CS3099 Deliverable 4 - Group Report

Team 3 - Supergroup C
University of St Andrews

November 2021

Supervisor	Git Master	Supergroup Rep	Scrum Master	Test Master	Product Owner
Dr Edwin Brady	190003657	200012696	190020048	200013403	190013199

`README.md`	Dependencies and guidance for installation, configuration, testing and running project code.
`Appendix/TestEvidence.pdf` `Appendix/T03-Weekly-Progress-Report.pdf`	Test evidences. Weekly Progress Report.

`scripts`
`src/Journals`
`src/back-end/`
`src/back-end/Server`
`src/back-end/Database`
`src/back-end/RecmdSys`
`src/back-end/validation`
`src/front-end/client`
	The scripts directory contain scripts that help speed up development
	Uploaded sample journals.
	Back-end source code.
	Back-end server source code.
	Database API source code.
	Recommender System source code.
	Recommender System validation data.
	Front-end source code.

Project Wiki Individual Contributions	https://gitlab.cs.st-andrews.ac.uk/cs3099group03/project-code/-/wikis/home. See individual report.

1 Abstract

190013199

An online academic code distribution and code review platform bringing research software and academic code to the wider programming audience. The platform was built using an SQL equivalent of a MERN stack and written almost entirely in JavaScript. As to ensure a timely delivery of this project, we have closely followed the Agile project management philosophy and particularly relied on scrum to facilitate this project. Our platform accommodates user registration, publication and review of academic material, publication specific community discourse, and both user information and content migration between other platforms. In addition, a collection of interactivity and accessibility features have been implemented to extend the platform further following HCI review and evaluation, and a TensorFlow based python recommender system for feeding content to the user has been created; with the model trained every 24 hours.

2 Declaration

We declare that the material submitted for assessment is our own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 12,607 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, we give permission for it to be made available for use in accordance with the regulations of the University Library. We also give permission for the report to be made available on the Web, for this work to be used in research within the University of St Andrews, and for any software to be released on an open source basis.

We retain the copyright in this work, and ownership of any resulting intellectual property.

Contents

3 Introduction

200013403

We designed and implemented a platform for distributed coding journals targeting publishing peer-review research software. As to continuously deliver code into production and ensure an ongoing flow of new features and bug fixes, we followed the Scrum / Agile development model and CI/CD strategy.

The platform was built using a slight adaptation of the MERN stack [3] which is an ideal approach to working with JavaScript and JSON. The top tier of the stack is React.js. We build encapsulated components that manage their own state, then compose them to make complex UIs. The next level down is the Express.js server-side framework, running inside a Node.js server. We handle HTTP requests and responses using Express's powerful models for URL routing. Database services are supported by the school's MariaDB database server. We describe the security problems of the database implementation and prevent database injection at the code level.

The user interface consists of five main components:

- The top navigation bar provides simple navigation across components and access to search.
- The homepage displays lists of posts and users' statuses, as well as a side bar for logged in users which includes additional features and functionality.
- The search page allows users fuzzy search users, tags, posts and collections across the database.
- The account page shows user profiles and their viewing history. Users can log in to supergroups or reset passwords via an email sent to their mailbox.
- The post page contains details of a post and a code editor. Images and streaming videos are available to view online. Syntax highlighting supports 13 programming languages, while interactivity features allow users to rate a post and write or delete comments.

Users are able to select different roles, access, vote, and download all journals of the supergroup, as well as leave comments with a single account. Authors can upload files in any format and migrate submissions between coding journals. To help users retrieve and manage their posts, we designed tags and collections to categorize them, which are functionally linked to our site-wide search implementation, providing users with additional options for finding and accessing content.

Additionally, we implemented a Neural Network for our recommender system based on TensorFlow with the model trained every 24 hours. An RS for the MovieLens dataset is built at first with approximately 62% accuracy for the top 100 recommendations. After some successful attempts, we then built another working RS for our journal project using the same strategies based on the knowledge and experiment we gained from the first model.

We provided comprehensive help and documentation, to help users understand how to complete their tasks and to minimize the users' memory load by making elements, actions, and options visible. Following Jakob Nielsen's 10 general principles for interaction design, we set up usability heuristics evaluations as described in the report.

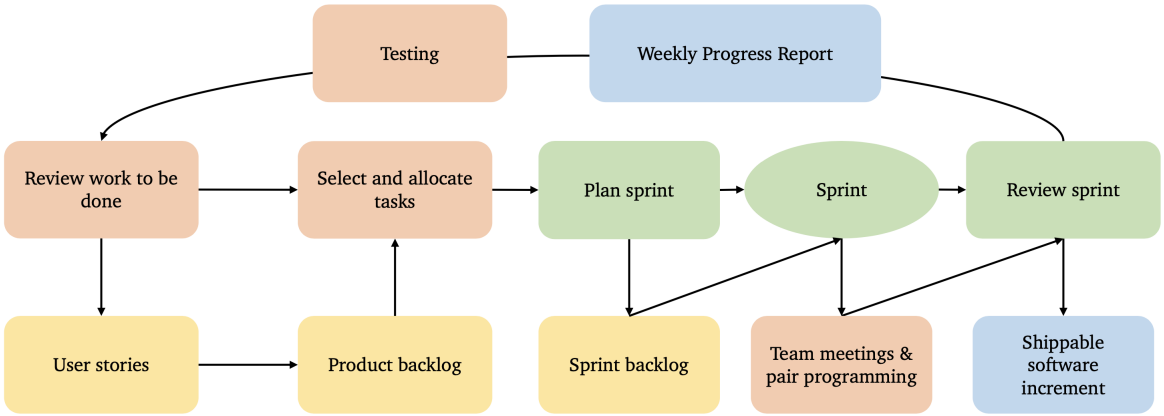
4 Project Details

4.1 Scrum

190020048

4.1.1 Organisational Practice

The team adhered to the Scrum development model for project management, and each sprint cycle lasted for a week. At the beginning of a sprint cycle, the product owner would prioritise the items in the product backlog to define the most important features to be developed in a cycle. Completed features were then checked off the to-do list. Items would be returned to the product backlog if they could not be completed within the allocated week for the sprint. The Scrum Master would then organise a follow-up review to assess the complications of the task which is not completed in the given time frame. The model below shows our sprint cycle. The input to the process is the product backlog, and each process iteration produces a shippable software increment. In-person team meetings and pair programming were incorporated into the cycle to encourage collaborative development and improve team cohesion via knowledge exchange. Continuous refactoring of project code was expected as soon as potential code improvements are found. This keeps the code simple and maintainable.



