SOFTWARE ENGINEERING

(Course ID: BSCS3001)

Online Ticketing System with Discourse and Gchat Integration

Milestone 1

By

Anupam Kumar Jha (21f1004905)

Kevin Joshua T (22f1001410)

Pranav Wankhedkar (21f1000120)

Sachin Singh (21f1003251)

Ujit Kumar (21f3000786)

Utpal Dutta (21f2000524)

Submission Date: 23 Feb. 2024

❖ **Identifying the various types of Users:**

- **Primary Users:** Students, Support Staff sand Admins
- **Secondary Users:** Managers
- **Tertiary Users:** Software Developers, Internet Service Provider, Hosting Platforms

❖ **User Stories:**

1.

As a student	I want to tag the concern person.
So that	My issues get resolved faster

2.

As a student	I want to keep track of my issue through Message/Mail whenever any updates is there.
So that	So that I can Check them

3.

As a student	I want to create my query on discourse platform
So that	My concern get attention on discourse as well!

4.

As a student	I want to search discussions based on a tag
So that	I can easily go to the discussion I want to refer to

5.

As a student	I want click on the name or id of the commenter
So that	I can look into the profile of the user

6.

As a student	I want to use filters based on category and tags
So that	I can easily navigate to the kind of discussion I want to see or participate

7.

As a Support staff	I want to mark a discourse thread as favourite
So that	I can directly go and visit them

8.

As a Support staff	I want that my replies to tickets should be shown on discourse in particular concern thread
So that	My query gets solved on discourse as well

9.

As a Support staff	I want that when I mark a query as solved or closed, it should reflect on discourse as well
So that	So as student if I visit discourse I can know that my query has been resolved or closed.

10.

As a Support staff	I want to get a message on Gchat when priority measure goes beyond 5.
So that	So that I can stop escalation on the portal.

11.

As a Support staff	I want to click on the name or id of the student
So that	I can view the profile of the student

12.

As a Support staff	I want to submit clickable links for a previous discussion
So that	Student can go and visit them in case of duplicate queries

13.

As a Support Manager	I want to see which Support Staff are not Responding when they are tagged within 24Hrs
So that	Proper action can be taken against them

14.

As a Support Manager	I want to get notified if any user created more than 'x' threads in a day
So that	We can check if the user is Spamming the discourse

15.

As a Support Manager	I want to get message when some posts have flagged
So that	I can check the query immediately and take appropriate action

16.

As a Support Manager	I want to see a pie chart of resolved and pending threads

So that	I get a hint of the discourse systems efficiency and also data for presenting to management

17.

As a Support Manager	I want to see a pie chart of resolved queries within and delayed timeline counts
So that	I can take appropriate steps to ensure least delay in issue resolution

18.

As a Support Manager	I want see support staff wise issue being handled
So that	I can ensure equal workload distribution among all support staff

SOFTWARE ENGINEERING

(Course ID: BSCS3001)

Online Ticketing System with Discourse and Gchat Integration

Milestone 2

By

Anupam Kumar Jha (21f1004905)

Kevin Joshua T (22f1001410)

Pranav Wankhedkar (21f1000120)

Sachin Singh (21f1003251)

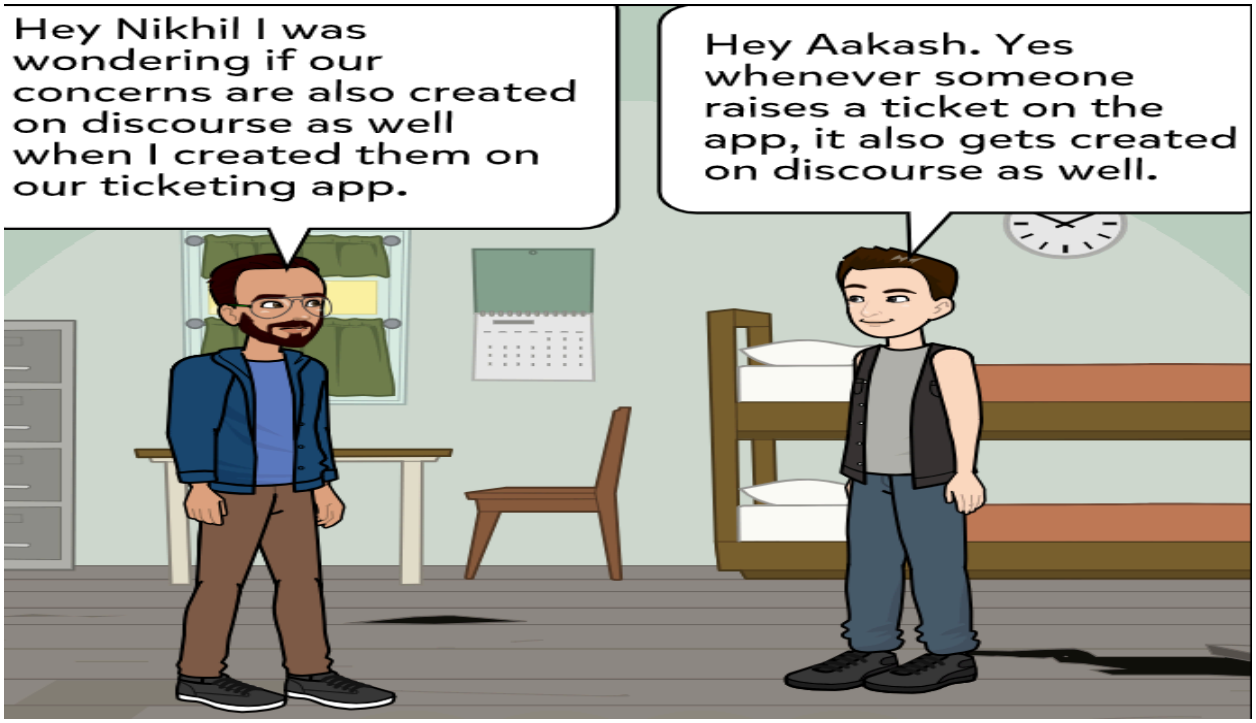
Ujit Kumar (21f3000786)

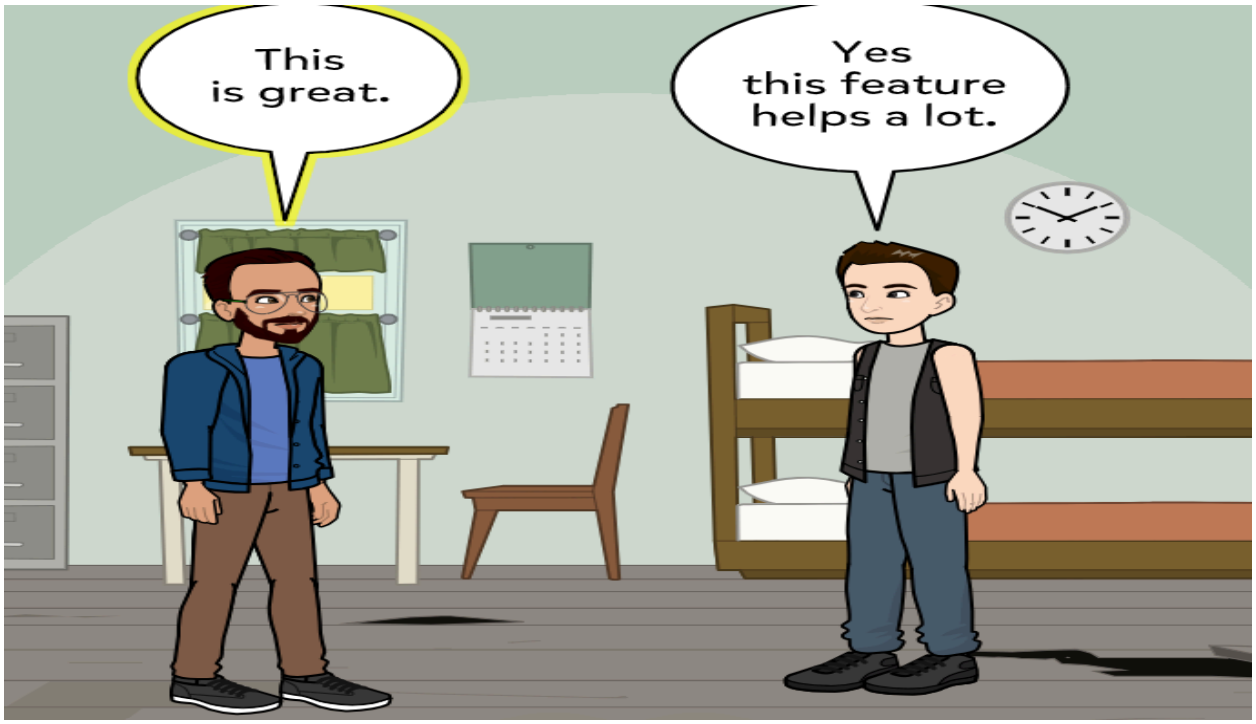
Utpal Dutta (21f2000524)

