

Milestone 06
30 March 2025

Software Engineering Project



Team 29

Ajay Thiagarajan (21f1003242)

A J R Vasu (21f3002975)

Anand K Iyer (21f1001185)

Anuj Gupta (21f3001598)

Ghanashyam R (21f1003387)

Jalaj Trivedi (21f2000730)

Niraj Kumar (21f1006589)



Milestone 6 - Project Submission

S. No	Item	Page #
1	About the project	3
2	Milestone 1 - 5 recap and summary	4
3	Milestone 6 - Project submission	50



Acknowledgements

We, Team 29 of the Software Engineering Project (January Term), extend our gratitude to everyone who contributed to our journey throughout this project.

We sincerely appreciate our IIT Madras and the Software Engineering course team, professors, mentors and clients their support and guidance.

And finally thank you to team 29 for coming this far in the journey collectively !

Meet The Team!

TEAM 29
Software Engineering Project

Meet The Team

 A J R Vasu 21F3002975	 Anand K Iyer 21F1001185	 Ajay Thiagarajan 21F1003242	
 Niraj Kumar 21F1006589	 Anuj Gupta 21F3001598	 Ghanashyam R 21F1003387	 Jalaj Trivedi 21F2000730





About the Project

Integrating Generative AI into the SEEK Learning Portal

With the rapid advancement of generative AI (GenAI) technologies, there are exciting opportunities to enhance learning experiences by integrating GenAI into self-paced platforms like the SEEK portal. The SEEK portal serves as a self-learning environment, offering lecture videos, resource materials (transcripts), programming quizzes, and links to discussion forums where students can post subject-specific queries.

The main idea is to find out how GenAI can be added to SEEK to make learning better and more interactive.

What SEEK Offers

i) Learning

- Video lectures that students can watch at their own pace.
- Resource materials and transcripts to help understand the topics better.
- Programming quizzes and graded assignments to test knowledge.

ii) Coding

- A built-in code editor where students can write code and instantly see the output.
- Students can also submit both practice and graded assignments to get their work checked.

How GenAI Can Help

Chatbot Support

A GenAI chatbot can answer students' questions about course content, videos, or any doubts they have while learning.

Code Help

While working on programming problems, GenAI can guide students with tips on syntax, good coding practices, and fixing errors.

Feedback

After submitting code, GenAI can suggest better or more efficient versions of the code and explain how they work.





Milestone 1





Objective

The objective of Milestone 1 was to systematically identify and define the key user groups of the SEEK portal—Primary (Students), Secondary (Faculty), and Tertiary (Admins, Developers, ML Research/Ops Teams)—and outline their specific requirements. This milestone aimed to enhance the existing system by integrating Gen AI-powered features that improve user experience, academic engagement, and administrative efficiency.

A crucial focus was placed on writing user stories following the SMART (Specific, Measurable, Achievable, Relevant, and Time-bound) guidelines. These user stories highlighted the AI-driven enhancements that will be incorporated into the SEEK portal, ensuring that students receive personalized learning support, faculty gain data-driven insights, and administrators have better engagement tracking tools.

1) USER TYPES

We have 3 user types for this application and a brief description is given below.

i) Primary Users

Students are the primary user for this application. They use the SEEK portal for accessing academic material and guidance as part of their academic program.

The Gen AI based enhancements will primarily focus on the requirements of the students as they are the key stakeholders and beneficiaries of the academic program.



ii) Secondary Users

Faculty which comprises teaching assistants and instructors.

They will be responsible for managing the study material, assignments, organizing and taking the live session and interacting with students to clarify their doubts.





iii) Tertiary Users

Institution's Administrators (admin), the team responsible for maintaining data assets on SEEK, managing study material, assignments, monitoring portal's usage and managing users' access to SEEK.



Developers, the team which will maintain/ develop the code based and maintain the SEEK infrastructure.

Assumptions

Before diving into the user requirements and stories, below are the list of assumptions we have made. Based on these assumptions we will omit some of the obvious requirements, which we assume are already part of the portal. Consequently, the user stories focused more on the AI agentic enhancements.

Further, we are focusing more on the three users' groups, namely, students, faculty and the admin team.

1. Students can browse through their,
 - a. Weekly content such as slides and lecture videos
 - b. They can view and respond to their assignments including programming assignments
 - c. They can access their profile page and view details
2. The faculty team can,
 - a. Upload/ view weekly content
3. The admin team,
 - a. Can upload/delete content
 - b. They have privileges to add/ remove access to students and faculty members
 - c. They can view activities by students and faculty

2. User Requirements

2.1 Students

- Students should be able to log in using their institute-provided credentials to simplify the sign-in process and ensure secure authentication.
- Over and above accessing study material, students should have access to a searchable knowledge graph to retrieve relevant course materials and additional resources efficiently for their topics of interest.





- Students should be able to extract concise summaries from multiple sources related to their query topics.
- Enable students to save summarization results as formatted PDF files for offline access and sharing.
- Students should get intelligent code suggestions, including error corrections and debugging tips, to help them improve their programming skills.
- Students should receive reminders for upcoming assignment deadlines, quizzes, and other academic events.
- Students should be able to view and mark tasks and milestones as completed, including deadlines and calendar views.

2.2 Faculty

- Over and above uploading content, provide faculty with analytics to identify the most queried topics by students over the last week, helping them address knowledge gaps in live sessions.
- Provide faculty with data on the most error-prone programming questions over the last week, enabling them to focus on these issues in live programming sessions.
- Provide a dashboard that gives an overview of student progress, including queried topics, engagement metrics, and milestones.
- Suggest live session content to the faculty, based on student queries and programming errors, making sessions more relevant to the students' needs.

2.3 Admin

- Implement clear role-based access for students, faculty, and admins, ensuring secure and appropriate access to system components.
- Provide admins with a dashboard to track student engagement, including metrics like resource usage, query trends, and attendance in sessions.
- Enable admins to send automated reminders and nudges to students based on engagement metrics to improve their academic involvement.

3. User stories

STUDENTS



1. Feature: Single Sign On

As a student,

I want a simplified sign in process which is integrated with my institute ID so that accessibility is easier.





2. Feature: Knowledge Graph

As a student,

I should be directed to relevant course materials and additional learning resources when searching for a topic of interest, so that I can find information quickly and efficiently.



3. Feature: Summarization

As a student,

I want to extract a summary response from across sources for my query topic, so that I have an overview of the topic.



4. Feature: Notes

As a student, I want to save a formatted PDF copy of the summarization response, so that I can access this information anytime later and share with fellow students.



5. Feature: Programming and Debugging

As a student,

I want suggestions on how to correct my code for my programming assignments and practice, so that I can improve my coding skills.



6. Feature: Smart Reminders

As a student,

I want reminders about upcoming assignments and deadlines, so that I do not miss any deadline.



7. Feature: Quiz Master

As a student,

I want to get a summary of the questions and the answers I queried in that term for a particular course so that I can revise the hot topics for the quiz.





FACULTY

8. Feature: Knowledge Gaps

As faculty,

I should be able to access the most queried topics (top 10, over the last one week) so that I can address these issues in live sessions.



9. Feature: Coding Challenges

As faculty,

I should be able to identify the most error-prone programming questions (top 5, over the last one week) so that I can address these issues in live programming sessions.



ADMIN

10. Feature: Engagement Metrics

As the admin,

I should see students' engagement metrics so that reminders can be sent to them to nudge them towards better involvement.



Conclusion

Milestone 1 successfully identifies all user roles and clearly defines their functionalities and requirements within the SEEK portal. The user stories ensure that the AI-driven features align with user needs while maintaining SMART principles. These stories emphasize student-centric enhancements like knowledge graphs, AI-driven summarization, intelligent coding assistance, and automated reminders, alongside faculty analytics and admin engagement tools.

By following a structured approach, this milestone lays the groundwork for the implementation phase, where these features will be integrated into the existing system. The next steps involve refining these user stories based on peer evaluations and ensuring alignment with development priorities before moving toward implementation.





Milestone 2





Objective

The goal of Milestone 2 was to design the user interface (UI) of the AI Agent for Academic Guidance. This involved creating a storyboard to visually represent user interactions and developing low-fidelity wireframes for each identified user story. The UI design will follow usability heuristics and best practices to ensure an intuitive, accessible, and user-friendly experience. These wireframes serve as the foundation for the final UI, ensuring that user requirements are effectively translated into a structured interface.

Storyboards

Meet DeepSEEK' Key Users

This storyboard illustrates the key users of DeepSEEK:

- 1 Student:** A boy sitting at a desk, writing in a notebook. He says: "I use SEEK to access study materials, submit assignments and progress through my degree."
- 2 Faculty:** A man standing, holding a book and a pen. He says: "I manage course content, track students' progress and conduct live sessions to support them further."
- 3 Admin:** Three people in a meeting room. One person is speaking while others listen. They say: "I ensure the SEEK portal runs smoothly by managing user access and monitoring engagement."

