BS in Data Science and Applications

Software Engineering

Milestone 6

(Final Submission)

Topic **Gen AI Integration with Seek portal.**

Team 30

1	Anjali Panchal	21F2000411
2	Sahil Sandhu	21F1002317
3	Abdul Ahad Rauf	21F3002590
4	Sanket Metrewar	21F3000961
5	Aman Verma	21F1006376
6	Neon Thapa	21F1000706
7	Pratham Sharma	21F1006197
8	Leo Tom	21F1005835

Table of Contents

Problem Statement

Integrating Generative AI in seek portal.

- **Overview**:
  The rapid advancement of technology has revolutionized the educational landscape, providing new opportunities for personalized learning and interactive content delivery. Generative AI (GenAI) offers the potential to transform these platforms by creating more engaging, customized, and efficient learning environments. We aim to integrate GenAI into seek portal to enhance the user experience, improve learning outcomes and streamline content creation. As a part of this project, we will be adding following functionalities:

  1. **Summary Generation**:
     GenAI can automatically generate concise summaries of lengthy lectures, making it easier for students to review and grasp key concepts quickly at the time of revision. This feature can be tailored to highlight essential points, critical discussions, and important takeaways, ensuring that students focus on the most relevant information. Additionally, AI-generate summaries can be personalized based on individual learning preferences and past interactions with the content.

  2. **Providing external resources**:
     Integrating Generative AI (GenAI) to provide external resources for content on an educational portal can greatly enrich the learning experience. GenAI can curate and recommend a wide array of supplementary materials, enhancing the core curriculum and supporting diverse learning needs.

  3. **Custom prompts**:
     Custom prompts are invaluable for addressing immediate doubts, offering tailored explanations that align with specific needs. They also facilitate connections with existing knowledge, making learning more intuitive and relatable. By integrating new information with familiar concepts, custom prompts enhance understanding, promote active engagement, and support a more personalized and effective learning experience.

Learner Journey Map

	Features	Without GenAI (Currently)	With GenAI	Possible Advantages
C H A T B O T	**Summary Generation**	Sometimes the lectures are a bit too long and we have limited time to go through them.	Generating summary for long lectures will save time during quick revisions.	• Saves time and effort in reviewing content. • Provides concise and relevant information. • Enhances learning efficiency.
	Providing external resources	Finding relevant additional resources for some topics that are not easy to understand manually can be time-consuming.	Recommending and providing links to external articles, papers, and tutorials relevant to the topic will save student's time.	• Saves students' time by quickly providing relevant materials. • Ensures access to high-quality external resources. • Enhances learning with a broader perspective.
	Custom prompts	Sometimes we have numerous questions about the topics being taught in lectures but lack the opportunity to get them answered in real time.	Provides immediate clarification on questions and actively engages students to keep them involved in the learning process.	• Instant clarification helps grasp concepts more effectively. • Keeps students attentive and actively involved. • Enhanced learning retention ensures better recall.

User Stories

As a student

1. Accessing Learning Material:
    - I want to receive concise summaries of learning videos and reading materials.
    - So that I can quickly grasp the key concepts and save time on revision.
2. Utilizing and inbuilt chatbot:
    - I want to interact with an inbuilt chatbot for quick assistance on course related queries.
    - So that I can get instant help without waiting for human intervention, enhancing my learning experience.

As a Faculty Member

1. Monitoring student progress:
    - I want to track student engagement with GenAI features (like summaries and chatbot usage),
    - So that I can identify areas where students are struggling and provide targeted support.
2. Enhancing course material:
    - I want to update and refine course content based on the insights provided by GenAI analytics,
    - So that the learning materials are always relevant and address common student issues effectively.

As a Support Staff

1. Addressing Technical Issues:
    - I want to receive notifications of any technical issues faced by students while using GenAI features,
    - So that I can resolve them promptly and ensure a smooth learning experience for students.
2. Assisting with GenAI Features:
    - I want to provide guidance and troubleshooting for students on using GenAI features,
    - So that they can effectively utilize these tools to enhance their learning process.

As an Admin

1. Managing User Roles:
    - I want to assign and manage user roles and permissions within the GenAI-enhanced portal,
    - So that access to different features and data is controlled and secure.
2. Ensuring Data Privacy:
    - I want to implement strict data privacy measures for all interactions with GenAI tools,
    - So that student data is protected and complies with relevant regulations.
3. Analysing GenAI Impact:
    - I want to analyse the impact of GenAI integrations on student performance and engagement,
    - So that I can make informed decisions on further enhancements and resource allocation.

Acceptance Criteria

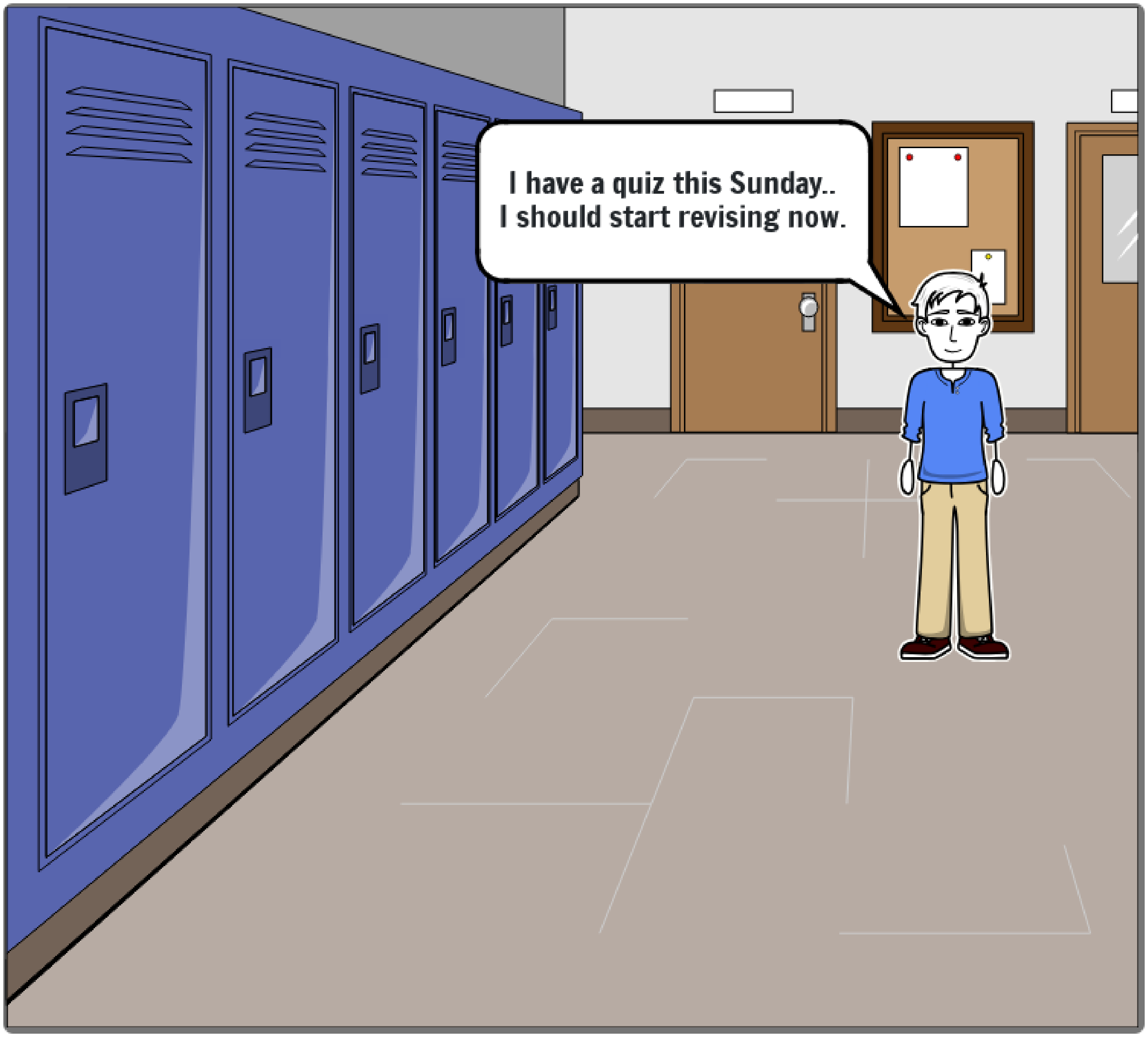
1. **Accessing Learning Material**
   - GenAI generates summaries for all learning videos and reading materials.
   - Summaries are concise, covering the key points of the content.
   - Summaries are available immediately after accessing the material.
   - Students can benefit by knowing what's being taught.
2. **Utilizing an Inbuilt Chatbot**
   - Inbuilt chatbot is available 24/7 within the SEEK portal.
   - Chatbot can answer a wide range of course-related query accurately.
   - Students can interact with the chatbot through a user-friendly interface.

Story Boards

Storyboards are vital tools in website development, serving as a visual narrative that outlines the user journey and key interactions within a site. They depict a sequence of frames, similar to a comic strip, illustrating how users will navigate through different pages and features. Each frame captures specific moments, such as landing on the homepage, accessing a menu, filling out a form, or completing a purchase.

By using storyboards, designers and developers can plan and communicate the user experience in a clear and structured manner. They help identify potential usability issues, ensure a logical flow of information, and maintain consistency across the site.

Integrating GenAI in Seek Portal



Integrating GenAI in Seek Portal

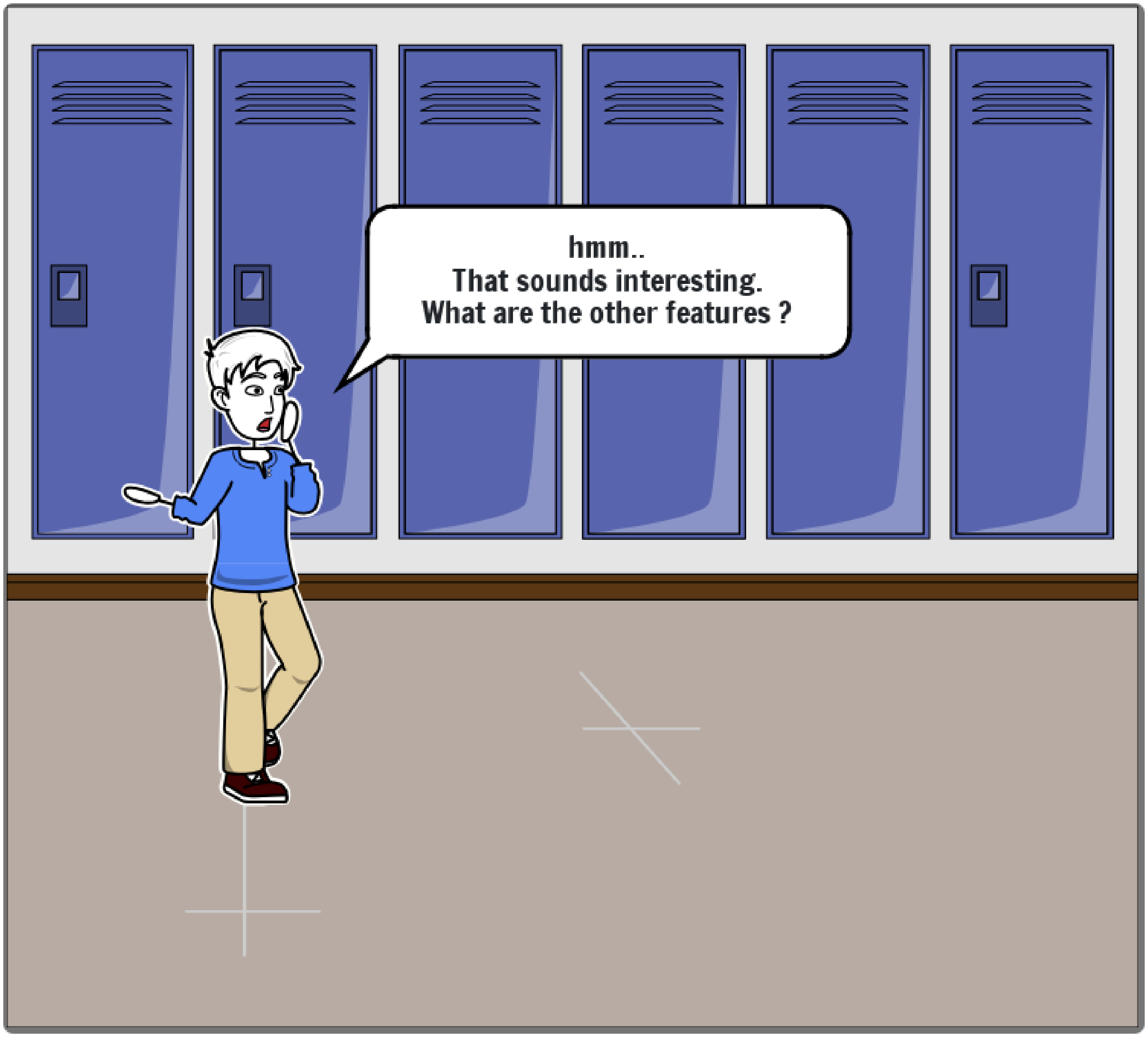
Integrating GenAI in Seek Portal

Hey miss Topper..
Can I borrow your notes for quick revision... please...

