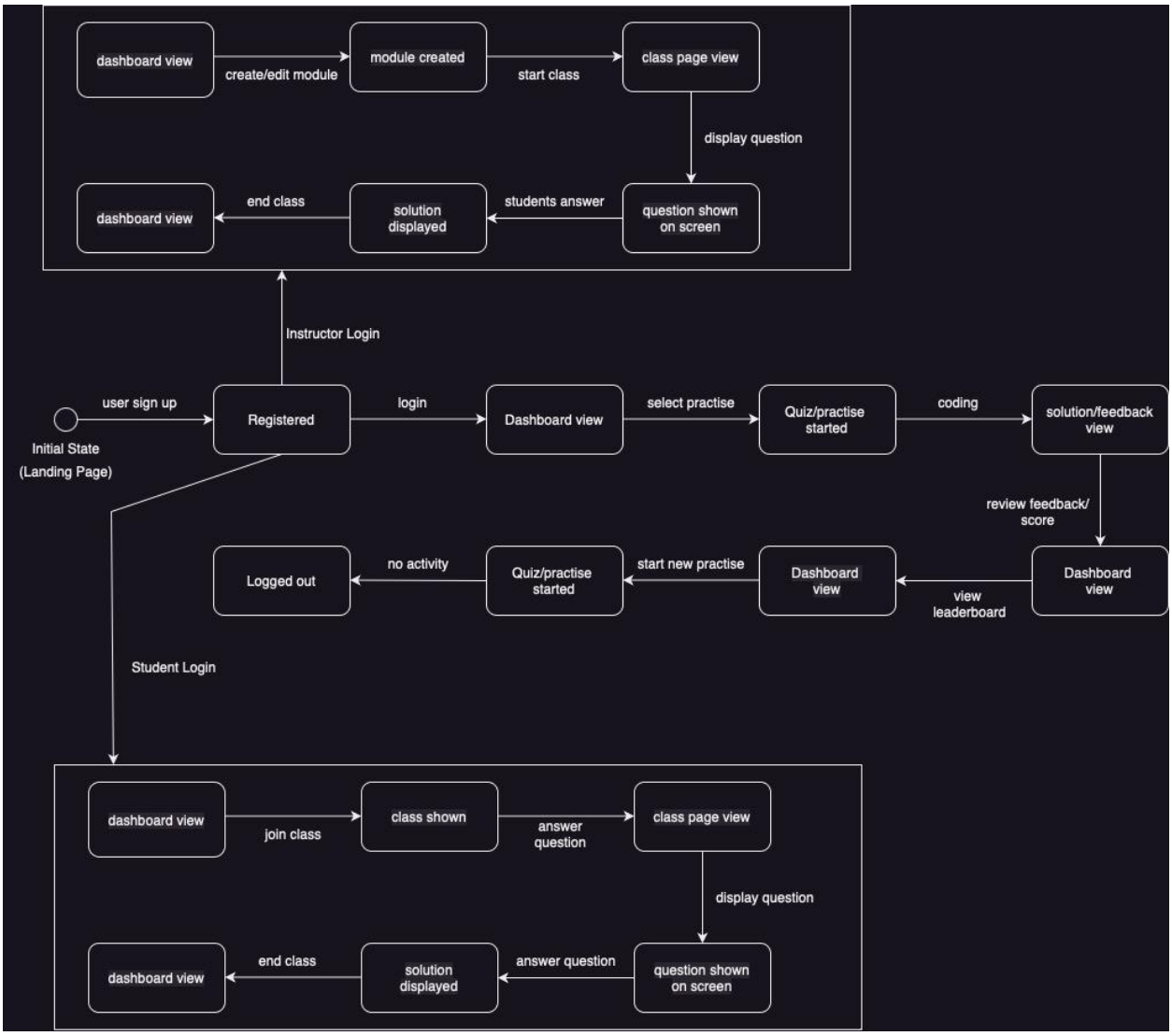


Figure 9.9: State diagram for GamifyGeeks

v. Sequence Diagram

The sequence diagram illustrates the flow of interactions and communication between different components of the system in the context of specific use cases.

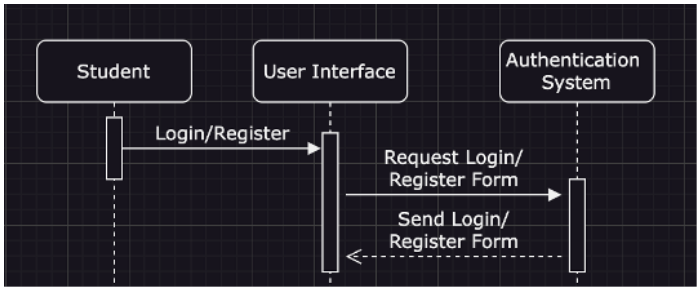


Figure 9.10: Student registration sequence diagram for GamifyGeeks

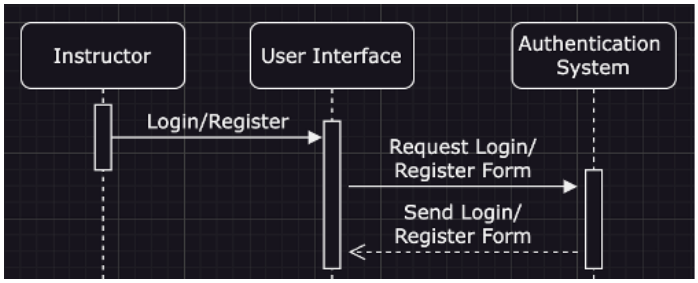


Figure 9.11: Instructor registration sequence diagram for GamifyGeeks

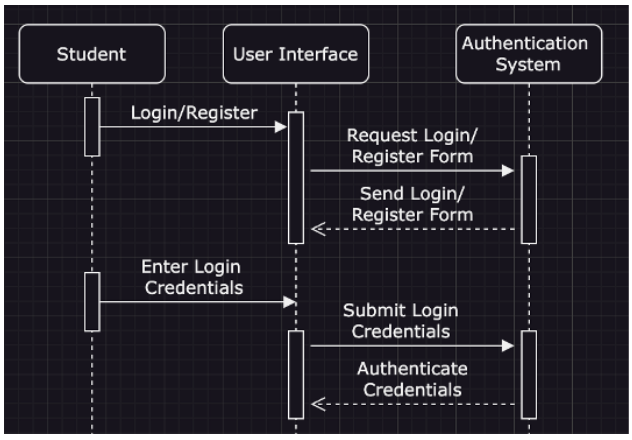


Figure 9.12: Student login sequence diagram for GamifyGeeks

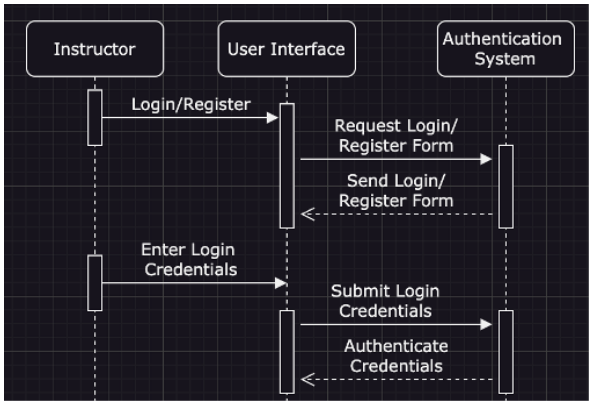


Figure 9.13: Instructor login sequence diagram for GamifyGeeks

After logging in:

Figure 9.14: Reset password sequence diagram for GamifyGeeks

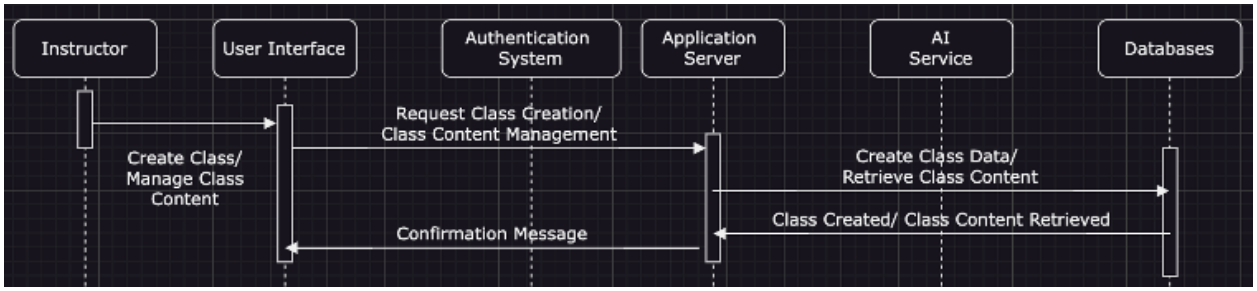


Figure 9.15: Instructor create/manage class content sequence diagram for GamifyGeeks

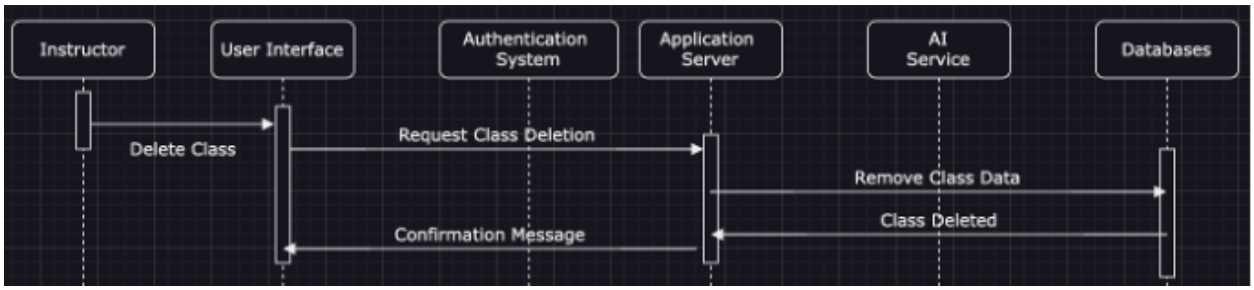


Figure 9.16: Instructors delete class sequence diagram for GamifyGeeks

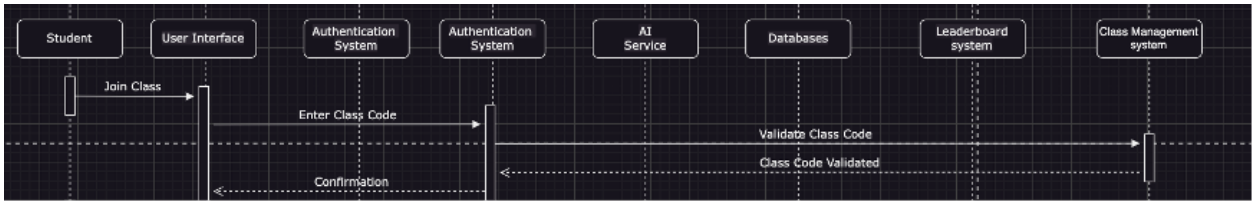


Figure 9.17: Student joining a class using code sequence diagram for GamifyGeeks

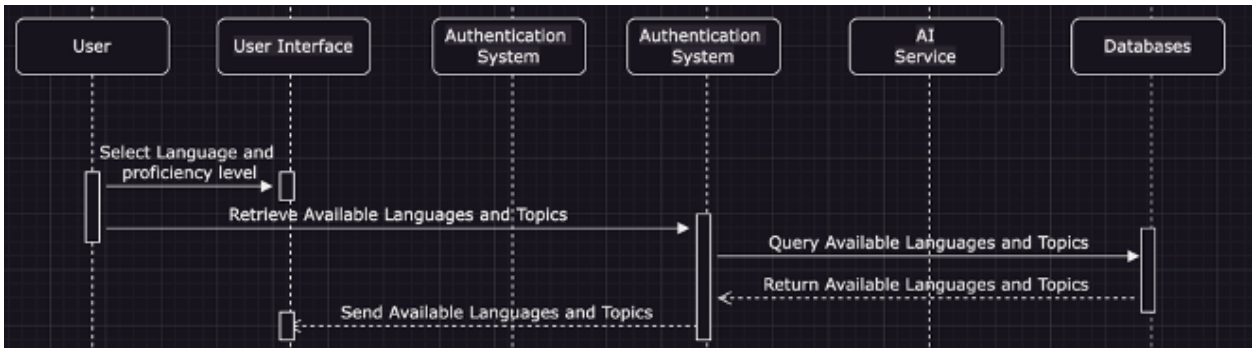


Figure 9.18: Learners choosing a language and proficiency sequence diagram for GamifyGeeks

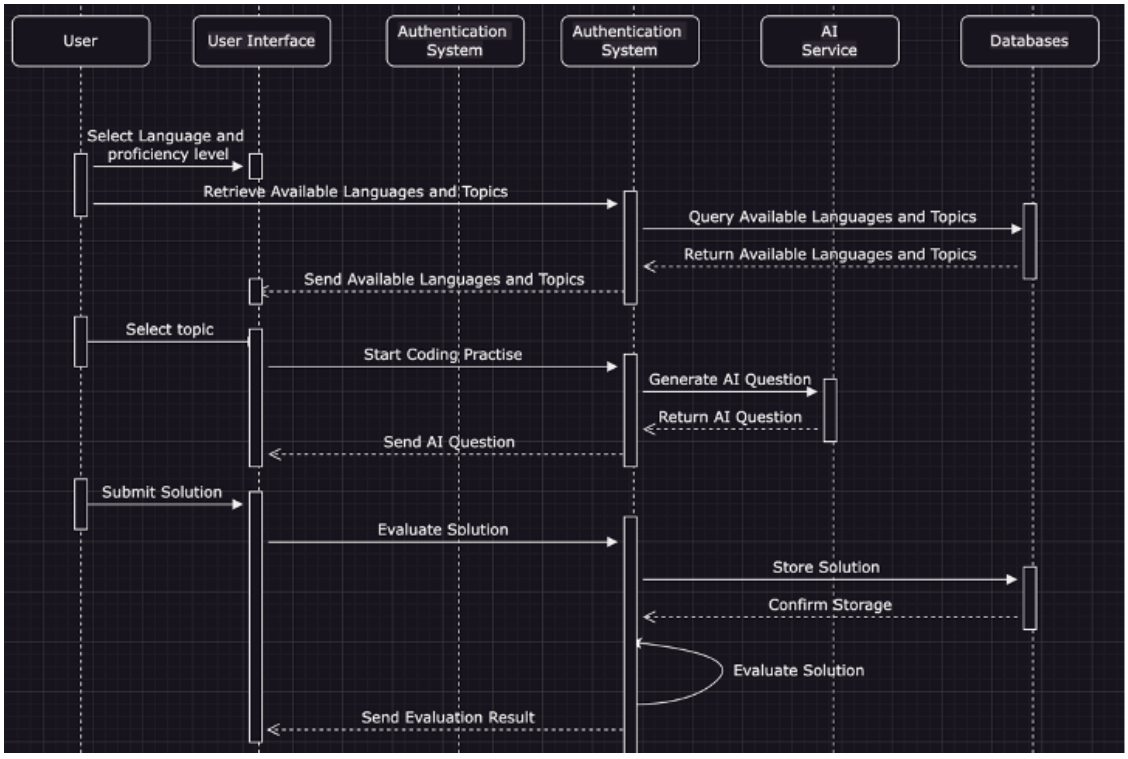


Figure 9.19: Learners choosing a topic and get an AI generated question and submitting solutions sequence diagram for GamifyGeeks

