

Smart Support

Final Project Report

Submitted by :

NAVIN KUMAR SINGH

UDAY RAVI PATIL

KRISHU GUPTA

User Requirements & User Stories

User Requirements

Our team identified the following users for the Support Ticket App:

- **Primary Users:** Students
- **Secondary Users:** Support team members
- **Tertiary Users:** Administration members

User Stories (Primary Users)

Student

- *As a student, I want to be able to create a new support ticket so that I can get help with my queries and concerns.*
- *As a student, I want to be able to search for similar tickets before creating a new one so that I don't have to create a duplicate ticket.*
- *As a student, I want to be able to upvote an existing ticket if it's similar to my concern or query so that the support team can prioritize the most popular concerns.*
- *As a student, I want to be able to see the status of my ticket and updates made by the support team so that I can stay updated on the progress of my concern or query.*
- *As a student, I want to be able to see a history of all my tickets and their status so that I can keep track of my past concerns and queries.*
- *As a student, I want to be notified when the support team updates my ticket or when the status of my ticket changes so that I can stay updated on the progress of my concern or query.*

User Stories (Secondary Users)

Support team

- *As a support team member, I want to be able to see a list of all the support tickets sorted by upvotes and date so that I can and respond to them in a timely manner.*
- *As a support team member, I want to be able to update the status (open, resolved) of a support ticket so that the student knows the progress of their concern or query.*
- *As a support team member, I want to be able to see the history of a support ticket so that I can understand the student's previous concerns and queries.*

User Stories (Tertiary Users)

Administration

- *As an administrator, I want to be able to see a list of all the support tickets and their status so that I can monitor the support activities of the program.*
- *As an administrator, I want to be able to see the history of support tickets sorted by upvotes and date so that I can understand the concerns and queries of the students.*
- *As an administrator, I want to be able to add new tags so that I can control the allowed types of tickets.*
- *As an administrator, I want to be able to assign different tags to different support team members so that they will be able to see only the tickets with relevant tags assigned to them.*

User Stories (Tertiary Users continued...)

- *As an administrator, I want to be able to see the support tickets categorized by tags so that I can monitor the types of concerns and queries the students are facing.*
- *As an administrator, I want to be able to send a notification when a support ticket has been resolved, so that I am make aware the concerned users*
- *As an administrator, I want to be able to add the support query and response to the FAQ section categorised by tags so that an updated FAQ will be readily available to students.*
- *As an administrator, I want to be able to allow users to enrol as students, support staff and admin, so that new users can access the platform*

Storyboard

A student is facing a problem with her course and wants to contact the support team. She accesses the support ticketing system through the university website.

Smart Support

Login

e-mail

password

LOGIN

Not a member? Register

Smart Support

Register

full name

e-mail

password

re-enter password

REGISTER

Already a member? Login

The student is presented with a login page where she needs to enter her credentials to access the support ticketing system. She can also click on **Register** link if she is new to the system.

Smart Support

HOME

PROFILE

MY TICKETS

SEARCH

LOGOUT

My Tickets

Subject	Votes	Date	Status
Ticket 1 subject	16	2023/02/01	Open
Ticket 2 subject	13	2023/02/01	Resolved
Ticket 3 subject	06	2023/02/01	Resolved
Ticket 4 subject	01	2023/02/01	Open

Raise a new support ticket

Subject

Tags

body

POST

After logging in, the student is directed to the home page of the support ticketing system where she can create a new ticket or view her existing tickets.

Smart Support

HOME

PROFILE

MY TICKETS

SEARCH

LOGOUT

My Tickets

Subject	Votes	Date	Status
Ticket 1 subject	16	2023/02/01	Open
Ticket 2 subject	13	2023/02/01	Resolved
Ticket 3 subject	06	2023/02/01	Resolved
Ticket 4 subject	01	2023/02/01	Open

Raise a new support ticket

Subject

Tags

body

POST

- In **Raise a new support ticket form** she can enter the details of her query or concern.
- The system automatically checks if there are any similar tickets already created by other students.
- If there are other similar tickets, the system displays the existing tickets to the student and prompts them to check if their concern has already been addressed.
- She can also **upvote** an existing ticket.

Support Dashboard

Smart Support

HOME

PROFILE

ALL TICKETS

SEARCH

LOGOUT

Open Tickets

Subject	Votes	Date	Action
Ticket 1 subject	16	2023/02/01	Respond
Ticket 2 subject	10	2023/02/01	Respond
Ticket 3 subject	09	2023/02/01	Respond
Ticket 4 subject	06	2023/02/01	Respond

Resolved Tickets

Subject	Votes	Date	Resolved on
Ticket 1 subject	16	2023/02/01	2023/02/01
Ticket 2 subject	10	2023/02/01	2023/02/01
Ticket 3 subject	09	2023/02/01	2023/02/01
Ticket 4 subject	06	2023/02/01	2023/02/01

Admin Dashboard

Smart Support

HOME

PROFILE

ALL TICKETS

SEARCH

LOGOUT

Open Tickets

Resolved Tickets

Subject	Votes	Date	Action	Subject	Votes	Date	Resolved on	Action
Ticket 1 subject	16	2023/02/01	Respond / Assign	Ticket 1	16	2023/02/01	2023/02/01	Add to FAQ
Ticket 2 subject	10	2023/02/01	Respond / Assign	Ticket 2	10	2023/02/01	2023/02/01	Add to FAQ
Ticket 3 subject	09	2023/02/01	Respond / Assign	Ticket 3	09	2023/02/01	2023/02/01	Add to FAQ
Ticket 4 subject	06	2023/02/01	Respond / Assign	Ticket 4	06	2023/02/01	2023/02/01	Add to FAQ

The system updates the ticket with the number of **upvotes** it has received and displays the most popular tickets at the top of the list on support and admin dashboards.

