Final Project Report

This document is the final project report for our support ticketing system, developed for our Software Engineering project. It includes all our previous milestone submissions and all other relevant details as required by the problem statement. This report is consistent with the intermediate milestone documents.

Adhil Ahmed P M Shums
21F1003972

Afnan Ahmad
21F1003730

Preetodeep Dev
21F1005636

Detailed Report On Work Done From Milestone 1 To Milestone 5:

The following pages include all the milestone submissions made as part of our previous milestones. The milestones have been separated by pages, which demarcate the different sections and indicate the milestone that follows.

MILESTONE 1 SUBMISSIONS FOLLOW

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Honour Code - 21F1003972

I, **ADHIL AHMED P M SHUMS, with roll no. 21F1003972** declare that I will not use any ideas, writings, code, or work that is not my own or my group's with the intention of claiming it my or my group's work. For all the work that I will submit as part of this project, I will not share it outside my group with anybody, directly or indirectly, or upload it to any of the public forums on the internet.

I acknowledge that failing in any of the above constitutes plagiarism and in that case, the institute will take appropriate disciplinary action.

Sign: **ADHIL AHMED P M SHUMS**
Date: 18/02/2023

Honour Code - 21F1003730

I, **AFNAN AHMAD, with roll no. 21F1003730** declare that I will not use any ideas, writings, code, or work that is not my own or my group's with the intention of claiming it my or my group's work. For all the work that I will submit as part of this project, I will not share it outside my group with anybody, directly or indirectly, or upload it to any of the public forums on the internet.

I acknowledge that failing in any of the above constitutes plagiarism and in that case, the institute will take appropriate disciplinary action.

Sign: **AFNAN AHMAD**
Date: 18/02/2023

Honour Code - 21F1005636

I, **PREETODEEP DEV, with roll no. 21F1005636**, declare that I will not use any ideas, writings, code, or work that is not my own or my group's with the intention of claiming it my or my group's work. For all the work that I will submit as part of this project, I will not share it outside my group with anybody, directly or indirectly, or upload it to any of the public forums on the internet.

I acknowledge that failing in any of the above constitutes plagiarism and in that case, the institute will take appropriate disciplinary action.

Sign: **PREETODEEP DEV**
Date: 18/02/2023

Application Users

In this document, we have identified and listed the primary, secondary and tertiary users of the system.

Adhil Ahmed P M Shums	**Afnan Ahmad**	**Preetodeep Dev**
21F1003972	21F1003730	21F1005636

Primary Users

- Students
- Support Staff

Secondary Users

- POD Team (Admins)
- Course Instructors

Tertiary Users

- Course Faculty
- Exam Partners
- IITM Senate

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

User Stories (Core Requirements)

In this document, we have described the user stories that are part of the essential project requirements.

Adhil Ahmed P M Shums	**Afnan Ahmad**	**Preetodeep Dev**
21F1003972	21F1003730	21F1005636

As a Student,

1. I want to be able to create support tickets,
   So that I can have my issues resolved.

2. I want to be able to set the visibility (private / public) of my tickets,
   So that I can regulate who all can view them.

3. I want to be able to upvote existing public tickets,
   So that those issues can be prioritized.

4. When raising a new ticket, I would like to be able to view related public tickets,
   So that I can upvote that and avoid creating a duplicate ticket.

5. I want to be able to track the status of tickets I have raised, or public tickets,
   So that I am able to understand the steps taken to resolve the issue.

6. I want to be notified when my ticket is resolved,
   So that I can have inner peace.

As a Support Staff,

1. I want to be able to see the tickets created on the portal,
   So that I can respond to them.

2. I want to be able to see the tickets I'm working on (i.e. assigned to me),
   So that I have a track of my work.

3. I want to be able to add a message to a ticket,
   So that an update can be provided on the ticket.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

4. I want to be able to update the status of the tickets,
   So that the student(s) can be notified of the resolution.

5. I want to be able to categorize the tickets,
   So that they can be posted as FAQs for future reference.

6. I want to be able to escalate a ticket to an admin and/or admin group(s),
   So that it can be prioritized and acted upon as necessary.

As an Admin,

1. I would like to be able to manage the user accounts,
   So that I can add new support staff / students, and archive old accounts.

2. I would like to view analytics of the tickets,
   So that I am able to improve the efficiency of the support team.

3. I would like to be able to close open tickets corresponding to the previous term at the beginning of the new term.

As a Super Admin,

1. I would like to be able to grant and/or revoke admin privileges to other users, apart from performing the administrative operations,
   So that the system can be moderated appropriately.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

MILESTONE 2 SUBMISSIONS FOLLOW

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

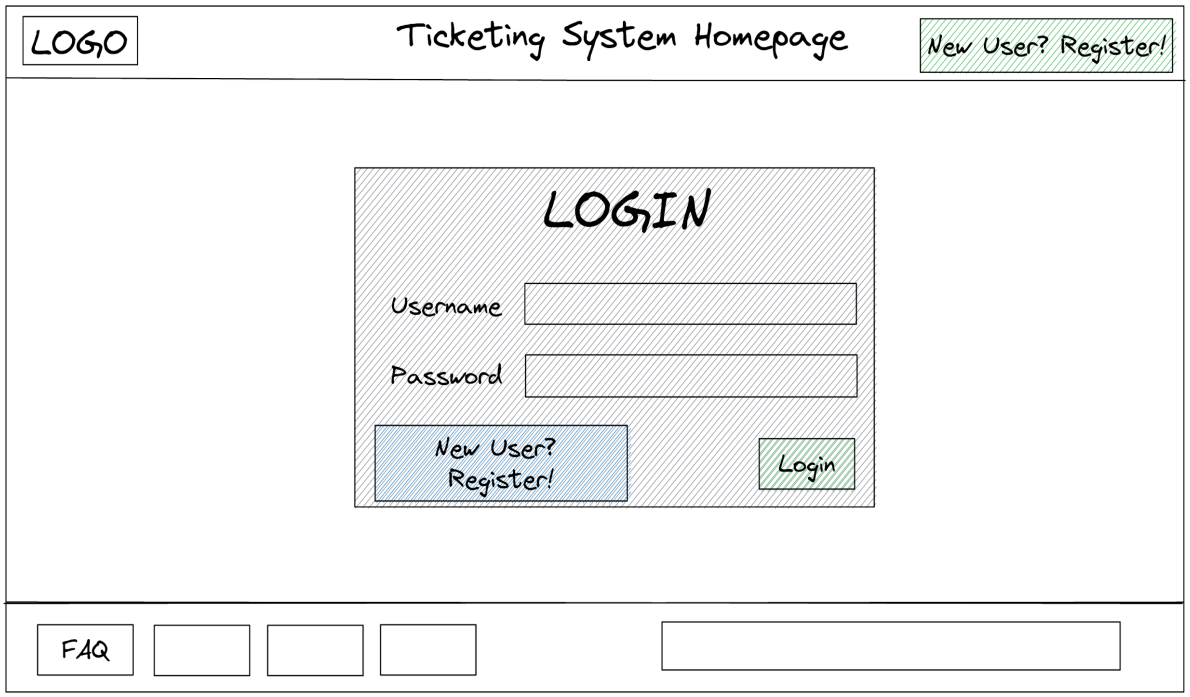
Wireframes - Common Views

In this document, we described the wireframes for the common views that are intended for all users of the application.

Adhil Ahmed P M Shums
21F1003972

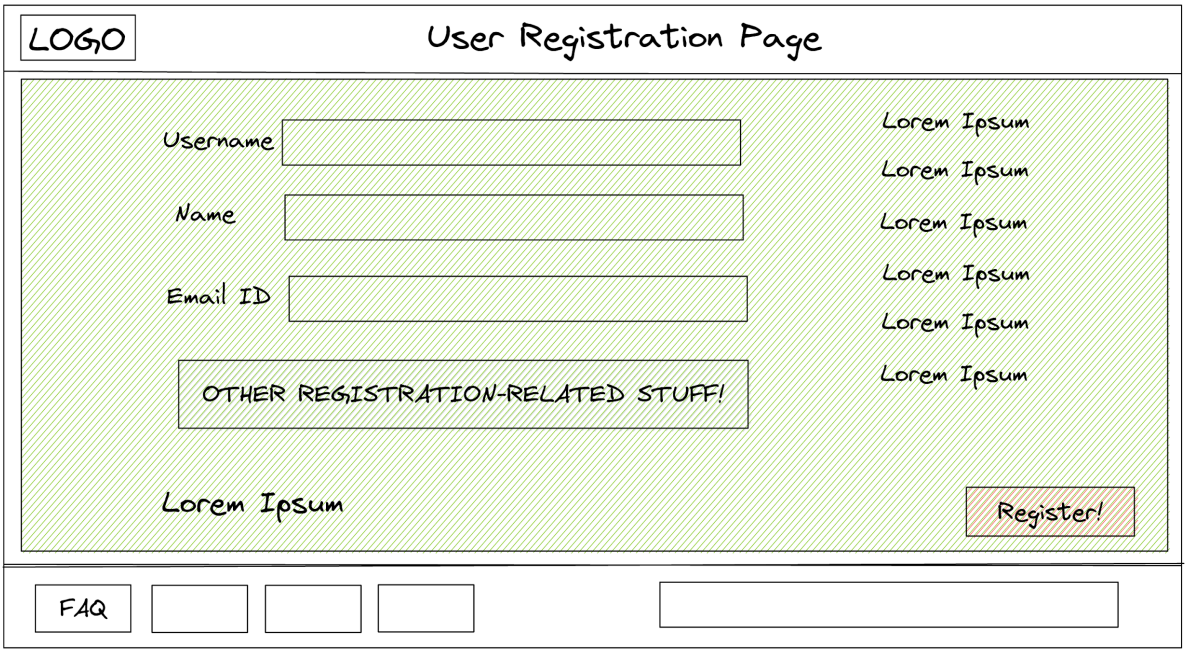
Afnan Ahmad
21F1003730

Preetodeep Dev
21F1005636



Description: This is the view that will come up when a user of the ticketing system (student, support staff, admin, or otherwise) visits the website (i.e the homepage) of the system. This page provides login options for all types of users, as well as options for registration of new users.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Description: This is the view that will come up when a user of the ticketing system (student, support staff, admin, or otherwise) clicks on either of the two register buttons on the homepage. This page collects the bare minimum information that is required to create a new account (such as name, username, email ID, etc.).

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

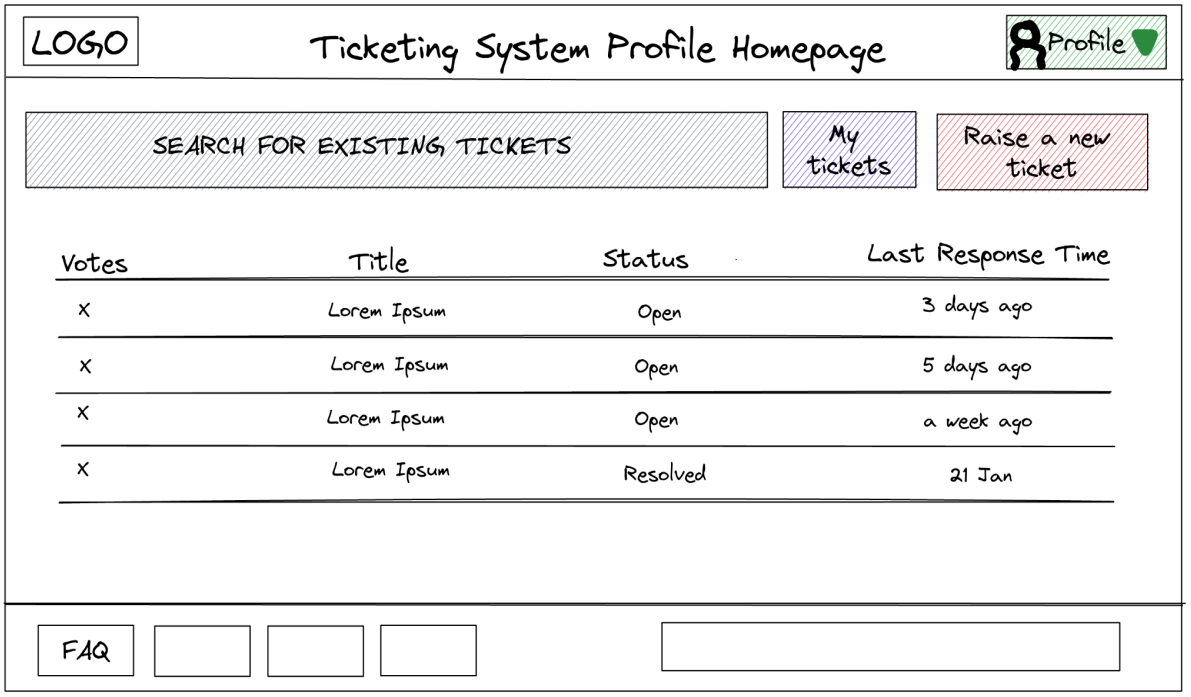
Wireframes - Student Views

In this document, we describe the wireframes for the views that are intended for the students.

Adhil Ahmed P M Shums
21F1003972

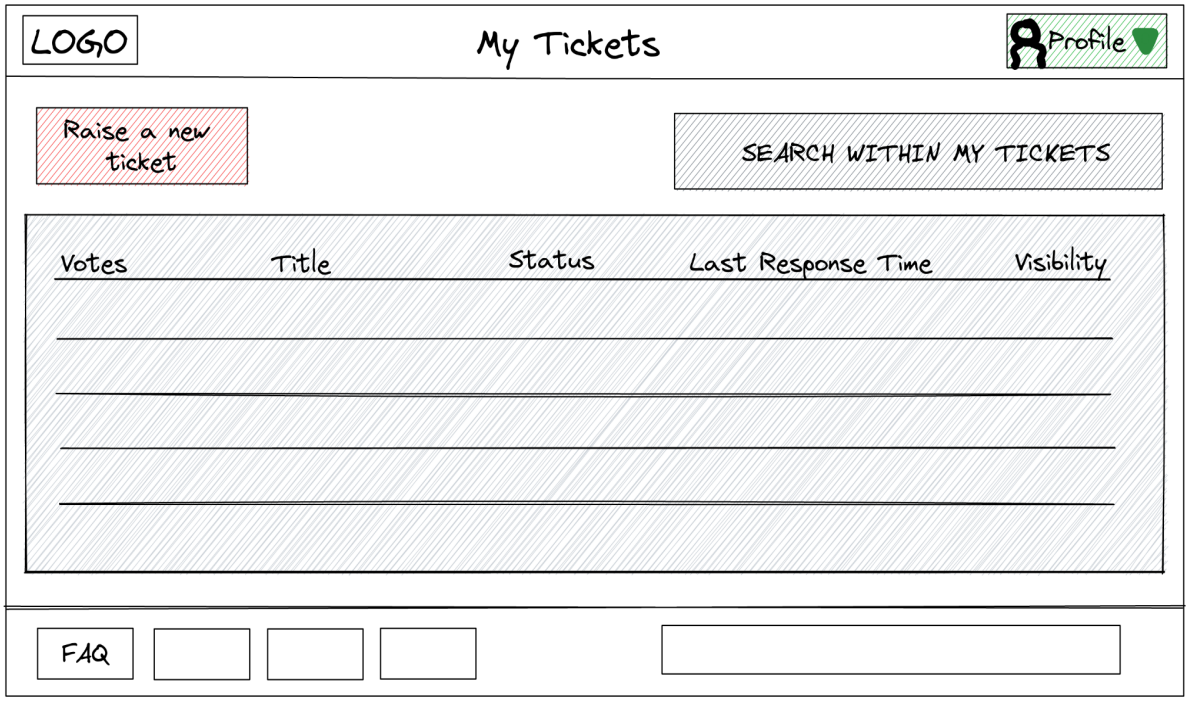
Afnan Ahmad
21F1003730

Preetodeep Dev
21F1005636



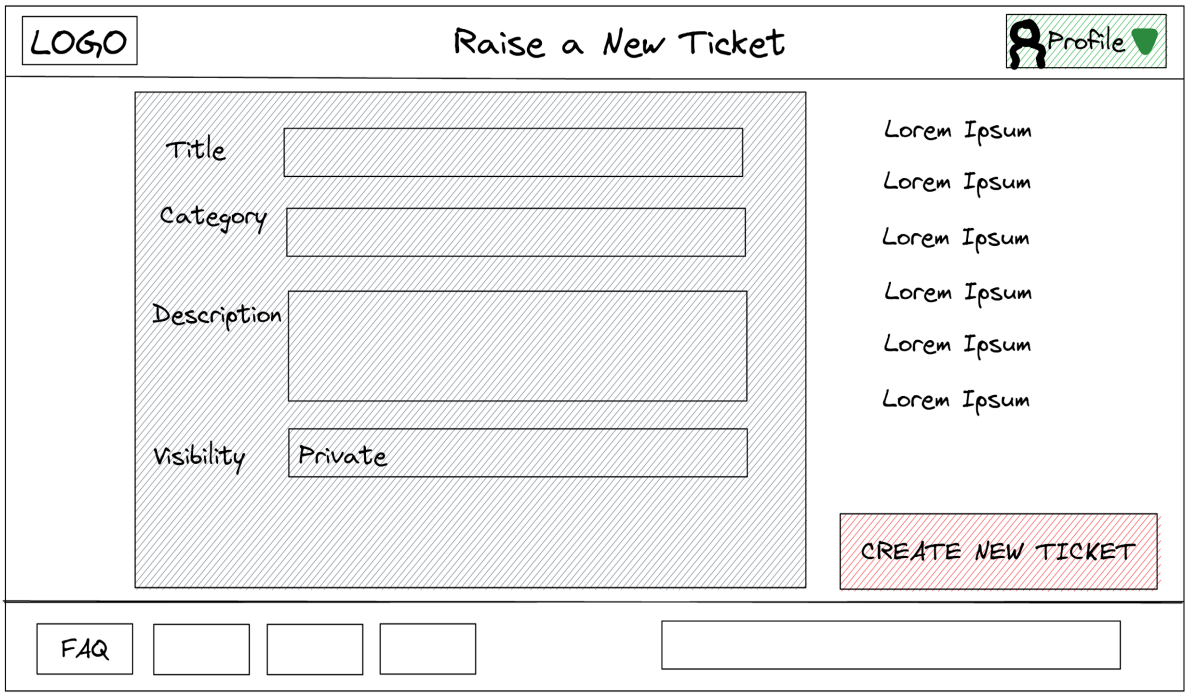
Description: This is the view that will come up when a student logs in to his/her profile (the profile homepage). This page provides a list of all publicly viewable tickets in a tabular format, as well as options for viewing tickets raised by the student himself/herself, and raising a new ticket.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