Submission Date: 29 Feb. 2024

Storyboard - 1

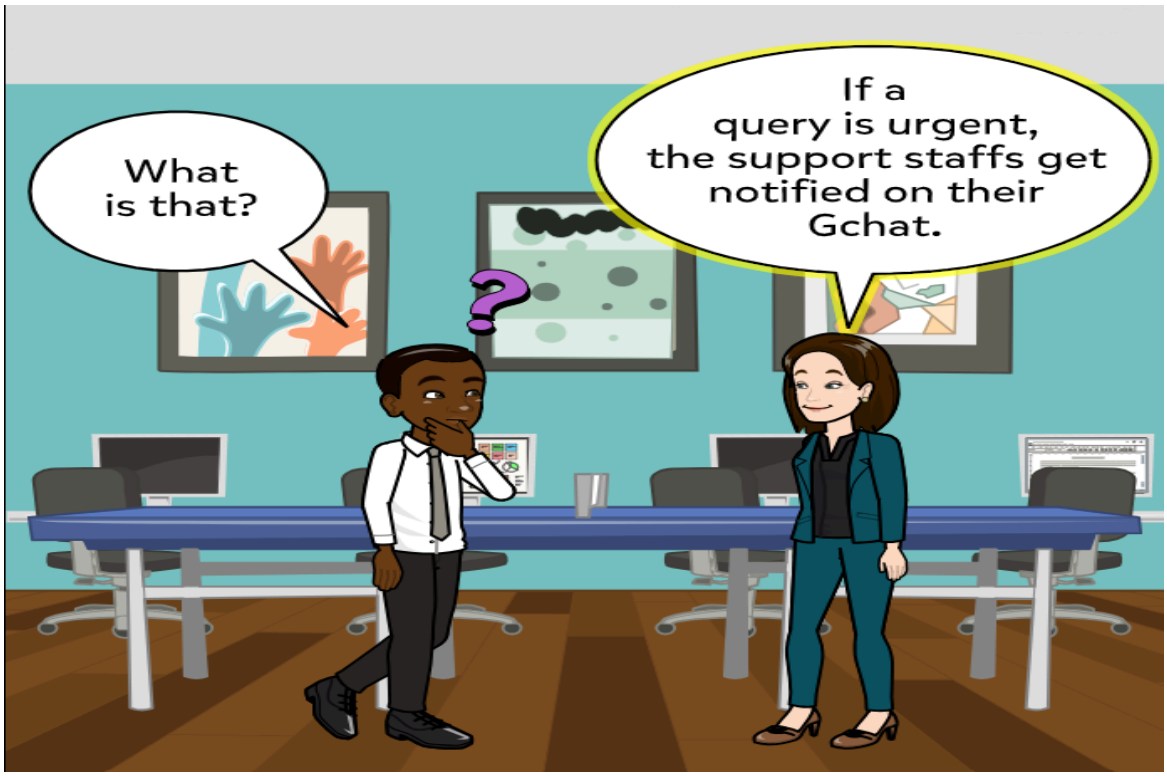
Student Perspective





Storyboard - 2

Support Staff Perspective


What is that?
If a query is urgent, the support staffs get notified on their Gchat.


Wow! It's amazing. So then I shall be able to respond to urgent query quickely. Thank you for introuducing me with this feature!
Always welcome!

Storyboard - 3

User Perspective

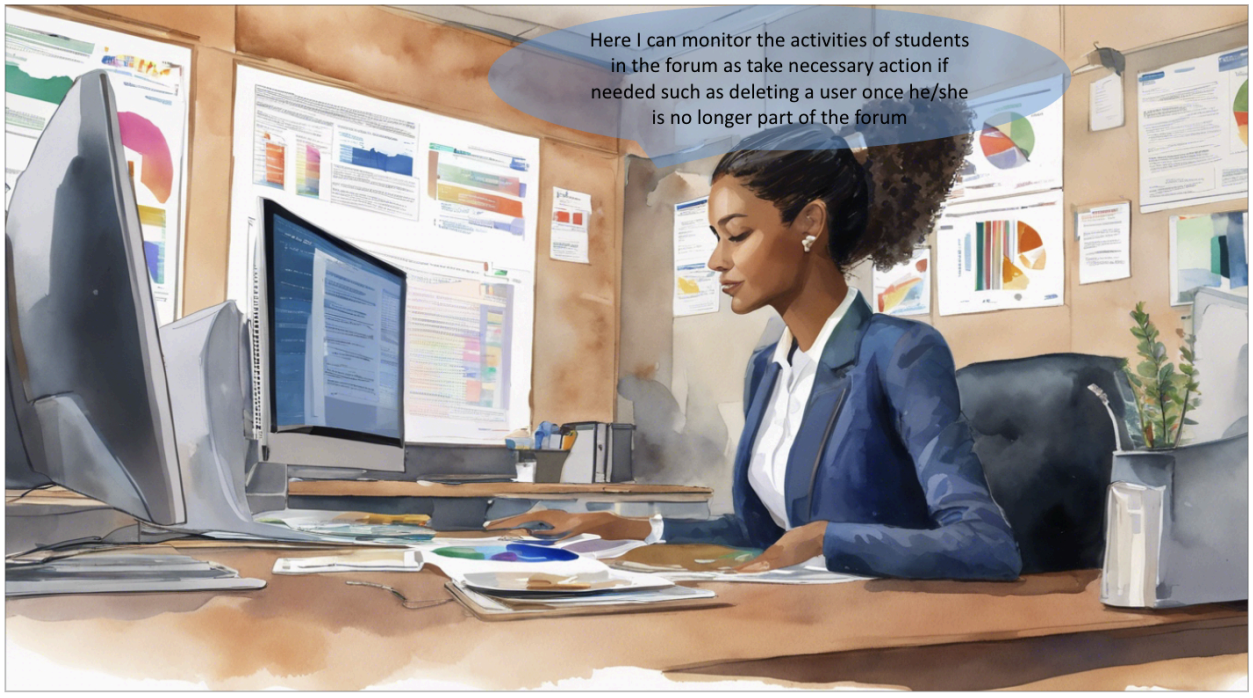
s

Storyboard - 4

Support Manager Perspective

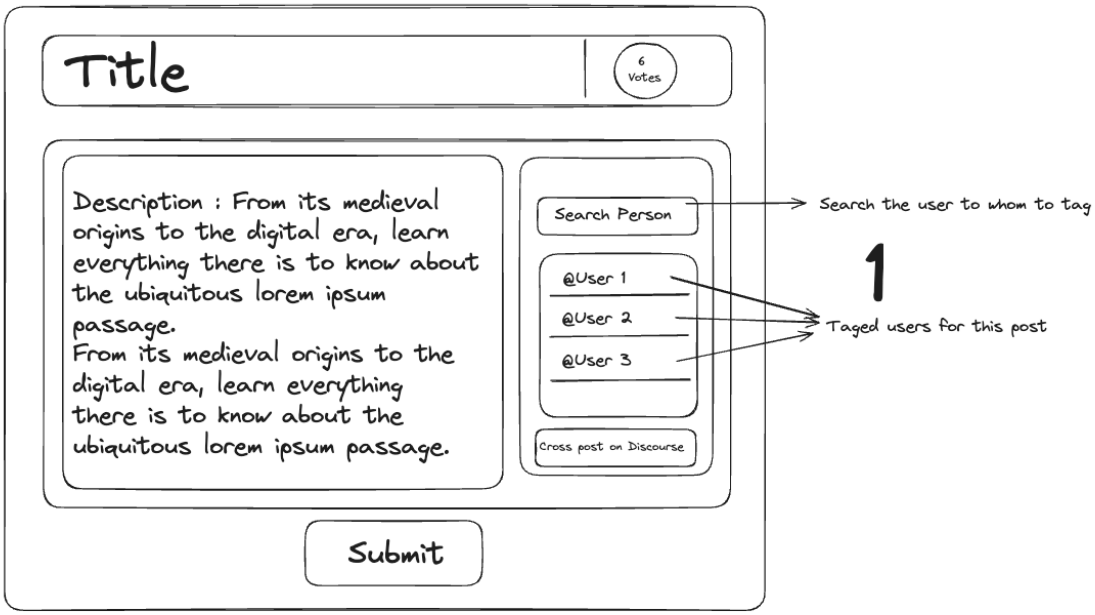
Storyboard - 5

Admin Perspective

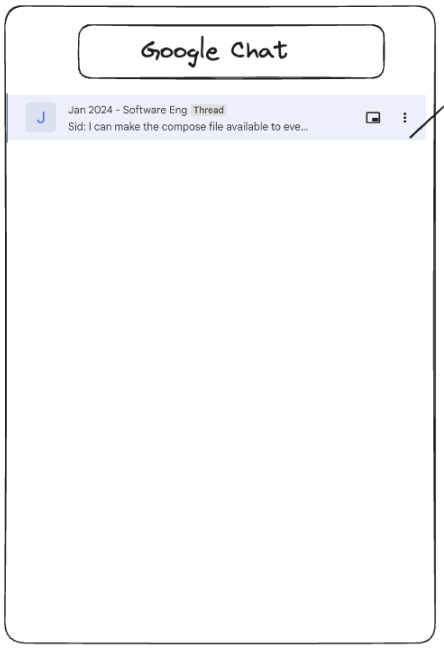


Wireframes

When a user creates a post want to tag concerned people in the thread. The use can search the person to be tagged from the search box and tag accordingly



