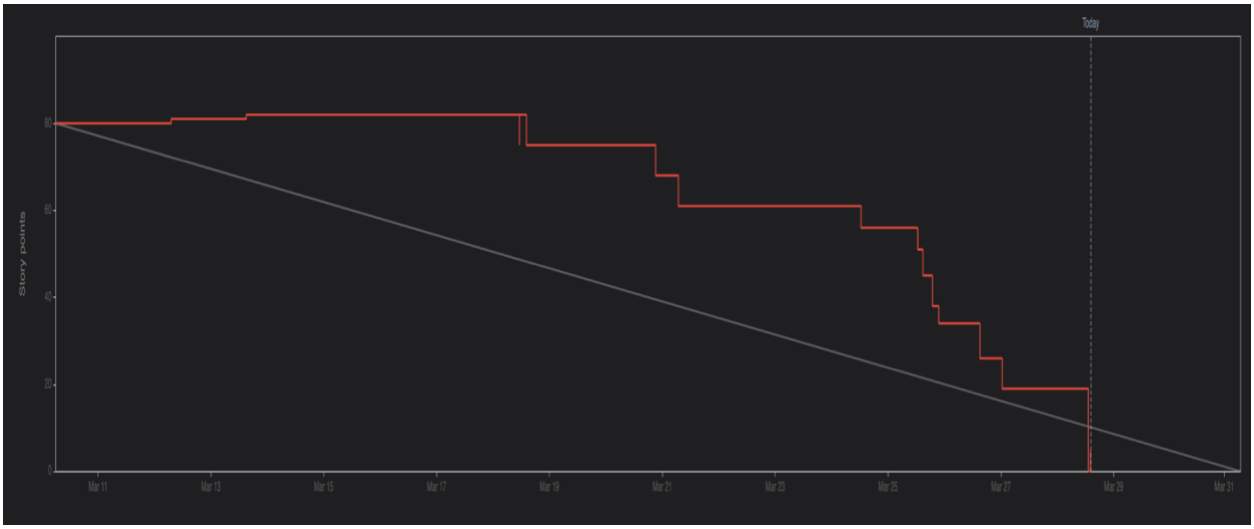
UFCFTR-30-3
Distribute & Enterprise Software Development

Sprint Review Form

Group:	19
Sprint:	2
Members:	Rohaan Aslam (21017718) Arthur Milner (21035478) William Barnes (21031340) Seif Mansour (23012749) Chaya Moore (21030599)

Burn-down Chart

Below is the Jira generated burn-down chart.

