PROJECT DETAILS JAN 2026

1. Milestone 1: Deadline by the end of Week 3 : February 22
2. Milestone 2 : Deadline by the end of Week 5 : March 22
3. Sprint 1 (Milestone 3): Deadline by the end of Week 7 : April 5
4. Sprint 2 (Milestone 4): Deadline by the end of Week 8 : April 14
5. Milestone 5: Deadline by the end of Week 9 : April 20

Problem Statement:
Small Business Operations Platform

Context

Small businesses (10–50 employees) often rely on a patchwork of tools such as spreadsheets, messaging apps, emails, and ad-hoc scripts to run daily operations. These systems do not scale, lack auditability, and are difficult to adapt as the business grows.

Your team is tasked with designing and incrementally building a **platform** to support the **core operational workflows of a small business**.

Objective

Design and implement a **software platform** that helps a small business manage its internal operations in a structured, scalable, and maintainable way.

The platform should be developed **incrementally,** with evolving requirements and explicit use of AI tools throughout the development process.

Scope

Each team must **choose one type of small business** and define:

- The primary operational pain points
- The key workflows to support
- The user roles involved

Examples of Small Businesses

- Retail store
- Restaurant or café
- Small manufacturing unit

- Logistics / delivery service
- Professional services firm (law, accounting, consulting)
- Local healthcare clinic (non-clinical operations only)
- Training / coaching institute

Functional Expectations

Your system must:

1. Support **at least two distinct user roles** (e.g., owner, manager, employee)
2. Manage **multiple interrelated workflows**, such as:
    - Orders / jobs / requests
    - Inventory / resources
    - Scheduling / assignments
    - Payments / invoices (no real payment processing required)
    - Customer or vendor interactions
3. Expose **well-defined APIs** for all core functionality
4. Include **automated tests** for critical business logic
5. Be designed assuming **future growth and change**

This project is aimed at providing opportunities for students to explore AI tools and workflows and how it can be effectively integrated into the software engineering workflow.

For each submission, a specific section on AI usage has to be written. This will involve how they have used AI in that submission, and also a URL of their conversations with the AI. (It is also possible to get chat histories from tools like Cursor). Students must submit the prompts they used and mention any agentic AI tools involved.Each submission is also expected to have a section where students reflect on the advantages and challenges they faced while using AI for that submission.

MILESTONE:1

➔ **Focus:** Identify User Requirements

➔ Identify users of the application - primary, secondary and tertiary users.

➔ Conduct interviews and/or observation studies with users to identify pain points and concerns **(DO NOT DIRECTLY ASK THEM IF A CERTAIN FEATURE WILL BE USEFUL. THE PURPOSE OF THE INTERVIEWS IS TO IDENTIFY PROBLEMS THAT THEY FACE) : Submit a part of the video/audio**

recording as proof

- ➔ Write user stories for the requirements, based on the SMART guidelines discussed in the lectures
- ➔ The user stories should be in the following format:
  - As a [type of user],
  - I want [an action],
  - So that [a benefit/value]

Deliverables: [Format: PDF]