Integrating GenAI in Seek Portal

Integrating GenAI in Seek Portal

They have introduced some new GenAI features in portal.. Summary is one of them. You can just ask chat bot to generate a summary for you..


hmm..
That sounds interesting.
What are the other features ?

Also.. you can get external resource for the topics you are currently working on..
and you can get your data generated synthetically so that you can gain comprehensive understanding of models and algos..

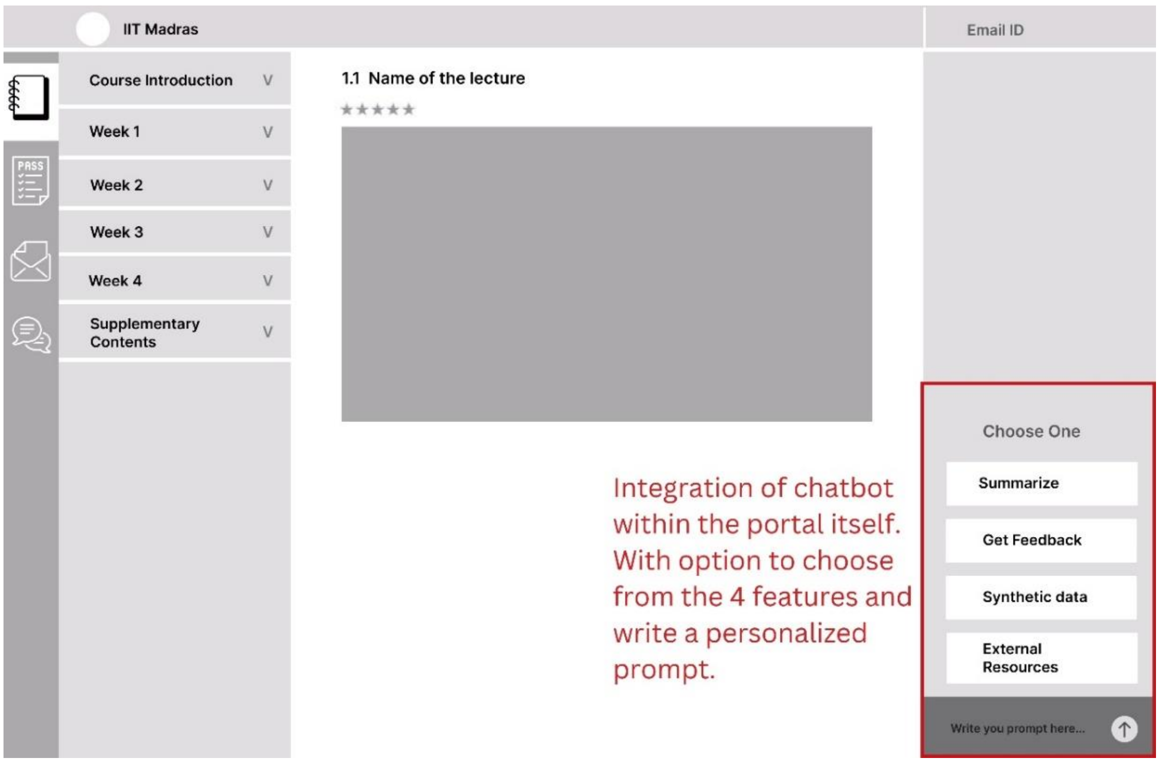
Thanks for letting me know all of these features.. I'll go home and try these for sure.

Wireframes

Wireframes provide a visual guide to the layout of a webpage, outlining where elements like headers, navigation menus, content sections, and interactive features will be placed. By focusing on the basic structure without delving into design details, wireframes help designers, developers, and stakeholders align on the page's functionality and flow. Below is the basic low fidelity prepared for this project:

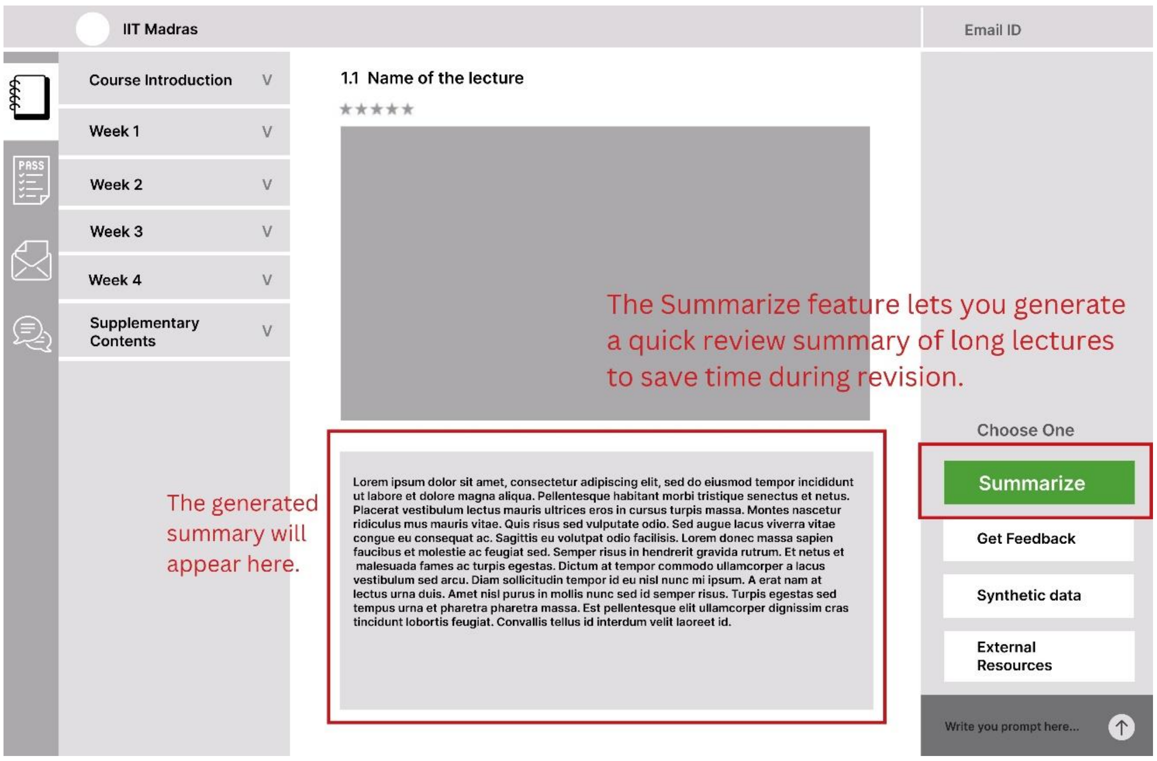
1. **Main content screen**: Chatbot Sidebar
   - The main content screen includes a dedicated chatbot sidebar, offering a comprehensive set of features.
2. **Summarise screen**: Summarization Feature
   - The summarize screen is designed to provide users with a streamlined functionality to condense lengthy content.
3. **External resources screen**: Fetching External Resources.
4. **Custom chat screen**: Getting responses for custom prompts.

The external resources screen is designed to fetch and display data from various external sources.



Integration of chatbot within the portal itself. With option to choose from the 4 features and write a personalized prompt.

Figure 1: Main content screen.

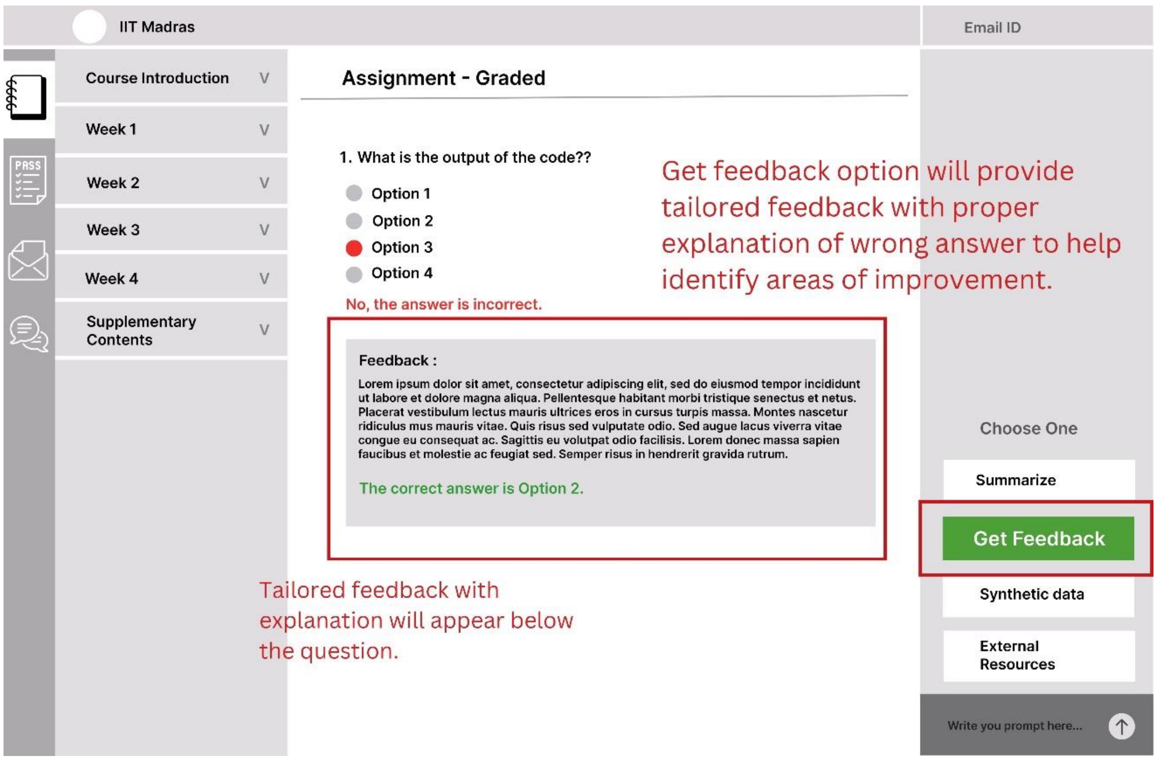


The Summarize feature lets you generate a quick review summary of long lectures to save time during revision.

The generated summary will appear here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Pellentesque habitant morbi tristique senectus et netus. Placerat vestibulum lectus mauris ultrices eros in cursus turpis massa. Montes nascetur ridiculus mus mauris vitae. Quis risus sed vulputate odio. Sed augue lacus viverra vitae congue eu consequat ac. Sagittis eu volutpat odio facilisis. Lorem donec massa sapien faucibus et molestie ac feugiat sed. Semper risus in hendrerit gravida rutrum. Et netus et malesuada fames ac turpis egestas. Dictum at tempor commodo ullamcorper a lacus vestibulum sed arcu. Diam sollicitudin tempor id eu nisl nunc mi ipsum. A erat nam at lectus urna duis. Amet nisl purus in mollis nunc sed id semper risus. Turpis egestas sed tempus urna et pharetra pharetra massa. Est pellentesque elit ullamcorper dignissim cras tincidunt lobortis feugiat. Convallis tellus id interdum velit laoreet id.

Figure 2: Get summary.



Assignment - Graded

1. What is the output of the code??

- Option 1
- Option 2
- ● Option 3
- Option 4

No, the answer is incorrect.