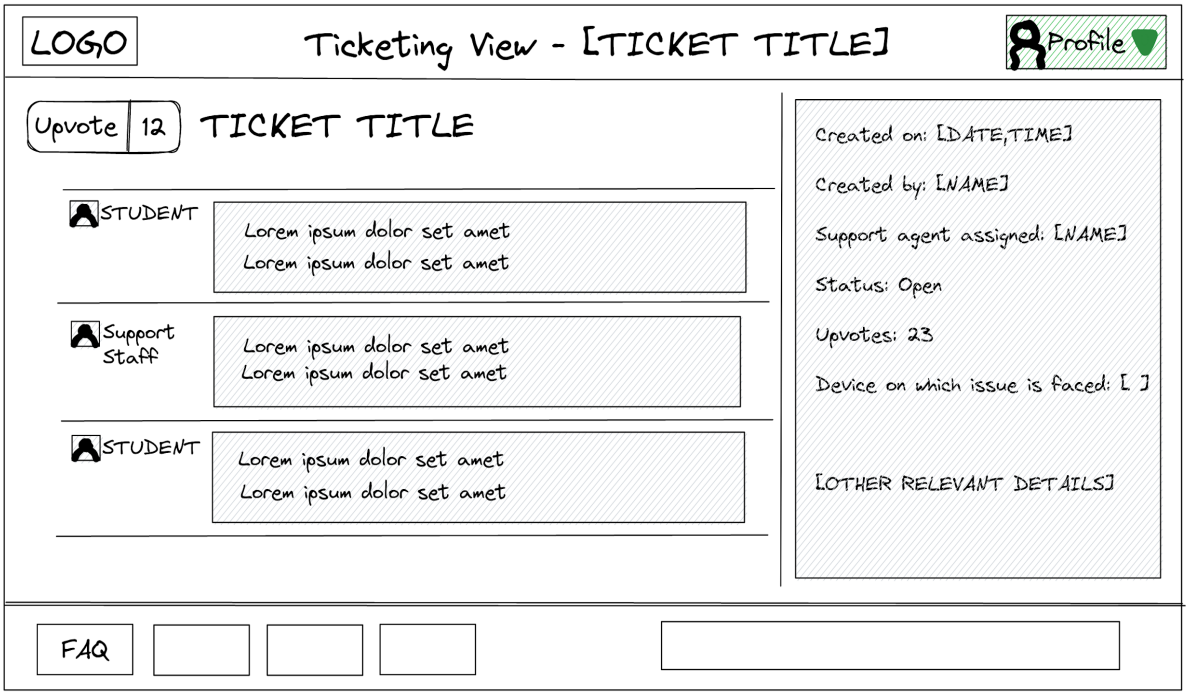
Description: This is the view that will come up when a student clicks on the “My tickets” button from the homepage. This is the page that lists all tickets raised by the logged-in student in a tabular format. Both public and private tickets raised by the student will be listed here. This page also provides options for searching within the preexisting tickets raised by the student.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



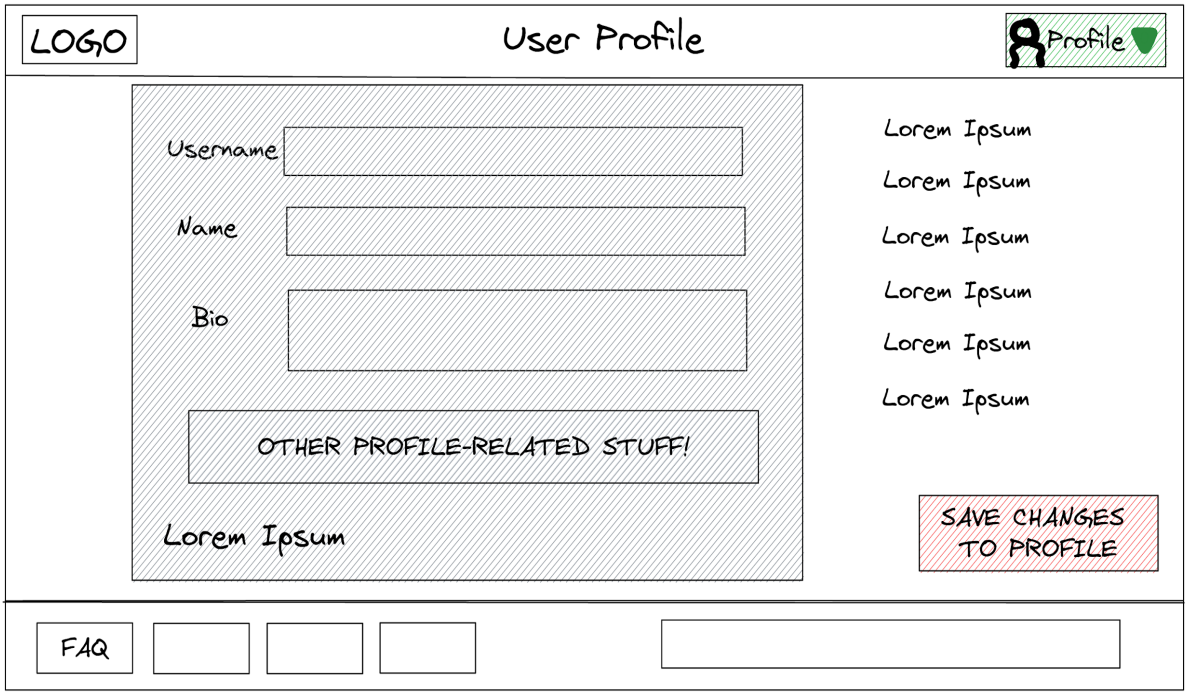
Description: This is the view that will come up when a student clicks on the “Raise new ticket” button from the homepage. This is the page that allows a student to create a new support ticket, and collects all the necessary data associated with a ticket. Various other options related to ticket creation (file uploads, visibility settings, etc.) are also present therein.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Description: This is the view that will come up when a student clicks on one of the public tickets listed on the homepage. This page allows a student to view a support ticket raised by another student, and upvote it if he/she wishes to. Only publicly-viewable tickets will be visible this way. Other relevant details of the public ticket will be visible on the sidebar, as indicated.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Description: This is the view that will come up when a student clicks on the “Profile” drop-down button from the homepage. This page allows a student to edit various attributes associated with his/her profile, such as profile picture, username, name, biography, etc.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

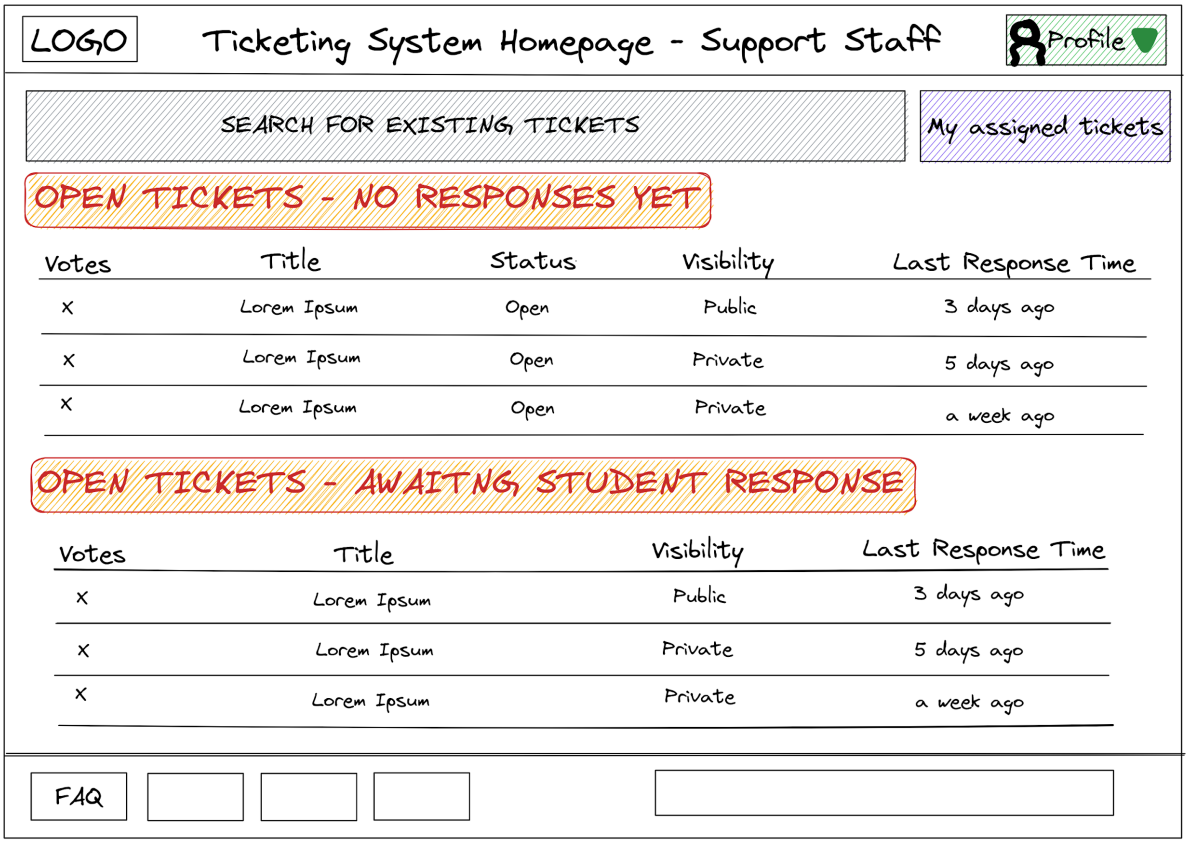
Wireframes - Support Staff Views

In this document, we describe the wireframes for the views that are intended for the support staff.

Adhil Ahmed P M Shums
21F1003972

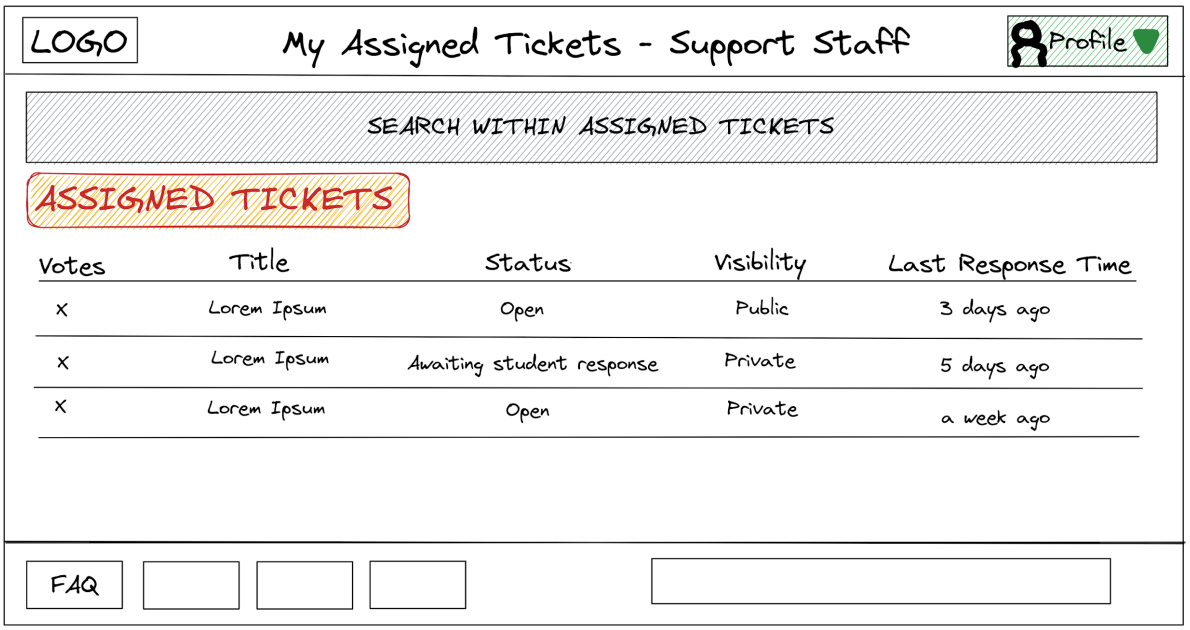
Afnan Ahmad
21F1003730

Preetodeep Dev
21F1005636



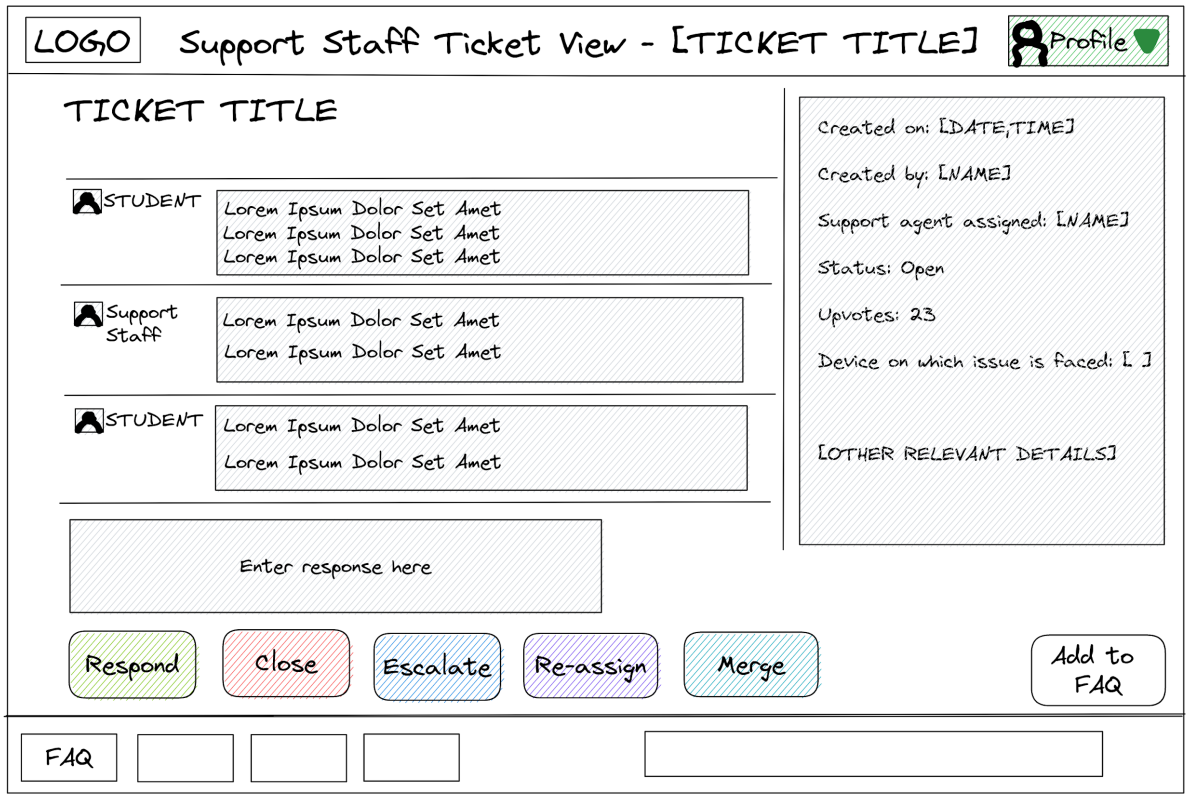
Description: This is the view that will come up when a support staff logs in to his/her profile (the profile homepage). This page provides a list of all open tickets in a tabular format and further divides them into tickets with no response and tickets that are awaiting student response. The support staff can then decide how to follow up with each ticket.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