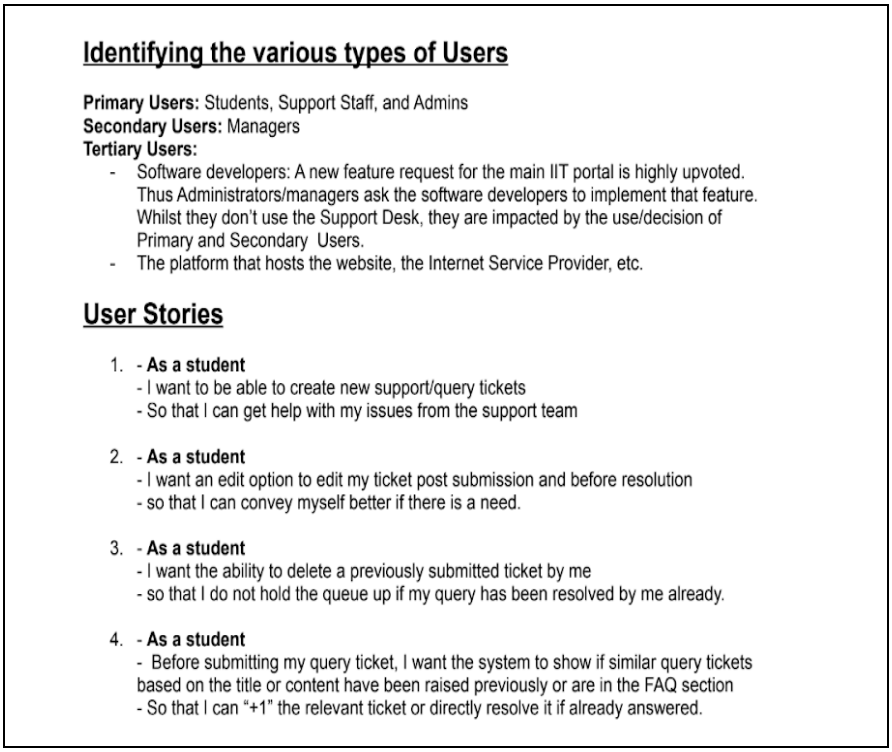
1. Identification of Users (Primary, secondary, and tertiary users.)
2. User Research Interviews and/or observation studies.
   Focus on identifying pain points (not suggesting features directly).
3. User Stories written using SMART guidelines.

PEER EVALUATION QUESTIONS:

- ➔ **1. Identify users (out of 5):** All users identified - 5, Partially identified-3, Absent - 0
- ➔ **2. Conduct interviews and/or observation studies with users,** identify key pain points and appropriate features to address concerns (out of 10) - Identified relevant and comprehensive pain points and features - 10, Partially identified - 5, Absent - 0
- ➔ **3. Write user stories (out of 15):** Fully identified - 15, Partially identified (as per the SMART guidelines) - 10, Partially identified (but mostly not as per the SMART guidelines) - 5, Absent - 0

  [Please check whether the user stories follow the SMART guidelines]

The following image is for illustration purposes.



Identifying the various types of Users

Primary Users: Students, Support Staff, and Admins
Secondary Users: Managers
Tertiary Users:

- Software developers: A new feature request for the main IIT portal is highly upvoted. Thus Administrators/managers ask the software developers to implement that feature. Whilst they don't use the Support Desk, they are impacted by the use/decision of Primary and Secondary Users.
- The platform that hosts the website, the Internet Service Provider, etc.

User Stories

1. - **As a student**
   - I want to be able to create new support/query tickets
   - So that I can get help with my issues from the support team
2. - **As a student**
   - I want an edit option to edit my ticket post submission and before resolution
   - so that I can convey myself better if there is a need.
3. - **As a student**
   - I want the ability to delete a previously submitted ticket by me
   - so that I do not hold the queue up if my query has been resolved by me already.
4. - **As a student**
   - Before submitting my query ticket, I want the system to show if similar query tickets based on the title or content have been raised previously or are in the FAQ section
   - So that I can "+1" the relevant ticket or directly resolve it if already answered.

MILESTONE 2

- ➔ **Focus:** Scheduling and Design
- ➔ Project Schedule - come up with a schedule of your overall project based on the user stories created in the previous milestones
- ➔ Create a schedule for your sprints and iterations, timings of your scrum meetings etc. Trello board, Gantt chart - specifying your tasks and contributions.
- ➔ Project Scheduling Tools - which tools are you using? E.g. Pivotal Tracker, Jira
- ➔ Design of Components - Describe different components of your system based on the user stories created in the previous milestones
- ➔ Software Design - Basic class diagrams of your proposed system
- ➔ Database Design - Design of your database
- ➔ Details/Minutes of a few scrum meetings
- ➔ Most of the pages in the UI (It may be modified (up to a certain limit) at a later time if necessary) - E.g.: HTML/CSS/Javascript (if you are using a framework like Vue.js or React.js, pages should be connected and redirection also implemented, have to submit the frontend folder without integration

with backend API's. Additionally **provide a Readme file** to run the frontend code).