(011) Student and Instructor view Class Leaderboard

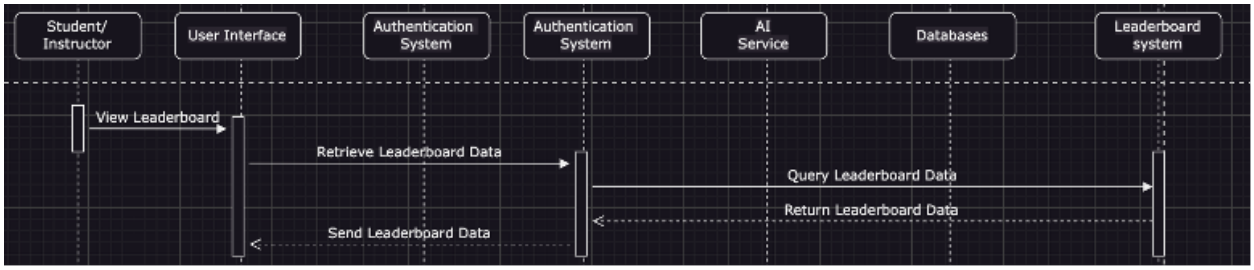


Figure 9.20: Student and Instructor view class leaderboard sequence diagram for GamifyGeeks

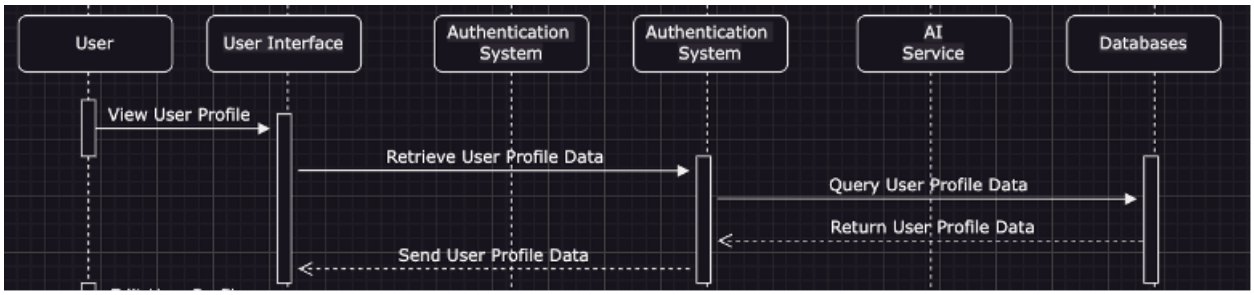


Figure 9.21: View own user profile sequence diagram for GamifyGeeks

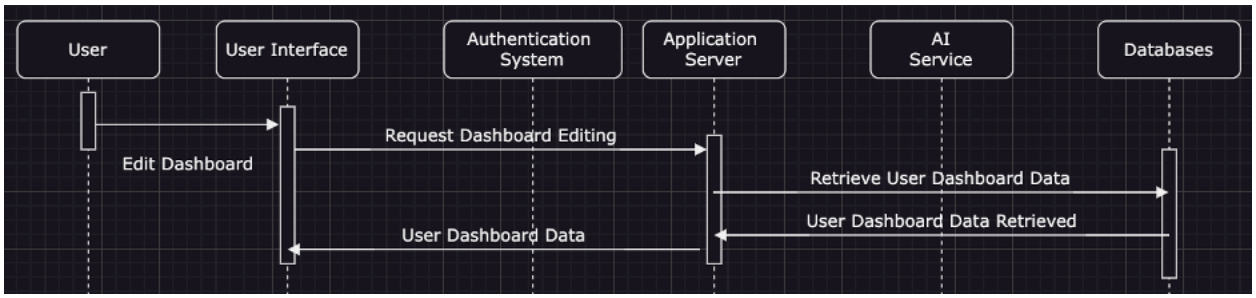


Figure 9.22: Edit own user dashboard sequence diagram for GamifyGeeks

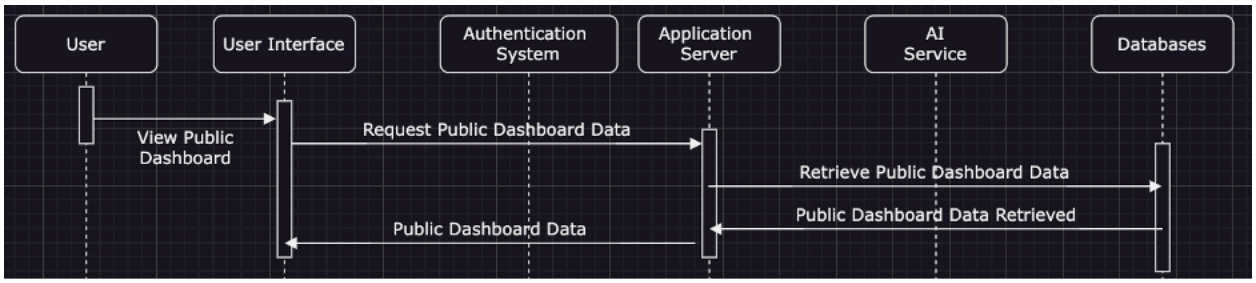


Figure 9.23: View public users dashboard sequence diagram for GamifyGeeks

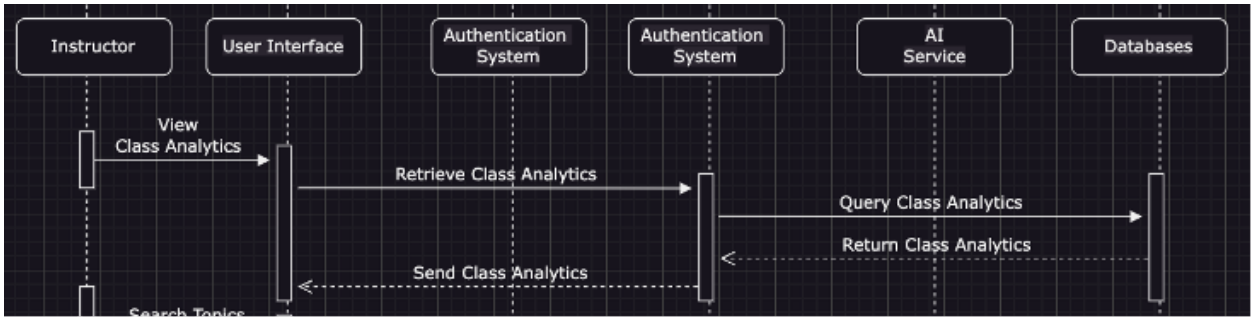


Figure 9.24: Instructor can view class and user analytics sequence diagram for GamifyGeeks

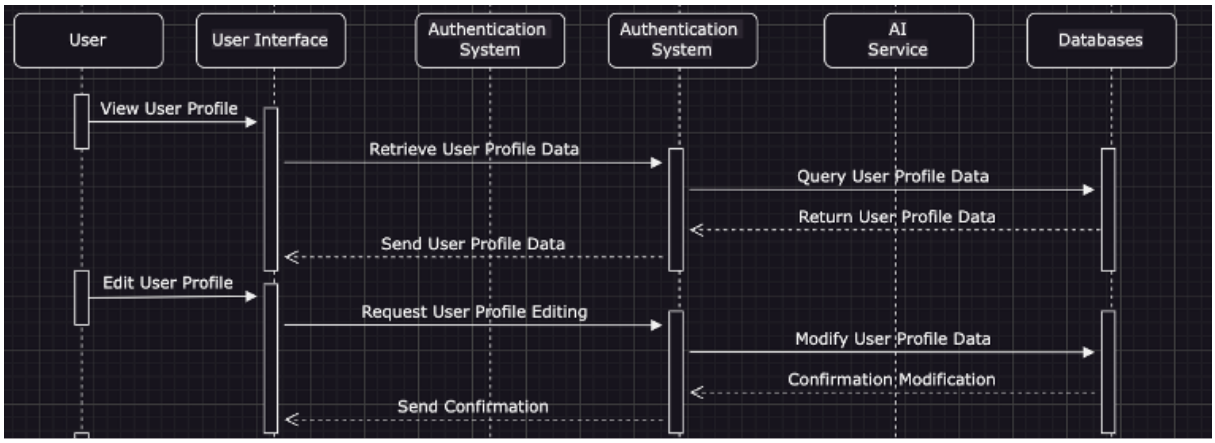


Figure 9.25: Edit user profile sequence diagram for GamifyGeeks

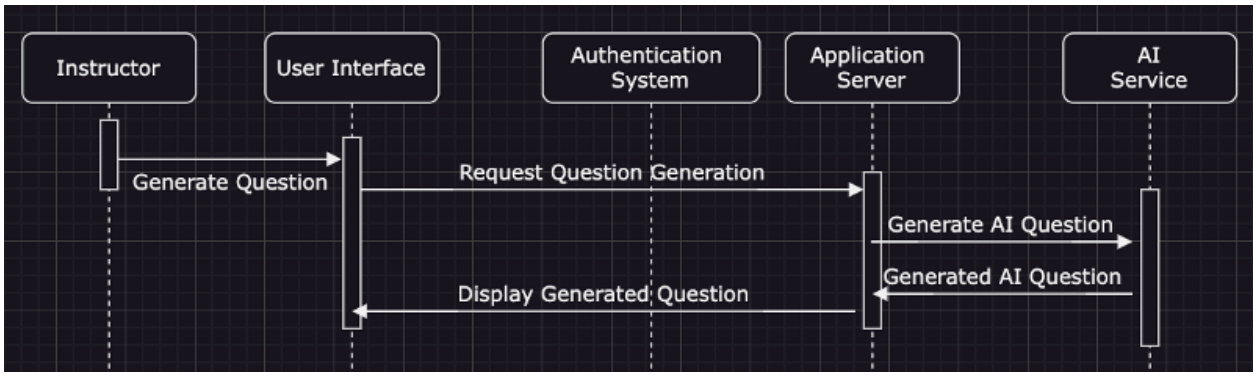


Figure 9.26:Instructor generating question sequence diagram for GamifyGeeks

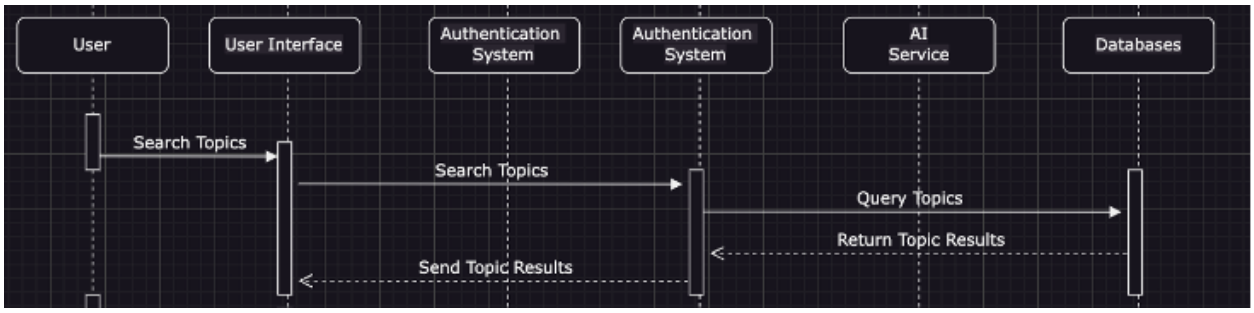


Figure 9.27: Search topics sequence diagram for GamifyGeeks

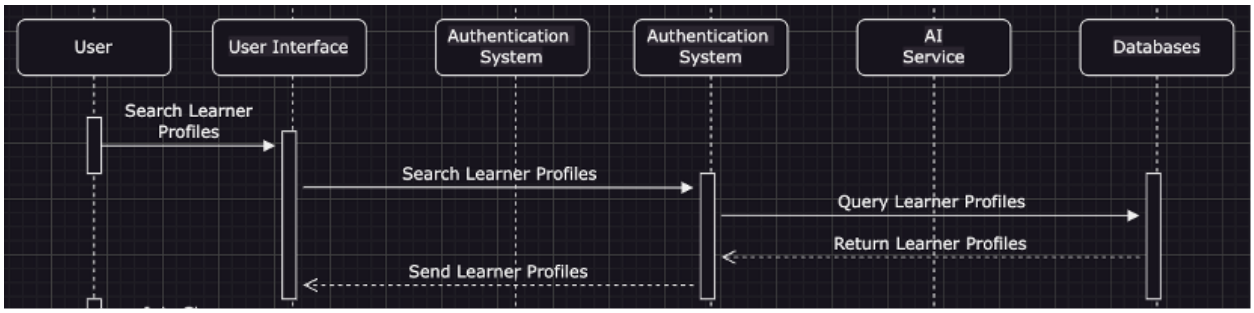


Figure 9.28: Search learner profiles sequence diagram for GamifyGeeks

vi. Activity Diagram

The activity diagram outlines the user journey on our Code Practice website. Users start by choosing a programming language, then select a topic or difficulty level. Our AI generates a question based on these choices, and the user submits their answer. The AI then evaluates the solution, providing instant feedback. After this, users can continue with new questions, retry the same question, or end their practice. This diagram emphasizes our aim to offer a personalized and efficient learning experience.