Smart Support

HOME

PROFILE

MY TICKETS

SEARCH

LOGOUT

Ticket Title

+

12 votes

Tag 1 Tag 2 Tag 3

Ticket body. This can be a big space text containing a user query that the user expects to be clarified by the support team. Below this section is the comment section which the students and support team may use for further clarifications.

ACCEPTED RESOLVE - THE ACCEPTED ANSWER FROM THE SUPPORT TEAM WILL SHOW HERE AS A RESOLVE FOR THE TICKET.

Comments

Comment- can be a long comment body
user details - date
Mark as answer

Comment- can be a long comment body
user details - date
Mark as answer

Comment- can be a long comment body
user details - date
Mark as answer

Post Comment

Comment body

POST

If the student decides to add a comment to the existing ticket instead of creating a new one, she can do so by visiting the relevant Ticket page

Support Dashboard

Smart Support

HOME

PROFILE

ALL TICKETS

SEARCH

LOGOUT

Open Tickets

Subject	Votes	Date	Action
Ticket 1 subject	16	2023/02/01	Respond
Ticket 2 subject	10	2023/02/01	Respond
Ticket 3 subject	09	2023/02/01	Respond
Ticket 4 subject	06	2023/02/01	Respond

Resolved Tickets

Subject	Votes	Date	Resolved on
Ticket 1 subject	16	2023/02/01	2023/02/01
Ticket 2 subject	10	2023/02/01	2023/02/01
Ticket 3 subject	09	2023/02/01	2023/02/01
Ticket 4 subject	06	2023/02/01	2023/02/01

The support team receives the ticket and begins working on a solution.

Ticket Page

Smart Support

HOME

PROFILE

MY TICKETS

SEARCH

LOGOUT

Ticket Title

+ 12 votes

Tag 1 Tag 2 Tag 3

Ticket body. This can be a big space text containing a user query that the user expects to be clarified by the support team. Below this section is the comment section which the students and support team may use for further clarifications.

ACCEPTED RESOLVE - THE ACCEPTED ANSWER FROM THE SUPPORT TEAM WILL SHOW HERE AS A RESOLVE FOR THE TICKET.

Post Comment

Comment body

POST

Comments

Comment- can be a long comment body
user details - date
Mark as answer

Comment- can be a long comment body
user details - date
Mark as answer

Comment- can be a long comment body
user details - date
Mark as answer

They can view the ticket, add comments, and update the ticket status as they work on the issue.

Student Dashboard

Smart Support

HOME

PROFILE

MY TICKETS

SEARCH

LOGOUT

My Tickets

Subject	Votes	Date	Status
Ticket 1 subject	16	2023/02/01	Open
Ticket 2 subject	13	2023/02/01	Resolved
Ticket 3 subject	06	2023/02/01	Resolved
Ticket 4 subject	01	2023/02/01	Open

Raise a new support ticket

Subject

Tags

body

POST

- Once the support team resolves the issue, they update the ticket status to **Resolved** and notify the student.
- The student receives a notification about the updated status of the ticket and can view the solution provided by the support team.
- The status of ticket changes from **Open** to **Resolved** on the *Student Dashboard*

Admin Dashboard

Smart Support				HOME	PROFILE	ALL TICKETS	SEARCH	LOGOUT
Open Tickets				Resolved Tickets				
Subject	Votes	Date	Action	Subject	Votes	Date	Resolved on	Action
Ticket 1 subject	16	2023/02/01	Respond / Assign	Ticket 1	16	2023/02/01	2023/02/01	Add to FAQ
Ticket 2 subject	10	2023/02/01	Respond / Assign	Ticket 2	10	2023/02/01	2023/02/01	Add to FAQ
Ticket 3 subject	09	2023/02/01	Respond / Assign	Ticket 3	09	2023/02/01	2023/02/01	Add to FAQ
Ticket 4 subject	06	2023/02/01	Respond / Assign	Ticket 4	06	2023/02/01	2023/02/01	Add to FAQ

- Admins can see the list of all **Open** and **Resolved** tickets on their dashboard sorted by *upvotes* and *date*.
- They can *assign* the tickets to specific support users or they can *respond* to the tickets themselves.
- They can also add any of the resolved tickets to FAQs list.

Schedule and Design

Sprint 1

▼ STS Sprint 1

0

0

0

Start sprint

...