➔ **Milestone 2 PDF Report**

Deliverables :

Milestone 2 PDF Report: Consolidated report including all the below components.

1. Design of Components
2. Class Diagram
3. Databased Design diagram
4. Sprint Schedule
5. Scrum Meetings Schedule and Minutes
6. Screenshot of Gantt Chart
7. Screenshot of Kanban Board
8. Screenshots of the Frontend pages have been developed.
9. Zip the frontend code with a README containing installation & run instructions.

PEER EVALUATION QUESTIONS

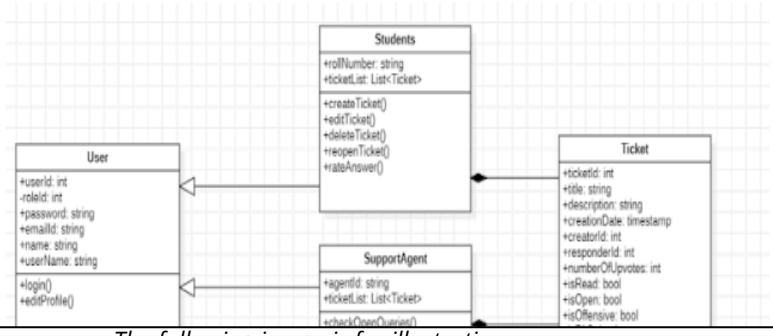
4. **Project Schedule (out of 7):** Full Gantt diagram and description present - 7,Partially present - 4, Absent - 0
5. **Use of project scheduling tools (out of 5):** Present - 5, Partially present - 3, Absent - 0
6. **Describe different components of your system (out of 8):** Components are fully identified with description - 8, Partially present - 4, Absent - 0
7. **Basic class diagrams of your proposed system (out of 10):** All classes with appropriate relationship - 10, Most classes with appropriate relationship - 7, A few classes with appropriate relationship - 5, A few classes with relationship which are not appropriate - 3, Absent - 0
8. **Details/Minutes of a few scrum meetings (out of 5):** Fully present - 5, Partially present - 3, Absent - 0
9. **UI pages (add screenshots) (out of 20):** All important pages - 20, Most of the pages - 15, A few pages - 8, Absent - 0
10. **Overall milestone 1-3 (out of 5):** Very impressive - 5, Impressive - 3, Poor - 0

Design of Components

- **Student View(APIs/Export Jobs)**
  - Create a new ticket
  - Edit an existing ticket (Or upvote an existing one, rate the resolution if already responded to, reopen a closed ticket)
  - Delete an existing ticket
  - Student Dashboard (Read their tickets)
  - Similar queries must be retrieved from the database
  - Celery Task for Notification for query response (within 10 mins)

The following image is for illustration purposes.

Class Diagram



The following image is for illustration purposes.

Sprint Schedule

- **Sprint 1:** Identify the types of users
  - Date: 06/02/2023 - 10/02/2023
- **Sprint 2:** SMART User Stories
  - Date: 11/02/2023 - 16/02/2023
- **Sprint 3:** Vetting & Submission
  - Date: 17/02/2023 - 19/02/2023
- **Sprint 4:** Storyboarding, Wireframe, Applying Usability Principles to Wireframe, Vetting Submission, Final Submission
  - Date: 20/02/2023 - 26/02/2023
- **Sprint 5:** Jira Roadmap Setup, Design of Components, UML Diagrams, Vetting Submission, Final Submission
  - Date: 27/02/2023 - 05/03/2023
- **Sprint 6:** Database Schema, Database Models, User Class, Students

The following image is for illustration purposes.