Feedback :

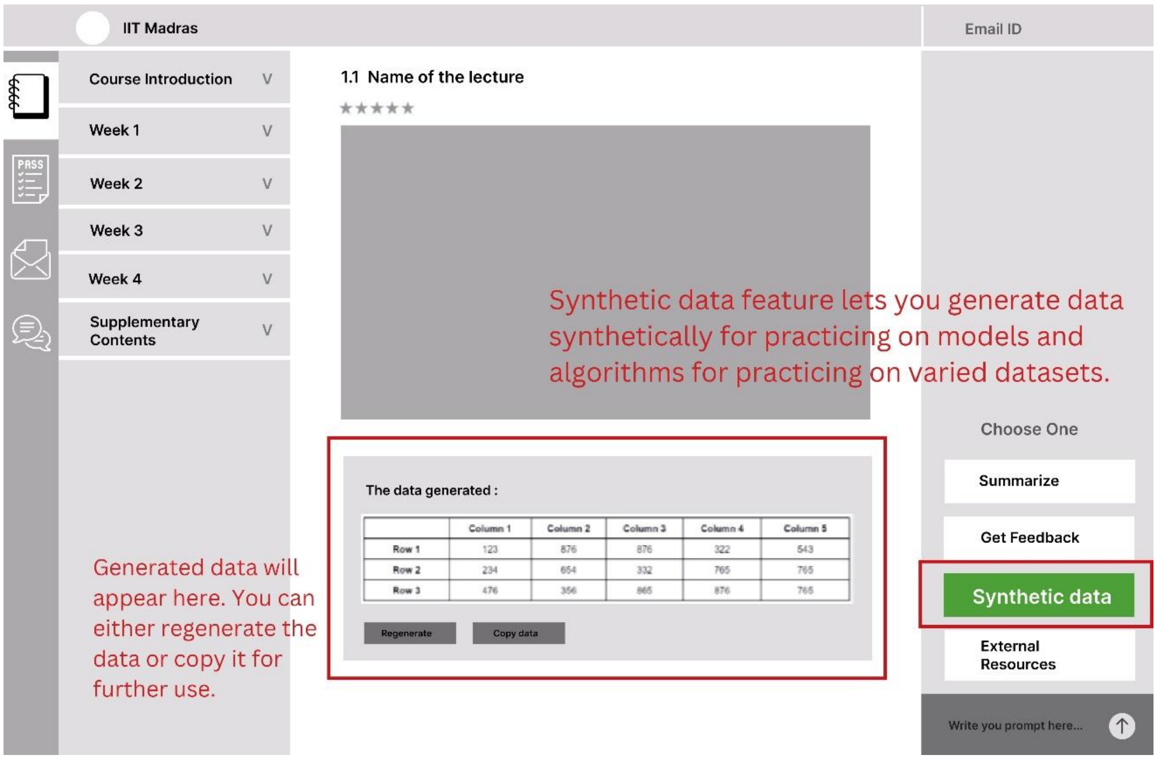
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Pellentesque habitant morbi tristique senectus et netus. Placerat vestibulum lectus mauris ultrices eros in cursus turpis massa. Montes nascetur ridiculus mus mauris vitae. Quis risus sed vulputate odio. Sed augue lacus viverra vitae congue eu consequat ac. Sagittis eu volutpat odio facilisis. Lorem donec massa sapien faucibus et molestie ac feugiat sed. Semper risus in hendrerit gravida rutrum.

The correct answer is Option 2.

Get feedback option will provide tailored feedback with proper explanation of wrong answer to help identify areas of improvement.

Tailored feedback with explanation will appear below the question.

Figure 3: Getting feedback.



1.1 Name of the lecture

★★★★★

Synthetic data feature lets you generate data synthetically for practicing on models and algorithms for practicing on varied datasets.

The data generated :

	Column 1	Column 2	Column 3	Column 4	Column 5
Row 1	123	876	876	322	543
Row 2	234	654	332	765	765
Row 3	476	356	865	876	765

Generated data will appear here. You can either regenerate the data or copy it for further use.

Figure 4: Get synthetic data.

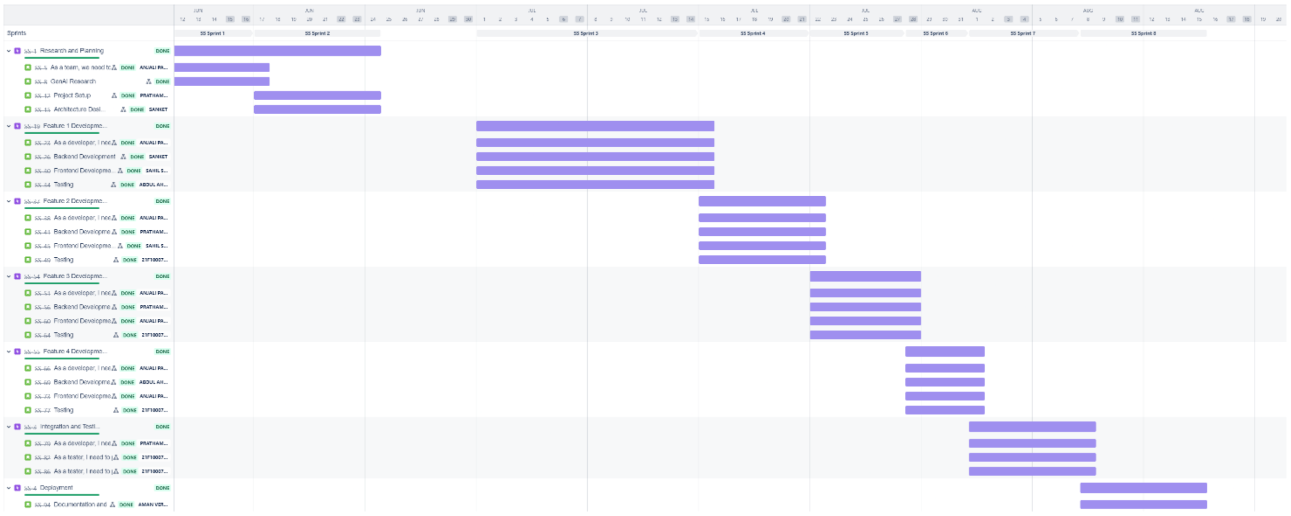
Project Schedule

Sprints Schedule

- **SS Sprint 1:** Brainstorm and Research.
  - **Date:** 10 Jun 2024 – 24 Jun 2024.
- **SS Sprint 2:** Architecture design for integrating GenAI, backend development, frontend development, testing.
  - **Date:** 1 Jul 2024 – 15 Jul 2024.
- **SS Sprint 3:** Architecture design, backend development, frontend development, testing.
  - **Date:** 15 Jul 2024 – 22 Jul 2024.
- **SS Sprint 4:** Architecture design, backend development, frontend development, testing.
  - **Date:** 22 Jul 2024 – 1 Aug 2024.
- **SS Sprint 5:** Integrate GenAI models with existing website, unit testing, integration testing, acceptance testing.
  - **Date:** 1 Aug 2024 – 8 Aug 2024.
- **SS Sprint 6:** Documentation and deployment.
  - **Date:** 8 Aug 2024 – 15 Aug 2024.

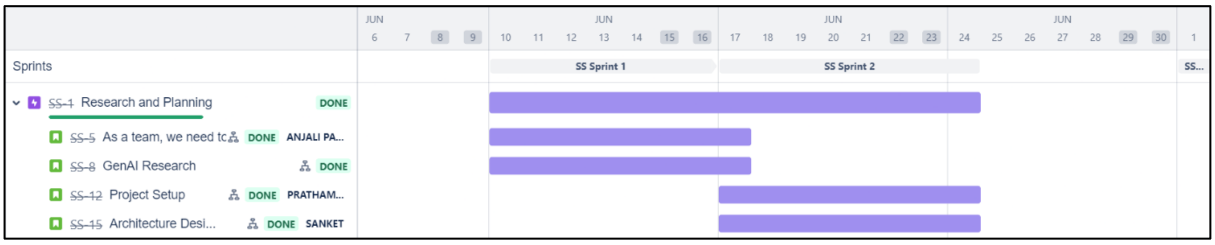
Project Scheduling Tools: JIRA

We use JIRA for project management, task tracking, and collaboration. Its user-friendly interface and customizable features make it perfect for agile development. JIRA's flexibility and robust tools ensure efficient workflow and seamless team communication, enhancing our project's productivity and overall success.



Sprints		
~~SS-1~~ Research and Planning	DONE	
~~SS-5~~ As a team, we need to…	DONE	ANJALI PA…
~~SS-8~~ GenAI Research	DONE	
~~SS-12~~ Project Setup	DONE	PRATHAM…
~~SS-15~~ Architecture Desi…	DONE	SANKET
~~SS-19~~ Feature 1 Developme…	DONE	
~~SS-22~~ As a developer, I nee…	DONE	ANJALI PA…
~~SS-26~~ Backend Development	DONE	SANKET
~~SS-30~~ Frontend Developme…	DONE	SAHIL S…
~~SS-34~~ Testing	DONE	ABDUL AH…
~~SS-37~~ Feature 2 Developme…	DONE	
~~SS-38~~ As a developer, I nee…	DONE	ANJALI PA…
~~SS-41~~ Backend Developme…	DONE	PRATHAM…
~~SS-45~~ Frontend Developme…	DONE	SAHIL S…
~~SS-49~~ Testing	DONE	21F10007…
~~SS-54~~ Feature 3 Developme…	DONE	
~~SS-51~~ As a developer, I nee…	DONE	ANJALI PA…
~~SS-56~~ Backend Developme…	DONE	PRATHAM…
~~SS-60~~ Frontend Developme…	DONE	ANJALI PA…
~~SS-64~~ Testing	DONE	21F10007…
~~SS-65~~ Feature 4 Developme…	DONE	
~~SS-66~~ As a developer, I nee…	DONE	ANJALI PA…
~~SS-69~~ Backend Developme…	DONE	ABDUL AH…
~~SS-72~~ Frontend Developme…	DONE	ANJALI PA…
~~SS-77~~ Testing	DONE	21F10007…
~~SS-3~~ Integration and Testi…	DONE	
~~SS-79~~ As a developer, I nee…	DONE	PRATHAM…
~~SS-83~~ As a tester, I need to…	DONE	21F10007…
~~SS-86~~ As a tester, I need to…	DONE	21F10007…
~~SS-4~~ Deployment	DONE	
~~SS-94~~ Documentation and…	DONE	AMAN VER…

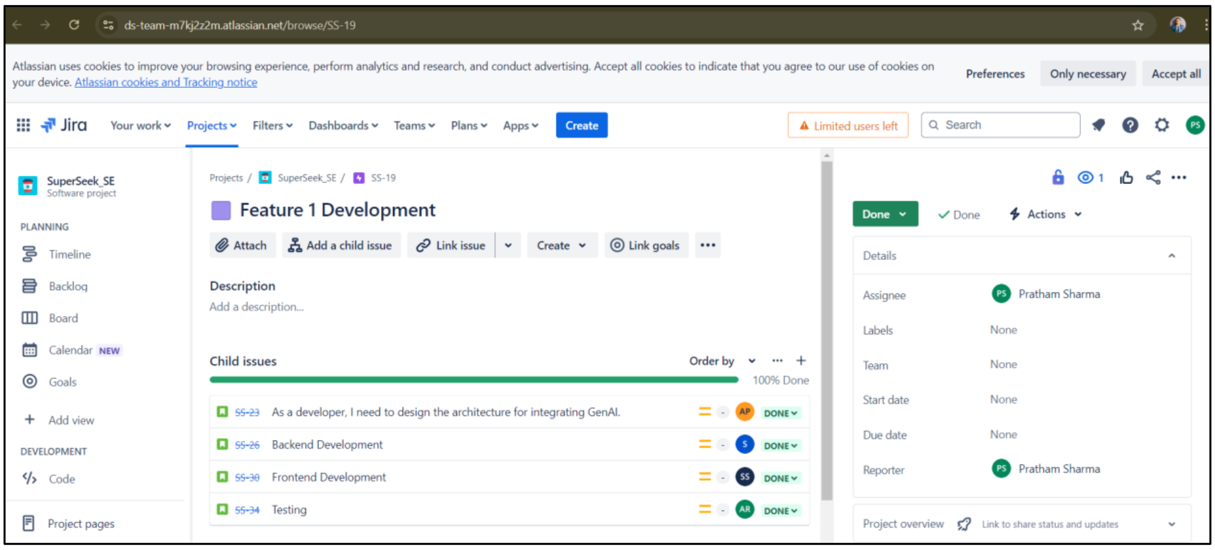
Sprint 1: Research and Planning



Sprints — JUN 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 — SS Sprint 1 — SS Sprint 2 — SS...

- ~~SS-1~~ Research and Planning DONE
  - ~~SS-5~~ As a team, we need tc DONE ANJALI PA...
  - ~~SS-8~~ GenAI Research DONE
  - ~~SS-12~~ Project Setup DONE PRATHAM...
  - ~~SS-15~~ Architecture Desi... DONE SANKET

Sprint 3: Feature 1 Development



ds-team-m7kj2z2m.atlassian.net/browse/SS-19

Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. Atlassian cookies and Tracking notice — Preferences — Only necessary — Accept all

Jira — Your work — Projects — Filters — Dashboards — Teams — Plans — Apps — Create — Limited users left — Search

SuperSeek_SE Software project — PLANNING: Timeline, Backlog, Board, Calendar NEW, Goals, Add view — DEVELOPMENT: Code — Project pages

Projects / SuperSeek_SE / SS-19

Feature 1 Development

Attach — Add a child issue — Link issue — Create — Link goals

Description
Add a description...

