

Project Report for Software Engineering

Submitted by

Milan [21f2001519]

Ritik Arora [21f1000635]

Vedanshi Tewari [21f2001522]

TAMANNA TAK [21f3002971]

Navdeep [21f1007036]

Shubham Gattani [21f3002082]

Rohan Ajay Ramani [21f3000618]

Tabish [21f2001449]

in partial fulfillment of the requirements for the degree of




**IITM Online BS Degree Program,
Indian Institute of Technology, Madras, Chennai
Tamil Nadu, India, 600036**

#	Topic/Content	Page No.
1	Honor Code	2
2	Problem Statement	3
3	Milestone 01 - Identifying Users of the Application	5
3.a	User Stories for New Features and Integrations	6
4	Milestone 02 - User Interfaces	9
4.a	Storyboard	9
4.b	Wireframes	13
5	Milestone 03 - Scheduling and Designing	18
5.a	Project Scheduling	18
5.b	Design	21
5.c	UML	25
6	Milestone 04 - API Endpoints	29
6.a	API development for User Stories	31
6.b	List of Created APIs	32
7	Milestone 05 : Test Suite and Test Cases	39
7.a	Test Cases for APIs	39
8	Code Review and Testing	53
9	Acknowledgement	58

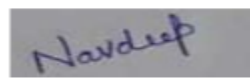
Team - 8 Honor Code

We, the members of team 8, declare that we will not use any ideas, writings, code or work that is not our own or our groups with the intention of claiming it our or our group's work. And all the work that we will submit as part of this project we will not share that outside our group with anybody directly or indirectly or upload that on any public forums on the internet.

We acknowledge that failing in any of the above constitutes as plagiarism and in that case the institute can take appropriate disciplinary action.

Signature: 
Date: 11/06/2024

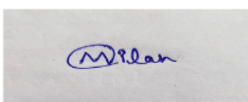
Ritik Arora
[21f1000635]

Signature: 
Date: 11/06/2024


Navdeep
[21f1007036]


Signature: _____
Date: 11/06/2024


Tabish
[21f2001449]

Signature: 
Date: 11/06/2024

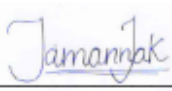
Milan
[21f2001519]

Signature: 
Date: 11/06/2024

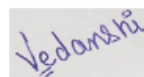
Rohan Ajay Ramani
[21f3000618]

Signature: 
Date: 11/06/2024

Shubham Gattani
[21f3002082]

Signature: 
Date: 11/06/2024

Tamanna Tak
[21f3002971]

Signature: 
Date: 11/06/2024

Vedanshi Tewari
[21f2001522]

Problem Statement :

Objective:

Enhance the SEEK learning portal by integrating Generative AI (GenAI) technologies to improve user experience and functionality. Few key objectives are:

- Implement GenAI for effective feedback on programming assignments.
- Integrate real-time GenAI support for coding problems.
- Enhance static content interactivity with GenAI.
- Develop a prototype of the GenAI-enhanced SEEK portal with one course.
- Utilize existing tools to streamline development.

Key Integrations:

- **Comprehensive Feedback:** Utilize GenAI to provide detailed feedback on programming tests, beyond just pass/fail indicators.
- **Real-Time Support:** Offer real-time hints, suggestions, and support for learners during coding practice sessions.
- **Interactive Content:** Transform static materials (PDFs, documents, videos) into interactive and engaging learning resources using GenAI.

Project Scope:

- Develop a GenAI-enhanced version of the SEEK portal focused on one pre-populated course. Use existing libraries, templates, and APIs (e.g., ACE code editor, online compiler API, and open-source LLM model APIs like Ollama).

Expected Outcomes:

- A prototype SEEK portal with GenAI enhancements for one course.
- Comprehensive GenAI feedback on programming assignments.
- Real-time GenAI support in the code editor.
- Interactive and engaging static content using GenAI.
- Best practices for integrating GenAI into learning environments.

Milestone 01

Identifying Users of the Application

When developing a GenAI-enhanced learning environment like the SEEK portal, it is essential to identify the different user groups who will interact with the application. Understanding these user groups help in tailoring the features and user experience to meet their specific needs. Users can be categorized into primary, secondary, and tertiary users.

Primary Users

- **Students/Learners:** Engage with learning materials and need feedback, real-time support, and personalized learning paths.
- **Instructors/Teachers:** Design course content and require tools for monitoring progress, generating insights, and providing feedback.

Secondary Users

- **Teaching Assistants (TAs):** Assist with course management and need access to student submissions and efficient grading tools.
- **Educational Administrators:** Oversee the learning environment and require analytical tools and insights for system assessment.

Tertiary Users

- **Parents/Guardians:** Interested in their children's academic progress and need access to performance reports.
- **Future Employers:** Verify skills and curriculum insights of graduates, using the portal rarely during recruitment.
- **Academic Researchers:** Study the impact of GenAI on education and need access to data on student performance and engagement.

User Stories for New Features and Integrations

1. As a learner,

I want to see a summary of the entire week's content displayed prominently at the beginning of each week,
So that I can quickly review the key topics covered.

2. As a learner,

I want the option to generate a summary for any specific video after watching it,
So that I can reinforce my understanding of the video content.

3. As a learner,

I want the system to generate questions based on the week's material, accessible with a Click, So that I can assess my understanding and retention.

4. As a learner,

I want to be able to generate questions specific to any video I watch,
So that I can test my comprehension immediately.

5. As a learner,

I want the system to provide personalized feedback on my programming assignments, using GenAI to highlight areas for improvement beyond basic test case results,
So that I can improve my skills effectively.

6. As a learner,

I want GenAI to assist me in real-time while I solve programming problems, offering hints and suggestions based on the code I'm writing,
So that I can receive immediate help and improve my coding skills.

7. As an instructor,

I want to review insights generated by GenAI on learner progress and challenges,
So that I can tailor my teaching approach and support students effectively.

8. As an administrator,

I want to ensure that all GenAI-enhanced features undergo rigorous unit testing,
So that I can verify their functionality and reliability.

9. As a learner,

I want GenAI to add interactive elements to static content such as PDFs and documents,
So that the learning materials are more engaging and easier to comprehend.

10. As a learner,

I want the system to recommend additional resources or learning paths based on my
performance and learning patterns, leveraging GenAI insights,
So that I can follow a personalized learning journey.

11. As a learner,

I want GenAI to analyze my programming quiz attempts and provide detailed feedback
on coding style, efficiency, and best practices,
So that I can enhance my programming skills.

12. As an administrator,

I want to monitor the performance and scalability of the GenAI-enhanced features,
So that I can ensure they meet the demands of our learner community.

13. As a learner,

I want the system to suggest personalized study schedules or paths based on my learning
pace and preferences, facilitated by GenAI capabilities, So that I can optimize my study time.

14. As a learner,

I want GenAI to generate personalized study recommendations based on my learning
preferences, previous quiz results, and identified knowledge gaps,
So that I can focus on areas that need improvement.

15. As a learner,

I want the option to receive automated reminders and notifications from GenAI about
upcoming deadlines for assignments and quizzes, So that I can stay on track with my coursework.

16. As an instructor,

I want to utilize GenAI to automatically evaluate and provide feedback on code
submissions, including identifying common errors and suggesting improvements,
So that I can offer more detailed and helpful feedback to students.

17. As a learner,

I want GenAI to generate interactive simulations or visualizations to help me understand
complex programming concepts or algorithms, So that I can grasp difficult topics more easily.

18. As an administrator,

I want to implement robust security measures and access controls for GenAI-enhanced features, So that I can protect learner data and ensure privacy.

19. As a learner,

I want to use GenAI-powered chatbots or virtual assistants to ask questions, receive immediate answers, and get guidance on course-related queries,
So that I can get help quickly and efficiently.

21. As an instructor,

I want to leverage GenAI to create adaptive learning paths that adjust based on learner Progress, So that I can provide personalized learning experiences.

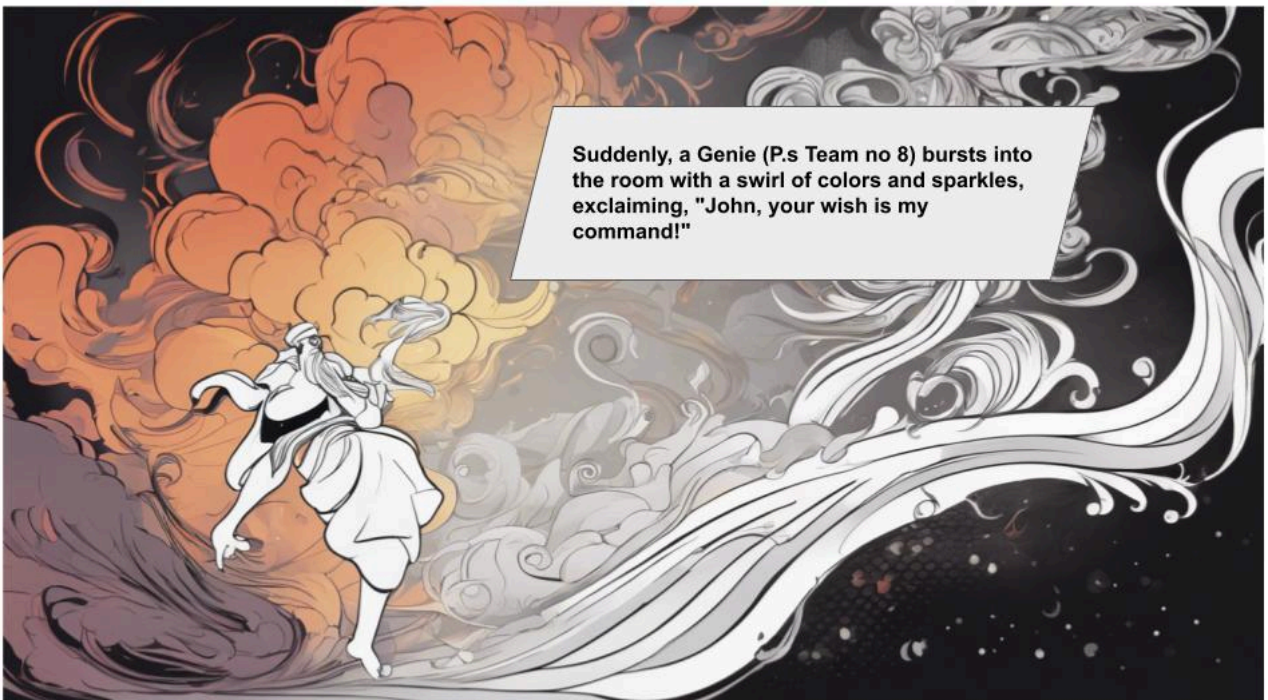
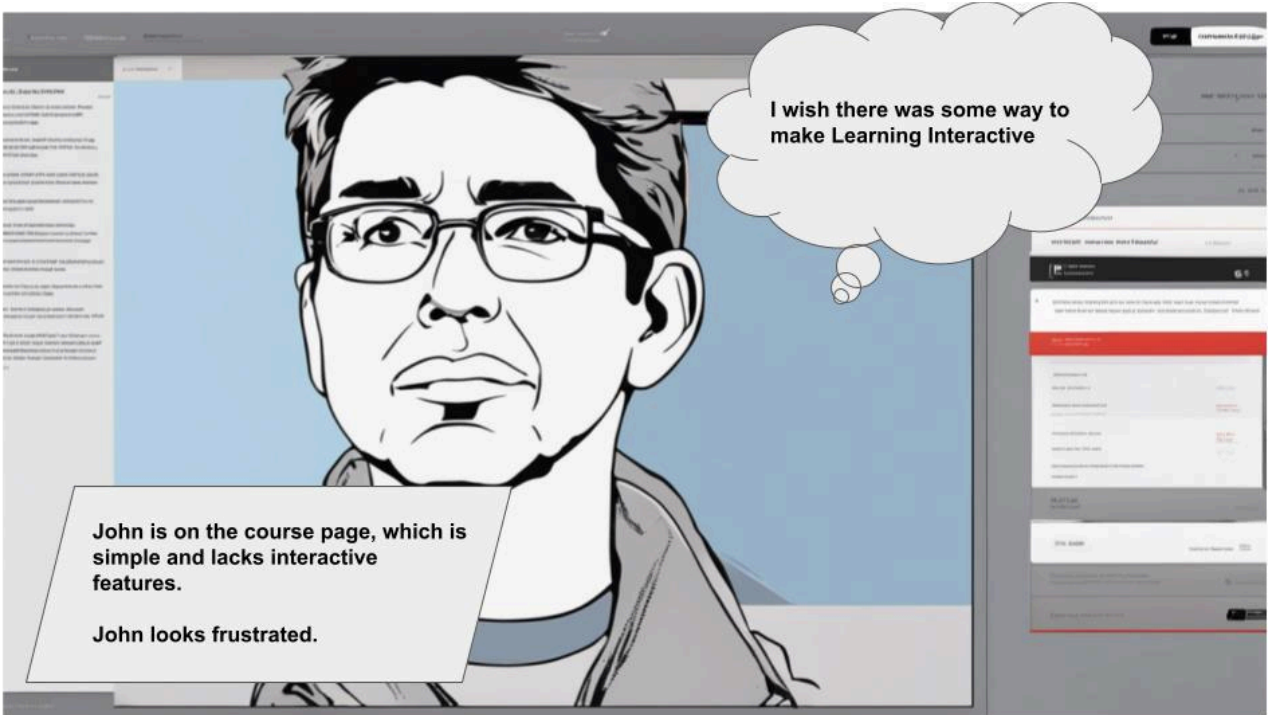
22. As a learner,