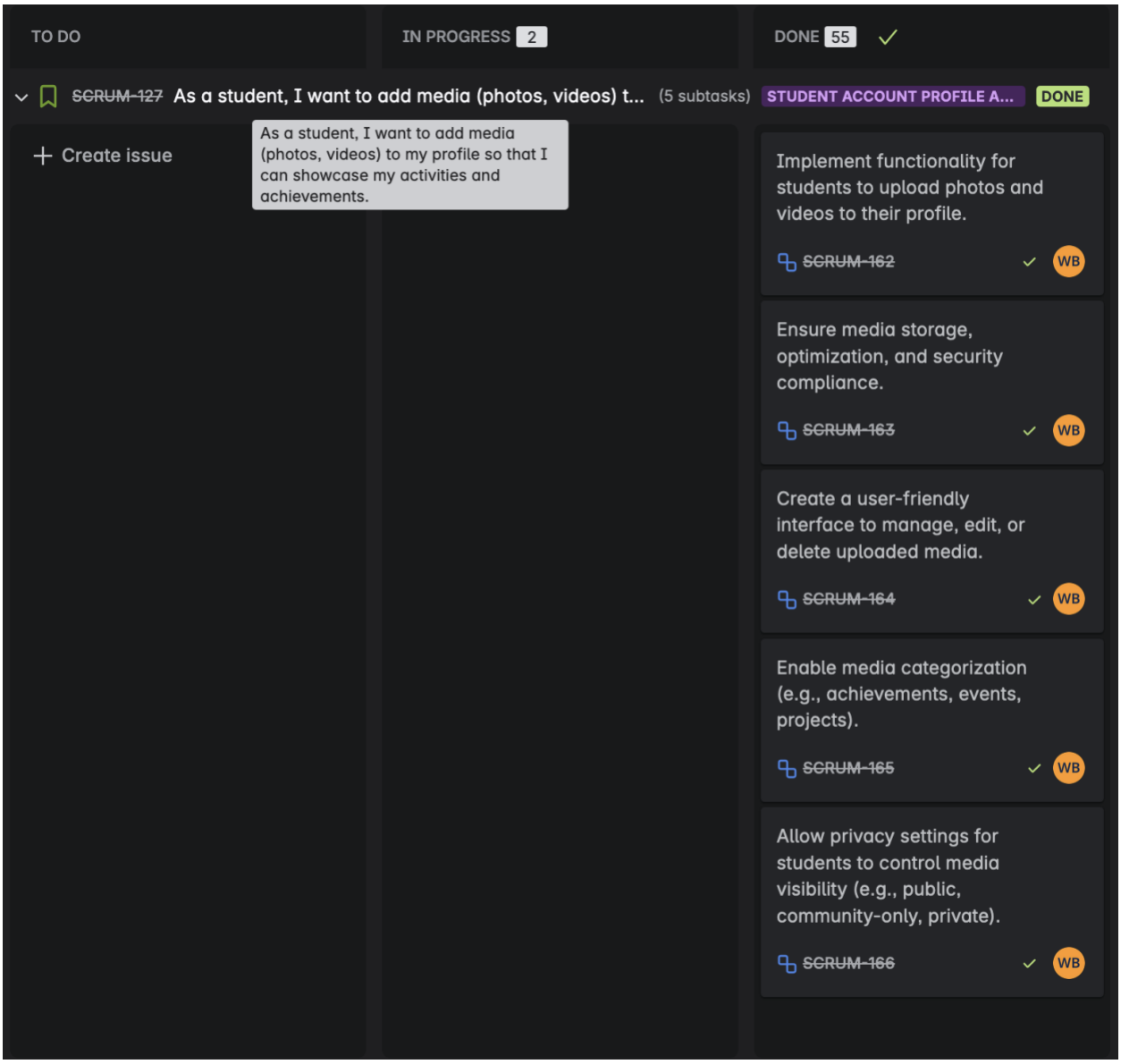


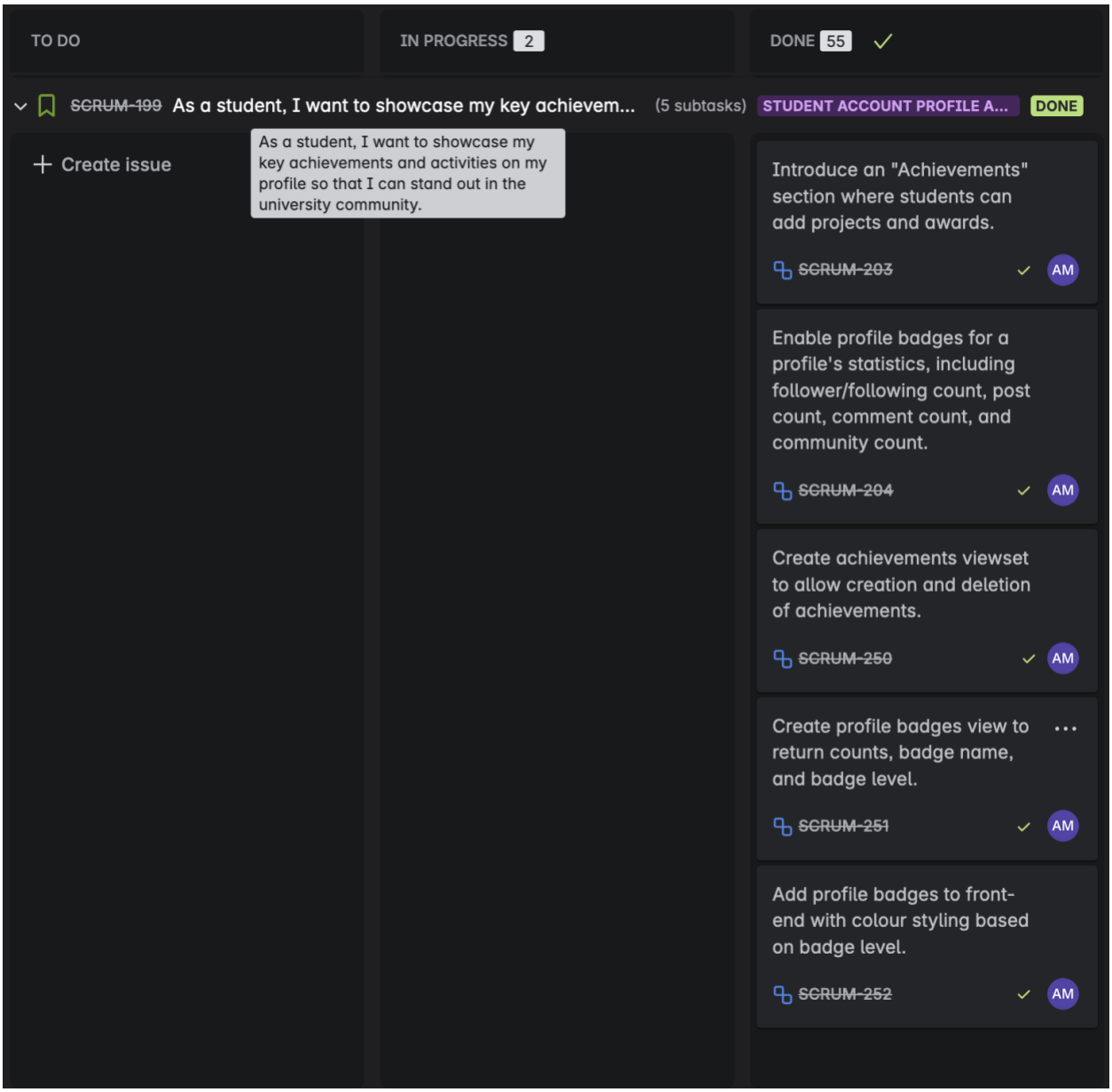
Project Schedule and Backlog

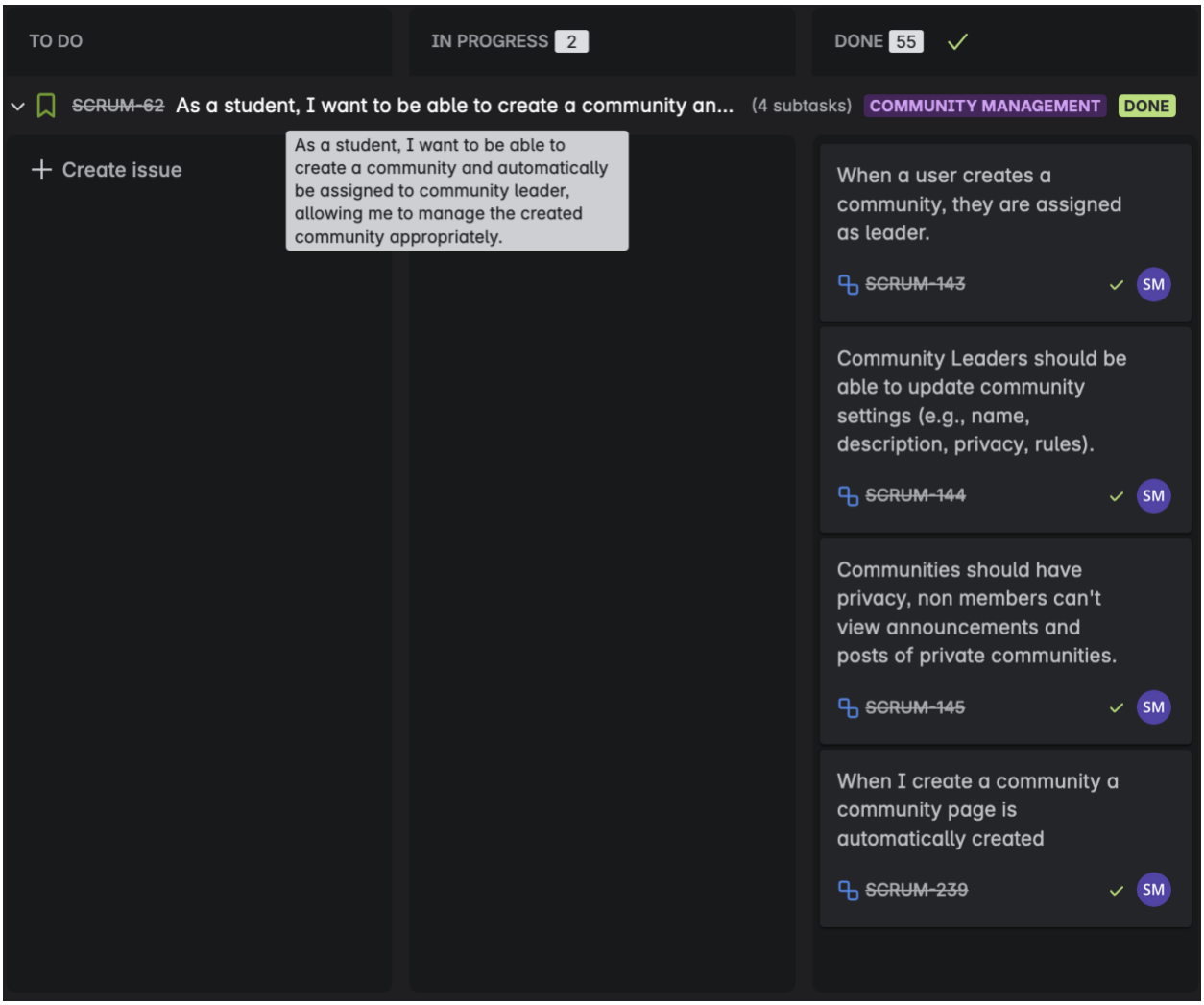
Our backlog consists of epics, which contain user stories, which then contain multiple sub-tasks which are assigned to a single group member.

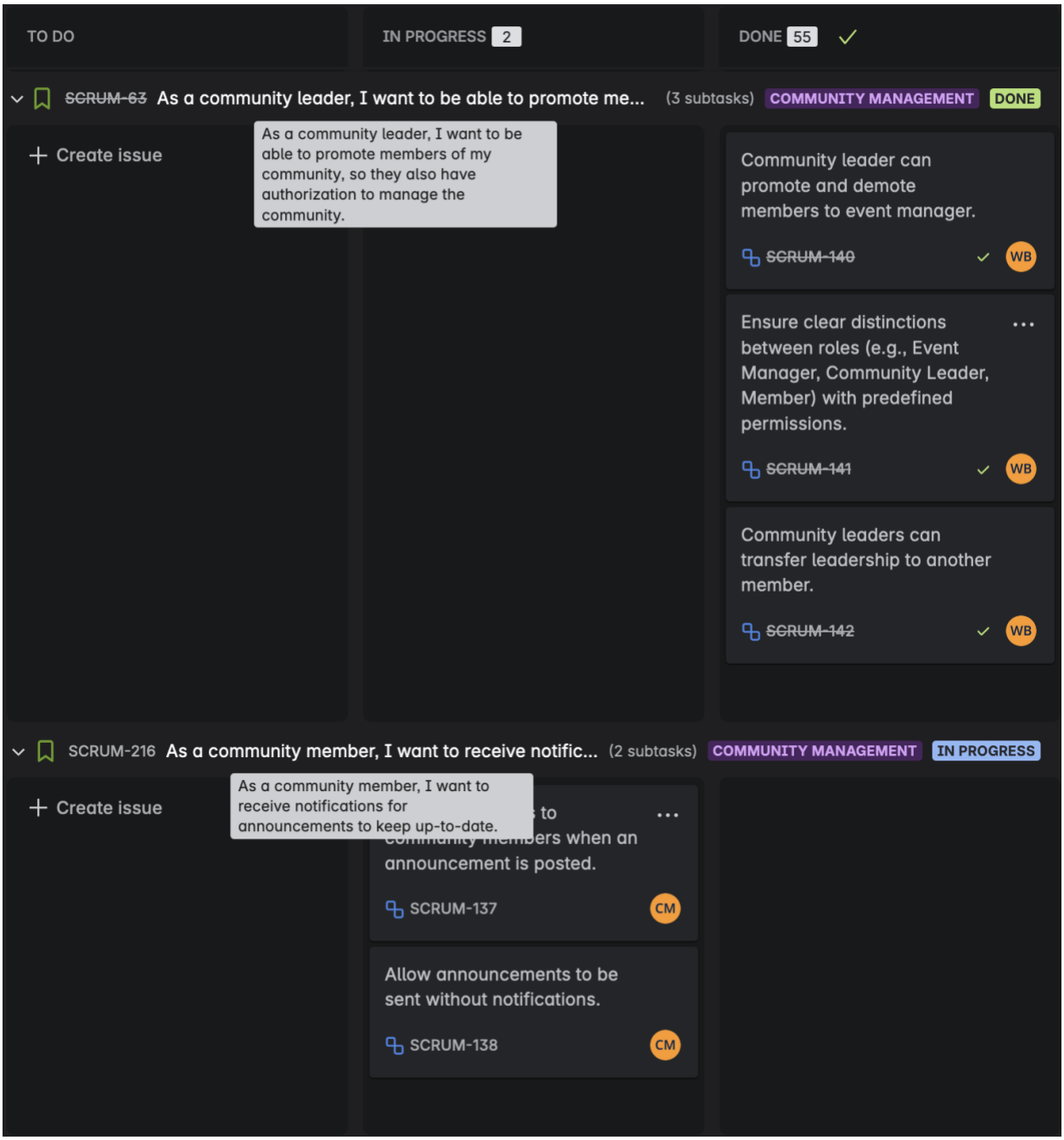
Sprint 2 Kanban Board:

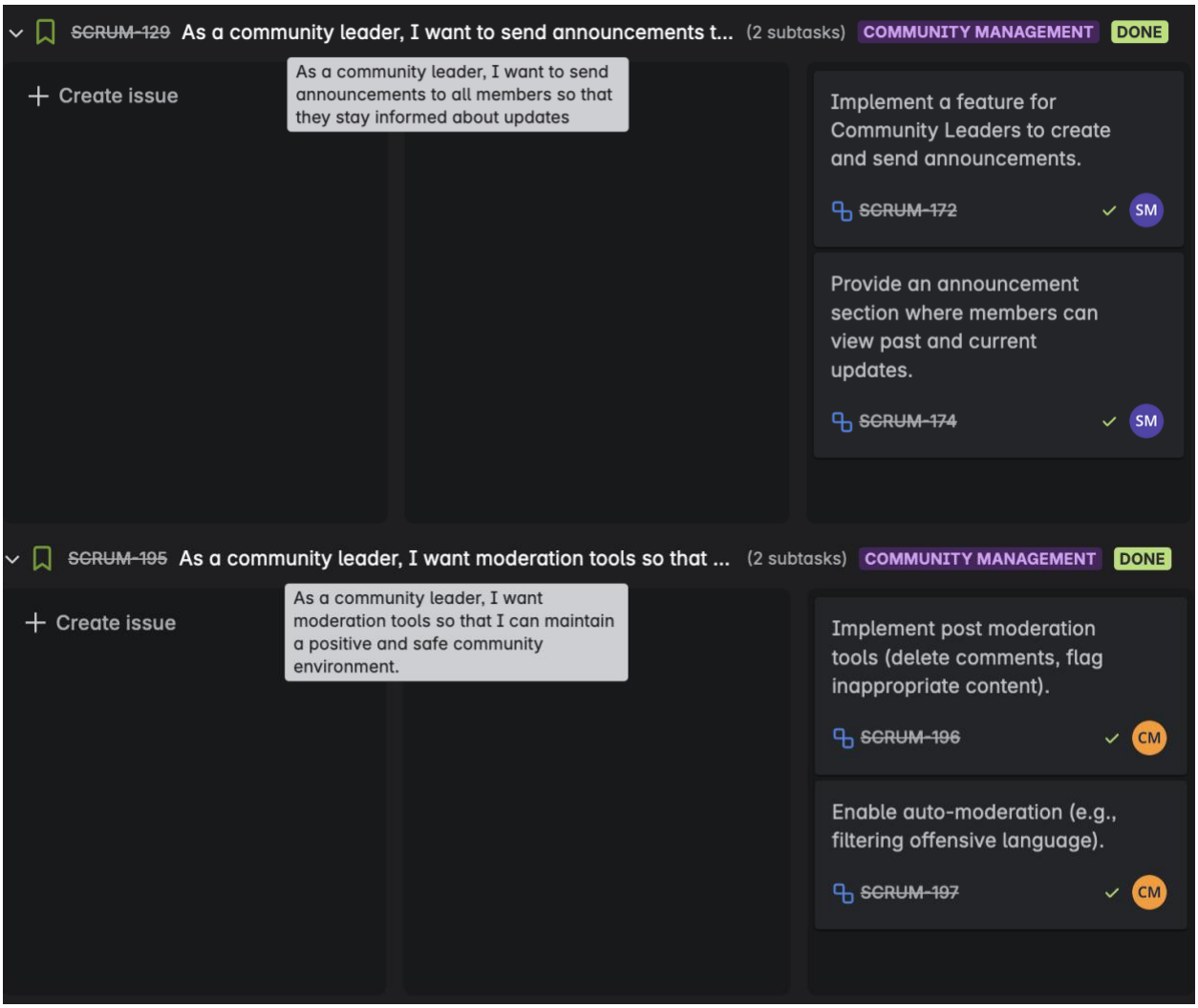
Below is the kanban board for sprint 2, the assignee for each subtask can be seen in the avatar, with the letters representing the group member's initials. The purple text represents the epic, the  icon represents a user story, whilst the  icon represents a subtask.