4.1.2 User Stories and Scrum Board

The Scrum board and teamwork organisation was consistently maintained on Microsoft Planner. For example, the implementation of the supergroup content sharing protocol would be assigned to members from the sprint backlog. As the team advanced in developing the sprints, tasks would be moved from the product backlog to the sprint backlog. User stories on our board were refined, analysed, and carefully taken into consideration, as they contain insightful descriptions of the wanted features from our business users' perspectives. Stories allow us to break the code journal features down into a set of manageable and understandable blocks that our stakeholders can relate to. Tasks were marked as completed after subsequent testing and progress reviews. In general, we have adhered to most of the user stories when selecting functionalities to be implemented to suit a wider demographic of users.

4.2 Weekly Progress Reports

The 'Weekly Progress Bullet Journal' is a documentation system used throughout the project duration. It was used to compile weekly progress reports and track the project's development progress. Before each Scrum meeting, each team member would fill out the progress form by Monday. The form includes three short questions relating to an individual's progress: 'What did you do this week?', 'What will you do next week?', 'Anything blocking your progress'. Members also have the option to schedule a team discussion or code review. The reports serve as reliable documentation for tracking the project progress. Weekly meeting notes and progress reports are attached to the Appendix for reference.

4.3 Meeting Reflections

Reflecting upon the weekly Scrum meetings, the team was involved in selecting the highest priority of tasks we believe could be completed. The team worked at a sustainable pace over the course of the project which increased our code quality and long-term productivity. With the team's collaborative effort, we came up with estimates of how much product backlog our team can cover in a single sprint. Understanding our team's velocity is crucial as it provides a basis for measuring improving performance. Scrum meetings were conducted on Microsoft Teams, while Discord was used for informal updates and discussions. All team members would describe their progress since the last sprint cycle and bring up any problems they have encountered. Everyone participates in short-term planning and there is no top-down management for team organisation. At the end of a sprint, there would be a follow-up review which allows the team to reflect on how things could have been done better and ensure the deliverable will be completed on time.

Constructive feedback was well-received from our project supervisor and we aim to incorporate the feedback in subsequent sprint cycles to constantly improve our project. Implementation of the recommendation system and syntax highlighting of source code in a post were some example changes made according to the 'customer involvement' from our supervisor. Upon reflection and critical appraisal, the team was driven to prioritise new system requirements and evaluate existing iterations of the system after each Scrum meeting.

Change of Plans since Deliverables 2 and 3

Since we had all basic features and most additional features implemented in Deliverable 2, we had a shift in focus on extending existing features and an ambitious plan to implement novel features like a notification and post recommendation system for Deliverable 4. Testing was also the focus in the final deliverable, as we want our users to have a smooth and complete experience when browsing the code journal. Automated unit testing framework and manual testing were both considered when monitoring the quality of implemented functionalities. A challenge was to manage inter-group interaction and schedule supergroup meetings for testing the supergroup login and content sharing protocol. Nonetheless, we scaled our agile methods and systematically organised testing sessions with other project groups during our sprint cycles.

Future Plans

Software maintenance and testing of existing systems will be our focus in future projects. Moreover, we have also recognised the importance of human-computer interaction design principles such as usability heuristics and accessibility. Hence, we had made changes in the responsiveness of our design and improved the help and documentation of the code journal to enhance the user experience. Exploring machine learning techniques like implementing neural networks also allow exciting extensions to our overall user experience, for example, our neural-network-based post recommendation system curates content that aligns with the interest of the users. If we were given more time to work on the project, we would explore further possibilities of version control technologies and optimise database and storage controls.

5 Supergroup Interaction

200012696

The Supergroup representative has been attending the Supergroup meetings and discussing how we can ensure our journals can be compatible with each other. The meetings were conducted on Microsoft Teams. The supergroup representative has made contributions to how the supergroup protocol must look like and relayed the decisions made from the meetings to the whole team. The supergroup meetings were also a great opportunity for us to ask questions about the proposed protocol and express our opinions on it.

A presentation was made to the group, by the supergroup representative, to inform them of the Supergroup login protocol. To aide with the understanding of the protocols a documentation was created, this enabled the group members to look up the functionality if they were unsure. An Endpoint documentation was also created for the group members to understand the exact data format required. This was done using a tool called 'swagger'.

We have also interacted with the supergroup to schedule testing sessions where we were able to see how compatible our implementations were and thus, we were able to make any necessary changes; This was especially useful when testing the resource sharing functionality of our systems. During the development of the supergroup login protocol our supergroup representative was able to reach out to other teams in the supergroup and test our implementation against theirs. Once a standard protocol for login and resource sharing was defined the frequency of the meetings was lowered, although we still met to discuss any issues which arose from the protocol design or subsequent testing.

6 Project Design

6.1 Summary Of Achievements

190013199

We have succesfully managed to acomplish our core objectives for this practical and have been able to achieve some additional functionality that we had hoped to complete. Overall, our implementation can be described as functionally complete, with room for further development via additional features and improvements across all aspects of our design. We have also been able to expand our horizons and develop features not included in older plans. New ideas were birthed throughout development and priorities were readjusted in favour towards the newer and more interesting ideas.

SUPER GROUP FUNCTIONALITY	DONE ?	TESTED ?	AGAINST ?
SG Features Accessible via Group Account *cs3099user03.host.cs.st-andrews.ac.uk/*	Yes	Yes	ALL
SG Login - Inbound *Other journal's users can log in to our one.*	Yes	Yes	t06, t27 t21*
SG Login - Outbound *Our users can log in to other journals.*	Yes	Yes	t06, t24, t15
SG Import File	Yes	Yes	t06, t21, t27
SG Import Metadata	Yes	Yes	t06, t21, t27
SG Export File	Yes	Yes	t06, t21, t27 t12*
SG Export Metadata	Yes	Yes	t06, t21, t27 t12*

'AGAINST' indicates what team's implementation we were able to test our own super group functionality against and achieve an outcome that indicates success. Supergroup functionality could only be tested with groups who have produced a live, working implementation on their group accounts. If a team is listed in AGAINST, that means we were able to successfully achieve the associated functionality with them as the other journal.

t21 - for team 21 you need to open console log on their accounts page, retrieve their user id which is hidden, and then you need to fix their user id into the protocol format by replacing @t21 with the protocol correct :t21, then it works!*

t12 - Export succeeds on protocol level but our files don't appear on their site.*