SCRUM Meetings Schedule and Minutes

SCRUM Meetings: Every Monday, Wednesday, and Friday 19:00-20:30

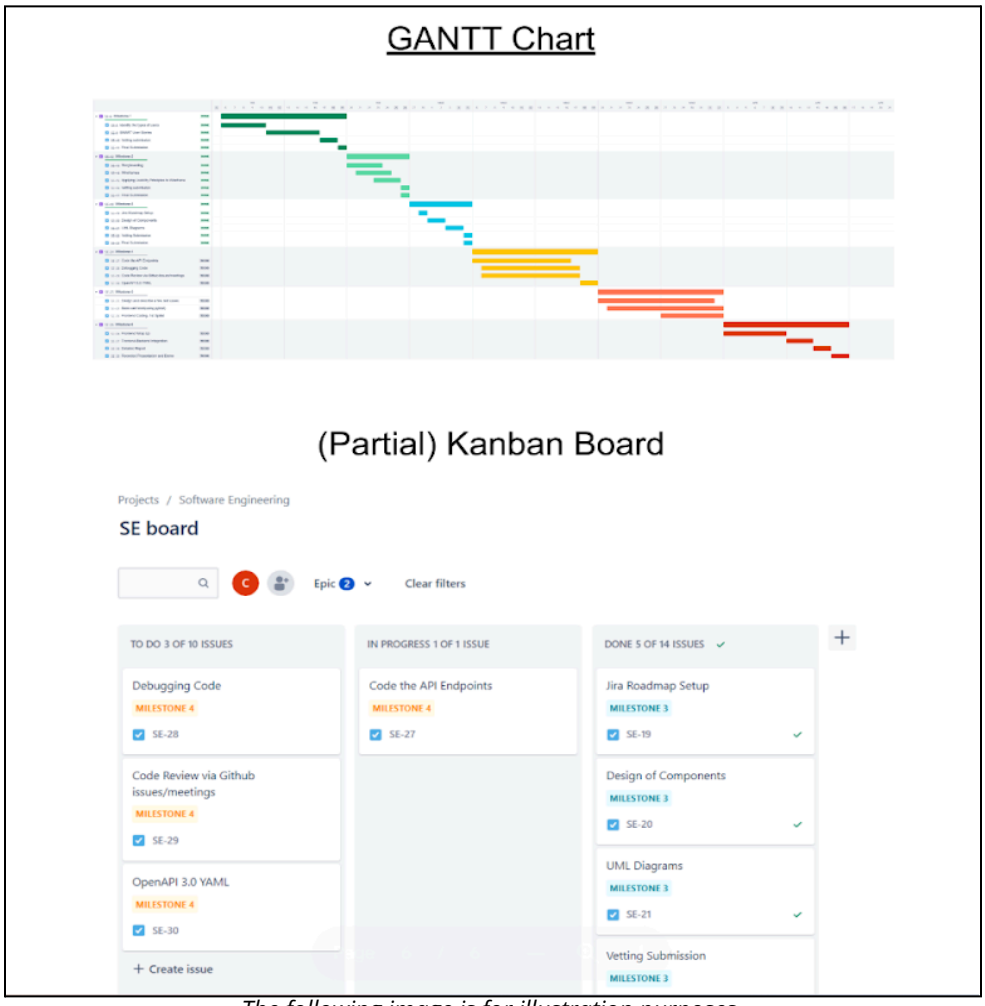
Minutes from Sprint 1 SCRUM Meetings:
Identified and discussed the different types of users (primary, secondary, and tertiary). Discussed a few User Stories associated with the aforementioned users and agreed to come up with at least 10 user stories each.

Minutes from Sprint 2 SCRUM Meetings:
Discussed our User Stories and applied SMART Guidelines to refine the User Stories.

Minutes from Sprint 4 SCRUM Meetings:
Discussed the software to design the storyboards and wireframes. Also worked on the initial draft of our wireframe. Chirag was tasked to mainly refine the wireframe; Varun and Arya were tasked to come up with a few storylines for our storyboards to discuss in our next SCRUM meeting. Discussed our storylines for our storyboards and agreed on a few to take further. Applied design heuristics and found our present software inadequate to express the same. Thus decided to switch to a different software to design our wireframe.