Easy, Secure Access

Student, Faculty and Admin

This storyboard illustrates the easy and secure access for different users:

- Student:** A boy is shown working on a laptop. A thought bubble says: "MLT week -2 deadline is today, let me re-check my answers".
- Faculty:** A monitor screen shows the "IIT MADRAS" login page with options to log in or create an account via Google, Facebook, or Apple.
- Admin:** A monitor screen shows the "IIT MADRAS" dashboard for "LECTURE 2 MLT" with navigation links for MODULE, INTRODUCTION, WEEK 1, WEEK 2, and WEEK 3. It also features a "LEAVE A REVIEW" section and a message box: "HEY! I AM SEEK PORTAL. I AM YOUR VERY OWN AI ASSISTANT. ASK AWAY! CONTACT ME!"



Knowledge Graph for a consolidate view

Student

Hey Alfred, what is the difference between a perceptron and a neural network?

Hey, this article from Medium.com is great. I must SUMMARISE it for reference for later

Hey Alfred, what is the difference between a perceptron and a neural network?

Alfred's:

- Sure! A perceptron is a single-layer neural network and one of the simplest models used for binary classification.
- A neural network, on the other hand, consists of multiple layers (input, hidden, and output layers).

"Here are some resources to dive deeper into this topic:"

- Lecture Slides: Introduction to Neural Networks - Prof. John Doe
- Video Lecture: What is a Neural Network? - YouTube
- Blog Post: Understanding Perceptrons - Towards Data Science
- Wikipedia: Perceptron

Multi-Source Summarization

Student

This is an important topic, and there is a lot of content across resources. I need to save the key points for reference

Alfred, can you please give me a one-page summary across all these resources and do mention the sources in the document along with the website links.

This summary is perfect! I can share it with my team as well

Alfred's:

- "Neural networks are inspired by the human brain and consist of layers of neurons."
- "Key concepts: perceptrons, activation functions, and neural network architecture."
- "Applications: image recognition, natural language processing, etc."
- "Source 1: Lecture Slides – Prof. John Doe"
- "Source 2: Neural Networks Explained – YouTube Video"
- "Source 3: Understanding Neural Networks – Data Science Blog"



Interactive Coding with AI Assistance

Student

Hey Alfred, can you explain what's wrong with my code?

```
def calculate_sum(a, b):
    return a + b

result = calculate_sum(5, "3")
print(result)
```

Hey Alfred, can you explain what's wrong with my code?

Alfred's:

- Error detected on line 3: Type mismatch.
- Tip: Convert `b` to an integer using `int(b)` before adding.
- Example fix: `return a + int(b)`

Oh, I see the error now. Thanks for the debugging tip, Alfred!

Riddle me, Quiz Master

Student

Hey, MLT quiz is not going to be easy.

My graded assignment scores are not good.

Let me revise the key topics.

Let me ask Alfred what the important topics are.

MAY
Sun Mon Tue Wed Thu Fri Sat
Canva 23

Alfred, can you tell me what topics I should revise for next week's quiz?

Alfred's:

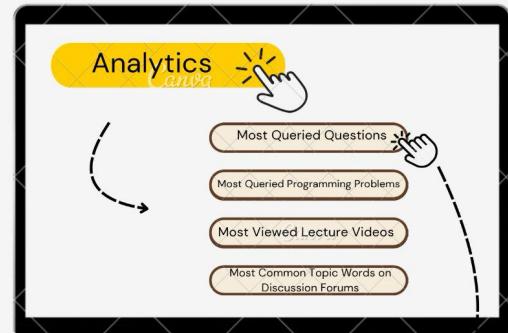
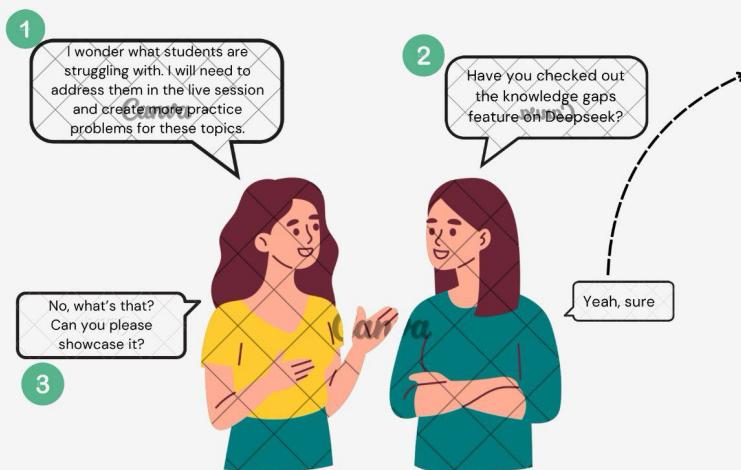
- After analyzing the last three terms' PYQs, questions on perceptron's threshold values, neural network architecture, and activation functions are important.
- Based on your queries, questions on neural network architecture are weak spots.
- Most students are also revising these three topics.

Great! I'm going to start with these topics and revise all the problems.



Which Knowledge Gaps to plug?

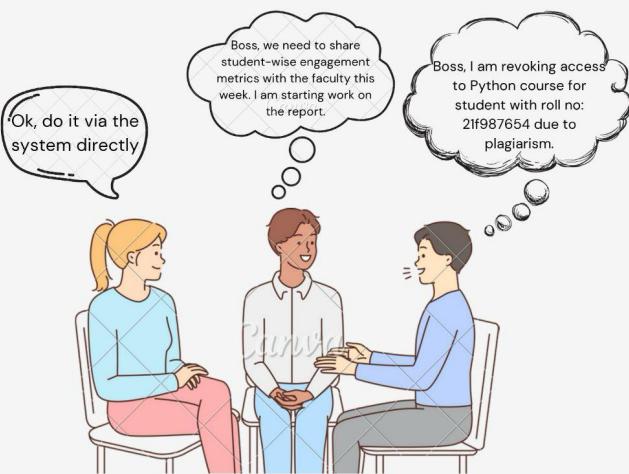
Faculty



Rank	Topic
1	Threshold in perceptron
2	How many neurons in a neural network
3	Vanishing gradient
4	Biases in neural networks
5	Overfitting vs underfitting
6	Activation functions
7	Backpropagation
8	Dropout regularization
9	Gradient descent optimization
10	Weight initialization techniques

Engagement Metrics

Admin





Wireframes

Single sign-on enabled registration and access

The wireframe shows a landing page for the DeepSEEK Portal. At the top, there's a logo with a lightbulb icon inside a curly brace. Below it, the title "DeepSEEK Portal" is displayed. A sub-headline "Powered by **Alfred** our AI agent" is followed by a small cartoon character of Alfred. To the right are two buttons: "Get Started" (in black) and "Register" (in green). On the left, under "Key features:", there's a bulleted list: Watch lectures, Solve assignments, Ask doubts to AI, Share your learning with peers, Get coding support from AI, Smart reminders, and Insights for quiz support.

Resource page for accessing content by the student

The wireframe displays a resource page for a Deep Learning course. On the left, a sidebar lists navigation items: Course: Deep Learning, Week1 (with a dropdown arrow), L1: What is Perceptron?, Activity Questions, Programming Assignments, Week2 (with an upward arrow), and Supplementary contents. The main content area has a title "Week1 L1: What is Perceptron?". It includes a table of inequalities:

x_1	x_2	OR
0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

Below the table, there's a graph showing a 2D coordinate system with axes x_1 and x_2 . A red shaded region represents the solution set of the inequalities. A blue line is drawn through the origin with the equation $-1 + -0.5x_1 + 1.1x_2 = 0$. A video player at the bottom right shows a man speaking, with controls for volume, playback, and transcript.



Question and answer, knowledge graph from AI chatbot - Alfred

DeepSEEK

Course: Deep Learning Week1 ▼ L1: What is the Perceptron? Activity Questions Programming Assignments Week2 ▲ Supplementary contents	<p>Week1 L1: What is the Perceptron?</p> <p>First gene Perceptron Ivakhnenko</p> <p>Ask Alfred</p> <p>Alfred: Hi! I'm here to help with your studies. Do you have any questions for me?</p> <p>Me: What is the difference between a perceptron and a neural network?</p> <p>Alfred: a perceptron is the simplest form of a neural network with just one layer, while neural networks are more complex structures with multiple layers capable of handling more complex tasks. Following resources will provide you with more detailed information and examples about perceptrons and neural networks.</p> <p>1. Perceptrons and Neural Networks: Basic Principles of Computer Vision 2. Multilayer Perceptrons In Machine Learning: A Comprehensive Guide</p> <p>Save</p>
--	--



Interactive coding and support from AI chatbot - Alfred

DeepSEEK

Course: Programming in C Week1 ▲ Week2 ▲ Week3 ▼ L1: Structured Programming Activity Questions Programming Assignments Week4 ▲ Supplementary contents	<p>Week1 PPA2</p> <pre>C AA + - ✎ ☐</pre> <p>1 #include <stdio.h> 2 void main() { 3 int Y, leap = 0; 4 scanf("%d", &Y); 5 if (Y % 4 == 0) { 6 if (!(Y % 100 == 0) && !(Y % 400 == 0)) { 7 leap = 1; 8 } 9 } 10 if (leap) { 11 printf("%d is a leap year", Y); 12 } else{ 13 }<p>Test Run Results</p><p>Summary: test.c: In function main: test.c:7: error: expected) before mod if (!(Y % 100 == 0)) { Public Test: 0/0 Passed</p><p>Ask Alfred</p><p>Alfred: Hi! I'm here to help with your programming queries. Do you have any questions for me?</p><p>Me: What is the highlighted error in my program? How to solve it?</p><p>Alfred: The error you're encountering is because the mod operator is not valid in C. In C, the modulo operation is performed using the % operator. Additionally, there might be a misplaced parenthesis. Here's the corrected version of your condition: if (!(Y % 100 == 0) && !(Y % 400 == 0)) { This will ensure that the modulo operation is correctly performed, and the syntax is valid. If you have more questions or need further assistance, feel free to ask!</p><p>Send</p></p>
--	--





A screenshot of the DeepSEEK platform showing a student's dashboard. On the left, a sidebar lists course modules: Week1 (up), Week2 (up), Week3 (down), L1: Structured Programming, Activity Questions, Programming Assignments, Week4 (up), and Supplementary contents. The main area shows a code editor for 'Week3 PPA2' with the following C code:

```
1 #include <stdio.h>
2
3 void main() {
4 }
```

A modal window titled 'Smart Reminders' displays a list of upcoming tasks:

- MLT Week2 (Due Wednesday)
- Python Week2 (Due Sunday)
- Programming in C (Due Sunday)
- Upcoming quiz1 on 21st, Feb

Below the code editor, 'Test Run Results' show a summary of a test run for 'test.c'. A message box from 'Alfred' asks about a highlighted error in the code, and a response provides a corrected version.