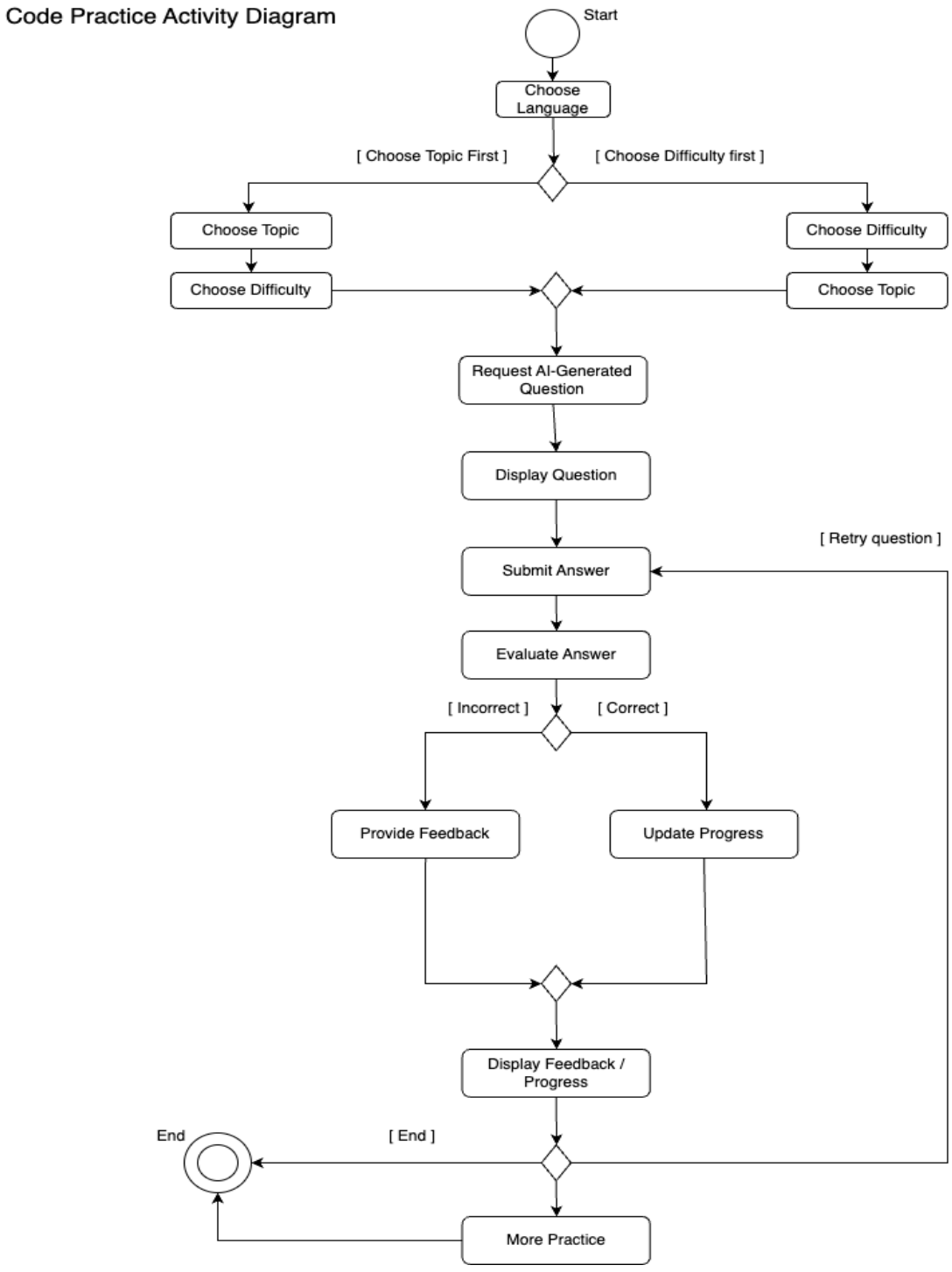


Figure 9.29: Activity diagram for GamifyGeeks

d. UI Mockups

Figure 9.30: Hero section

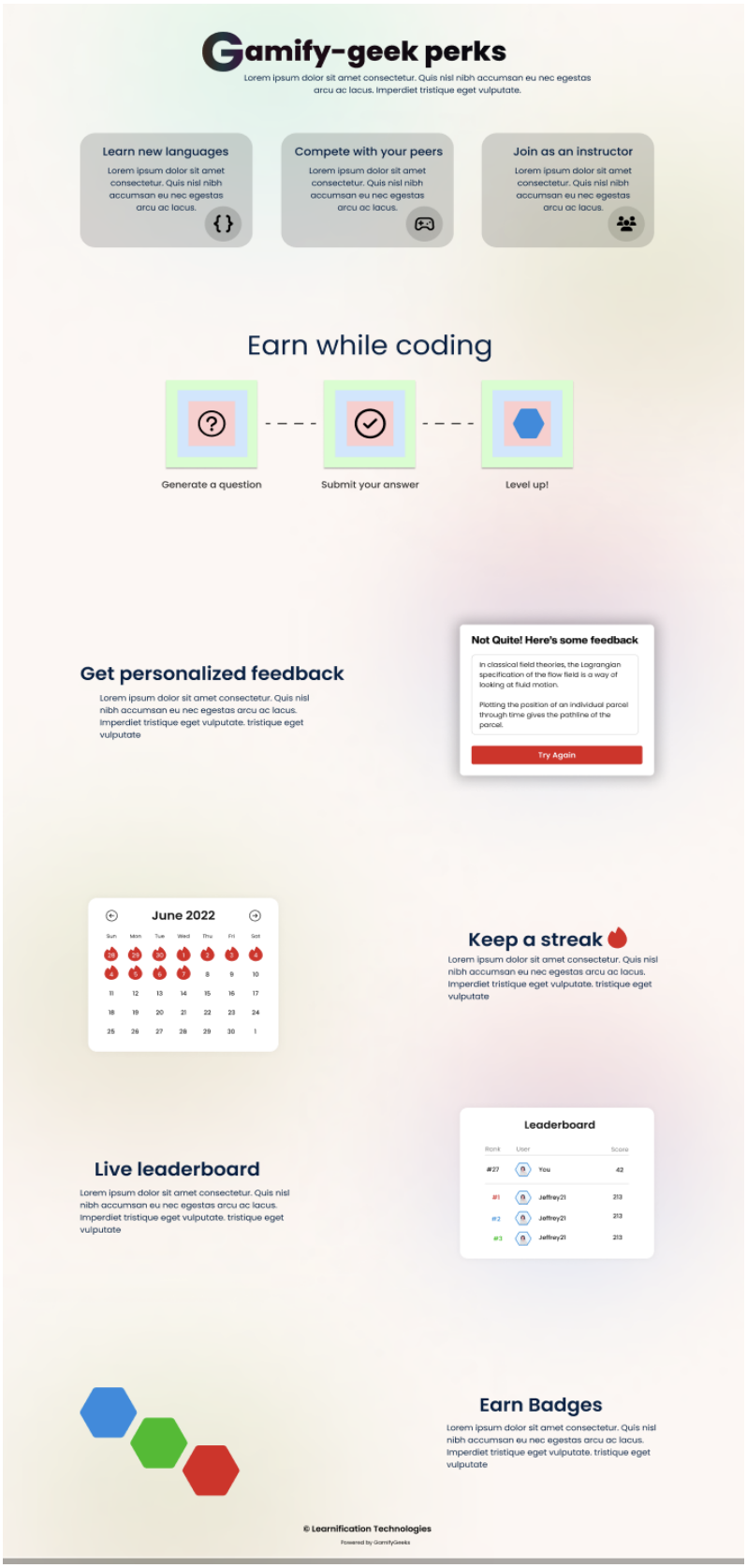


Figure 9.31: Full landing page

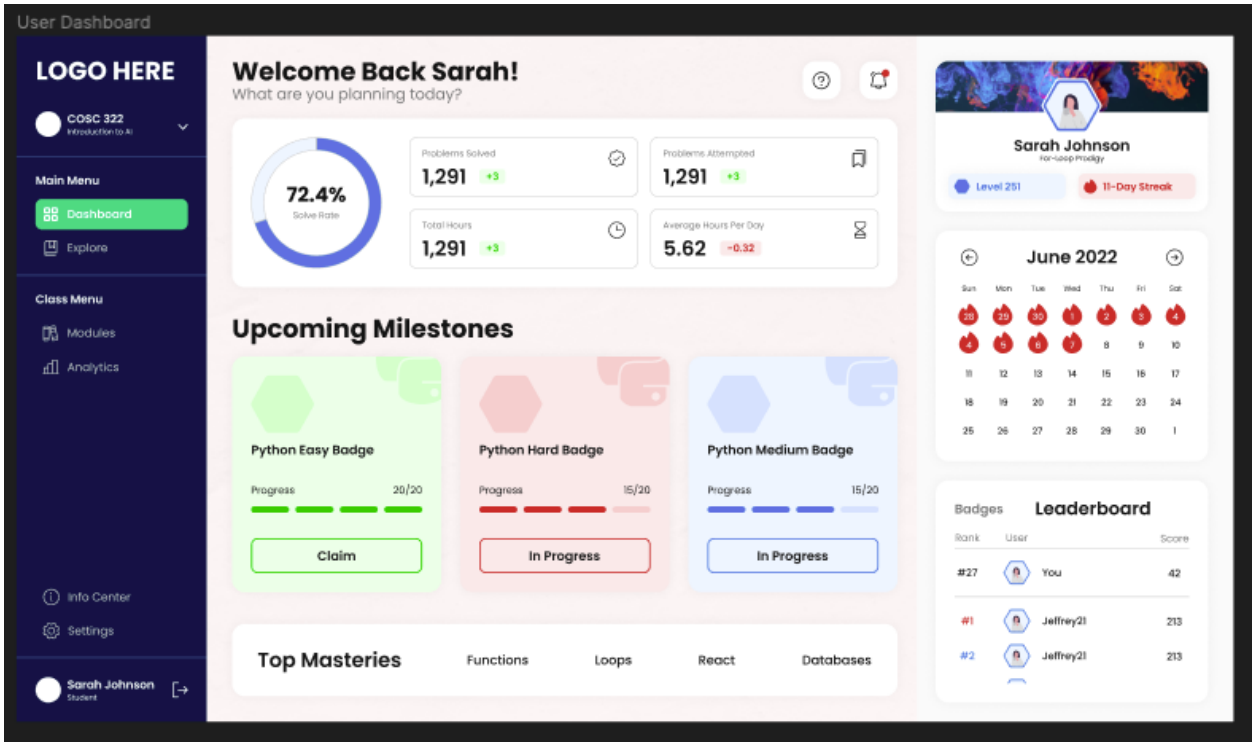


Figure 9.32: User dashboard page

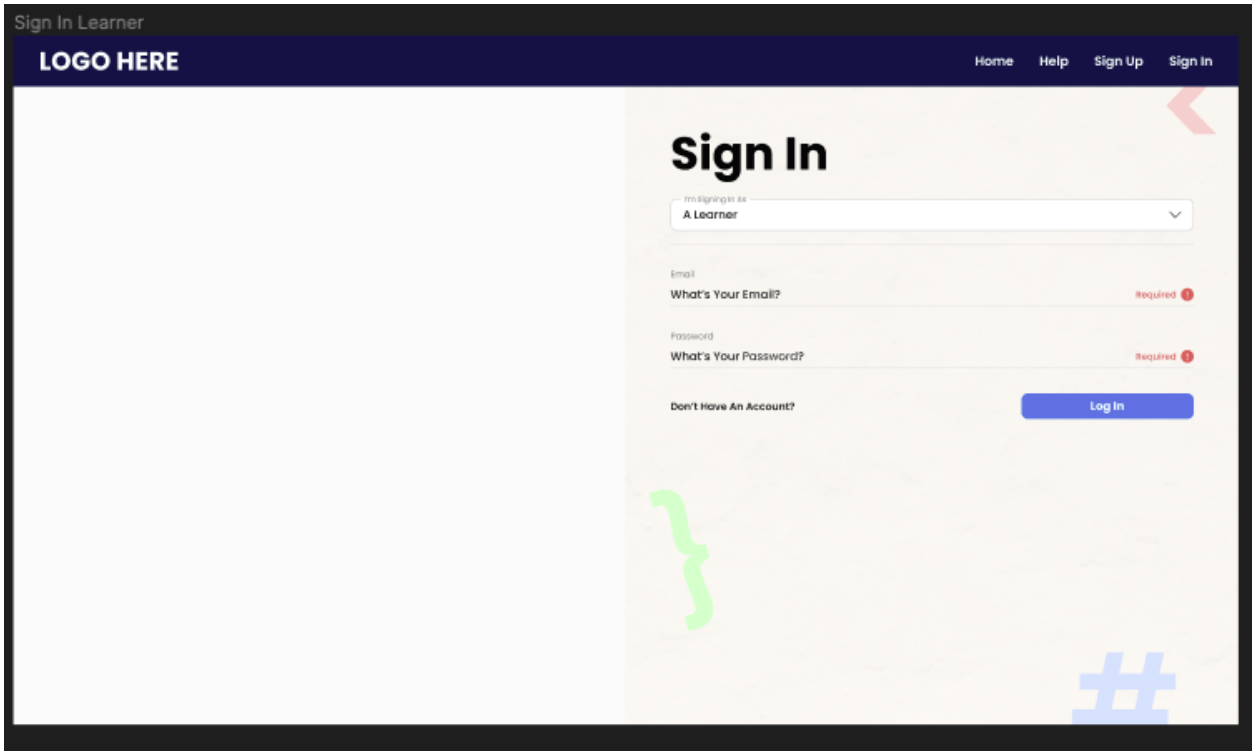


Figure 9.33: User sign in page

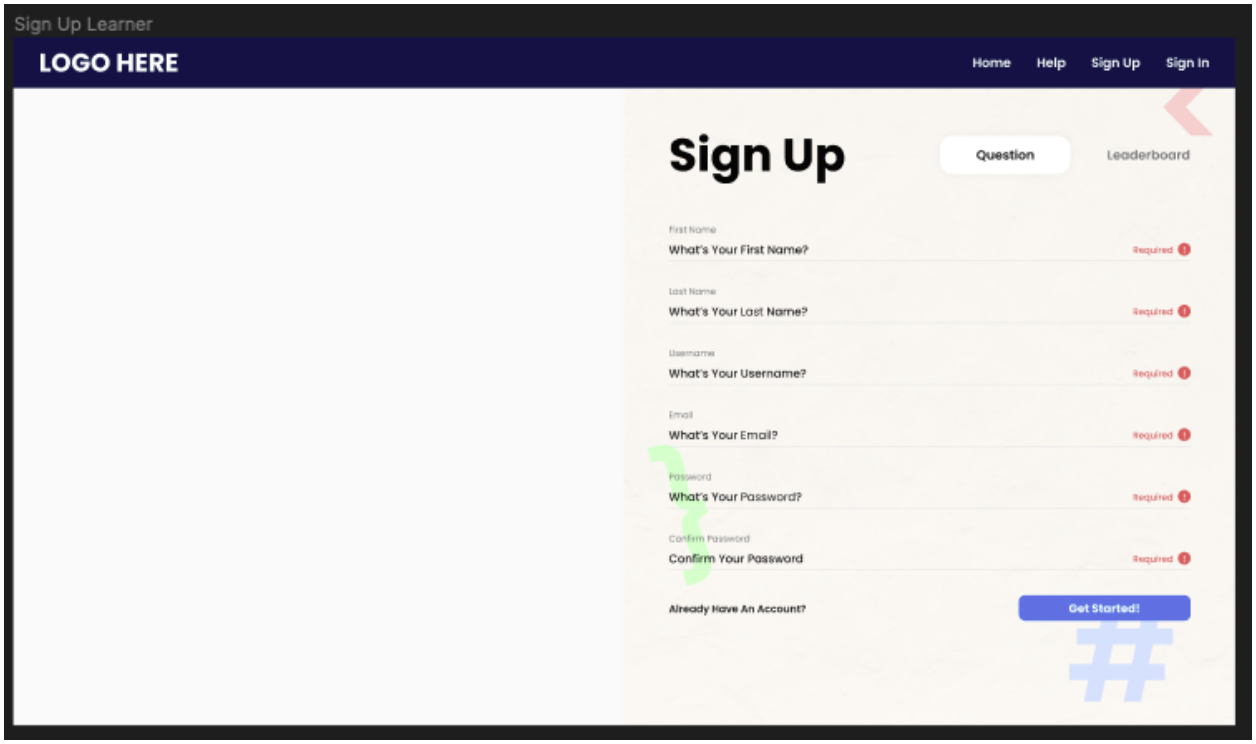


Figure 9.34: User sign up page

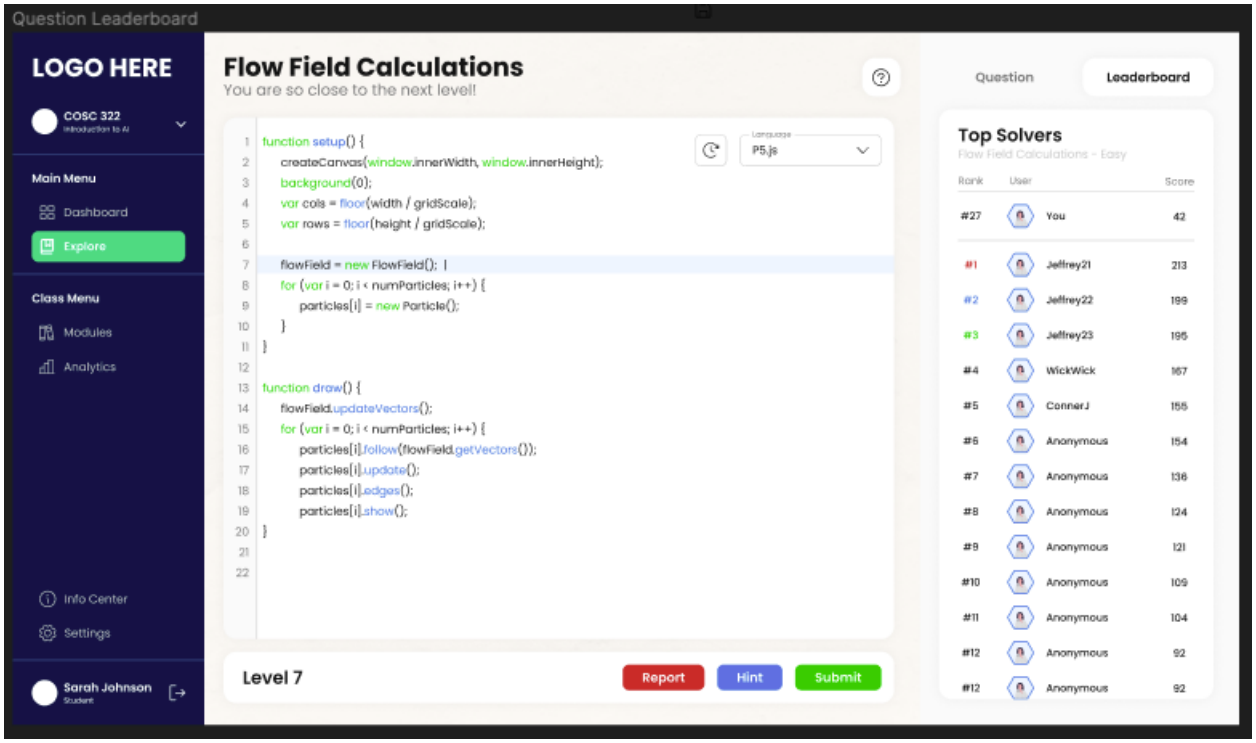


Figure 9.35: Question/IDE page

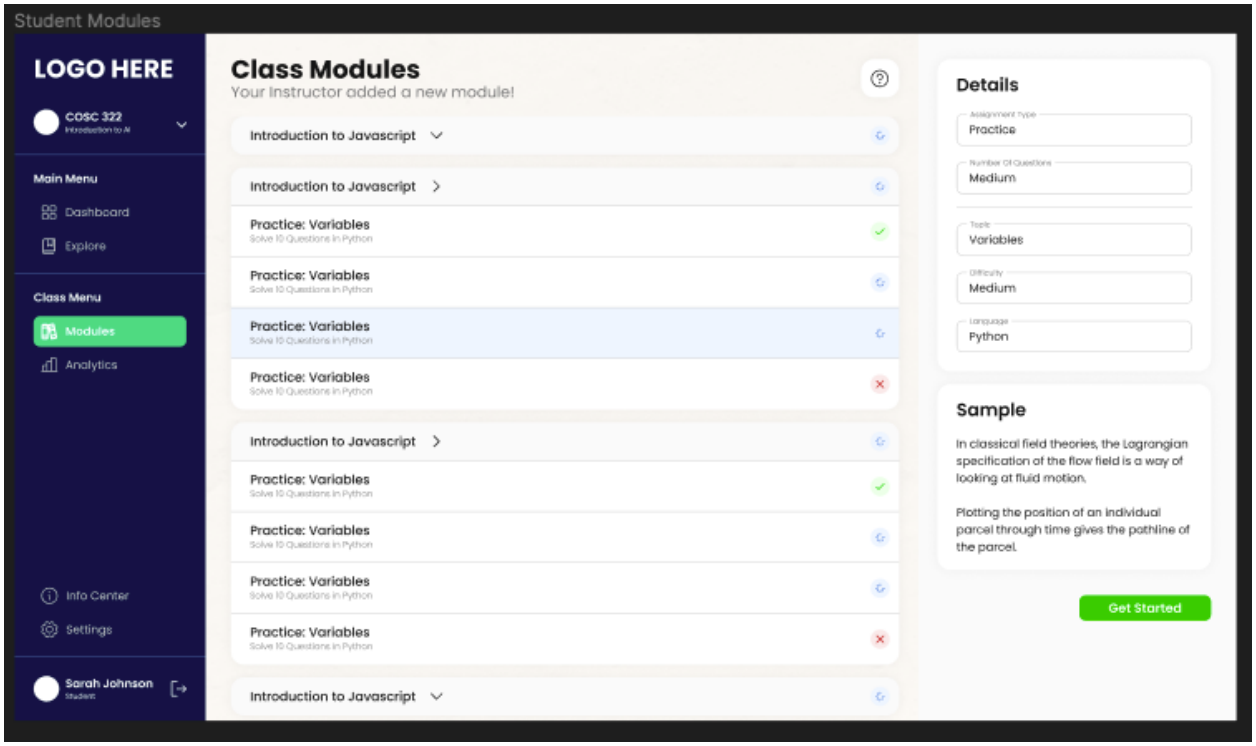


Figure 9.36: Student modules page