Child issues — Order by — 100% Done

SS-23	As a developer, I need to design the architecture for integrating GenAI.	AP	DONE
SS-26	Backend Development	S	DONE
SS-30	Frontend Development	SS	DONE
SS-34	Testing	AR	DONE

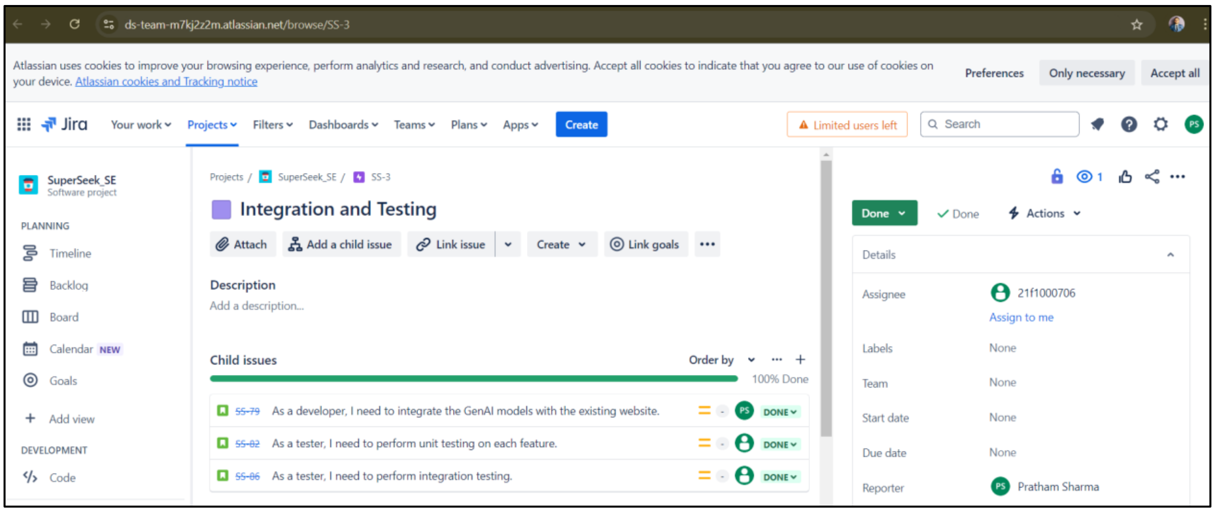
Done — Done — Actions

Details

Assignee	Pratham Sharma
Labels	None
Team	None
Start date	None
Due date	None
Reporter	Pratham Sharma

Project overview — Link to share status and updates

Sprint 7: Integration and Testing



ds-team-m7kj2z2m.atlassian.net/browse/SS-3

Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. Atlassian cookies and Tracking notice — Preferences — Only necessary — Accept all

Jira — Your work — Projects — Filters — Dashboards — Teams — Plans — Apps — Create — Limited users left — Search

SuperSeek_SE Software project — PLANNING: Timeline, Backlog, Board, Calendar NEW, Goals, Add view — DEVELOPMENT: Code

Projects / SuperSeek_SE / SS-3

Integration and Testing

Attach — Add a child issue — Link issue — Create — Link goals

Description
Add a description...

Child issues — Order by — 100% Done

SS-79	As a developer, I need to integrate the GenAI models with the existing website.	PS	DONE
SS-82	As a tester, I need to perform unit testing on each feature.		DONE
SS-86	As a tester, I need to perform integration testing.		DONE

Done — Done — Actions

Details

Assignee	21f1000706 Assign to me
Labels	None
Team	None
Start date	None
Due date	None
Reporter	Pratham Sharma

Task Assignment

S. No.	Milestone	Sub-Tasks	Sprint	Assigned To
1	**Identify User Requirements**	Brainstorming & Identify Users	**1**	All
		User Stories		Abdul
		Learner Journey Map		Anjali
		Report		Sahil
2	**User Interfaces**	Story Board	**2**	Sahil
		Wireframes		Anjali
		Report		Sahil
3	**Scheduling and Design**	Project Scheduling	**3**	Anjali & Pratham
		Components Design		Abdul & Neon
		Software Design		Aman
		Minutes of Meetings		Leo
		Report		Sahil
4	**API Endpoints**	API Design	**4**	Sanket
		API Description		Pratham & Abdul
		YAML File		Sanket
5	**Testing**	Test Case Design	**5**	Pratham, Abdul & Anjali
		Unit Testing		Aman, Neon & Leo
		Report		Sahil
6	**Final Submission**	Frontend Design	**6**	Sahil & Anjali
		Backend Implementation		Abdul, Aman & Pratham
		Report		Sahil
		Presentation		All

Design of Components

Backend components

To effectively integrate Generative AI into the SEEK our team discussed the necessary backend components that we will require for smooth functioning:

1. **User Component**
    - The User Component manages user authentication and stores user information, including attributes such as userId, name, and email. It provides methods for users to log in and access their profiles.
2. **Course Component**
    - The Course Component handles course details and content management. It includes attributes such as courseId, course Name, description, and content, and provides methods for viewing course content.
3. **Video Component**
    - The Video Component manages video playback and controls for course materials. It includes attributes like title and URL, and methods to play, pause, and resume videos.
4. **Feedback Component**
    - The Feedback Component provides immediate feedback on quiz answers through the method generate Feedback, which takes a student's answer and returns a string with the feedback.
5. **Summary Component**
    - The Summary Component generates concise summaries of learning materials using the method generate Summary, which takes the content as input and returns a summary.
6. **LinkFetcher Component**
    - The Link Fetcher Component retrieves relevant links for additional resources. It includes the method fetchLinks, which takes content as input and returns a list of links.
7. **SyntheticDataGenerator Component**
    - The SyntheticDataGenerator Component creates synthetic data sets for practice exercises in data science courses. It uses the method generateData, which takes parameters and returns a DataFrame.
8. **Code Compiler Component**
    - The Code Compiler Component compiles and executes code, providing a real-time coding environment. It includes methods such as compileCode to compile code and runCode to execute compiled code with given input, returning the execution result.
9. **Chatbot Component**
    - The Chatbot Component interacts with students to provide instant help and resolve queries. It includes the method, which takes a query as input and returns a response.

Frontend Components

The frontend of the Seek Portal with integrated GenAI features consists of a user interface designed to enhance the learning experience. Here's an overview of the key components:

1. **Main Content Screen with Chatbot Sidebar:**

   - Users can access course content while interacting with an AI-powered chatbot sidebar. This sidebar offers a comprehensive set of features, allowing students to choose from various AI functionalities and input personalized prompts.

2. **Summarize Screen:**

   - Users can generate quick summaries of long lectures or course materials. The system provides concise overviews of key points, aiding in revision and comprehension.

3. **Feedback Screen:**

   - Users receive tailored feedback on their assignments, with explanations for incorrect answers. This feature helps identify areas for improvement and enhances the learning process.

4. **Synthetic Data Screen:**

   - Users can generate artificial datasets for practicing with models and algorithms. This feature allows for hands-on experience with varied data, enhancing understanding of data analysis concepts.

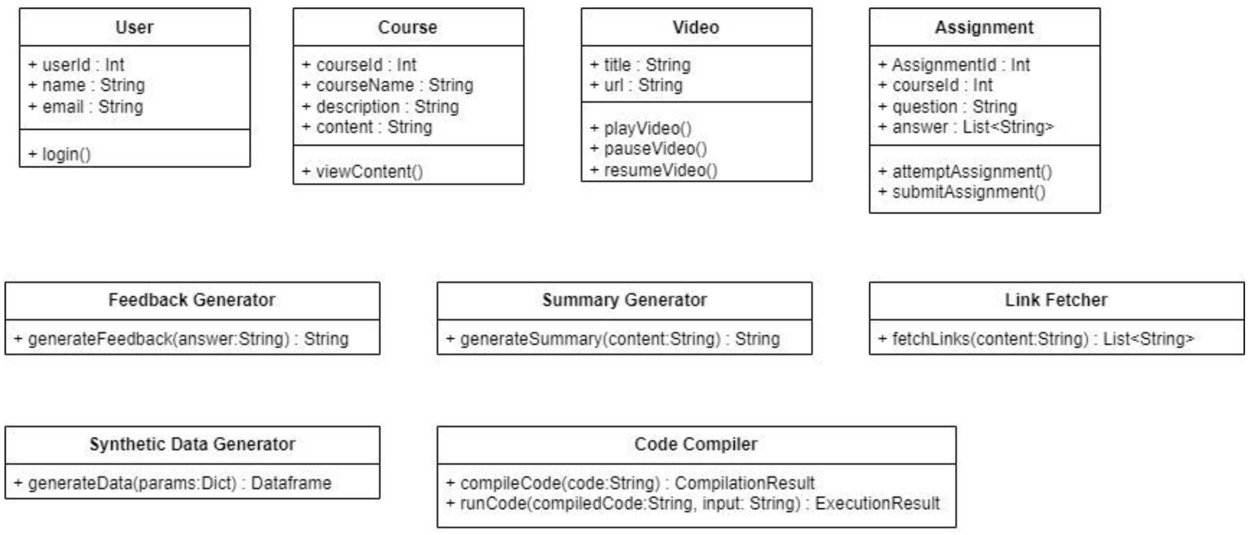
5. **External Resources Screen:**

   - Users can access a curated list of external resources including articles, online courses, and video tutorials related to their current topics of study. This feature broadens the learning scope beyond course materials.

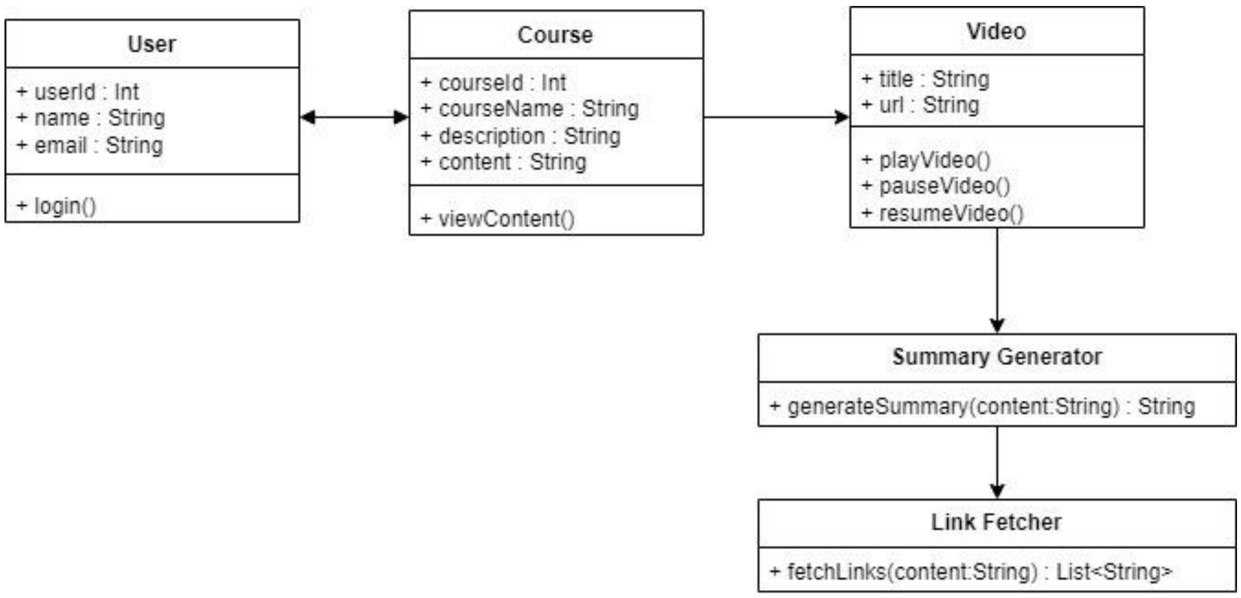
Software Design

The UML class diagram offers a visual overview of the nine components and their interconnections within the Super Seek project. It depicts the system's structure, highlighting the various user types—students, support staff, admins, and IT staff—as well as the primary entity, GenAI. Additionally, it showcases the interactions between these components, including summary generation, link fetching, and feedback generation. This class diagram serves as a blueprint for the system architecture, facilitating the understanding and communication of the Super Seek system's design.