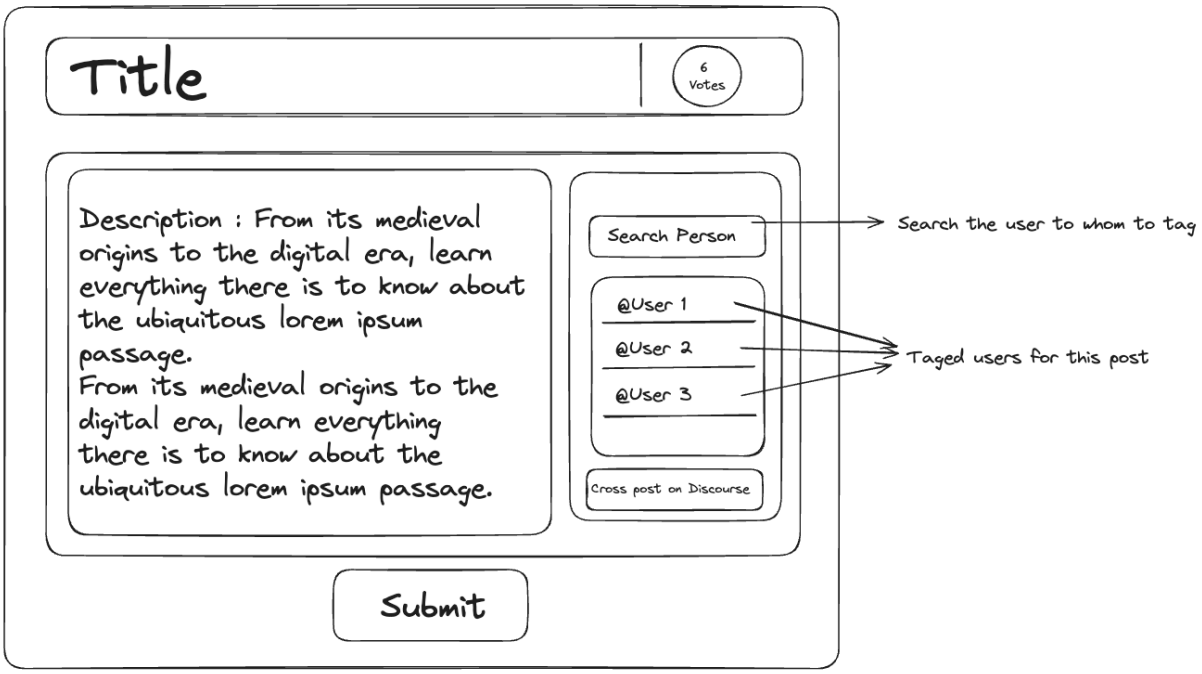
2



Notification/message
via Gchat for important/urgent
tickets/issues

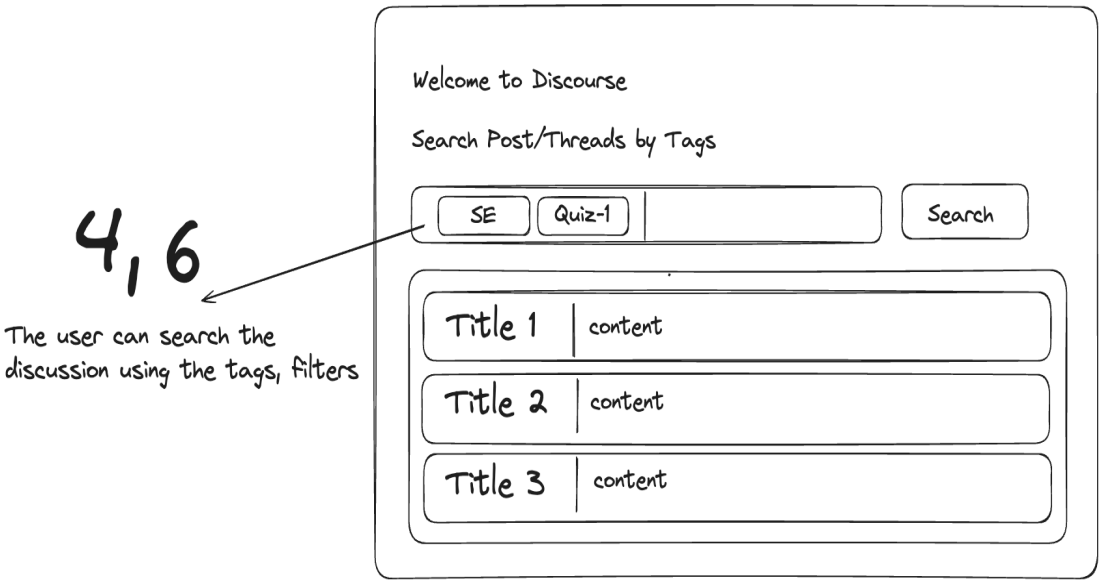
3

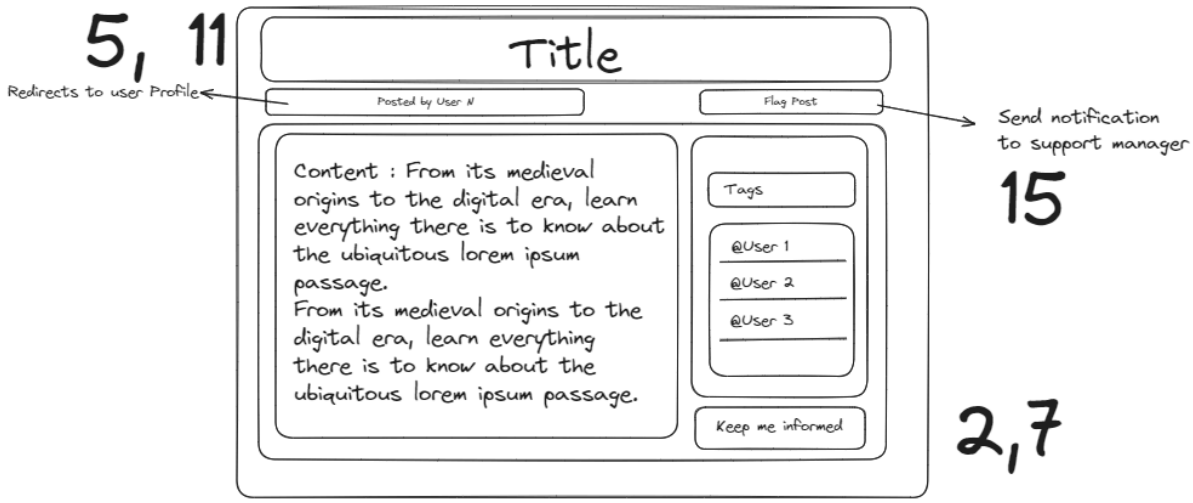
Use can create a query on discourse as well