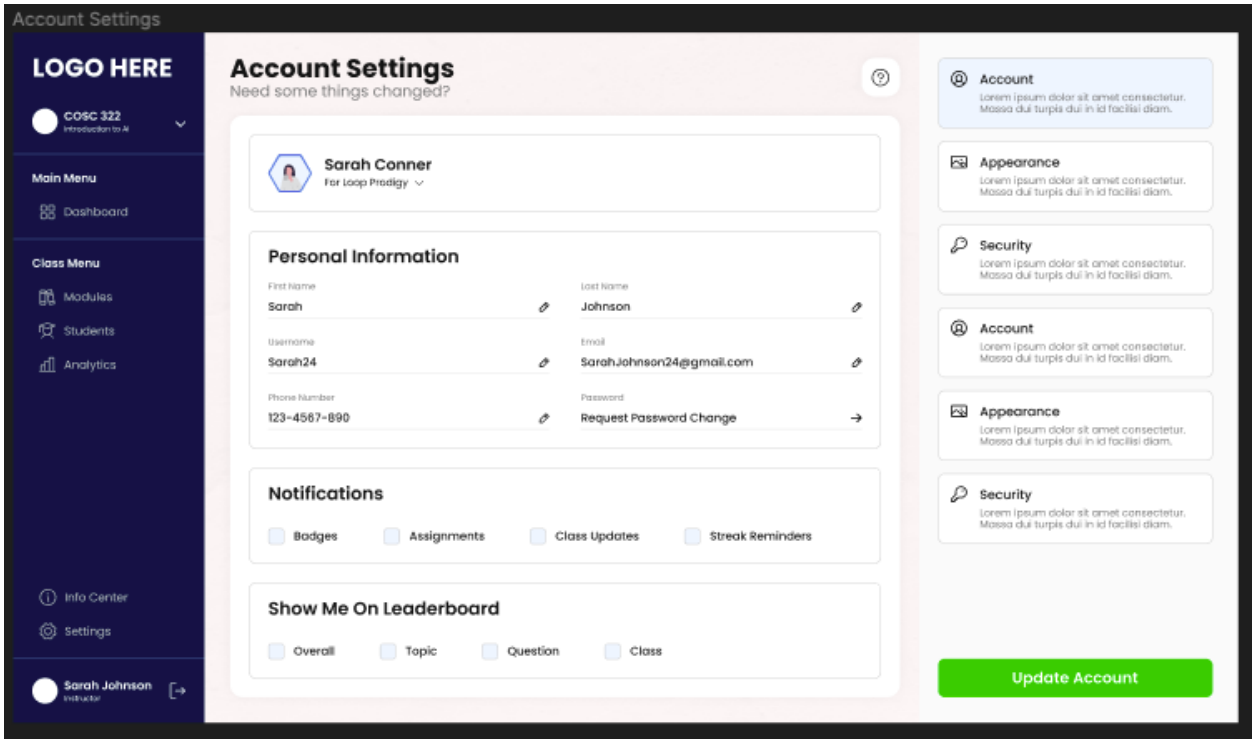


Figure 9.37: Accounts page

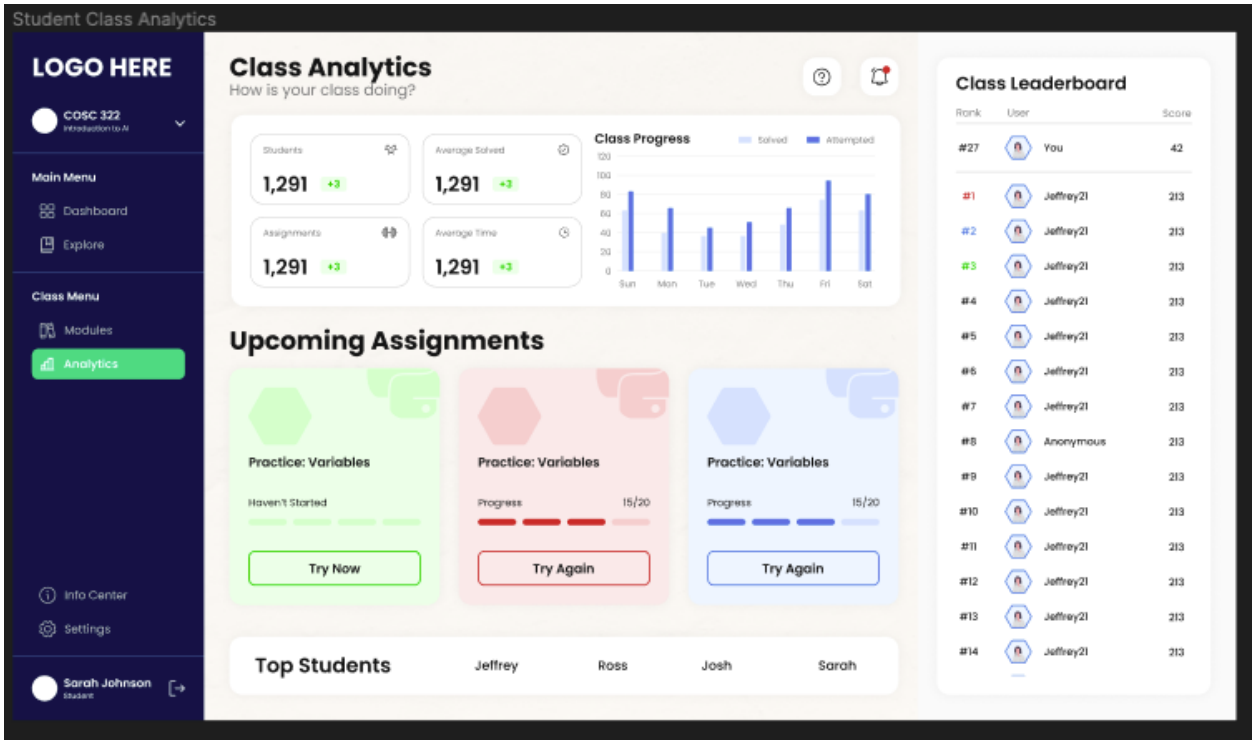


Figure 9.38: Student Analytics page

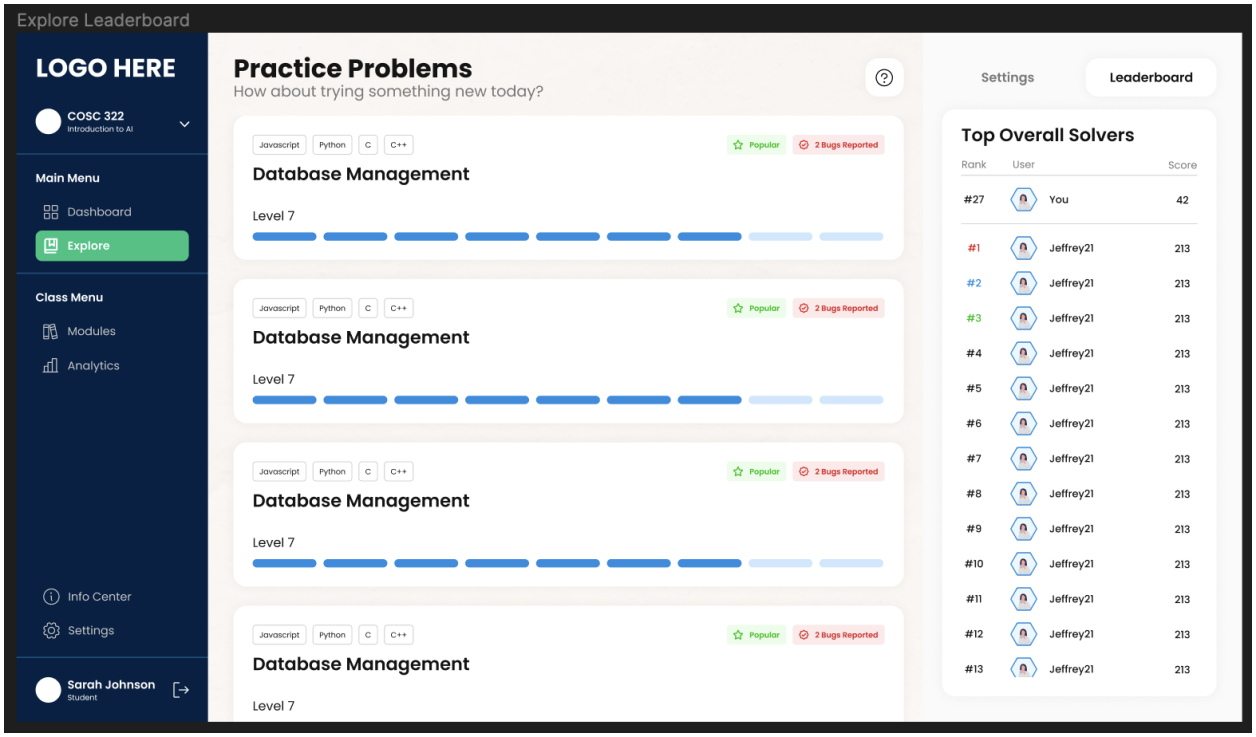


Figure 9.39: Topic exploration page

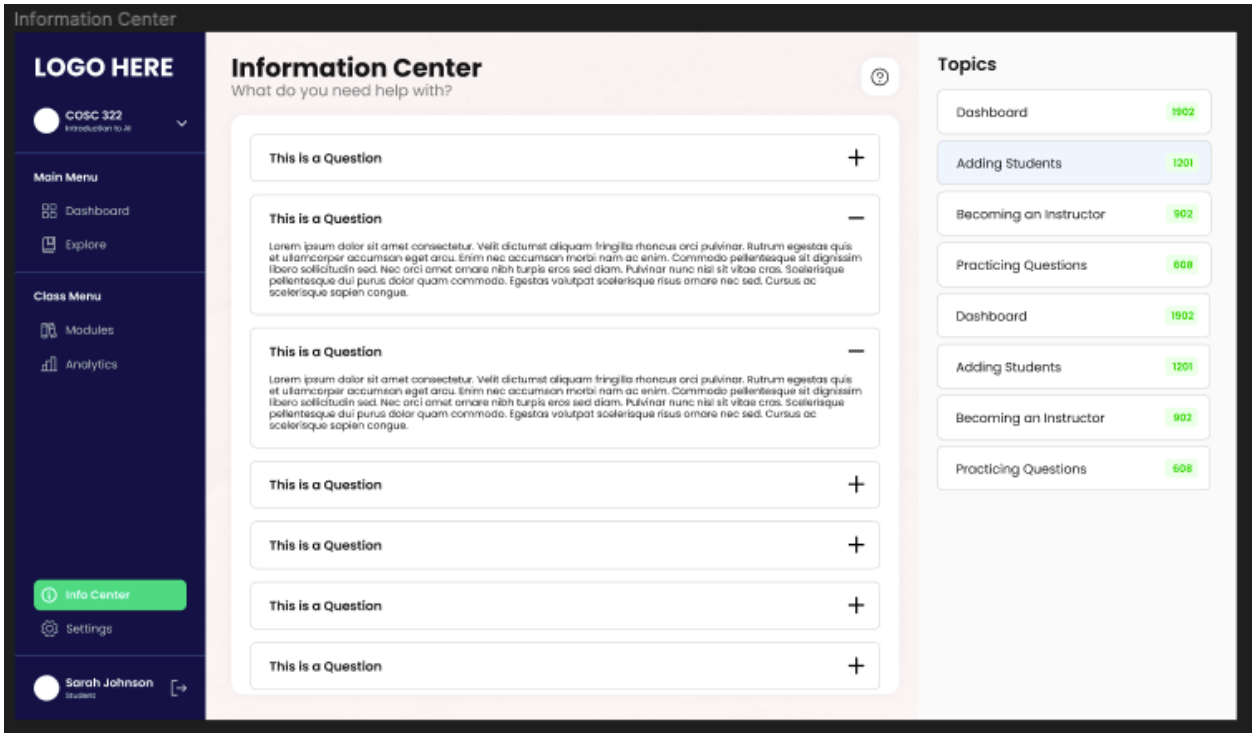


Figure 9.40: Information center page

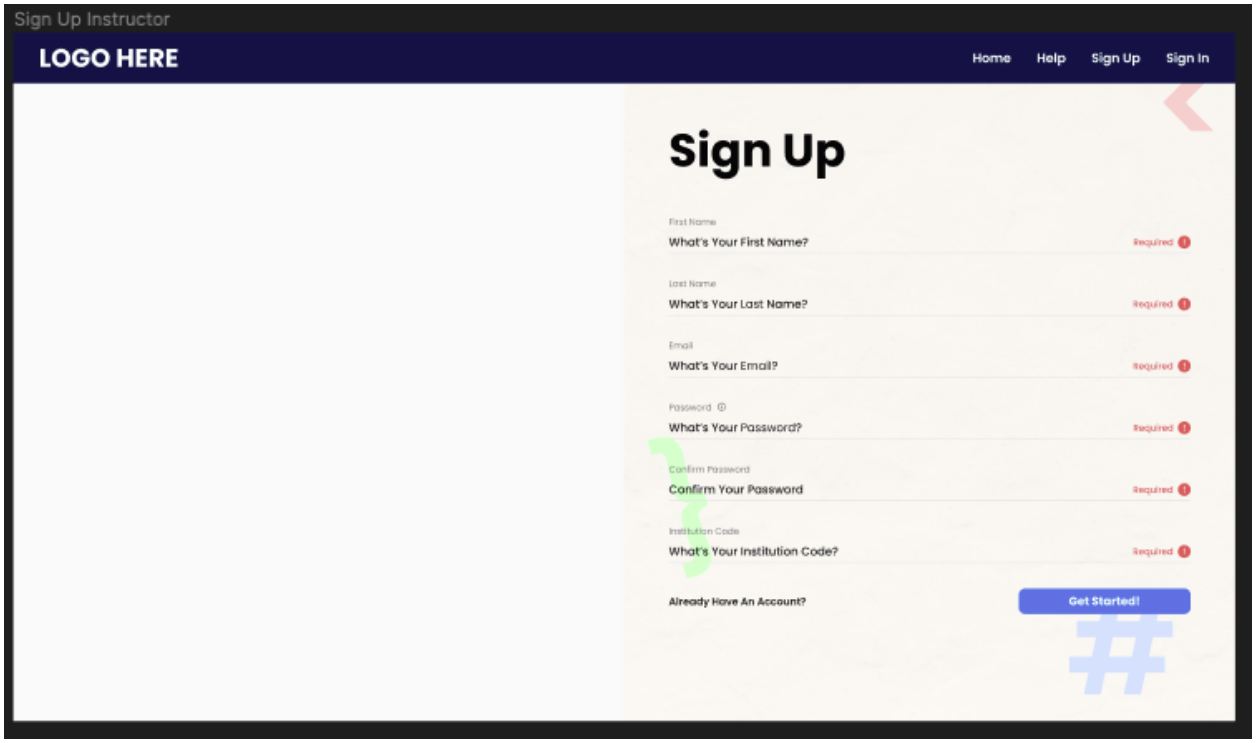


Figure 9.41: Instructor sign up page

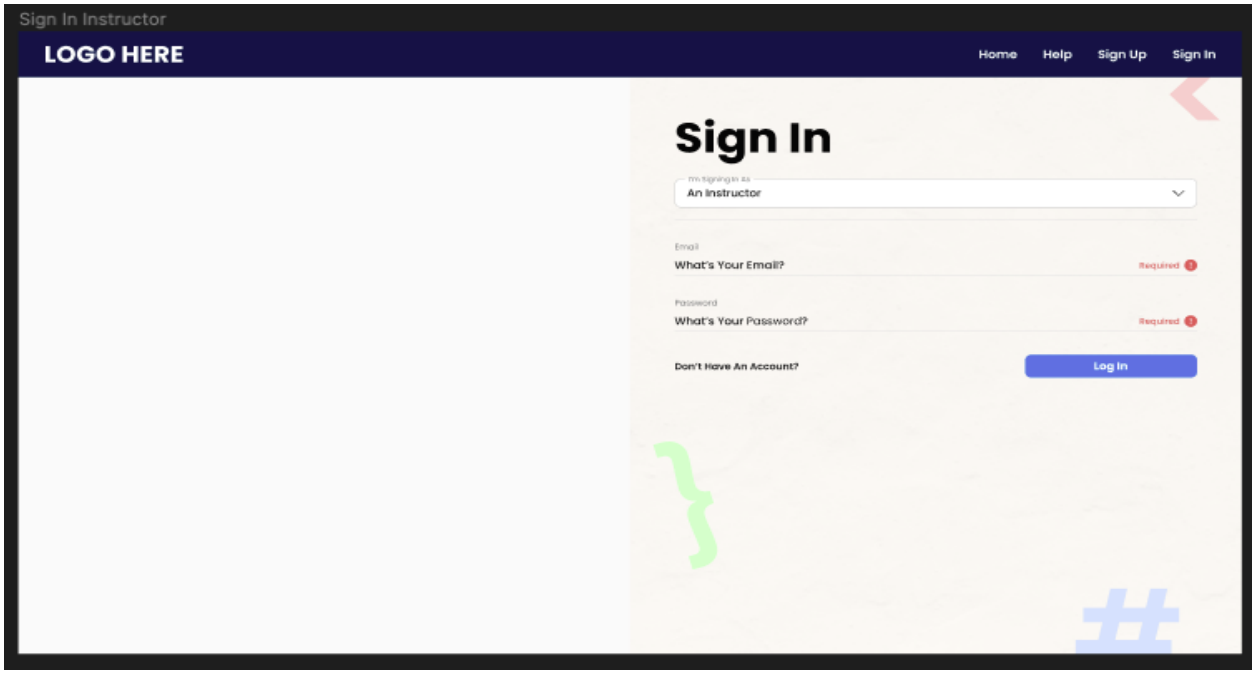


Figure 9.42: Instructor sign in page

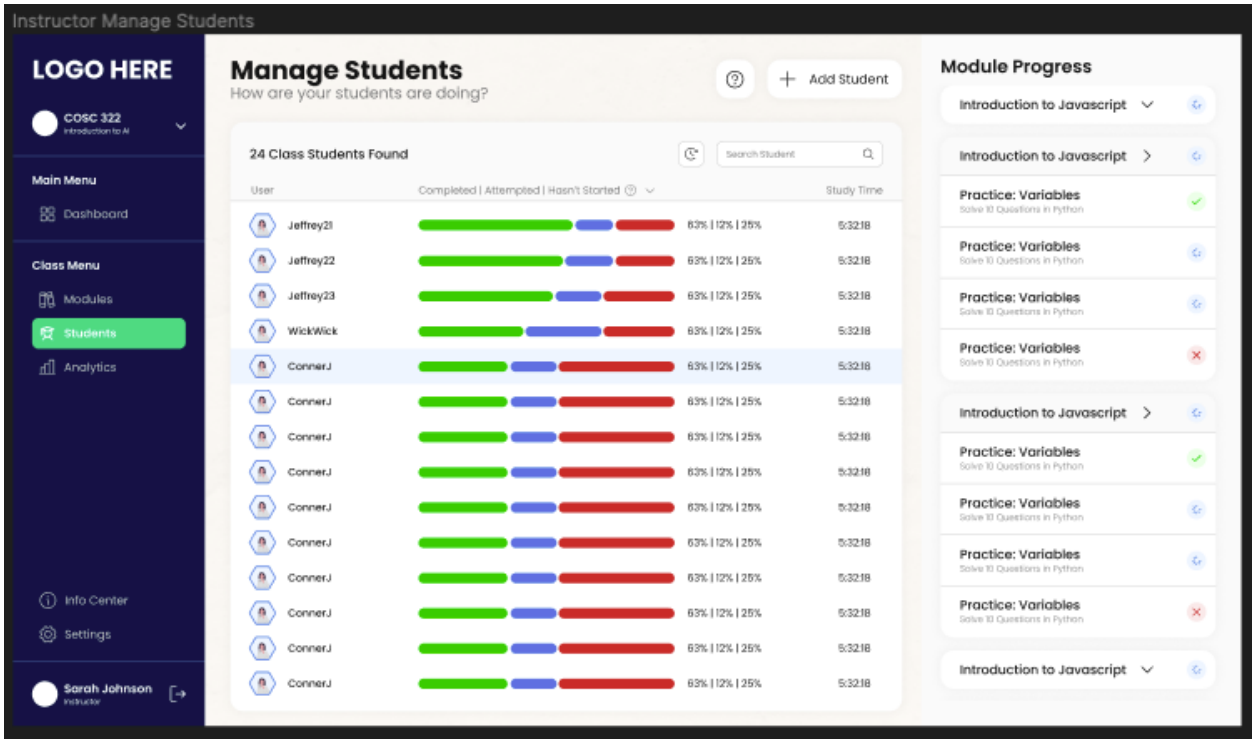


Figure 9.43: Instructor manage students page