🔖

STS-15

BE - API Docs

BE - DOCUMENTATION

TO DO ▼

U

🔖

STS-3

BE - Prepare Codebase

BE - BOILERPLATE

TO DO ▼

NS

🔖

STS-4

BE - Prepare DB

BE - BOILERPLATE

TO DO ▼

NS

🔖

STS-16

BE - Add Fake Data

BE - BOILERPLATE

TO DO ▼

NS

🔖

STS-5

BE - Setup Auth

BE - DEVELOPMENT

TO DO ▼

U

🔖

STS-6

BE -Implement CRUD APIs

BE - DEVELOPMENT

TO DO ▼

KG

🔖

STS-7

BE - Implement Issue Search API

BE - DEVELOPMENT

TO DO ▼

U

🔖

STS-10

BE - Implement unit testing


BE - TESTING

TO DO ▼










KG

+ Create issue

Sprint 2

▼ STS Sprint 2  Add dates (9 issues)

0 0 0 Start sprint ...

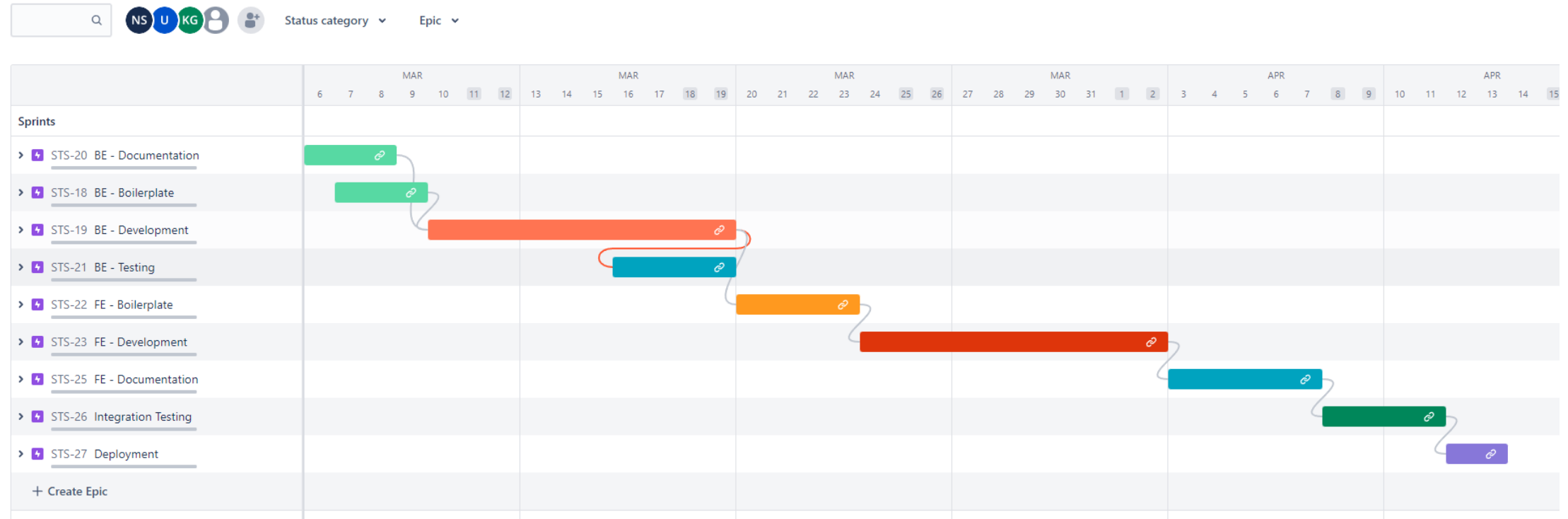
 STS-8 FE - Prepare Codebase FE - BOILERPLATE	TO DO ▼	NS
 STS-9 FE - Implement Auth FE - DEVELOPMENT	TO DO ▼	U
 STS-11 FE - Landing Page FE - DEVELOPMENT	TO DO ▼	U
 STS-12 FE - Issue Page FE - DEVELOPMENT	TO DO ▼	NS
 STS-13 FE - Ticket Page FE - DEVELOPMENT	TO DO ▼	U
 STS-14 FE - All Tickets Page FE - DEVELOPMENT	TO DO ▼	NS
 STS-24 FE - Documentation and Project Report FE - DOCUMENTATION	TO DO ▼	NS
 STS-28 Integrate Testing INTEGRATION TESTING	TO DO ▼	U
 STS-29 Deploy on Web DEPLOYMENT	TO DO ▼	U