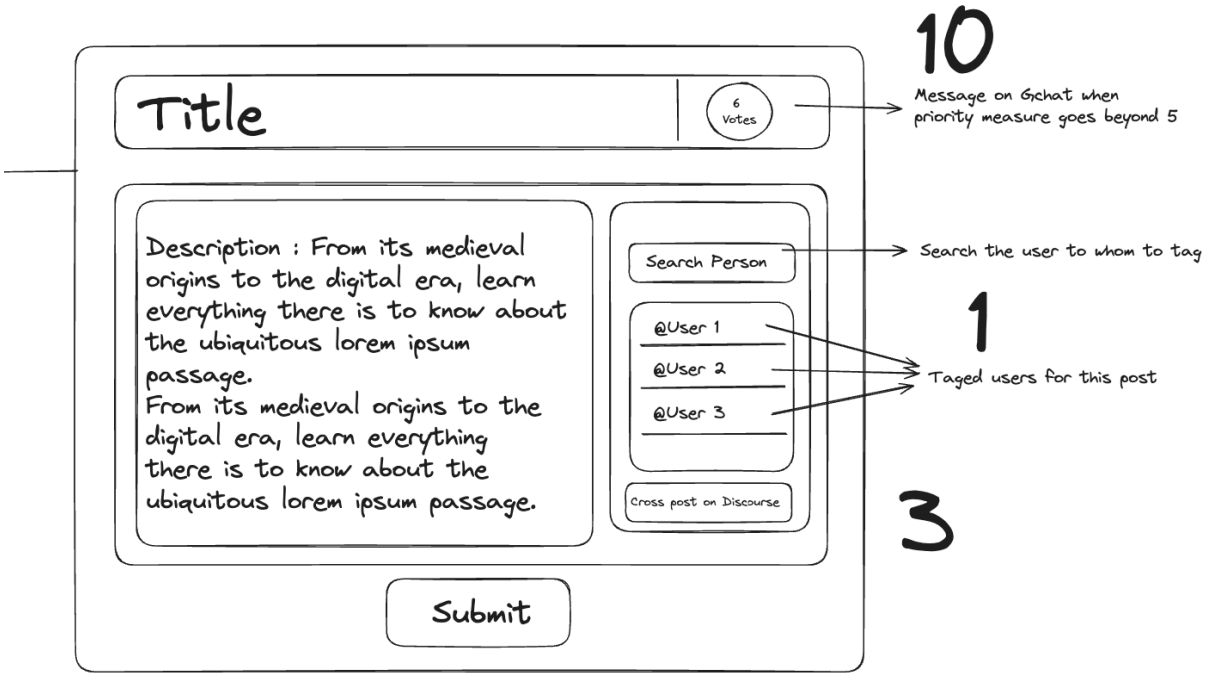
Search the user to whom to tag

Taged users for this post

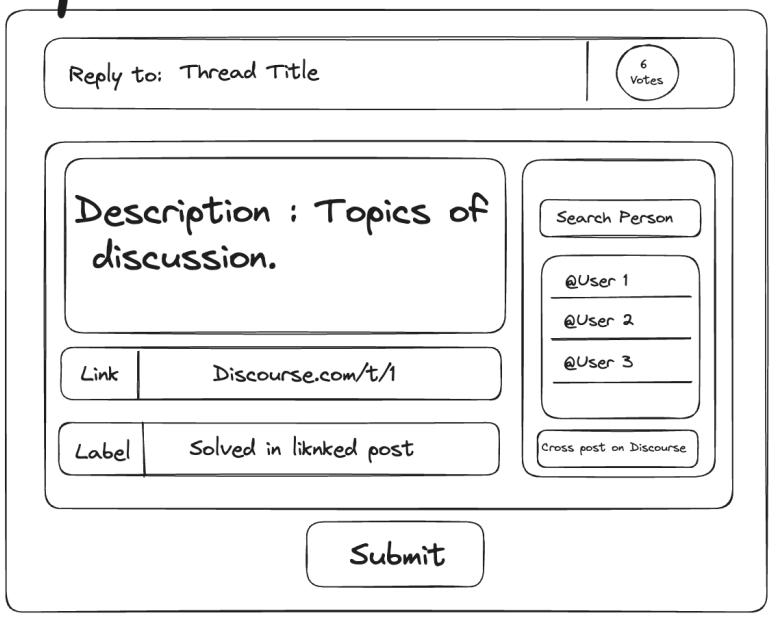

4, 6
The user can search the discussion using the tags, filters
Welcome to Discourse
Search Post/Threads by Tags
SE
Quiz-1
Search
Title 1
content
Title 2
content
Title 3
content


5, 11
Redirects to user Profile
Title
Posted by User N
Flag Post
Send notification to support manager
15
Content : From its medieval origins to the digital era, learn everything there is to know about the ubiquitous lorem ipsum passage.
From its medieval origins to the digital era, learn everything there is to know about the ubiquitous lorem ipsum passage.
Tags
@User 1
@User 2
@User 3
Keep me informed
2,7

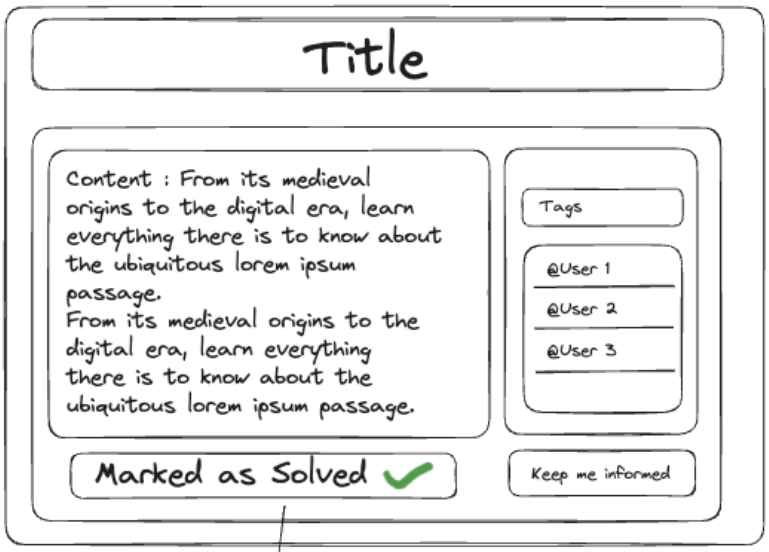
When a user creates a post want to tag concerned people in the thread. The use can search the person to be tagged from the search box and tag accordingly



8, 12



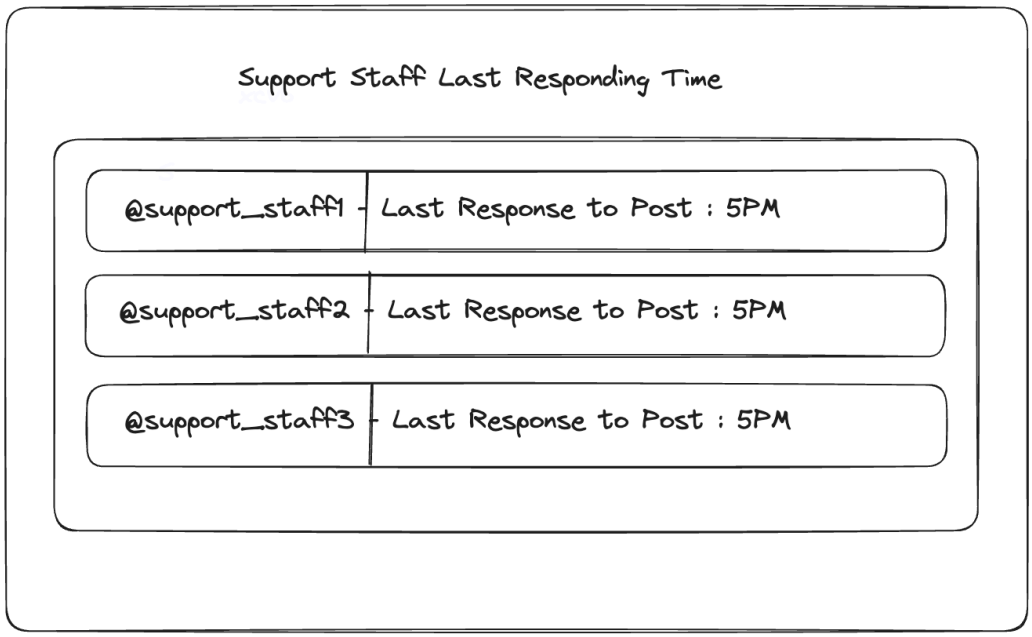
9



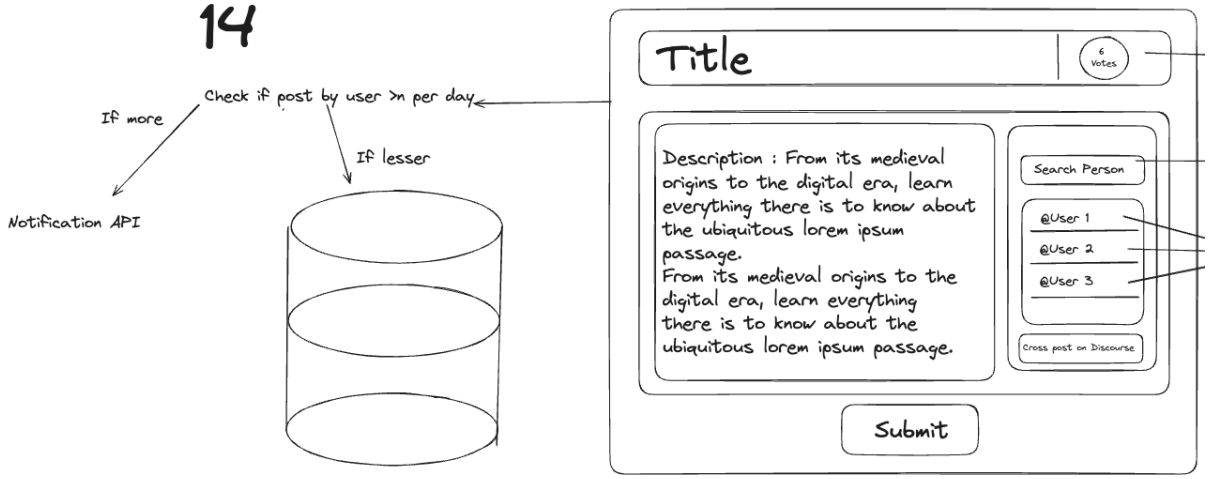
The support staff can mark the post as solved once he feels it has been answered properly