A screenshot of the DeepSEEK platform showing a 'Quiz Master' section. At the top, there are three green buttons: 'Most Queried Topics', 'GA Mistakes', and 'PYQ Topics'. Below them are three columns of frequently asked questions:

Most Queried Topics	GA Mistakes	PYQ Topics
1. Perceptron 2. Sigmoid 3. Momentum 4. AdaGrade 5. Convergence	1. Week2 Q3 2. Week2 Q4 3. Week3 Q8 4. Week4 Q2 5. Week4 Q10	1. Update Rule 2. Momentum 3. Neurons In NN





Profile and complete access of resources

The screenshot shows the 'Student Profile' section of the DeepSEEK application. At the top, there is a placeholder for a profile picture with the name 'John Doe' below it. To the right, detailed student information is listed: Program (BS Degree in Data Science and Application), Level (Diploma), DOB (dd/mm/yyyy), Course Registered (dd/mm/yyyy), and Address (IIT Madras). Below this information are several navigation buttons: 'Student Details', 'Course done', 'Grade Card', 'Documents', 'Payments', and 'Quiz'. In the top right corner, there is a 'Logout' button.



Accessing and managing content for the Faculty

The screenshot shows the 'Resources for Deep Learning T1 2025' section for the faculty. It displays the current week (Week 2 3- 10 Feb 2025) and the number of students enrolled (252). On the left, there is a profile picture of 'Faculty DL' with their details: Program (Lead Faculty for Deep Learning), Level (Degree), and Email (faculty@email.com). To the right, there are several buttons for managing content: 'Weekly content', 'Practice Assignments', 'Graded Assignments', 'Coding Assignments', and 'Notes'. A green box on the right lists 'Trending queries for Week 02': 1. Perceptron threshold, 2. AdaGrad, 3. Sigmoid, 4. Momentum, 5. Convergence.

Conclusion

By the end of this milestone, a comprehensive storyboard and low-fidelity wireframes has been completed, aligning with the defined user stories. The designs adhere to usability principles, providing a clear and structured layout for the AI agent's interface. This milestone ensures that the UI is user-centric, easy to navigate, and functionally efficient, setting the stage for further development in subsequent milestones.





Milestone 3





Objective

Milestone 3 focused on project management and front-end development to ensure the seamless execution of tasks and alignment with project goals. This involved:

- Defining components, epics, user stories, and tasks for effective sprint planning.
- Setting up the backlog and organizing development priorities.
- Finalizing the system architecture, including the class diagram, database schema, ER diagram, and RAG framework.
- Developing the front-end components for user management, course management, and learning modules.

1. Project Schedule

1.1. Task Distribution

1. The entire project was divided into SDLC segments of Planning, Design, Development, Testing and Implementation as prescribed by the requirements.
2. The sprints were aligned to meet these deadlines.
3. The project segments and consequently the milestones were aligned to team members.
4. Finally, the sprints were aligned to meet the milestone deadlines while keeping the overall project progress in mind.

Milestone #	Milestone	SPOC	Key Roles
01.	User requirements/ stories	Vasu	• Vasu, Ghanashyam
02.	Storyboarding	Ghanashyam	• Ghanashyam
03.	Project Management, Front End	Vasu	• Vasu for Jira • Anuj for Front End • Jalaj for RAG Modeling
04.	APIs	Ajay	• Ajay for APIs • Niraj for RAG Backend
05.	Testing	Anand	• Anand
06.	Implementation	Jalaj	• Jalaj, Ajay, Niraj, Anuj





1.2 Roles by Team Members

Key roles and development responsibility of the team members is aligned based on their strengths and interests.

S.No	Team Member	Key Roles
01.	Anuj Gupta	Front End
02.	Ghanashyam R	Planning, Research, Design
03.	A J R Vasu	Scrum, Product Development
04.	Jalaj Trivedi	RAG R&D, development and implementation
05.	Niraj Kumar	RAG R&D, Backend development and implementation
06.	Anand Iyer	Testing and Correctness
07.	Ajay T	Application Backend

2. Sprints' Schedule

The Sprints have been aligned to both meet the milestone deadlines and keep the overall project progress on track.

2.1. Sprints' details

1. There are overall 8 sprints covering user requirements, storyboarding, UI development, implementing backend models/ logic, API development, RAG development, integration of all components and integrated testing.
2. The schedule for the sprints is given below.

Sprint #	Topic/ EPIC	Start Date	End Date
Sprint-01	Identifying User Requirements	20 January 2025	26 January 2025
Sprint-02	User Interfaces and Wireframes	28 January 2025	02 February 2025
Sprint-03	Project Management	03 February 2025	09 February 2025
Sprint-04	Front End Development	10 February 2025	18 February 2025
Sprint-05	API Dev and Implementation	19 February 2025	28 February 2025
Sprint-06	AI Learning Assistant Development	03 March 2025	12 March 2025
Sprint-07	APIs, RAG and UI Integration	17 March 2025	23 March 2025
Sprint-08	Integrated Testing	24 March 2025	28 March 2025

2.2 Gantt Chart





1. We used Jira for project management.
2. Below is the Sprints timelines

Sprints	JAN	FEB	MAR				
User req...	Story...	Project...	Front End...	API Dev an...	AI Learning...	API, R...	Inte...
DS-13 Identifying User Requirements	DONE						
DS-14 Identify user types for the application	DONE	AJR VASU					
DS-15 Identify requirements by user type	DONE	GHANASH...					
DS-16 Identify features required for fulfilling user needs	DONE	AJR VASU					
DS-17 Build user stories for the features	DONE	GHANASH...					
DS-18 Complete milestone 01 report	DONE	AJR VASU					
DS-19 User Interfaces and Wireframes	DONE						
DS-20 Storyboarding	DONE	GHANASH...					
DS-22 Wireframes	DONE	NIRAJ KUM...					
DS-23 Complete Milestone 02 report	DONE	AJR VASU					
DS-105 Project Management							
DS-106 Jira for Scrum and Sprints	DONE	AJR VASU					
DS-107 Github for software development	IN PROGRESS	AJR VASU					
DS-117 Complete Milestone 03 report	IN PROGRESS	AJR VASU					
DS-24 User Management							
DS-25 User registration and login	DONE	ANUJ					
DS-35 RBAC	DONE	AJAY					
DS-53 Course Management							
DS-54 Course Enrollment	DONE	ANUJ					
DS-58 Learning Management System							
DS-59 Video player	DONE	ANUJ					
DS-60 Quizzes	IN PROGRESS	ANUJ					
DS-61 Coding Assignments	IN PROGRESS	AJAY					
DS-120 API Dev (non RA)							
DS-121 API end points	IN PROGRESS	AJAY					
DS-122 Documentation	TO DO	AJAY					
DS-130 Complete Milestone 04 report	TO DO	ANAND IYER					
DS-73 AI Learning Assistant							
DS-74 Q & A	TO DO	JALAJ T...					
DS-75 Summarization	TO DO	JALAJ T...					
DS-86 AI Powered Quiz Support							
DS-87 Hints/solutions for quiz	TO DO	NIRAJ KUM...					
DS-95 AI Powered Coding Support							
DS-97 Hints/solutions for coding assignments	TO DO	NIRAJ KUM...					
DS-132 API Integration							
DS-133 API Integration across the entire Application	TO DO	AJAY					
DS-136 Integrated Testing							
DS-137 API app end points testing	TO DO	ANAND IYER					
DS-138 API RAG end points testing	TO DO	ANAND IYER					
DS-139 Milestone 05 report	TO DO	ANAND IYER					