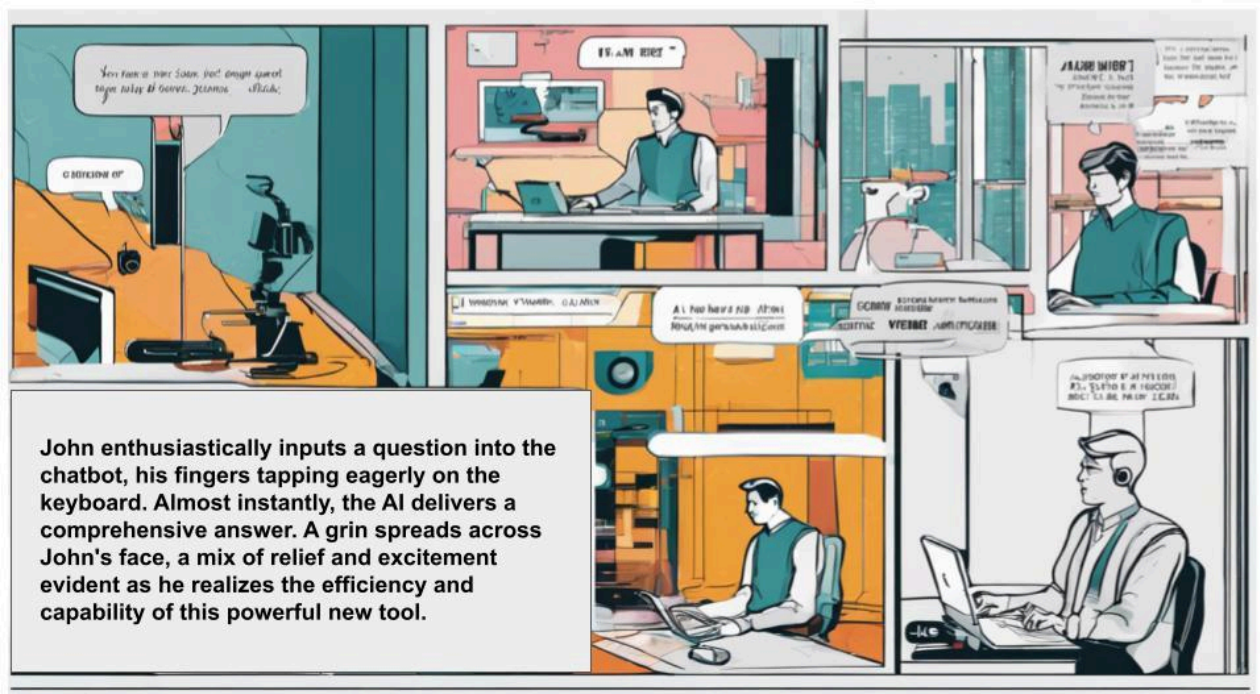
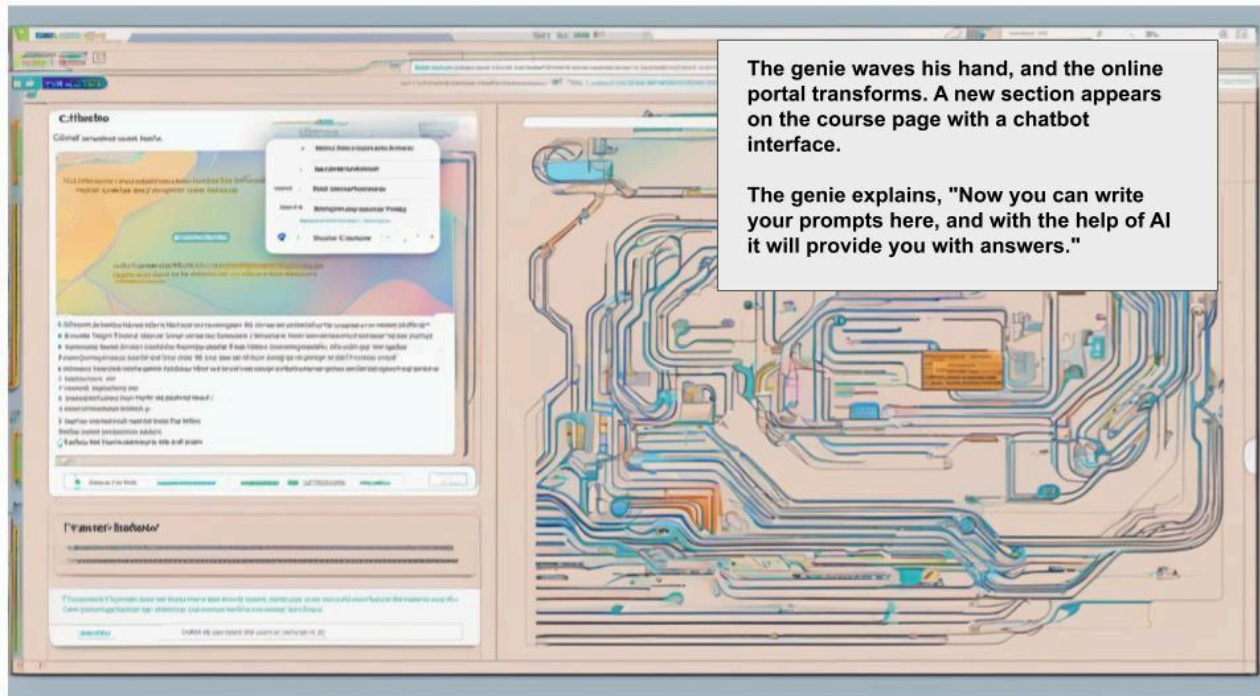
I want to use GenAI for collaborative coding sessions, where it can facilitate real-time code reviews, debugging assistance, and pair programming support,
So that I can enhance my learning through collaboration.

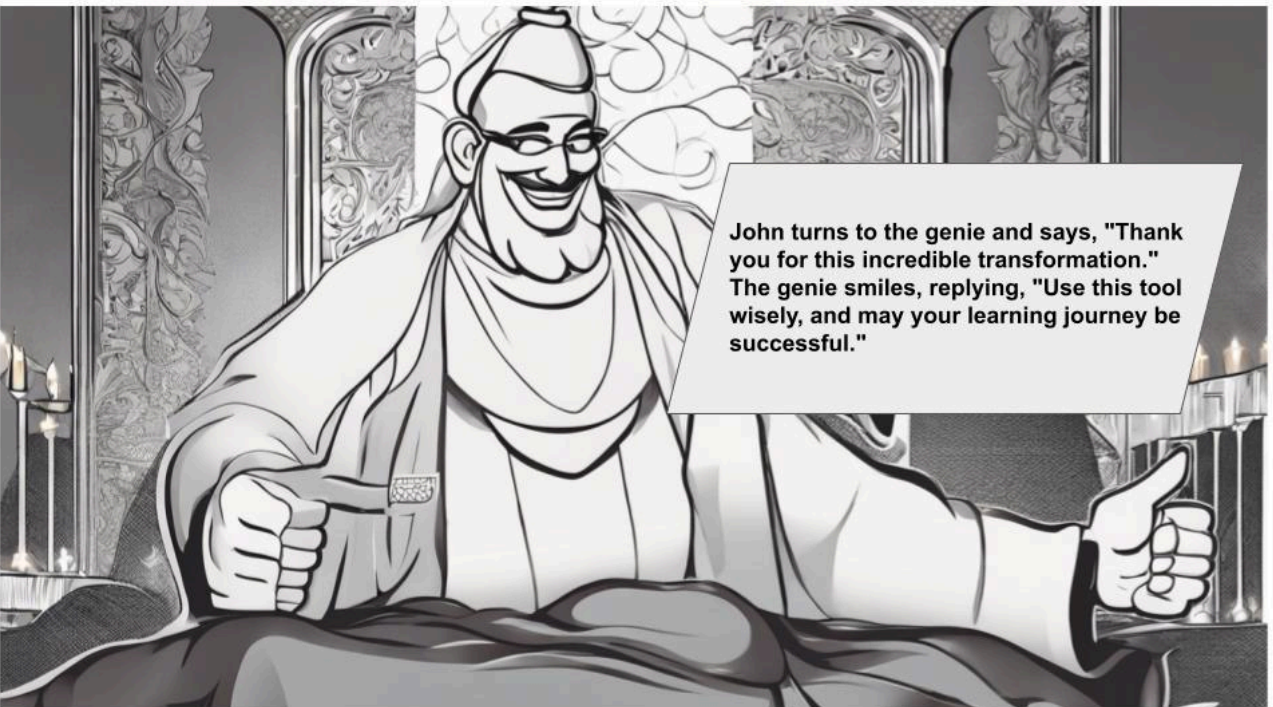
Milestone 02: User Interfaces

Storyboard:







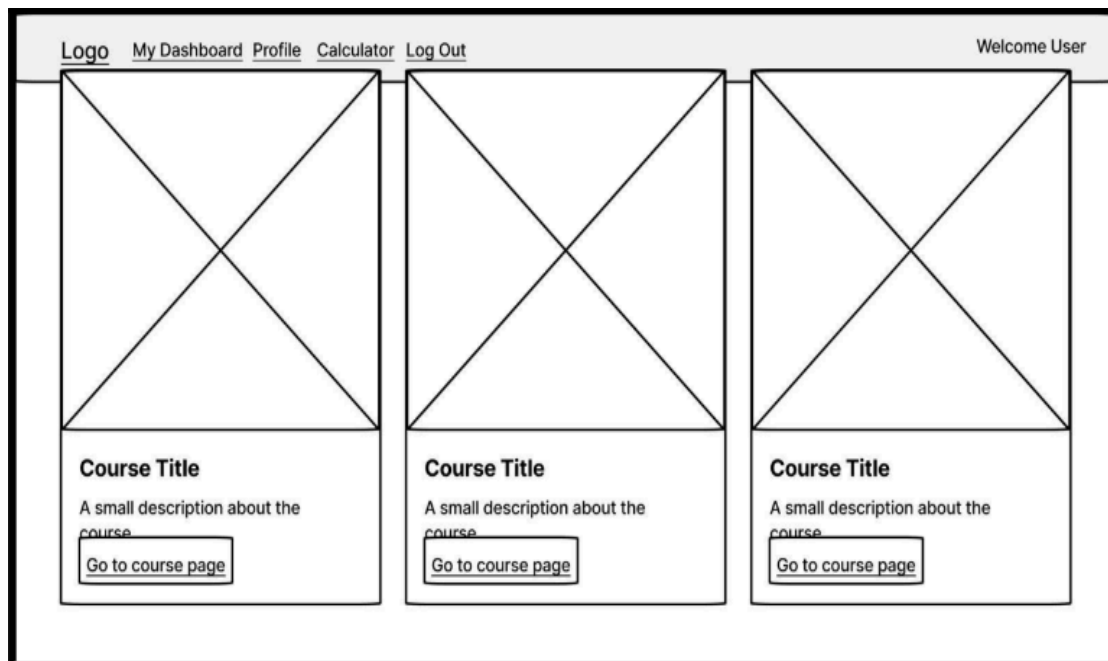


Wireframes

These low-fi wireframes cover all the selected user stories. These are intended to convey the information architecture and usability of each UI/screen.