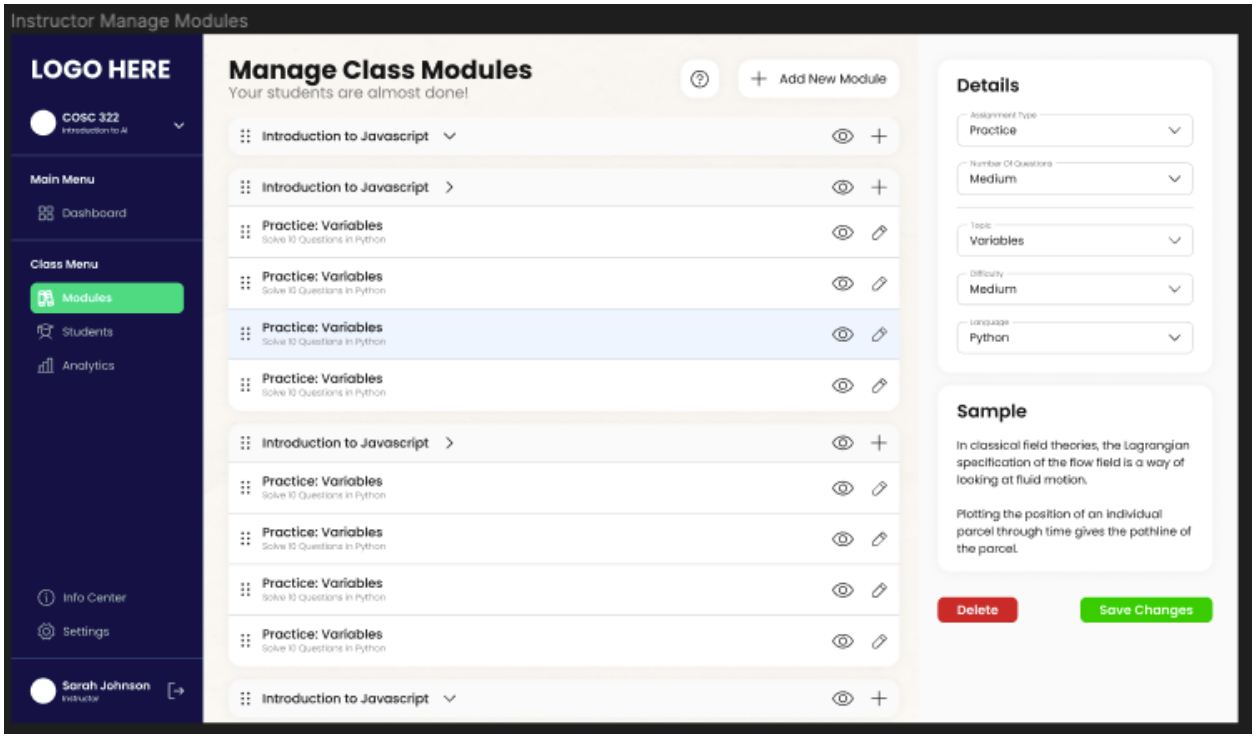


Figure 9.44: Instructor manage modules page

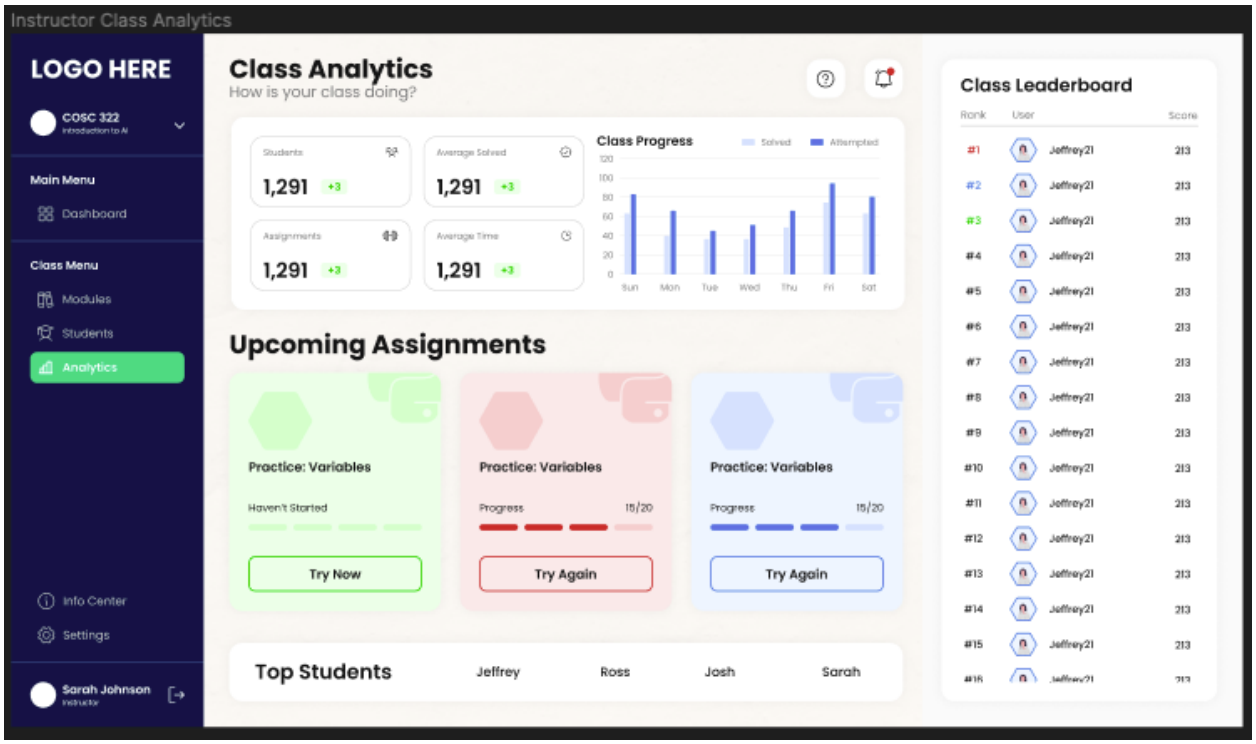


Figure 9.45: Instructor class analytics page

e. Technical Specifications

i. Tech Stack

The project will utilize the SERN tech stack, also known as the SQL, Express, React, and Node tech stacks. This is a modern web development tech stack that allows the use of Javascript for both front and backend development, eliminating the need to switch between different programming languages when developing the full stack. This will consist of the following:

1. **SQL:** Programming language primarily used for managing and manipulating relational databases. SQL will be used in the database management system to query and update the stored data.
2. **Express:** A minimal web application framework for NodeJS primarily focused on providing a set of server-side features and tools. Express will be used to handle HTTP requests, routing, and middleware management.
3. **React:** A popular JavaScript library for building reusable UI components and user interfaces. React will be used for the client-side rendering to create dynamic and interactive user interfaces.
4. **NodeJS:** A JavaScript runtime environment used in the creation of server applications. NodeJS will be used for handling external libraries, modules, and packages for different features required in the system.

ii. Tools

IDE: Visual Studio Code
Version control systems: Git
Project management tools: Google Drive, Slack, and Discord
Unit testing: Jest
CI/CD: GitHub Action
Deployment & Hosting: Docker, Digital Ocean
APIs: Cohere API, OpenAI API
Software Development Methodologies: ScrumBan

10. Testing

a. User Testing Results (Test-O-Rama)

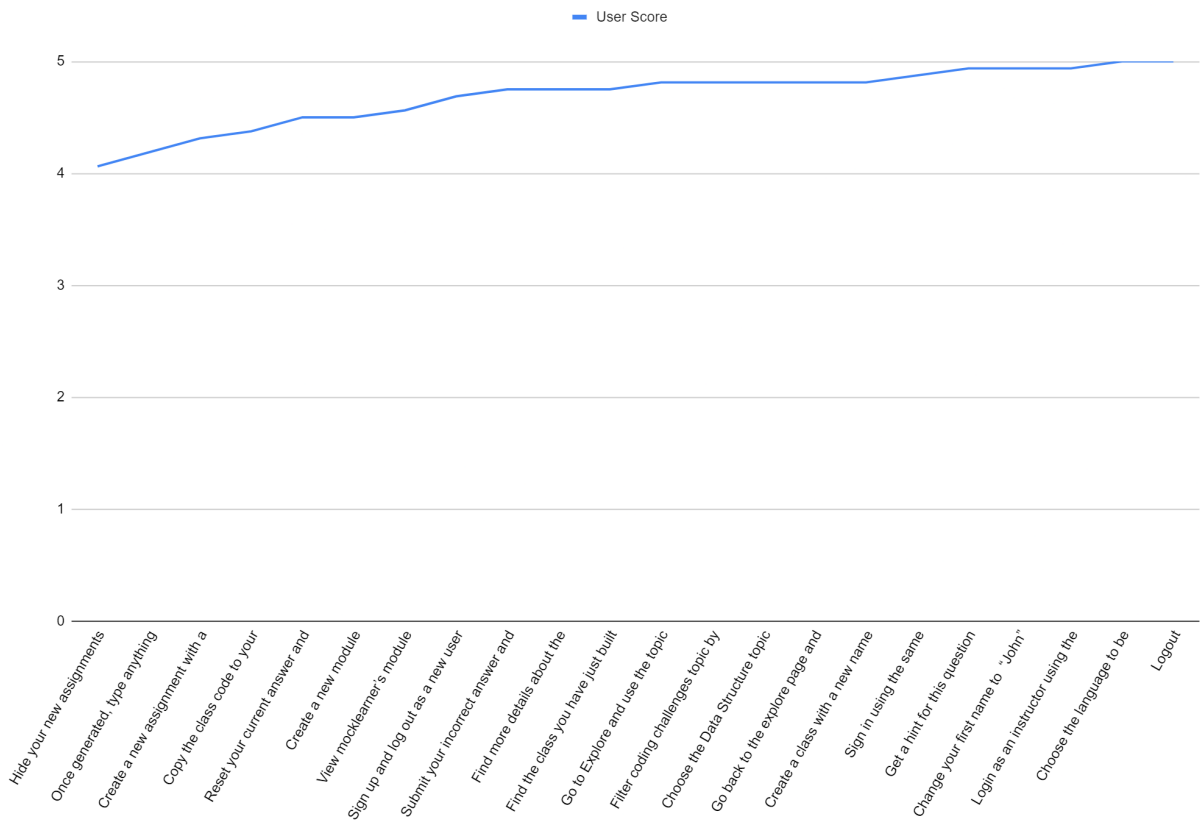


Figure 10.1: Features ranked worst to best based on user testing results

b. User Testing Takeaways

Top-Performing Features: The features associated with logging out, logging in, and making choices within the platform garnered the highest scores among all the features. These favorable scores are indicative of users perceiving these features as intuitive, effective, and user-friendly. It can be inferred that users were able to effortlessly complete these tasks.

Areas for Enhancement: While the scores maintain a level above 4 out of 5, the features that received the relatively lower scores were those linked to course data management and utilization of the embedded IDE. Notably, these features coincide with some of the most intricate pages in terms of functionality. This suggests that there might be room for improvements in

terms of enhancing both user accessibility and overall user experience within these particular features.

Found Issues: The concerns identified during the test-o-rama pertained to the pre-loaded SQL data, which may not have adhered to the identical procedures as the corresponding APIs. Additionally, a number of recurrent discrepancies were observed within the more intricate pages, primarily involving data misalignment or errors. Moreover, it was noted that the extensive array of features inherent to the platform imparts a learning curve upon initial users.

c. Applied Solutions

Rethinking Initial DDL: The data initially inserted into the Data Definition Language (DDL) underwent a thorough examination to identify and rectify any errors. These inserted data entries were meticulously cross-referenced with their respective Application Programming Interface (API) steps to ensure congruence. Additionally, a supplementary set of data was introduced into the system. This augmentation aimed to comprehensively assess the platform's responsiveness in scenarios involving expanded datasets of varying compositions.

Bug Fixes & New Tests: A substantial portion of the identified bugs were, in fact, attributed to minor oversights in the development process, stemming from the constraints imposed by time limitations. The remedial course of action initiated by the team encompassed the creation of a comprehensive checklist detailing the known bugs. Systematically, each item on this checklist was meticulously addressed and rectified. Moreover, in order to fortify the robustness of the platform moving forward, novel testing protocols were implemented. These tests were devised to ascertain the sustained functionality of these features in subsequent iterations.

User Accessibility & Help Features: Subsequent to the resolution of the bugs, the team directed their attention towards the pages that had garnered relatively lower scores. In pursuit of enhancing user experience, concerted efforts were made to optimize user accessibility and facilitate interactions. One notable measure involved refining the management of courses by eliminating redundant clicks and minimizing potentially overwhelming features. Furthermore, to bolster user engagement and prevent confusion, subtle yet effective adjustments were implemented. This encompassed the strategic deactivation of buttons and texts when their functionality was inapplicable, thereby affording a clearer indication of the platform's readiness for their use.

d. Testing Report

Gamify Geeks Test Report

This table explains the various activities performed as part of the Testing of the 'Gamify Geeks' web application

	REQUIREMENTS	**Type of Test** (I: Integration Testing, S: System Functionality Testing, O: Operational Acceptance Testing, UN: Unit Testing, US: Usability Testing, A: Acceptance Testing, EE: End to End Testing)	**Pass or Fail** (P: Pass, F: Fail)	**Contributor** (JR: Jason Ramos, GM: Gyumin Moon, JG: Joseph Gaspari, AG: Archita Gattani, AV: Alrick Vincent)
Functional Requirements				
1 - Must Have				
1.2	The system must be able to accept user feedback on questions and reward feedback with bonus points and badges, which prompts system improvements.	UN, EE	p	JG, JR
1.3	The system must be able to prompt AI-generated questions based on a specified topic, level of proficiency, and programming language chosen by the user.	UN, EE	p	JG, JR
1.4	The system must be able to track and display user progress through course milestones, user points, achievement badges, and usage analytics.	UN, EE	P	AV, JR
1.5	The system must allow instructors the ability to view student and class analytics including students' progress in the assigned questions, how many they have completed, how many they have left, percent success rate, visual dashboard of progress, number of questions they have completed, their streak/points and the number of	EE, UN	P	JR, AV

	badges earned.			
1.6	The system must support user login, logout, and registration.	UN, I, EE	P	AV, JR
1.7	The system must be able to indicate whether user-provided answers are correct or incorrect and provide meaningful AI-generated feedback with high confidence e.g. 95%.	UN, I, EE	p	JG, JR
1.8	The system must be able to validate whether an AI-generated question relates to the topic selected with a high degree of confidence (95%)	UN	p	JG
1.9	The system must support JavaScript, Python, and HTML/CSS.	UN, EE	p	JG, JR
1.10	The system must support the main user archetype: learners.	UN, EE	P	AV, EE
1.11	The system must include tooltips to guide users.	EE	P	JR
1.13	The system should be able to give users AI-generated suggestions or hints when prompted by the user or a series of incorrect answers.	UN, EE	p	JG, JR
1.14	The system should have a public leaderboard categorized into topics and programming languages. This can be turned off, and the scores will be calculated by correct answers (weighted by question difficulty), milestones, streaks and badges.	UN, EE	P	AV, JR
1.15	The system should contain user dashboards for students showing course progress, badges earned, and other user achievements.	UN, EE	P	AV, JR
1.16	The system should support the main user archetype: Instructors.	UN, EE	P	AV, JR
1.19	The system should allow learners to join classes through a join code similar to Kahoot platform.	UN, EE	P	AV, JR
2 - Should Have				
2.1	The system should easily accommodate additional programming languages for students to practice.	/		JG
2.2	The system should have an FAQ page that	EE		JR

	includes questions/answers that may be popular.			
Non-Functional Requirements				
2.2	The AI-generated questions should have less than a 5% user report rate.	UN	P	JG
Technical Requirements				
1 - Must Have				
1.4	The system must integrate with an AI API to generate practice questions.	UN	P	JG
1.5	The system must be dockerized for deployment.	UN	P	AV
1.6	The system must securely store user passwords through salting or hashing.	UN	P	AV
1.7	The system must have server-side security for user inputs.	UN, EE	P	AV, GM, JR
1.8	The system's database must be a relational database.	UN	P	AV
1.9	The system must have validation for all user-input fields.	UN, EE	P	AV, GM, JR
1.11	The system must include server-side logging that captures all database queries and events that may lead to a failure.	UN	P	AV
2 - Should Have				
2.1	The system should indicate to users any errors found within the system.	UN	P	AV
2.2	The system should have client-side security for user inputs.	EE	P	JR
User Requirements				
1 - Must Have				
1.1	All users must be able to register and login to the system.	UN, EE	P	AV, GM, JR
1.3	Learner users must be able to choose their programming language and topic and level of proficiency.	EE	P	JR

1.4	Learner users must have an alias or username.	UN, EE	P	AV, GM, JR
1.5	Learners must be able to view their streak while accessing the platform.	UN	P	AV, JG
1.7	Instructor users must be able to create a class and share a join code.	UN, EE	P	AV, JR
1.8	Instructor users must be able to create a set of topics and customize the parameters such as proficiency level and question format for the to-be-generated questions.	EE	P	JR
1.9	Instructors must be able to create and keep track of multiple courses at the same time.	UN	P	AV
2 - Should Have				
2.2	All users should be able to edit their profile information.	UN, I, EE	P	AV, JR
2.3	All users should be able to view other public user's achievements through their dashboard.	UN	P	AV
3 - Could Have				
3.2	Learner users could receive notifications about their progress and streak reminders.	UN, EE	P	AV, JR

11. Project Management

a. Estimated & Actual Hours

	Estimated Hours and Assignment				
1.0 Learning	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
1.1 Team Meetings (Cumulative)	25	25	25	25	25
1.2 The developers must be comfortable developing in React	10	10	10	2	20
1.3 The developers must be comfortable developing in NodeJS	10	5	5	1	10
1.4 The developers must be comfortable	5	5	5	1	5

using SQL					
1.5 The developers must be comfortable with the CI/CD workflow	5	6	6	4	10
1.6 The developers must learn the fundamentals of Docker	5	10	10	4	5
1.7 The developers must research relavant popular React/NodeJS libraries	5	5	15	5	5
Total number of Hours Assigned per Team Member	**65**	**66**	**76**	**42**	**80**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**5.91**	**6.00**	**6.91**	**3.82**	**7.27**

	Actual Hours and Assignment				
1.0 Learning	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
1.1 Team Meetings (Cumulative)	25	30	25	25	41
1.2 The developers must be comfortable developing in React	10	12	10	10	7
1.3 The developers must be comfortable developing in NodeJS	25	10	5	5	5
1.4 The developers must be comfortable using SQL	5	3	3	2	5
1.5 The developers must be comfortable with the CI/CD workflow	15	5	5	20	25
1.6 The developers must learn the fundamentals of Docker	10	15	15	2	20
1.7 The developers must research relavant popular React/NodeJS libraries	1	5	15	10	3
Total number of Hours per Team Member	**91**	**80**	**78**	**74**	**106**
Weekly Average of Hours per Team Member (11 weeks)	**8.27**	**7.27**	**7.09**	**6.73**	**9.64**

	Estimated Hours and Assignment				
2.0 Front-end Design	**Joe**	**Archita**	**Gyumin**	**Jason**	**Alrick**

	Gaspari	Gattani	Moon	Ramos	Vincent
2.1 Research and conduct case studies on current market platforms	3	5	5	3	2
2.2 Design relevant userflow and user experience diagrams	10	10	5	5	15
2.3 Produce color themes, typography, brand elements	10	15	10	5	10
2.4 Produce initial design mockups	10	20	10	15	15
2.5 Gather and implement client feedback for final interface	11	11	11	4	11
Total number of Hours Assigned per Team Member	**44**	**61**	**41**	**32**	**53**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**4.00**	**5.55**	**3.73**	**2.91**	**4.82**

	Actual Hours and Assignment				
2.0 Front-end Design	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
2.1 Research and conduct case studies on current market platforms	2	5	5	4	2
2.2 Design relevant userflow and user experience diagrams	5	15	5	8	7
2.3 Produce color themes, typography, brand elements	10	5	10	5	3
2.4 Produce initial design mockups	15	35	10	35	5
2.5 Gather and implement client feedback for final interface	15	10	11	10	10
Total number of Hours per Team Member	**47**	**70**	**41**	**62**	**27**
Weekly Average of Hours per Team Member (11 weeks)	**4.27**	**6.36**	**3.73**	**5.64**	**2.45**

	Estimated Hours and Assignment				
3.0 Back-end Design	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
3.1 Define system architecture	5	5	5	5	5

3.2 Design routing	5	5	5	5	5
3.3 Design relational database schema	10	10	10	10	10
3.4 Design API & Format API requests	10	5	5	5	10
3.5 Generate test data	5	2	2	2	2
3.6 Design Open AI integration	5	5	5	0	5
3.7 Design question confidence engine	10	5	5	2	10
Total number of Hours Assigned per Team Member	**50**	**37**	**37**	**29**	**47**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**4.55**	**3.36**	**3.36**	**2.64**	**4.27**

	Actual Hours and Assignment				
3.0 Back-end Design	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
3.1 Define system architecture	5	0	5	0	15
3.2 Design routing	4	0	5	1	5
3.3 Design relational database schema	8	0	10	8	10
3.4 Design API & Format API requests	4	0	7	1	5
3.5 Generate test data	10	0	5	5	8
3.6 Design Open AI integration	2	0	0	8	2
3.7 Design question confidence engine	5	0	0	0	0
Total number of Hours per Team Member	**38**	**0**	**32**	**23**	**45**
Weekly Average of Hours per Team Member (11 weeks)	**3.45**	**0.00**	**2.91**	**2.09**	**4.09**

	Estimated Hours and Assignment				
4.0 Front-end Implementation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
4.1 Prepare Routing System	2	5	5	4	4
4.2 Implement initial sematic site structure through JSX	2	25	30	15	2