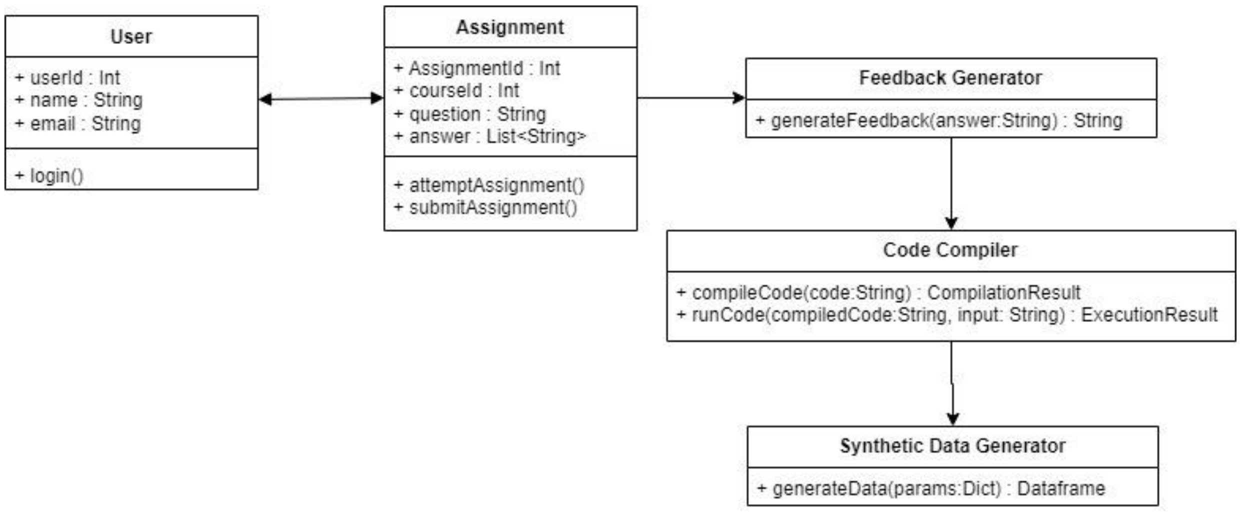
Components



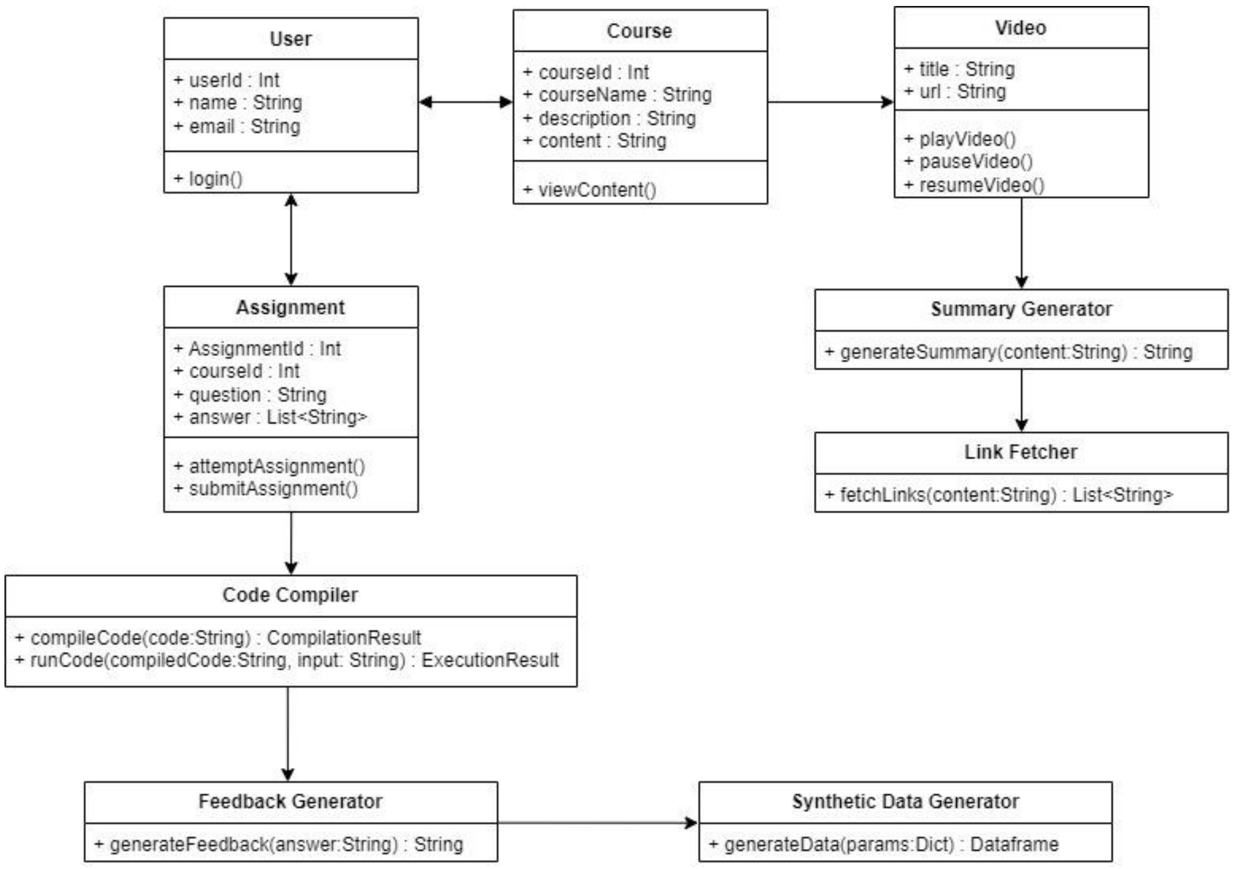
Course Usage



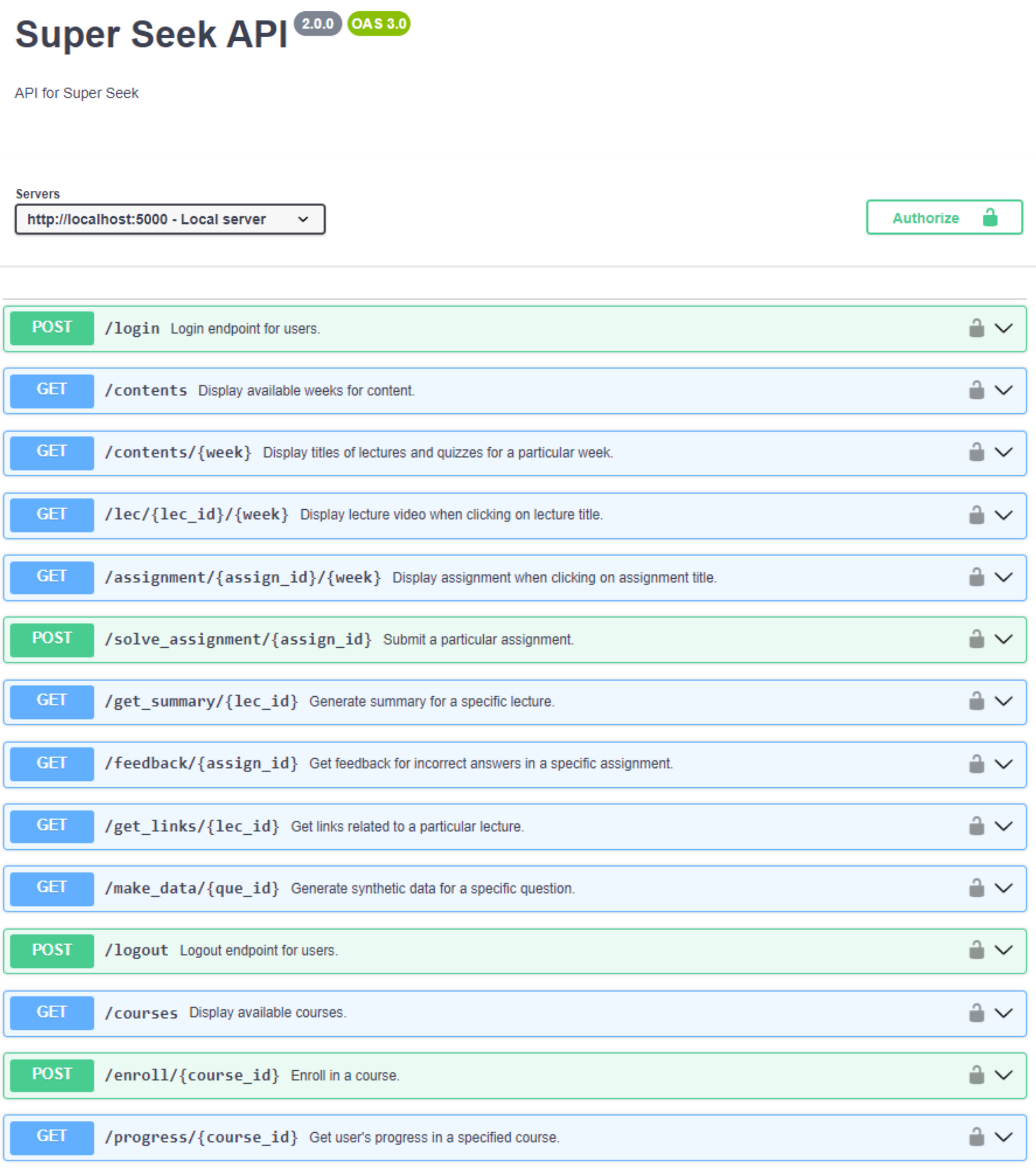
Assignment Usage



Final Compilation



API Endpoints



Super Seek API 2.0.0 OAS 3.0

API for Super Seek

Servers
http://localhost:5000 - Local server

Authorize

```
POST  /login  Login endpoint for users.
GET   /contents  Display available weeks for content.
GET   /contents/{week}  Display titles of lectures and quizzes for a particular week.
GET   /lec/{lec_id}/{week}  Display lecture video when clicking on lecture title.
GET   /assignment/{assign_id}/{week}  Display assignment when clicking on assignment title.
POST  /solve_assignment/{assign_id}  Submit a particular assignment.
GET   /get_summary/{lec_id}  Generate summary for a specific lecture.
GET   /feedback/{assign_id}  Get feedback for incorrect answers in a specific assignment.
GET   /get_links/{lec_id}  Get links related to a particular lecture.
GET   /make_data/{que_id}  Generate synthetic data for a specific question.
POST  /logout  Logout endpoint for users.
GET   /courses  Display available courses.
POST  /enroll/{course_id}  Enroll in a course.
GET   /progress/{course_id}  Get user's progress in a specified course.
```

Technologies used

Frontend:

Category	Library/Tool	Version
Framework/Library	Vue	^3.2.13
	Vue Router	^4.0.3
Styling	@fortawesome/fontawesome-free	^6.6.0
Utility	Axios	^1.7.4
	Core-js	^3.8.3
Development Tools	@babel/core	^7.12.16
	@babel/eslint-parser	^7.12.16
	@vue/cli-plugin-babel	~5.0.0
	@vue/cli-plugin-eslint	~5.0.0
	@vue/cli-plugin-router	~5.0.0
	@vue/cli-service	~5.0.0
	ESLint	^7.32.0
	eslint-plugin-vue	^8.0.3

Backend:

Category	Library/Tool	Version
Frameworks	Flask	3.0.0
	Flask-RESTful	0.3.10
	Flask-SQLAlchemy	3.1.1
Database	SQLAlchemy	2.0.25
	Flask-Migrate	4.0.5
	alembic	1.13.1
Task Queue	Celery	5.3.6
	kombu	5.3.5
	redis	5.0.1
	billiard	4.2.0
	vine	5.1.0
HTTP Libraries	httpcore	1.0.5
	httplib2	0.22.0
	httpx	0.27.0
Utility	amqp	5.2.0
	anyio	4.4.0
	aniso8601	9.0.1
	blinker	1.7.0
	cachelib	0.9.0
	certifi	2024.7.4
	checksumdir	1.2.0
	click	8.1.7
	click-didyoumean	0.3.0
	click-plugins	1.1.1