1. Update Course Description with GenAI

Here, we will find a detailed description of the course, outlining key objectives and learning outcomes. Stay informed with a schedule of upcoming lectures, each meticulously designed to deepen our understanding of the subject. Dive in to enhance our knowledge and achieve Academics.



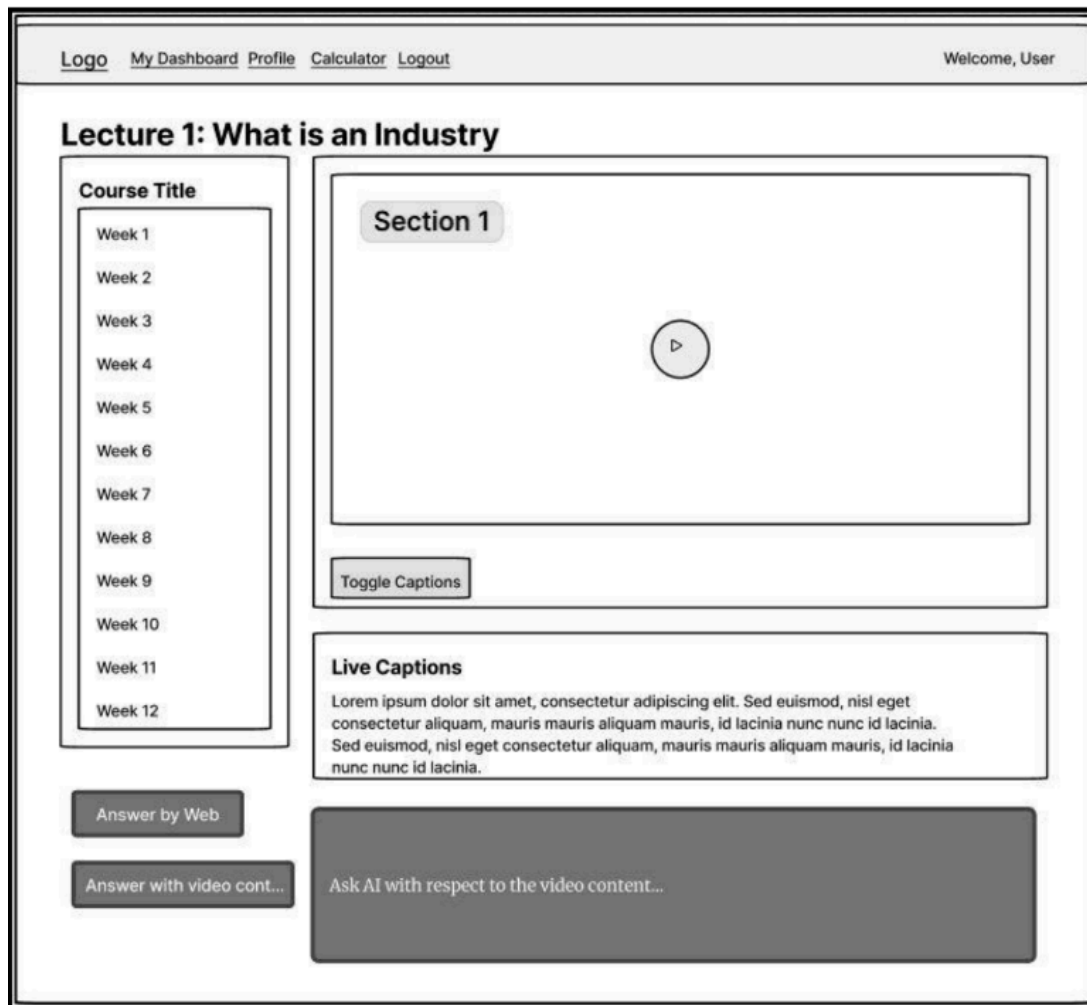
2. Course Overview / Tracker

Summarized form of weeks and video for quick recap and revision.

Logo		Welcome User My Dashboard Profile Calculator	
	Weekly Summary Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla nec elit auctor, incidunt nunc id, aliquam nisl. Sed euismod, nunc id aliquet tincidunt, nunc nunc lacinia nunc, id ligam nunc nunc id. <input type="text"/>		<input type="button" value="Mark as Completed"/> <input type="button" value="Generate Summary"/>
	Video Summary Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla nec elit auctor, incidunt nunc id, aliquam nisl. Sed euismod, nunc id aliquet tincidunt, nunc nunc lacinia nunc, id ligam nunc nunc id. <input type="text"/>		
	Video Summary Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla nec elit auctor, incidunt nunc id, aliquam nisl. Sed euismod, nunc id aliquet tincidunt, nunc nunc lacinia nunc, id ligam nunc nunc id. <input type="text"/>		
Feedback <input type="button" value="Provide Feedback"/>			

3. Video Tab

The updated video tab is designed to enhance your learning experience. It now includes capabilities to provide summarized formats of lecture content, making it easier to grasp the key points and context. Additionally, you can ask questions related to the lectures, and these will be addressed using information sourced both from the lectures themselves and from credible web sources. This feature ensures a comprehensive understanding and allows for an interactive and engaging learning process.



4. Weekly Assessment Questions

Assessment generated with GenAI is unique for every learner.

[Logo](#) [Home](#) [Courses](#) [Assessments](#) [Profile](#)

Weekly Assessment

Test your understanding of this week's material
00:00:00

Question 1

What is the capital of France?

☐ Paris

☐ London

☐ Berlin

☐ Madrid

Results

Score: 80%

Correct Answers: 4

Incorrect Answers: 1

Question 1 of 4

@2024 IIT Madras BS Degree Program

📄 📱 📺

5. Coding Environment with GenAI

Assessments/GRPAs coding environment with capabilities of getting a hint and solution based on the learner progress

[Logo](#) [My Dashboard](#) [Profile](#) Welcome User

Industry 4.0

[Week 1](#)
[Week 2](#)
[Week 3](#)
[Week 4](#)
[Week 5](#)
[Week 6](#)
[Week 7](#)
[Week 8](#)
[Week 9](#)
[Week 10](#)
[Week 11](#)
[Week 12](#)
[GRPA 1](#)
[GRPA 2](#)
[GRPA 3](#)
[GRPA 4](#)

Coding Question

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam auctor, nisl id lacinia tincidunt, nunc mauris liquam nunc, id lacinia nunc nunc id nunc. Sed auctor, nunc id lacinia tincidunt, nunc mauris aliquam nunc, id acinia nunc nunc id nunc.

Test

Submit

Ask AI

AI Reply

Milestone 03 : Scheduling and Designing

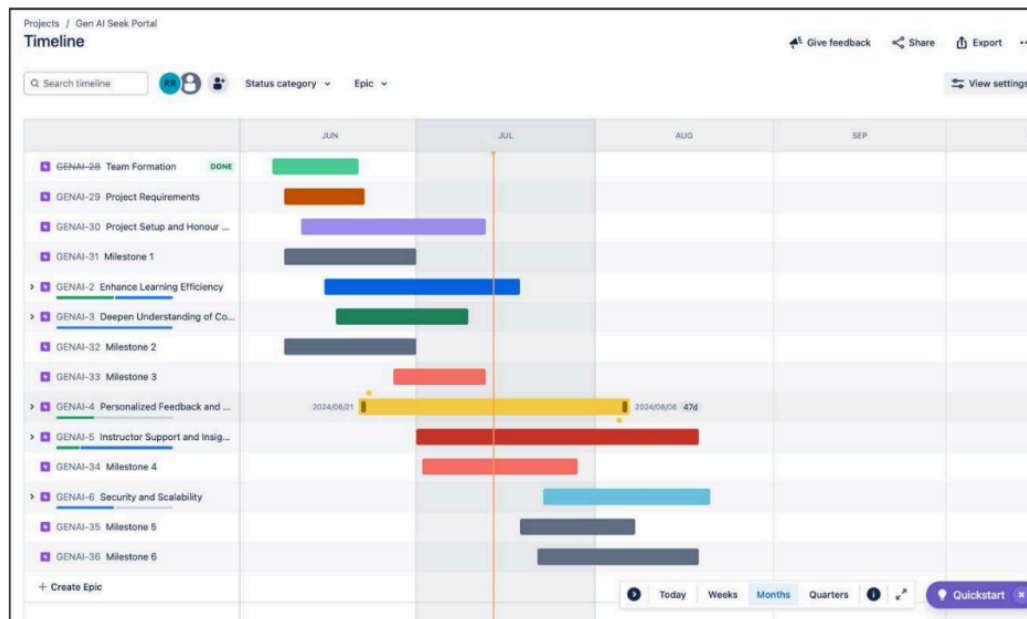
Project Schedule

We are using JIRA/ATLAS for managing our project schedule. Jira provides us with project management tools to create and assign Epics, Activities and Tasks. We have customized Jira features for custom workflows, integrated dashboards and Goal planning. The project schedule is aligned to the course project timeline:

Milestone 1	30th June
Milestone 2	30th June
Milestone 3	12th July
Milestone 4	28th July
Milestone 5	7th August
Milestone 6	18th August