3. Sprints Scrum Board

The snapshot of the Kanban scrum board is given below,

Projects / DeepSEEK-Team-29
All sprints

Search AV A A AI NK JT +2 🔍 Epic Type Sprint

TO DO 10	IN PROGRESS 5	DONE 8
Q & A AI LEARNING ASSISTANT DS-74 JT	Quizzes LEARNING MANAGEMENT DS-60 A	Storyboarding USER INTERFACES AND WIREFRAMES DS-20 AV
Summarization AI LEARNING ASSISTANT DS-75 JT	Coding Assignments LEARNING MANAGEMENT DS-61 A	Wireframes USER INTERFACES AND WIREFRAMES DS-22 NK
Hints/solutions for quiz AI POWERED QUIZ SUPPORT DS-87 NK	Github for software development PROJECT MANAGEMENT DS-107 AV	Complete Milestone 02 report USER INTERFACES AND WIREFRAMES DS-23 AV
Hints/solutions for coding AI POWERED CODING SUPPORT DS-97 NK	Complete Milestone 03 PROJECT MANAGEMENT DS-117 AV	User registration and login USER MANAGEMENT DS-25 A
Documentation API DEV (NON RAG) DS-122 A	API end points API DEV (NON RAG) DS-121 A	RBAC USER MANAGEMENT DS-35 A
Complete Milestone 04 API DEV (NON RAG) DS-130 AI		Course Enrolment COURSE MANAGEMENT DS-54 A
API Integration across the entire App API INTEGRATION DS-133 A		Video player LEARNING MANAGEMENT DS-59 A
API app end points testing INTEGRATED TESTING DS-137 AI		Jira for Scrum and Sprints PROJECT MANAGEMENT DS-106 AV
API RAG end points testing INTEGRATED TESTING DS-138 AI		
Milestone 05 report INTEGRATED TESTING DS-139 AI		





4. Scrum Meeting Details

Sprint 1 Scrum meetings minutes/details

Dates: 15 January, 2025 - 25 January, 2025

- The team started by finalizing roles within the project. Each member was assigned responsibilities based on their expertise and project requirements.
- Following this, user types and their requirements for the problem statement (SEEP portal) were identified.
- Subsequently user stories were developed to represent different users' interactions with the system.

Sprint 2 Scrum meetings minutes/details:

Dates: 26 January, 2025 - 2 February, 2025

- The team began work on milestone-2, starting with storyboard creation using Canva.
- Sample storyboards and wireframes were presented and discussed, allowing team members to provide feedback and refine the designs.

Sprint 3 Scrum meetings minutes/details:

Dates: 3 February, 2025 - 9 February, 2025

- The team will shift focus to milestone-3, which involves defining components, epics, stories, tasks, and sprint planning.
- Subsequently, the team discussed all the backlogs and set up the sprints basis mutually agreed timelines.
- Following this, a session was conducted to finalize the system architecture, including the class diagram, database schema, ER diagram and RAG framework and workflow which are also required for the milestone-3 report.

Sprint 4 Scrum meetings minutes/details:

Dates: 10 February, 2025 - 18 February, 2025

- Following sprint 3, discussions and demo were held to review the front-end implementation for user management, course management and learning management modules, ensuring it aligns with project requirements.
- Based on the feedback, the UI was further refined to accommodate the client suggestions and our own internal feedback.





5. Application Design Components

5.1 Front End

- The front end of DeepSEEK portal, the learning application, is built using React.js.
- It designed as a self-paced learning portal with AI assistance for more effective learning.
- The main components are,
 1. **Lecture Videos:** Allows students to watch lectures and access an AI Chatbot for queries.
 2. **Practice Assignments:** Offers interactive questions with AI assistance for hints, detailed explanations, and additional practice.
 3. **Programming Challenges:** Enables code submission with test case evaluation, plus AI support for debugging and optimizing code efficiency.
 4. **Assessments & Performance:** Supports graded assignments and provides AI-generated performance reports to highlight strengths and areas for improvement.
 5. **Dashboard:** Supports understanding of frequently asked questions to help revise.

5.2 Frontend UI (updated screens)

- The AI assistant chatbot is available all through the UI for asking concept led questions.
- In case of quiz questions and coding support the AI assistant in embedded and provides hints and debugging support.

Viewing videos, interacting with chatbot

The screenshot shows the DeepSEEK learning portal. On the left, there's a sidebar with navigation links for 'Problem Solving and Data Structures Algorithms (PDSA)', 'Week 1: Python Recap and ...', 'Week 2: Searching and Sortin...', 'Week 3: Trees and Graphs', 'Week 4: Dynamic Programming', 'Week 5: Greedy Algorithms', and 'Week 6: Advanced Topics'. A central video player is displaying '1.2 – Python Recap-2 - Video' titled 'W1L4_Python Recap - II'. The video content shows a graduation cap on a laptop screen with the text 'IIT MADRAS ONLINE DEGREE PROGRAMME'. Below the video, there's a transcript of the video content. On the right side, there's an 'Alfred: AI Assistant' section with a message history and a sidebar for 'What is Classification?' containing text about classification algorithms like logistic regression and decision trees. At the bottom, there's a footer with the text '© 2025 DeepSEEK. All rights reserved.'





Login and registering for courses

Welcome to the DeepSEEK Portal

Course Registration

Full Name
ANUJ GUPTA

Email
21f300159@stds.study.iitm.ac.in

Select Courses:
You can select multiple courses.

Machine Learning Fundamentals (MLF)
 Learn the fundamentals of Machine Learning, including regression, classification, and dimensionality reduction.

Problem Solving and Data Structures Algorithms (PDSA)
 Learn problem-solving techniques and data structures algorithms, including sorting, searching, and binary search.

Register

Coding and debugging support via AI

Graded Programming: Goldbach's Conjecture

Deadline: No deadline provided

Write a function to find prime pairs that sum to an even number.

```
1 v def prime(n):
2 v     if n < 2:
3 v         return False
4 v     for i in range(2, n//2 + 1):
5 v         if n % i == 0:
6 v             return False
7 v     return True
8 v
9 v def Goldbach(n):
10 v    Res = []
11 v    for i in range((n//2) + 1):
12 v        if prime(i) == True:
13 v            if prime(n - i) == True:
14 v                Res.append((i, n - i))
15 v    return Res
16 v
17 v n = int(input())
18 v print(Goldbach(n))
```

Light Mode

Hide Hint Debug Submit

💡 Hint: Try breaking the problem into checking prime numbers and finding pairs.

The provided code has an issue in the Goldbach function where it only checks up to $n/2$ for prime pairs, it should check up to n . The corrected code should iterate over a range that covers all possible pairs of primes that sum to n , not just up to $n/2$.

Test Cases: 2/2 passed

Input	Expected Output	Actual Output
12	[(5, 7)]	[(5, 7)]
26	[(3, 23), (7, 19), (13, 13)]	[(3, 23), (7, 19), (13, 13)]

Alfred: AI Assistant

Hello! How can I assist you today?

What is Merge Sort?

Merge sort is a popular sorting algorithm that uses a divide-and-conquer approach to sort lists of elements. Here are the key points about merge sort:
It works by dividing the input list into two halves, sorting each half recursively, and then merging the two sorted halves.
The base case for the recursion is when the list has one or zero elements, in which case it is already sorted.
The merge step takes two sorted lists and combines them into a single sorted list.
Merge sort has a time complexity of $O(n \log n)$ and is suitable for large datasets.
You can learn more about merge sort on Google or check out the Merge Sort Wikipedia page for a detailed explanation.
Since the question is closely related to a programming concept, here's a one-line hint for a similar practice question: "Think about how recursive division and merging of sublists can be used to sort a list of elements."

Typ Submission successful
Your code has been submitted successfully.





User Dashboard

The screenshot shows the DeepSEEK User Dashboard. At the top, there's a navigation bar with links for 'About Us', 'Course', 'Dashboard', and 'Logout'. The main area displays the user's profile picture, name (ANUJ GUPTA), email (211300159@ds.study.iitm.ac.in), completed credits (96), and current level (DEGREE). Below this, there are two main sections: 'Completed Courses' and 'Asked Questions'. The 'Completed Courses' section lists 'Introduction to Python', 'Data Structures & Algorithms', and 'Machine Learning Theory'. The 'Asked Questions' section lists several questions related to algorithms and data structures, each with a timestamp (e.g., 2025-03-28) and a 'View' button. At the bottom, there's a footer with the text '© 2025 DeepSEEK. All rights reserved.'