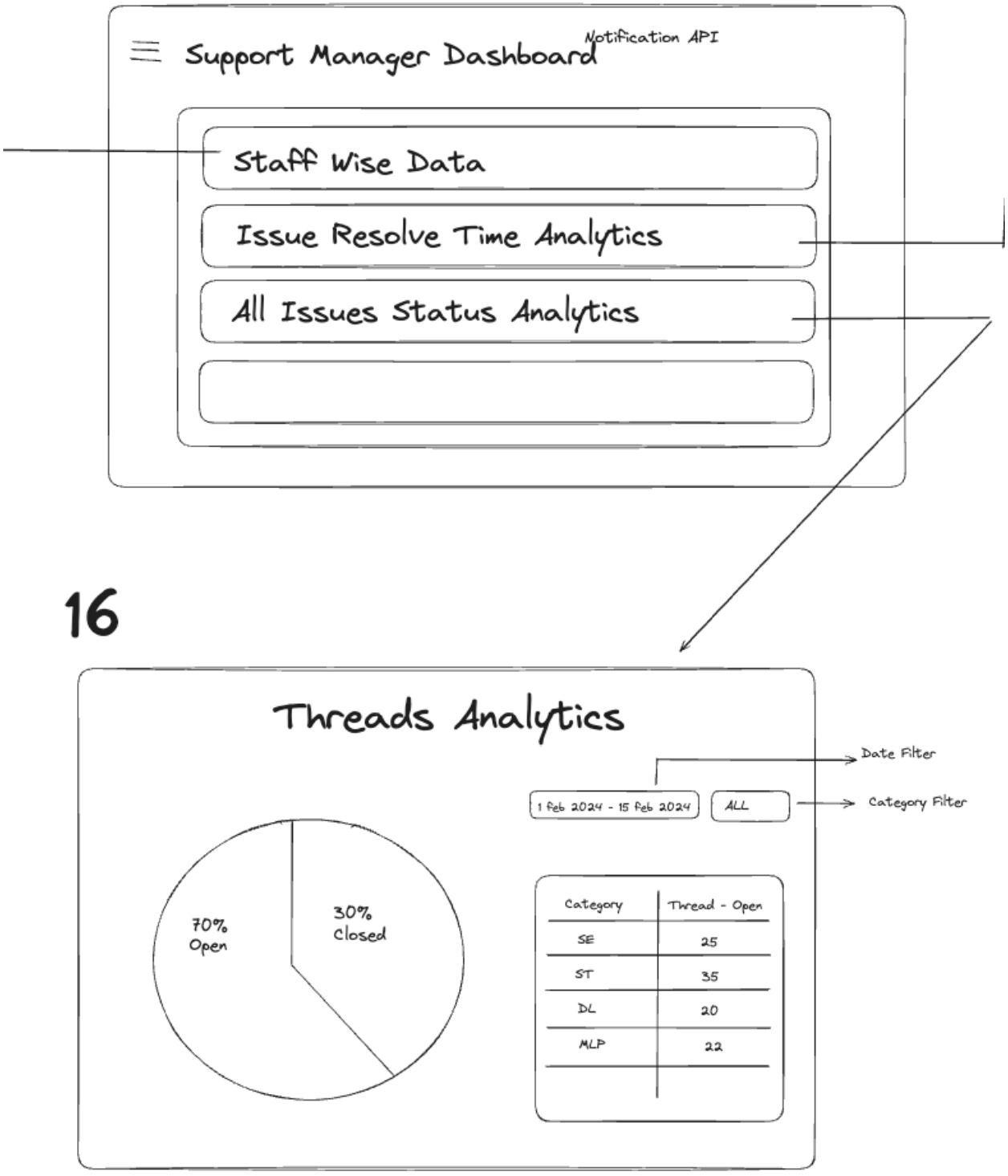
13

The Support Manager can see which support staff is active and which ones are not active so that proper action can be taken against them



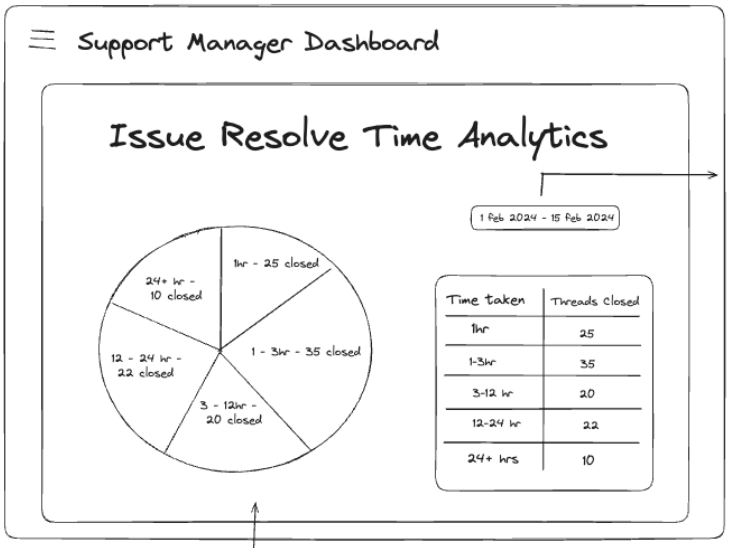
14





Pie chart for resolved and pending threads

17



The Support Manager can see the analytics for resolve time of queries

18

Staff Wise Data

S.No	Staff Name	Active Tags	Resolved	Subjects
1	staff-1	23	96	maths-2 , stats2
2	staff-2	25	56	mad-1 , mad-2
3	staff-3	15	40	ct , python
4	staff-4	16	35	java, python pdsa
5	staff-5	17	64	dbms, mlf
6	staff-6	24	48	maths-2 , stats2
7	staff-7	21	80	mad-1 , english-2

The Support Manager can see number of issues being handled and solved by each support staff

SOFTWARE ENGINEERING

(Course ID: BSCS3001)

Online Ticketing System with Discourse and Gchat Integration

Milestone 3

By

Anupam Kumar Jha (21f1004905)

Kevin Joshua T (22f1001410)

Pranav Wankhedkar (21f1000120)

Sachin Singh (21f1003251)

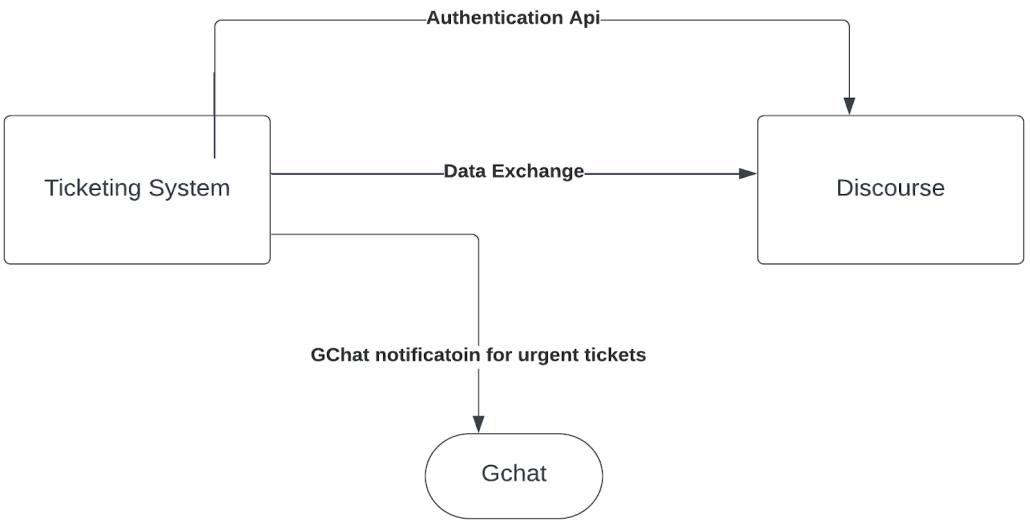
Ujit Kumar (21f3000786)

Utpal Dutta (21f2000524)

Submission Date: 06 Mar. 2024

Design of Components

❖ **Overall System Design:**



❖ **Authentication Mechanism:** Authentication of users will happen at the ticketing system level. Once the user is authenticated, they can be seamlessly signed in and have access to the discourse portal. Alternatively users can also sign up by registering themselves in the portal.

❖ **Student View Enhancements:**

- **Cross-Platform Query Display:** When a ticket is created in the app, it also appears on the discourse platform, ensuring visibility and community engagement. During data exchange process via api Ticketing system shares description, user detail of the ticket owner, severity level. Once the ticket is closed in the ticketing system, the status of the respective thread is also updated in the discourse portal.
- **Advanced Search Capabilities:** Users can easily search for queries using tags and categories, enhancing the user experience by providing quick access to relevant information.

❖ **Support Agent View Enhancements:**

- **Integrated Communication:** Replies made in one platform (ticketing app or discourse) are synchronised across both, streamlining the resolution process.
- **Priority Notifications:** Urgent or high-priority queries trigger notifications via Google Chat, aiming for rapid responses and reduced resolution times.
- **Visualisation Tools:** A pie-chart visualisation of pending and resolved queries helps support agents manage their workload more effectively.