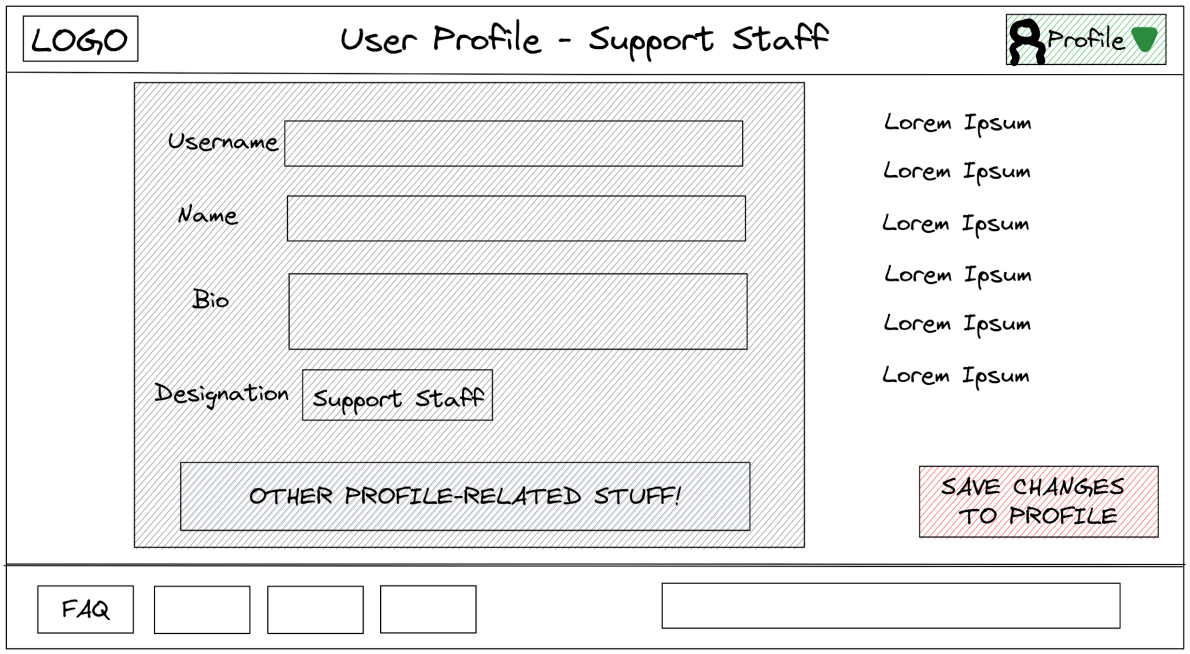
Description: This is the view that will come up when the support staff clicks on the “My assigned tickets” button from the homepage. This is the page that lists all tickets raised that have been assigned to the particular staff in a tabular format. Both public and private tickets raised by the students will be listed here. This page also provides options for searching within the assigned tickets.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Description: This is the view that will come up when the staff clicks on any ticket. This is the page that allows the staff to respond to the ticket. Various other administrative options related to ticket management (respond, close, escalate, re-assign, merge, etc.) are also present therein.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Description: This is the view that will come up when a support staff clicks on the “Profile” drop-down button from their homepage. This page allows the staff to edit various attributes associated with his/her profile, such as profile picture, username, name, biography, etc.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

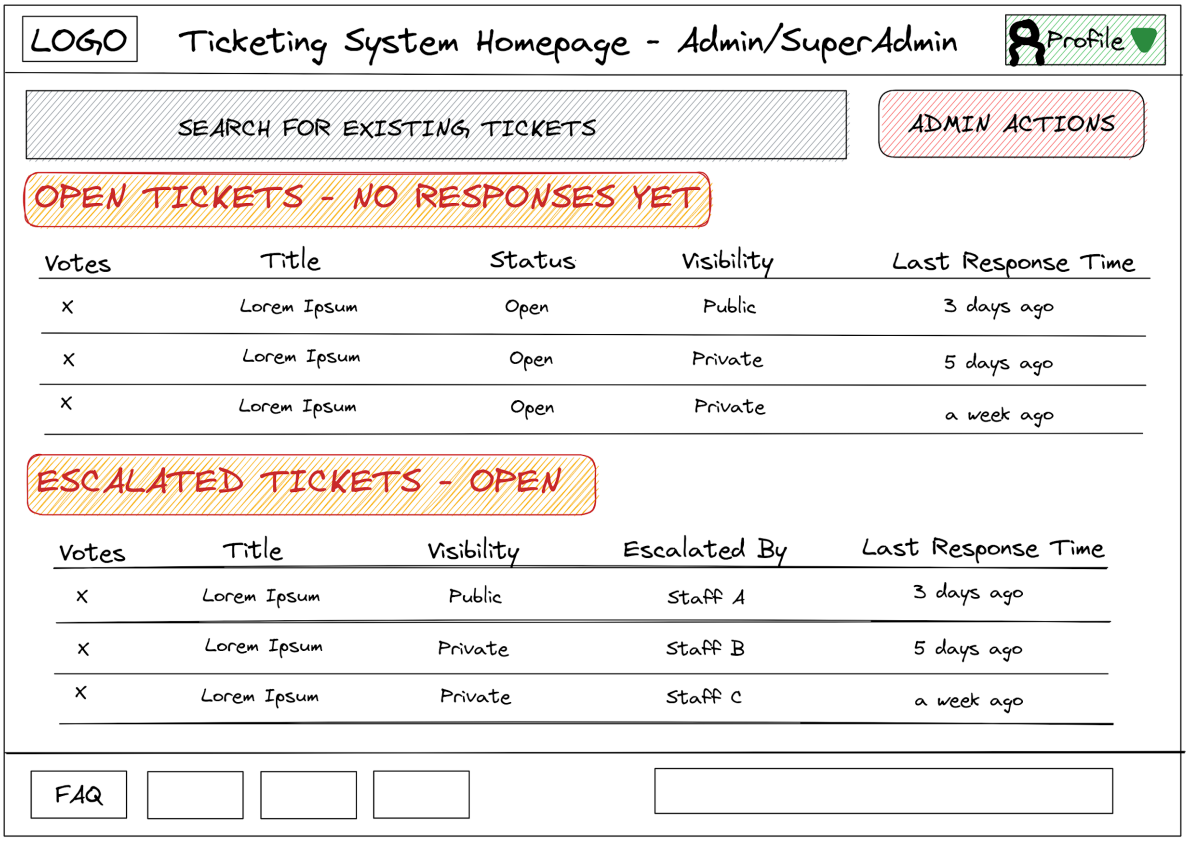
Wireframes - Admin / Super Admin Views

In this document, we describe the wireframes for the views that are intended for the admins/super admins.

Adhil Ahmed P M Shums
21F1003972

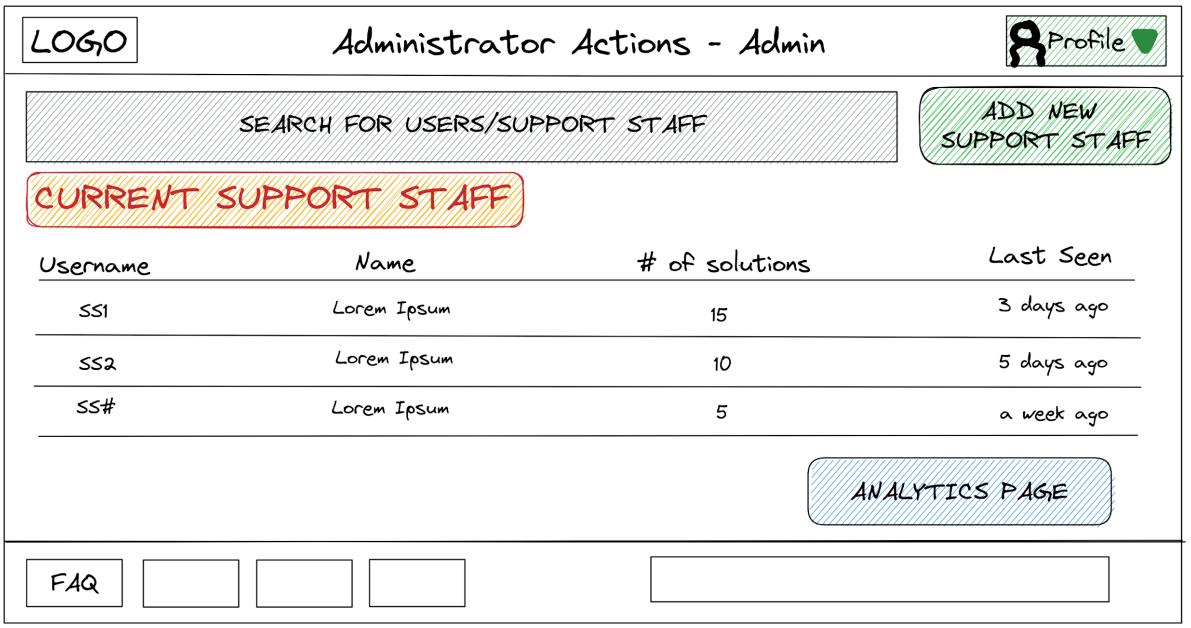
Afnan Ahmad
21F1003730

Preetodeep Dev
21F1005636



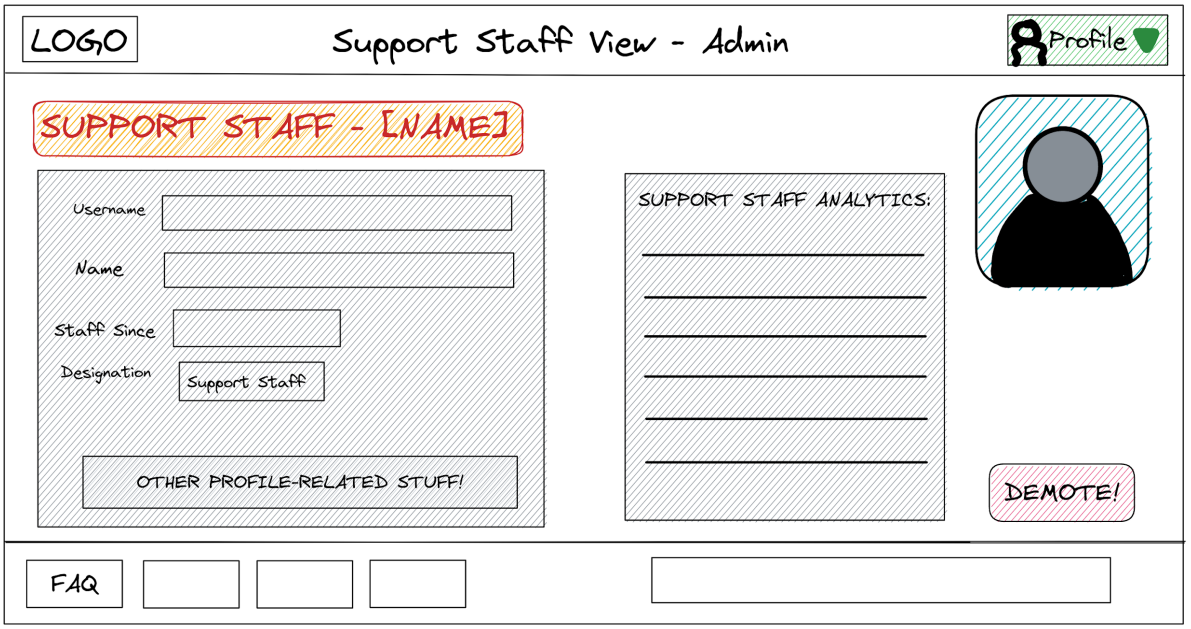
Description: This is the view that will come up when an administrator/super-administrator logs in to his/her profile (the profile homepage). This page provides a list of all open tickets in a tabular format and further divides them into tickets with no response and tickets that are escalated for the administrators to look at. The admins can then decide how to follow up with each escalated ticket.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



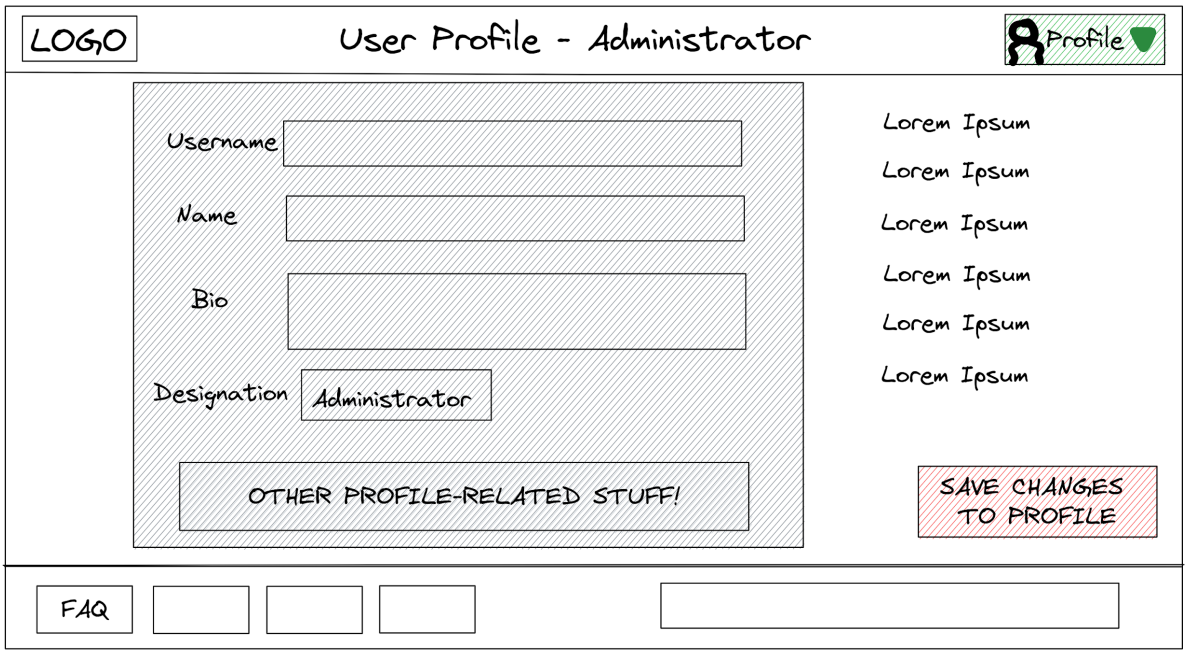
Description: This is the view that will come up when an administrator clicks on the "Administrator Actions" button from the homepage. This is the page that lists all the support staff in a tabular format, as well as some basic metrics related to the support staff. This page also provides options for searching for support staff/students, and adding new support staff. Furthermore, there is also an option for administrators to view analytics related to the ticketing system.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Description: This is the view that will come up when an administrator clicks on any support staff from the list of staff from the admin actions page. This page allows the administrators to view profile details for a support staff, as well as support staff-specific metrics. This page also allows an administrator to demote (i.e remove) a support staff from his/her current position.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

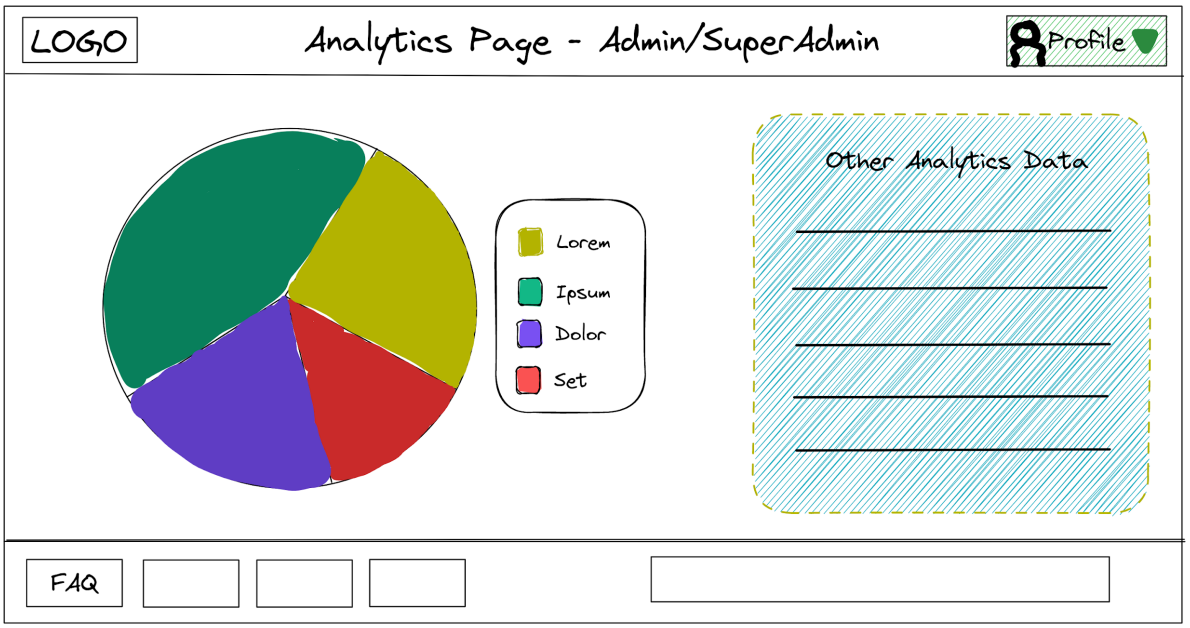


Description: This is the view that will come up when an administrator clicks on the “Profile” drop-down button from his/her homepage. This page allows the administrator to edit various attributes associated with his/her profile, such as profile picture, username, name, biography, etc.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