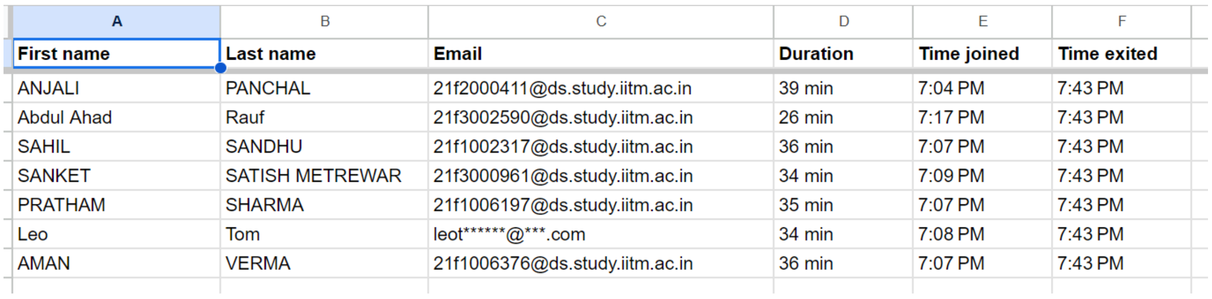
	click-repl	0.3.0
	colorama	0.4.6
	Flask-Caching	2.1.0
	Flask-Login	0.6.3
	greenlet	3.0.3
	h11	0.14.0
	idna	3.7
	itsdangerous	2.1.2
	Jinja2	3.1.3
	Mako	1.3.0
	MarkupSafe	2.1.3
	prompt-toolkit	3.0.43
	pyparsing	3.1.1
	python-dateutil	2.8.2
	pytz	2023.3.post1
	setuptools	69.0.3
	six	1.16.0
	sniffio	1.3.1
	tzdata	2023.4
	Werkzeug	3.0.1
	wcwidth	0.2.13

Details/Minutes of scrum meetings

Sprint 1 (Intro and Ice Breaking)

Date	Jun 10Th 2024
Time	7:00 p.m.
Meeting called to order by	Anjali Panchal

- **In Attendance:**



First name	Last name	Email	Duration	Time joined	Time exited
ANJALI	PANCHAL	21f2000411@ds.study.iitm.ac.in	39 min	7:04 PM	7:43 PM
Abdul Ahad	Rauf	21f3002590@ds.study.iitm.ac.in	26 min	7:17 PM	7:43 PM
SAHIL	SANDHU	21f1002317@ds.study.iitm.ac.in	36 min	7:07 PM	7:43 PM
SANKET	SATISH METREWAR	21f3000961@ds.study.iitm.ac.in	34 min	7:09 PM	7:43 PM
PRATHAM	SHARMA	21f1006197@ds.study.iitm.ac.in	35 min	7:07 PM	7:43 PM
Leo	Tom	leot******@***.com	34 min	7:08 PM	7:43 PM
AMAN	VERMA	21f1006376@ds.study.iitm.ac.in	36 min	7:07 PM	7:43 PM

- **Team:**
  - Each member introduced themselves, providing a brief background including their name, major/department, interests, and any previous project experience.
  - Anjali Panchal was nominated to be the team leader

- **Setting up communication channels:**
  - Agreed on using WhatsApp for regular communication and to use G-Meets for regular team meetings

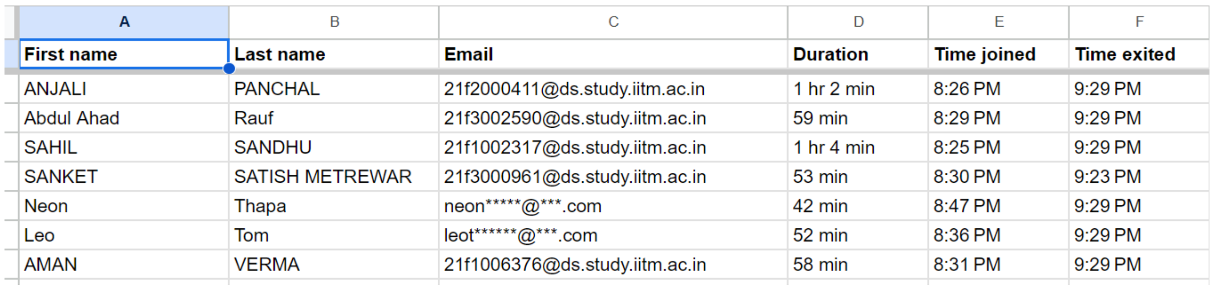
- **Scheduling regular meetings:**
  - Regular meetings are scheduled on G-Meet and the next meeting will be decided during each meeting
  - Any immediate meeting will be called through the WhatsApp group
  - Agreed to review and adjust the schedule, if necessary, based on members' availability.

- **Next meeting:**
  - https://meet.google.com/rxb-hbuw-wpi
  - Jun 21st 2024

Sprint 2 (Brainstorming the project)

Date	Jun 21st 2024
Time	8:30 p.m.
Meeting called to order by	Anjali Panchal

- **In Attendance:**



A	B	C	D	E	F
First name	**Last name**	**Email**	**Duration**	**Time joined**	**Time exited**
ANJALI	PANCHAL	21f2000411@ds.study.iitm.ac.in	1 hr 2 min	8:26 PM	9:29 PM
Abdul Ahad	Rauf	21f3002590@ds.study.iitm.ac.in	59 min	8:29 PM	9:29 PM
SAHIL	SANDHU	21f1002317@ds.study.iitm.ac.in	1 hr 4 min	8:25 PM	9:29 PM
SANKET	SATISH METREWAR	21f3000961@ds.study.iitm.ac.in	53 min	8:30 PM	9:23 PM
Neon	Thapa	neon*****@***.com	42 min	8:47 PM	9:29 PM
Leo	Tom	leot******@***.com	52 min	8:36 PM	9:29 PM
AMAN	VERMA	21f1006376@ds.study.iitm.ac.in	58 min	8:31 PM	9:29 PM

- **Approval of minutes:**
  - The minutes from the previous meeting were read and approved
- **Project:**
  - A list of ideas that could be implemented were drawn and members discussed the feasibility of each idea. Approval of features to be implemented will be taken up in the next meeting.

- **Sub teams:**
  - Team members shared their strengths and it was decided there would be sub-teams focusing on backend, frontend and documentation.
  - Members all chose roles they could contribute to (Scrum Master, Tester, Developer etc.)

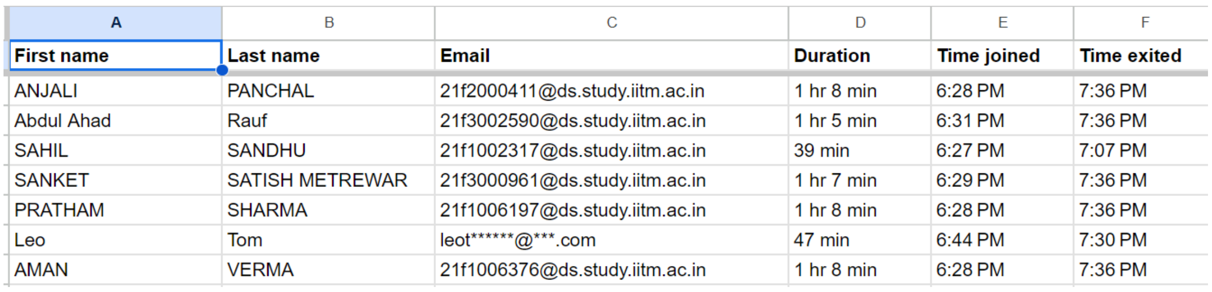
- **Tasks:**
  - Members were informed to fill and upload the honor code for milestone 1

- **Next meeting:**
  - https://meet.google.com/rxb-hbuw-wpi
  - Jun 24th 2024

Sprint 3 (Finalising the user requirements)

Date	Jun 24th 2024
Time	6:30 p.m.
Meeting called to order by	Anjali Panchal

- **In Attendance:**



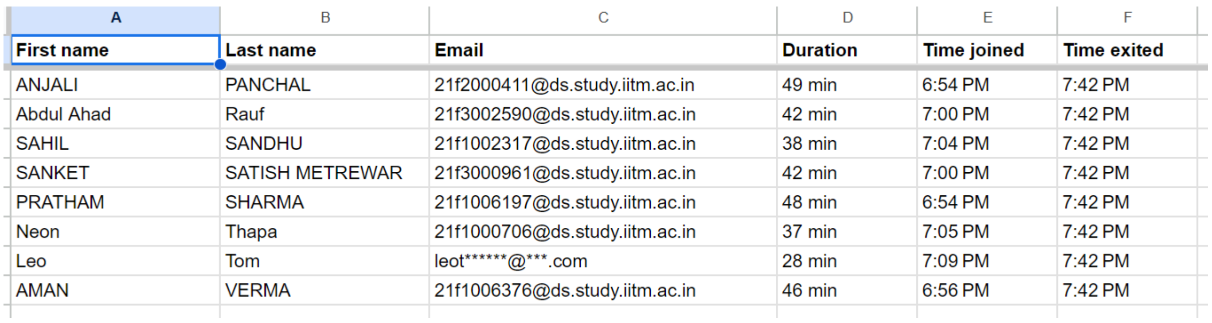
First name	Last name	Email	Duration	Time joined	Time exited
ANJALI	PANCHAL	21f2000411@ds.study.iitm.ac.in	1 hr 8 min	6:28 PM	7:36 PM
Abdul Ahad	Rauf	21f3002590@ds.study.iitm.ac.in	1 hr 5 min	6:31 PM	7:36 PM
SAHIL	SANDHU	21f1002317@ds.study.iitm.ac.in	39 min	6:27 PM	7:07 PM
SANKET	SATISH METREWAR	21f3000961@ds.study.iitm.ac.in	1 hr 7 min	6:29 PM	7:36 PM
PRATHAM	SHARMA	21f1006197@ds.study.iitm.ac.in	1 hr 8 min	6:28 PM	7:36 PM
Leo	Tom	leot******@***.com	47 min	6:44 PM	7:30 PM
AMAN	VERMA	21f1006376@ds.study.iitm.ac.in	1 hr 8 min	6:28 PM	7:36 PM

- **Approval of minutes:**
  - The minutes from the previous meeting were read and approved
- **Project:**
  - The feasibility of seven ideas were discussed and the it was decided to proceed with three of them
    - Summary Generation
    - Getting Answer Feedback
    - Synthetic Data Generation
  - Story Board and User Stories will be created by respective teams for Milestone 2 based on above features
- **GitHub and drive:**
  - GitHub repository and Drive folder will be created to organize and store project artefacts and shared
  - Members are to upload Honor Code, User Stories and Story Boards
  - Jira will be used to track task completion
- **Next meeting:**
  - https://meet.google.com/rxb-hbuw-wpi
  - Jul 2nd 2024

Sprint 4 (Discussion & division of further task)

Date	Jul 2nd 2024
Time	7:00 p.m.
Meeting called to order by	Anjali Panchal

- **In Attendance**:



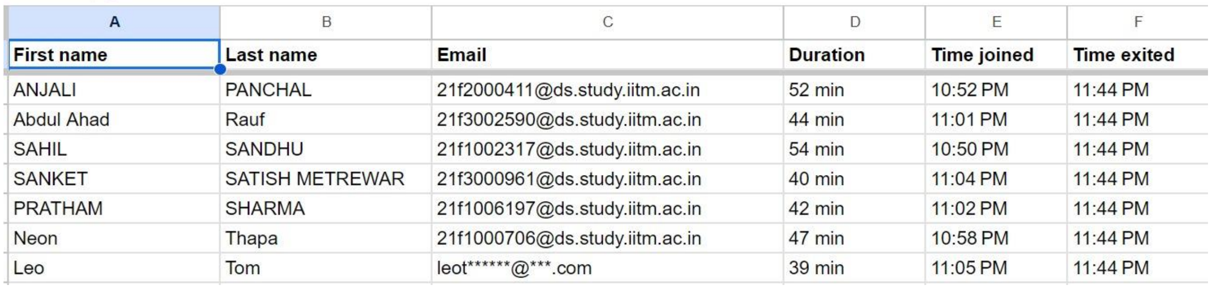
First name	Last name	Email	Duration	Time joined	Time exited
ANJALI	PANCHAL	21f2000411@ds.study.iitm.ac.in	49 min	6:54 PM	7:42 PM
Abdul Ahad	Rauf	21f3002590@ds.study.iitm.ac.in	42 min	7:00 PM	7:42 PM
SAHIL	SANDHU	21f1002317@ds.study.iitm.ac.in	38 min	7:04 PM	7:42 PM
SANKET	SATISH METREWAR	21f3000961@ds.study.iitm.ac.in	42 min	7:00 PM	7:42 PM
PRATHAM	SHARMA	21f1006197@ds.study.iitm.ac.in	48 min	6:54 PM	7:42 PM
Neon	Thapa	21f1000706@ds.study.iitm.ac.in	37 min	7:05 PM	7:42 PM
Leo	Tom	leot******@***.com	28 min	7:09 PM	7:42 PM
AMAN	VERMA	21f1006376@ds.study.iitm.ac.in	46 min	6:56 PM	7:42 PM