5.3 Back End

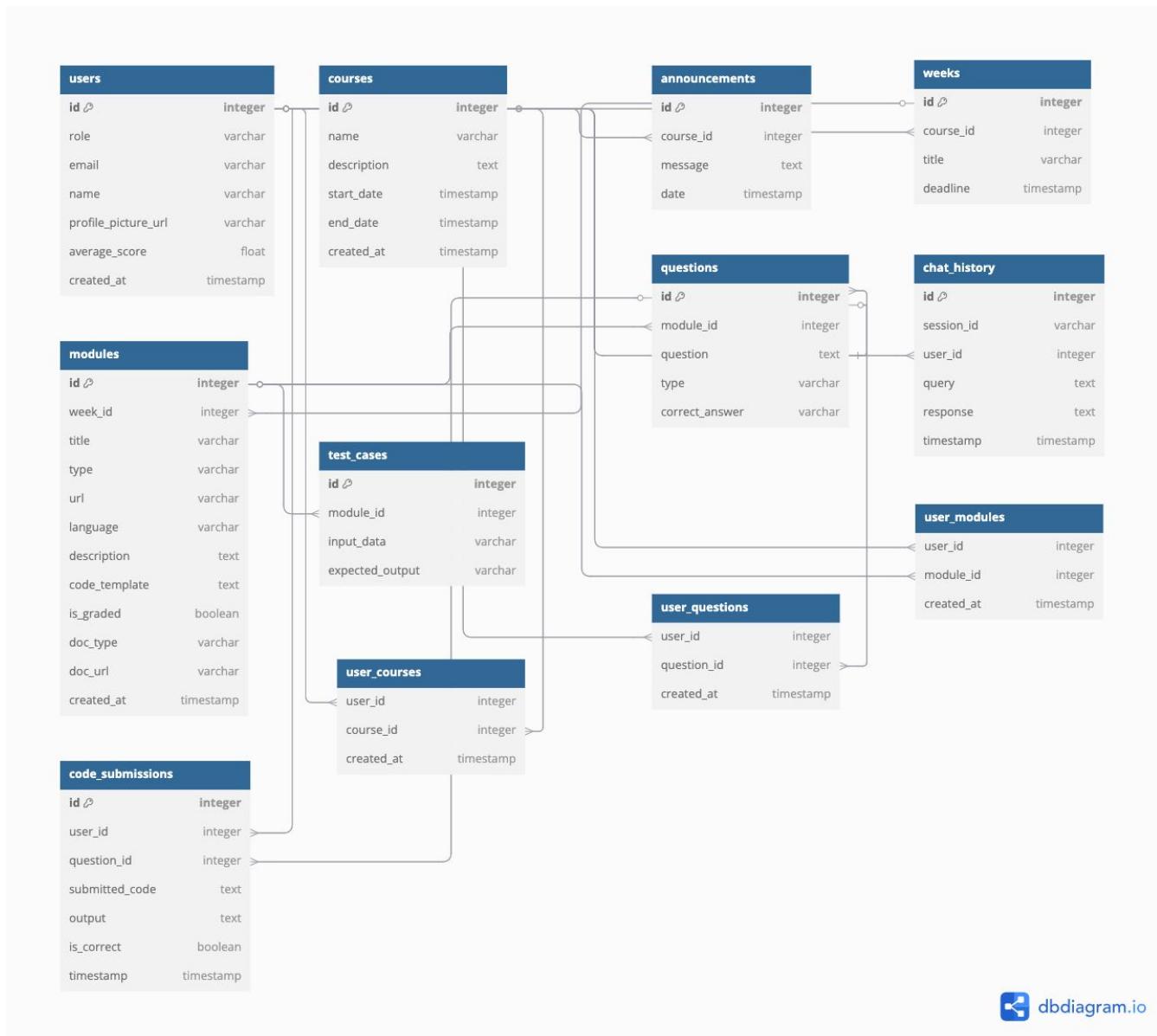
- The application utilizes the Flask micro webservices framework and Flask RESTful to connect the backend and frontend components.
- With MongoDB as the database the backend utilizes PyMongo and MongoEngine (ODM) to connect with the dB.
- RAG implementation utilizes FAISS (Vector database) and FastAPI for connectivity.
- The main components are,
 1. **Request Management:** Flask and Flask RESTful handles user actions such as watching videos, submitting responses and completing assignments.
 2. **Business Logic & Data Processing:** Flask based APIs and custom functions process user requests, validates data, and manages database operations with integrated logging for error tracking.
 3. **Security Enforcement:** Implements user authentication leveraging Google's single sign-on and Flask based RBAC to ensure secure access and reliable error handling.
 4. **Gen AI Integration:** Leverages LangChain framework and connects with open-source LLMs via APIs to support chatbot interactions.





5.4 ER diagram

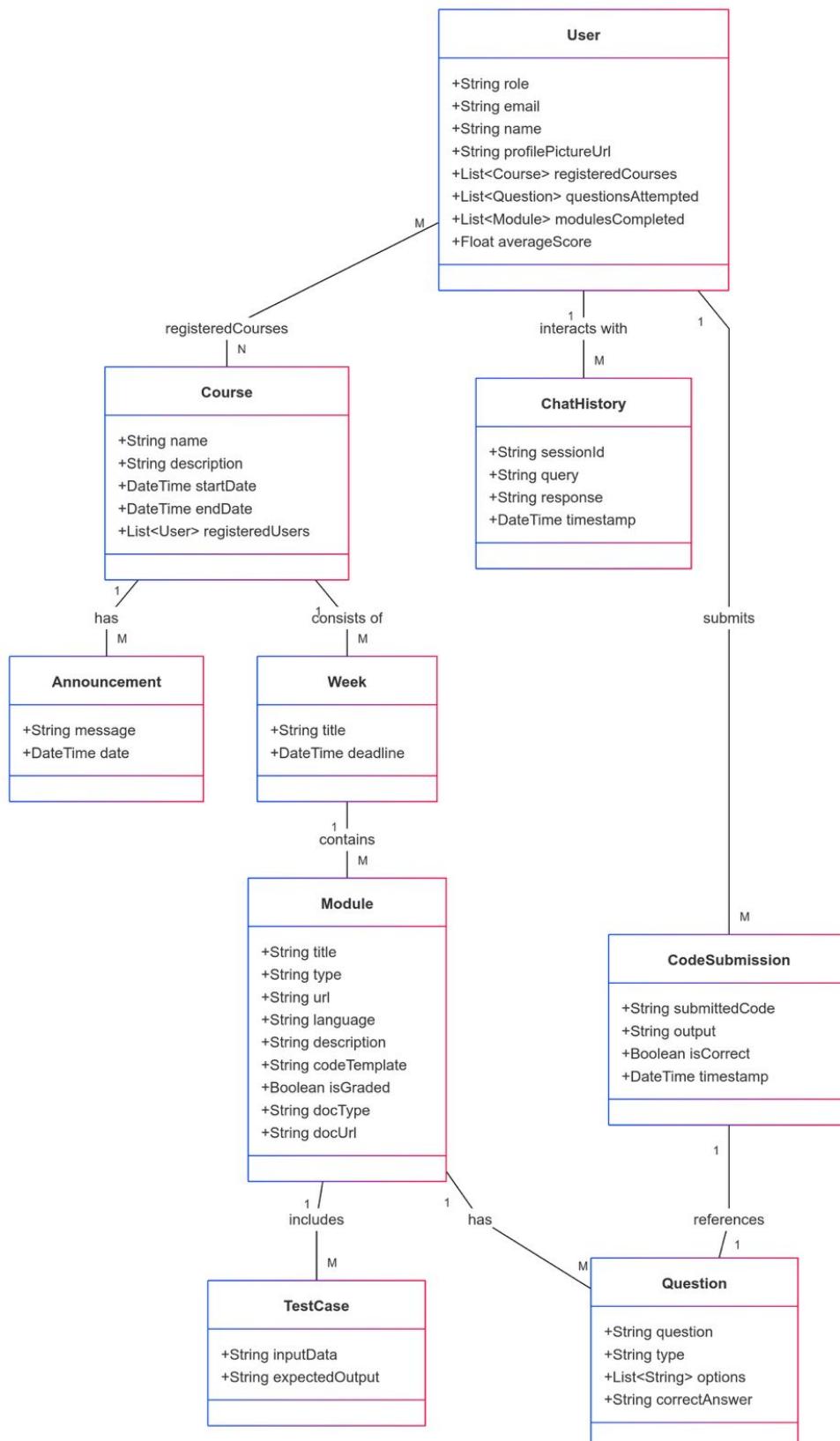
The below ER diagram shows entities and their relationships in the system. The database used for storing the tables and data for the application is MongoDB.





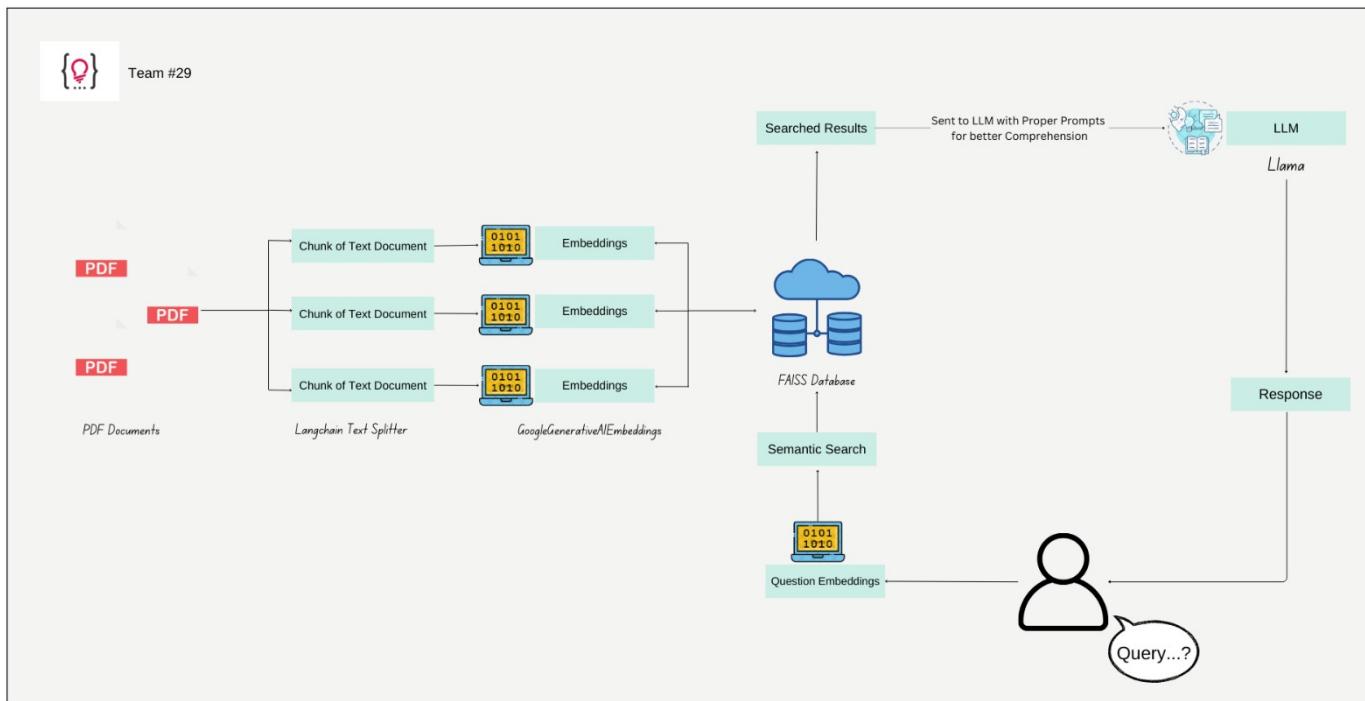
5.5 UML - Class diagram

Class diagram of the learning application depicts the various classes like Users, Courses, Lessons, Assignments, their attributes and methods and interactions.