+ Create issue

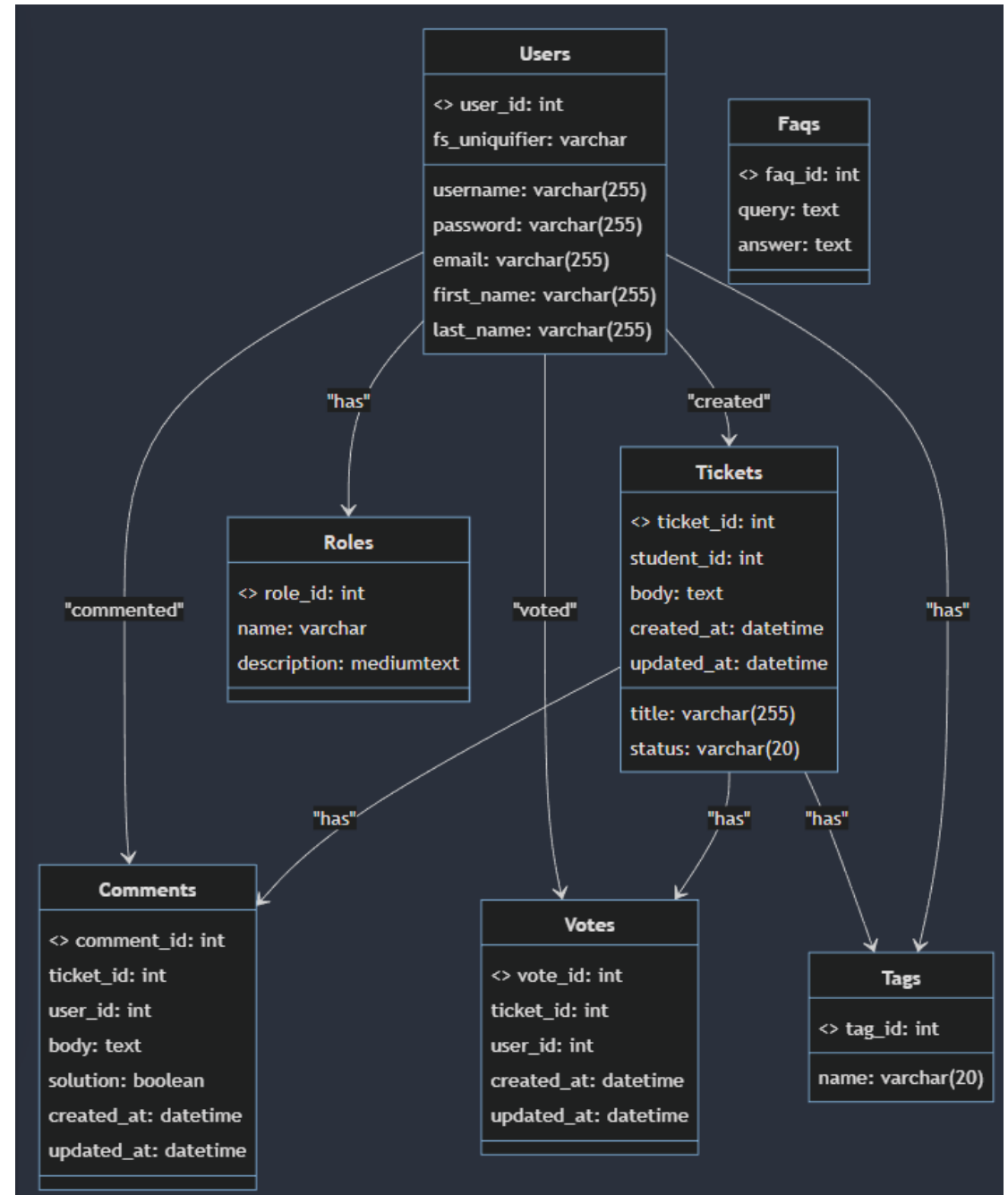
Gant Chart

Roadmap

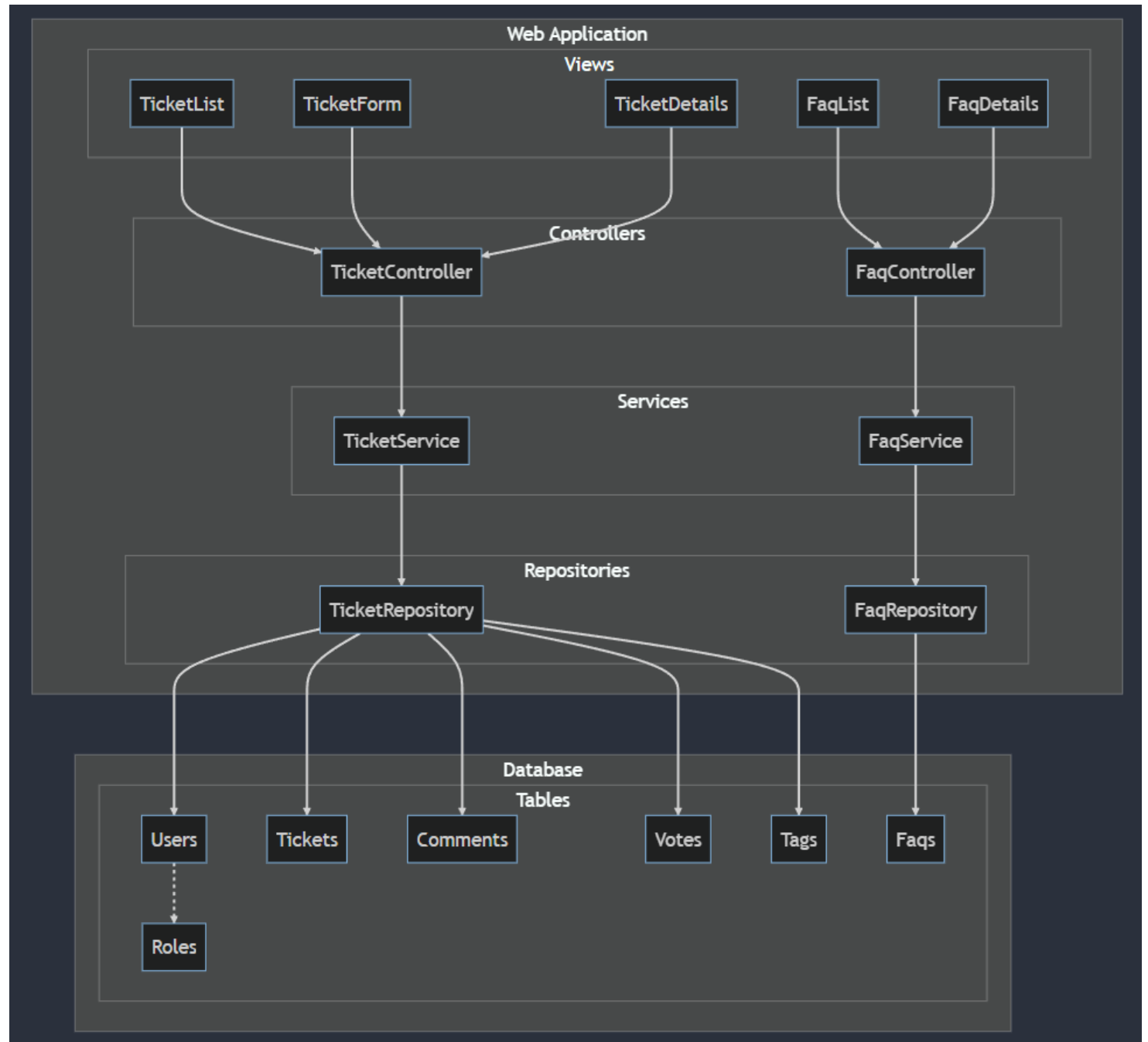
 Give feedback



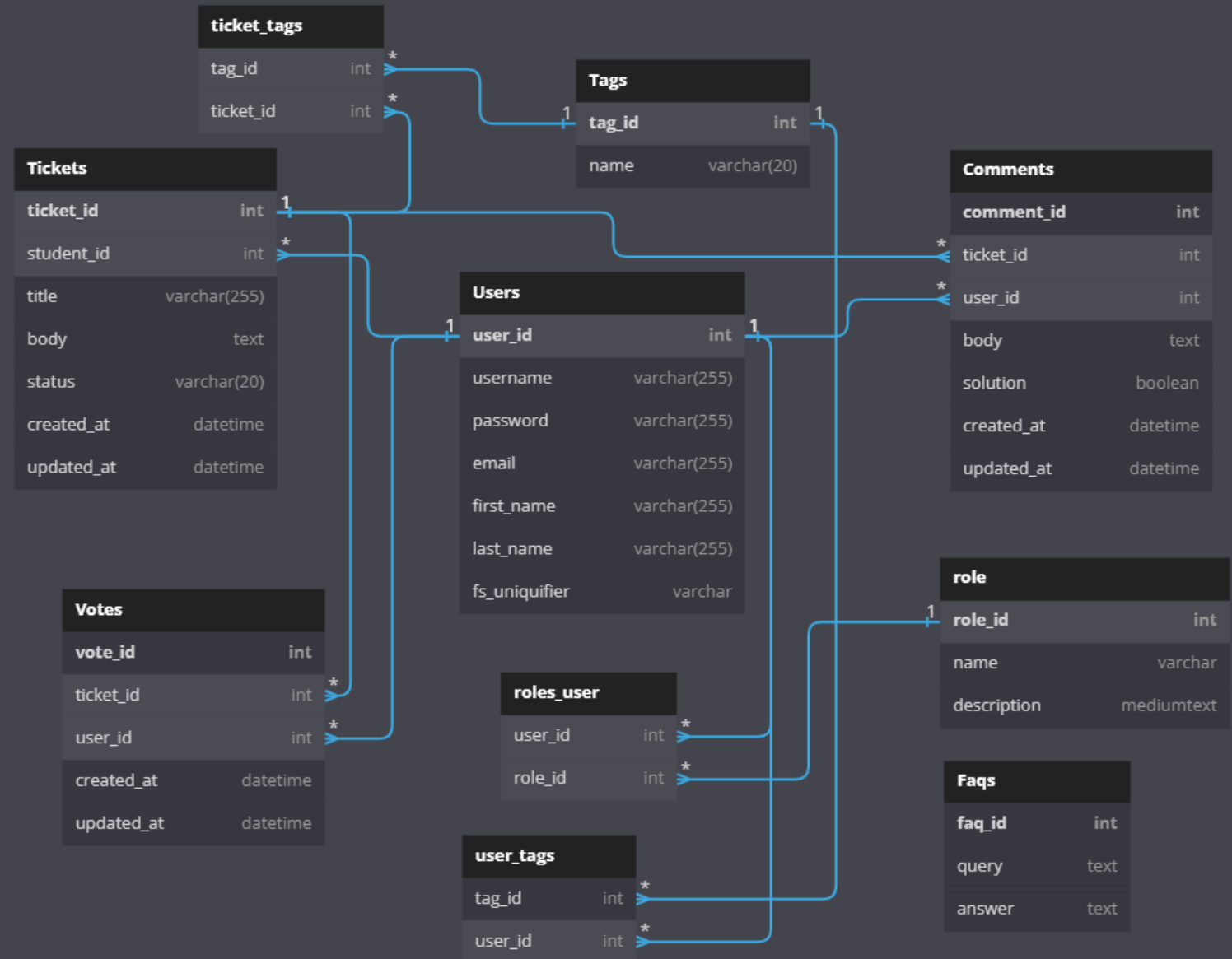
Class Diagram



Component Design



DB Schema



Scrum Meeting 1: Date: 2023-02-10 Time: 9:00 PM

Attendees:

- Uday
- Navin
- Krishu

Agenda:

1. Getting to know each other
2. Assign roles
3. Discuss general structure and timeline of the project

Minutes:

- Each member introduced themselves.
- Roles were discussed
- Discussed general structure of the project.

Scrum Meeting 2: Date: 2023-02-15 Time: 10:00 PM

Attendees:

- Uday
- Navin

Agenda:

1. Review of Milestone 2 Submission
2. Discussion of upcoming milestone goals
3. Standardize theme of all wireframes

Minutes:

- Some mock user stories discussed.
- General theme for all wireframe pages were discussed.
- A few mock wireframes were developed.
- The team discussed other business, including the need for a clearer definition of roles and responsibilities.

Scrum Meeting 3: Date: 2023-02-23 Time: 9:30 PM

Attendees:

- Uday
- Navin

Agenda:

1. Finalise Milestone 2 Submission
2. Discussion of upcoming milestone goals

Minutes:

- Final user stories and wireframe were designed by Navin