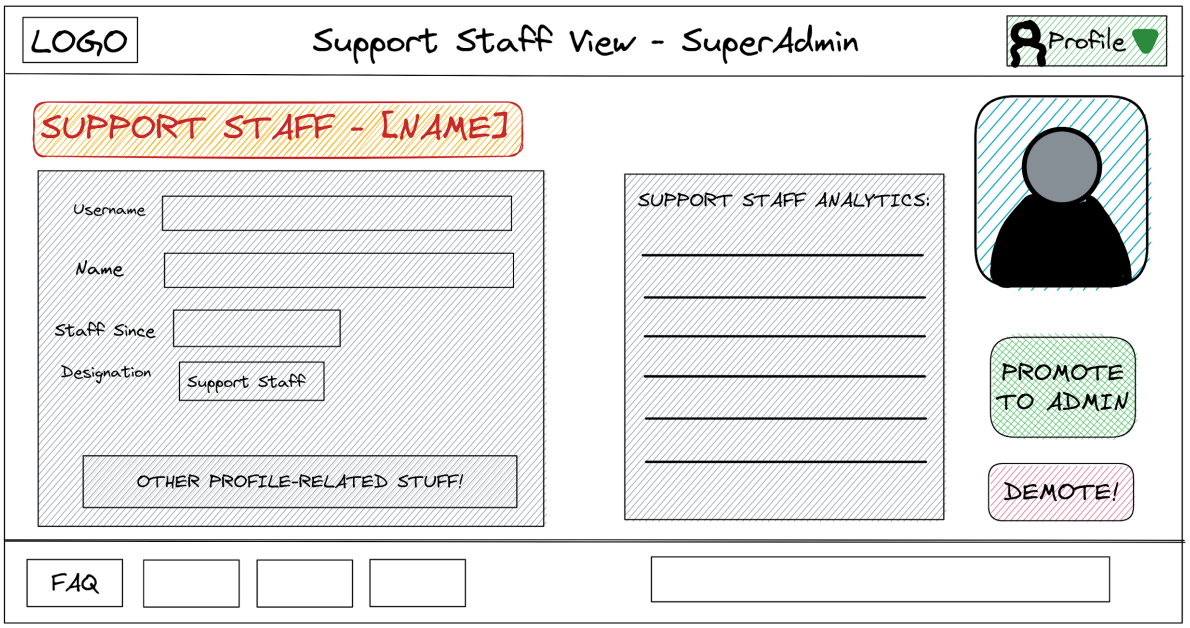
Description: This is the view that will come up when a super-administrator clicks on the "Administrator Actions" button from the homepage. This is the page that lists all the support staff and administrators in a tabular format, as well as some basic metrics related to both those user categories. This page also provides options for searching for support staff/students, and adding new support staff. Furthermore, there is also an option for super-administrators to view analytics related to the ticketing system.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



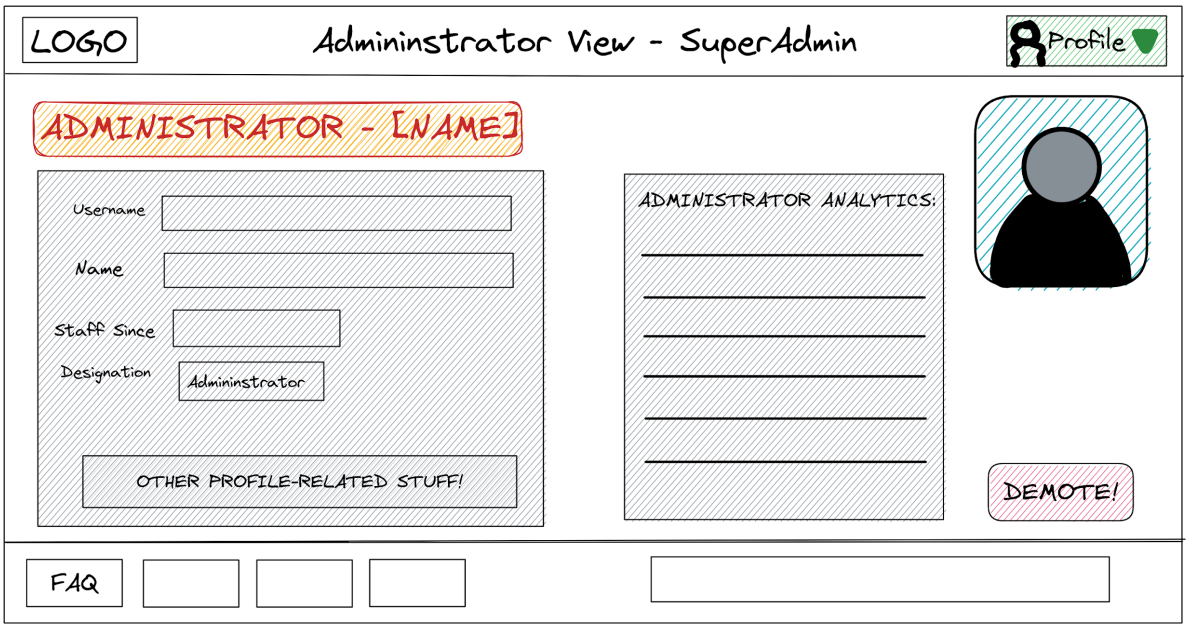
Description: This is the view that will come up when an administrator/super-administrator clicks on the “Analytics Page” button from their respective “Administrator Actions” page. This page displays various analytics associated with the ticketing system, such as number of open tickets, resolved tickets, escalated tickets, etc.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



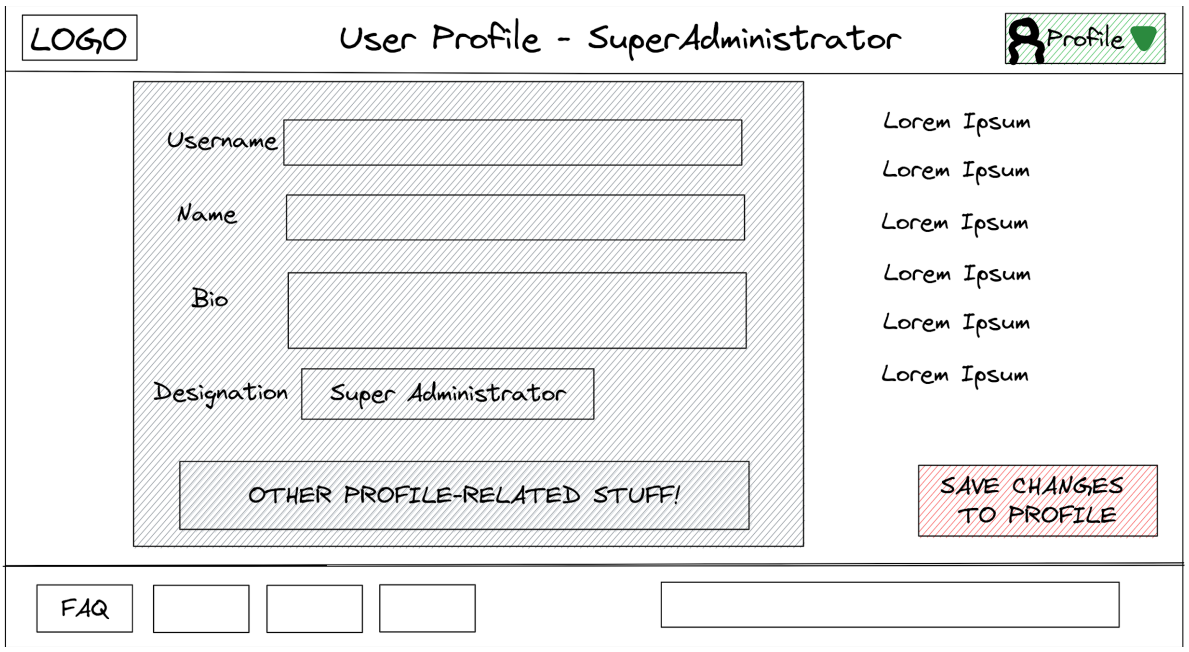
Description: This is the view that will come up when a super-administrator clicks on any support staff from the list of staff from the admin actions page. This page allows the super-administrator to view profile details for a support staff, as well as support staff-specific metrics. This page also allows a super-administrator to promote a support staff to an admin, or demote (i.e remove) a support staff from his/her current position.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Description: This is the view that will come up when a super-administrator clicks on any administrator from the list of admins from the admin actions page. This page allows the super-administrator to view profile details for an administrator, as well as administrator-specific metrics. This page also allows a super-administrator to demote (i.e remove) an administrator from his/her current position.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Description: This is the view that will come up when a super-administrator clicks on the "Profile" drop-down button from his/her homepage. This page allows the super-administrator to edit various attributes associated with his/her profile, such as profile picture, username, name, biography, etc.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

MILESTONE 3 SUBMISSIONS FOLLOW

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Component Design and Class Diagrams

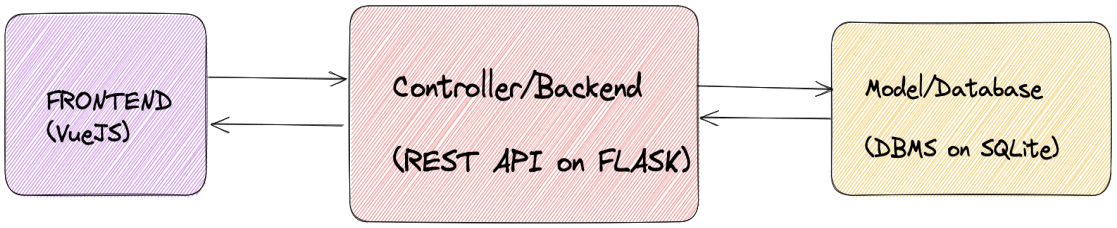
In this document, we have described the different components of our ticketing system, as well as added a few sample class diagrams for the same. (Revision 1)

Adhil Ahmed P M Shums
21F1003972

Afnan Ahmad
21F1003730

Preetodeep Dev
21F1005636

Components



MVC ARCHITECTURE OF THE TICKETING SYSTEM

The overall ticketing system comprises of the following basic components:

- Authentication component: This manages the auth-tokens and authorisation-related methods.

- User registration component: This component deals with the user registration, and acts as a common onboarding point for all types of users.

- Ticket management component: This is the API controller for ticket management. It contains methods for ticket creation, upvoting, closing tickets and more.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

- Support staff management component: API controller for support staff, this component deals with the creation, promotion, and deletion of support staff for the ticketing system.

- Administrator management component: API controller for administrator management. This component deals with the creation, deletion, and management of administrators for the ticketing system.

- Analytics component: API controller that computes and displays various analytics related to the ticketing system.

- User profile management component: This component allows users to edit and add information related to their accounts, e.g changing of profile pictures, names, email IDs, etc.

- Escalation and notification management component: This component manages the notifications and escalation mechanisms for the ticketing system.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Primary Modules for Backend

- Flask: The WSGI web server used for serving the RESTful API.
- SQLAlchemy: ORM module used for the SQLite database.
- Flask-Security: Flask add-on used for handling user registration and authentication.

Frontend Components (Vue.js)

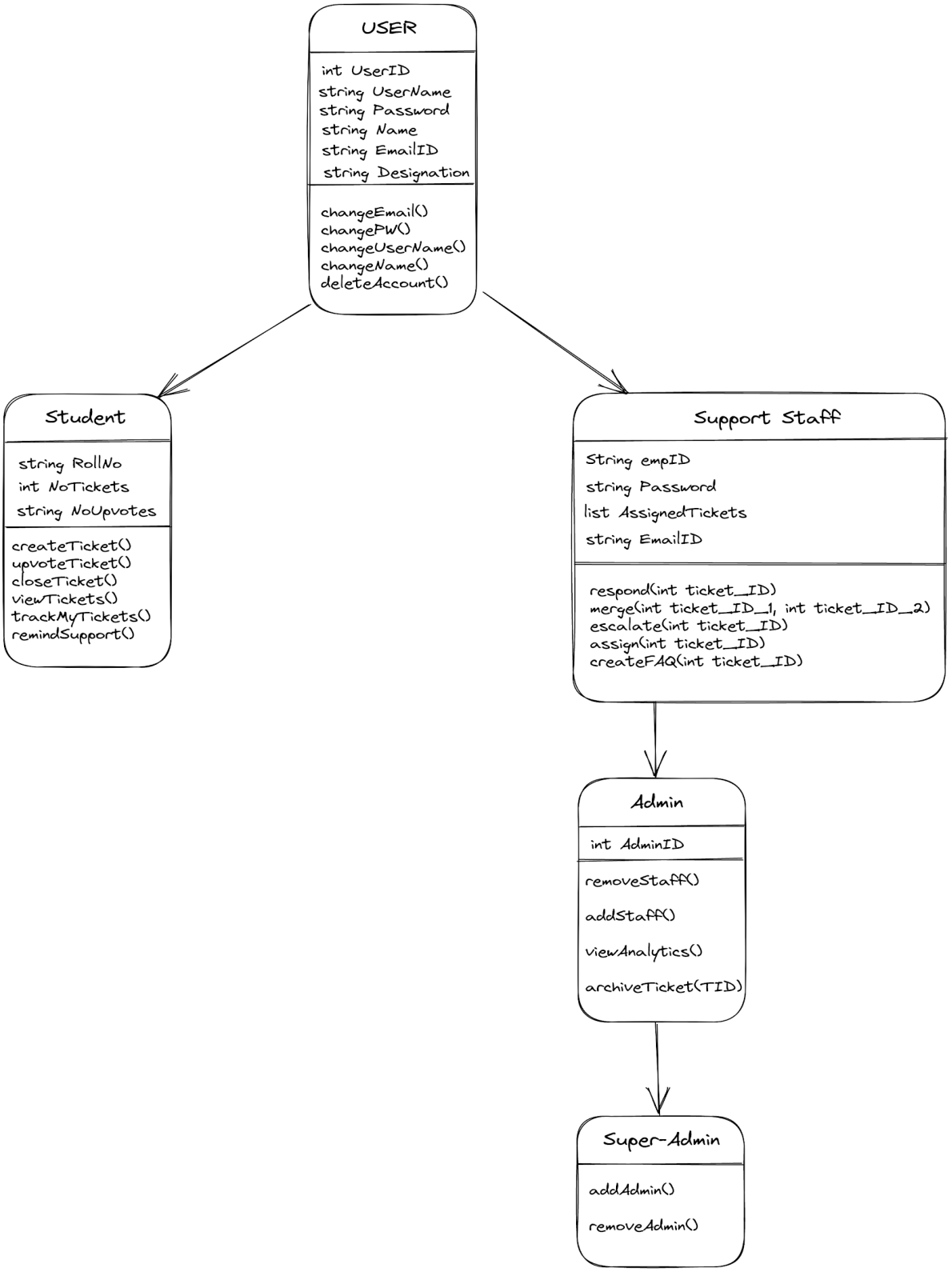
- Vue Components
- Bootstrap for Responsive Design

Database Models

- User
- Student (inherits from User)
- Support Staff (inherits from User)
- Admin (inherits from Support Staff)
- Super Admin (inherits from Admin)

Please refer to the following page for the UML diagram corresponding to the aforementioned database models.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Minutes of Scrum Meetings

In this document, we have added fairly detailed minutes of the scrum meetings that we, as a team, have conducted so far.

Adhil Ahmed P M Shums	**Afnan Ahmad**	**Preetodeep Dev**
21F1003972	21F1003730	21F1005636

Meeting on 17 February 2023 (Friday), 15:00 – 17:00

Agenda: Milestone 1 - Identification of user classes and requirements

Key points discussed in the meeting:

- Potential users for the ticketing system were brainstormed. We came up with the various primary, secondary, and tertiary users for the system, the results of which have already been described in detail in our first milestone submission document.

- The members considered changing a few of the words in the honour code which they felt didn't reflect the true essence of the code. The honour code was modified to their liking, and was kept aside as an agenda item for the next meeting, with the intention of getting approval from the instructors before using that as part of final submission.

- We decided to work asynchronously on a common Google Docs document, preparing user stories for each class of user as identified above.

- Afnan gave the rest of the team members a brief introduction to the git workflow, as well as an introduction to GitHub and the usage of GitHub Desktop.

- All the team members configured the necessary tooling and local workflows on their PCs.