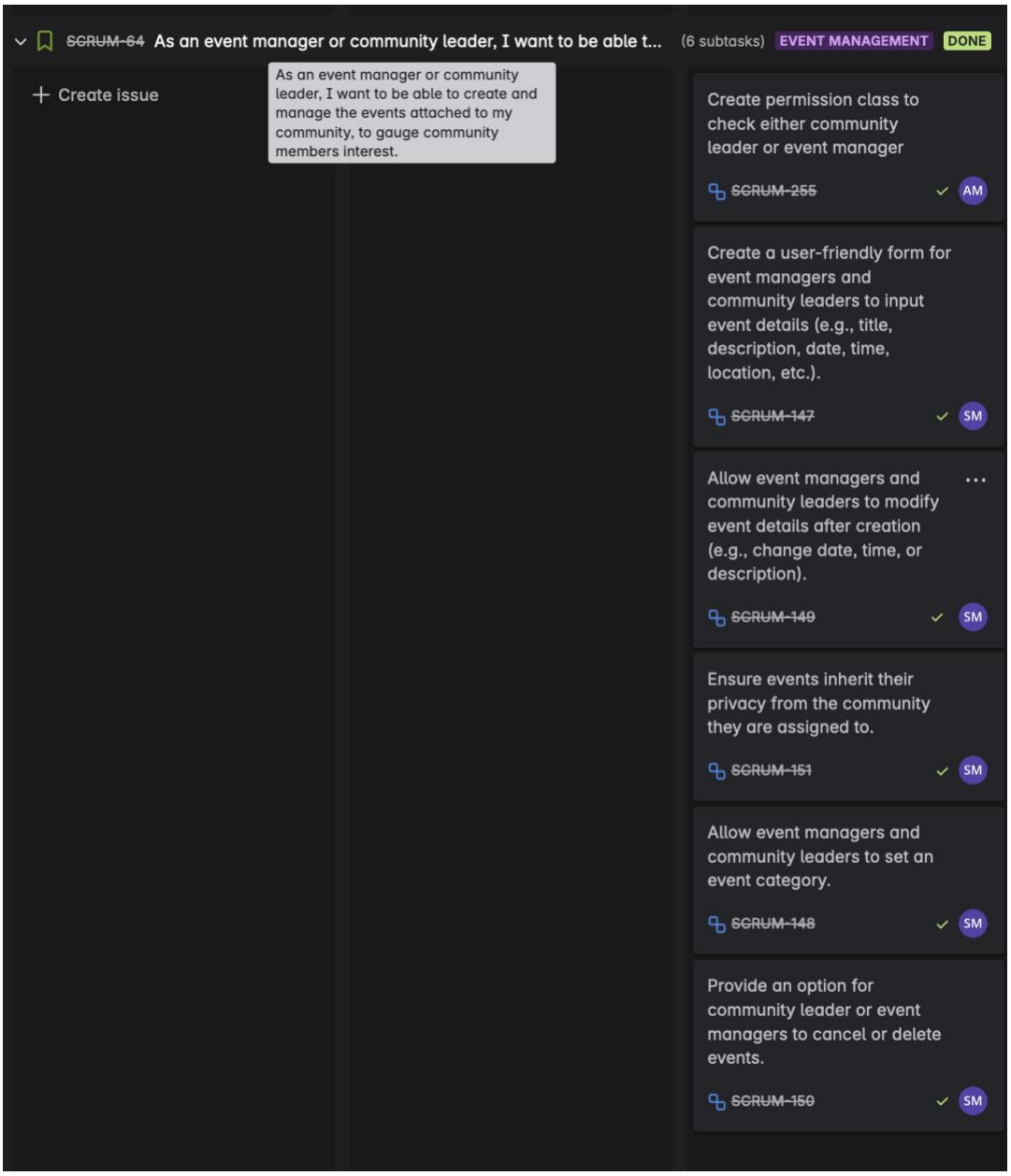

TO DO
IN PROGRESS 2
DONE 55
SCRUM-127 As a student, I want to add media (photos, videos) t... (5 subtasks) STUDENT ACCOUNT PROFILE A... DONE
Create issue
As a student, I want to add media (photos, videos) to my profile so that I can showcase my activities and achievements.
Implement functionality for students to upload photos and videos to their profile.
SCRUM-162 WB
Ensure media storage, optimization, and security compliance.
SCRUM-163 WB
Create a user-friendly interface to manage, edit, or delete uploaded media.
SCRUM-164 WB
Enable media categorization (e.g., achievements, events, projects).
SCRUM-165 WB
Allow privacy settings for students to control media visibility (e.g., public, community-only, private).
SCRUM-166 WB

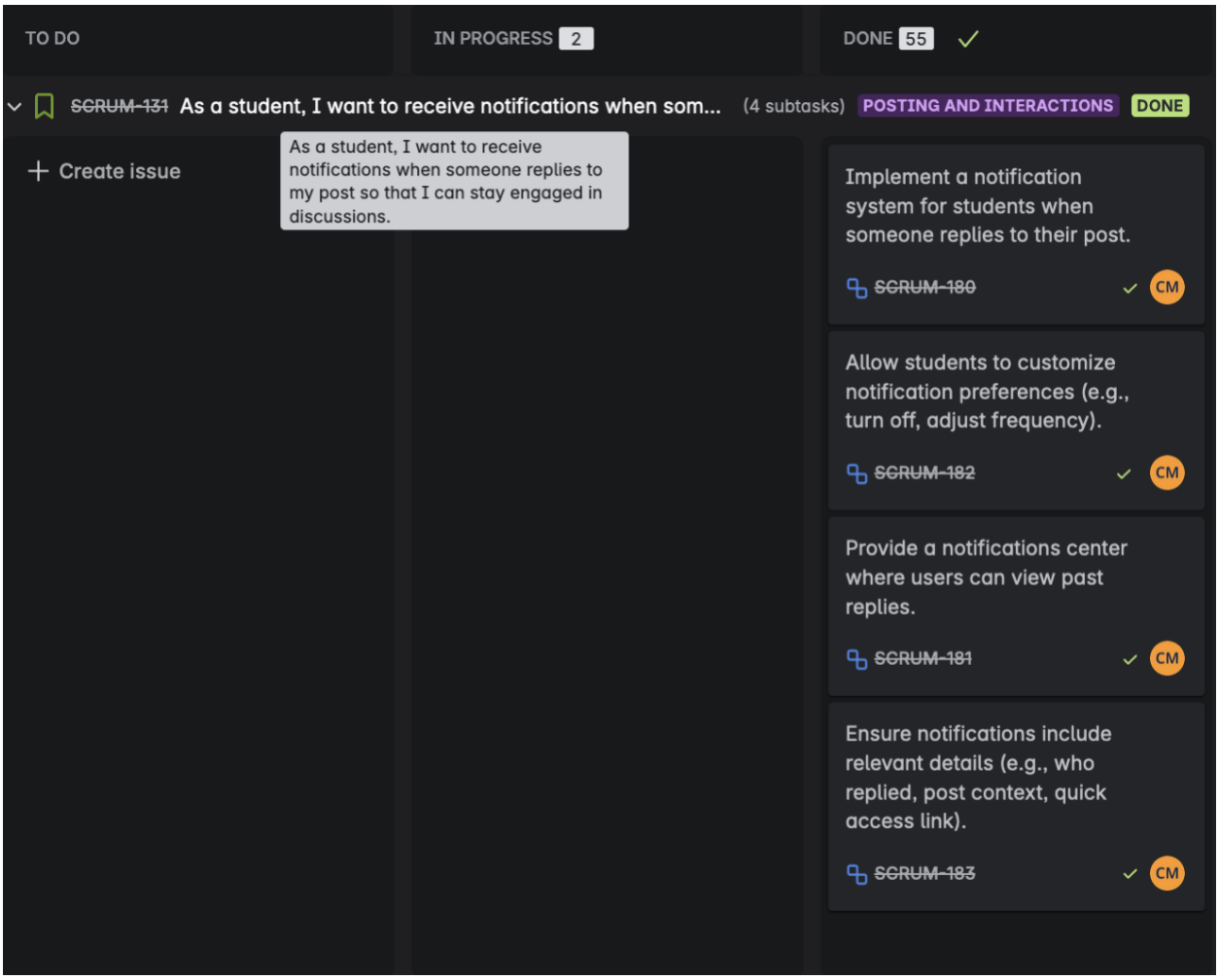

TO DO
IN PROGRESS 2
DONE 55
~~SCRUM-199~~ As a student, I want to showcase my key achievem...
(5 subtasks)
STUDENT ACCOUNT PROFILE A...
DONE
Create issue
As a student, I want to showcase my key achievements and activities on my profile so that I can stand out in the university community.
Introduce an "Achievements" section where students can add projects and awards.
~~SCRUM-203~~
AM
Enable profile badges for a profile's statistics, including follower/following count, post count, comment count, and community count.
~~SCRUM-204~~
AM
Create achievements viewset to allow creation and deletion of achievements.
~~SCRUM-250~~
AM
Create profile badges view to return counts, badge name, and badge level.
~~SCRUM-251~~
AM
Add profile badges to front-end with colour styling based on badge level.
~~SCRUM-252~~
AM