Minutes from Sprint 5 SCRUM Meetings:
Came up with a schedule for our project and an appropriate project scheduling tool, which was decided to be **Jira**. We then designed the components on pen and paper and finally used StarUML to produce the digital version of the same.

The following image is for illustration purposes.



The following image is for illustration purposes.

SPRINT 1 and 2 (Separate Submissions for Each Sprint)

- ➔ Focus: API Endpoints, Test Cases, and User Testing
- ➔ For user story implementations decided for Sprint 1 or 2, create new API endpoints or use appropriate API endpoints from libraries
  - ◆ List of APIs integrated
  - ◆ List of APIs created

- ➔ Description of API endpoints. (As per the problem statement)
- ➔ **Submission of YAML (for the APIs created by dev-team) (When submitting YAML file for API, make sure it is Swagger compatible)**
- ➔ **Code for the APIs (implementation)**
- ➔ For each API endpoint, design extensive test cases. Test cases should be in the following format:

  [API being tested,
  Inputs,
  Expected output,
  Actual Output,
  Result- Success/Fail]
- ➔ Show the features to the end users, get feedback and plan for next Sprint

Deliverables:

1. Documentation of API's in a Swagger-compatible YAML file.
2. Backend code of the implemented API's
   Note: YAML file must have all the error handling, User stories mapping and description of each API listed.
3. Proper test cases to test an API.
4. For each test case, include the input, expected output, and actual output.
5. You can also showcase any API where the actual and expected outputs differ. (This demonstrates how testing helps improve your API.)
6. Submit pytest code for these APIs.
7. User feedback and plan for next sprint

PEER EVALUATION QUESTIONS FOR SPRINT 1 and 2:

1. API Creation and integration (out of 15):

- Provide a detailed description of the APIs in the YAML file, ensuring all API's are clearly listed.
- Explain how the developed API's are linked with the user stories, specifying how each API helps to implement the different user stories provided in Milestone 1. If APIs (such as GenAI API's) have been integrated to create new ones, describe their usage in the next section.

YAML contains all of the required API's and they are mapped to all the user stories. Additionally , any integrated APIs from GenAI (created by dev-team) are clearly listed - 15, YAML has some required APIs and some user stories are implemented, and integrated and GenAI Api's listed- 8, YAML has some required API, but it is not

formatted properly and the user stories not implemented, GenAI API's not listed - 3, Absent - 0

2. Code for the APIs - Implementation (out of 20):

- Provide the complete code for the APIs that have been implemented.
- Ensure that the code is well commented and with proper error handling, validation and responses. The implementation should match the YAML and User stories.

Code for all APIs are present, well documented adheres the best practices (error handling, validation and responses), fully implements the user stories - 20, Code for all the APIs is according to YAML and user stories but best practices (error handling, validation and responses) not followed- 15, Code is incomplete, poorly formatted, lacks proper documentation, but according to User Stories - 10 Code is incomplete, poorly formatted, lacks proper documentation and failed to implement User Stories - 5, Absent - 0

3. Design and describe extensive test cases (out of 20):

- Test cases for APIs created.
- Test cases for other functionalities.

Most of the important test cases with proper format - 20, Some of the important test cases with proper format - 15, Some of the important test cases (not formatted properly)- 10, Test cases are not correct - 5, Absent - 0

4. Some basic unit tests using pytest (out of 5):

pytest code/output present - 5, pytest code/output present, but error are not highlighted - 3, Absent - 0

Sample screenshots based on the previous term's problem statement:

A new item suggested by a support agent is approved/rejected by the admin for FAQ

Page being tested: http://127.0.0.1:5000/api/faq
Inputs:
- Request Method: POST
- JSON: { "category": "operational", "is_approved": false, "ticket_id": 2}
- Header: secret_authtoken: abcxyz

Expected Output:
- HTTP Status Code: 200
- JSON: {"message": "FAQ item added successfully"}

Actual Output:
- HTTP Status Code: 200
- JSON: {"message": "FAQ item added successfully"}