The meeting was concluded, and a decision was made to schedule the next meeting on 19th February 2023 from 18:00 onwards, for the final submission.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Meeting on 19 February 2023 (Sunday), 18:00 – 23:30

Agenda: Final submission of Milestone 1

Key points discussed in the meeting:

- The types of users were revised once again, and a few modifications were made. Amongst the primary users, a choice was made to stick with user, support agent, administrator, and super-administrator as the user hierarchy.
- The honour code document template was finalised and all the team members created their own honour code documents.
- A final Github Desktop workflow demonstration was done by Afnan
- A Google Drive folder was created to host the collaborative documents that the team members are working on as part of the project
- A common template document for milestone submissions was made by Afnan
- The user stories and application users were formatted as per the aforementioned template, and thereafter exported as a PDF.
- The final commit was done by Afnan, with the remaining members being added as co-authors.

The meeting was concluded upon successful submission of milestone 1. A decision was taken to meet informally on 25th February 2023, to revise topics for quiz 1 as well as for the upcoming Milestone 3 submission.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Meeting on 26 February 2023 (Sunday), 19:30 – 23:50

Agenda: Preliminary submission of Milestone 2

Key points discussed in the meeting:

- Initial design wireframes were discussed, with Preetodeep drawing them out by hand
- Wireframes for common views and student views were made on Excalidraw and exported to separate images thereafter
- The exported images of wireframes were imported into the Google Doc template for submission by Afnan
- The descriptions for the wireframe images were made by Adhil and Preetodeep and the documents were thereafter exported to PDFs.
- The commit on the Github repository was done by Adhil, with the remaining members being added as co-authors.

The meeting was concluded upon successful (preliminary) submission of Milestone 1. A decision was taken to meet again on 28th February 2023, to continue with the wireframe views for the support staff and administrators.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Meeting on 28 February 2023 (Tuesday), 19:30 – 21:30

Agenda: Creation of wireframes for support staff as part of Milestone 2

Key points discussed in the meeting:

- Wireframes for support staff views were created on Excalidraw, and detailed deliberations were made while designing them
- Basic ideas for wireframes for admin and super-admin views were discussed
- A decision was taken that Preetodeep would complete the remaining wireframes for admin and super-admin views within 2nd March 2023

The meeting was concluded upon successful preparation of the support staff wireframes. A decision was taken to meet again on 2nd March 2023, to continue with the wireframe views and finally submit all the remaining ones by EOD 2nd March 2023.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Meeting on 2 March 2023 (Thursday), 19:00 – 23:50

Agenda: Final submission of wireframes as part of Milestone 2

Key points discussed in the meeting:

- Wireframes for support staff, administrator, and super-administrator views were revised one last time, and some minor modifications were made

- The wireframes were exported individually as images, and those images were imported into the template document (which was, in turn, exported as a PDF finally).

- The final commit for the remaining wireframes were made by Adhil, with Afnan and Preetodeep as commit co-authors.

- Details about some future feature suggestions, viz. reminder option for students, ticket escalation and merging mechanisms for support staff, automatic ticket assignment having many upvotes/starvation, etc. were discussed

The meeting was concluded upon successful preparation of the support staff wireframes. A decision was taken to meet again on 5th March 2023, to discuss (and submit) the next milestone.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Meeting on 5 March 2023 (Sunday), 15:00 – 23:55

Agenda: Discussion on objectives of Milestone 3, and final submission thereof

Key points discussed in the meeting:

- Project scheduling and scrum meeting timelines were discussed. An initial proposal of meetings on Mondays, Wednesdays, and Fridays (17:00-22:00) was brought about and accepted unanimously. The members decided to have a flexible yet consistent schedule for the scrum meetings.

- For project scheduling tools, all the members decided to keep using Trello and Google Calendar for the time being.

- Formats for minutes of meetings were finalised and the minutes of meetings of earlier meetings were prepared and approved by all present.

- Discussion on UML and class diagrams were made, and relevant tools were used in preparation of the same.

- Separate documents pertaining to the various milestone objectives were prepared, and exported to PDF.

- The final commit to the team Github repository was made by Afnan, with Adhil and Preetodeep as the commit co-authors.

The meeting was concluded upon successful submission of the required milestone documents. A decision was taken to meet again on 8th March 2023, to discuss the API endpoints and other details pertaining to Milestone 4.

Attendees present for all the aforementioned meetings:
Adhil Ahmed P M Shums, Afnan Ahmad, Preetodeep Dev

Meeting location:
All the aforementioned meetings were held virtually on Google Meet

---

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Project Schedule

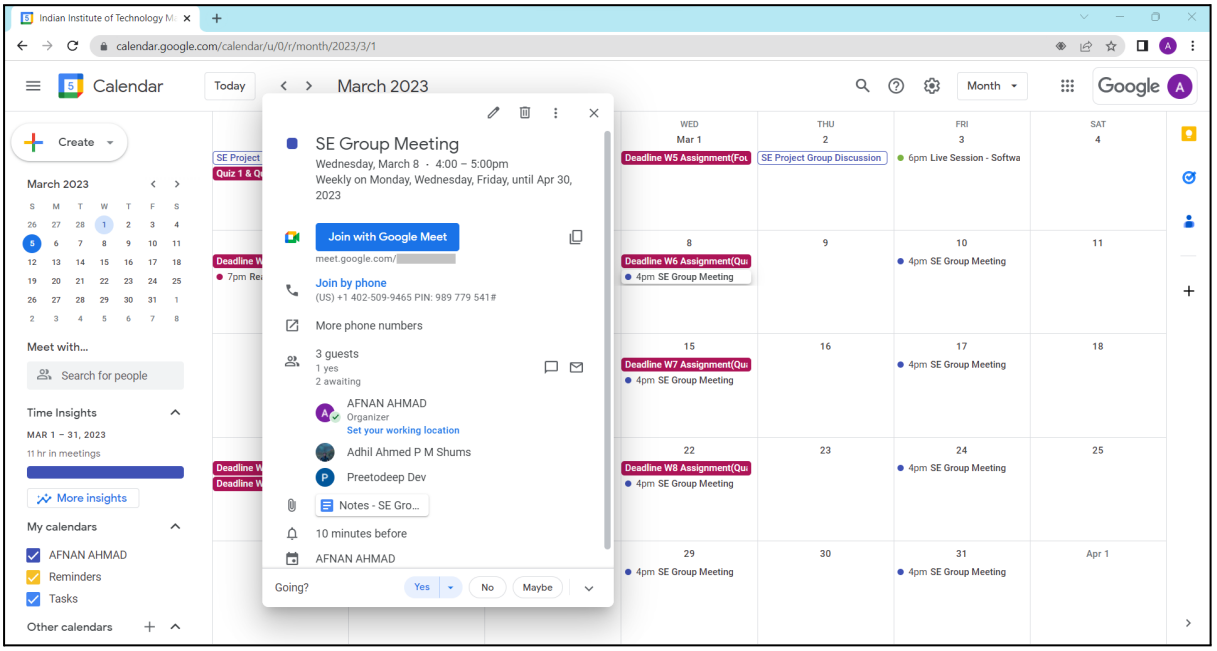
In this document, we have described our project schedule and project scheduling tools, and also provided some screenshots of the same.

Adhil Ahmed P M Shums
21F1003972

Afnan Ahmad
21F1003730

Preetodeep Dev
21F1005636

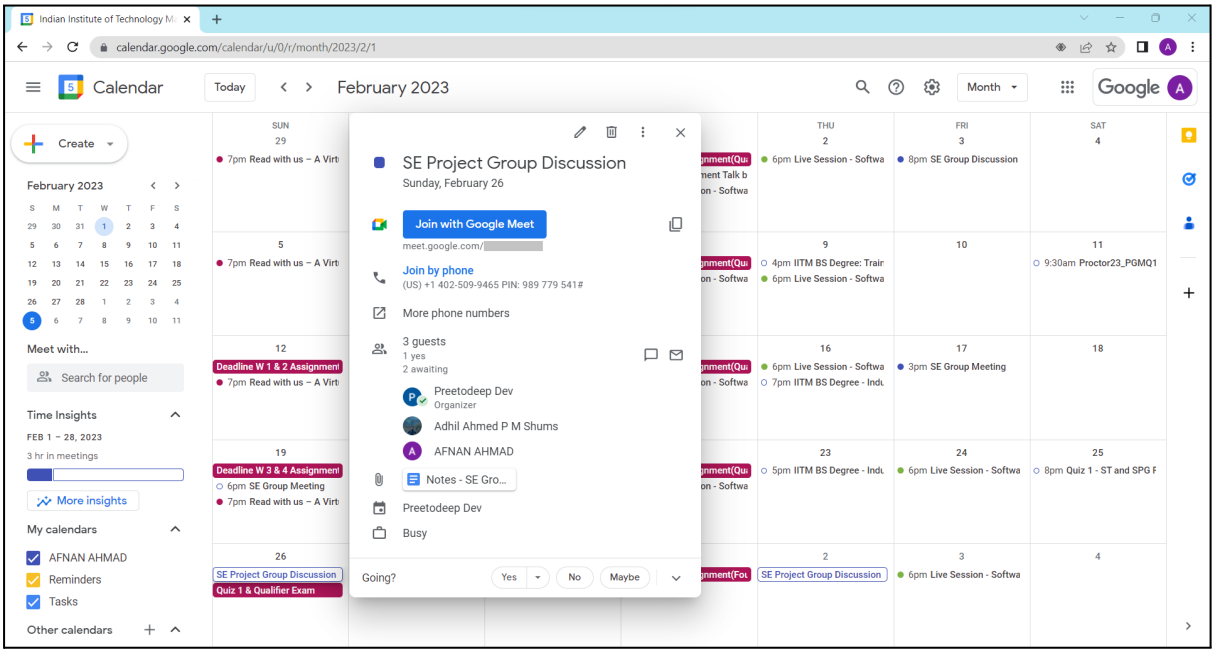
PROJECT SCHEDULING: We have decided upon a tri-weekly schedule for our scrum meetings. We will meet on Mondays, Wednesdays, and Fridays; from 17:00 to 22:00. The timings may be changed slightly owing to clashes with subject-specific live sessions at times. We have decided to have a flexible yet consistent schedule for our scrum meetings. The screenshot below demonstrates our upcoming scrum meetings as per this schedule on Google Calendar.



CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

PROJECT SCHEDULING TOOLS: So far, we have used Trello boards and Google Calendar for all our project scheduling purposes. We plan to continue using the same tools in the future as well, and add more tools as and when such a need arises.

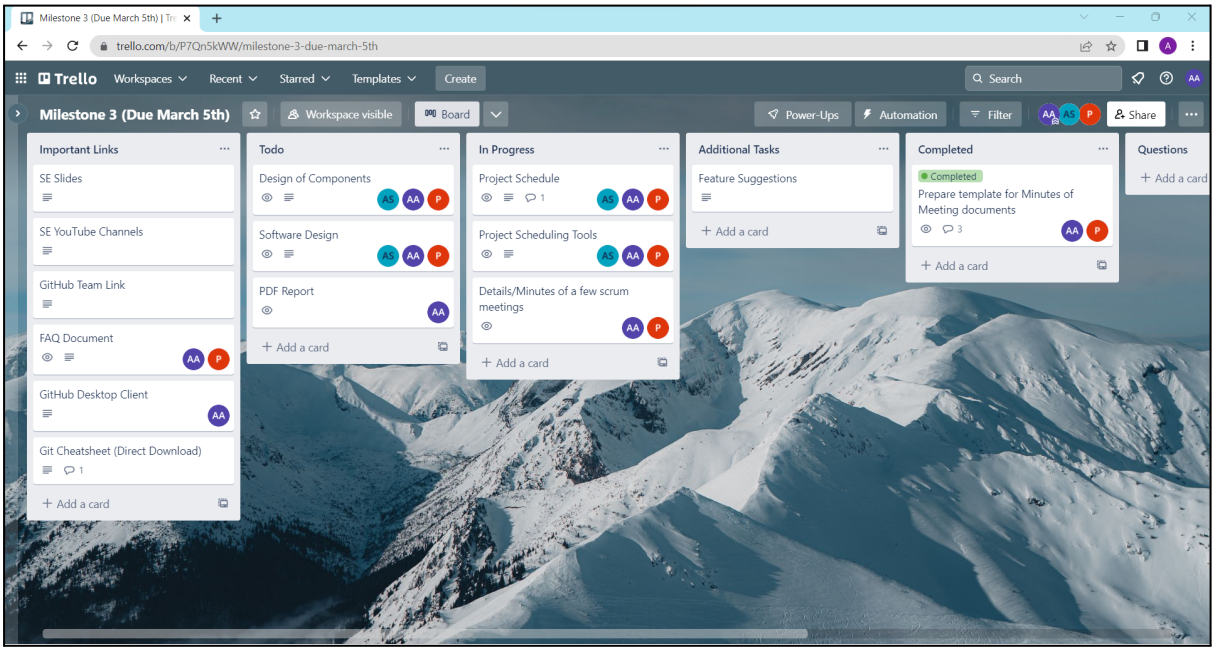
Our Google Calendars, being part of the same Google domain (the IITM organizational domain, in this case), have helped us tremendously in scheduling and conducting meetings in a smooth manner. The following screenshot demonstrates a part of our Google Calendar containing the previous scrum meetings that have taken place.

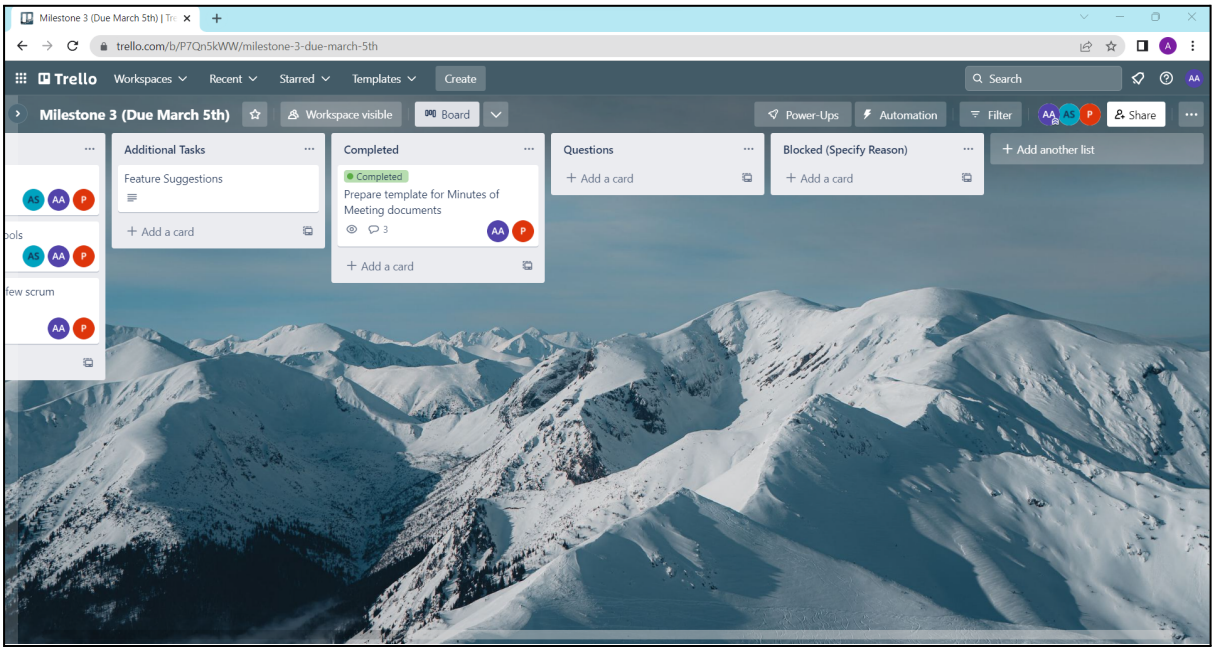


CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Trello boards have allowed us to micromanage our objectives in a time-bound and efficient manner. They have also allowed us to assign objectives to specific members of our team. The following screenshots demonstrate the Trello boards we have used for the various milestones so far.

