SOFTWARE ENGINEERING PROJECT (Jan 24-May 24)

GROUP 21

- Isha Nayar : CS21B035
- Varshita Devi : CS21B036
- Kshitiz Singh : CS21B044
- Teja Vardhan : CS20B046
- Nisanth D : CS20B057

User Stories

Student

As a student,

I want to be able to create a discourse thread for each ticket in portal,

So that I can effectively communicate with support staff and obtain additional assistance or insights.

As a student,

I want the ability to initiate a new discourse thread for a pre-existing ticket,

So that I can start a conversation even after creating a ticket previously.

As a student,

I want to be able to have a faster mode of ticket resolution,

So that my very important and high-priority tickets can be resolved sooner.

As a student,

I want the ability to view all public discourse threads,

So that I understand the issues other students are facing within the community.

Support Staff & Admin

As a support staff

I want to have the ability to view the discourse thread linked to a ticket

So that I can continue the discussion on discourse thread.

As a support staff

I want the high priority tickets to be notified through g-chat.

So that high priority tickets can be dealt with in time.

As a support staff

I want the student to receive search results of similar discourse threads while creating a new ticket on the portal.

So that repeated queries are avoided and I get to spend my time somewhere better.

As an Admin,

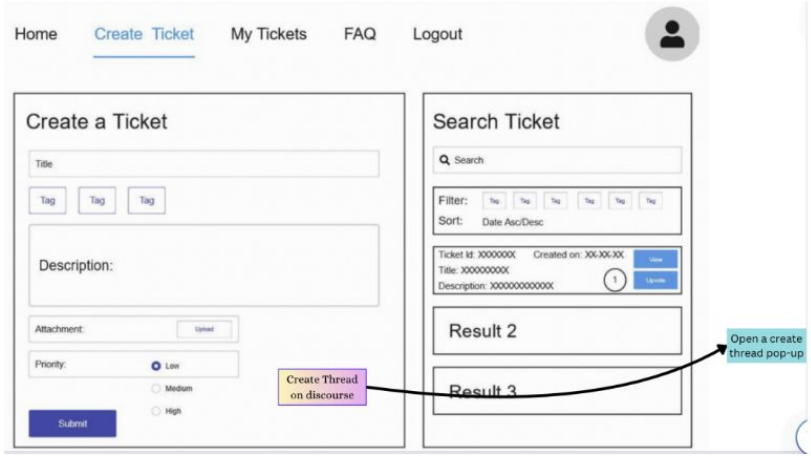
I want to have a mapping of the discourse threads and the corresponding tickets

So that I can ensure orderly resolution and delete any message that is offensive or irrelevant.