5.6 RAG (Retrieval Augmented Generation) framework visualization

The below visual demonstrates how we plan to implement RAG workflow into the SEEK application using LangChain and how the various sub-systems interact to enable the AI chatbot.



Conclusion

By completing Milestone 3, the project team successfully established a well-structured project management framework and developed essential front-end components. The Jira-based sprint setup ensured effective tracking of tasks and backlog management. Additionally, the system architecture, including ER diagrams, class diagrams, and RAG workflows, was finalized, providing a robust foundation for subsequent API and AI development. The front-end UI was refined based on feedback, ensuring a user-friendly and intuitive design. With this milestone completed, the project is now ready for backend and API integration in the next phase.



Milestone 4





Objective

The objective of this milestone was to design and implement API endpoints for authentication, user management, course handling, AI-powered chatbot interactions, transcript retrieval, and code execution. The APIs were developed to facilitate seamless user onboarding, efficient course enrollment, AI-assisted learning support, and real-time analytics for administrators. Additionally, Swagger-compatible YAML documentation was created for the newly developed APIs, ensuring standardization and ease of integration.

API Endpoints

1. Authentication

POST /login

Handles user authentication by email and automatically creates accounts for new users. Returns user details including ID, role, name, email, profile picture, and last login timestamp. Validates required email field and handles server errors appropriately. Designed for seamless first-time user onboarding with minimal required fields.

2. Users

GET /users

Retrieves complete list of all registered users (admin/faculty only). Returns array with each user's ID, role, email, name, profile picture, and registered courses. Does not support pagination or filtering yet. Maintains consistent error response structure.

GET /user/{userId}

Fetches detailed profile for specific user after validating MongoDB ObjectId format. Returns 404 if user not found. Includes registered courses list and all basic user information in response.

DELETE /user/{userId}

Admin-only endpoint to permanently delete user accounts after ID validation. Returns a simple success message without confirmation step. Does not implement soft-delete functionality for recovery.

GET /user-statistics/{userId}

Provides learning analytics (questions attempted, modules completed, average scores) for individual users. Restricted to admin/faculty roles. Returns structured data for analytics dashboards.



**GET /dashboard/user/questions**

Organizes user's chatbot questions by course with timestamps. Sorts questions newest-first. Requires email parameters to identify users and fetch their question history.

3.Courses

GET /courses

Lists all available courses with basic info (ID, name, description, dates). Response optimized for course selection interfaces.

GET /course/{courseId}

Returns complete course structure with nested weekly content, announcements, and materials. Coding modules show test cases; assignments include questions with hints. Complex response supports rich navigation.

GET /registered-courses

Shows courses enrolled by specific user (email parameter). Returns simplified info (ID, name, description). Handles new users by returning empty array.

POST /registered-courses

Enrolls users in courses or creates new accounts. Accepts email + course ID array. Creates bidirectional user-course relationships automatically.

4.Transcripts

GET /video-transcript

Fetches YouTube captions using video URL parameter. Extracts video ID automatically and concatenates transcript chunks. Caches transcripts in the database to avoid repeated API calls.

5.AI Chatbot

POST /chatbot

AI course assistant processing natural language queries with conversation history. Logs questions by course/week for analytics. Integrates with external RAG API for responses.

POST /ask

Processes user queries by retrieving relevant document chunks from the FAISS database and combining them with chat history to generate a contextual response using an LLM. If no relevant documents are found, the LLM responds based on chat history alone.

POST /top-questions

Instructor-only AI analysis of frequent question topics. Processes all course questions to detect common themes. Returns top five formatted topics.



**POST /debug/code**

Provides AI-powered debugging via Groq API with strict prompt engineering. Returns concise two-line error explanations in consistent format.

POST /pdf

Retrieves a list of available PDFs from the document store. When a user clicks on the PDF icon, it generates and downloads the chat history as a PDF file for reference.

6. Code Execution

POST /run-code

Executes and validates code against module's test cases. Returns detailed input/output comparisons. Marks module complete if all tests pass.

POST /submit/code

Tests programming assignments with comprehensive feedback. Validates syntax before execution. Updates user progress upon successful completion.

7. Admin

GET /admin-statistics

System metrics for admins: user counts, content totals, aggregate activity. Shows sum of all questions attempted across platforms.

Authentication User login and session management

POST

/login User login



Users Manage user data and statistics

GET

/users Get all users



GET

/user/{userId} Get user details



DELETE

/user/{userId} Remove User Account



GET

/user-statistics/{userId} Get user statistics



GET

/dashboard/user/questions Get user's course-wise questions





Courses

Manage courses, modules, and announcements

GET	/courses	Get all courses	🔒	▼
GET	/course/{courseId}	Get course details	🔒	▼
GET	/registered-courses	Get registered courses	🔒	▼
POST	/registered-courses	Register courses	🔒	▼

Transcripts

Fetch and manage YouTube video transcripts

GET	/video-transcript	Fetch video transcript	🔒	▼
-----	-------------------	------------------------	---	---

Chatbot

Interact with the chatbot and manage chat history

POST	/chatbot	Chatbot interaction	🔒	▼
------	----------	---------------------	---	---

Code Execution

Execute code and test against test cases

POST	/run-code	Run code	🔒	▼
POST	/admin-statistics	Submit and test code	🔒	▼
POST	/debug/code	Debug code with AI assistance	🔒	▼

Admin

Admin-specific statistics and operations

GET	/admin-statistics	Get admin statistics	🔒	▼
POST	/top-questions	Get top questions by topic	🔒	▼

Conclusion

This milestone successfully delivered the required API endpoints for user authentication, course management, AI chatbot interactions, and code execution. Key functionalities, such as user authentication via google email, dynamic course enrollment, AI-driven question analysis, and automated transcript retrieval, have been implemented. Swagger documentation for the created APIs has been completed, ensuring API usability and integration readiness.





Milestone 5





Objective

The objective of this milestone was to design and execute a comprehensive test suite for all API endpoints to ensure robustness, reliability, and correctness of the system. Each API was tested using well-structured test cases, covering expected inputs, outputs, and edge cases. Automated tests were developed to validate API responses against expected results, ensuring that the authentication, user statistics, chatbot interactions, and other critical functionalities operate as intended.

1. Test Suite

This test suite is designed to verify the core functionalities of the system, including user authentication, course registration, user and admin statistics, and AI chatbot interactions. Each test module covers specific aspects of the application, ensuring reliability and consistency across various features.

Test Modules and Cases

Database Status

- **db_status:** Get DB connection status.

Login Tests (test_login.py)

- **1_student_login:** Verify login for student users.
- **2_admin_login:** Verify login for admin users.
- **3_faculty_login:** Verify login for faculty users.
- **4_email_required:** Ensure email is required for login.

Course Listing (test_courses.py)

- **list_courses:** Retrieve a list of all available courses.

Course Details (test_course.py)

- **course_details:** Fetch detailed information about a specific course using its ID.

Course Registration (test_register_courses.py)

- **1_register_course:** Allow a user to register for a course.
- **2_list_registered_courses:** Retrieve a list of registered courses for a user.
- **3_identify_user:** Identify a user during the registration process.
- **4_user_delete:** Delete a user record.
- **5_create_user:** Create a new user.
- **6_register_two_courses:** Register two courses for a user.
- **7_list_registered_courses:** Retrieve registered courses after multiple registrations.





- **8_register_invalid_course:** Attempt to register an invalid course.
- **9_list_registered_courses_without_email:** Try to list registered courses without providing an email.
- **10_list_registered_courses_invalid_user:** Attempt to list registered courses for an invalid user.
- **11_register_courses_empty_payload:** Handle registration with an empty request payload.

User Listing (test_users.py)

- **list_users:** Retrieve a list of all registered users in the system.

User Information (test_user.py)

- **1_login:** Authenticate a user.
- **2_user_details:** Fetch details of a specific user by their ID.
- **3_user_delete:** Delete a user record.
- **4_user_delete_user_not_found:** Handle cases where a non-existent user is deleted.
- **5_user_details_user_not_found:** Handle cases where details are requested for a non-existent user.