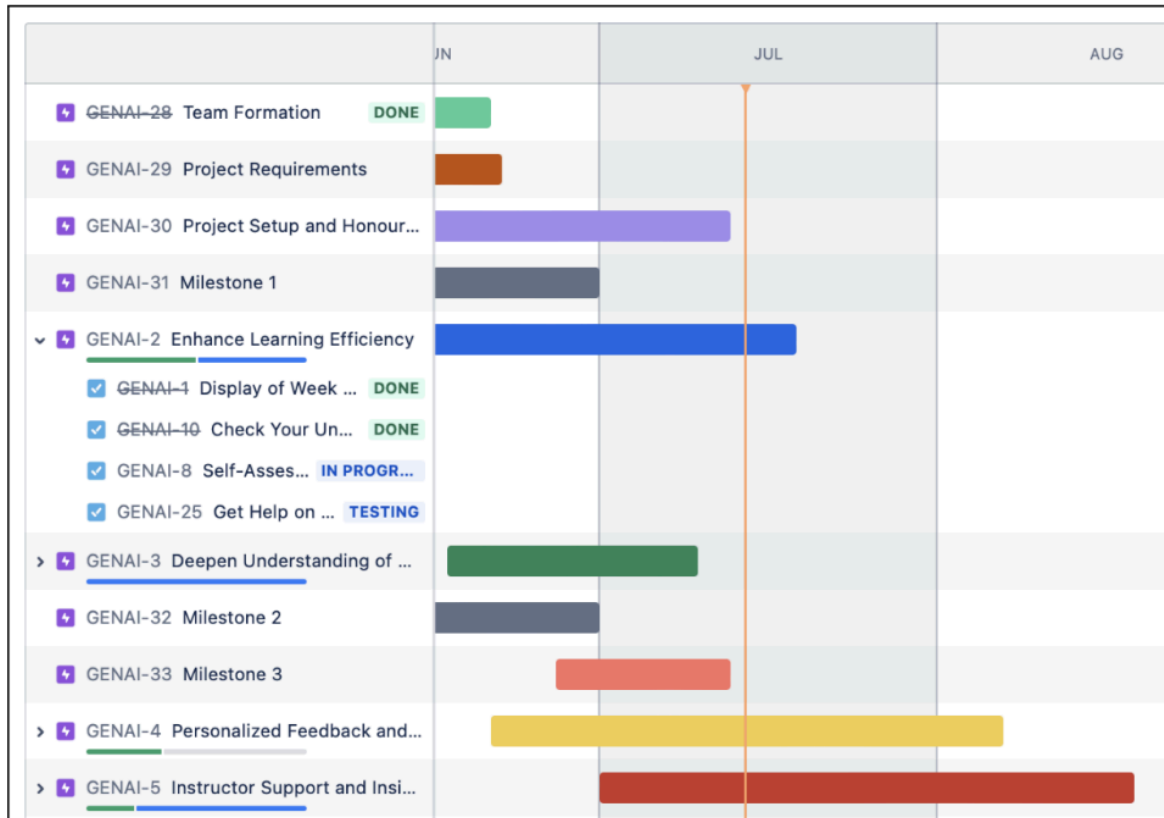
Project Timeline: Gantt Chart - Epics and Sprints

Jira also has Gantt Charts. The Gantt chart is a commonly used project management tool that displays work completed over a period of time. Individual tasks were assigned to specific members. Tasks were also assigned backup members based on availability and priority.



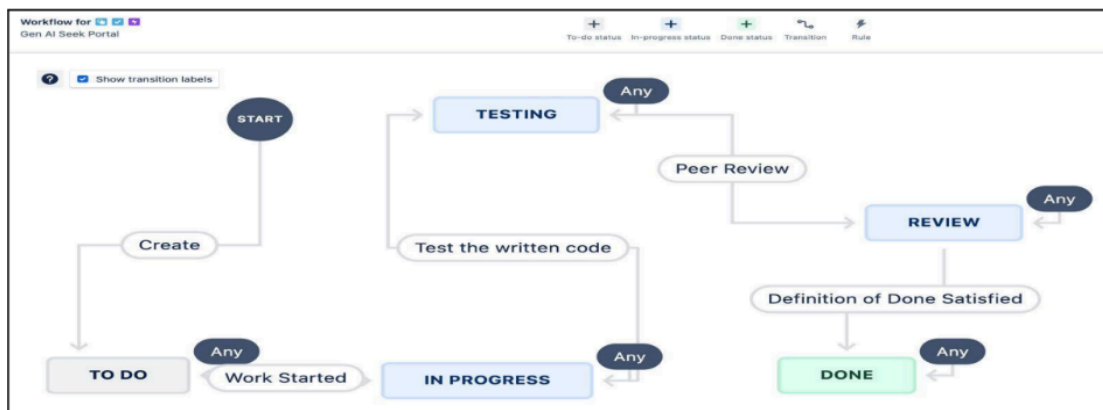
Project Timeline: Gantt Chart - Tasks/Issues

The tasks allow us to integrate scheduling with start date and end date. Here we can also assign tasks to team members for action. Each task also has a status that shows as To-Do, In progress or Done.



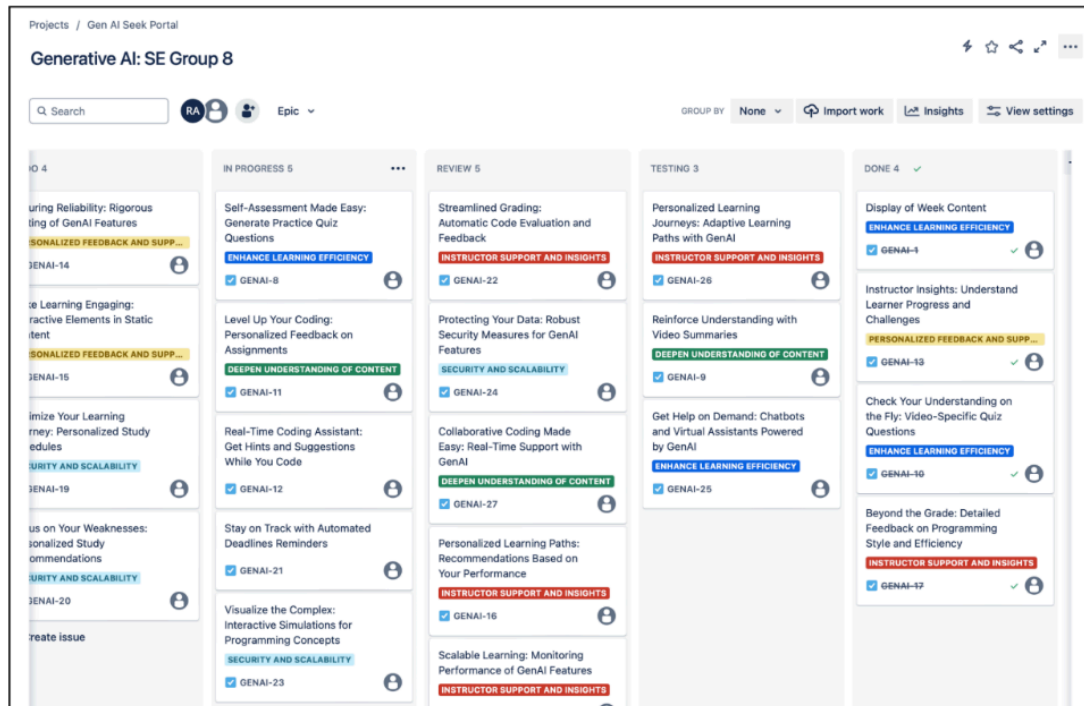
Project Timeline: Custom Workflows

We used a Jira feature to customize the task workflows. We did this by assigning a sequential flow of a task from one status to another. We added a review status that is triggered when a task is completed. After the review is completed, the task status changes to Done.

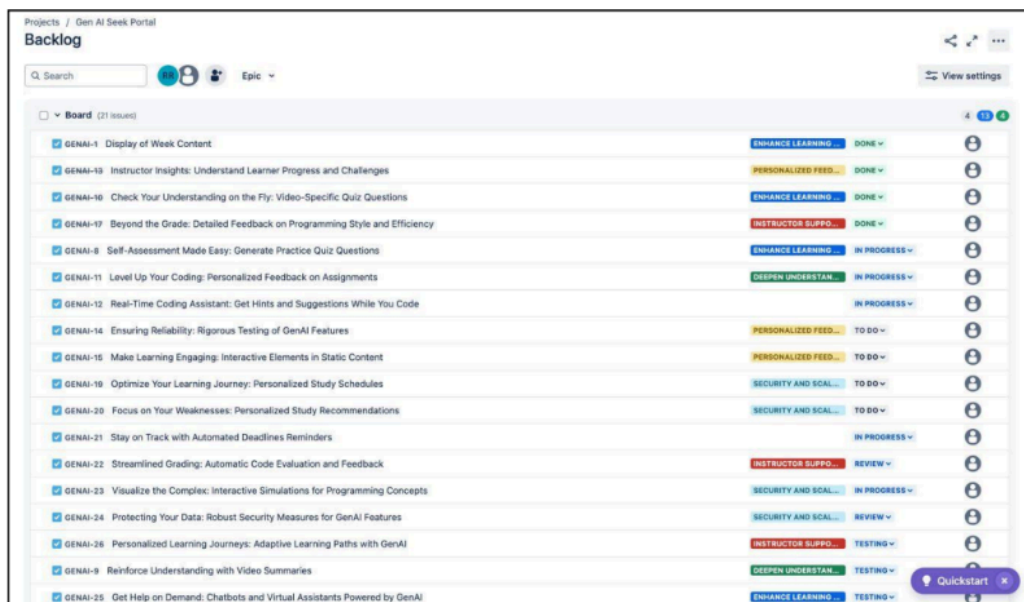


Project Scheduling: KANBAN Board - Overall

Kanban is a popular framework used to implement agile software development. All our tasks are visualized as story cards represented visually on a kanban board under their job status: To do, In progress, Review/Test, Done, and Blocked. This allows our team members to see the overall requirement and the current state of all tasks.



Project Scheduling: Backlog



Design

Designing components in a structured manner ensures modularity, simplifying maintenance and development. It promotes reusability, scalability, and maintainability. Clear interfaces enable parallel development, while encapsulation enhances security. This approach allows for independent upgrades and targeted performance optimization, resulting in a robust, scalable, and maintainable web application.

User Authentication and Management

Description :

Handles user registration, login, profile management, and session handling for both admin and regular users.

Functionality :

- **User Registration:**
 - Users can create an account by providing a username, email, and password.
 - Passwords are hashed for security using Flask-Bcrypt.
- **User Login:**
 - Registered users can log in by providing their credentials.
 - Session management is handled using Flask-Login.
- **Profile Management:**
 - Users can view and edit their profiles.
 - Admins have additional functionalities to manage user accounts.

Interfaces :

- **Frontend:**
- **HTML/CSS:**
 - Forms for registration and login with input fields for username, email, and password.
 - Profile page template displaying user information.
- **JavaScript:**
 - Handles form submissions and validation.
- **Backend**
 - **Routes:**
 - /register (POST): Handles new user registration.
 - /login (POST): Authenticates users and starts a session.
 - /logout (GET): Logs out the user and ends the session.
 - /profile (GET/POST): Displays and updates user profile information.
 - **Database Models:**
 - User: Attributes include id, username, email, password_hash, is_admin.

Dependencies :

- Flask
- Flask-Login
- Flask-WTF
- Flask-Bcrypt
- WTForms
- SQLAlchemy

Course Content Management

Description :

Allows admins to create, update, list, and delete courses and their content, including video lectures, MCQs, and coding questions.

Functionality :

- **Course Creation:**
 - Admins can create new courses by providing a course name and description.
- **Content Management:**
 - Admins can add various types of content (videos, MCQs, coding questions) to courses.
 - Admins can update or delete existing content.

Interfaces :

- **Frontend:**
- **HTML/CSS:**
 - Forms for creating and editing courses and their content.
 - Admin dashboard displaying a list of courses with options to manage content.
- **JavaScript:**
 - Functions to handle form submissions and update the UI dynamically.
- **Backend:**
 - **Routes :**
 - /admin/courses (GET): Lists all courses for the admin.
 - /admin/course/new (POST): Creates a new course.
 - /admin/course/<int:course_id>/edit (POST): Edits an existing course.
 - /admin/course/<int:course_id>/delete (POST): Deletes a course.
 - /admin/course/<int:course_id>/content/new (POST): Adds new content to a course.
 - /admin/course/content/<int:content_id>/edit (POST): Edits existing content.
 - /admin/course/content/<int:content_id>/delete (POST): Deletes content.

- **Database Models :**

- Course: Attributes include id, name, description.
- Content: Attributes include id, course_id, type (video, mcq, coding), data.

Dependencies :

- Flask
- SQLAlchemy
- WTForms
- Flask-WTF

Content Interaction

Description :

Enables users to interact with course content, including viewing videos, answering MCQs, and solving coding questions.

Functionality :

- **Video Lectures:**
 - Users can view embedded YouTube videos.
- **MCQs:**
 - Users can select answers to MCQs and receive immediate feedback.
- **Coding Questions:**
 - Users can view coding problems and submit solutions, which are processed via an API.