- **Approval of minutes**:
  - The minutes from the previous meeting were read and approved.
- **Project**:
  - Anjali and Pratham were assigned with tracking and scheduling the project sprints.
  - Abdul and Neon were assigned with component design.
  - Sanket works on API design.
  - Sahil is tasked with creating the report.
- **JIRA:**
  - The JIRA project ownership will be transferred to Anjali for smoother tracking of issues and user stories
- **Next meeting:**
  - https://meet.google.com/rxb-hbuw-wpi

Sprint 5 (Current progress and Milestone 3 discussion)

Date	Jul 9th 2024
Time	11:00 p.m.
Meeting called to order by	Anjali Panchal

- **In Attendance:**



First name	Last name	Email	Duration	Time joined	Time exited
ANJALI	PANCHAL	21f2000411@ds.study.iitm.ac.in	52 min	10:52 PM	11:44 PM
Abdul Ahad	Rauf	21f3002590@ds.study.iitm.ac.in	44 min	11:01 PM	11:44 PM
SAHIL	SANDHU	21f1002317@ds.study.iitm.ac.in	54 min	10:50 PM	11:44 PM
SANKET	SATISH METREWAR	21f3000961@ds.study.iitm.ac.in	40 min	11:04 PM	11:44 PM
PRATHAM	SHARMA	21f1006197@ds.study.iitm.ac.in	42 min	11:02 PM	11:44 PM
Neon	Thapa	21f1000706@ds.study.iitm.ac.in	47 min	10:58 PM	11:44 PM
Leo	Tom	leot******@***.com	39 min	11:05 PM	11:44 PM

- **Approval of minutes:**
  - The minutes from the previous meetings were read and approved.
- **Project:**
  - Everyone discussed their assigned task and current progress.
  - Milestone 3 tentative document discussed along with suggestions from everyone.
- **Next meeting:**
  - https://meet.google.com/rxb-hbuw-wpi

Super Seek API Test Cases

This Markdown file documents tests for the Super Seek API, which manages educational content and user interactions.

1. CONTENT MANAGEMENT

1.1. /contents

1.1.1 Test: test_get_available_weeks_successful()

- **Description:** Tests successful retrieval of available weeks for content.
- **Example - True Scenario:**

```
response = requests.get(f"{BASE_URL}/contents")
assert response.status_code == 200
assert "weeks" in response.json()
```

- **Example – False Scenario:**

```
response = requests.get(f"{BASE_URL}/contents")
assert response.status_code != 404
```

1.2. /contents/{week}

1.2.1 Test: test_get_content_for_week_successful()

- **Description:** Tests successful retrieval of content titles for a specific week.
- **Example - True Scenario:**

```
week_number = 1
response = requests.get(f"{BASE_URL}/contents/{week_number}")
assert response.status_code == 200
assert response.json()["week"] == week_number
assert "lectures" in response.json()
assert "assignments" in response.json()
```

- **Example – False Scenario:**

```
week_number = 10
response = requests.get(f"{BASE_URL}/contents/{week_number}")
assert response.status_code == 404
```

1.2.2 Test: test_get_content_for_invalid_week()

- **Description:** Tests retrieval of content for a non-existing week.
- **Example - True Scenario:**

```
week_number = -1
response = requests.get(f"{BASE_URL}/contents/{week_number}")
assert response.status_code == 400
```

- **Example – False Scenario:**

```
week_number = 1
response = requests.get(f"{BASE_URL}/contents/{week_number}")
assert response.status_code != 400
```

1.3. /lec/{lec_id}/{week}

1.3.1. Test: test_get_lecture_details_successful()

- **Description:** Tests successful retrieval of lecture details.
- **Example - True Scenario:**

```
lec_id = 101
week_number = 1
response = requests.get(f"{BASE_URL}/lec/{lec_id}/{week_number}")
assert response.status_code == 200
assert response.json()["lec_id"] == lec_id
```

- **Example – False Scenario:**

```
lec_id = 999
week_number = 1
response = requests.get(f"{BASE_URL}/lec/{lec_id}/{week_number}")
assert response.status_code == 404
```

1.3.2. Test: test_get_lecture_details_invalid_id()

- **Description:** Tests retrieval of lecture details with an invalid lecture ID.
- **Example - True Scenario:**

```
lec_id = "invalid"
week_number = 1
response = requests.get(f"{BASE_URL}/lec/{lec_id}/{week_number}")
assert response.status_code == 400
```

- **Example – False Scenario:**

```
lec_id = 101
week_number = 1
response = requests.get(f"{BASE_URL}/lec/{lec_id}/{week_number}")
assert response.status_code != 400
```

1.4. /assignment/{assign_id}/{week}

1.4.1. Test: test_get_assignment_details_successful()

- **Description:** Tests successful retrieval of assignment details.
- **Example - True Scenario:**

```
assign_id = 201
week_number = 1
response = requests.get(f"{BASE_URL}/assignment/{assign_id}/{week_number}
assert response.status_code == 200
assert response.json()["assign_id"] == assign_id
```

- **Example – False Scenario:**

```
assign_id = 888
week_number = 1
response = requests.get(f"{BASE_URL}/assignment/{assign_id}/{week_number}")
assert response.status_code == 404
```

1.4.2. Test: test_get_assignment_details_invalid_id()

- **Description:** Tests retrieval of assignment details with an invalid assignment ID.
- **Example - True Scenario:**

```
assign_id = -1
week_number = 1
response = requests.get(f"{BASE_URL}/assignment/{assign_id}/{week_number}")
assert response.status_code == 400
```

- **Example – False Scenario:**

```
assign_id = 201
week_number = 1
response = requests.get(f"{BASE_URL}/assignment/{assign_id}/{week_number}")
assert response.status_code != 400
```

1.5. /solve_assignment/{assign_id}

1.5.1. Test: test_submit_assignment_successful()

- **Description:** Tests successful submission of an assignment.
- **Example - True Scenario:**

```
assign_id = 201
submission_data = {
    "user_id": 1,
    "question_id": 301,
    "selected_answer": "4",
    "code_submission": "def add(a, b):\n  return a + b"
}

response = requests.post(f"{BASE_URL}/solve_assignment/{assign_id}",
                         json=submission_data)
assert response.status_code == 200
assert "message" in response.json()
assert "is_correct" in response.json()
```

- **Example - False Scenario:**

```
assign_id = 201
submission_data = {
    "user_id": 1
}
response = requests.post(f"{BASE_URL}/solve_assignment/{assign_id}",
                         json=submission_data)
assert response.status_code == 400
```

1.5.2. Test: test_submit_assignment_missing_fields()

- **Description:** Tests submission of an assignment with missing required fields.
- **Example - True Scenario:**

```
assign_id = 201
submission_data = {}
response = requests.post(f"{BASE_URL}/solve_assignment/{assign_id}",
                         json=submission_data)
assert response.status_code == 400
```

- **Example – False Scenario:**

```
assign_id = 201
submission_data = {
    "user_id": 1,
    "question_id": 301,
    "selected_answer": "4",
    "code_submission": "def add(a, b):\n  return a + b"
}
response = requests.post(f"{BASE_URL}/solve_assignment/{assign_id}",
                          json=submission_data)
assert response.status_code != 400
```

1.6. /get_summary/{lec_id}\

1.6.1. Test: test_get_lecture_summary_successful()

- **Description:** Tests successfully getting a summary for a lecture.
- **Example - True Scenario:**

```
lec_id = 101
response = requests.get(f"{BASE_URL}/get_summary/{lec_id}")
assert response.status_code == 200
assert "summary" in response.json()
```

- **Example – False Scenario:**

```
lec_id = 999
response = requests.get(f"{BASE_URL}/get_summary/{lec_id}")
assert response.status_code == 404
```

1.6.2. Test: test_get_lecture_summary_invalid_id()

- **Description:** Tests getting a summary with an invalid lecture ID.
- **Example - True Scenario:**

```
lec_id = "abc"
response = requests.get(f"{BASE_URL}/get_summary/{lec_id}")
assert response.status_code == 400
```

- **Example – False Scenario:**

```
lec_id = 101
response = requests.get(f"{BASE_URL}/get_summary/{lec_id}")
assert response.status_code != 400
```

1.7. /feedback/{assign_id}

1.7.1. Test: test_get_feedback_successful()

- **Description:** Tests successfully getting feedback for an assignment.
- **Example - True Scenario:**

```
assign_id = 201
response = requests.get(f"{BASE_URL}/feedback/{assign_id}")
assert response.status_code == 200
assert "feedback" in response.json()
```

- **Example - False Scenario:**

```
assign_id = 888
response = requests.get(f"{BASE_URL}/feedback/{assign_id}")
assert response.status_code == 404
```

1.7.2. Test: test_get_feedback_invalid_id()

- **Description:** Tests getting feedback with an invalid assignment ID.
- **Example - True Scenario:**

```
assign_id = "wrong_id"
response = requests.get(f"{BASE_URL}/feedback/{assign_id}")
assert response.status_code == 400
```

- **Example – False Scenario:**

```
assign_id = 201
response = requests.get(f"{BASE_URL}/feedback/{assign_id}")
assert response.status_code != 400
```

1.8. /get_links/{lec_id}

1.8.1. Test: test_get_links_successful()

- **Description:** Tests successfully getting links for a lecture.
- **Example - True Scenario:**

```
lec_id = 101
response = requests.get(f"{BASE_URL}/get_links/{lec_id}")
assert response.status_code == 200
assert "links" in response.json()
```

- **Example – False Scenario:**

```
lec_id = 999
response = requests.get(f"{BASE_URL}/get_links/{lec_id}")
assert response.status_code == 404
```

1.8.2. Test: test_get_links_invalid_id()

- **Description:** Tests getting links with an invalid lecture ID.
- **Example - True Scenario:**

```
lec_id = "not_a_number"
response = requests.get(f"{BASE_URL}/get_links/{lec_id}")
assert response.status_code == 400
```

- **Example – False Scenario:**

```
lec_id = 101
response = requests.get(f"{BASE_URL}/get_links/{lec_id}")
assert response.status_code != 400
```

1.9. /make_data/{que_id}

1.9.1. Test: test_generate_data_successful()

- **Description:** Tests successful generation of data for a question.
- **Example - True Scenario:**

```
que_id = 301
response = requests.get(f"{BASE_URL}/make_data/{que_id}")
assert response.status_code == 200
assert "data" in response.json()
```

- **Example – False Scenario:**

```
que_id = 777
response = requests.get(f"{BASE_URL}/make_data/{que_id}")
assert response.status_code == 404
```

1.9.2. Test: test_generate_data_invalid_id()

- **Description:** Tests generating data with an invalid question ID.
- **Example - True Scenario:**

```
que_id = 0
response = requests.get(f"{BASE_URL}/make_data/{que_id}")
assert response.status_code == 400
```

- **Example – False Scenario:**

```
que_id = 301
response = requests.get(f"{BASE_URL}/make_data/{que_id}")
assert response.status_code != 400
```

1.10. /logout

1.10.1. Test: test_logout_successful()

- **Description:** Tests successful user logout.
- **Example - True Scenario:**

```
response = requests.post(f"{BASE_URL}/logout")
assert response.status_code == 200
assert "message" in response.json()
```

2. COURSE MANAGEMENT

2.1. /courses

2.1.1. Test: test_get_available_courses_successful()

- **Description:** Tests successful retrieval of available courses.
- **Example - True Scenario:**

```
response = requests.get(f"{BASE_URL}/courses")
assert response.status_code == 200
assert isinstance(response.json(), list)
```

2.2. /enroll/{course_id}

2.2.1. Test: test_enroll_in_course_successful()

- **Description:** Tests successful enrollment in a course.
- **Example - True Scenario:**

```
course_id = 1
response = requests.post(f"{BASE_URL}/enroll/{course_id}")
assert response.status_code == 200
assert "message" in response.json()
```

2.2.2. Test: test_enroll_in_course_invalid_id()

- **Description:** Tests enrolling in a course with an invalid course ID.
- **Example - True Scenario:**

```
course_id = 999
response = requests.post(f"{BASE_URL}/enroll/{course_id}")
assert response.status_code == 404
```

2.3. /progress/{course_id}

2.3.1. Test: test_get_user_progress_successful()

- **Description:** Tests successful retrieval of user progress for a course.
- **Example - True Scenario:**

```
course_id = 1
response = requests.get(f"{BASE_URL}/progress/{course_id}")
assert response.status_code == 200
assert "progress" in response.json()
```