Result: Success

```
def test_faq_authorized_role_post_valid_data():
    input_dict = { "category": "operational","is_approved": False, "ticket_id": 2}
    data = json.dumps(input_dict)
    header={"secret_authtoken":token_login_admin(), "Content-Type":"application/json"}
    request=requests.post(url_faq,data=data, headers=header)
    assert request.status_code==200
    assert request.json()['message']=="FAQ item added successfully"
    faq = FAQ.query.filter_by(ticket_id=2).first()
    assert input_dict["category"] == faq.category
    assert input_dict["is_approved"] == faq.is_approved
```

An existing ticket in the FAQ is updated with a new category

Page being tested: http://127.0.0.1:5000/api/faq
Inputs:
- Request Method: PATCH
- Json body: { "category": "random","is_approved": false, "ticket_id": 2}
- Header: secret_authtoken: abcxyz

Expected Output:
- HTTP Status Code: 200
- JSON: {"message": "FAQ item updated successfully"}

Actual Output:
- HTTP Status Code: 200
- JSON: {"message": "FAQ item updated successfully"}

Result: Success

```
def test_faq_authorized_role_patch_valid_data():
    input_dict = { "category": "random","is_approved": False, "ticket_id": 1}
    data = json.dumps(input_dict)
    header={"secret_authtoken":token_login_admin(), "Content-Type":"application/json"}
    request=requests.patch(url_faq,data=data, headers=header)
    assert request.status_code==200
    assert request.json()['message']=="FAQ item updated successfully"
    faq = FAQ.query.filter_by(ticket_id=1).first()
    assert input_dict["category"] == faq.category
    assert input_dict["is_approved"] == faq.is_approved
```

FINAL SUBMISSION

- ➔ **Focus:** Final Submission pdf
- ➔ Week-12 is completely focused on the project, and no new course content is released this week
- ➔ Complete implementation along with a working prototype.

- ➔ Final project report (consistent with intermediate milestones and sprint documents).
- ➔ Detailed report on work done across all milestones and sprints
- ➔ Implementation details of your project
  - ◆ 1. Technologies and tools used
  - ◆ 2. And instructions to run your application.
- ➔ A section describing code review, issue reporting and tracking using screenshots.
- ➔ Recorded presentation and presentation slides of the working model of your system.

Deliverables:

1. Video presentation (anyone from the team can record a demo of the working of the Project.
2. Presentation (please include a brief overview of the whole project, mainly focusing on user stories and user identification)
3. Complete code of the working Project in zip format.
4. Readme file, which has all the instructions to run the project.
5. The final pdf report, which has a complete compilation of all milestones and sprints .
   a. Including tools and technologies used
   b. Issue tracking that you have used during the project

PEER EVALUATION QUESTIONS:

5. Presentation and recorded demo of your application(out of 5):
Submitted - 5,
Not submitted - 0

6. Tool & Technology (Out of 15):
All tools and technologies are presented - 15,
Most of the tools and technologies are presented - 10,
Some of the tools and technologies are presented- 5,
Absent - 0

7. Issue tracker (Out of 10):
Issue trackers are present with regular updates- 10, Issue trackers are present with irregular updates - 5, Absent -0

8. Instructions to run your application (Out of 5):
Available in full details - 5, Some information available - 3, Absent - 0

9. Overall (out of 5): Very impressive - 5, Impressive - 3, Poor - 0

Implementation Details of the Project

● Technologies and tools used

○ **Technologies for the backend**
■ Flask
■ Flask Restful (For creating API endpoints)
…..

○ **Technologies for the frontend**
■ Vue 3 CLI
■ JavaScript
■ Vue Router
….

○ **Technologies for GenAI integration**
■ OpenAI API
■ Gemini API
■ HuggingFace API
■ LangChain
….

○ **General technologies used**
■ GitHub (for versioning, code management, tracking, reviewing, issues, etc.)
■ Algolia Search is used in the backend and frontend to create a smooth search experience
■ Jira (for project management)
….

● **Hosting:** info regarding hosting

● Instructions to run the application

○ On Ubuntu/MAC OS:
■ Git clone the repository.