FEATURE	STARTED ?	DONE ?	TESTED ?
Register User	Yes	Yes	Yes
Login User	Yes	Yes	Yes
Create Post with a Single File	Yes	Yes	Yes
Create Post with Multiple Files	Yes	Yes	Yes
Delete Post	Yes	Yes	Yes
Delete entire journal (multiple posts) **(*)**	Yes	Yes	Yes
Edit Post	Yes	Yes	Yes
Post Categories	Yes	Yes	Yes
Download Post	Yes	Yes	Yes
Upvote/Downvote Posts	Yes	Yes	Yes
Favourite Posts	Yes	Yes	Yes
Cookies	Yes	Yes	Yes
Create Comment	Yes	Yes	Yes
Delete Comment	Yes	Yes	Yes
Edit Comment	No	No	No
View All Posts	Yes	Yes	Yes
View Your Own Posts	Yes	Yes	Yes
User Account Page	Yes	Yes	Yes
User Viewing History **(*)**	Yes	Yes	Yes
Viewing History Display Options **(*)**	Yes	Yes	Yes
Global Search **(*)**	Yes	Yes	Yes
Post Ranking System	No	No	No
Recommender System	Yes	Yes	Yes
Moderation Features	No	No	No
Notifications **(*)**	Yes	Email Only	Email Only
Keyboard Shortcuts/Accessibility Features	Yes	Yes	Yes
Interactive Guide and Help Menu **(*)**	Yes	Yes	Yes
Homepage sidebar + User Statistics **(*)**	Yes	Yes	Yes
Interactive User and Post tags **(*)**	Yes	Yes	Yes
Text Highlighting	Yes	Yes	Yes
Valid file type expansion (video + image) **(*)**	Yes	Yes	Yes
Website Footer	Yes	Yes	Yes
Advanced Peer Review Process	No	No	No

The (*) *Symbol Indicates a feature that was initially omitted from product backlog or in our plans until after deliverable 3 but became something we decided to work on / prioritize for the deliverable 4 deadline.*

These tables highlights and summarises our core features and system elements that we had either planned, worked on, or achieved and indicates how well we have managed to acomplish our goals. Of course, we were not able to implement every feature we had hoped for, and certain planned features were never reached either due to a lack of importance in our greater vision or due to a lack of time with other features taken priority. Regardless, we were still able to acomplish all 'essential' functionality as described by the requirements for the MVP and we have built on those features granting them with further technical enchancements and additions. Furthermore, we have added our own seperate and advanced functionality, breaking out of the minimum viable product with our own ideas for the product, and expanding the scope of what we have to offer.

It is worth noting that the table given here does not list every feature of our user stories, or every feature idea we ever had. Instead, the table is an ambitous set of objectives we hoped to achieve by the final deadline. Of this list, we had managed to complete the majority of features and some of our thought process can be seen from what is remaining in yellow or red. Towards the last few weeks of development, it became clear that there will not be enough time to complete everything, thus, decisions had to be made on what features to prioritise. Features like a recommender system, search, and viewing history were decided to be more important for our project than the alternatives. Furthermore, the chosen features were viewed as more completable in the time remaining and thus we actually managed to implement them to a good degree before the deadline. Meanwhile some features like editing a comment did not see much attention due to the considerably lower value that they added in comparison and thus were left out of the final sprint cycles. Some features like the peer review process were just too large in scale to complete in the time remaining and it was deemed that no usable implementation could be delivered in the remaining time. Overall, we believe that the right decisions were made and that our time was effectively utalised to create the best work we could while still providing fully baked features that provde the expected functionality.

6.2 Database

200013403

As systems become more complex, so many requests come to the database that we need an agent to handle them by hiding the SQL details, thus other back-end developers won't need to worry about the SQL, table structure, their relationships and relating errors. The design of database data exchange is inspired by the proxy design pattern. It again displays flexibility since the agent uniformly schedules objects that are expensive to create or need to be protected. For example, creating a new connection for every request and releasing it later would significantly damage the user experience.

6.3 Users

200012696

6.3.1 Register

When the user first visits our system, they can view posts made by other users. If they would like to contribute to the system- to comment or post code of their own, they will have to register to be part of the system. This can be done through the account menu, where the user is able to input their details to register.

The registration details are collected in the front end and verified. This is where the user selects their status in the system (viewer, author, or reviewer) Once verified successfully, it is then sent to the back end (where it is verified again in case the frontend java script was not enabled).

Once the backend verification is completed, a user id is assigned to the user and checks if the user has already been registered. A token is also generated for the user during registration, which is used for super group login functionality. The user is then added to the database.

If the user is added to the system successfully then all the details regard the new user is send to the

frontend. The frontend displays the username (which the user can take a note of) and enables the user to explore our journal according to their status and make contributions.

6.3.2 Login

We have decided to offer a personal experience with our system. This would mean each user has a profile where their history, posts, comments etc. are stored. The profile is accessed by their username (assigned when they register to our service) and password. Once these login credentials are collected, A basic syntax check is done on the front-end and then it is sent to the back end to be validated.

The login credential is verified using the data base and if it is successful then the user details are sent to the front end enabling them to access their profile. The user's login status is stored as a state using cookies, which will enable them to remain logged in even if the browser window is closed. This choice was made with user convenience in mind.

6.3.3 Recovery

We identified that the user could run into the risk of forgetting their password. If this happens then they will permanently be locked out of their account. To prevent this, we have decided to provide the user with password recovery functionality.

If the user happens to forget their password, then they select 'forgot password' option after entering their username. A request is sent to the backend, which then sends an email with a code to the user. The user then accesses the email and enters the code into our system along with their new password.

The code is then checked at the backend and on success updates the user's password.

6.4 Posts

190013199

Posts are the core component of our system and a large degree of importance has been placed upon how our systems handle operations like uploading, viewing, and interacting with posts. In our system a 'post' is the name given to a construct that consists of a title, description and a file, while a 'journal' is a construct made up of multiple 'posts'. Journals can be made up of many posts but each post belongs to exactly one journal. Additional information is also kept track of like create and edit dates as well as of course the author of the submission/s. Only logged in users can attempt to make a submission but all users can view them. Currently, there exists no limit regarding the number of posts a user can make other than the physical storage limitations of our database.