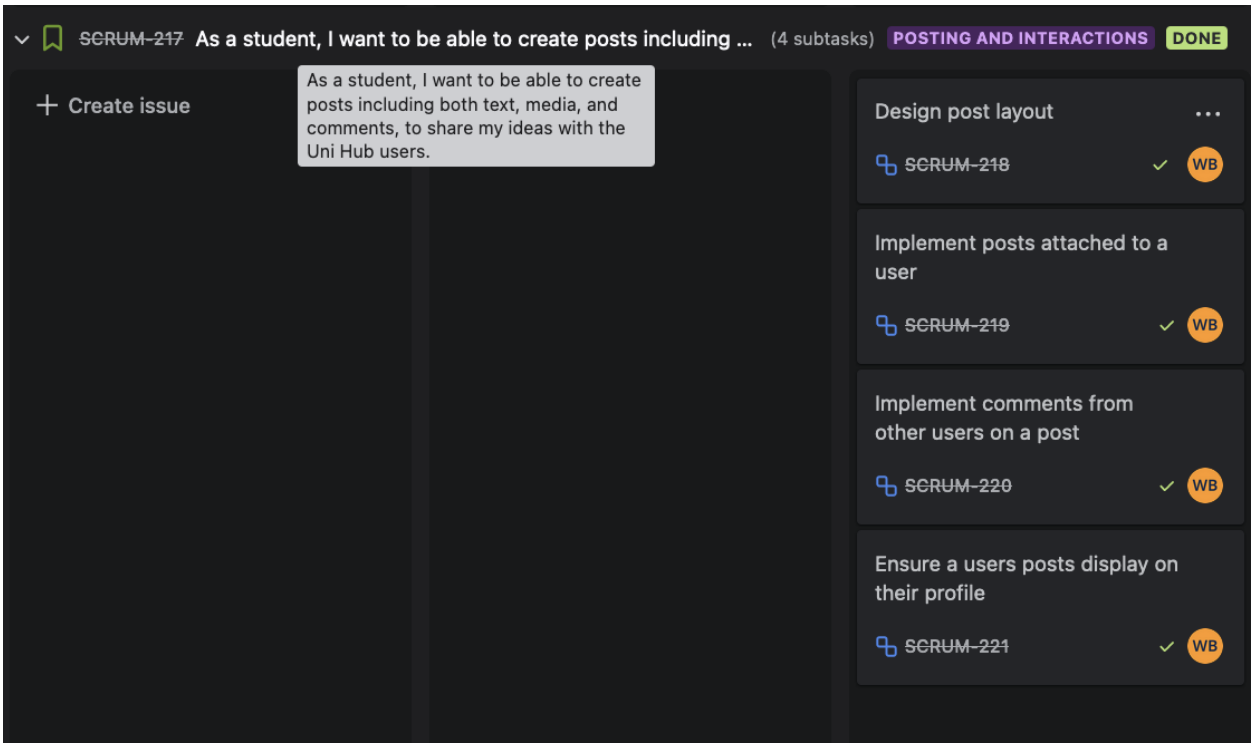

TO DO
IN PROGRESS 2
DONE 55
SCRUM-62 As a student, I want to be able to create a community an... (4 subtasks) COMMUNITY MANAGEMENT DONE
Create issue
As a student, I want to be able to create a community and automatically be assigned to community leader, allowing me to manage the created community appropriately.
When a user creates a community, they are assigned as leader.
SCRUM-143
SM
Community Leaders should be able to update community settings (e.g., name, description, privacy, rules).
SCRUM-144
SM
Communities should have privacy, non members can't view announcements and posts of private communities.
SCRUM-145
SM
When I create a community a community page is automatically created
SCRUM-239
SM

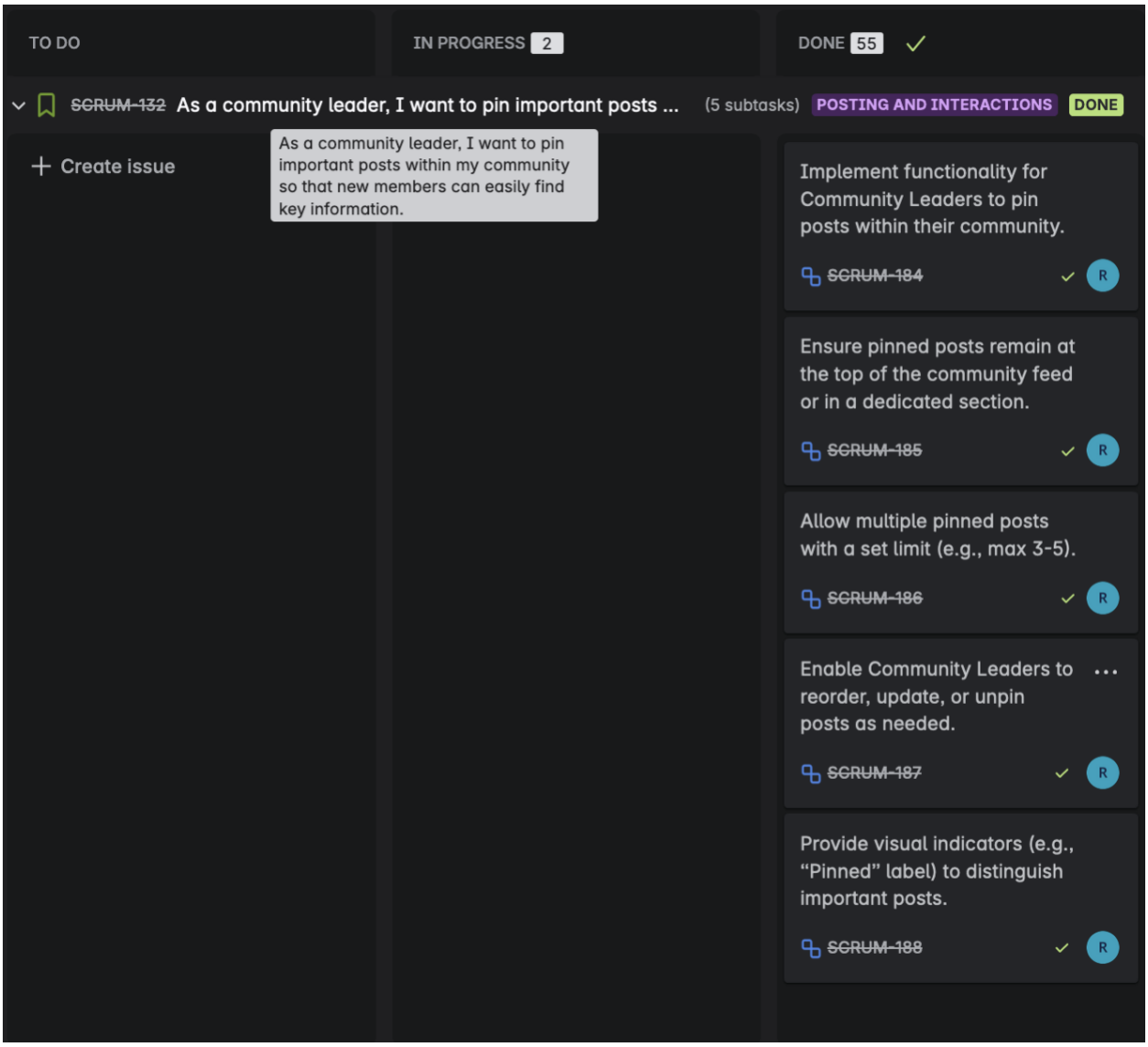

TO DO
IN PROGRESS 2
DONE 55
~~SCRUM-63~~ As a community leader, I want to be able to promote me...
(3 subtasks)
COMMUNITY MANAGEMENT
DONE
Create issue
As a community leader, I want to be able to promote members of my community, so they also have authorization to manage the community.
Community leader can promote and demote members to event manager.
~~SCRUM-140~~
WB
Ensure clear distinctions between roles (e.g., Event Manager, Community Leader, Member) with predefined permissions.
~~SCRUM-141~~
WB
Community leaders can transfer leadership to another member.
~~SCRUM-142~~
WB
SCRUM-216 As a community member, I want to receive notific...
(2 subtasks)
COMMUNITY MANAGEMENT
IN PROGRESS
Create issue
As a community member, I want to receive notifications for announcements to keep up-to-date.
announcement is posted.
SCRUM-137
CM
Allow announcements to be sent without notifications.
SCRUM-138
CM