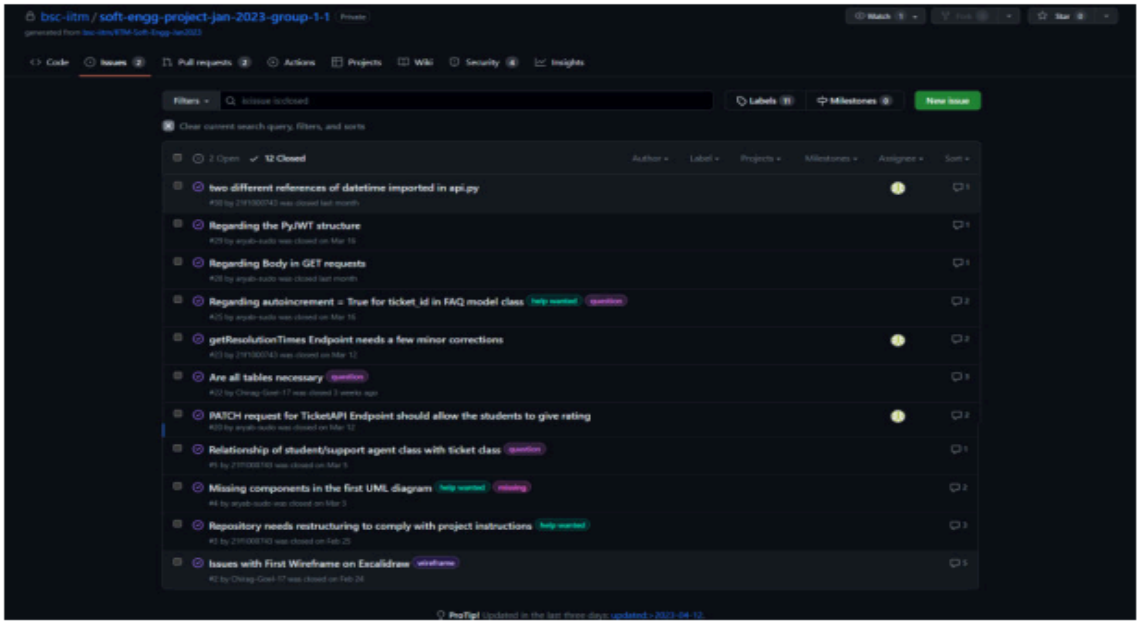
■ Change the directory to the “backend” directory inside the “Milestone-6-Final-Submission” directory using the command :
cd ./Milestone-6-Final-Submission/Code/backend
■ Create a Python virtual environment using the command:
python3 -m venv “<<Name of the virtual environment”>>
■ Activate the virtual environment using the command:
source <<Name of the virtual environment>>/bin/activate
■ Install the requirements using the command :
pip3 install -r requirements.txt…..**and so on**

○ On Windows:
■ Git clone the repository.
■ Change the directory to the “backend” directory inside the “Milestone-6-Final-Submission” directory using the command :
cd .\Milestone-6-Final-Submission\Code\backend…**and so on**