User Statistics (test_user_statistics.py)

- **1_login:** Authenticate a user before retrieving statistics.
- **2_user_statistics:** Retrieve statistics for a specific user.
- **3_user_delete:** Delete a user account.
- **4_user_statistics_user_not_found:** Handle cases where user statistics are requested for a non-existent user.

Video Transcript Retrieval (test_video_transcript.py)

- **1_transcript_video:** Retrieve a transcript for a lecture video using its URL.
- **2_transcript_invalid_video:** Handle cases where an invalid video URL is provided.
- **3_no_parameters:** Handle cases where no parameters are passed.

AI Chatbot Interaction (test_chatbot.py)

- **1_chatbot:** Verify chatbot query and response functionality.
- **2_incomplete_payload:** Handle cases where an incomplete payload is sent.
- **3_invalid_user:** Handle cases where an invalid user attempts to interact with the chatbot.

Admin Statistics (test_admin_statistics.py)

- **admin_statistics:** Retrieve administrative statistics, including user and system data management.





2. API and Individual Tests

Note: Only a few key test cases are shown here. Full report can be viewed [here](#).

Login

Test 1: Student Login

API URL: <https://api-deepseek.vercel.app/login>

API Method: POST

Input Data:

```
{  
    "email": "student_mail",  
    "name": "student_name",  
    "picture": "profile_picture"  
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Actual Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Python Code:

```
def test_1_student_login(student_mail,  
                        student_name,  
                        profile_picture,  
                        login_success_msg,
```





```
        helpers):
input_data = {
    "email": student_mail,
    "name": student_name,
    "picture": profile_picture
}
payload = json.dumps(input_data)

response = requests.post(API_LOGIN, data=payload, headers=headers)

assertEquals(response.status_code, 200)

assertEquals(response.headers["Content-Type"], "application/json")

data = response.json()

assertInstanceOf(data, dict)

required_keys = {"userId":str,
                 "name":str,
                 "email":str,
                 "role":str,
                 "message":str,
                 "picture":str}
verify_keys(required_keys, data)

assertEquals(data['email'], student_mail)

assertEquals(data['message'], login_success_msg)

assertEquals(data['role'], "student")
```

Test 2: Admin Login

API URL: <https://api-deepseek.vercel.app/login>

API Method: POST

Input Data:

```
{
    "email": "admin_mail",
    "name": "admin_name",
    "picture": "profile_picture"
}
```

Expected Status Code:

200

Actual Status Code:

200





Expected Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Actual Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Python Code:

```
def test_2_admin_login(admin_mail,
                      admin_name,
                      profile_picture,
                      login_success_msg):
    input_data = {
        "email": admin_mail,
        "name": admin_name,
        "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)
    assertEquals(response.status_code, 200)
    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"userId":str,
                     "name":str,
                     "email":str,
                     "role":str,
                     "message":str,
                     "picture":str}
    verify_keys(required_keys, data)

    assertEquals(data['email'], admin_mail)
    assertEquals(data['message'], login_success_msg)
```





```
assertEquals(data['role'], "admin")
```

Test 3: Faculty Login

API URL: <https://api-deepseek.vercel.app/login>

API Method: POST

Input Data:

```
{
  "email": "faculty_mail",
  "name": "faculty_name",
  "picture": "profile_picture"
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Actual Keys and Data Types in Response:

- userId : str
- name : str
- email : str
- role : str
- message : str
- picture : str

Python Code:

```
def test_3_faculty_login(faculty_mail,
                          faculty_name,
                          profile_picture,
                          login_success_msg):
    input_data = {
        "email": faculty_mail,
        "name": faculty_name,
```





```
"picture": profile_picture
}
payload = json.dumps(input_data)

response = requests.post(API_LOGIN, data=payload, headers=headers)

assertEquals(response.status_code, 200)

assertEquals(response.headers["Content-Type"], "application/json")

data = response.json()

assertInstanceOf(data, dict)

required_keys = {"userId":str,
                 "name":str,
                 "email":str,
                 "role":str,
                 "message":str,
                 "picture":str}
verify_keys(required_keys, data)

assertEquals(data['email'], faculty_mail)

assertEquals(data['message'], login_success_msg)

assertEquals(data['role'], "faculty")
```

Test 4: Email Required

API URL: <https://api-deepseek.vercel.app/login>

API Method: POST

Input Data:

```
{
  "email": "faculty_mail",
  "name": "faculty_name",
  "picture": "profile_picture"
}
```

Expected Status Code:

400

Actual Status Code:

400

Expected Error Message:

Email is required





Actual Error Message:

Email is required

Expected Keys and Data Types in Response:

- error: str

Actual Keys and Data Types in Response:

- error: str

Python Code:

```
def test_4_email_required(student_name,
                           profile_picture,
                           email_required_msg):
    input_data = {
        "email": "",
        "name": student_name,
        "picture": profile_picture
    }
    payload = json.dumps(input_data)

    response = requests.post(API_LOGIN, data=payload, headers=headers)
    assertEquals(response.status_code, 400)

    data = response.json()

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], email_required_msg)
```





AI Chatbot API Test Documentation

Test 1: Chatbot Interaction

API URL: <https://api-deepseek.vercel.app/chatbot>

API Method: POST

Input Data:

```
{  
    "query": "What is ML?",  
    "option": "option",  
    "sessionId": "S1234",  
    "userEmail": "student_mail"  
}
```

Expected Status Code:

200

Actual Status Code:

200

Expected Keys and Data Types in Response:

- sessionId : str
- question : str
- answer : str
- chatHistory : list

Actual Keys and Data Types in Response:

- sessionId : str
- question : str
- answer : str
- chatHistory : list

Additional Data Verification

Chat History Structure:

Each item in the chatHistory list must contain:

- query : str
- answer : str
- timestamp : str
- user : dict

User Structure:

Each user object in the chatHistory must contain:





- id:str
- name:str
- email:str
- role:str
- profilePictureUrl:str

Python Code:

```
def test_1_chatbot(student_mail,
                    student_id,
                    student_name,
                    student_pp):
    session_id = "S1234"
    query = "What is ML?"
    input_data = {
        "query": query,
        "option": "option",
        "sessionId": session_id,
        "userEmail": student_mail
    }
    payload = json.dumps(input_data)

    response = requests.post(API_CHATBOT, data=payload, headers=headers)

    assertEquals(response.status_code, 200)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"sessionId":str,
                     "question":str,
                     "answer":str,
                     "chatHistory": list
                    }
    verify_keys(required_keys, data)

    assertEquals(data['sessionId'], session_id)

    assertEquals(data['question'], query)

    chathistories = data["chatHistory"]
    for chathistory in chathistories:
        assertInstance(chathistory, dict)

        required_keys = {"query":str,
                        "answer":str,
                        "timestamp":str,
                        "user": dict
                       }
        verify_keys(required_keys, chathistory)
```





```
user = chathistory['user']
required_keys = {"id":str,
                 "name":str,
                 "email":str,
                 "role": str,
                 "profilePictureUrl": str
                }
verify_keys(required_keys, user)

assertEquals(user['name'], student_name)
assertEquals(user['email'], student_mail)
assertEquals(user['role'], "student")
assertEquals(user['profilePictureUrl'], student_pp)
```

Test 2: Incomplete Payload

API URL: <https://api-deepseek.vercel.app/chatbot>

API Method: POST

Input Data:

```
{
  "option": "option",
  "sessionId": "S1234",
  "userEmail": "student_mail"
}
```

Expected Status Code:

400

Actual Status Code:

400

Expected Error Message:

Query and option are required

Actual Error Message:

Query and option are required

Expected Keys and Data Types in Response:

- error: str

Actual Keys and Data Types in Response:

- error: str





Python Code:

```
def test_2_incomplete_payload(student_mail,
                               query_option_required_msg):
    input_data = {
        "option": "option",
        "sessionId": "S1234",
        "userEmail": student_mail
    }
    payload = json.dumps(input_data)

    response = requests.post(API_CHATBOT, data=payload, headers=headers)

    assertEquals(response.status_code, 400)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], query_option_required_msg)
```

Test 3: Invalid User

API URL: <https://api-deepseek.vercel.app/chatbot>

API Method: POST

Input Data:

```
{  
    "query": "What is ML?",  
    "option": "option",  
    "sessionId": "S1234",  
    "userEmail": "invalid_student_mail"  
}
```

Expected Status Code:

404

Actual Status Code:

404

Expected Error Message:

User not found





Actual Error Message:

User not found

Expected Keys and Data Types in Response:

- error: str

Actual Keys and Data Types in Response:

- error: str

Python Code:

```
def test_3_invalid_user(invalid_student_mail,
                        user_not_found_msg):
    input_data = {
        "query": "What is ML?",
        "option": "option",
        "sessionId": "S1234",
        "userEmail": invalid_student_mail
    }
    payload = json.dumps(input_data)

    response = requests.post(API_CHATBOT, data=payload, headers=headers)
    assertEquals(response.status_code, 404)

    assertEquals(response.headers["Content-Type"], "application/json")

    data = response.json()

    assertInstance(data, dict)

    required_keys = {"error":str}
    verify_keys(required_keys, data)

    assertEquals(data['error'], user_not_found_msg)
```

Conclusion

This milestone successfully developed and executed a test suite to validate the key API functionalities, ensuring the robustness of user authentication, chatbot interactions, and user statistics retrieval. The structured test cases and automation scripts provide a strong foundation for ongoing validation and maintenance. With all tests passing successfully, the milestone objectives have been fully accomplished.