❖ **Admin and Manager View Enhancements:**

- **Security and Oversight:** Admins can monitor for suspicious activities and take action against malicious users, maintaining a safe and secure platform.
- **Resolution Oversight:** Managers have the capability to investigate unresolved tickets further, ensuring accountability and consistent support quality.
- **Insight from discourse:** Managers can have course/tag/timeline wise discourse thread counts enabling them to have a complete view of the overall activities, nature and severity of discussions going on in the portal.

❖ **Admin View Refinements:**

- **Proactive Security Measures:** Admins are equipped with tools to closely monitor the platform for any suspicious activities, ensuring a safe and secure environment for all users. This capability is crucial for maintaining user trust and preventing abuse.
- **Robust User Management:** The ability to dynamically manage users, including the addition of new users via email IDs and the removal of users based on their username, allows for efficient platform administration. This flexibility is essential for adapting to the evolving needs of the user base.

❖ **Manager View Refinements:**

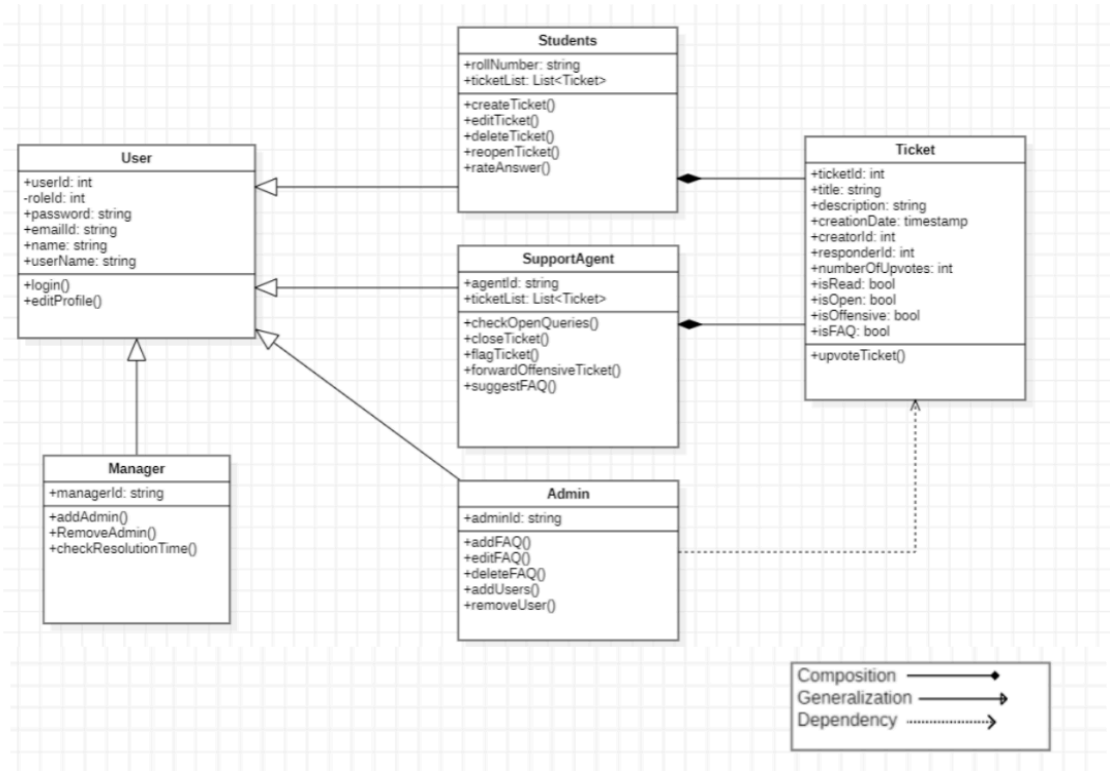
- **In-depth Ticket Resolution Oversight:** Managers possess enhanced tools to delve into tickets that remain unresolved, providing an additional layer of scrutiny and

ensuring that no query goes unanswered. This function underscores the commitment to high-quality support and customer satisfaction.

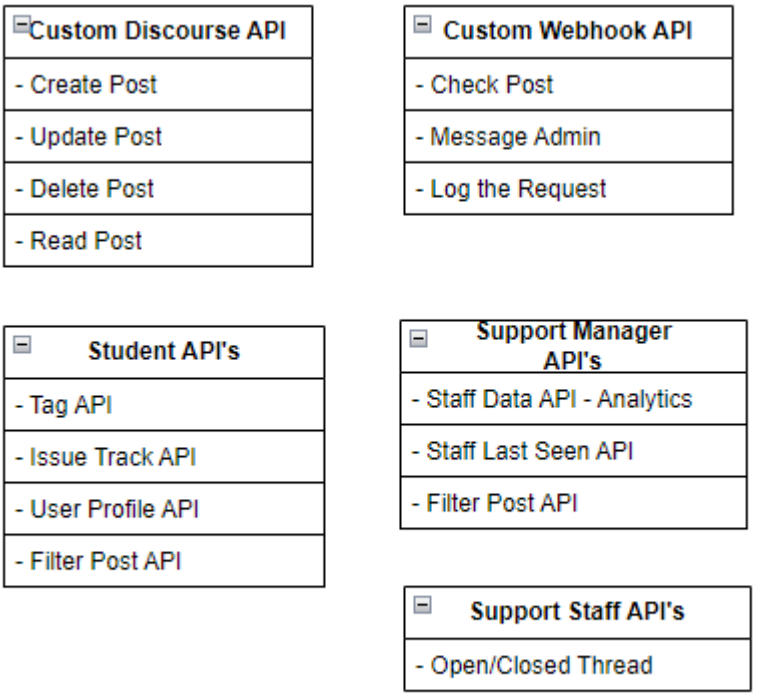
- **Performance Monitoring and Improvement:** Through scheduled cron jobs, managers can identify support agents with resolution times exceeding the desired thresholds, fostering a culture of accountability and continuous improvement. This system helps highlight areas for training and development, ensuring the team's performance aligns with service standards.

Software Design

❖ **Class Diagram:**



❖ **APIs:**



Sprint Schedule

- **Sprint 1:** Identify different users and write user stories for them.

  Deadline: 10/02/24 – 15/02/24

- **Sprint 2:** Create storyboards for different users and make wireframes for every user keeping the usability design and heuristics in mind.

  Deadline: 15/02/24 – 23/02/24

- **Sprint 3:** Design the software architecture of the app and schedule the implementation of user stories.

  Deadline: 29/02/24 – 05/03/24

- **Sprint 4:** Implementing different user stories and creating API endpoints for them and documenting all the API endpoints in a consolidated YAML file.

  Deadline: 06/03/24 – 15/03/24

- **Sprint 5:** Design test cases and suits to do unit, integration, system and acceptance testing.

  Deadline: 16/03/24 – 28/03/24

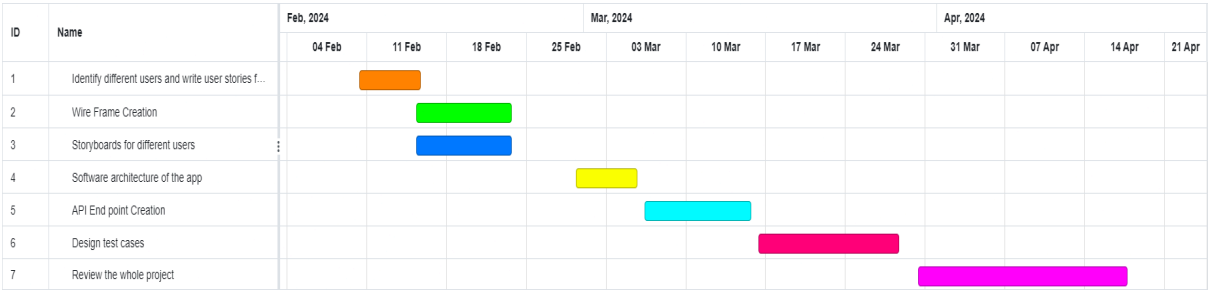
- **Sprint 6:** Review the whole project, find any bugs and fix them, creating a consolidated report on the whole project and making a presentation video of the app.