Multiple files can be uploaded as part of a single 'journal', and each file will become its own post, linked to the parent journal. We have decided that, as well as allowing multiple files to make up a single submission, it would be beneficial to allow each file to be treated independantly at a fundamental level. This allows our users to comment on each file of a journal submission specifically, and allows for file specific user interactivity that we allow users to conduct on our platform. This includes the ability to favourite, 'upvote', and 'downvote' posts scoring them for the benefit of potential algorithms or other users while also providing additional means to express ones opinion about a given file specificially. In our design, it is posible to positively rate one file of a journal, while negatively rate all other files. Therefor users can target both their verbal feedback via comments and overal satisfaction via voting on a file by file basis ensuring that feedback is more directed and specific rather than broadly covering an entire submission. We have not included any limits on the amount of posts a journal could include, though if the project were to be shipped, some arbitary limit would be placed as a precaution against potential bot spam. Lastly, we have decided that the user cannot upload multiple files of the same name into one journal as yet another precaution, while this may not cause any problems internally due to our use of unique id generation for filenames, it could cause confusion amongst users so the safeguard was included.

Files that have been published to our system can be edited at any point by the author. To do this we have created a text editor within our web application that allows the author to edit any line as they would in an IDE, but without IDE features. This has been expanded further with text highlighting for supported file formats and we decided to add a cancel button for when the author changes their mind or has made a mistake as a quality of life improvement. Deleting posts is another author specific feature and we allow for the deletion of single posts (keeping the rest of the journals intact) as well as entire journals. Given the deletion of the last post belonging to a journal, the journal should be deleted automatically. We have decided to however add a 'are you sure?' pop up for any post related delete functionality as a safeguard for any authors who press it by accident. Given that most people don't like deleting their work by accident this is a somewhat overlooked but nevertheless important feature to have.

6.5 Comments

190020048

A comment allows users to respond to a post in a journal. To comment on a post, users click into the input box that has the placeholder – 'Add a comment'. Users then type what they want to say and press the 'Post' button to publish the comment. Users must be logged into their accounts to post a comment. A comment displayed for a post consists of the user's name, date and time posted, and the respective comment paragraph listed in chronological order. A post can have multiple comments from multiple users.

`validateComment` is a function in `validate-client.js` that is designed to check if the data that is sent consists of the comment input value, `postId` and `userID`. Otherwise, if validation fails, an error message will be displayed on the console. `validateInput` was designed to check both the length and the syntax of the given data. It is usable for most String data and comments must have a minimum of 10 characters, a maximum of 100 characters and should be a title Regex.

In `APostComments.js`, comment elements are displayed alongside any required buttons, including the comment, an HTML box, `userID and \verb`APost— – the post related to the comment. `printComments` is designed to display the actual comment in the paragraph tag. For each comment, the 'delete comment' button is displayed if the user has ownership of the comment. Otherwise, the 'flag comment' button is displayed if the comment does not belong to the user. The functionality to report other users' comment is yet to be implemented which has the potential to censor undesired content through admin moderation. The function `delCommentActiveState` consists of the back button and the delete button, which allows owners of comments to go back to a post and leave the delete comment active state as well as to delete a comment respectively.

In `APost.js`, `getComment` obtains a comment using the `postId` - GET `cmmtUrl` specifies the path of a comment to be obtained. `showCommentContent` is designed to display the comment in an HTML box. If the box is not empty, the comments are printed. The function `postComment` takes an HTML input value, the `userId` from the cookie and the `postId`. `sendComment` then returns an OK success status response code to indicate that the request has succeeded. A 200 response is cacheable by default. `renderComments` check if the user is logged in as a comment can only be posted after a user logs in to their account. `Comments.css` styles the buttons to post and delete comments. The comment paragraphs are styled and displayed in a box located below the post information.

6.6 User Interface

190013199

The way users interact with our system is something we wanted to focus on as to achieve a very minimlist user experience with a focus on clarity and ease of use. The core idea was to reduce unnecessary clutter, with us aiming to give the user powerful tools without flooding the websites UI with undesirable components. Additionally, we wanted our site to be accessible to those suffering from physical impairments, which would otherwise restrict a users ability to independantly navigate our site.

The top navigation bar is very straight forward and offers indication of where the usser currently is. Additionally, it has been purposefuly split into left and right sections with navigation on the left side and utility functionality on the right side. This helps make our navigation clean to the user and creates a visual seperation for the two sections which offer different types of functionality. Lastly, we have included on hover functionality to visually indicate what the user is about to select. While many would precieve this as a stylistic choice is also help users with disabilities or other impairments who would find it more difficult to use this site. Thanks to the on hover functionality, we hope that impaired individuals will have less trouble navigating our site.

On our home page there exist up to 4 sections tha the user can visit and each will show a different set of posts or journal, as indicated by the name of each clickable tag; These include 'Recommendations', 'Explore Posts', 'Your Journals' and 'Your posts'. When logged out, only the first two are available however upon logging in the user will gain access to the other two. The reason behind this is that non logged in users cannot have uploaded posts or journals so by removing those we create a cleaner UI which does not include elements that offer no value to the user. Additionally, have made the currently selected tag substantially visually distinct from the others as that is the only clear indicator of where the user currently is.

For the actual journals and posts, we decided to display more important information in larger font sizes and bold (Titles, authors, description) while less important data like the date is intentionally less visible. This allows us to keep all information available wwhile minimising on clutter as we focus on what the users would be most interesting when viewing posts, that being the preview information regarding what it is and who created it.

For displaying posts we have seperated our page into 4 sections. Each section offers a different feature or feature set for the users and thus it made sense to seperate them visually. For example the second section displays comments and allows users to post new or delete existing comments, while the third sections shows the file contents of the post. We have decided to keep most functionality hidden and unavailble to users who are not logged in such as voting and comments as they all require the user to be logged in. We also had to hide author specific functionality like editing and deleting posts. Lastly, we have implemented both light and dark mode for displaying file contents to the user, who is able to freely switch between them and use their preffered mode. We decided to implement this to help people suffering from visual impairments as when it comes to viewing multiple lines of code, some people might find one option better on their eyes than the other.

To provide greater interactivity, as well as to provide the user with more options when navigating our site, interactive tags were added that allow the user to access content that shares the given tags. This functionality is largely equivalent to using the search bar but is simpler to use given that it requires just one click and should be familiar to users given the prevelence of interactive tags across all social networks. A similar idea has also been extended to author names which, when pressed, will allow you to access all posts published by the said author. Again, we expect this functionality to be largely familiar to users and makes accessing the work of a given person simpler using design concepts familiar to the users.

We have decided to include loading icons and animations to our design to communicate with the user that things are working behind the scenes. This clarification makes it easy for the user to understand when something is still being fetched whilst the animations remain relatively minimalist and do not detract from or interfere with any other elements. Lastly, the footer is an asthetic addition to our site to make it look more official and complete. Currently, the buttons in the footer are just a placeholder as we never intended to create a privacy policy or about page. Whilst both would be good additions, they are nothing more than paragraphs of text, and we wanted to focus on more functional work that added greater value to our implementation. rather than typing out something that likely wouldn't even be read.