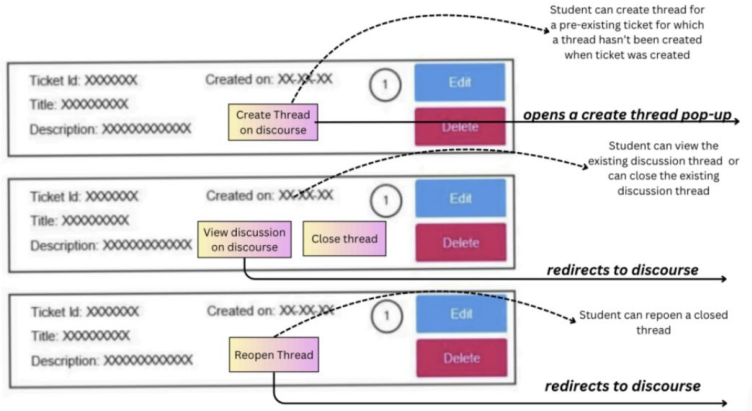
Wireframes

Student

[2.2]Create ticket page



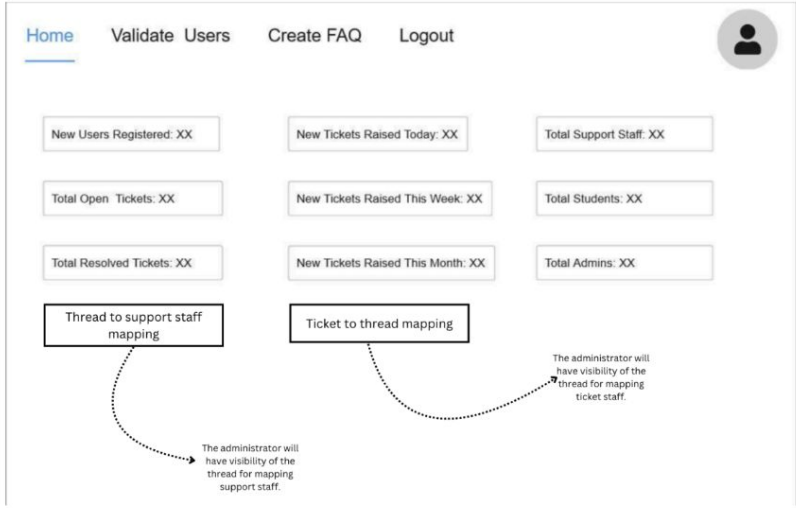
[1.1]Unresolved tickets page



Support staff & Admin

[2.6] Home page

[2.7] Home page



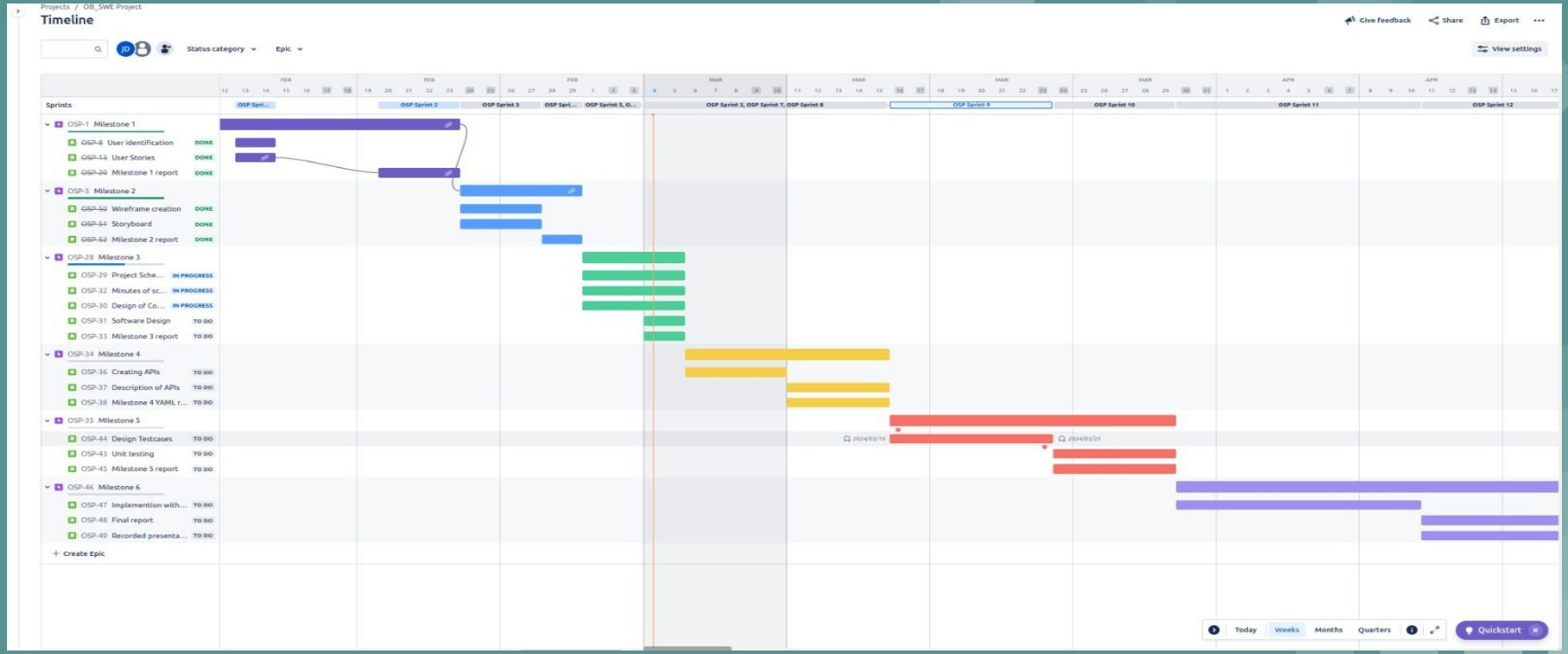
PROJECT SCHEDULING AND SOFTWARE DESIGN

Task Distribution

Milestone	Sub Task	Sprint	Assigned to
1- User Requirement	User identification	1	All
	User Stories	1	All
	Report	2	Kshitiz, Isha,Varshita
2-User Interface	Wireframes	3	Varshita,Isha
	Storyboards	3	Kshitiz,Nishanth,Teja Vardhan
	Report	4	All
3- Project scheduling	Project Scheduling	5	Isha,Varshita
	Component Design	5	All
	Software design	6	Nishanth,Teja Vardhan,Kshitiz
	Scrum meetings	6	Isha,Kshitiz
	Report	6	All
4-API	Design API	7	Varshita,Isha
	Code Review	8	Nishanth,Teja Vardhan,Kshitiz
	YAML Document	9	All
5-Testing	Test case Design	10	Nishanth,Teja Vardhan,Kshitiz
	Unit testing	11	Varshita,Isha
	Report	11	All
6-Submission	Frontend Design	12	Varshita,Isha,Kshitiz
	Demo	13	Nishanth,Teja Vardhan
	Final Report	13	All
	Presentation	14	All