- Uday recommended some changes to wireframes
- Final Milestone 2 draft was prepared.

Scrum Meeting 4: Date: 2023-03-02 Time: 10:30 PM

Attendees:

- Uday
- Navin

Agenda:

- Discuss tools to be used for Milestone 3
- Assign roles
- Create Timeline of the project.

Minutes:

- Jira was decided upon for project scheduling
- Two sprints were discussed and scheduled on Jira
- Categories (Epics) and their issues/tasks were created and added to their respective sprints.
- Timeline was decided for each issues/task.
- Issues/Tasks were assigned to each member.

API Endpoints

User Endpoints

Users APIs related to managing all users			^
GET	/user/all	Get all users	✓ 🔒
POST	/user/register	Create a new user	✓ 🔒
POST	/user/login	login	✓ 🔒
GET	/user	Get a user by JWT	✓ 🔒
PUT	/user	Update a user by JWT	✓ 🔒
DELETE	/user	Delete a user by JWT	✓ 🔒
POST	/user/tags	Assign a tag to a user	✓ 🔒
PUT	/user/tags	Revoke a tag from a user	✓ 🔒
PUT	/user/roles	Add role to a user	✓ 🔒
DELETE	/user/roles	Remove role from a user	✓ 🔒

Ticket Endpoints

Ticket APIs related to managing tickets



GET	/tickets	Get all tickets	✓	🔒
POST	/tickets	Create a new ticket	✓	🔒
GET	/tickets/open	Get all open tickets	✓	🔒