Interfaces :

- **Frontend :**
- **HTML/CSS:**
 - Pages for displaying video content, MCQs, and coding questions.
- **JavaScript:**
 - Functions for handling MCQ selections and submitting coding solutions.
- **Backend:**
- **Routes:**
 - /course/<int:course_id>/content/<int:content_id> (GET): Displays specific content to the user.
 - /course/<int:course_id>/content/<int:content_id>/submit (POST): Handles user submissions for MCQs or coding solutions.

- **Database Models:**

- Content: Attributes include id, course_id, type, data, user_submissions.

Dependencies :

- Flask
- SQLAlchemy
- WTForms
- Flask-WTF

Prompt Processing

Description :

Allows users to submit prompts related to video content, which are processed by an external API and displayed on the frontend.

Functionality

- **Prompt Submission:**

- Users can submit questions or prompts related to video content.

- **API Processing:**

- Prompts are sent to an external API for processing.

- **Response Display:**

- Processed responses are displayed to the user.

Interfaces :

- **Frontend:**

- **HTML/CSS:**

- Forms for prompt submission.

- **JavaScript:**

- Functions to handle prompt submissions and display responses.

- **Backend:**

- **Routes:**

- /course/<int:course_id>/content/<int:content_id>/prompt (POST): Submits user prompt to the external API and returns the response.

- **Database Models:**

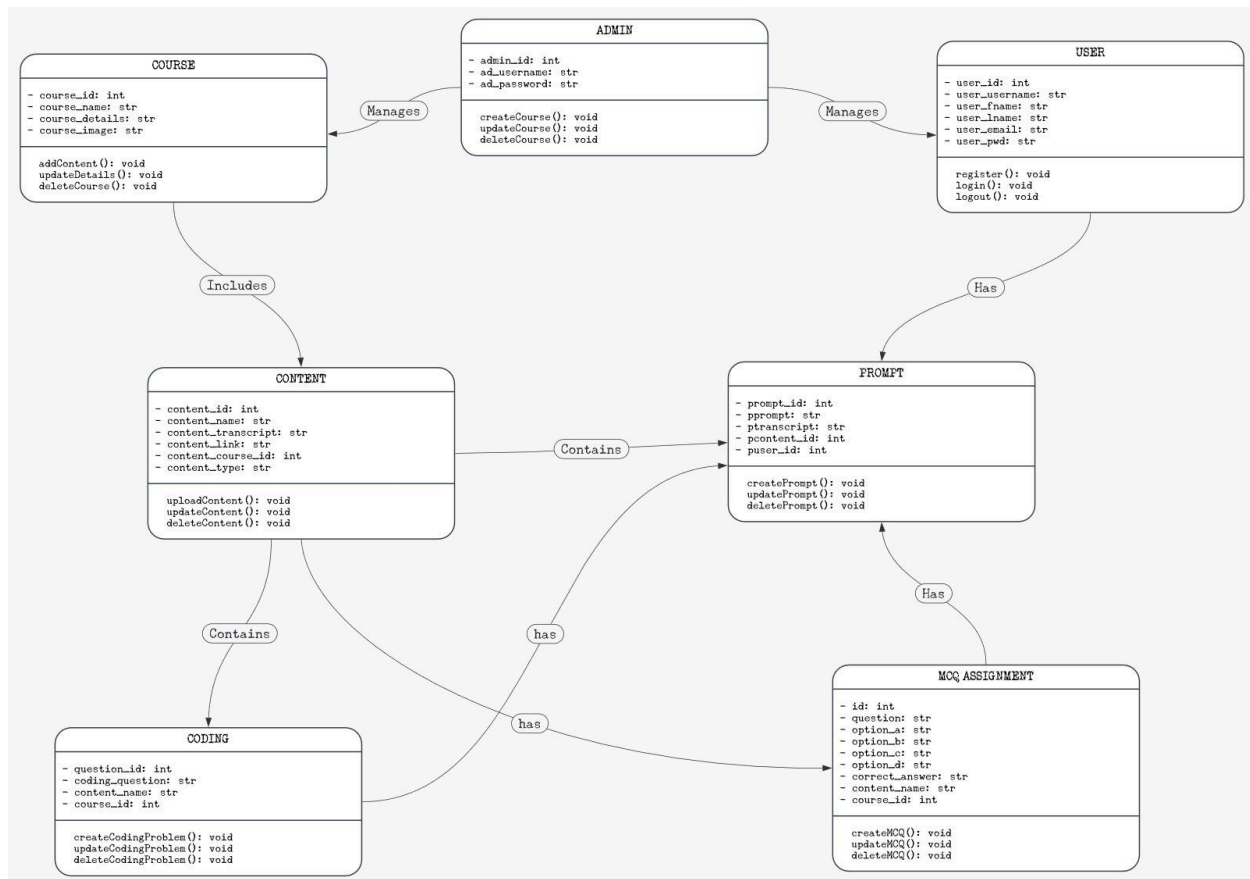
- Prompt: Attributes include id, content_id, prompt_text, response_text.

Dependencies

- Flask
- SQLAlchemy
- WTForms
- Flask-WTF
- Requests (for making API calls)

UML

This UML diagram models a system with entities related to courses, content, users, and assessments. Here are the key components and their relationships:



Entities

- **Course**
 - Attributes: `course_id`, `course_name`, `course_detail`, `course_link`
 - Methods: `addContent()`, `updateDetails()`, `deleteCourse()`
 - Relationships: Includes Content

- **Content**
 - Attributes: content_id, content_name, content_transcript, content_link, content_course_id, content_type
 - Methods: uploadContent(), updateContent(), deleteContent()
 - Relationships: Contains Prompt, Contains MCQ_ASSIGNMENT, Contains Coding
- **Prompt**
 - Attributes: prompt_id, prompt, ptranscript, pcontent_id, puser_id
 - Methods: createPrompt(), updatePrompt(), deletePrompt()
 - Relationships: Has Content
- **MCQ_ASSIGNMENT**
 - Attributes: id, question, option_a, option_b, option_c, option_d, correct_answer, content_name, course_id
 - Methods: createMCQ(), updateMCQ(), deleteMCQ()
 - Relationships: Has Content

Relationships

- **Manages:** Admin manages Courses and Users.
- **Includes:** Courses include Content.
- **Contains:** Content contains Prompts, MCQ_ASSIGNMENTS, and Coding problems.
- **Has:** Users have Prompts and Prompts have Content.

Scrum Meetings


Scrum meetings are essential gatherings where the Scrum Master, Product Owner, and Scrum Teams collaborate to ensure the smooth progress of a project. These meetings serve multiple purposes, including sharing updates, discussing any open issues or impediments, reviewing feedback, and planning or assigning subsequent work.

Each Scrum meeting plays a vital role in maintaining alignment and transparency among team members. During these meetings, team members provide updates on their current tasks, allowing everyone to stay informed about the overall progress of the project. Additionally, any challenges or roadblocks that may hinder progress are identified and addressed, ensuring that the team can continue working efficiently.

This feedback loop is crucial for refining the project's direction and making sure that the team is delivering value consistently.


Planning and assigning work is another key aspect of these meetings. The team collaborates to prioritize tasks, estimate the effort required, and assign responsibilities, ensuring that everyone is clear on their role and what needs to be accomplished in the upcoming period.

Meeting Planning




Join with Google Meet

meet.google.com/umc-gyve-tha




Join by phone


(US) +1 575-518-3121 PIN: 213 191 710#



Take meeting notes

Start a new document to capture notes








8 guests

1 yes

7 awaiting







Ritik Arora

Organizer


Set your working location




NAVDEEP




ANSARI MOHD TABISH FATEH MOHD (21f20014...




MILAN




VEDANSHI TEWARI




Rohan Ajay Ramani



SHUBHAM GATTANI



TAMANNA TAK




The Milestone 03 final review before submission.

Milestone 03 Document

: <https://docs.google.com/document/d/1Jc03foN3r-Bm-CjbC9Vu3FU4IMm8htvPvC6xCa8WpoE/edit>

Upcoming Next: MILESTONE 04

We will be closing Milestone 03 requirements by Monday, 22 July.



10 minutes before

Meeting Report

Date	4 July 2024 22:00 to 23:00	
Agenda	Milestone 3 Discussion and Progress Updates	
Meeting Link	https://meet.google.com/iob-icai-yoe	

Attendance

1	Milan	<input checked="" type="checkbox"/>
2	Ritik Arora	<input checked="" type="checkbox"/>
3	Vedanshi Tiwari	<input checked="" type="checkbox"/>
4	Tamanna Tak	<input checked="" type="checkbox"/>
5	Navdeep	<input checked="" type="checkbox"/>
6	Shubham Gattani	<input checked="" type="checkbox"/>
7	Rohan Ajay Ramani	<input checked="" type="checkbox"/>
8	Tabish	<input checked="" type="checkbox"/>

Table of Content

1. [Review of last meeting](#)

2. [Planning Milestone 3 and 4](#)

3. [Development Preparation Tasks](#)

4. [Open Discussion](#)

5. [Meeting schedule for Feb-Mar 2024](#)





Document Link - [SE Project Meeting Report](#)

My Drive > SE Project ▾ 👤

Type ▾

People ▾

Modified ▾

Name	Owner
 Meeting Recordings	 me
 Meeting Notes	 me

Milestone 04 : API Endpoints

CHAPTER 1 API INTEGRATION OVERVIEW

This comprehensive documentation provides an in-depth overview of the API endpoints that have been meticulously designed to align with the project's diverse user stories. The primary goal is to develop new API endpoints or leverage existing ones from well-established libraries, ensuring seamless functionality and user experience. This includes the strategic integration with external services, notably ollama, to incorporate advanced language model capabilities. By doing so, we aim to enhance interactive features, content creation, and user engagement on the Platform.

The document outlines a detailed list of both integrated and newly created APIs, offering a clear and organized view of the system's architecture. Each API endpoint is described comprehensively, covering its purpose, the HTTP methods used, expected inputs (parameters and body data), and the structure of responses. These descriptions follow the OpenAPI 3.0 specifications, ensuring that the API documentation is standardized, clear, and accessible to developers. This adherence to widely recognized standards not only facilitates easier integration and collaboration but also ensures that the APIs are robust and future-proof. In addition to the endpoint descriptions, the documentation highlights the rationale behind choosing specific external APIs, such as ollama. This choice is driven by the need to provide sophisticated natural language processing capabilities, which can greatly enhance the platform's interactive features. For instance, ollama's language models can be used to generate content, respond to user queries in a conversational manner, and provide personalized learning assistance, thereby enriching the overall user experience.

Furthermore, the documentation includes a section on best practices for using these APIs, emphasizing secure authentication, efficient data handling, and proper error management. This is crucial for maintaining the integrity and security of the system, especially when dealing with sensitive user information and interactions with third-party services. The final section of the document includes the YAML file submission, which serves as a crucial element for API documentation and developer guidance. The YAML file provides a machine-readable specification of the API, including detailed metadata about the API's paths, methods, parameters, and response formats. This file is essential for several reasons:

- **Automated Documentation:** The YAML file can be used to generate human-readable API documentation, which is invaluable for developers who need to understand the system quickly.

- **Client Library Generation:** It enables the automatic generation of client libraries in various programming languages, facilitating easier integration with the API for external developers.
- **API Testing and Validation:** The YAML specification can be used with testing tools to validate the API's functionality and ensure that it meets the defined specifications.
- **CI/CD Integration:** Including the YAML file in CI/CD pipelines ensures that any changes to the API are automatically tested and deployed, maintaining high standards of reliability and performance.