4.3 Implement site styling through Tailwind	2	15	15	15	4
4.4 Implement site interactions through JSX	2	30	65	50	4
4.5 Refactoring and modularization	15	25	35	15	15
4.6 Integration with back-end API	10	10	15	8	10
Total number of Hours Assigned per Team Member	**33**	**110**	**165**	**107**	**39**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**3.00**	**10.00**	**15.00**	**9.73**	**3.55**

	Actual Hours and Assignment				
4.0 Front-end Implementation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
4.1 Prepare Routing System	0	2	10	0	0
4.2 Implement initial sematic site structure through JSX	0	30	40	20	0
4.3 Implement site styling through Tailwind	0	25	20	20	0
4.4 Implement site interactions through JSX	0	30	65	60	0
4.5 Refactoring and modularization	0	10	55	20	0
4.6 Integration with back-end API	5	5	23	10	2
Total number of Hours per Team Member	**5**	**102**	**213**	**130**	**2**
Weekly Average of Hours per Team Member (11 weeks)	**0.45**	**9.27**	**19.36**	**11.82**	**0.18**

	Estimated Hours and Assignment				
5.0 Back-end Implementation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
5.1 Prepare Routing System	15	5	2	10	15
5.2 Dockerize SQL database, with appropriate schema	15	3	2	0	15
5.3 Build question confidence engine	20	2	1	0	20
5.4 Format data returning from Open AI API for the front-end	15	0	1	0	15
5.5 Develop server-side validation and error handling	10	0	0	0	10

5.6 Build data recall queries for SQL database	10	2	3	0	15
5.7 REST API creation	25	5	3	5	25
Total number of Hours Assigned per Team Member	**110**	**17**	**12**	**15**	**115**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**10.00**	**1.55**	**1.09**	**1.36**	**10.45**

	Actual Hours and Assignment				
5.0 Back-end Implementation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
5.1 Prepare Routing System	22	0	5	6	16
5.2 Dockerize SQL database, with appropriate schema	10	0	0	0	20
5.3 Build question confidence engine	5	0	0	0	0
5.4 Format data returning from Open AI API for the front-end	2	0	1	1	0
5.5 Develop server-side validation and error handling	10	0	1	1	15
5.6 Build data recall queries for SQL database	25	0	3	5	31
5.7 REST API creation	50	0	6	5	70
Total number of Hours per Team Member	**124**	**0**	**16**	**18**	**152**
Weekly Average of Hours per Team Member (11 weeks)	**11.27**	**0.00**	**1.45**	**1.64**	**13.82**

	Estimated Hours and Assignment				
6.0 Test & Testing	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
6.1 Regression testing	2	0	2	0	2
6.2 Performance testing	2	5	5	2	2
6.3 Unit testing	20	20	20	20	20
6.4 Integration testing	10	10	10	4	10

6.5 System testing	5	3	5	3	2
6.6 Usability testing	10	5	5	5	10
6.7 User Acceptance testing	10	5	5	3	10
Total number of Hours Assigned per Team Member	**59**	**48**	**52**	**37**	**56**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**5.36**	**4.36**	**4.73**	**3.36**	**5.09**

	Actual Hours and Assignment				
6.0 Test & Testing	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
6.1 Regression testing	2	0	0	0	2
6.2 Performance testing	2	0	0	0	2
6.3 Unit testing	40	5	5	5	30
6.4 Integration testing	0	2	2	25	0
6.5 System testing	10	0	0	0	7
6.6 Usability testing	5	5	5	5	10
6.7 User Acceptance testing	5	5	5	5	2
Total number of Hours per Team Member	**64**	**17**	**17**	**40**	**53**
Weekly Average of Hours per Team Member (11 weeks)	**5.82**	**1.55**	**1.55**	**3.64**	**4.82**

	Estimated Hours and Assignment				
7.0 Reports and Documentation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
7.1 Scope & charter document	2	2	2	2	2
7.2 Design document	10	10	10	10	10
7.3 Weekly team/personal logs	5	5	5	5	5
7.4 Mock ups/Diagrams	5	5	5	5	5
7.5 Final Report	20	20	20	20	20
7.8 Practice final presentation	5	5	5	5	5
Total number of Hours Assigned per Team Member	**47**	**47**	**47**	**47**	**47**

Weekly Average of Hours Assigned per Team Member (11 weeks)	**4.27**	**4.27**	**4.27**	**4.27**	**4.27**

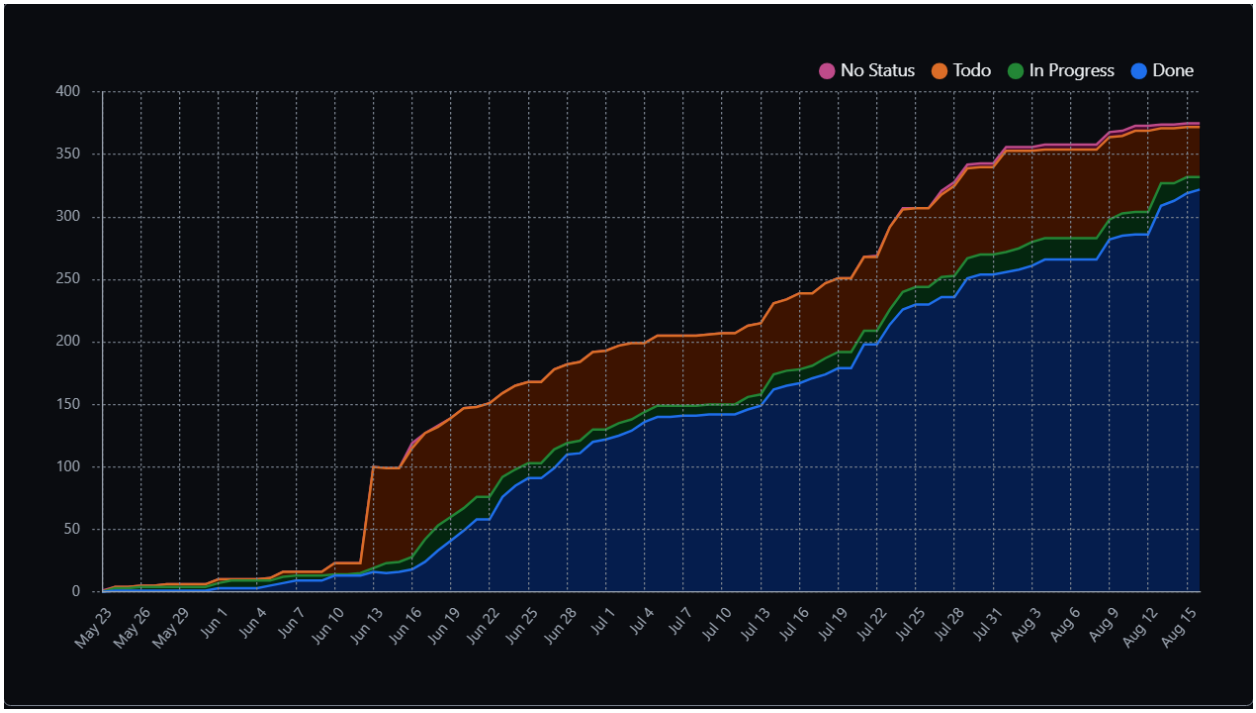
	Actual Hours and Assignment				
7.0 Reports and Documentation	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
7.1 Scope & charter document	5	10	2	5	7
7.2 Design document	10	15	5	5	14
7.3 Weekly team/personal logs	4	5	4	4	5
7.4 Mock ups/Diagrams	5	25	5	30	10
7.5 Final Report	5	30	1	5	3
7.8 Practice final presentation	2	5	1	1	5
Total number of Hours per Team Member	**31**	**90**	**18**	**50**	**44**
Weekly Average of Hours per Team Member (11 weeks)	**2.82**	**8.18**	**1.64**	**4.55**	**4.00**

	Estimated Hours and Assignment				
8.0 Deployment	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
8.1 Build and maintain CI/CD workflow with Github actions	10	5	5	6	10
8.2 Set up a Digital Ocean host for system	10	5	2	2	10
8.3 Post-deployment monitoring	5	5	2	3	5
Total number of Hours Assigned per Team Member	**25**	**15**	**9**	**11**	**25**
Weekly Average of Hours Assigned per Team Member (11 weeks)	**2.27**	**1.36**	**0.82**	**1.00**	**2.27**

	Actual Hours and Assignment				
8.0 Deployment	**Joe Gaspari**	**Archita Gattani**	**Gyumin Moon**	**Jason Ramos**	**Alrick Vincent**
8.1 Build and maintain CI/CD workflow with	10	5	3	25	20

Github actions					
8.2 Set up a Digital Ocean host for system	2	0	0	0	2
8.3 Post-deployment monitoring	0	0	0	0	2
Total number of Hours per Team Member	**12**	**5**	**3**	**25**	**24**
Weekly Average of Hours per Team Member (11 weeks)	**1.09**	**0.45**	**0.27**	**2.27**	**2.18**

b. Burn up Chart



12. Reflection

a. Learnings

- **Effective Communication:** We quickly realized that clear and open communication is important. Regular team meetings, both in-person and virtually, allowed us to discuss project progress, share ideas, and address challenges. We learned the value of active listening, respectful feedback, and the importance of staying connected to ensure everyone was on the same page.

- **Team Collaboration:** Working together as a cohesive unit was essential. Collaborative tools and platforms enabled us to share files, code, and resources seamlessly. We discovered the power of leveraging each team member's strengths, distributing tasks effectively, and supporting one another to achieve our common goal.

- **Role Definition:** Assigning specific roles and responsibilities early on helped streamline our project. Designating roles such as project manager, developer, researcher, and tester allowed us to focus on tasks aligned with our skills. This taught us the significance of role clarity in maximizing efficiency.

- **Adaptability:** As we progressed, we encountered unexpected challenges and changes in project scope. Learning to adapt and pivot when necessary was crucial. We grew adept at adjusting our strategies, managing unforeseen roadblocks, and finding creative solutions to keep the project on track.

- **Time Management:** Balancing individual commitments alongside the project's demands taught us the importance of effective time management. We honed our ability to set realistic deadlines, allocate time for different tasks, and prioritize efficiently to ensure project milestones were met.

- **Documentation and Tracking:** Maintaining clear and organized documentation was important. We learned to document our progress, decisions, and code changes thoroughly.

- **Quality Assurance and Testing:** Rigorous testing and quality assurance became integral parts of our workflow. We understood the significance of identifying and rectifying errors early on, ensuring a polished final product.

- **Celebrating Achievements:** Recognizing and celebrating milestones, no matter how small, fostered a positive team dynamic. Acknowledging our achievements motivated us to stay focused and maintained a sense of accomplishment throughout the project.

b. What could be done differently

- **Timeline:** Establishing a realistic timeline is crucial. While we all want to complete projects quickly, setting overly ambitious deadlines can lead to rushed work, increased stress, and a higher likelihood of errors. A well-planned timeline should consider the complexity of tasks, potential roadblocks, and available resources.

- **Risk Assessment:** Identifying potential risks and challenges early on allows for better contingency planning. Realistic expectations acknowledge that not everything will go as planned and account for possible setbacks.

- **Quality vs. Perfection:** While striving for excellence is important, it's essential to strike a balance between delivering a high-quality product and aiming for perfection. Realistic expectations recognize that perfection can be time-consuming and may not always be feasible within the project's constraints.

13. Handover Guide

a. Workflow

We have two Continuous integration (CI) workflows set up for this project on GitHub Actions. Both workflows utilize Docker to bring up the required containers and run tests.

1. Back-end Workflow
    - Purpose: To test the back-end code, running unit testing and regression testing on existing codebase
    - Tools: Jest and Supertest
    - Steps:
        i. When new code is pushed to the master branch, or a pull request is started for the master branch, the CI workflow will automatically trigger.
        ii. The required dependencies will be installed on the Ubuntu virtual machine
        iii. Docker will bring up two containers, one being the node application and the other the test database
        iv. The back-end tests will be executed using Jest and Supertest.
        v. The results of the tests will be displaying in the job logs, either passing or failing the checks

2. Front-end Workflow
    - Purpose: End-to-end and integration testing
    - Tools: Selenium and Mocha
    - Steps:
        i. When new code is pushed to the master branch, or a pull request is started for the master branch, the CI workflow will automatically trigger.
        ii. The required dependencies will be installed on the Ubuntu virtual machine, including Google Chrome Drivers
        iii. Docker will bring up all the containers, the React application, node application and test database

iv. The front-end tests will be executed through the Chrome headless browser with Selenium
v. The results of the tests will be displayed in the job logs, either passing or failing the checks.
vi. Any failing tests will have their pages automatically screenshotted and saved in the summary of the Github Actions workflow.

b. Deployment

Currently, we have a manual deployment for this project

1. Deployment Target: Digital Ocean Droplet
2. Steps
    a. Open a local terminal and SSH into the Digital Ocean Droplet
    b. Navigate to the directory where the repository is located. If the repository is not already cloned, clone it to the desired directory using git clone <http clone link>
    c. If the repository is already cloned, run git pull after navigating to the directory.
    d. Execute the command docker-compose down to stop any running containers
    e. Run the command docker-compose up -d to restart the containers with the latest changes

Please note that at the time of writing this report, an automated deployment process is in the works, but not yet operational, hence the solution is not described here.

c. Code

Parsa Rajabi (the client) has been provided access to the repository.

d. User Manual/User Documentation

Student view:

1. Sign up

Follow these steps to create your account and begin your learning journey.

1. Access our platform's website.
2. Locate the "Sign Up" button on the top right of the page.
3. Make sure the toggle is on the Student Sign up.
4. Provide your first and last name, username, email, and password.
5. Click "Get started" to complete registration.

2. **Sign in**

Use your credentials to access the platform. Here's how:

1. Visit our platform's website.
2. Click the "Sign In" button on the top right of the page.
3. Enter your registered email or username and password.
4. Click "Get started" to access your account.

3. **Dashboard**

Your dashboard is your central hub for managing your learning experience. Here's an overview:

Features:

- Upcoming milestones: This section provides you with a preview of the upcoming milestones, such as deadlines for assignments, quizzes, or projects.
- Solve rate: Shows the problems solved, problems attempted, total hours and average hours per day.
- Badges: Earn badges as you achieve milestones or demonstrate proficiency in specific areas.
- Leaderboard: See where you stand in comparison to other learners, fostering a healthy sense of competition and motivation.
- Navbar: Navigate to different sections such as Explore, Modules, Analytics, Information Center and Account Settings.
- Notifications

4. **Explore**

Explore the available topics and languages. Here's how:

1. Visit the "Explore" section in the navbar.
2. Use the filter search bar on the top right to search for a particular topic of interest or filter your choices by the language of interest.
3. Click on the topic of interest.
4. Select a language and difficulty, then generate a question.
5. **Got stuck?** Use the hint button to help guide you through the question
6. **Want to come back later to the question?** Save your answer and come back again by clicking on the topics under “Attempted Questions”.
7. **Found a mistake/bug in a question?** Tell us what happened.
8. Submit your answer and get personalized feedback to help you excel at coding.

5. **Modules**

Access your enrolled courses and modules. Here's how to navigate and progress:

1. Open the "Modules" section.
2. Select a module to see its assignments.
3. Complete assignments and engage with material as you progress.

6. **Analytics**

Monitor your progress and performance using analytics. Here's how:

Features:

- Upcoming Assignments: Get a quick overview of pending assignments and their due dates.
- Average Time Spent/Questions Solved: Understand your study habits and productivity by tracking the time you spend and questions you solve on average.
- Assignments Completed: Keep track of the assignments you've successfully completed.
- Leaderboard: Compare your performance with others, encouraging improvement.