  Deadline: 30/03/24 – 17/04/24

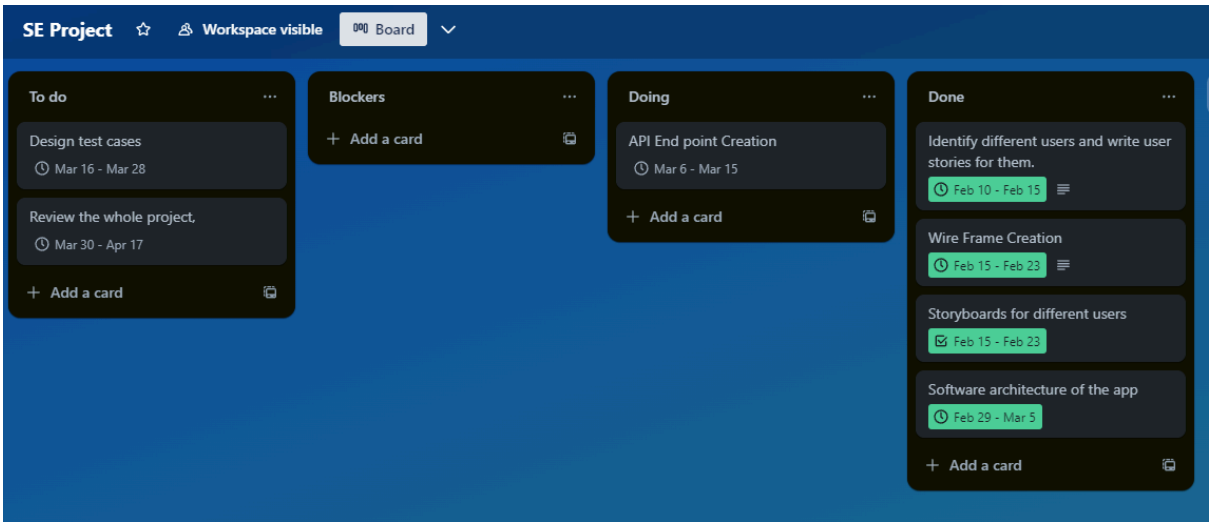
Minutes of Scrum Meetings

- **Scrum Meeting 1 (09/02/24):** This was an introductory meeting. Team members gathered and got to know each other and discussed how the team would proceed for the project.
- **Scrum Meeting 2 (13/02/24):** Team creates a roadmap to complete milestone 1 after discussing the problem statement thoroughly. We also identified different users of the app (Primary, Secondary, Tertiary). Everyone agreed to think and write user stories offline on a consolidated sheet.
- **Scrum Meeting 3 (15/02/24):** Team creates a roadmap to complete milestone 2. Different tasks have been assigned to different members. Team has discussed the structure of the storyboard and wireframe.
- **Scrum Meeting 6 (04/03/24):** Team creates a roadmap to complete milestone 3. We identified a tool to manage our project in our case is Trello. Different team members have been assigned for different tasks of milestone 3.

❖ Gantt Chart:



❖ Trello Board:



SOFTWARE ENGINEERING

(Course ID: BSCS3001)

Online Ticketing System with Discourse and Gchat Integration

Milestone 5

By

Anupam Kumar Jha (21f1004905)

Kevin Joshua T (22f1001410)

Pranav Wankhedkar (21f1000120)

Sachin Singh (21f1003251)

Ujit Kumar (21f3000786)

Utpal Dutta (21f2000524)

Milestone 5 : API Endpoints, Test Cases and Results

API being tested	API Description	Input	Expected Output	Actual Output	Result
/api/send_gchat_msg	Send message to Gchat if priority is beyond 5	Payload: {ticket_id: 3, priority_measure: 6, message: "...", webhook_url: ""}	Status code: 200, Message: "Message posted successfully."	Status code: 200, Message: "Message posted successfully."	Success
/api/send_gchat_msg	Send message to Gchat if priority is beyond 5	Payload: {priority_measure: 6, message: "...", webhook_url: "https://example.com/gchat/webhook"}	Status code: 400, Message: "Invalid request body..."	Status code: 400, Message: "Invalid request body..."	Success
/api/send_gchat_msg	Send message to Gchat if priority is beyond 5	Payload: {ticket_id: 3, priority_measure: 6, message: "...", webhook_url: "https://invalid-url"}	Status code: 500, Message: "Internal server error..."	Status code: 500, Message: "Internal server error..."	Success
/check_flagged_posts	Check for flagged posts and send message to Support Manager	Payload: {post_id: 123, flag_message: "...", user_id: 456, message: "...", webhook_url: "..."}	Status code: 200, Message: "Message sent to Support Manager..."	Status code: 200, Message: "Message sent to Support Manager..."	Success

/check_flagged_posts	Check for flagged posts and send message to Support Manager	Payload: {flag_message: "...", user_id: 456, message: "...", webhook_url: ""}	Status code: 400, Message: "Invalid request body..."	Status code: 400, Message: "Invalid request body..."	Success
/check_flagged_posts	Check for flagged posts and send message to Support Manager	Payload: {post_id: 123, flag_message: "...", user_id: 456, webhook_url: "https://invalid-url"}	Status code: 500, Message: "Internal server error..."	Status code: 500, Message: "Internal server error..."	Success
/submit_previous_discussion_links	Submit clickable links for a previous discussion.	Payload: {post_id: 123, discussion_links: ["...", "..."]}	Status code: 200, Message: "Previous discussion links submitted successfully."	Status code: 200, Message: "Previous discussion links submitted successfully."	Success
/submit_previous_discussion_links	Submit clickable links for a previous discussion.	Payload: {discussion_links: ["...", "..."]}	Status code: 400, Message: "Invalid request body..."	Status code: 400, Message: "Invalid request body..."	Success
/submit_previous_discussion_links	Submit clickable links for a previous discussion.	Payload: {post_id: 123, discussion_links: ["https://invalid-url"]}	Status code: 500, Message: "Internal server error..."	Status code: 500, Message: "Internal server error..."	Success
/api/search_discussions_by_tag	Search discussions based on a tag.	Tag: "python"	Status code: 200, Error message: None	Status code: 200	Success

/api/search_discussions_by_tag	Search discussions based on a tag.	Tag: "java"	Status code: 404, Error message: "No discussions found for the provided tag."	Status code: 404, Error message: "No discussions found for the provided tag."	Success
/api/search_discussions_by_tag	Search discussions based on a tag.	Tag: ""	Status code: 400, Error message: "Invalid request. Please provide a valid tag."	Status code: 400, Error message: "Invalid request. Please provide a valid tag."	Success
/api/search_discussions_by_tag	Search discussions based on a tag.	Tag: "invalid_tag"	Status code: 400, Error message: "Invalid request. Please provide a valid tag."	Status code: 400, Error message: "Invalid request. Please provide a valid tag."	Success
/profile	Provides details of a specific user by taking user id as parameter.	User ID: 1001	Status code: 200, Message: None	Status code: 200	Success
/profile	Provides details of a specific user by taking user id as parameter.	User ID: None	Status code: 403, Error message: "Please provide a user id."	Status code: 403, Error message: "Please provide a user id."	Success
/profile	Provides details of a specific user by taking user id as parameter.	User ID: 9999	Status code: 404, Error message: "There are no user with user id 9999"	Status code: 404, Error message: "There are no user with user id 9999"	Success
/api/search_discussions_by_category	Filter based on category and tag.	Category: "python", Tag: "quiz-2"	Status code: 200, Message: None	Status code: 200	Success