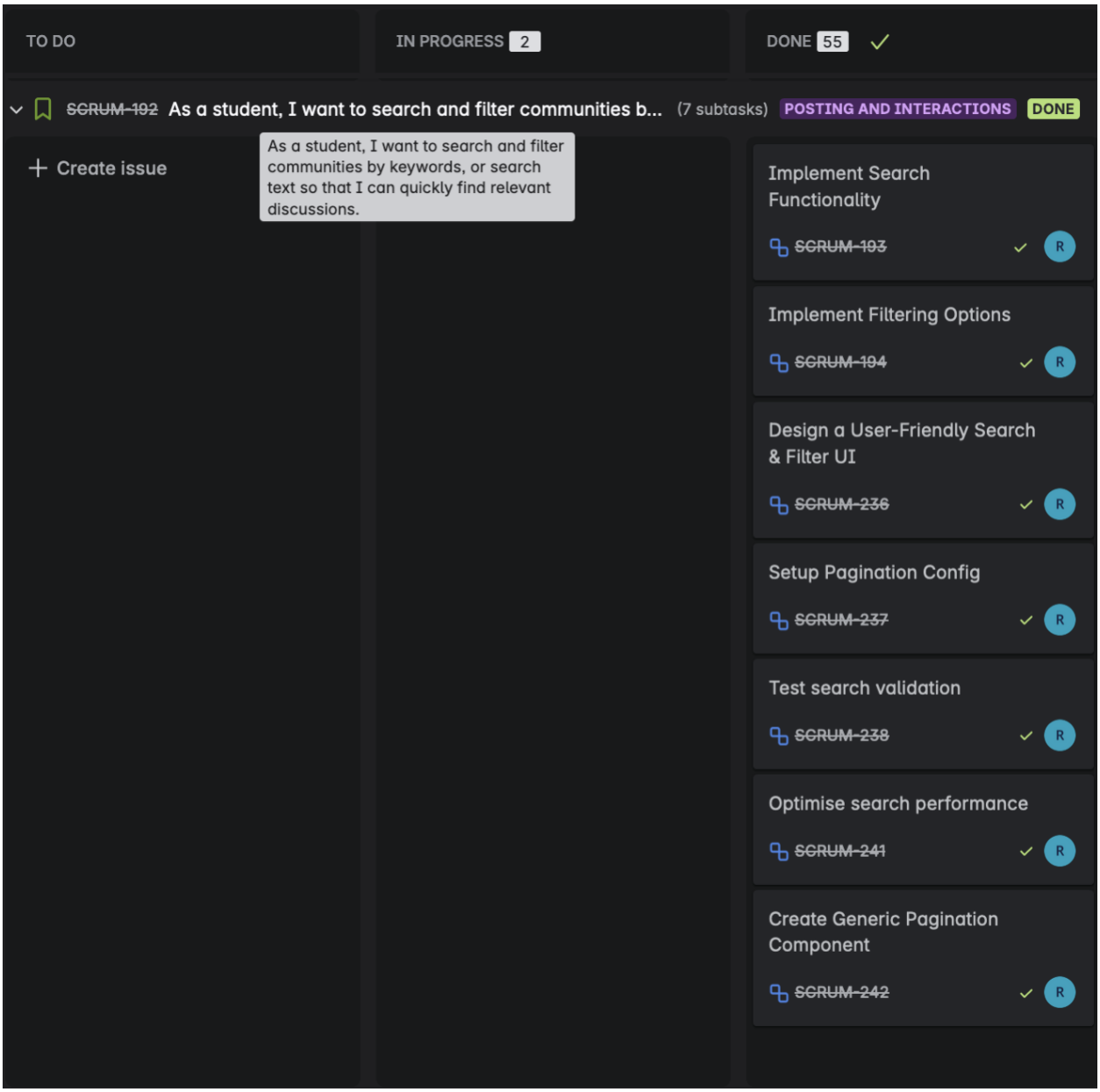

SCRUM-129 As a community leader, I want to send announcements t... (2 subtasks) COMMUNITY MANAGEMENT DONE
Create issue
As a community leader, I want to send announcements to all members so that they stay informed about updates
Implement a feature for Community Leaders to create and send announcements.
SCRUM-172
SM
Provide an announcement section where members can view past and current updates.
SCRUM-174
SM
SCRUM-195 As a community leader, I want moderation tools so that ... (2 subtasks) COMMUNITY MANAGEMENT DONE
Create issue
As a community leader, I want moderation tools so that I can maintain a positive and safe community environment.
Implement post moderation tools (delete comments, flag inappropriate content).
SCRUM-196
CM
Enable auto-moderation (e.g., filtering offensive language).
SCRUM-197
CM

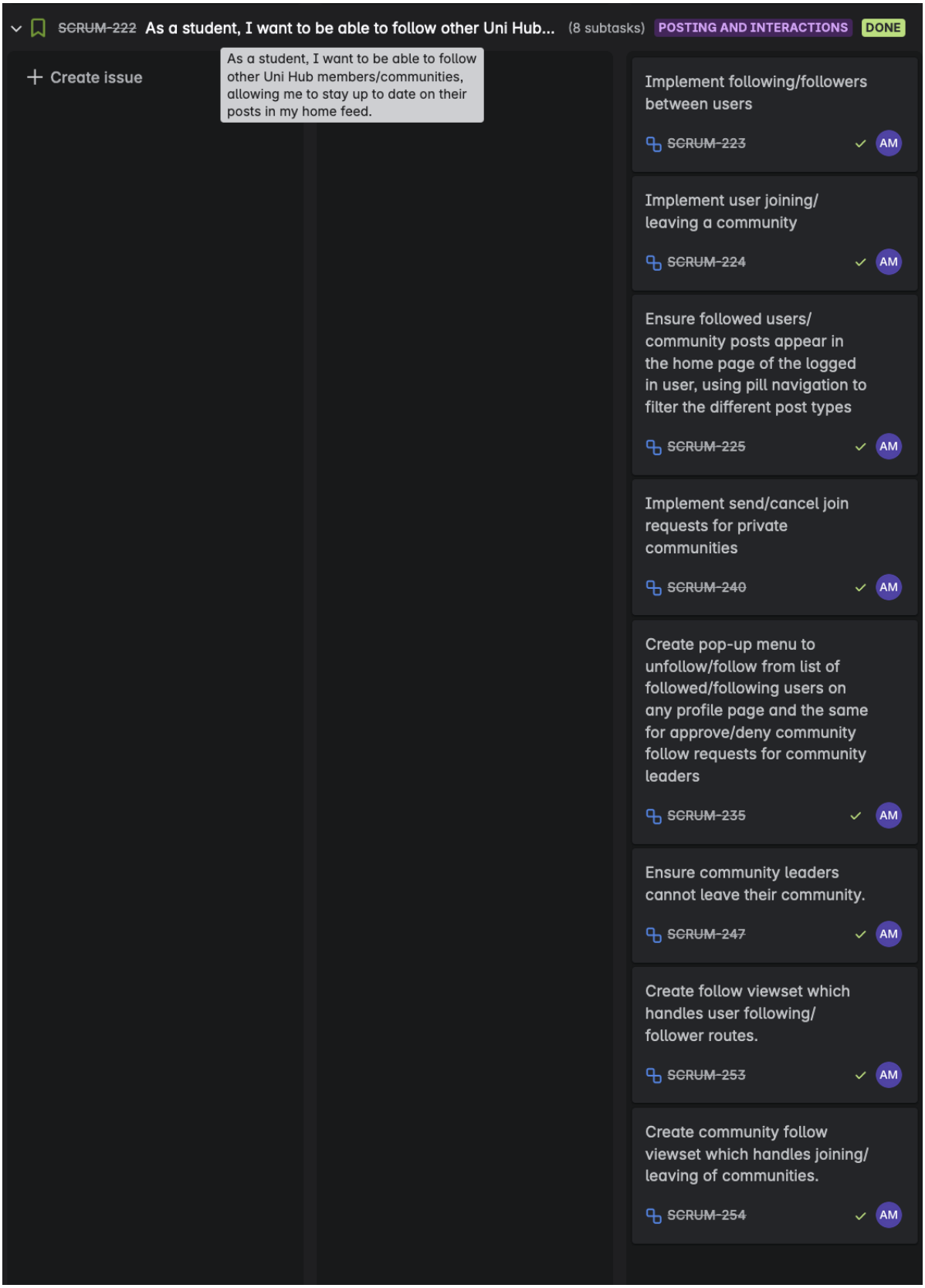

SCRUM-64 As an event manager or community leader, I want to be able t...
(6 subtasks)
EVENT MANAGEMENT
DONE
Create issue
As an event manager or community leader, I want to be able to create and manage the events attached to my community, to gauge community members interest.
Create permission class to check either community leader or event manager
SCRUM-255
AM
Create a user-friendly form for event managers and community leaders to input event details (e.g., title, description, date, time, location, etc.).
SCRUM-147
SM
Allow event managers and community leaders to modify event details after creation (e.g., change date, time, or description).
SCRUM-149
SM
Ensure events inherit their privacy from the community they are assigned to.
SCRUM-151
SM
Allow event managers and community leaders to set an event category.
SCRUM-148
SM
Provide an option for community leader or event managers to cancel or delete events.
SCRUM-150
SM