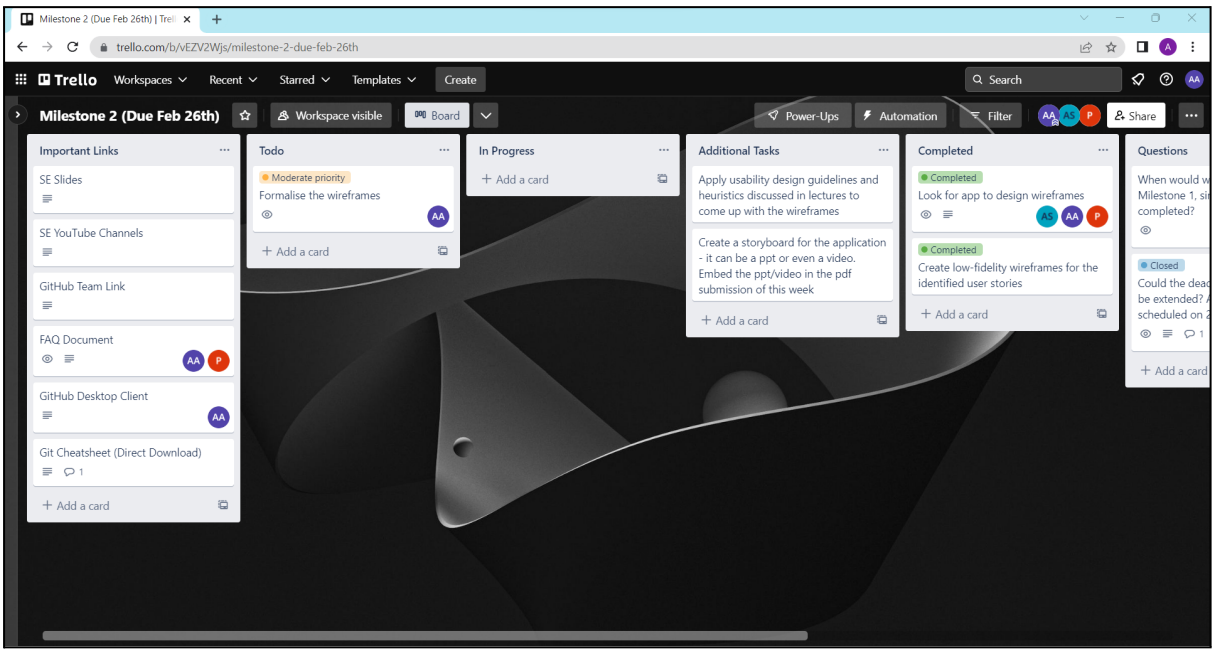
These screenshots are of the Trello board for Milestone 3:

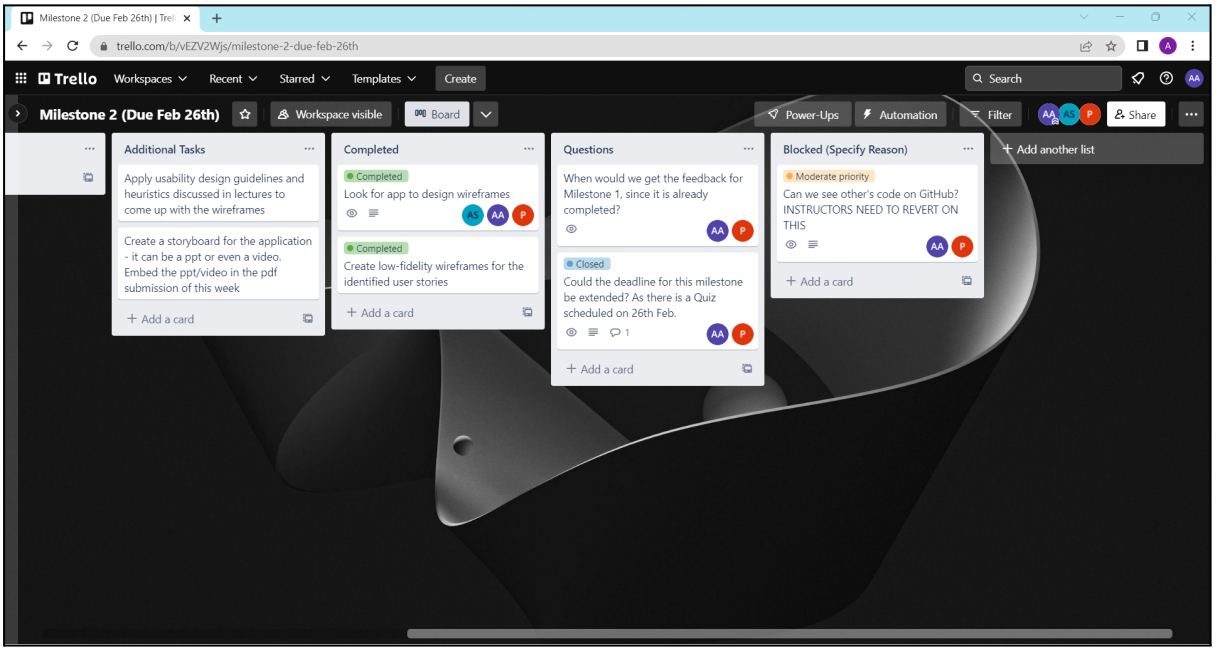




CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

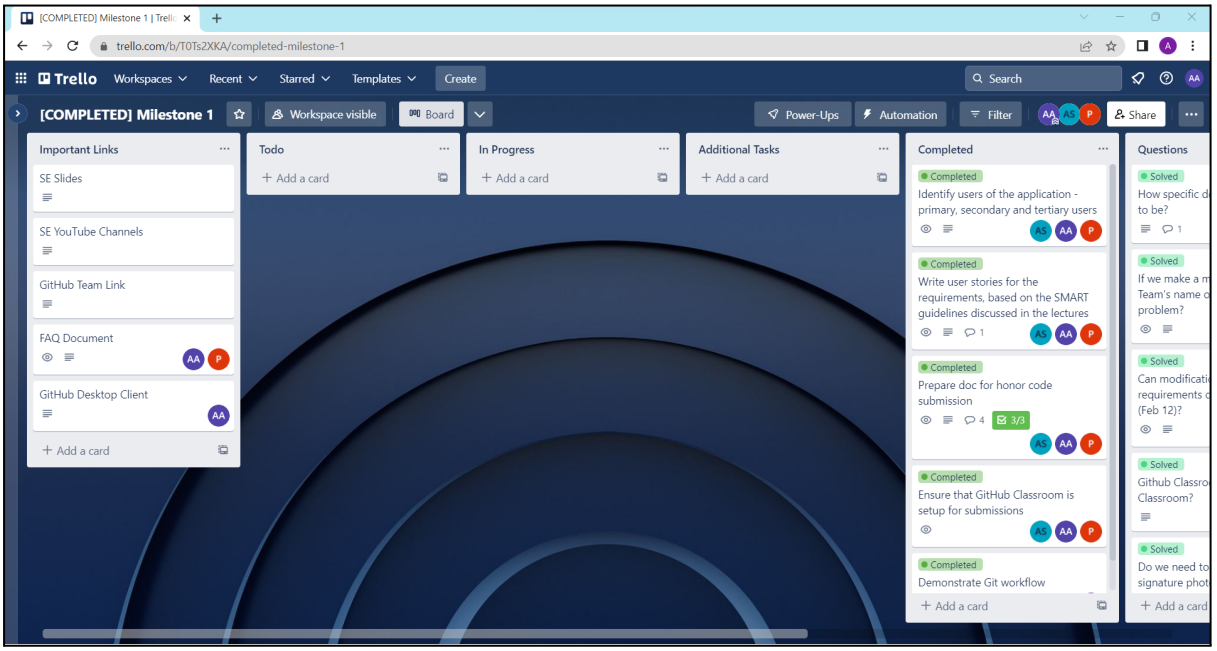
These screenshots are of the Trello board for Milestone 2:

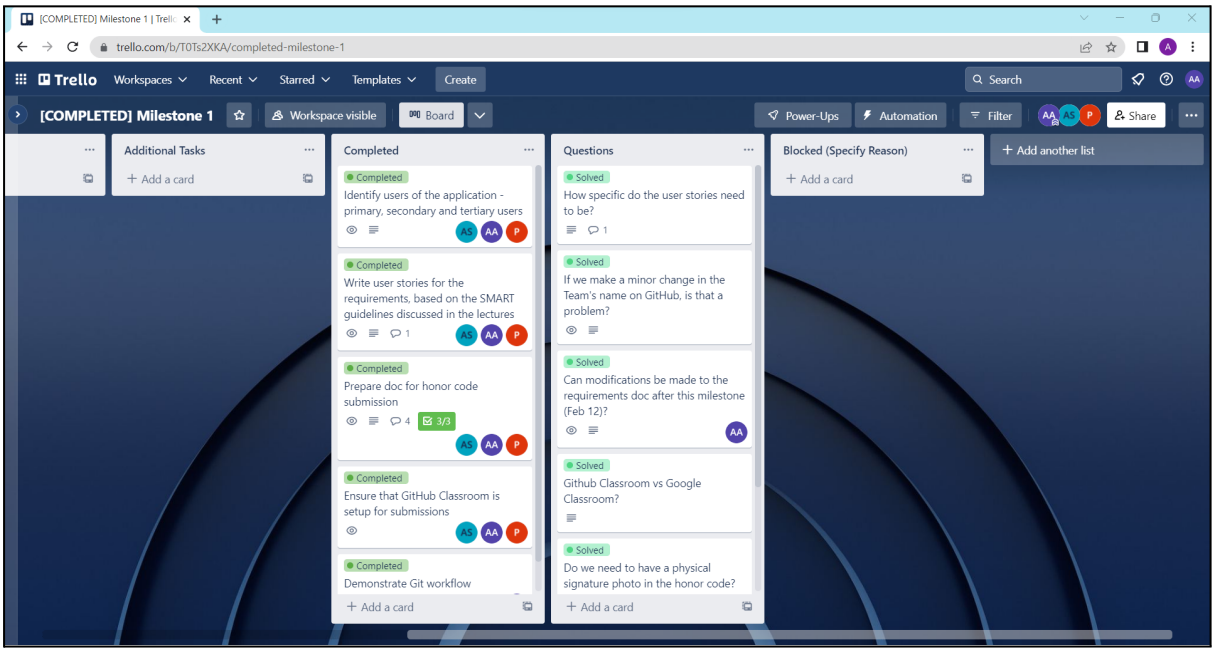




CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

These screenshots are of the Trello board for Milestone 1:





CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

MILESTONE 4 SUBMISSION: YAML file only, no PDF available!

Kindly check the Github repository for the submission.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

MILESTONE 5 SUBMISSIONS FOLLOW

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Description of Test Cases Run

This document contains descriptions of test cases that have been run on the API endpoints in the format that is given as part of the problem statement, and also screenshots of a few unit tests run using PyTest.

Adhil Ahmed P M Shums	**Afnan Ahmad**	**Preetodeep Dev**
21F1003972	21F1003730	21F1005636

---

1. Endpoint: GET */api/v1/tags?tagIDList=all*

Input: The query parameter *"tagIDList*" specifies the tag IDs that we need to query. If "*all*" is specified, then every tag would be returned.

Expected Output: All the tags that are there in the database. For example:

```
{   "status": 200,
    "message": "Request successful",
    "tags": [
        {   "tagID": 1,
            "name": "t12022",
            "description": "Posts created on term 1
(January-April) of 2022"
        },
        {   "tagID": 2,
            "name": "portal",
            "description": "Posts related to the seek portal"
        },
        {   "tagID": 3,
            "name": "oppe",
            "description": "OPPE-related posts"
        }
   ]
}
```

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Actual Output: All the tags that have been created in the test database.


```
{   "status": 200,
    "message": "Request successful",
    "tags": [
        {   "tagID": 1,
            "name": "t12022",
            "description": "Posts created on term 1
(January-April) of 2022"
        },
        {   "tagID": 2,
            "name": "portal",
            "description": "Posts related to the seek portal"
        },
        {   "tagID": 3,
            "name": "oppe",
            "description": "OPPE-related posts"
        }
   ]
}
```


Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

2. Endpoint: GET */api/v1/tags?tagIDList=9,10,11*

Input: "9,10,11" input is passed as query parameter "tagIDList" in the URL

Expected Output: Tags that are there in the database with the given IDs 9,10, and 11.


```
{   "status": 200,
    "message": "Request successful",
    "tags": [
        {   "tagID": 9,
            "name": "quiz1",
            "description": "Posts related to quiz 1"
        },
        {   "tagID": 10,
            "name": "quiz2",
            "description": "Posts related to quiz 2"
        },
        {   "tagID": 11,
            "name": "end-term",
            "description": "Posts related to the end-term
exam"
        }
   ]
}
```


Actual Output: Tags that are there in the database with the given IDs 9,10, and 11.


```
{   "status": 200,
    "message": "Request successful",
    "tags": [
        {   "tagID": 9,
            "name": "quiz1",
            "description": "Posts related to quiz 1"
        },
        {   "tagID": 10,
```


CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

```
                "name": "quiz2",
                "description": "Posts related to quiz 2"
            },
            {   "tagID": 11,
                "name": "end-term",
                "description": "Posts related to the end-term
exam"
            }
        ]
    }
```


Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

3. Endpoint: GET */api/v1/tags?tagIDList=35*

Input: “35” input is passed as query parameter “tagIDList” in the URL. This is designed to test for a non-existent tag being requested from the API.

Expected Output: The API should return a 404 error, saying “no matching tags found”.

```
{
    "status": 404,
    "message": "No matching tags found!"
}
```

Actual Output: The API returns a 404 error, saying “no matching tags found”.

```
{
    "status": 404,
    "message": "No matching tags found!"
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

4. Endpoint: POST */api/v1/tags*

Input: The following input is passed request body:

```
{
    "name": “TestTagCreation”,
    "description": "Test check for tag creation"
}
```

Expected Output: The API should return a 201 code along with the newly created tag as a JSON object.

```
{
    "status": 201,
    "tagID": 14,
    "name": “TestTagCreation”,
    "description": "Test check for tag creation"

}
```

Actual Output: The API returns a 201 code along with the newly created tag as a JSON object.

```
{
    "status": 201,
    "tagID": 14,
    "name": “TestTagCreation”,
    "description": "Test check for tag creation"

}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

5. Endpoint: POST */api/v1/tickets*

Input: The following input is passed to the request body. A valid authorization token is also passed in the request header.

```
{
  "title": "Test Ticket Title",
  "firstMessage": "This is the first message of test ticket",
  "tags": "8,7,4",
  "visibility": "public"
}
```

Expected Output: The API should return a 201 code along with the newly created ticket as a JSON object.

```
{
  "status": 201,
  "message": "Ticket created successfully",
  "ticketID": "1",
  "title": "Sample Title",
  "messageID": "1",
  "firstMessage": "This is the first message",
  "visibility": "public",
  "senderID": "2",
  "senderName": "UnitTester",
  "timestamp": "09/04/2023 22:30"
}
```

Actual Output: The API returns a 201 code along with the newly created ticket as a JSON object.

```
{
  "status": 201,
  "message": "Ticket created successfully",
  "ticketID": "1",
```

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

```
    "title": "Sample Title",
    "messageID": "1",
    "firstMessage": "This is the first message",
    "visibility": "public",
    "senderID": "2",
    "senderName": "UnitTester",
    "timestamp": "09/04/2023 22:30"
  }
```


Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

6. Endpoint: POST */api/v1/tickets*

Input: The following malformed input is passed to the request body. A valid authorization token is also passed in the request header.

```
{
  "title": "Test Ticket Title"
}
```

Expected Output: The API should return a 400 code along with an error message.

```
{
  "status": 400,
  "message": "Malformed request",
}
```

Actual Output: The API returns a 400 code along with an error message.

```
{
  "status": 400,
  "message": "Malformed request",
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

7. Endpoint: POST */api/v1/tickets*

Input: The following input is passed to the request body. No authorization token is passed in the request header.

```
{
  "title": "Test Ticket Title",
  "firstMessage": "This is the first message of test ticket",
  "tags": "8,7,4",
  "visibility": "public"
}
```

Expected Output: The API should return a 403 code along with an error message wrapped in a JSON object

```
{
  "status": 403,
  "message": "No access rights, forbidden!"
}
```

Actual Output: The API returns a 403 code along with the error message wrapped in a JSON object

```
{
  "status": 403,
  "message": "No access rights, forbidden!"
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

8. Endpoint: GET */api/v1/tickets*

Input: The authorization token is passed in the request header.

Expected Output: The API should return a 200 code along with all the public tickets wrapped in a JSON object

```
{
  "status": 200,
  "message": "Request successful",
  "tickets": [
    {
      "ticketID": "1",
      "Votes": 6,
      "title": "Ticket Title 1",
      "Status": "In progress",
      "LastResponseTime": "<time_in_format>"
    },
    {
      "ticketID": "2",
      "Votes": 4,
      "title": "Ticket Title 2",
      "Status": "Open",
      "LastResponseTime": "<time_in_format>"
    },
    {
      "ticketID": "3",
      "Votes": 8 ,
      "title": "Ticket Title 3",
      "Status": "Closed",
      "LastResponseTime": "<time_in_format>"
    }
  ]
}
```

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Actual Output: The API returns a 200 code along with all the public tickets wrapped in a JSON object


```
{
  "status": 200,
  "message": "Request successful",
  "tickets": [
    {
      "ticketID": "4",
      "Votes": 6,
      "title": "Ticket Title 4",
      "Status": "In progress",
      "LastResponseTime": "<time_in_format>"
    },
    {
      "ticketID": "5",
      "Votes": 2,
      "title": "Ticket Title 5",
      "Status": "Open",
      "LastResponseTime": "<time_in_format>"
    },
    {
      "ticketID": "6",
      "Votes": 7,
      "title": "Ticket Title 6",
      "Status": "Closed",
      "LastResponseTime": "<time_in_format>"
    }
  ]
}
```


Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

9. Endpoint: GET */api/v1/mytickets*

Input: The authorization token is passed in the request header.