GET	/tickets/resolved-or-closed	Get all resolved or closed tickets	✓	🔒
GET	/tickets/support/all	Get all open tickets	✓	🔒
GET	/tickets/support/open	Get all open tickets	✓	🔒
GET	/tickets/support/resolved-or-closed	Get all open tickets	✓	🔒
GET	/tickets/user	Get all tickets raised by a given user	✓	🔒
GET	/tickets/{ticket_id}	Get a ticket by ID	✓	🔒
PUT	/tickets/{ticket_id}	Update a ticket by ID	✓	🔒

Ticket Endpoints continued...

DELETE	/tickets/{ticket_id}	Delete a ticket by ID	✓	🔒
GET	/tickets/{ticket_id}/comments	Get all comments for a ticket by ticket_id	✓	🔒
POST	/tickets/{ticket_id}/comments	Create a new comment on a Ticket with ticket_id	✓	🔒
GET	/tickets/search	Search tickets by keyword	✓	🔒
PUT	/tickets/{ticket_id}/close	Mark a ticket as resolved	✓	🔒
POST	/tickets/notify	Send email notification to student about resolved ticket	✓	🔒
POST	/tickets/{ticket_id}/faqs	Convert a ticket to a FAQ	✓	🔒

Comment and Tag Endpoints

Comment APIs related to managing comments

**GET**`/comments/{comment_id}` Get a comment by ID**PUT**`/comments/{comment_id}` Update a comment by ID**DELETE**`/comments/{comment_id}` Delete a comment by ID**PUT**`/comments/{comment_id}/solution` Mark a comment as solution