Overall, this documentation not only provides a detailed roadmap for implementing and using the API endpoints but also ensures that all stakeholders have a clear understanding of the system's capabilities and potential. It serves as a vital resource for developers, testers, and product managers, guiding them through the complexities of API integration and maintenance. By combining thorough descriptions, best practices, and a standardized YAML submission, the documentation lays a strong foundation for the project's success and scalability.

CHAPTER 2

API INTEGRATION IMPLEMENTATION

To significantly elevate the functionality and user experience of our platform, we will strategically integrate with ollama's API for language models (LLMs). This integration is pivotal in harnessing cutting-edge natural language processing capabilities to offer a suite of advanced features that cater to diverse aspects of user interaction and content creation. By incorporating ollama's LLMs, our platform will provide users with more personalized, responsive, and insightful educational experiences. The LLMs will adapt to user needs and queries in real-time, enhancing interactive learning, content generation, and user support.

The development process involved a meticulous approach to address each user story, focusing on either creating new API endpoints or utilizing existing endpoints from reputable libraries. This approach ensures that the platform remains robust, scalable, and user-friendly. Each API was designed with a specific purpose, reflecting the requirements outlined in the user stories. The process included:

- **Thorough Requirement Analysis:** Understanding the specific needs outlined in user stories to ensure the API endpoints meet those needs effectively.
- **Endpoint Design and Implementation:** Crafting new endpoints where necessary and integrating existing ones to provide a seamless user experience.
- **Utilization of Established APIs:** Leveraging well-established APIs to enhance

development efficiency, ensure reliability, and integrate advanced features without reinventing the wheel.

- **Integration Testing:** Rigorous testing of integrated APIs to verify their functionality and performance within the platform.

API Development for User Stories

User Authentication and Profile Management:

New APIs Created:

User Registration API: Allows new users to create an account by submitting their personal details, such as username, email, and password.

User Login API: Facilitates user authentication, enabling access to the platform's features based on their credentials.

User Profile API: Provides endpoints for viewing and updating user profiles, ensuring users can manage their personal information securely.

Integrated APIs:

OAuth Authentication API: Integrated with external OAuth providers (such as Google and Facebook) to offer users convenient and secure login options without the need for separate credentials.

Admin Dashboard and Course Management:

New APIs Created:

Admin Login and Management APIs: Dedicated endpoints for admin authentication and profile management, separate from regular user endpoints.

Course Management APIs: Include endpoints for creating, updating, and deleting courses. These APIs allow admins to manage course content, including text, multimedia, and assessments.

Content Creation and Update APIs: Specific endpoints to facilitate the creation and management of various types of content (e.g., assignments, quizzes, videos) within each course.

Integrated APIs:

Media Hosting APIs: Utilized for storing and streaming multimedia content, ensuring that videos and other media types are delivered efficiently and securely.

Interactive Learning and User Engagement:

New APIs Created:

Interactive Query APIs: Allow users to submit queries related to course content, which can be processed to provide tailored responses and additional resources.

Integrated APIs:

ollama API for Language Models: Leveraged to enhance the platform's interactivity. This includes generating detailed responses to user queries, creating content, and providing learning aids.

List of Integrated APIs

1. ollama API

- **Description:** The ollama API is integrated to leverage advanced language model functionalities that significantly enhance the platform's capabilities. This integration allows for the deployment of sophisticated natural language processing tools to:

- **Interactive Learning:** Provide users with detailed explanations, tailored responses to queries. This helps create a more engaging and responsive learning environment.

This extended description provides a more detailed understanding of each integrated API's role and benefits within the platform, highlighting how they contribute to the overall functionality and user experience.

List of Created APIs

User Registration API

- **Endpoint:** /user_register
- **Description:** This API endpoint is designed to facilitate the creation of new user accounts on the platform. Users can register by submitting their personal details, including a unique username, email address, and password. The endpoint handles:
 - **Account Creation:** Collects and validates the input data to ensure it meets security and format requirements.
 - **User Verification:** Initiates any necessary verification processes, such as email confirmation, to validate the user's identity.
 - **Initial Setup:** Creates a new user profile and initializes user-specific settings, allowing for immediate access upon successful registration.

User Login API

- **Endpoint:** /user_login

- **Description:** This endpoint provides a secure authentication mechanism for users to access the platform. By submitting their credentials, including username and password, users can:

- **Authenticate:** Verify their identity against stored credentials to ensure secure access.
- **Session Management:** Establish a user session, granting access to personalized features and content based on successful authentication.
- **Security Measures:** Implement security practices such as rate limiting and encryption to protect user data during the login process.

User Profile API

- **Endpoint:** /user/profile

- **Description:** This API allows users to view and update their profile information. It provides:

- **Profile Viewing:** Access to current profile details, including username, email, and other personal information.
- **Profile Update:** Ability to modify and save changes to personal details, ensuring that users can keep their information up to date.
- **Data Security:** Ensures that profile updates are securely processed and validated to protect user privacy.

User Logout API

- **Endpoint:** /user_logout

- **Description:** This endpoint manages user logout operations, ensuring a secure exit from the platform. It includes:

- **Session Termination:** Clears session data and tokens to prevent unauthorized access after logout.
- **User Feedback:** May provide feedback or confirmation to the user that the logout process was successful.
- **Security Measures:** Ensures that all user session data is securely removed to protect against potential security risks.

Admin Registration API

- **Endpoint:** /admin_register

- **Description:** This API enables the creation of admin accounts with essential details. Admins can register by submitting:

- **Personal Information:** Including name, email, and password.

- **Account Verification:** Processes to verify admin credentials and ensure secure account setup.
- **Initial Configuration:** Initializes admin-specific settings and permissions to enable access to administrative features.

Admin Login API

- **Endpoint:** /admin_login
- **Description:** Facilitates secure login for administrators. The endpoint allows admins to:
 - **Authenticate:** Verify their identity using provided credentials (username and password).
 - **Session Management:** Establish an admin session with access to administrative functions.
 - **Enhanced Security:** Employ advanced security measures to protect against unauthorized access.

Admin Profile API

- **Endpoint:** /admin/profile
- **Description:** This API provides functionality for administrators to manage their profile information. Features include:
 - **Profile Viewing:** Access to current admin profile details, such as contact information and role.
 - **Profile Updating:** Ability to update personal details and save changes securely.
 - **Data Protection:** Ensures that profile updates are processed with stringent security measures.

Admin Logout API

- **Endpoint:** /admin_logout
- **Description:** Manages the logout process for administrators, ensuring secure session termination. This endpoint:
 - **Session Termination:** Clears admin session data to prevent unauthorized access post-logout.
 - **Logout Confirmation:** Provides feedback to confirm that the logout process was successfully completed.
 - **Security Assurance:** Ensures that all session information is removed to maintain platform security.

Create Course API

- **Endpoint:** /admin/create_course
- **Description:** Allows administrators to create new courses by providing necessary details.

Features include:

- **Course Creation:** Submission of course name, description, and related image.
- **Data Validation:** Ensures that course details are correctly formatted and meet platform standards.
- **Initial Setup:** Initializes the course within the platform, making it available for further management and content addition.

Update Course API

- **Endpoint:** /admin/{cat_id}/update
- **Description:** Enables administrators to update existing course details. The endpoint supports:

- **Course Modification:** Updates course name, description, and associated resources.
- **Data Integrity:** Ensures that changes are validated and applied correctly.
- **Content Management:** Facilitates updates to course content and presentation.

Delete Course API

- **Endpoint:** /admin/{cat_id}/delete_course
- **Description:** Facilitates the removal of courses from the platform. This endpoint:
 - **Course Deletion:** Handles the deletion process, including any data dependencies associated with the course.
 - **Data Management:** Ensures that all related content and records are appropriately managed and removed.
 - **Confirmation:** Provides confirmation of course deletion to the admin.

Create Content API

- **Endpoint:** /admin/{cat_id}/create_content
- **Description:** Supports the addition of various types of content to a course. Features include:

- **Content Addition:** Allows admins to add assignments, quizzes, multimedia, and other educational materials.
- **Content Validation:** Ensures that content is correctly formatted and adheres to platform guidelines.
- **Content Management:** Integrates new content into the specified course.

Update Content API

- **Endpoint:** /admin/{cat_id}/{content_id}/update
- **Description:** Provides functionality for updating existing content items within courses.

The endpoint includes:

- **Content Modification:** Allows changes to be made to assignments, quizzes, and other content elements.
- **Data Validation:** Ensures that updates are validated and applied correctly.
- **Content Integration:** Updates are seamlessly integrated into the course structure.

Delete Content API

- **Endpoint:** /admin/{cat_id}/{content_id}/delete
- **Description:** Allows administrators to delete content items from a course. Features include:
 - **Content Removal:** Facilitates the removal of specific content items, managing content lifecycle.
 - **Data Management:** Ensures that deletion processes handle dependencies and related data correctly.
 - **Confirmation:** Provides confirmation of content deletion to the admin.

List Courses for Users API

- **Endpoint:** /user/course_page
- **Description:** Displays a list of available courses for users, providing details such as:
 - **Course Information:** Includes course names, descriptions, and enrollment status.
 - **User Interaction:** Allows users to view and select courses for enrollment.
 - **Progress Tracking:** May include information on user progress and course completion.

List Contents for Admin API

- **Endpoint:** /admin/{cat_id}/contents
- **Description:** Lists all contents within a specific course for administrative management.

The endpoint:

- **Content Listing:** Displays a comprehensive list of content items associated with a course.
- **Content Management:** Facilitates the management and organization of course materials.
- **Administrative Access:** Provides tools for admins to manage content effectively.

Interactive Query API

- **Endpoint:** /prompt
- **Description:** Allows users to submit questions related to course content and receive AI-driven responses. Features include:
 - **Interactive Learning:** Enhances the learning experience by providing detailed and relevant responses to user queries.
 - **Natural Language Processing:** Utilizes advanced language models to understand and respond to user questions.
 - **User Engagement:** Encourages user interaction and engagement with course materials.

Evaluation of Subjective Assignment API

- **Endpoint:** /evaluate_subjective
- **Description:** Used to evaluate the subjective_assignment questions' answers. Features include:
 - **Accuracy of Answer:** How accurate the answer is in context of the question asked.
 - **Grammatical Mistakes:** Utilizes advanced language models to understand and find the grammatical errors in the answers written by the user.
 - **Additional points:** Encourages user to write better answer by providing a list of additional points that can be included.

Processing of web_prompt API

- **Endpoint:** /process_web_prompt
- **Description:** Used to answer the user's question without using any pre-defined context.

Processing of web_prompt API

- **Endpoint:** /process_prompt
- **Description:** Used to answer the user's question using any pre-defined context which is the transcript of the video for which the user is asking the question.

Description of API endpoints.

User Management APIs:

- **/user_register:** Registers a new user by collecting their username, email, and password.
- **/user_login:** Authenticates users by verifying their credentials and grants access to the platform.
- **/user/profile:** Allows users to view and update their personal profile information.

- **/user_logout:** Manages the logout process, ensuring that user sessions are properly Terminated.

Admin Management APIs:

- **/admin_register:** Registers a new admin by collecting details like name, email, and password.
- **/admin_login:** Authenticates admins and provides access to administrative functionalities.
- **/admin/profile:** Allows admins to view and update their profile information.
- **/admin_logout:** Manages admin logout, ensuring secure termination of sessions.

Course and Content Management APIs:

- **/admin/create_course:** Enables the creation of new courses, specifying details like name, description, and image.
- **/admin/{cat_id}/update:** Updates existing course details, such as name, content, and image.
- **/admin/{cat_id}/delete_course:** Deletes a course and handles any related data dependencies.
- **/admin/{cat_id}/create_content:** Adds new content to a course, including assignments, quizzes, and multimedia.
- **/admin/{cat_id}/{content_id}/update:** Updates existing content within a course.
- **/admin/{cat_id}/{content_id}/delete:** Deletes specific content items from a course.
- **/user/course_page:** Lists available courses for users with enrollment and progress details.
- **/admin/{cat_id}/contents:** Lists all contents for a specific course for admin management.