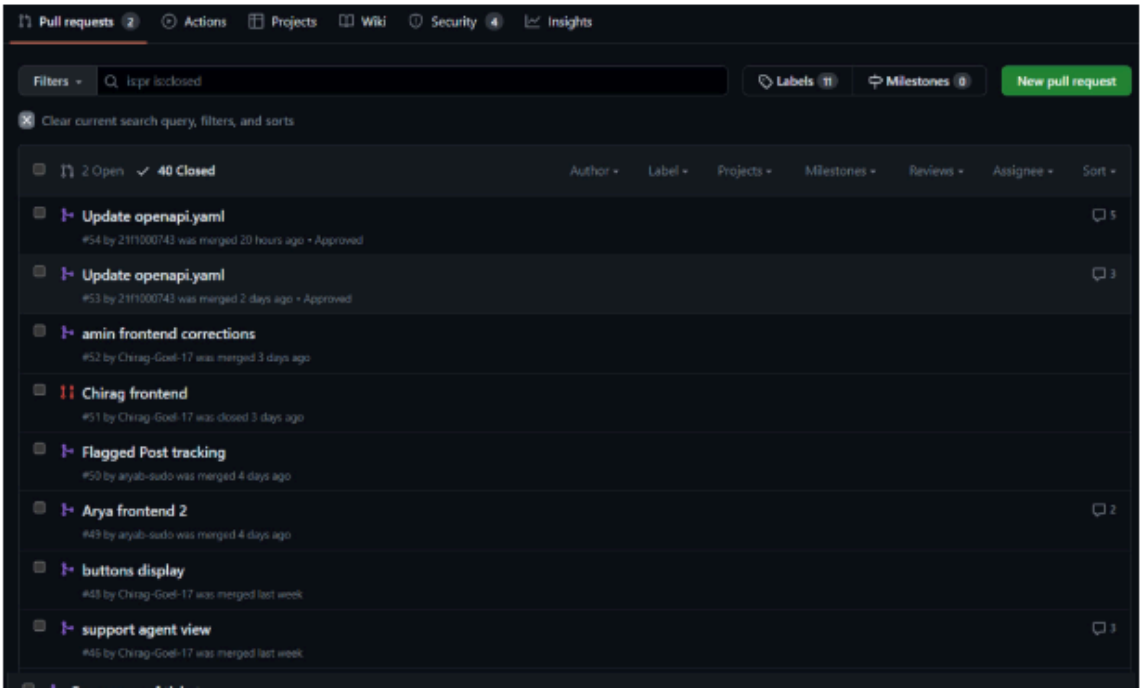
Code Review, Issue Reporting and Tracking

This was completely done on GitHub.

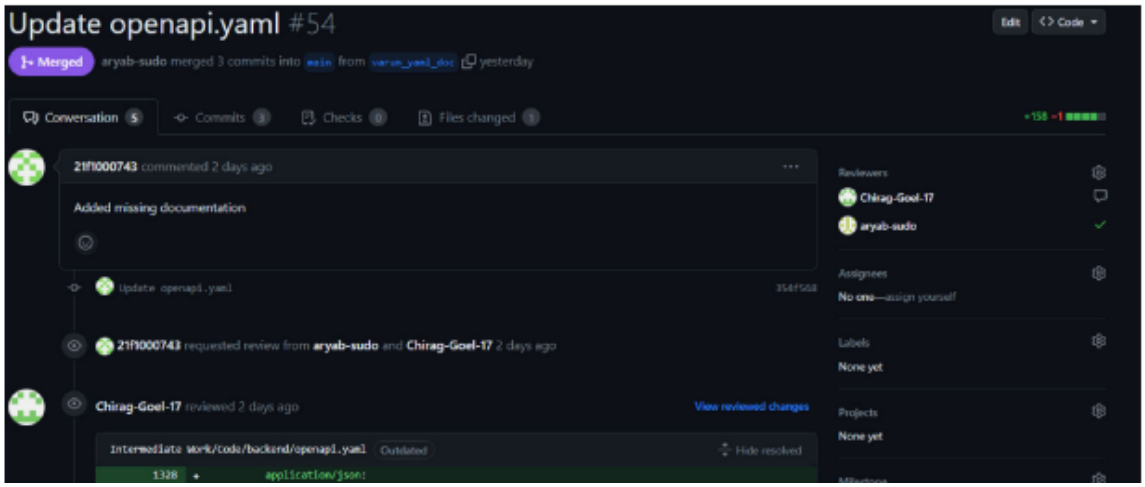
ISSUES:



Pull Requests:



Code Reviews



Project Presentation

- ➔ We will schedule a Software Engineering Project Showcase, where each group will present their project to instructors, other groups, and even other students in the BS program
- ➔ The presentation will be graded.
- ➔ Components evaluated in final Project Evaluation:

    1. The overall design of the project.

    2. Teamwork that includes code review, issue reporting and tracking etc.

    3. Efficient coding practices.

    4. Consistent design through different milestones