Tag APIs related to managing tags

**GET**`/tags` Get all tags**POST**`/tags` Create a new tag**PUT**`/tags/{tag_id}` Update a tag**DELETE**`/tags/{tag_id}` Delete a tag

Vote and FAQs Endpoints

Vote APIs related to managing votes

POST	/tickets/{ticket_id}/upvote	Upvote a ticket	✓	🔒
DELETE	/tickets/{ticket_id}/revoke-vote	Revoke vote on a ticket	✓	🔒

FAQ APIs related to managing FAQs

GET	/faqs	Get all FAQs	✓	🔒
POST	/faqs	Create a new FAQ	✓	🔒
GET	/faqs/{faq_id}	Get an FAQ by ID	✓	🔒
PUT	/faqs/{faq_id}	Update an existing FAQ	✓	🔒
DELETE	/faqs/{faq_id}	Delete an existing FAQ	✓	🔒

Test Cases

We have implemented API testing using **pytest** for CRUD APIs for the **Ticket** model in our application

```
def test_login():
    global JWT
    response = app.test_client().post('/api/user/login', json={
        "username": "craig_fox",
        "password": "password"
    })
    JWT = response.get_json()['access_token']
    assert response.status_code == 200

def test_get_tickets():
    response = app.test_client().get(
        '/api/tickets?page=0&per_page=10', headers={'Authorization': 'Bearer ' + JWT})
    assert response.status_code == 200

def test_post_tickets():
    global TICKET_ID
    response = app.test_client().post(
        '/api/tickets', headers={'Authorization': 'Bearer ' + JWT},
        json={
            "title": "Need help with Python",
            "body": "I'm having trouble understanding Python classes. Can someone help me?"
        })
    TICKET_ID = response.get_json()['ticket_id']
    print(TICKET_ID)
    assert response.status_code == 200

def test_get_ticket_by_id():
    response = app.test_client().get(
        '/api/tickets/{}'.format(TICKET_ID), headers={'Authorization': 'Bearer ' + JWT})
    assert response.status_code == 200
```

So far we have implemented 6 test cases for Ticket APIs
(including 1 for the Login API)

```
● (project) navin@DESKTOP-J1K386M:~/iit/SE/project/soft-engg-project-jan-2023-group-13/project/code/backend/smartSupport$ pytest -v
===== test session starts =====
platform linux -- Python 3.11.0rc1, pytest-7.2.2, pluggy-1.0.0 -- /home/navin/iit/SE/project/.venv/bin/python
cachedir: .pytest_cache
rootdir: /home/navin/iit/SE/project/soft-engg-project-jan-2023-group-13/project/code/backend/smartSupport
plugins: Faker-18.4.0
collected 6 items

tests/test_tickets_api.py::test_login PASSED [ 16%]
tests/test_tickets_api.py::test_get_tickets PASSED [ 33%]
tests/test_tickets_api.py::test_post_tickets PASSED [ 50%]
tests/test_tickets_api.py::test_get_ticket_by_id PASSED [ 66%]
tests/test_tickets_api.py::test_update_ticket_by_id PASSED [ 83%]
tests/test_tickets_api.py::test_delete_ticket_by_id PASSED [100%]

===== 6 passed in 1.43s =====
○ (project) navin@DESKTOP-J1K386M:~/iit/SE/project/soft-engg-project-jan-2023-group-13/project/code/backend/smartSupport$
```

Some other test cases that we intend to
implement in future versions

Test to get all comments for a valid ticket ID

- **Page being tested:** get_comments function
- **Inputs:** A valid ticket ID in the URL and an authenticated JWT token
- **Expected output:** A JSON response with status code 200 and a list of all comments for the given ticket ID in descending order of creation time
- **Actual output:** Same as expected output
- **Result:** Success

Test to get a single comment by ID

- **Page being tested:** get_comment function
- **Inputs:** A valid comment ID in the URL and an authenticated JWT token
- **Expected output:** A JSON response with status code 200 and the comment details for the given ID
- **Actual output:** Same as expected output
- **Result:** Success

Test to add a new comment to a valid ticket

- **Page being tested:** post_comment function
- **Inputs:** A valid ticket ID in the URL, a valid JWT token, and a JSON payload with a "body" field containing the comment text
- **Expected output:** A JSON response with status code 200 and the details of the newly created comment, including the comment ID and creation time
- **Actual output:** Same as expected output
- **Result:** Success

Test to update an existing comment

- **Page being tested:** put_comment function
- **Inputs:** A valid comment ID in the URL, a valid JWT token, and a JSON payload with a "body" field containing the updated comment text
- **Expected output:** A JSON response with status code 200 and the details of the updated comment, including the comment ID and creation time
- **Actual output:** Same as expected output
- **Result:** Success

Test to delete an existing comment

- **Page being tested:** delete_comment function
- **Inputs:** A valid comment ID in the URL and a valid JWT token
- **Expected output:** A JSON response with status code 204 and no content
- **Actual output:** Same as expected output
- **Result:** Success