2.3.2. Test: test_get_user_progress_invalid_id()

- **Description:** Tests retrieving user progress with an invalid course ID.
- **Example - True Scenario:**

```
course_id = -5
response = requests.get(f"{BASE_URL}/progress/{course_id}")
assert response.status_code == 400
```

Actual Output

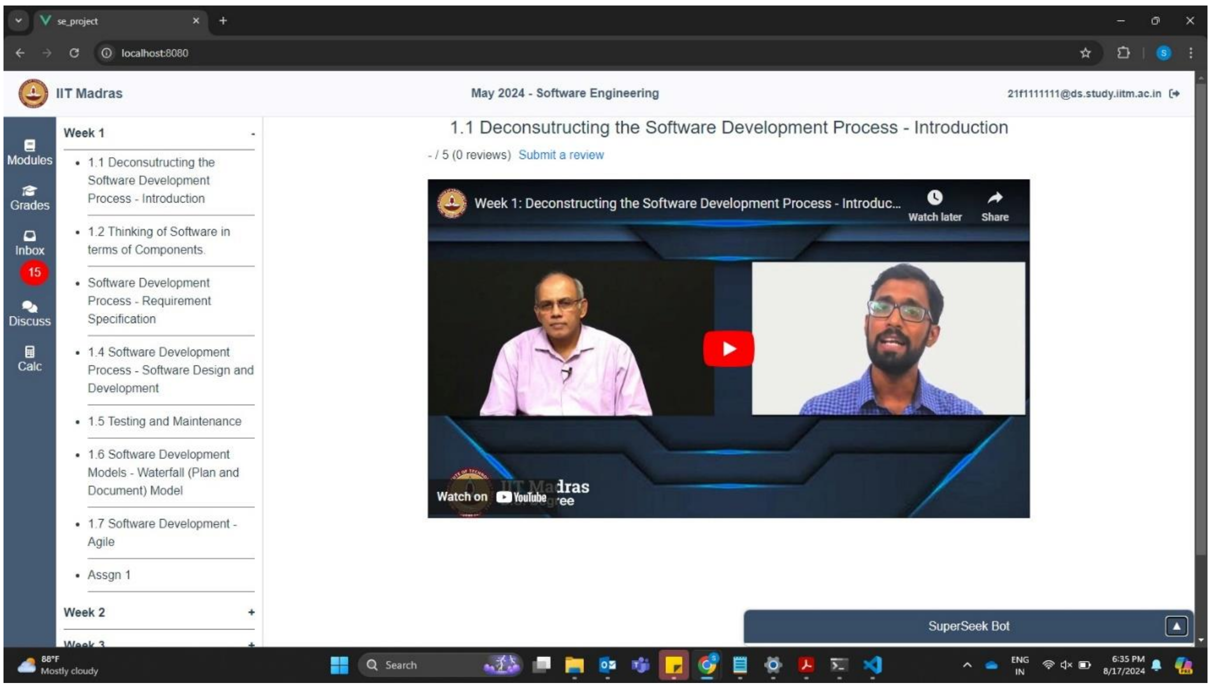
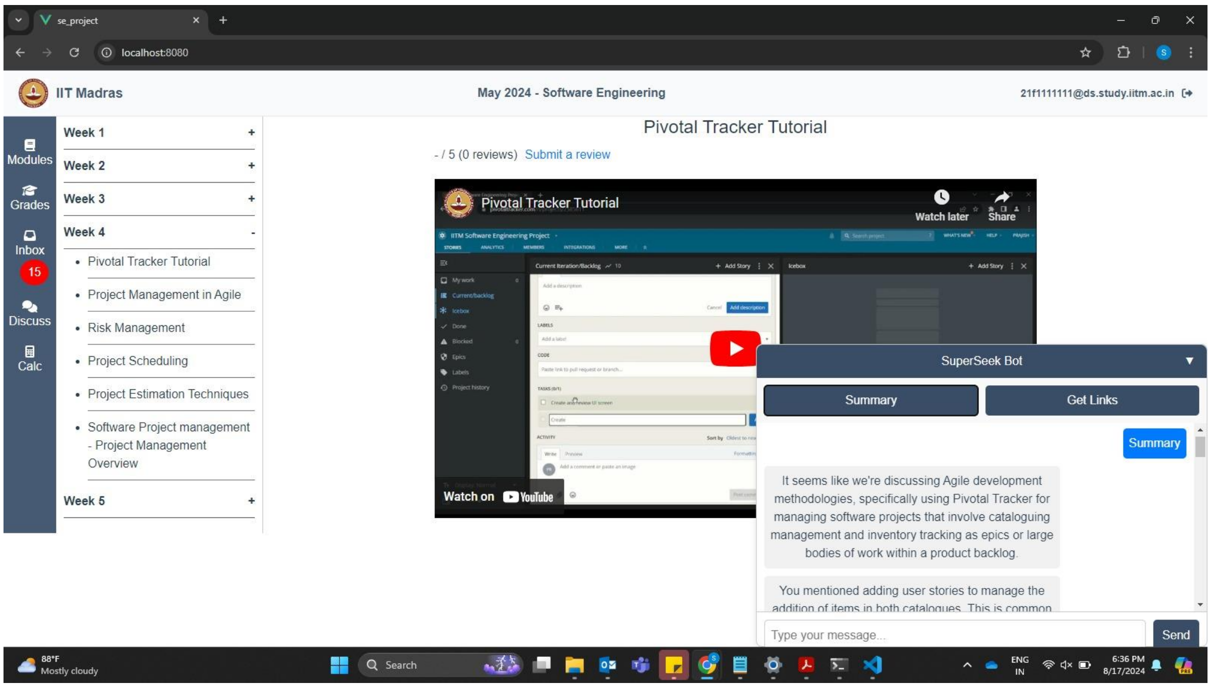

se_project
localhost:8080
IIT Madras
May 2024 - Software Engineering
21f1111111@ds.study.iitm.ac.in
Modules
Grades
Inbox
15
Discuss
Calc
Week 1
1.1 Deconsutructing the Software Development Process - Introduction
1.2 Thinking of Software in terms of Components
Software Development Process - Requirement Specification
1.4 Software Development Process - Software Design and Development
1.5 Testing and Maintenance
1.6 Software Development Models - Waterfall (Plan and Document) Model
1.7 Software Development - Agile
Assgn 1
Week 2
Week 3
1.1 Deconsutructing the Software Development Process - Introduction
- / 5 (0 reviews) Submit a review
Week 1: Deconstructing the Software Development Process - Introduc...
Watch later
Share
Watch on YouTube
SuperSeek Bot

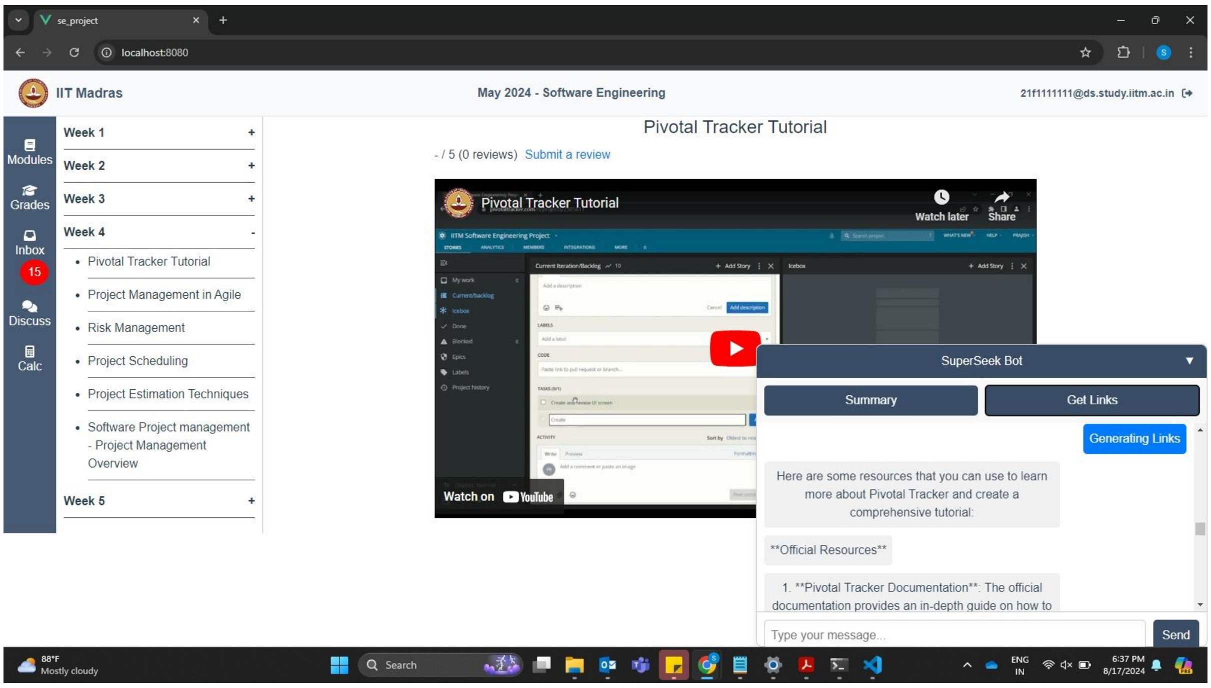
Figure 5: Main screen.



It seems like we're discussing Agile development methodologies, specifically using Pivotal Tracker for managing software projects that involve cataloguing management and inventory tracking as epics or large bodies of work within a product backlog.

You mentioned adding user stories to manage the addition of items in both catalogues. This is common

Figure 6: Get Summary Screen.



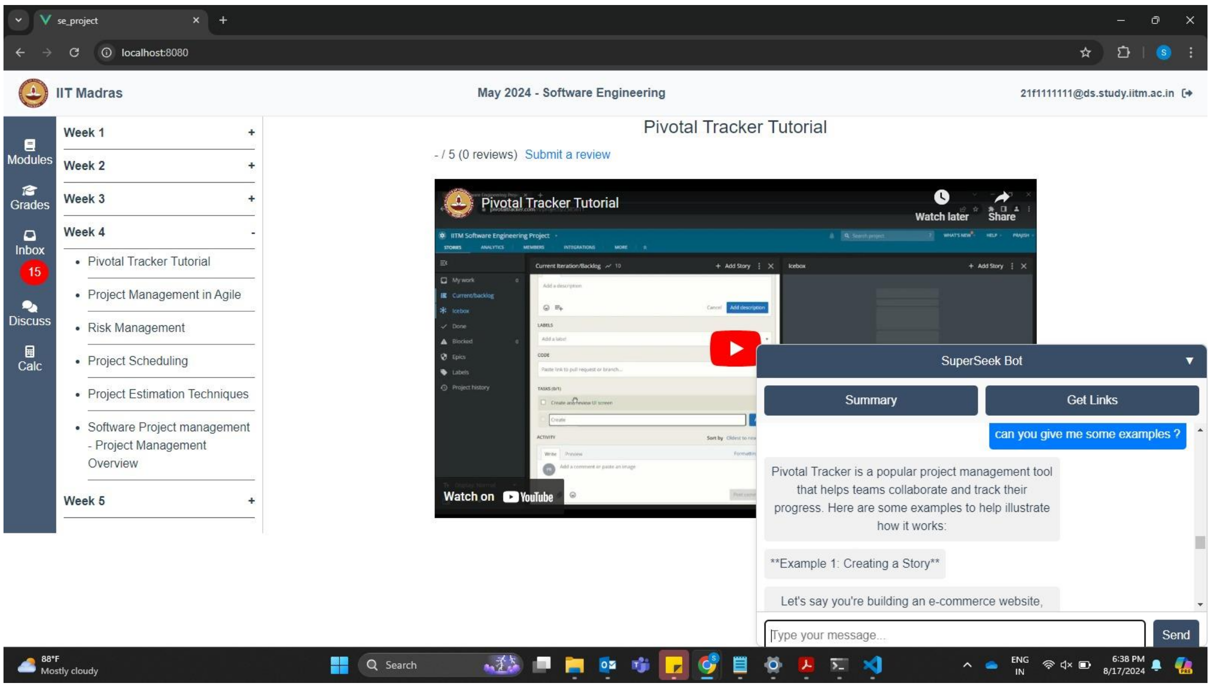
Generating Links

Here are some resources that you can use to learn more about Pivotal Tracker and create a comprehensive tutorial:

Official Resources

1. **Pivotal Tracker Documentation**: The official documentation provides an in-depth guide on how to

Figure 7: Generating external links.



can you give me some examples ?

Pivotal Tracker is a popular project management tool that helps teams collaborate and track their progress. Here are some examples to help illustrate how it works:

Example 1: Creating a Story

Let's say you're building an e-commerce website,

Figure 8: Getting response for custom prompt.