Interactive and Support APIs:

- **/prompt:** Processes user queries related to course content, utilizing AI for responses.
- **/user/{cat_id}/delete_prompt_item:** Manages user feedback and ratings for courses and content.
- **/process_web_prompt:** Processes the prompt/question and answers it without any context (unless explicitly specified by the user in the prompt itself)
- **/process_prompt:** Processes the prompt and answers it in the context of the respective transcript.
- **/evaluate_subjective:** Used to evaluate the subjective_assignment with respect to the accuracy of the answer, the grammatical mistakes, additional points that could've been added by the user in the answer and any further scope of improvement.

Milestone 05 : Test Suite and Test Cases

1. Setting up Test Suite and Test Cases

Setting up API test cases using pytest allows our developers to check and confirm that all APIs and functions of the seek_clone app are working properly. Pytest is a commonly used library used for setting up the testing suite with test cases. It helps us write and run tests easily to ensure that our APIs are doing what they're supposed to do, like sending and receiving data correctly. Our APIs are located in app.py.

2. Test cases for APIs

1. User Registration API

1.1 API being tested: /user_register

1.2 Inputs:

```
{  
  
  "u_fname": "John",  
  
  "u_lname": "Doe",  
  
  "u_email": "john.doe@example.com",  
  
  "u_uname": "johndoe",  
  
  "u_pwd": "securePassword123"  
  
}
```

1.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to login page

1.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /user_login

1.5 Result: Success

1.6 API being tested: /user_register

1.7 Inputs:

```
{  
  
  "u_fname": "John",  
  
  "u_lname": "Doe",  
  
  "u_email": "john.doe[at]example.com",  
  
  "u_uname": "johndoe",  
  
  "u_pwd": "securePassword123"  
  
}
```

1.8 Expected Output:

- Status Code: 302
- Response Body: Redirect to registration page

1.9 Actual Output:

- Status Code: 302
- Response Body: Redirect to /user_register

1.10 Result: Success

2. User Login API

2.1 API being tested: /user_login

2.2 Inputs:

```
{  
  
  "username": "johndoe",  
  "password": "securePassword123"  
  
}
```


2.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to user dashboard

2.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /user_dashboard

2.5 Result: Success

2.6 API being tested: /user_login

2.7 Inputs:

```
{  
  
  "username": "johndoe",  
  "password": "wrong_PASSword"  
}
```

2.8 Expected Output:

- Status Code: 302
- Response Body: Redirect to user login

2.9 Actual Output:

- Status Code: 302
- Response Body: Redirect to /user_login

2.10 Result: Success

3. User Profile API

3.1 API being tested: /user_profile

3.2 Inputs:

```
{  
  
  "Authorization": "Bearer validToken"  
}
```

3.3 Expected Output:

- Status Code: 200

- Response Body:

```
{  
  
  "u_fname": "John",  
  
  "u_lname": "Doe",  
  
  "u_email": "john.doe@example.com",  
  
  "u_username": "johndoe"  
}
```

3.4 Actual Output:

- Status Code: 200
- Response Body: As Expected

3.5 Result: Success

4. User Logout API

4.1 API being tested: /user_logout

4.2 Inputs:

```
{  
  
  "Authorization": "Bearer validToken"  
}
```

4.3 Expected Output:

- Status Code: 302
- Response: Redirect to login page

4.4 Actual Output:

- Status Code: 302
- Response : As Expected

4.5 Result: Success

5. Admin Registration API

5.1 API being tested: /admin_register

5.2 Inputs:

```
{  
  
  "ad_fname": "Jane",  
  
  "ad_lname": "Smith",  
  
  "email": "jane.smith@example.com",  
  
  "ad_username": "janesmith",  
  
  "ad_pwd": "adminSecurePassword456"  
}
```

5.3 Expected Output:

- Status Code: 302
- Response : Redirect to admin login page

5.4 Actual Output:

- Status Code: 302
- Response: Redirect to /admin_login

5.5 Result: Success

6. Admin Login API

6.1 API being tested: /admin_login

6.2 Inputs:

```
{  
  "ad_username": "janesmith",  
  "passwd": "adminSecurePassword456"  
}
```

6.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to admin dashboard

6.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /admin

6.5 Result: Success

7. Admin Profile API

7.1 API being tested: /a_profile

7.2 Inputs:

```
{  
  
  "Authorization": "Bearer validToken"  
  
}
```

7.3 Expected Output:

- Status Code: 200
- Response Body:

```
{  
  
  "ad_fname": "Jane",  
  
  "ad_lname": "Smith",  
  
  "email": "jane.smith@example.com",  
  
  "ad_username": "janesmith"  
  
}
```

7.4 Actual Output:

- Status Code: 200
- Response Body: As Expected

7.5 Result: Success

8. Admin Logout API

8.1 API being tested: /admin_logout

8.2 Inputs:

```
{  
  
  "Authorization": "Bearer validToken"  
  
}
```

8.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to admin login page

8.4 Actual Output:

- Status Code: 302
- Response Body: As Expected

8.5 Result: Success

9. Create Course API

9.1 API being tested: /admin/create_course

9.2 Inputs:

```
{  
  "cat_name": "Introduction to Programming",  
  "cat_details": "A beginner course on programming.",  
  "cat_img": "<binary data>"  
}
```

9.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to course list

9.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /admin

9.5 Result: Success

10. Update Course API

10.1 API being tested: /admin/<cat_id>/update

10.2 Inputs: Path Parameter: {cat_id} = 1

```
{  
  
  "cat_name": "Advanced Programming",  
  
  "cat_text": "An advanced course on programming.",  
  
  "cat_img": "<binary data>"  
}
```

10.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to course list

10.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /admin

10.5 Result: Success

11. Update Course API

11.1 API being tested: /admin/{cat_id}/update

11.2 Inputs: Path Parameter: {cat_id} = 1

```
{  
  
  "cat_name": "Advanced Programming",  
  "cat_text": "An advanced course on programming.",  
  "cat_img": "<binary data>"  
}
```

11.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to course list

11.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /admin

11.5 Result: Success

12. Create Content API

12.1 API being tested: /admin/<cat_id>/create_content

12.2 Inputs: Path Parameter: {cat_id} = 1

```
{
  "content_type": "assignment",
  "assignment_question": "What is the complexity of binary search?",
  "option_a": "O(n) ",
  "option_b": "O(log n) ",
  "option_c": "O(n log n) ",
  "option_d": "O(1) ",
  "correct_answer": "O(log n) "
}
```

12.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to content list

12.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /admin/<cat_id>/contents

12.5 Result: Success

13. Update Content API

13.1 API being tested: /admin/<cat_id>/<content_id>/update

13.2 Inputs: Path Parameters: {cat_id} = 1, {content_id} = 1

```
{
  "content_type": "assignment",
  "assignment_question": "Updated question?",
  "option_a": "Updated option",
}
```

```
"correct_answer": "Updated option"
}
```

13.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to content list

13.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /admin/{cat_id}/contents

13.5 Result: Success

14. Delete Content API

14.1 API being tested: /admin/<cat_id>/<content_id>/delete

14.2 Inputs: Path Parameters: {cat_id} = 1, {content_id} = 1

14.3 Expected Output:

- Status Code: 302
- Response Body: Redirect to content list

14.4 Actual Output:

- Status Code: 302
- Response Body: Redirect to /admin/<cat_id >/contents

14.5 Result: Success

15. List Contents for Admin API

15.1 API being tested: /admin/{cat_id}/contents

15.2 Inputs: Path Parameters: {cat_id} = 1, {content_id} = 1

```
{
  "Authorization": "Bearer validUserToken"
}
```

15.3 Expected Output:

- Status Code: 200
- Response Body:

```
{
  "cat_id": 1,
  "cat_name": "Introduction to Programming",
  "cat_details": "A beginner course on programming."
}
```



```
}
```

15.4 Actual Output:

- Status Code: 200
- Response Body:

15.5 Result: Success

16. Prompt related to video API

16.1 API being tested: /process_prompt

16.2 Inputs:

```
{  
  "prompt": "Explain the concept of polymorphism."  
}
```

16.3 Expected Output:

- Status Code: 200
- Response Body:

```
{  
  "Answer": "Polymorphism allows objects to be treated as instances of their  
parent class rather than their actual class."  
}
```

16.4 Actual Output:

- Status Code: 200
- Response Body: As Expected

16.5 Result: Success

17. Prompt with web API

17.1 API being tested: /process_web_prompt

17.2 Inputs:

```
{  
  "prompt": "Explain the concept of polymorphism."  
}
```

```
}
```

17.3 Expected Output:

- Status Code: 200
- Response Body:

```
{  
  "Answer": "Polymorphism allows objects to be treated as instances of their  
parent class rather than their actual class."  
}
```

17.4 Actual Output:

- Status Code: 200
- Response Body: As Expected

17.5 Result: Success

18. Coding Answer API

18.1 API being tested: /get_coding_answer

18.2 Inputs:

```
{  
  "coding_question": "Write square root function"  
}
```

18.3 Expected Output:

- Status Code: 200
- Response Body:

```
{  
  "Answer": "def square_root(x): if x < 0: return "Square root of a negative  
number is not real." else: return x ** 0.5 # Example usage:result =  
square_root(16) print("The square root is:", result)"  
}
```

18.4 Actual Output:

- Status Code: 200
- Response Body: As Expected

18.5 Result: Success

19. Coding Answer hint API

19.1 API being tested: /get_coding_hint

19.2 Inputs:

```
{
  "coding_question": "Write square root function",
  "additional_input": "a=b^2",
}
```

19.3 Expected Output:

- Status Code: 200
- Response Body:

```
{
  "Answer": "Consider using the exponentiation operator ** with 0.5 to compute the square root of a number."
}
```

19.4 Actual Output:

- Status Code: 200
- Response Body: As Expected

19.5 Result: Success

20. Check User Code API

20.1 API being tested: /check_code

20.2 Inputs:

```
{
  "code": "def square_root(x):
return x ** 0.5
",
  "input": "16",
  "expected_output": "4",
}
```

20.3 Expected Output:

- Status Code: 200
- Response Body:

```
{  
  "is_correct": "True"  
}
```

20.4 Actual Output:

- Status Code: 200
- Response Body: As Expected

20.5 Result: Success

21. Subjective Evaluation API

21.1 API being tested: /evaluate_subjective

21.2 Inputs:

```
{  
  "question": "What is phenomenon",  
  "answer": "a fact or situation that is observed to exist or happen,  
  especially  
  one whose cause or explanation is in question."  
}
```

21.3 Expected Output:

- Status Code: 200
- Response Body:

```
{  
  "cohesiveness_feedback": "The answer is concise and directly addresses the  
  question, providing a clear definition of 'phenomenon'.",  
  "grammar_feedback": "The answer is grammatically correct, but the spacing  
  between words could be improved for better readability.",  
  "plagiarism_feedback": "The given content seems to be 95% AI generated"  
}
```

21.4 Actual Output:

- Status Code: 200
- Response Body: As Expected (with minor changes because AI generated answer is not the same every time)