Test to mark a comment as the solution for a ticket

- **Page being tested:** mark_comment_as_solution function
- **Inputs:** A valid comment ID in the URL and a valid JWT token
- **Expected output:** A JSON response with status code 200 and the details of the marked comment, including the comment ID and creation time. The associated ticket status should also be updated to "Resolved"
- **Actual output:** Same as expected output
- **Result:** Success

Besides the tests mentioned in previous pages, there can be tests written for:

- User CRUD operations
- Tickets Voting operations
- Tags CRUD operations
- FAQs CRUD operations

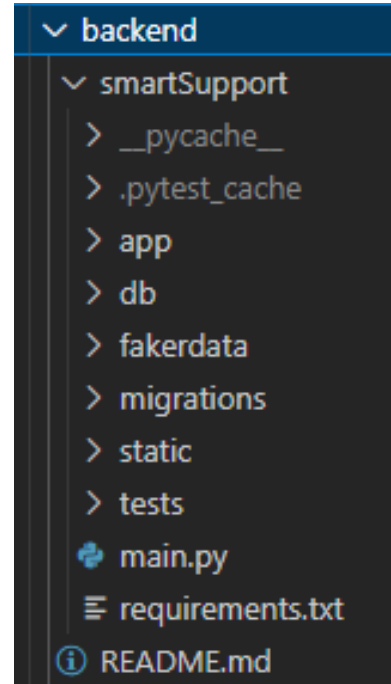
Implementation details

Folder structure

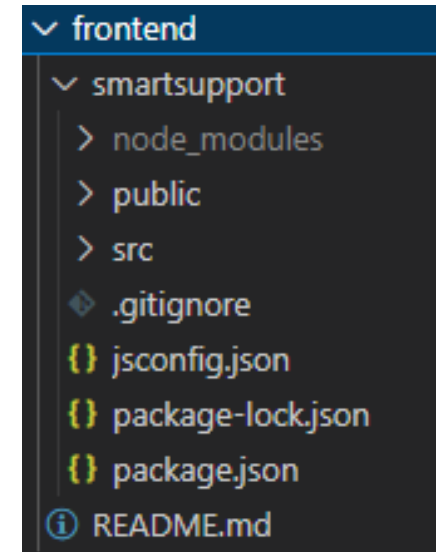
code/

backend/smartSupport
frontend/smartsupport

Backend Folder Structure:



Frontend Folder Structure:



Tech Stack

Backend:

- Flask – as scripting Framework
- SQLite – for database
- MailHog – for SMTP testing
- Faker – for generating fake data
- PyTest – for testing

Frontend:

- Vue3 – as scripting Framework
- Bootstrap 5 – as css Framework

Tools used

- **vsCode** for IDE
- **dbBrowser** for managing SQLite database
- **MailHog** for SMTP testing
- <https://dbdiagram.io/> for generating DB Schema
- <https://imagineui.netlify.app/> for generating wireframes
- **JIRA** for project management and scheduling
- **Github** for version control

Steps to start Backend server

Step 1:

- CD to ``code/backend/smartSupport`` directory.

Step 2:

- Run ``pip install requirements.txt`` or ``pip3 install requirements.txt`` to install all dependencies

Step 3:

- Run ``python main.py`` or ``python3 main.py``

To start MailHog server run:

- ``~/go/bin/MailHog``
- or
- ``go/bin/MailHog``

Mailhog installation guide

<https://github.com/mailhog/MailHog>

Steps to start Frontend server

For project setup run:

```
`npm install`
```

To start Server run:

```
`npm run serve`
```

To compile and minify for production run:

```
`npm run build`
```

Issue Tracking and Reporting

We used JIRA for Issue Tracking:

The screenshot displays the JIRA Backlog interface. On the left, a list of issues is shown, each with a key, description, status, and assignee. The issue STS-13, 'FE - Ticket Page', is highlighted. On the right, the detailed view of STS-13 is shown, including a description field, a 'Save' button, and a comment section with two entries.

Backlog

Search: [] Assignees: NS, U, KG, [] Filter: Epic

Key	Description	Status	Assignee
STS-9	FE - Implement Auth	FE - DEVELOPMENT	U
STS-11	FE - Landing Page	FE - DEVELOPMENT	U
STS-12	FE - Issue Page	FE - DEVELOPMENT	NS
STS-13	FE - Ticket Page	FE - DEVELOPMENT	U
STS-14	FE - All Tickets Page	FE - DEVELOPMENT	NS
STS-24	FE - Documentation and Project R...	FE - DOCUMENTATION	NS
STS-28	Integrate Testing	INTEGRATION TESTING	U
STS-29	Deploy on Web	DEPLOYMENT	U

+ Create issue

Issue Details: STS-13

ST-23 / STS-13

Ad, B, I, A, List, +

Save Cancel

U Uday 3 minutes ago
Okay done
Edit · Delete · [icon]

NS NAVIN KUMAR SINGH 12 minutes ago
Please remove DELETE button from tickets for non owners
Edit · Delete · [icon]