Track your learning journey over time.

7. **Account settings**

Customize your account settings. Here's how:

1. Click on your name located on the left bottom corner of the page.
2. Click on the information you would like to edit.
3. Click on the update account to save the new changes successfully.

8. **Join a class**

Join a class to view the instructor's created modules. Here’s how:

1. Click on the arrow next to the class name on the navbar.
2. Either search for the class or add the class code given by the instructor.
3. The class will be added and you can click to view the class modules.

Instructor View

1. **Add students**

The "Add Students" feature allows you to invite and manage students within your classes or groups. Follow these steps to add students:

1. Click on the "Add Students" option.
2. Generate a unique code for your class.
3. Share this code with your students.
4. Students enter the code on their end to join your class.

2. Create class

With the "Create Class" feature, you can establish new classes or groups to organize your students. Here's how to create a class:

1. Click on "Create Class."
2. Provide a class name and a class description.
3. Save the class details.

3. Create module

Instructors have the ability to create coding modules for their students. Modules help structure the learning experience. Here's how you can create a module:

1. Click on "Add a new Module."
2. Edit the module by clicking the pencil icon and name it.
3. Click on the plus icon next to the module to add a new assignment.
4. Edit the assignment and fill in the details:
    - Name
    - Number of questions
    - Topic
    - Difficulty
    - Language
    - Deadline

Optional: Click on the eye to hide the module for the students

4. Students

The "Students" feature helps the instructor to monitor every student's progress and performance.

1. Navigate to the Students section.

2. View individual student progress by clicking on the student's name.

5. Analytics

Introduction:
Monitor the class's progress and performance using analytics. Here's how:

Features:

- Upcoming Assignments: Get a quick overview of pending assignments and their due dates.
- Average Time Spent/Questions Solved: Understand your study habits and productivity by tracking the time you spend and questions you solve on average.
- Assignments Completed: Keep track of the assignments you've successfully completed.
- Leaderboard: Compare your performance with others, encouraging improvement.

6. Modules

The "Modules" section allows you to manage and edit the existing modules. Here's what you can do:

1. Browse through the list of created modules.
2. Edit module content.
3. Adjust difficulty levels, due dates, number of questions and update module details.

e. Developer's Instructions

These instructions will guide you through setting up and running the system using Docker. Please follow each step carefully to ensure a smooth setup.

Prerequisites

Before you begin, make sure you have the following prerequisites installed on your system:
Docker: Ensure Docker is installed and running on your machine.

Step 1: Clone Repository

1. Navigate to the directory where you want to clone the repository.
2. Run the following command to clone the repository:

3. ‘git clone https://github.com/UBCO-COSC-499-Summer-2023/project-6-gamified-coding-practice-platform-gamifygeeks.git’

Step 2: Configure Permissions

1. Navigate to the src directory of the cloned repository.
2. Create a file named .env if it doesn't already exist.
3. Open the .env file in a text editor.
4. Set the following environment variables with the provided values:

JWT_SECRET=e22a414ff76ff7cb37059f724239d25ebf7b486f5dfc180f3b6eb8d637584243
JWT_EXPIRES_SEC=86400
BCRYPT_SALT_ROUNDS=12
HOST_PORT=8080
OPENAI_API_Key= <INSERT YOUR OPENAI KEY>
EMAIL=gamifygeeks@gmail.com
EMAIL_PASSWORD=srejabnmyvivxrks

Step 3: Run Docker Containers

1. In the terminal, navigate to the src directory of the cloned repository.
2. Run the following command to start the Docker containers in detached mode:
3. ‘docker-compose up -d’

This command will start all the services defined in the `docker-compose.yml` file in detached mode, meaning the containers will run in the background independently of your terminal

Due to the way Docker Compose starts services, the Node.js service might start before the MySQL service is fully ready.

If you see the following connection errors in the gamify-node service logs:

```
[nodemon] starting `node app.js`
Server is running on port 8080
Unable to connect
```

You may need to restart the gamify-node service by clicking the restart button or running the command in /src `docker-compose restat gamify-node`.

You will know the connection is successful if you see the following in the gamify-node service logs:

```
Server is running on port 8080
Executing (default): SELECT 1+1 AS result
Connection has been established!
```

The application should now be running and accessible on `http://localhost.8080` for the gamify-node and `http://localhost:3000` for the gamify-react front-end. To stop the application at any point you can simply go in the Docker GUI and click the Stop square button or run the command `docker-compose down`. This command will stop and remove the containers.

f. Front-end Dependencies

Package Name	Version	Description	Link
@tanstack/react-query	^4.29.12	React Query library for managing asynchronous data	https://github.com/tannerlinsley/react-query
@testing-library/jest-dom	^5.16.5	Custom jest matchers for DOM testing	https://github.com/testing-library/jest-dom
@testing-library/react	^13.4.0	React testing utilities	https://github.com/testing-library/react-testing-library
@testing-library/user-event	^13.5.0	Synthetic event simulation library for testing	https://github.com/testing-library/user-event
@uiw/codemirror-extensions-langs	^4.21.4	CodeMirror language extensions by UIW	https://github.com/uiwjs/codemirror-extensions-langs
@uiw/codemirror-themes-all	^4.21.4	All themes for CodeMirror by UIW	https://github.com/uiwjs/codemirror-themes-all

@uiw/react-codemirror	^4.21.4	React wrapper for CodeMirror by UIW	https://github.com/uiwjs/react-codemirror
antd	^5.7.0	A popular UI library for React applications	https://github.com/ant-design/ant-design
chart.js	^4.3.0	JavaScript charting library	https://github.com/chartjs/Chart.js
react	^18.2.0	React library for building user interfaces	https://github.com/facebook/react
react-accessible-accordion	^5.0.0	Accessible accordion component for React	https://github.com/springload/react-accessible-accordion
react-calendar	^4.3.0	Calendar component for React applications	https://github.com/wojtekmaj/react-calendar
react-chartjs-2	^5.2.0	React wrapper for Chart.js	https://github.com/reactchartjs/react-chartjs-2
react-dom	^18.2.0	React package for working with the DOM	https://github.com/facebook/react
react-icons	^4.9.0	SVG icons for popular icon packs in React	https://github.com/react-icons/react-icons
react-input-mask	^3.0.0-alpha.2	Input masking for React forms	https://github.com/sanniassin/react-input-mask
react-loader-spinner	^5.3.4	Loader spinner component for React	https://github.com/mhnpd/react-loader-spinner
react-markdown	^8.0.7	Markdown rendering for React	https://github.com/remarkjs/react-markdown
react-movable	^3.0.4	Drag-and-drop and swipe functionality for React components	https://github.com/tajo/react-movable
react-router-dom	^6.11.2	Declarative routing for React applications	https://github.com/remix-run/react-router

react-scripts	5.0.1	Scripts and configuration for React development	https://github.com/facebook/create-react-app
react-toastify	^9.1.3	Toast notifications for React	https://github.com/fkhadra/react-toastify
react-tooltip	^5.15.0	Tooltip library for React applications	https://github.com/wwayne/react-tooltip
react-transition-group	^4.4.5	Animation utility for managing transitions in React	https://github.com/reactjs/react-transition-group
selenium	^2.20.0	Web testing framework	https://www.selenium.dev/
selenium-webdriver	^4.10.0	WebDriver library for controlling web browsers	https://www.selenium.dev/selenium/docs/api/javascript/module/selenium-webdriver/
styled-components	^6.0.1	Styling library for React components using tagged template literals	https://styled-components.com/
tailwind	^4.0.0	Utility-first CSS framework	https://tailwindcss.com/
tailwindcss-animation-delay	^1.0.7	Tailwind CSS plugin for adding animation delay utilities	https://github.com/pingyhq/tailwindcss-animation-delay
web-vitals	^2.1.4	Library for measuring web vitals	https://github.com/GoogleChrome/web-vitals

g. Back-end Dependencies

Package Name	Version	Description	Link
bcrypt	^5.1.0	Library for hashing passwords	https://github.com/kelektiv/node.bcrypt.js

cors	^2.8.5	Middleware for enabling CORS in Express applications	https://github.com/expressjs/cors
dotenv	^16.1.3	Loads environment variables from a .env file	https://github.com/motdotla/dotenv
express	^4.18.2	Web application framework for Node.js	https://github.com/expressjs/express
express-async-errors	^3.1.1	Express error handling for async functions	https://github.com/davidbanham/express-async-errors
express-validator	^7.0.1	Express middleware for input validation	https://github.com/express-validator/express-validator
helmet	^7.0.0	Security middleware for Express applications	https://github.com/helmetjs/helmet
jest	^29.5.0	JavaScript testing framework	https://github.com/facebook/jest
jsonwebtoken	^9.0.0	JSON Web Token library for Node.js	https://github.com/auth0/node-jsonwebtoken
morgan	^1.10.0	HTTP request logger middleware for Node.js	https://github.com/expressjs/morgan
mysql	^2.18.1	MySQL database driver for Node.js	https://github.com/mysqljs/mysql
mysql2	^3.3.5	Improved MySQL driver for Node.js	https://github.com/mysqljs/mysql
nodemailer	^6.9.4	Library for sending emails from Node.js applications	https://github.com/nodemailer/nodemailer
openai	^3.3.0	OpenAI API client for Node.js	https://github.com/openai/openai-node

sequelize	^6.32.0	Promisified ORM for SQL databases	https://github.com/sequelize/sequelize
supertest	^6.3.3	HTTP testing library for Node.js	https://github.com/visionmedia/supertest
@babel/core	^7.22.8	Babel compiler core	https://github.com/babel/babel
@babel/plugin-transform-runtime	^7.22.7	Babel plugin for using the runtime instead of inlined helpers	https://github.com/babel/babel/tree/main/packages/babel-plugin-transform-runtime
@babel/preset-env	^7.22.7	Babel preset for transforming JavaScript	https://github.com/babel/babel
eslint	^8.43.0	JavaScript linter	https://github.com/eslint/eslint
eslint-config-airbnb-base	^15.0.0	ESLint configuration following Airbnb's style guide	https://github.com/airbnb/javascript/tree/master/packages/eslint-config-airbnb-base
eslint-plugin-import	^2.27.5	ESLint plugin for linting import statements	https://github.com/benmosher/eslint-plugin-import
nodemon	^2.0.22	Utility for automatically restarting Node.js applications	https://github.com/remy/nodemon

14. Successful Outcome of Project

a. Core Requirements/Features Completed

i. Functional Requirements

Must Have:	
The system must be able to accept user feedback on questions and reward	**Complete**

feedback with bonus points and badges, which prompts system improvements.	
The system must be able to prompt AI-generated questions based on a specified topic, level of proficiency, and programming language chosen by the user.	**Complete**
The system must be able to track and display user progress through course milestones, user points, achievement badges, and usage analytics.	**Complete**
The system must allow instructors the ability to view student and class analytics including students' progress in the assigned questions, how many they have completed, how many they have left, percent success rate, visual dashboard of progress, number of questions they have completed, their streak/points and the number of badges earned.	**Complete**
The system must support user login, logout, registration, and password retrieval via email linked to the account via email linked to the account.	**Complete**
The system must be able to indicate whether user-provided answers are correct or incorrect and provide meaningful AI-generated feedback with high confidence e.g. 95%.	**Complete**
The system must be able to validate whether an AI-generated question relates to the topic selected with a high degree of confidence (95%)	**Complete**
The system must support JavaScript, Python, and HTML/CSS.	**Complete**
The system must support the main user archetype: learners.	**Complete**
The system must include tooltips to guide users.	**Complete**
The system must be visually appealing to the users with a modern design.	**Complete**
The system should be able to give users AI-generated suggestions or hints when prompted by the user or a series of incorrect answers.	**Complete**
The system should contain user dashboards for students showing course progress, badges earned, and other user achievements.	**Complete**
The system should support the main user archetype: Instructors.	**Complete**
The system should further support Java, C, C#, and C++ .	**Complete**

The system should have a class platform for instructors to create a set of topics and customize the parameters such as proficiency level and question format for the to-be-generated questions.	**Complete**
The system should allow learners to join classes through a join code similar to Kahoot platform.	**Complete**
The system should have a public leaderboard.	**Complete**
The system should easily accommodate additional programming languages for students to practice.	**Complete**
The system must be accessible through a publicly available online domain.	**Complete**
The system should include terms of service and privacy policy information.	**In progress**
Should have:	
The system should have an FAQ page that includes questions/answers that may be popular.	**Complete**

ii. Non-Functional Requirements

Must Have:	
The system must have a seamless and straightforward user interface that minimizes user clicks or actions to achieve the goal.	**Complete**
The system must be properly documented for future developers through written documents and in-line code documentation.	**Complete**
The system must have a guide for teachers and students on how to use the platform.	**Complete**
The developers must create test cases before writing or committing new additional features or components.	**Complete**
The developers must be comfortable using HTML, CSS, JS, React, NodeJS, and SQL.	**Complete**
The system must be responsive for all non-mobile devices.	**Complete**
The system must pass an accessibility audit (WCAG 2.1).	**Incomplete**

Should have:	
The system should not store personally identifiable information except email.	**Complete**
The AI-generated questions should have less than a 5% user report rate.	**Complete**
The system should allow instructors the ability to view the confidence score of each of the questions generated for the student.	**Incomplete**

iii. User Requirements

Must Have:	
All users must be able to register and login to the system.	**Complete**
Learner users must be able to choose their programming language and topic and level of proficiency.	**Complete**
Learner users must have an alias or username.	**Complete**
Learners must be able to view their streak while accessing the platform.	**Complete**
Instructor users must be able to create a class and share a join code.	**Complete**
Instructor users must be able to create a set of topics and customize the parameters such as proficiency level and question format for the to-be-generated questions.	**Complete**
Instructors must be able to create and keep track of multiple courses at the same time.	**Complete**
All users must have the ability to change their password and reset it through the email linked to their account.	**Complete**
Leaders must be able to see a timer when their streak is about to expire via the platform and/or email notification.	**Complete**
Should Have:	
All users should be able to edit their profile information.	**Complete**
Learner users should be able to emphasize chosen achievements through chosen badges and courses.	**Complete**
Learner users could have the option to opt into or out of public leaderboards.	**Complete**
All users should be able to delete their accounts.	**Complete**

All users should be able to view other public user's achievements through their dashboard.	**Complete**
Instructor users should be able to archive courses after the term has ended.	**Incomplete**

iv. Technical Requirements

Must Have:	**Complete**
The system must be developed using ReactJS for the front-end.	**Complete**
The system must be developed using NodeJS for the back-end.	**Complete**
The system must update asynchronously.	**Complete**
The system must integrate with an AI API to generate practice questions. The system must be dockerized for deployment.	**Complete**
The system must securely store user passwords through salting or hashing.	**Complete**
The system must have server-side security for user inputs.	**Complete**
The system's database must be a relational database.	**Complete**
The system must have validation for all user-input fields.	**Complete**
The system must make use of stable, popular libraries instead of in-house solutions.	**Complete**
The system must include server-side logging that captures all database queries and events that may lead to a failure.	**Complete**
Should Have:	
The system should indicate to users any errors found within the system.	**Complete**
The system should have client-side security for user inputs.	**Complete**

b. Pending Features

The following are the pending features of the platform that are included in the original requirements which may require more development time.

1. The system could have an additional user archetype: administrators.
2. The system could have an admin portal to view site usage analytics and (update/delete/add) users and classes.
3. The system could provide users with a dark theme option.
4. The system could be responsive for mobile devices.
5. Learner users could have personalized and customizable profile cards to emphasize favorite achievements.
6. Learner users could be alerted when class lessons are updated.
7. Learner users could be alerted about their course progress and streaks via notifications (either via email or via the web application).