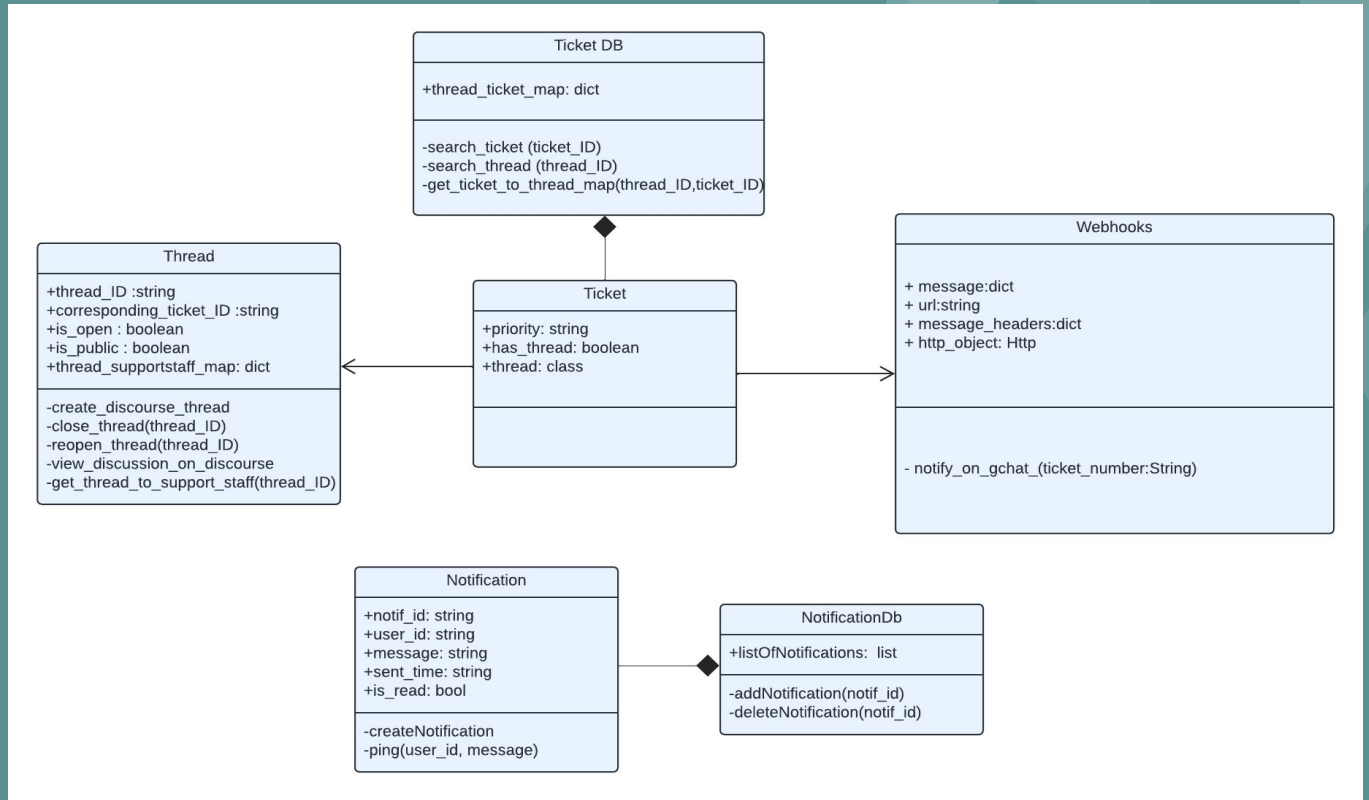
Gantt Chart

We divided the milestones into sub-tasks using the SMART guidelines and scheduled our sprint in the form of Gantt chart using the JIRA software tool.