Expected Output: The API should return a 200 code along with all the tickets created by the logged-in user (as identified via the supplied authorization token) wrapped in a JSON object


```
{
  "status": 200,
  "message": "Request successful",
  "tickets": [
      {
      "ticketID": "4",
      "Votes": 6,
      "title": "TestUser Ticket 1",
      "Status": "In progress",
      "Visibility": "Public",
      "LastResponseTime": "09/04/2023 22:35"
      },
      {
      "ticketID": "7",
      "Votes": 1,
      "title": "TestUser Ticket 2",
      "Status": "In progress",
      "Visibility": "Private",
      "LastResponseTime": "09/04/2023 21:13"
      },
      {
      "ticketID": "8",
      "Votes": 2,
      "title": "TestUser Ticket 3",
      "Status": "In progress",
      "Visibility": "Public",
      "LastResponseTime": "09/04/2023 20:01"
      }
  ]
}
```


CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Actual Output: The API returns a 200 code along with all the tickets created by the logged-in user (as identified via the supplied authorization token) wrapped in a JSON object


```
{
  "status": 200,
  "message": "Request successful",
  "tickets": [
      {
      "ticketID": "4",
      "Votes": 6,
      "title": "TestUser Ticket 1",
      "Status": "In progress",
      "Visibility": "Public",
      "LastResponseTime": "09/04/2023 22:35"
      },
      {
      "ticketID": "7",
      "Votes": 1,
      "title": "TestUser Ticket 2",
      "Status": "In progress",
      "Visibility": "Private",
      "LastResponseTime": "09/04/2023 21:13"
      },
      {
      "ticketID": "8",
      "Votes": 2,
      "title": "TestUser Ticket 3",
      "Status": "In progress",
      "Visibility": "Public",
      "LastResponseTime": "09/04/2023 20:01"
      }
  ]
}
```


Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

10. Endpoint: PUT */api/v1/tickets/<int:ticket_id:>*

Input: The following input is passed to the request body. A valid authorization token is also passed in the request header.

```
{
  "title": "Sample Title",
  "visibility": "public"
}
```

Expected Output: The API should return a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Ticket modified successfully"
}
```

Actual Output: The API returns a 200 code along with the success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Ticket modified successfully"
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

11. Endpoint: DELETE */api/v1/tickets/<int:ticket_id:*

Input: A valid authorization token is passed in the request header.

Expected Output: The API should return a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Ticket deleted successfully"
}
```

Actual Output: The API returns a 200 code along with the success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Ticket deleted successfully"
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

12. Endpoint: GET */api/v1/messages?start=1?count=3?ticket_id=4*

Input: The query parameter *"start"* specifies the index of the fist message to be loaded, the query parameter *"count"* specifies the number of messages to be loaded and the query parameter *"ticket_id"* specifies which ticket's messages we need to query.

Expected Output: The API should return a 200 code along with the requested messages


```
{
  "status": 200,
  "message": "Request successful",
  "ticketID": "4",
  "LastResponseTime": "09/04/2023 13:01"
,
  "messages": [
    {
      "messageID": "1",
      "text": "Message Text 1",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 12:59"
    },
    {
      "messageID": "2",
      "text": "Message Text 2",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 13:00"
    },
    {
      "messageID": "3",
      "text": "Message Text 3",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 13:01"
    }
  ]
```


CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

```
}
```

Actual Output: The API returns a 200 code along with the requested messages wrapped in a JSON object


```
{
  "status": 200,
  "message": "Request successful",
  "ticketID": "4",
  "LastResponseTime": "09/04/2023 13:01"
,
  "messages": [
    {
      "messageID": "1",
      "text": "Message Text 1",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 12:59"
    },
    {
      "messageID": "2",
      "text": "Message Text 2",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 13:00"
    },
    {
      "messageID": "3",
      "text": "Message Text 3",
      "senderID": "6",
      "senderName": "papa_delta",
      "timestamp": "09/04/2023 13:01"
    }
  ]
}
```


Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

13. Endpoint: DELETE */api/v1/messages/<int:message_id>*

Input: A valid authorization token is passed in the request header. Message ID 4 is passed in the URL for deletion.

Expected Output: The API should return a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Article deleted successfully",
  "articleID": "4"
}
```

Actual Output: The API should returns a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 200,
  "message": "Article deleted successfully",
  "articleID": "4"
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

14. Endpoint: POST */api/v1/articles*

Input: A valid authorization token is passed in the request header. Article contents are passed in the request body.

```
{
  "title": "Test Article Title",
  "content": "Test Article Content",
  "visibility": "public"
}
```

Expected Output: The API should return a 201 code along with a success message and the newly created article object wrapped in a JSON object.

```
{
  "status": 201,
  "content": "Article created successfully",
  "articleID": "2",
  "title": "Test Article Title",
  "content": "Test Article Content",
  "visibility": "public"
}
```

Actual Output: The API returns a 201 code along with a success message and the newly created article object wrapped in a JSON object.

```
{
  "status": 201,
  "content": "Article created successfully",
  "articleID": "2",
  "title": "Test Article Title",
  "content": "Test Article Content",
  "visibility": "public"
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

15. Endpoint: DELETE */api/v1/admin/articles/<int:article_id:>*

Input: A valid authorization token is passed in the request header.

Expected Output: The API should return a 200 code along with a success message and the article ID wrapped in a JSON object

```
{
  "status": 200,
  "message": "Article deleted successfully",
  "articleID": "1"
}
```

Actual Output: The API returns a 200 code along with the success message and the article ID wrapped in a JSON object

```
{
  "status": 200,
  "message": "Article deleted successfully",
  "articleID": "1"
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

16. Endpoint: POST */api/v1/admin/comments*

Input: The following input is passed to the request body. A valid authorization token is also passed in the request header.

```
{
  "article_id": "1",
  "content": "Comment content 1",
  "hidden": "False"
}
```

Expected Output: The API should return a 200 code along with a success message wrapped in a JSON object

```
{
  "status": 201,
  "message": "Comment created successfully",
  "articleID": "1",
  "commentID": "1",
  "content": "Comment content 1"
}
```

Actual Output: The API returns a 200 code along with the success message wrapped in a JSON object

```
{
  "status": 201,
  "message": "Comment created successfully",
  "articleID": "1",
  "commentID": "1",
  "content": "Comment content 1"
}
```

Result: SUCCESS

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

A FEW TEST CASES RUN USING PYTEST - ALL PASSED

```
(venv) afnan@AFNAN-PC:/mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy_4apd/src$ pytest test_api.py
========================== test session starts ==========================
platform linux -- Python 3.10.6, pytest-7.3.0, pluggy-1.0.0
rootdir: /mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy_4apd/src
collected 3 items

test_api.py ...                                                    [100%]

=================================================== 3 passed in 4.36s ===================================================
(venv) afnan@AFNAN-PC:/mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy_4apd/src$
```

```
(venv) afnan@AFNAN-PC:/mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy
_4apd/src$ pytest test_api.py
========================= test session starts =========================
platform linux -- Python 3.10.6, pytest-7.3.0, pluggy-1.0.0
rootdir: /mnt/c/Users/Afnan/Documents/GitHub/soft-engg-project-jan-2023-group-4-synergy_4apd/src
collected 3 items

test_api.py ...                                                  [100%]