Milestone 6





Objective

The objective of this milestone is to bring together all the individual components of the project into a cohesive and fully functional application, ready for deployment. This includes the seamless integration of the Frontend interface, Backend services, Retrieval-Augmented Generation (RAG) system, and the Large Language Model (LLM), which is connected through APIs. The focus is on ensuring that these components communicate effectively and function as a unified system.

In addition to the technical integration, the milestone report also provides an overview of the various software technologies, libraries, frameworks, and tools that were utilized throughout the project lifecycle. This includes tools for version control, code reviews, issue tracking, and collaborative development, all of which played a critical role in ensuring smooth functioning.

1. Technologies and tools used

The below table summarizes the libraries, frameworks and tools used for executing the project and the application area for each of the tools.



Library / Framework / Tool/ Model	Application area
Vercel, Render	Hosting, CI
React JS, Router, TypeScript, TailwindCSS, ShadCN, Vite	Application Frontend
Python	Core programming
Flask	Application Backend
MongoDB	Databasing
PyMongo and MongoEngine (ODM)	Database Connectivity
FAISS Vector Database plus FastAPIs	RAG Database and Connectivity
Flask RESTful	Application Interface
Google SSO (Google OAuth)	Authentication, Security, Access
Swagger API Editor	API Documentation
Langchain	AI Chatbot
Llama (llama-3.3-70b-versatile)	Language Model
Postman	Manual Testing
Pytest	Automated Testing
GitHub for Code review , Issue reporting and tracking	Versioning and Development
Jira	Productivity
Canva, draw.io	UML



2. Instructions to run the application

Getting Started

Clone the Repository

```
git clone https://github.com/21f3002975/seek-portal-ai-agent.git  
cd seek-portal-ai-agent
```

Backend Setup (Flask)

Set Up a Virtual Environment

```
cd backend  
python3 -m venv venv  
source venv/bin/activate # MacOS/Linux  
venv\Scripts\activate # Windows  
pip install -r requirements.txt
```

Configure Environment Variables

Create a `.env` file and add:

```
MONGO_URI=  
RAG_API=  
GROQ_API_KEY=
```

Run the Backend Server

```
python -m api.app
```

API is live at: `http://localhost:5000/`





Frontend Setup (React + Vite)

Install Dependencies

```
cd frontend  
npm install
```

Configure Environment Variables

Create a `.env` file in the root directory and add:

```
VITE_API_URL=  
VITE_GOOGLE_CLIENT_ID=
```

Start the Frontend Server

```
npm run dev
```

App is live at: <http://localhost:3000/>

RAG & AI Setup (FastAPI + LangChain)

Set Up Virtual Environment

```
cd rag  
python -m venv venv  
source venv/bin/activate # MacOS/Linux  
venv\Scripts\activate # Windows
```

Install Dependencies

```
pip install -r requirements.txt
```

Configure Environment Variables

Create a `.env` file and add:

```
GROQ_API_KEY=your_groq_api_key  
GOOGLE_API_KEY=your_google_api_key
```

Run the RAG & AI Server

```
uvicorn app:app --reload --host 0.0.0.0 --port 8000
```

API is live at: <http://localhost:8000/>





3. Code development, review, issues tracking and deployment

- We have leveraged GitHub for versioning, code management and issue tracking.
 - Additionally, we have implemented continuous integration using Git and Vercel. The updated repo from Git is fetched by Vercel and the application is deployed if it is a successful build.
 - WhatsApp group chat has also been leveraged for updating each other and to conduct reviews.

3.1 Commits, builds and deployment

Code Issues Pull requests Actions Projects Security Insights Settings

Commits

main All users All time

- Commits on Mar 28, 2025
 - Integration & Testing done**
anujgupta95 committed 32 minutes ago
a55635c
- Commits on Mar 15, 2025
 - .md files**
ananddotiyer committed 2 weeks ago
09ac74c
- Commits on Mar 14, 2025
 - Reordered APIs in the documentation and added description**
ananddotiyer committed 2 weeks ago
2b99482
- Commits on Mar 12, 2025
 - Update README.md**
ananddotiyer committed 2 weeks ago
8c6efab
 - Documentation**
ananddotiyer committed 2 weeks ago
874d50f

The screenshot shows the Azure portal interface with the following details:

- Deployment Summary**: Status: **Success**, Last updated: 10 minutes ago. It lists deployment steps:
 - Imported logic app definition
 - Deployed logic app definition
 - Deployed trigger function
 - Deployed action function
- Logs**: Shows deployment logs with entries like "Importing logic app definition" and "Deploying logic app definition".
- Metric**: Shows CPU usage over time.
- Static Assets**: Lists assigned custom domains: `depossees-testapp123.com` (Custom Domain) and `depossees-testapp123.onion` (Custom Domain).





3.2 Issues tracking

The screenshot shows a Jira search results page with the query "is:issue state:open". There are 9 open issues listed:

- Previously submitted answers to quiz are lost, when the user moves to another page. #9 · 21f1001185 opened last month
- In quiz pages, check the answers only when 'Check answers' button is clicked. #8 · 21f1001185 opened last month
- Uncluttering the Chat window. #7 · 21f1001185 opened last month
- Registering for more courses #6 · 21f1001185 opened last month
- Wrong popup regarding course registration #5 · 21f1001185 opened last month
- 'Course data' for completed courses #4 · 21f1001185 opened last month
- Need to implement RBAC #3 · 21f1001185 opened last month

Previously submitted answers to quiz are lost, when the user moves to another page. #9

Closed

21f1001185 opened last month

Bring up the previously submitted answers for the quizzes (Practice and Graded), when the quiz page is brought up again after navigating elsewhere in the application.

Steps to repro:

1. Open the 'Graded quiz' page.
2. Select answers for some Graded questions and submit.
3. Move to another page.
4. Reopen the 'Graded quiz' page. The previously submitted answers are lost.

Create sub-issue

21f1001185 assigned anujgupta95 last month

21f3002975 closed this as completed 1 minute ago





3.3 Reviews and Updates

Guys our viva slot is on 9th April, 6:20-6:35pm

Anand Iyer
Have been talking with Anuj and Ajay separately, but there are few things that I don't have clarity on still. I'm not filing defects on these, since the existing ones are lying unattended for the last 10 days in github 😊

- When trying to test run/submit the PDSA Week-1 graded coding assignment (Goldbach's Conjecture...) for the course, the space is pre-filled with some code. Initially, it appeared to be template code, but it's not. The pre-filled code includes the 'Goldbach' function itself. Who in the team is working on this area? Please look into this.
- Need to work on creating the APIs for 'Test run' and 'Submit' buttons on this screen. @Anuj Gupta , @Ajay Thiagarajan , not sure if you want to meet up and discuss on this. 'Debug' button also doesn't appear to do anything.
- 'Show hint' button doesn't appear to give any hints, and Anuj confirmed to me it indeed fetches hint from the db; it appears that there're no hints available on the db. @Ajay Thiagarajan I'm assuming you can take care of this.
- The chatbot API hasn't been integrated with RAG yet. As of now, the API response is returning some static string for the 'answer'. However, the front-end chat seems to be working. Anuj clarified on this and said @Niraj Kumar and @Jalaj Trivedi needs to work on this. I'm not entirely clear what the work is, but please meet up and attend to this.

NOTE: This list is in addition to those already filed in GitHub.

Edited 11:37 AM

Anand Iyer
Have been talking with Anuj and Ajay separately, but there are few things that I don't have clarity on still. I'm not filing defects on these, since the existing ones are lying unattended for the last 10 days in github 😊

@Anuj Gupta @Ajay Thiagarajan am guessing you guys will take care of it

1:53PM ✓

Pending tasks

- 1) Jira updation - Vasu
- 2) RAG restrictions - Jalaj/Niraj
- 3) RAG follow up query implementation - Jalaj/ Niraj
- 4) Sign up FE - Anuj
- 5) FE testing inputs to be clarified - Anuj
- 6) Overall app beautification - Vasu/ Anuj

Anything else?

10:31PM ✓

@Niraj Kumar @Jalaj Trivedi can you set up the call with Bhargava for tomorrow 11.30pm? Do it asap so that he knows its in his calendar

10:32 PM ✓

You
@Niraj Kumar @Jalaj Trivedi can you set up the call with Bhargava for tomorrow 11.30pm? Do it asap so that he knows its in his calendar

looks like it is already done, thanks

10:34 PM ✓

Anand Iyer
• Work on remaining APIs. Ajay to consult with Anuj.
• Anand to finalize pytests on APIs. 70% done now.

10:35 PM

90% coverage will be very good and we have to make sure that the main behaviours are tested well.

10:38 PM ✓

Conclusion

This milestone successfully integrated and deployed the final application, DeepSEEK. The seamless combination of the frontend, backend, RAG pipeline, and LLM through API integration marks a significant achievement in the development cycle. This milestone lays a strong foundation for future enhancements and scaling of DeepSEEK.



Thank you for reading and reviewing !



For questions and suggestions :
21f3002975@ds.study.iitm.ac.in