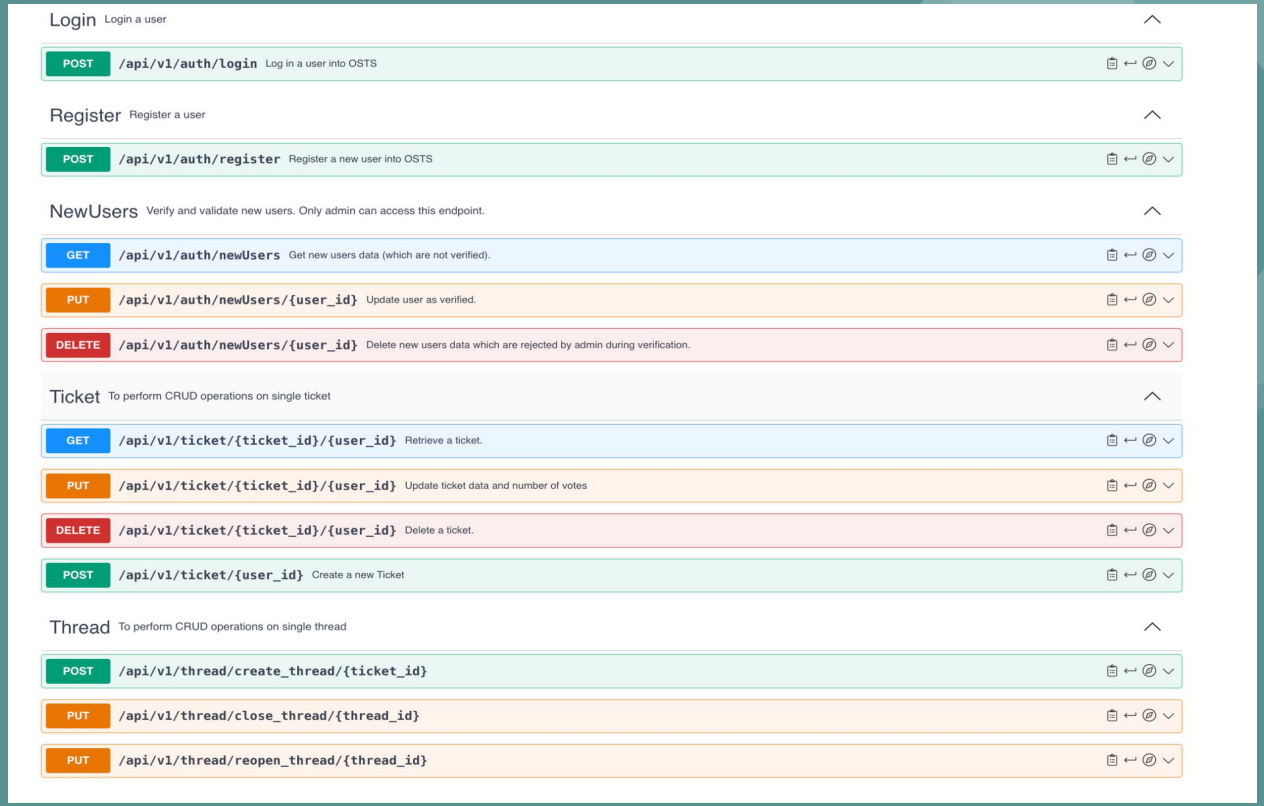


CLASS DIAGRAM

The class diagram depicting the components of the project is shown below:



API Documentation



AllTickets
Get all tickets for different categories and different types of users.