========================== 3 passed in 4.02s ==========================
(venv) afnan@AFNAN-PC:/mnt/c/Users/Afnan/Documents/GitHub/soft-engg-pro
ject-jan-2023-group-4-synergy_4apd/src$
```

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Technologies and Tools Used: Overall Development Process

- Git and Github: As asked in the problem statement, git was used as the very basis for the entire project. Github repositories were used right from the very beginning, and milestone submissions were made therein. The steps involved in git usage became second-nature to us by the end of the project.

- Postman: Postman was the go-to tool that we used for testing out API endpoints while building them. The easy-to-use and feature-rich nature of Postman meant that we didn't have to face much trouble in testing and debugging our APIs.

- Trello: We used Trello boards for planning our sprints, adding feature requests, issue tracking and assigning, and managing and scheduling our overall workflow. Trello notifications were quite useful for our work.

- Visual Studio Code: It was the IDE of choice for all of us. It is quite powerful and its extensions were very useful.

- Swagger OAS Editor: We've used Swagger to create and edit our API specification YAML document.

- Firefox, Safari, and Chrome: We used these browsers and their inbuilt developer consoles and extensions while coding

- DB Browser for SQLite: It was used to manage the backend SQLite database during the development and testing process.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Technologies and Tools Used: Frontend

- Flask Jinja2: The frontend was built using Flask - Jinja2 templating. Features of Jinja2 such as dynamic web page generation, extensions, template inheritance, etc. have been used extensively.

- Bootstrap: Bootstrap was used throughout the entire app, for styling purposes. Various Bootstrap elements, such as buttons, tables, cards, etc. have been used.

- HTML/CSS/JS: The basic tech stack for web pages

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Technologies and Tools Used: Backend

- Python: The programming language that brought it all together. Almost 100% of the submissions as part of our repository is Python code. Python-based frameworks like Flask were the basis for the application. Various Python libraries, such as `uuid`, `datetime`, `humanize`, etc. were used.

- Flask: As mentioned in the problem statement, the Flask web framework was used for the backend server. Various Flask sub-components, such as Flask-Security-Too (for the authentication mechanisms), Flask-RESTful (for the API), Flask-CORS (for enabling cross-origin requests), etc. have been used.

- SQLite: It was used as the database for storing the app data at the backend. Various tables corresponding to the users, tickets, articles (FAQs), notifications, etc. were created

- SQLAlchemy: Flask-SQLAlchemy was used as the ORM to enable the interactions between the app at the backend Flask API and the SQLite database. The various data models were made using the appropriate classes in SQLAlchemy.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Deviations from initial milestone submissions

1. **Slight deviations from the initial wireframes (Milestone 2)**: There have been some minor deviations from the initial wireframes that have been submitted, such as placement of buttons, slight changes in the UI elements, layout, colours, etc.

2. **Changes to YAML (Milestone 4)**: We've added API endpoints even after our milestone submission. Endpoints for message creation, managing tags, managing notifications, managing users, etc. have been modified/added after the initial submission. Hence, the YAML file has been updated to reflect the changes.

3. **Change from VueJS frontend to Flask-Jinja2 frontend (Milestone 3)**: Frontend implementation was changed to Flask templates from VueJS. Our original plan was to have a separate backend API and a separate frontend. Communication between them would happen through VueJS. However, upon beginning work on the frontend, we realized that VueJS made the coding a lot more complicated than it needed to be for the purposes of this project. Thus, we made the switch to simple Flask templates which made the frontend implementation much more simpler.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Where is our application hosted?

Our application (i.e the backend server itself) isn't hosted on the cloud. It runs on localhost, on port 8000.

The codebase itself is hosted on Github, though.

Steps to run our application

1. Download the application from the Github project repository, either as a ZIP file or by cloning the repository.

2. Locate the `src` folder within the repository, and open it using a terminal.

3. Run `pip install -r requirements.txt` in the terminal. This will install all the necessary requirements for the app.

4. Run python main.py in the terminal. The app is now running on `http://localhost:8000`.

5. Visit `http://localhost:8000` using a web browser. You will now be able to use the app.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

Challenges faced - Lessons for the future

1. **Selecting the appropriate data structure for the application given the various contexts** was a major chunk of the debugging/testing that had to be done. Often, we've seen that it is easier to store data in a particular structure, but easier to read from another structure. **The tradeoff has to be thought about carefully**.

2. **The strict milestone deadlines** meant that we were mainly hopping from milestone deadline to milestone deadline, without actually getting much time to think about the entire project in an organic manner.

3. **Initial coordination** and getting into a proper workflow was slightly more difficult than we initially thought.

4. **Prior unfamiliarity with git in practical usage** was a major hurdle that we had to overcome. We always found ourselves hesitating while trying to resolve merge conflicts and approve pull requests initially.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

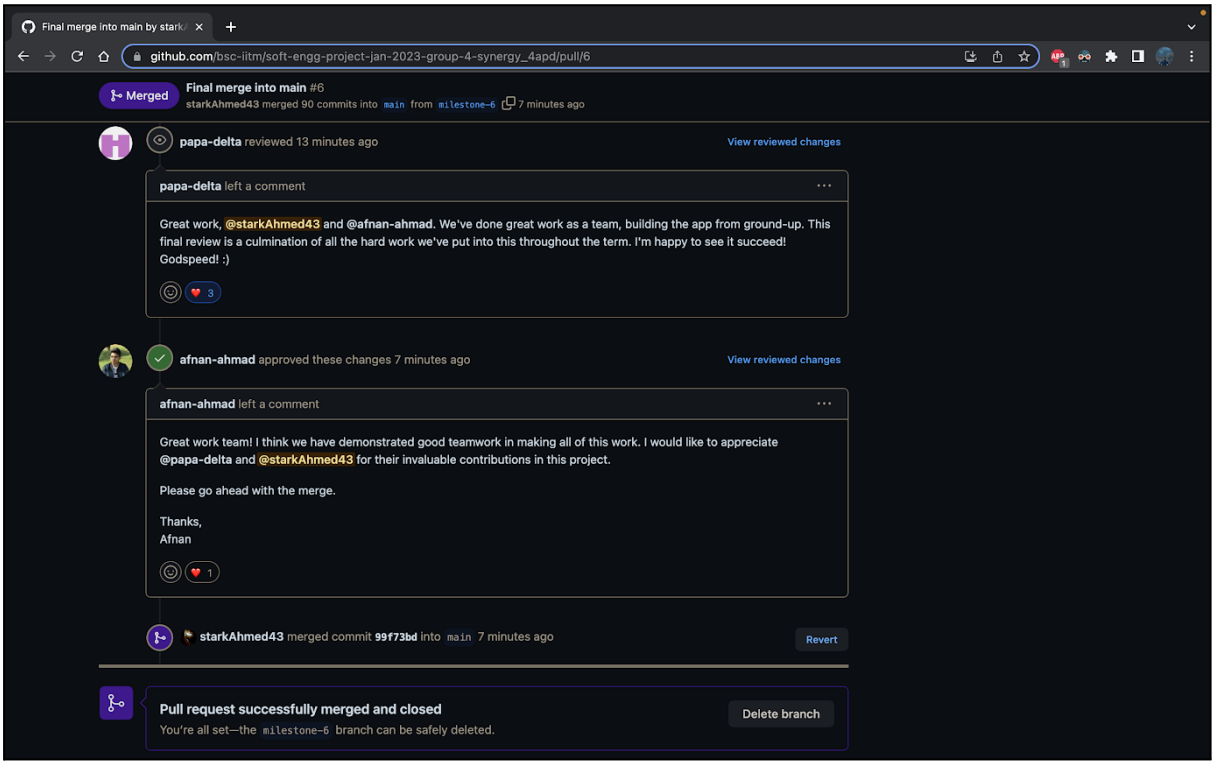
Beyond the project - Next steps

- Addition of the email and chat/SMS notification features, over and above the notifications implemented in the API/database
- Automatic support staff assignment to tickets with large numbers of upvotes or to tickets facing starvation
- Ticket merging option for similar tickets, accessible to support staff
- Moderation controls for support staff and administrators
- Student-initiated reminder/alert in case their ticket hasn't been responded to in a long time
- Introduce localization features and allow users to set their locales
- Implementation of a VueJS frontend. The app should ideally be an SPA.
- Creation of a mobile app based on the existing API, implementation of a CI/CD workflow, and a more comprehensive test suite
- Improving accessibility features of the app, such as screen reader support
- Caching within the overall stack - the app should work well even at scale

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

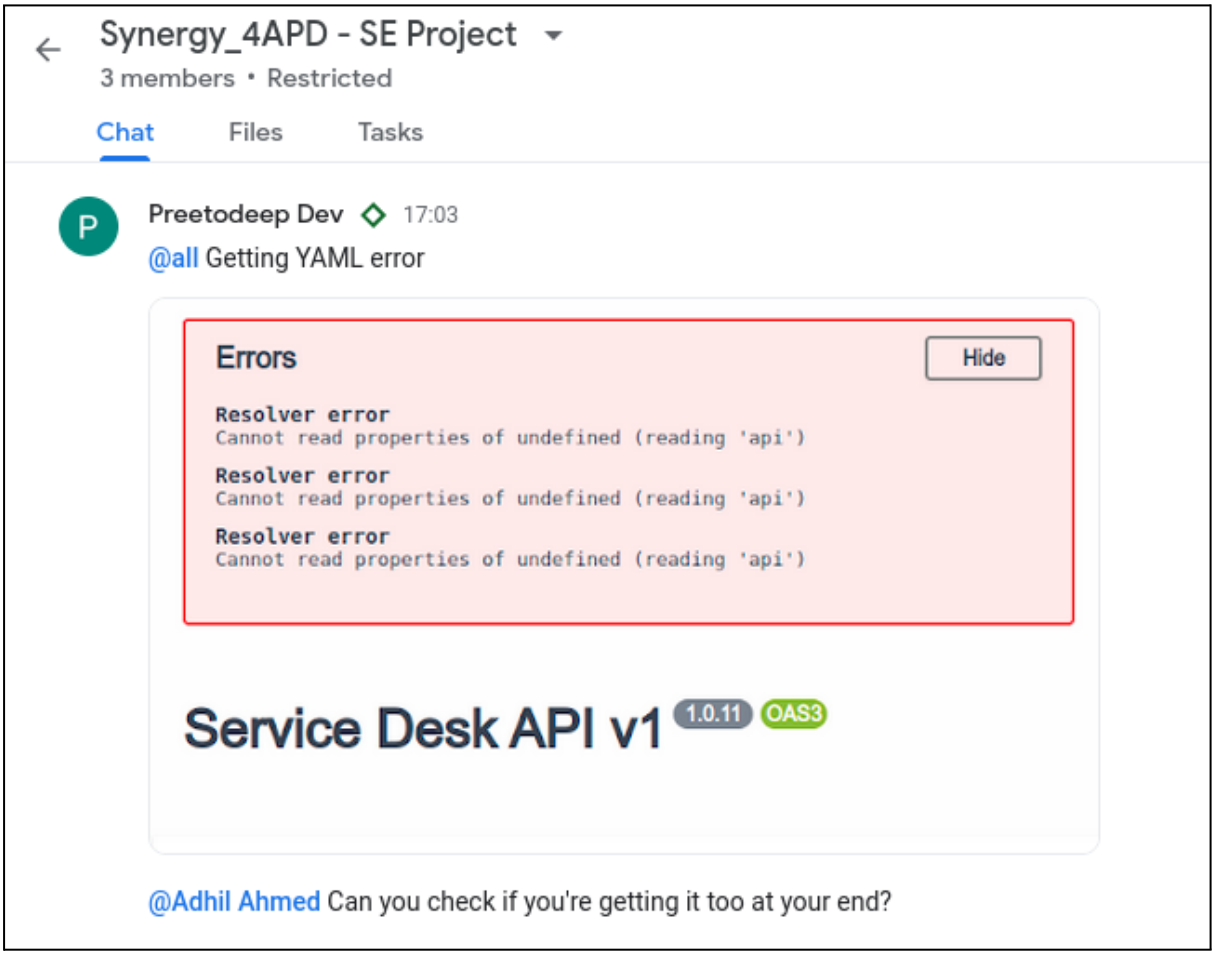
Code Review & Issue Tracking Screenshots

The following screenshots demonstrate the various code review, issue reporting, and issue tracking actions that we've undertaken. Specifically, we have used tools such as Trello, Google Spaces, Github Code Review, and Google Meet for our needs in this regard.



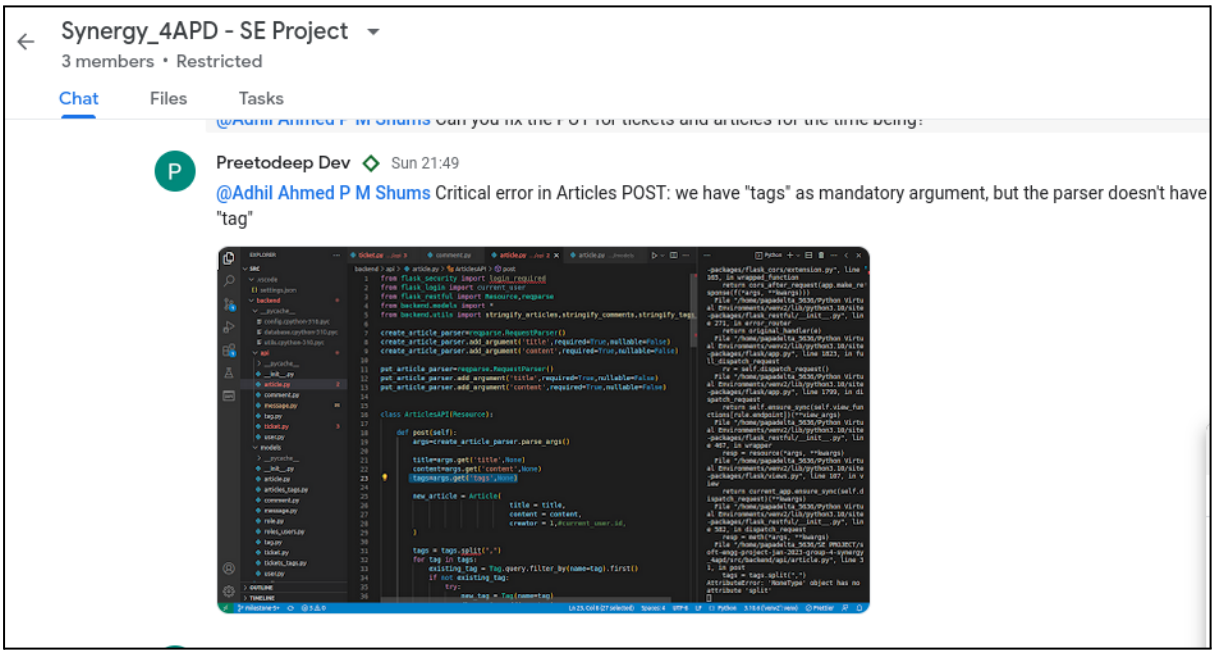
Final pull request reviewed and merged into the main branch.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

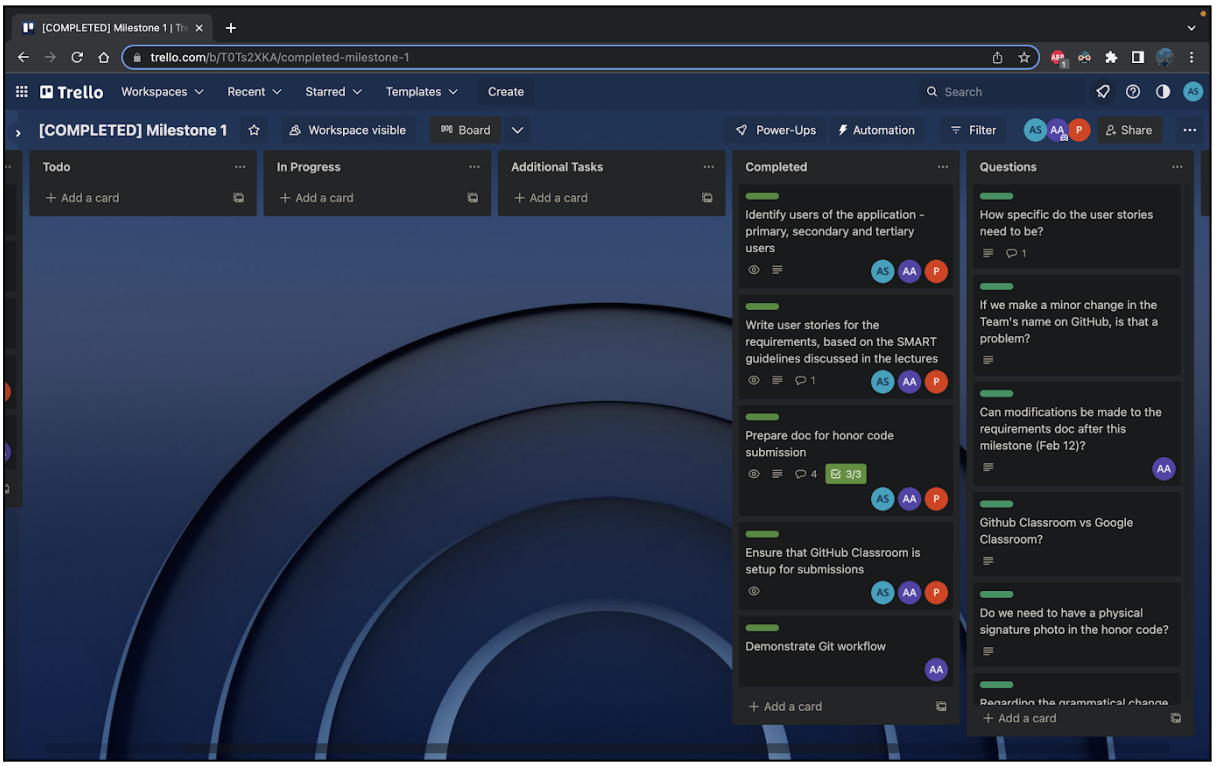


An issue being discussed on Google Space

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

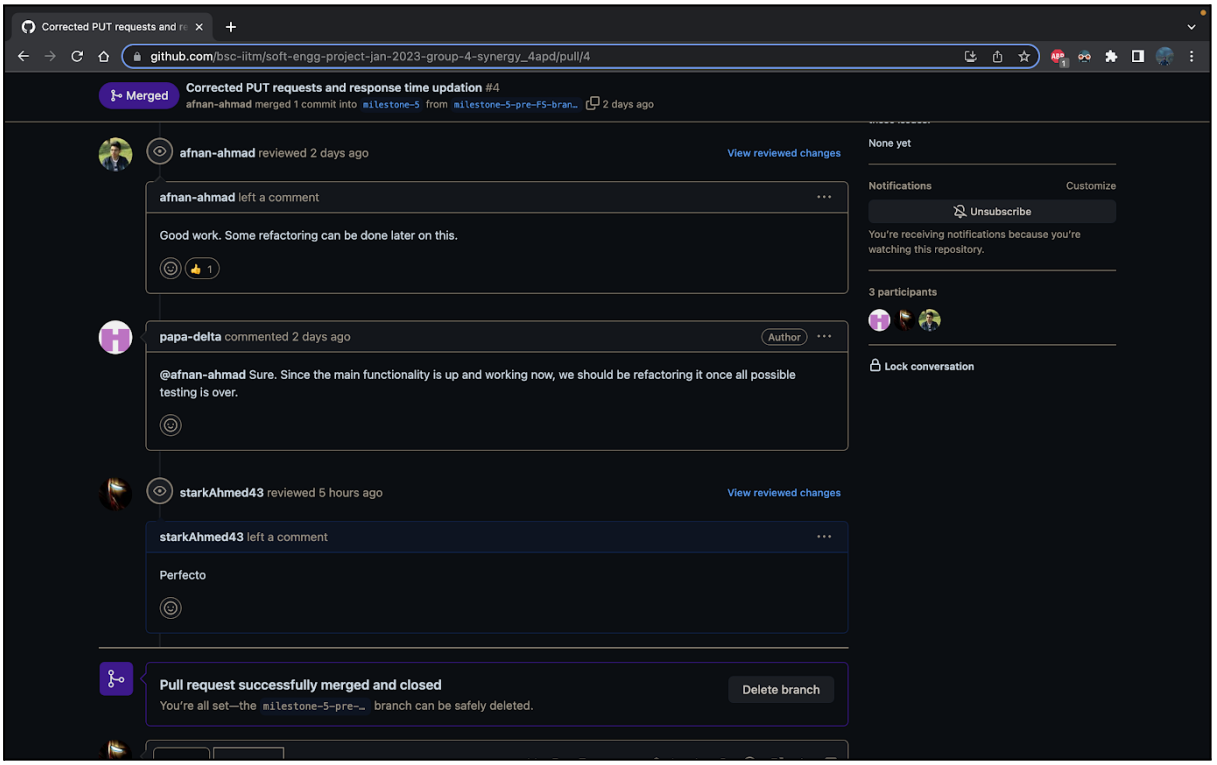


An issue being discussed on Google Space



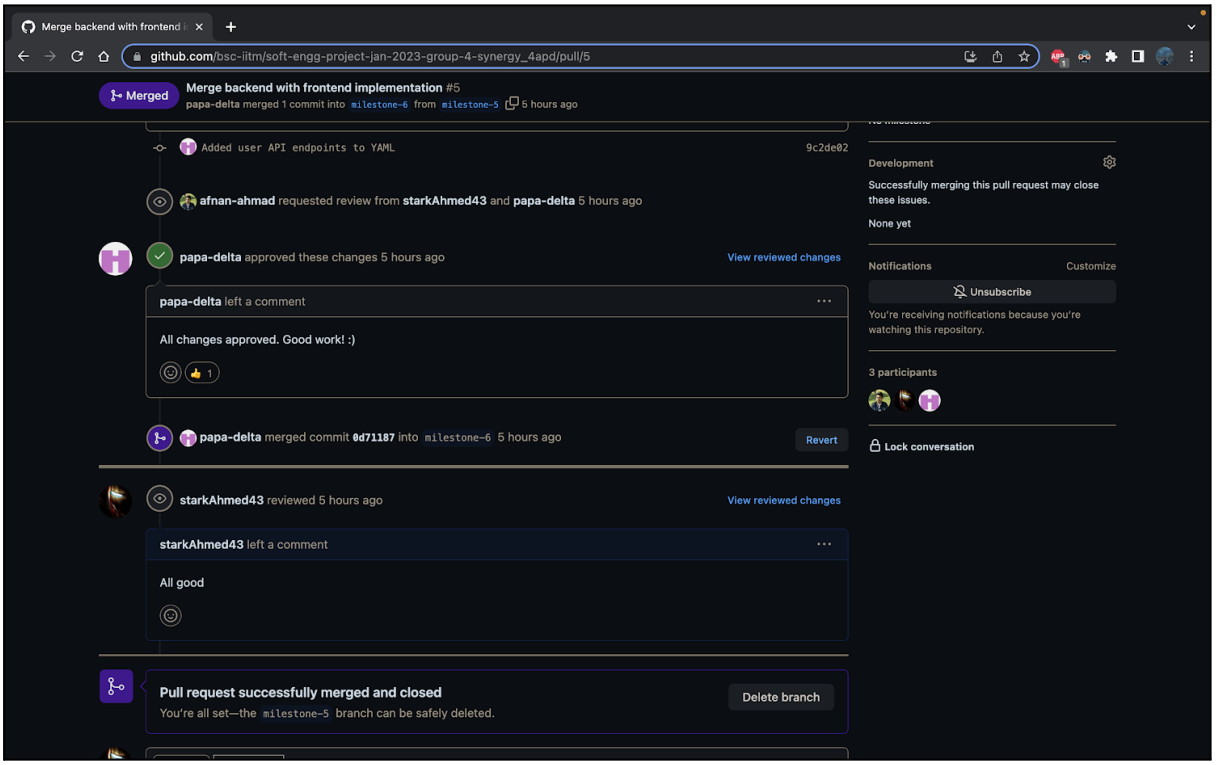
Trello board for tracking project tasks

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.

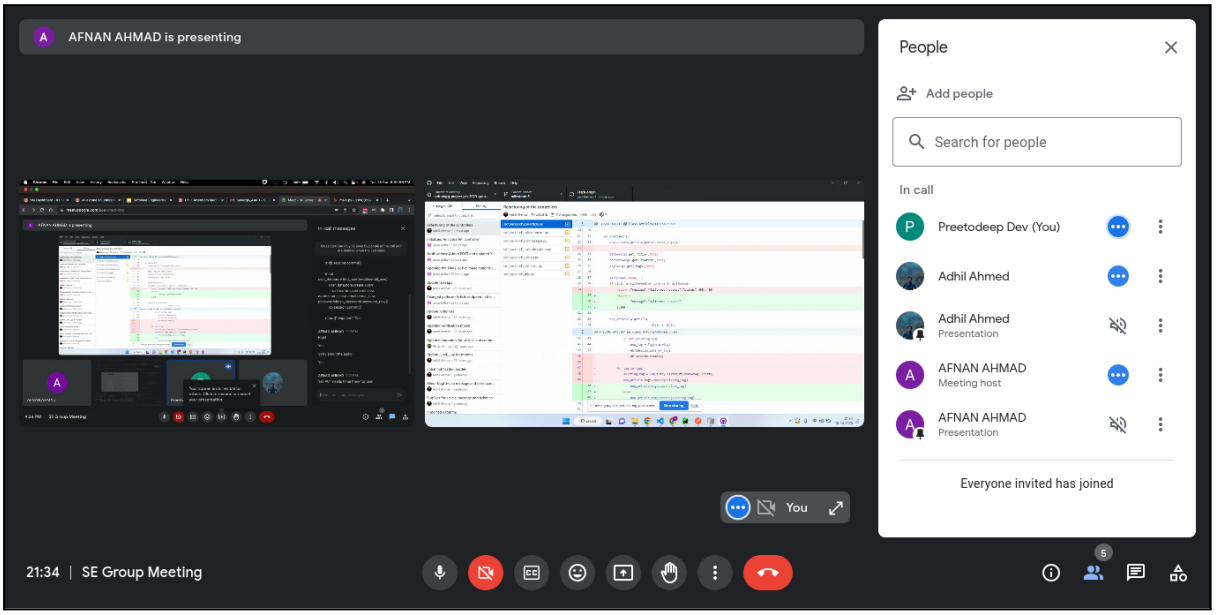


Discussion on a pull request

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.



Discussion on a pull request



One of the Scrum meetings being held on Google Meet.

CONFIDENTIAL: This document is part of a group project for the Software Engineering course. It is not permissible to share this document without an explicit permission from the authors.