6.7 SG Login

200012696

We have decided to adopt the Single Sign-On(SSO) Protocol to implement Supergroup Login. When Home journal is mentioned, it is referring to the journal at which the user created their account and when

Secondary journal is mentioned it is referring to the journal which the user wants to login using their home journal login.

If a user from another journal wants to login to ours, they can do so by typing in the name of their journal in the format `t<team number>`. The user is then redirected to their home journal along with a state value which was created in the secondary journal and asked to login at the home journal. Once the user has logged in successfully the user is redirected back to our journal along with the state and a token that was generated by their home journal. The secondary journal checks if the state is correct and then queries the home journal to verify the token. If the token matches, then the details are sent over for the user and are inserted into the database and the user is logged. The user can then be treated as a native user.

We have fully tested this functionality with team 6. We were unable to find any of the other teams to test with as their systems were not deployed while we were testing.

6.8 SG Import + Export

190003657

6.8.1 Introduction

Super group import and export was implemented following the protocol arranged and shown at the following address: https://app.swaggerhub.com/apis/feds01/supergroup-c_api/1.0.0#/

Figure 1 is a diagram showing the flow of data when following the defined protocol.

The process conducted when designing and implementing import and export was as follows:

- create rough draft and test against ourselves
- test against other groups
- iterate using postman and reading response yaml files to determine errors

I will now detail the design decisions we had to make in order to follow the protocol and send data like in the diagram.

6.8.2 Design decisions

Overview

For security reasons, we are using javscript web tokens. When we send a request for them to import from us, we send a unique web token that we have signed using a secret that we have created following the guidance on the official jwt website (https://jwt.io/introduction). This is used to authorise requests to export a post, verifying that we were the ones who started the process.

Using axios, the server will send and retrieve post and get requests and handling those it is sent, and returning the relevant data to those that need it. In our implementation, we send responses to both the server we're on and a response to the asker following the super group protocol's set responses.¡br¿ Once the initial import request is sent, all request to the exporter requires a jwt, with the exporter then responding with either a success or an error, from which the import process will stop.

Exporting a post

Because of the implementation of our backend, when sending an export request, we need to get the details for that post, reconfigure the provided path from the previous request to an absolute one, compress the file, then export it to them. This was done using the JSZip module and the node filesystem handler.

Exporting MetaData

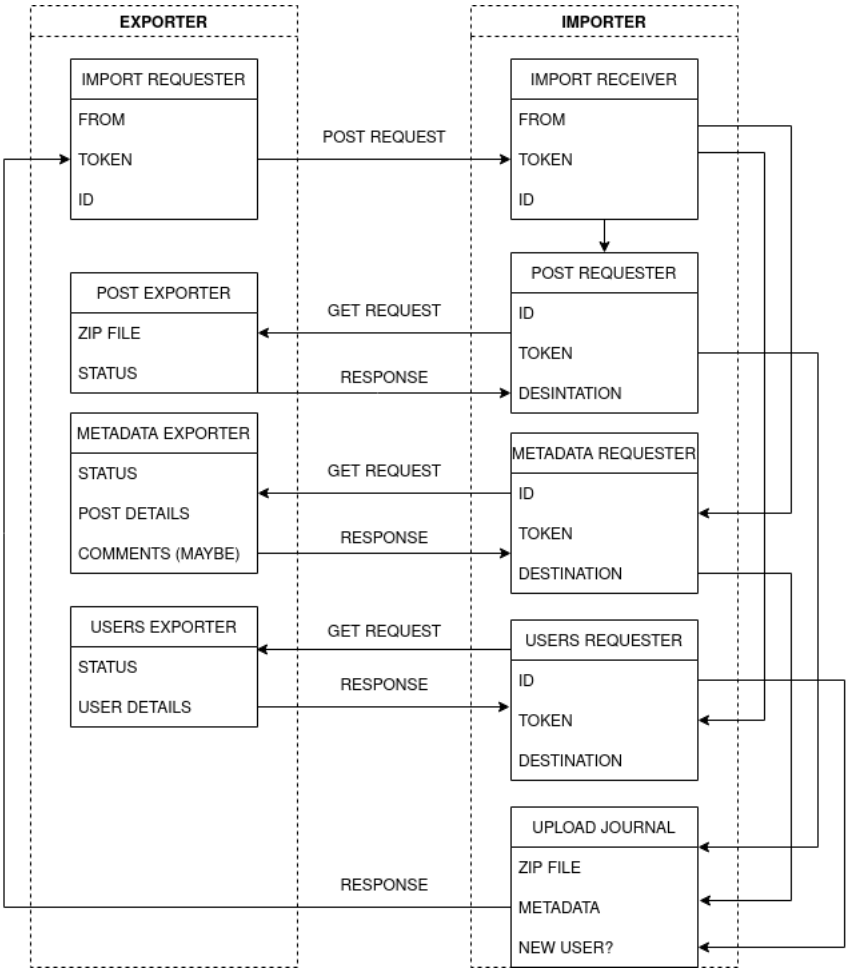


Figure 1

Due to some issues with getting metadata to export correctly, postman was used extensively to verify the correctness of our export. Once we had exhausted all our options on our end, we got in contact with the other group, from which they debugged their system and allowed us to export to them.

Sometimes, a post will not have a description on our end. As these cannot be null, we simply declare that the post has no introduction before exporting it to the other journal.

As we are using timestamps on our journal, and the protocol asks for time in ms, we have to convert this to follow the protocol via the Date class.

Exporting Users

Because of how we have defined the users on our end, we need to create a single string for usernames to export them to the other journal.

Receiving an import request

When an import request is received, we first have to set the axios config to follow the protocol. This means using the token given to set an authroization header, and in the case of importing a post, specifiying the response type as an arraybuffer - as otherwise the imported zip file will be missing bytes leading to it being corrupted.

Once this is done, we use the address of the journal sent to us and the ID given to conduct all requests with the required configs. After verifying that the exporter has responded with a success status, we can then go

on to beginning the upload.

When uploading, a new upload method was defined as there was trouble using the predefined version for uploading posts on the site. It used File objects which were exclusive to the frontend, and did not exist in node.js. Leading to a unique implementation being required.