- GET `/api/v1/ticket/all-tickets` Retrieve all tickets for searching.
- GET `/api/v1/ticket/all-tickets/{user_id}` Retrieve all tickets for the user as per user role.

Student
Get or update user details

- GET `/api/v1/student/{user_id}` Get student details and metadata of activities.
- PUT `/api/v1/student/{user_id}` Update student profile data.
- GET `/api/v1/student/view_notification/{user_id}`
- GET `/api/v1/student/view_discussion_on_discourse/{thread_id}`

Support
Get or update support staff details

- GET `/api/v1/support/{user_id}` Get support details and metadata of activities.
- PUT `/api/v1/support/{user_id}` Update support profile data.
- GET `/api/v1/support_staff/view_notification/{user_id}`
- GET `/api/v1/support_staff/view_discussion_on_discourse/{thread_id}`

Admin
Get or update admin details

- GET `/api/v1/admin/{user_id}` Get admin details and metadata of activities.
- PUT `/api/v1/admin/{user_id}` Update admin profile data.
- GET `/api/v1/view_thread_to_ticket_map/{thread_id}`
- GET `/api/v1/admin/view_thread_to_support_staff_map/{thread_id}`

FAQ
Get all FAQs or create a new FAQ.

- GET `/api/v1/faq` Get all FAQ question and answer.
- POST `/api/v1/faq` Create new FAQ.

Webhook
Get all tickets with high priority and sending gchat notification for them.

- POST `/api/v1/webhook/notify_on_gchat/{ticket_id}`

Issue Tracking and code review

We used VSCode as our IDE and git & gitHub as version control system. For each milestone, We had weekly meetings offline where we discussed challenges that we faced during the various phases of the project like requirement gathering, design and development, testing and deployment.

We used the jira website to track our progress sprint wise. We split the project into two parts: discourse integration and webhooks integration, where 3 people worked on discourse part and 2 people on webhooks integration. We faced issues while merging our two parts where we sat together and carefully debugged the issues.

We faced some issues in integration of the frontend and backend part at the end, where we met offline in our department lab and figured out the issues together. Meeting offline and whatsapp was our primary mode of communication.

Tools and Technologies Used

Requirement Gathering and User Stories

Wireframes : Canva and Figma
Report Writing : Google docs

Project Scheduling and Project Design

Planning and Scheduling : Jira Website
Class Diagrams : Figma

Development and Testing

Backend Development:

- Python libraries : httplib2, json, requests, flask framework
- Discourse API

Development and Testing contd.

Frontend Development:
- Vue framework

IDE:
- VsCode

VCS:
- Git and Github

Testing:
- Python unittest

Deployment

- *Setup discourse on localhost*
- *Deployed the web application on localhost*

HOW TO RUN

So, this application is currently hosted locally only.

The ‘backend.bat’ file, ‘frontend.bat’ file and ‘MailHog_windows_amd64.exe’ file are present in the ‘code’ directory.

Steps for Running

STEP 1:
Start the ‘MailHog_windows_amd64.exe’ so that the emails sent during the usage of app, will be captured by ‘MailHog’ at http://127.0.0.1:8025/.

STEP 2:
Then run the ‘backend.bat’ file. It starts the backend server which handles database manipulations and API functions. It runs at http://127.0.0.1:5000/

STEP 3:
Then run the ‘frontend.bat’ file. It starts the frontend server which serves web pages for
the frontend user.

FINALLY, VISIT ‘HTTP://127.0.0.1:8080/HOME’ PAGE ON BROWSER.

Thank you!