21.5 Result: Success

Code Review and Testing

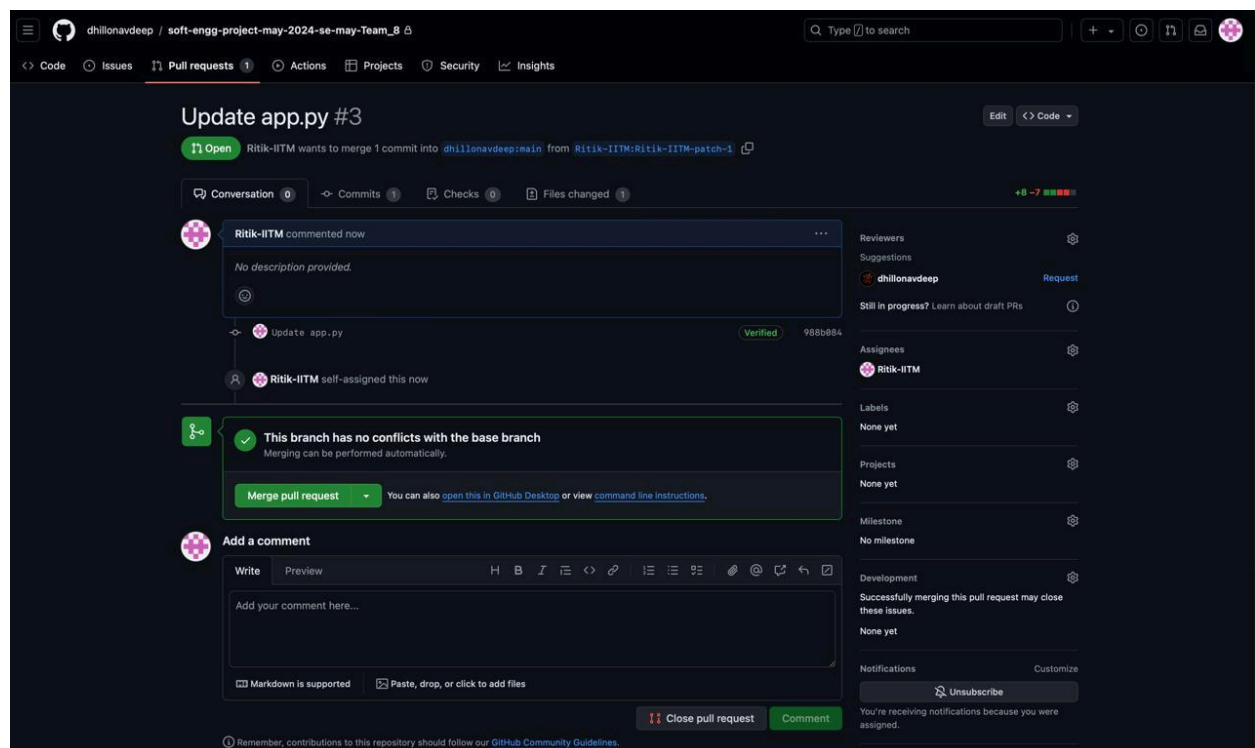
Development Sprint Reviews

Throughout our project, we held comprehensive weekly sprint reviews that played a crucial role in keeping our development process on track. These reviews involved key stakeholders, including the developers, product manager, and scrum master. The primary objectives of these reviews were to assess the team's progress, identify and resolve any issues that had arisen during the sprint, and collaboratively plan the next sprint.

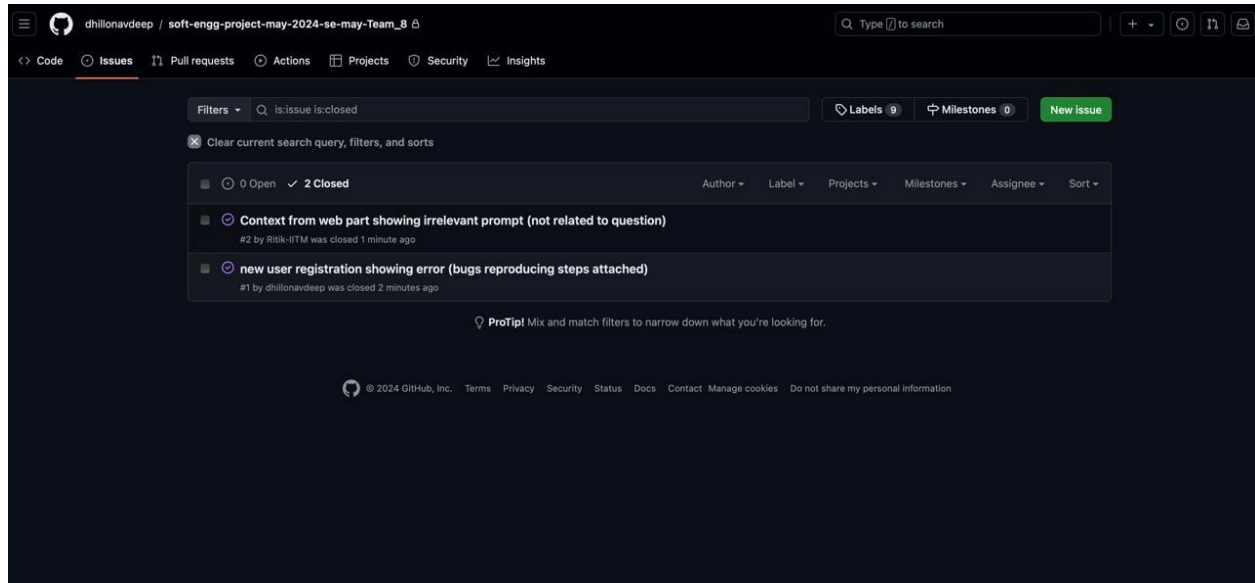
The involvement of the product manager ensured that the team's work remained aligned with the product vision and customer requirements. Meanwhile, the scrum master facilitated these meetings, helping to remove any impediments that were blocking the team's progress. By the end of each review, we had a clear understanding of what had been achieved, what needed improvement, and the tasks that needed to be tackled in the upcoming sprint. This process of regular assessment and adjustment ensured that the project moved forward efficiently, with minimal delays or disruptions.

Code Review on Git:

Pull Requests



Issue Raised



Tools and Technologies Used

Our development process was supported by a robust set of tools and technologies that facilitated efficient project management, version control, coding, and testing. For project management, we utilized JIRA/ATLAS, which provided a powerful platform for tracking tasks, bugs, and project milestones. The Kanban boards within JIRA allowed us to visualize the flow of tasks, from development to completion, ensuring that everyone on the team was aware of the current state of the project.

Git served as our version control system, enabling us to manage code changes across multiple branches and environments seamlessly. The use of Git allowed for collaborative development, where multiple developers could work on different parts of the project simultaneously without the risk of overwriting each other's work. This also facilitated code reviews and integration, as changes could be easily tracked and merged.

Integrated Development Environments (IDEs) were used by developers to write, test, and debug their code. These IDEs provided essential features like code completion, syntax highlighting, and debugging tools, which significantly enhanced productivity. For testing, especially API testing, we employed specialized tools that enabled us to perform unit testing and ensure the reliability of our APIs. These tools were essential in catching errors early in the development process, which helped maintain the stability and performance of the application.

#	Tools/Technology	Usage
1	Adobe	Designing and editing content
2	Flask	Python Web Framework
3	Flask-SQLAlchemy	ORM Integration
4	Flask-Restful	API Framework
5	Git/Github	Version Control System
6	Google Docs	Collaborative Documents
7	Google Mail	Communication and Sharing Resources
8	Google Meet	Meetings, Managing Team
9	Google Sheet	Issue tracking for end user testing
10	Google Slides	Used for making Presentation
11	HTML,CSS,JS	Base Technologies for web development
12	JIRA	PM tool for tracking tasks, issues etc
13	Kanban	A board system used for managing tasks
14	MS Powerpoint	Presentation and announcement Slides
15	MS Word	Documentation and announcement
16	ollama	Language model
17	PyTest	Python testing framework
18	SQLite	Application Database
19	VsCode	IDE for developing code

How We Worked

Our development process was grounded in the Scrum methodology, which emphasizes iterative progress through regular, structured meetings. Scrum meetings brought together the scrum master, product manager, and the scrum teams to ensure that everyone was on the same page. These meetings were held regularly and served multiple purposes.

Daily stand-ups were brief meetings where team members provided updates on what they had accomplished the previous day, what they planned to do that day, and any obstacles they were facing. This ensured that the team remained aligned and any issues were addressed promptly.

Sprint planning meetings were held at the beginning of each sprint to plan the work for the upcoming iteration. The product backlog was reviewed, and tasks were assigned based on priority and team capacity. These meetings were crucial in setting clear expectations and goals for the sprint.

At the end of each sprint, sprint review meetings were held to assess the completed work. During these reviews, the team demonstrated the functionality they had developed, and stakeholders provided feedback. This feedback loop was vital for ensuring that the product evolved in line with user needs and expectations.

Lastly, sprint retrospectives were conducted to reflect on the sprint and discuss what went well, what didn't, and how the team could improve in the next sprint. These retrospectives fostered continuous improvement by allowing the team to learn from their experiences and make adjustments to their process.

Step 1.Availability Calendar	Meetings scheduled based on availability. We have created an availability calendar to manage planned absences and common events (quizzes, festivals).
Step 2. Communication and Resource Sharing	Apart from our personal WhatsApp and Gmail accounts, we are also using a common WhatsApp Group to consolidate and streamline communications.
Step 3.Attendance Management	Attendance management and recording help track participation, maintain accountability, and ensure everyone stays informed by providing access to meeting records.
Step 4. Publish Team Calendar	The all-hands meet is scheduled two days in a week at 10:00 to 11:00PM. Scrum teams & milestone delivery teams schedule additional meets.
Step 5.Agenda & Meeting Report	The meeting invites include the agenda and prep work required. After meetings, a formal meeting report is published to all members.

Meeting Report

Join with Google Meet

meet.google.com/umc-gyve-tha

Join by phone

(US) +1 575-518-3121 PIN: 213 191 710#

Take meeting notes

Start a new document to capture notes

8 guests

1 yes

7 awaiting

Ritik Arora

Organizer

Set your working location

NAVDEEP

ANSARI MOHD TABISH FATEH MOHD (21f20014...

MILAN

VEDANSHI TEWARI

Rohan Ajay Ramani

SHUBHAM GATTANI

TAMANNA TAK

The Milestone 03 final review before submission.

Milestone 03 Document

: <https://docs.google.com/document/d/1Jc03foN3r-Bm-CjBc9Vu3FU4IMm8htvPvC6xCa8WpoE/edit>

Upcoming Next: MILESTONE 04

We will be closing Milestone 03 requirements by Monday, 22 July.

10 minutes before

Date	4 July 2024 22:00 to 23:00	
Agenda	Milestone 3 Discussion and Progress Updates	
Meeting Link	https://meet.google.com/lob-jcai-yoe	

Attendance

1	Milan	<input checked="" type="checkbox"/>
2	Ritik Arora	<input checked="" type="checkbox"/>
3	Vedanshi Tiwari	<input checked="" type="checkbox"/>
4	Tamanna Tak	<input checked="" type="checkbox"/>
5	Navdeep	<input checked="" type="checkbox"/>
6	Shubham Gattani	<input checked="" type="checkbox"/>
7	Rohan Ajay Ramani	<input checked="" type="checkbox"/>
8	Tabish	<input checked="" type="checkbox"/>

Table of Content

1. [Review of last meeting](#)

2. [Planning Milestone 3 and 4](#)

3. [Development Preparation Tasks](#)

4. [Open Discussion](#)

5. [Meeting schedule for Feb-Mar 2024](#)

My Drive > SE Project ▾ 👤

Type ▾

People ▾

Modified ▾

Name	Owner
Meeting Recordings	me
Meeting Notes	me

Acknowledgement :

We would like to express our sincere gratitude to the Software Engineering Course Management Team at IIT Madras for their invaluable support throughout our project. Your willingness to provide assistance and guidance, even when we faced delays, was instrumental in our progress. Your prompt and clear responses to our queries greatly facilitated our work and ensured we remained on track. We deeply appreciate the time and effort you dedicated to helping us succeed, and your dedication has significantly contributed to our learning experience. Thank you for your unwavering support and commitment.