TO DO
IN PROGRESS 2
DONE 55
~~SCRUM-131~~ As a student, I want to receive notifications when som...
(4 subtasks)
POSTING AND INTERACTIONS
DONE
Create issue
As a student, I want to receive notifications when someone replies to my post so that I can stay engaged in discussions.
Implement a notification system for students when someone replies to their post.
~~SCRUM-180~~
CM
Allow students to customize notification preferences (e.g., turn off, adjust frequency).
~~SCRUM-182~~
CM
Provide a notifications center where users can view past replies.
~~SCRUM-181~~
CM
Ensure notifications include relevant details (e.g., who replied, post context, quick access link).
~~SCRUM-183~~
CM


~~SCRUM-217~~ As a student, I want to be able to create posts including ... (4 subtasks) POSTING AND INTERACTIONS DONE
Create issue
As a student, I want to be able to create posts including both text, media, and comments, to share my ideas with the Uni Hub users.
Design post layout
~~SCRUM-218~~
WB
Implement posts attached to a user
~~SCRUM-219~~
WB
Implement comments from other users on a post
~~SCRUM-220~~
WB
Ensure a users posts display on their profile
~~SCRUM-221~~
WB

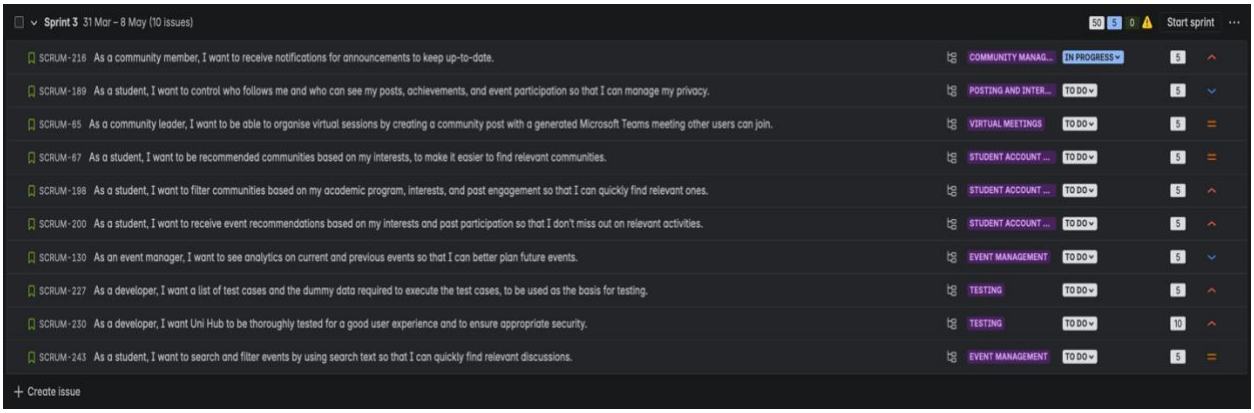

TO DO
IN PROGRESS 2
DONE 55
SCRUM-132 As a community leader, I want to pin important posts ...
(5 subtasks)
POSTING AND INTERACTIONS
DONE
Create issue
As a community leader, I want to pin important posts within my community so that new members can easily find key information.
Implement functionality for Community Leaders to pin posts within their community.
SCRUM-184
R
Ensure pinned posts remain at the top of the community feed or in a dedicated section.
SCRUM-185
R
Allow multiple pinned posts with a set limit (e.g., max 3-5).
SCRUM-186
R
Enable Community Leaders to reorder, update, or unpin posts as needed.
SCRUM-187
R
Provide visual indicators (e.g., “Pinned” label) to distinguish important posts.
SCRUM-188
R


TO DO
IN PROGRESS 2
DONE 55
SCRUM-192 As a student, I want to search and filter communities b... (7 subtasks) POSTING AND INTERACTIONS DONE
Create issue
As a student, I want to search and filter communities by keywords, or search text so that I can quickly find relevant discussions.
Implement Search Functionality
SCRUM-193
Implement Filtering Options
SCRUM-194
Design a User-Friendly Search & Filter UI
SCRUM-236
Setup Pagination Config
SCRUM-237
Test search validation
SCRUM-238
Optimise search performance
SCRUM-241
Create Generic Pagination Component
SCRUM-242
R



Product/Sprint 3 Backlog:

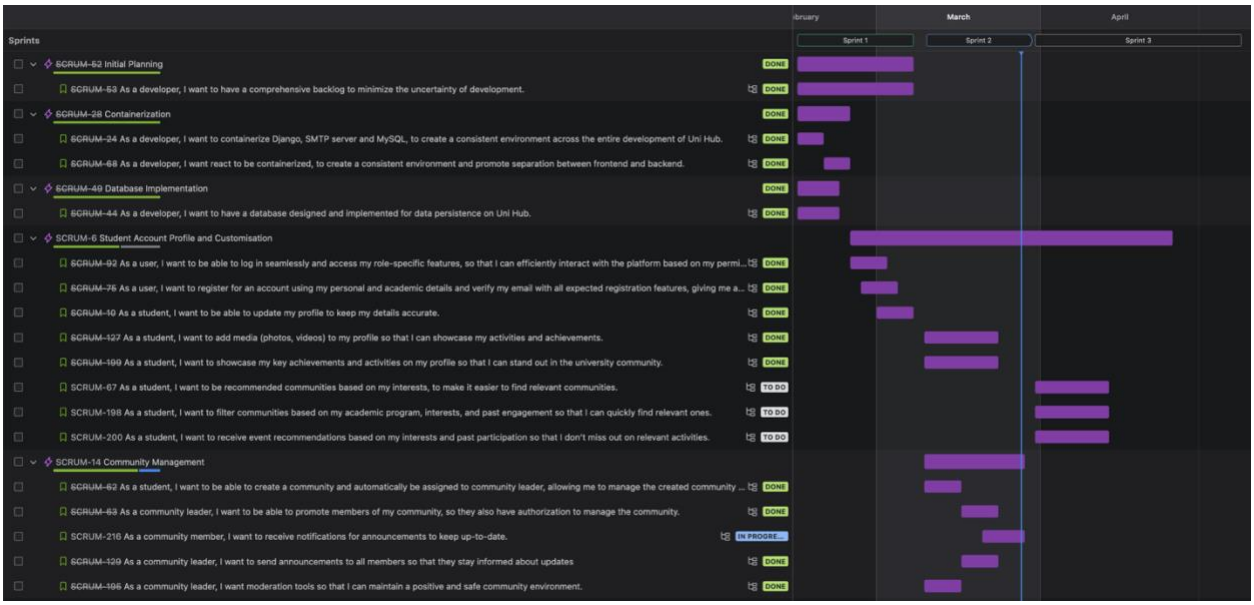
Here is the remaining product backlog to be implemented in Sprint 3 including the incomplete story from Sprint 2, the table below displays the story id to the intended assignee before the Sprint begins. The backlog displays the priority for each user story via the icon on the left (low/medium/high), each story in the backlog has assigned tasks.

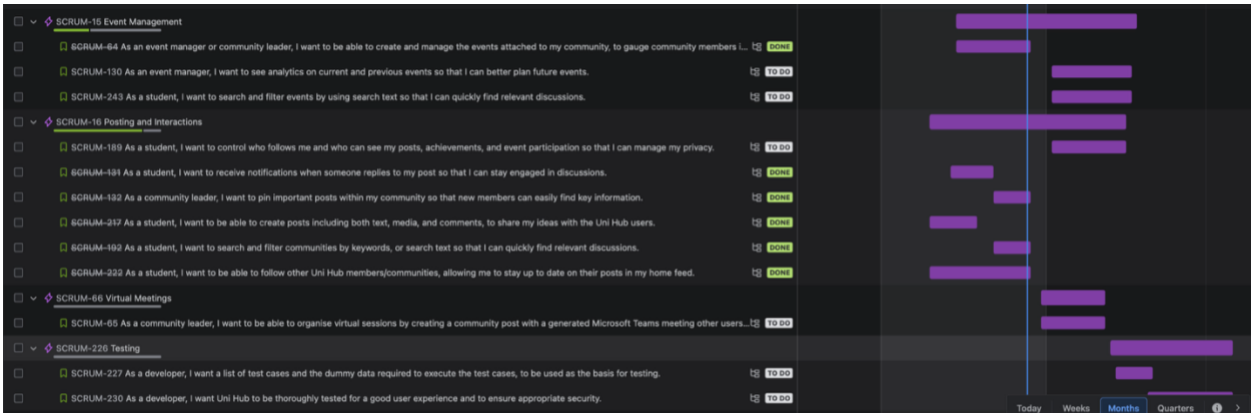


Story ID	Assignee(s)
SCRUM-230 (10 story points)	Arthur, Rohaan
SCRUM-227	Arthur
SCRUM-243	Rohaan
SCRUM-130	Seif
SCRUM-198	Seif
SCRUM – 189	William
SCRUM – 65	William
SCRUM-216	Chaya (From Sprint 2)
SCRUM-200	Chaya
SCRUM-19	Chaya

Current Project Schedule/Progress:

Below is the entire schedule, all tasks except one (task is in-progress) have been completed within the expected timeline.