Finally, after unzipping the zip file, the structure of the folder had to be flattened in order to accomdate the flat file structure of the journals on our site. The zip folder also couldn't be uploaded as is, as our site does not have the extension to read zip files and to see what is held within them.

6.9 Recommender System

200013403

Recommendations algorithms are part of all major online businesses these days. We realized that we need a system that can help users discover new posts and make posts suggestions based on which ones given users are more likely to view and vote.

Paul Covington et al [1] described the YouTube industry recommendation systems (RS) at a high level: a deep candidate generation model and a separate deep ranking model. Referring to their insights derived from designing, iterating and maintaining a massive RS with enormous user-facing impact, we build our RS based on TensorFlow open sources platform using Python.

The overall structure of our RS is illustrated in Figure 2. The system is comprised of two neural networks: one for candidate generation and one for ranking.

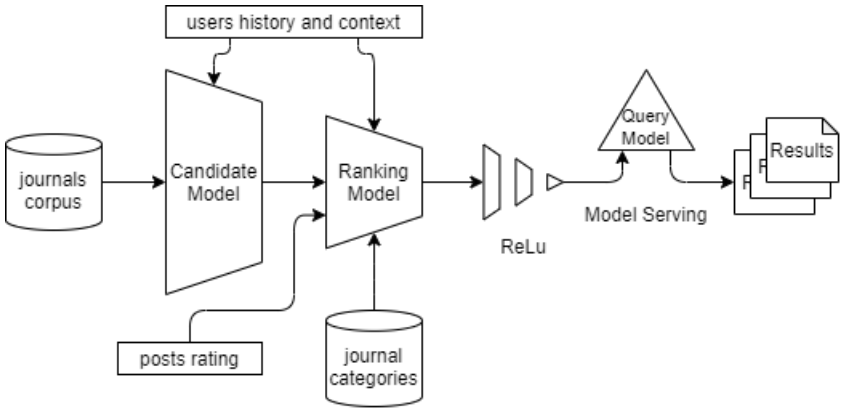


Figure 2: Recommender System Structure

- The candidate generation network is responsible for selecting an initial set of hundreds of candidates from all possible candidates. The main objective of this model is to efficiently weed out all candidates that the user is not interested in. Because the retrieval model may be dealing with millions of candidates, it has to be computationally efficient.
- The ranking network takes the outputs of the retrieval model and fine-tunes them to select the best possible handful of recommendations. Its task is to narrow down the set of items the user may be interested into a shortlist of likely candidates.

We then deploy these two components by exporting a query model which speeds up making predictions.

Due to the limitation of the development, we are unable to collect practical data and set up experiments to test our RS. We decided to build an RS for MovieLens [2] dataset at the first stage of implementation. This dataset was collected and maintained by GroupLens, a research group at the University of Minnesota. It contains a set of movie ratings from the MovieLens website, a movie recommendation service. The main reason for choosing this dataset is that it involves features that are very similar to our system.

In order to ensure that our method of building MovieLens RS can be applied to our own journal system, we abstracted the commonalities between the two systems. We encapsulated 'tensorflow.keras.Model' as customized abstract 'Model'. Because the implementation of candidate and ranking models varies by dataset themselves, we then further abstract them as a 'Tower' model. We carefully use the same strategies to design the architecture as illustrated in Figure 3.

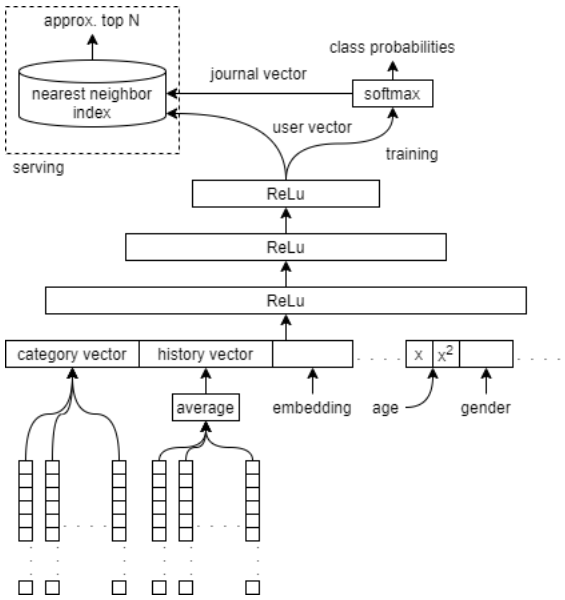


Figure 3: Recommender System Structure

After some successful attempts, we then build a RS for our journal project based on the knowledge and experiment we gained from the first one:

- We randomly split the dataset, putting 80% of the ratings in the train set, and 20% in the test set.
- The features we used in our ranking models are split between categorical and continuous: gender, role, voting, favourite, viewing history etc.
- **Data Pre-processing**
  - Strings: we turns raw strings into an encoded representation that can be read by an 'Embedding' layer or 'Dense' layer.
  - Continuous features: we perform feature-wise normalisation of continuous input features and turn continuous numerical features into integer categorical features.
  - Categorical features: we turns integer categorical features into one-hot, multi-hot, or count dense representations.
- **Depth**: We compared the performance of our MovieLens RS at different depth of ReLu layers, the evaluation details are described in evaluation section. To balance accuracy and time, we set the depth to 3: 1024ReLu $\rightarrow$ 512ReLu $\rightarrow$ 256ReLu.
- **Activations**: Softmax.
- **Loss and Metrics**: top K categorical accuracy. (i.e. how often the true candidate is in the top K candidates for a given query.)
- **Secheduled Task**: Server exports data from the database as `.csv` files and retrain the recommender system model at 24:00:00 every day.

Given more time we would explore metrics (precision, recall, ranking loss, etc.) which can guide iterative improvements to our system.

6.10 DevOps

190003657

6.10.1 Scripts

To help streamline development, scripts were implemented to automate processes that were difficult for the individual to deal with on their own. The scripts are as follows:

- `backend.sh`
  Starts up the Server
- `cs3099user03.sh`
  Once on klovia, connects you to the cs3099user03 group account
- `frontend.sh`
  Starts up the front end for the Deployed Site
- `frontendlocal.sh`
  Starts up the front end for local development
- `killSite.sh`
  Kills running processes related to the site, allowing the user to restart in case of an error
- `klovia.sh`
  Allows the user to sign in to the klovia server
- `nginx.sh`
  Restarts the nginx proxy to reflect changes in the server config file
- `setup.sh`
  Installs all the required modules and dependencies in order to run the site
- `startup.sh`
  Starts up the site depending on certain flags. No flag starts up development locally on your client. The -s flag starts up development with the output of the server going to the console, and the output of the front end going to a file called nohup.out - this occurs when you have no flags as well (you can use the browser console instead of reading nohup.out though!). The -o flag starts the site with output off, i.e. it's running in the background. This means a user can log out and still have the site running in the background on the server for others to access.