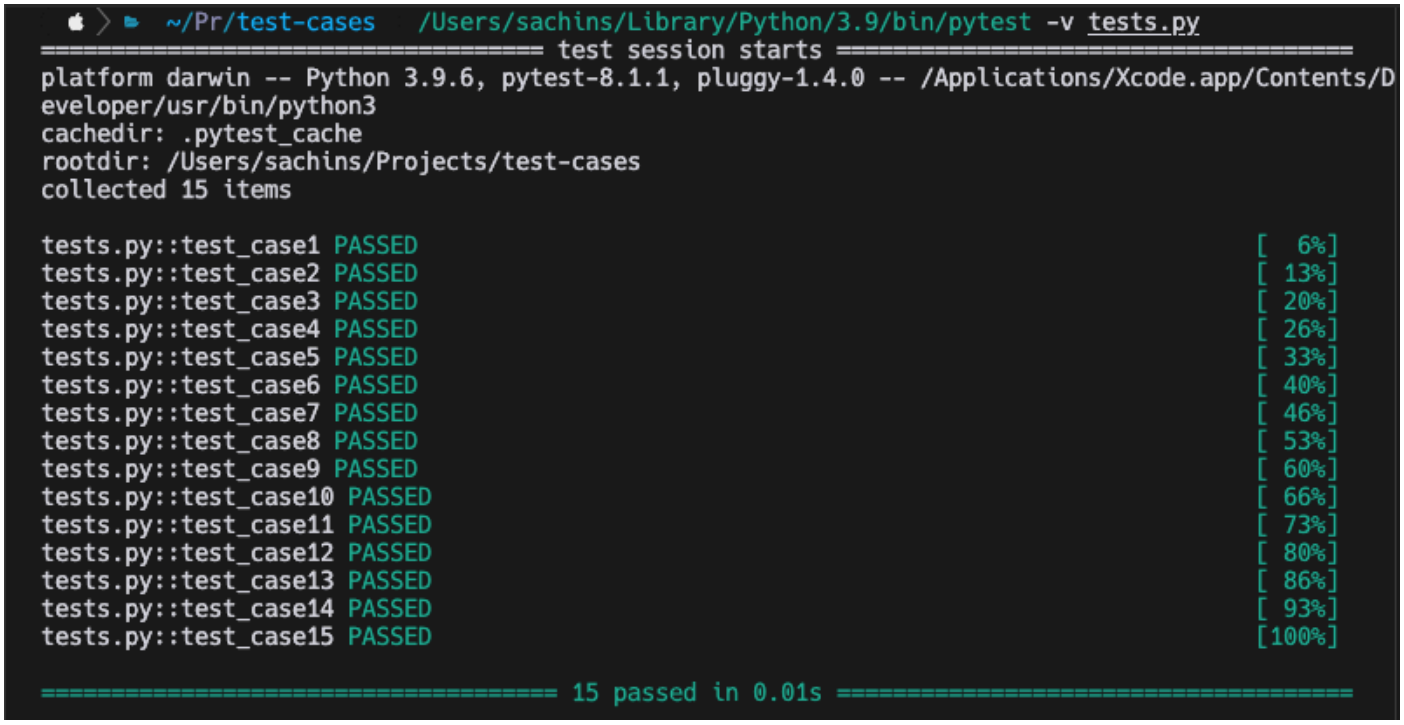
/api/search_discussions_by_category	Filter based on category and tag.	Category: "", Tag: ""	Status code: 400, Error message: "Invalid request. Please provide a valid category."	Status code: 400, Error message: "Invalid request. Please provide a valid category."	Success
/api/search_discussions_by_category	Filter based on category and tag.	Category: "nonexistent_category", Tag: "quiz-2"	Status code: 404, Error message: "No discussions found for the provided category."	Status code: 404, Error message: "No discussions found for the provided category."	Success
/api/search_discussions_by_category	Filter based on category and tag.	Category: "python", Tag: "nonexistent_tag"	Status code: 404, Error message: "No discussions found for the provided tag."	Status code: 404, Error message: "No discussions found for the provided tag."	Success
/api/search_discussions_by_category	Filter based on category and tag.	Category: "", Tag: "nonexistent_tag"	Status code: 400, Error message: "Invalid request. Please provide a valid category."	Status code: 400, Error message: "Invalid request. Please provide a valid category."	Success
/api/search_discussions_by_category	Filter based on category and tag.	Category: "nonexistent_category", Tag: "nonexistent_tag"	Status code: 404, Error message: "No discussions found for the provided category."	Status code: 404, Error message: "No discussions found for the provided category."	Success
/support_staff/activity_status	Get the list of all support staff members which are not responding.	N/A	Status code: 200, Response is a list of staff members with: username, isActive, lastActiveTime	Status code: 200, Response is a list of staff members with: username, isActive, lastActiveTime	Success
/support_staff/activity_status	Get the list of all support staff members which are not	Invalid API key	Status code: 401, Error message: "Unauthorized"	Status code: 401, Error message: "Unauthorized"	Success

	responding.				
/posts	Tag the concerned person.	Title: "Test Post", Description: "This is a test post", Tags: ["user1", "user2"]	Status code: 201	Status code: 201	Success
/posts	Tag the concerned person.	Missing title and description fields	Status code: 400	Status code: 400	Success
/search_users	Search for users to tag.	Search term: "user"	Status code: 200, Response contains data	Status code: 200, Response contains data	Success
/support_staff/activity_status	Get the list of all support staff members which are not responding.	Invalid API key	Status code: 401, Error message: "Unauthorized"	Status code: 401, Error message: "Unauthorized"	Success
/posts	Tag the concerned person.	Title: "Test Post", Description: "This is a test post", Tags: ["user1", "user2"]	Status code: 201	Status code: 201	Success
/posts	Tag the concerned person.	Missing title and description fields	Status code: 400	Status code: 400	Success

/profile	Provides details of a specific user by taking user id as parameter.	User ID: 1004	Output name should not be an integer	Output name is not an integer	Success
/profile	Provides details of a specific user by taking user id as parameter.	User ID: 1004	isspam: True, num_posts: > 14	isspam: True, num_posts: 15	Success
/profile	Provides details of a specific user by taking user id as parameter.	User ID: 1005	isspam: False, num_posts: 10	isspam: False, num_posts: 10	Success
/posts/{id}/locked.json	Lock a post from being edited	API key: your-api-key, API username: your-api-username, Post ID: 123	Status code: 200, Response contains "locked" field set to True	Status code: 200, Response contains "locked" field set to True	Success
/posts.json	Api for creating a query or replying on a query on discourse	Valid payload: Dictionary containing post data	Status code: 200, Response contains various fields related to the created post	Status code: 200, Response contains various fields related to the created post	Success
/search_users	Search for users to tag.	Search term: "user"	Status code: 200, Response contains data	Status code: 200, Response contains data	Success

/profile	Provides details of a specific user by taking user id as parameter.	User ID: 1004	Output name should not be an integer	Output name is not an integer	Success
/profile	Provides details of a specific user by taking user id as parameter.	User ID: 1004	isspam: True, num_posts: > 14	isspam: True, num_posts: 15	Success
/profile	Provides details of a specific user by taking user id as parameter.	User ID: 1005	isspam: False, num_posts: 10	isspam: False, num_posts: 10	Success
/posts/{id}/locked.json	Lock a post from being edited	API key: your-api-key, API username: your-api-username, Post ID: 123	Status code: 200, Response contains "locked" field set to True	Status code: 200, Response contains "locked" field set to True	Success
/posts.json	Api for creating a query or replying on a query on discourse	Valid payload: Dictionary containing post data	Status code: 200, Response contains various fields related to the created post	Status code: 200, Response contains various fields related to the created post	Success

Test Cases Result:


~/Pr/test-cases /Users/sachins/Library/Python/3.9/bin/pytest -v tests.py
test session starts
platform darwin -- Python 3.9.6, pytest-8.1.1, pluggy-1.4.0 -- /Applications/Xcode.app/Contents/Developer/usr/bin/python3
cachedir: .pytest_cache
rootdir: /Users/sachins/Projects/test-cases
collected 15 items
tests.py::test_case1 PASSED [6%]
tests.py::test_case2 PASSED [13%]
tests.py::test_case3 PASSED [20%]
tests.py::test_case4 PASSED [26%]
tests.py::test_case5 PASSED [33%]
tests.py::test_case6 PASSED [40%]
tests.py::test_case7 PASSED [46%]
tests.py::test_case8 PASSED [53%]
tests.py::test_case9 PASSED [60%]
tests.py::test_case10 PASSED [66%]
tests.py::test_case11 PASSED [73%]
tests.py::test_case12 PASSED [80%]
tests.py::test_case13 PASSED [86%]
tests.py::test_case14 PASSED [93%]
tests.py::test_case15 PASSED [100%]
15 passed in 0.01s

Sample Test Cases:

```python
import pytest
import requests

base_url = 'http://127.0.0.1:5000'


@pytest.mark.parametrize("payload, expected_status, expected_message", [
    ({
        'ticket_id': 3,
        'priority_measure': 6,
        'message': 'Priority measure exceeded 5. Immediate action required!',
        'webhook_url': 'https://chat.googleapis.com/v1/spaces/AAAAaBileJo/messages?key=AIzaS
    }, 200, 'Message posted successfully.'),
    ({
        'priority_measure': 6,
        'message': 'Priority measure exceeded 5. Immediate action required!',
        'webhook_url': 'https://example.com/gchat/webhook'
    }, 400, 'Invalid request body. Please provide valid ticket ID and priority measure.'),
    ({
        'ticket_id': 3,
        'priority_measure': 6,
        'message': 'Priority measure exceeded 5. Immediate action required!',
        'webhook_url': 'https://invalid-url'
    }, 500, 'Internal server error. Please try again later.')
])
def test_send_gchat_msg(payload, expected_status, expected_message):
    response = requests.post(f'{base_url}/api/send_gchat_msg', json=payload)
    assert response.status_code == expected_status
    assert response.json().get('message') == expected_message
```

```python
@pytest.fixture
def valid_payload():
    return {
        "title": "Test Post",
        "raw": "This is a test post.",
        "topic_id": 123,
        "category": 456,
        "created_at": "2024-04-04T12:00:00Z",
        "reply_to_post_number": None
    }
```

```python
def test_create_post_with_tags(client):
    # Test creating a new post with tags
    post_data = {
        "title": "Test Post",
        "description": "This is a test post",
        "tags": ["user1", "user2"]
    }
    response = client.post('/posts', json=post_data)
    assert response.status_code == 201

def test_create_post_missing_fields(client):
    # Test creating a post with missing fields
    post_data = {
        "description": "This is a test post",
        "tags": ["user1", "user2"]
    }
    response = client.post('/posts', json=post_data)
    assert response.status_code == 400

def test_search_users(client):
    # Test searching for users
    response = client.get('/search_users?search_term=user')
    assert response.status_code == 200
    data = json.loads(response.data)
    assert len(data) > 0

def test_search_users_no_results(client):
    # Test searching for users with no results
    response = client.get('/search_users?search_term=nonexistentuser')
    assert response.status_code == 404
```