Communication Issues

N/A

Reflections

Review of Progress

The project is very much on track when considering the project schedule above, all tasks assigned to this sprint have been completed except community announcement notifications, which is close to completion. For Sprint 2 we also had a larger number of tasks/story points than the amount planned for Sprint 3, so any modifications and minor additions required will have the appropriate amount of time to complete, consequently we are very much on schedule for completion. For this sprint we have implemented the functionalities for user interactions, communities, events, notifications, some additional quality of life improvements, cloud image storage with S3, and started searching/filtering capabilities.

The remaining tasks for Sprint 3 are generally lower priority and do not involve core functionalities of the project, with things such as additional privacy settings, event analytics, and the option to create virtual meetings. The higher priority tasks involve the thorough testing of the project, additional search options, and tailored user suggestions. The well-defined user stories and tasks going into the sprint allowed us to improve the burndown chart from Sprint 1, as no new tasks were required to be added to the sprint.

The group's synergy has been greatly improved from Sprint 1, member's roles were decided based on a specific user story each week, alongside having a designated SCRUM master which was rotated weekly. The SCRUM master would lead the weekly virtual meeting, checking the projects progress is as expected, ensuring members are clear on their tasks, and helping to delegate work going forward. Following feedback from Sprint 1 all code is now commented for clarification of its purpose to both other group members and stakeholders/tutors. Tasks were assigned to create minimal overlap between another group member's code, and tasks which might block another member's task were completed with the highest priority.

The table below details each member and their Sprint contribution in story points, we had decided before beginning the sprint that Arthur and Rohaan would complete fewer story points due to the disproportionate workload from Sprint 1, aiming for 14 story points each, whilst the other members would aim for 18 story points each. Furthermore, have already assigned the tasks for Sprint 3, showcased in the product backlog, to ensure we continue to maintain an even contribution of work.

Group Member	Story Points Assigned	Story Points Complete
Arthur Milner	14	14
Rohaan Aslam	14	14
William Barnes	18	18
Seif Mansour	18	18
Chaya Moore	18	13

Code Architecture

This application utilizes a multi-container architecture managed by Docker Compose. It consists of four main services: a React frontend user interface, a Django django-web-app backend DRF API, a MySQL db for data storage, and a MailHog service for capturing development emails. Each service runs in its own isolated container. They communicate over a shared Docker network: the frontend calls the backend API (using JWTs for authentication), the backend interacts with the database and is also configured to communicate with MailHog. Volumes are used to persist database data and allow for live code reloading in the frontend and backend containers during development. We are also using AWS S3 to store user uploaded images and

videos in the cloud, with plans to incorporate further cloud services in Sprint 3 for virtual meeting options.

Code & Project Management Limitations

Some tasks in the sprint were completed very close to the sprint deadline, which is understandable considering the number of tasks we had for the sprint, but this can hopefully be avoided in Sprint 3 with better time management.

The views/viewsets in our code could be more refined, for example, some views could benefit from being grouped into a viewset, particularly if there are separate views for creating, deleting, updating, etc on the same type of data. This would also have the benefit of being easier to extend the functionalities as required for Sprint 3. It is also possible group members have made a view separately which achieves the same result, these can be combined to reduce code redundancy before the end of Sprint 3.

A further limitation in our current project is the fact we are fetching every post on the home page, this could get very computationally expensive in an enterprise environment, consequently, pagination can be applied in a way that only fetches posts that are visible to the user, similar to the methods used in the community discover page.

Within our views we could apply additional permission classes more often instead of just IsAuthenticated. This would remove the need to make checks such as whether the user has the EventManager role inside each individual view, instead simply calling the appropriate permission class. This has been done for some views already, such as IsCommunityLeader on approving/denying community join requests. Permission classes to either be created or utilised more often includes IsAccountOwner, IsEventManager, and IsCommunityLeader.

Django DRF: One main challenge we all faced was with serializers, particularly when handling nested relationships or when multiple serializers were needed for similar data. This sometimes led to repetitive code or confusion around which serializer to use where.

We used pagination to handle large result sets, which was helpful for performance, but we also noticed that some endpoints could be optimized further by limiting unnecessary data being queried or serialized. In a few cases, our ORM queries weren't as efficient as they could be, especially when dealing with related models or larger datasets.

Authentication/Authorisation (We missed including our authentication and authorisation development challenges in Sprint 1. Below is a summary of our experience and implementation decisions):

It turned out to be one of the more complex parts of the project. We used Djoser on the

backend to handle authentication endpoints, React on the frontend, and MailHog for our SMTP server. A major challenge was finding the most secure way to manage JWTs in this architecture. While many tutorials suggest storing tokens in localStorage, that's not secure due to XSS vulnerabilities. Instead, we went with a more secure approach: storing the access token in memory (as a React state variable) and the refresh token in an HTTP-only cookie, protected with a CSRF token from Django. Since access tokens are short-lived by design, this helps limit exposure if they're ever compromised. But because React state gets wiped on hard page refreshes, we had to implement a mechanism to refetch the access token using the refresh token to keep the app functional and seamless for the user.

We also added a retry mechanism so that if a request returns a 401 (unauthorized) due to an expired access token, the app will automatically attempt to fetch a new one using the refresh token, if it's still valid and then retry the original request. On the email side, MailHog was a great tool for capturing outgoing emails like signup activation and password resets. It gave us a much better testing experience than Django's console email trap and helped us ensure those flows worked end-to-end.

MySQL: N/A

React: Component reusability was also a bit of a missed opportunity. In some places, similar components (like forms or cards) were recreated separately when they could've been abstracted into a shared, reusable component. This created some minor inconsistencies in the UI and will be an improvement point for refactoring during Sprint 3. This was mostly because React was a new technology for a few members of the group so there was an initial learning curve which impacted development speed early in the sprint.

MailHog: The major limitation of MailHog is the fact that it is unsuitable for production, it lacks the security features expected for email handling and only emulates an SMTP server, it does not have the ability to send outbound emails. For production, options such as SendGrid can be implemented to provide the necessary features for an enterprise application, with the downside of additional cost.

S3: The main drawback for S3 with this sort of use case would be its high cost of storage especially as the amount of data increases on the drive and when making frequent access requests. Furthermore, file retention can experience latency espeically with larger image and video files this latency issue will also increase further when users are located further away from the storage region. Finally, whilst S3 is backed by AWS offering some robust security features ensuring correct access control can be complex and there is a risk of misconfiguration which can lead to compliance issues or users being able to get access to data they should not have.

Plan for Testing/Security

A decent amount of Sprint 3 is designated to testing, the plan for testing is to first create a comprehensive list of test cases, then develop unit tests which cover these test cases. We also plan to apply manual testing where applicable. Testing is planned to be executed by Arthur and Rohaan. Arthur will create the tests cases and required dummy data, whilst Arthur and Rohaan will collaborate on creating unit tests to match the test cases. We will also ensure both the front-end and back-end include the same level of validation.

The project already includes security measures through built-in Djoser functionalities, such as hashed passwords within the database, and role-based permission checks. Security was also heavily considered during implementation of authorization and authentication, which was discussed above. To expand the project's security, we intend to expand the project's permission classes.