With the above, assuming you are on a new lab machine, you would be able to simply run the setup scripts and then the startup script and have the site running on your machine. If you have already got the site setup, then you can simply run startup whenever you want to develop locally.¡br¿ Developing on the group account requires the use of the -s flag. Running it in the background on the group acccount, the -o one.

6.10.2 Nginx

Nginx was configured so that it would be listening on two ports - 3102 and 5102. These two were chosen as the previous 3000 and 5000 that we were using for the front and backend respectively ended up coinciding with the port of another group, leading to development stalling whilst this had to be fixed. The backend requires the request to be sent to the site url then `/api/` in order to communicate with the server. This was done as there didn't seem to be an option of having two proxies listening on the same port whilst having the same path.

6.11 Search

200013403

We allow users to search posts, tags and users that are likely to be relevant to a search argument even when the argument does not exactly correspond to the desired information. The fuzzy search is done by fuzzy matching SQL queries, which returns a list of results based on likely relevance even though search argument words and spellings may not exactly match. Exact and highly relevant matches appear near the top of the list. Fuzzy searching is much more powerful than exact searching when used for research and investigation. Fuzzy searching is especially useful when researching unfamiliar, foreign language, or sophisticated terms, the proper spellings of which are not widely known. Fuzzy searching can also be used to locate individuals based on incomplete or partially inaccurate identifying information.

There are four different ways that users can search for a keyword:

- Search everything across the database: search `<keyword>` in the search bar or access URL `/search?all=<keyword>`.
- Search by tag: search `tag:<tag_name>` in the search bar or access URL `/search?tag=<tag_name>`.
- Search by role: access URL `/search?role={viewer ; author ; reviewer}`
- Search by user: access URL `/search?userid=<uid>`

6.12 Voting and Favourites

200013403

Users can vote for posts and add them to favourites. Voting and favouriting data will be recorded by the journal system for training the recommender system model.

6.13 Code Editor

200013403

Authors can change the content of their submissions without deleting and re-uploading by using the code editor on the post detail page. Two themes (light and dark) and syntax highlights for 16 programming languages formats are supported (i.e. `.js`, `.jsx`, `.hpp`, `.h`, `.c`, `.cpp`, `.html`, `.css`, `.java`, `.json`, `.md`, `.php`, `.py`, `.rs`, `.sql`, `.xml`). Syntax highlighting data are provided from codemirror[**?**]. Editor also allows users to fold regions of source code using the folding icons on the gutter between line numbers and line start.

Images and videos in supported formats can be displayed online. The code editor will be replaced by media. To ensure users can submit and play large videos (i.e. larger than 1GB), videos are streamed to the html5 video player.

6.14 History

190013199

The History sections located in the Account page is a feature designed to keep track of the posts that the user has accessed in the past. As the main purpose of this feature is to aid the user in finding something they have recently looked at, we have added functionality to make the search easier for our users. For starters, the history posts are presented in descending order from the most recently visited, as it is assumed that users will be looking for something they visited relatively recently. Secondly, the user has been granted great flexability in accessing and navigating their viewing history; with buttons allowing the user to display up to 50 history elements at a time and easily navigate to both the start and end of their history. Lastly, what we display to the user is important and thus the name, author, and the date that the user has visited

a given post have been decided to be the 3 key pieces of information to display. In addition, as a quality of life addition, the key navigation buttons are available at both the top and bottom of the history elements. They are in two places so that the user does not have to scroll all the way back up to view more.

6.15 Notifications

200012696

We identified a need for our system to notify users once there has been a change made to their post. This feature was giving a substantial amount of value to the whole usability of the application.

We realised that it was quite easy for a user to not realise a comment has been made on their post unless they specifically go back to that post and manually view its comments. This would have made the whole experience very clunky, and user would not be able to immerse in the application fully.

To rectify this issue, we decided to notify the user (using the email they provided), every time a new comment is being made. The use is also told which post the comment was made in and the comment itself. The user can then navigate to the post and reply to the comment making for a seamless experience.

6.16 Help and Documentation

190020048

Help and documentation are necessary for the learnability and usability of the code journal. Onboarding users from supergroups can easily assess help in the Help Menu on the navigation bar. The online documentation focused on the user's tasks and minimise the interaction cost of the user. Frequently asked questions relate to the users' needs and provide answers to resolve problems. Keyboard shortcuts to switch between pages were also implemented for user acceleration especially tailored for expert users. Keyboard shortcuts such as 'alt + b', 'alt + n' and 'alt + m' redirect the user to the Home, Upload and Account pages respectively. Moreover, beginner users are introduced to the code journal through an onboarding walkthrough tour. The 'Start Tour' button begins a step-by-step guide of the features on the code journal. This allows users to understand the key principles and features of the platform.

7 Evaluation and critical appraisal

7.1 Database

200013403

In the previous version of our system, we noticed that attackers can target SQL-like databases by entering malicious SQL code into input fields in the web app to gain access to or alter the data in the database. For example, to fetch users with given conditions:

```
let gender = "femal';--";
let status = 0;
connection.query(`SELECT * FROM users
                  WHERE gender = '${gender}'
                  AND   status = '${status}'`,
                 (error, results) => {...});
```

The problem here is that the parameters are not filtered before being passed to the query. As a result of the way MySQL works, this will be interpreted as

```
SELECT * FROM users WHERE gender = 'femal';-- AND status = 0
```

The second part of the query gets ignored because MySQL sees the semicolon and assumes that's the end of the query due to the '–' sign, which denotes a comment. Attackers can exploit this weakness to gain unauthorized access to the data or even destroy it. We made some code-level tweaks to prevent SQL injection vulnerabilities. The idea is to make sure the data being passed to the query is escaped before being executed.

- using placeholders:
  `connection.query("SELECT * FROM users WHERE id = :id",{id:user.id},()=>{})`
- avoid insecure packages.
- using the `node-mysql` module build-in methods such as `connection.escape()`

7.2 Users

200012696

7.2.1 Register

Although the current username system works perfectly, I think we can massively improve it by allowing the user to create their own username. Currently, the usernames assigned to them is very long string of random letters and number which is quite hard to remember. We decided to opt for this type of username to comply with the super group protocol for sign in. We could have used this same syntax for a user id which would uniquely identify each user according to the super group protocol but also have a username which is also unique but created by the user. This will make the program a lot more user friendly and useable on a day-to-day basis.

When looking at other systems which require a sign in – such as Gmail, it is quite evident that they verify whether the email or phone which is provided is valid. This could be done through sending a code to email and asking the user to input that into the system to confirm that the user has access to the contact details.

Although at a first glance it does not seem very important, it is crucial if the user wants to recover their password using their email. If they have accidentally inputted the wrong email when registering, they will never be able to recover their account.

While looking at features in other systems such as Instagram, it gives the user the ability to edit and change their details which was used during registration. I believe this is an essential feature which would have made our system much more accessible and robust. For example, if a user inputs the wrong email, they could always edit it which would mean the whole process is flexible and not rigid.

We could really improve the security aspect of the system by implementing password hashing so the passwords are not stored as unencrypted plain text.

7.2.2 Login

As the system is storing quite important information like someone's work, it is crucial their account can only be accessed by only themselves.

I believe, our system has a flaw with regards to this. This is because anyone would be able to access your profile if they have the correct password and username. This could be avoided by implementing 2 factor authentication.

So, every time the user logs in an email is sent to them containing a code which they must verify to gain access to the system. This could be quite tedious so we could have added this as an option. I believe this would massively improve the security of our system.

Looking at systems like Facebook, they enable their users to access their profiles using their emails. This would mean it is a lot less effort for the users to use the system as they don't have to remember a username

which is assigned to them. Our system lacks in this sense, but it could quite easily be implemented. This would drastically improve the usability of the system.

7.2.3 Recovery

In other system which uses login functionality and password recovery, they tend to enable users to reset password by sending a link to their email which would then lead them to a page where they could reset their password. This saves them having to copy a code over from their email. Lacking this feature really affects the seamlessness of the whole process. We considered using a security question to reset the password as seen in older system. But we decided against this as the user could as well forget the answer to the question. I believe this is a good choice we made as the program is more flexible.

To reset the password, an email must be sent to the user. The 'nodemailer' module which was used to send the email happened to fail to establish connection with the Gmail server when it was deployed, although it was working perfectly on the locally. We have concluded that it is an issue with the user profile settings. We would have tried to rectify this, but we were not able to edit the setting. So, for now the recovery functionality does not work.

7.2.4 Posts

We have implemented a fair amount of working features regarding the access, publication, editing, and deletion of posts on our site. All post interactivity systems we wanted have been implemented, and we have achieved additional features like text highlighting for different languages and the ability switch between light and dark mode. However there is still some useful functionality that we are currently missing and some unexpected drawbacks that we have discovered following implementation and internal review.

One such potential drawback that we have identified is that it is could be harder to look at a submission 'as a whole' as one could not view all contents of a journal submission within one web page. However, it is worth noting that all posts belonging to a journal are kept together and the user could open each file in a new tab and switch between them when viewing the submission at large (which is the intended way to look at a journal in full at this time). We expect that having multiple tabs open, each with a different post (and hence a different web page), should be more comfortable to a user than having the same web page open multiple times with different sections open in each page, or repeatedly having to switch sections. However, we conclude that there must be a more user friendly solutions and additional research aimed at identifying said solutions is required.

A slight oversight of our current system is that a user cannot currently edit the title or description of a post. This is something we failed to consider and have not accounted for in our user stories, and thus, have not developed. Here is is worth noting that a lot of these ommissions we were able to detect thanks to the HCI evaluation we have conducted, alongside the follow up review of how users might interact with our system and what they currently are not able to do. Unfortunately, we only managed to begin working on some identified areas and such would only be able to continue given additional time.

7.3 Comments

190020048

Commenting allows users to engage and respond to a post. Users can create and delete comments, in which both features are fully functional and tested. The comments are organised chronologically, so users can have a look at the history and time record of comments. Nonetheless, there are additional features that could be implemented to enhance the usability of the commenting system and meet users with different needs. One suggestion to improve commenting is to allow the editing of comments. Users can therefore have the freedom to change and then update comments rather than deleting the existing comment and posting a new version of it. Another suggestion for improvement is to implement admin moderation in the commenting system.

A commenting solution with spam and moderation filters will help users to manage comments and keep them in good order. Different admin privileges would be helpful for assigning additional moderators to help monitor comments. Users might also generate reports to flag undesired comments that violate community guidelines. Existing social media platforms like Facebook or Instagram have commenting systems that allow users to reply and react to individual comments. This feature if implemented has the potential to increase user engagement and foster interaction between users who are reviewing similar posts.

7.4 User Interface

190013199

Overall, we were able to make a clean and easy to follow user interface which offers simple and familiar ways for the user to navigate and interact with all our systems. We made sure to follow good UX design principles across all components and have successfully achieved our objectives regarding clarity and accessability.

One problem with our User Interface is that the website is not currently usable on mobile and should only be accessed on wide screen devices. This problem was originally birthed with our initial rush to implement a MVP, in which we forgot to consider the mobile experience. As such, the design we created (and later substantially expanded) was not responsive to any degree and the problem remained unaddressed until late into the development of the final deliverable. Overall, we decided to focus on finalising in-progress features and debugging as opposed to responsiveness issues and thus, we were only able to address some of these problems. This paired with a team wide low level of technical css experience really impeaded any progress that could be made towards improving the responsiveness of our site on mobile and thus unfortunately the UI experience and usability on mobile remains generally inadequate.

In comparison to modern desktop web applications, our UI generally holds up and definetly matches a lot of the design principles used in the modern web. While we cannot hope to compete with enitre UX departments forging cutting edge user interfaces, it remains fair to say that we have done more things right than we have wrong. Therefore, our user interface, and the user experience it offers, is generally positive and we conclude that it is suitible for our implementation given our feature set.

7.5 SG Login

200012696

We initially though asking the user for the login and password and then querying the users home journal to verify it. but we decided against this due to security reasons as we were not too keen on sending confidential data. Instead, we decided to adopt functionality offered by google which logins in the user themselves and then informs the application which wishes to verify the user using their Gmail credentials.

We have decided to utilise tokens and state values to enhance the security on the system. When data is being sent from one server to another this ensures that both parties can be trusted and is not forged. Although, this makes the protocol a lot more complicated, it has massively benefited the security aspect.

Evaluating the development process and interaction, I would say that the user of the API definition tool ‘swagger’ was a huge success. This meant that although there were multiple teams working on different journals, we were all able to receive the same information and coordinate our selves in such a way our systems were compatible.

Looking at other journal I believe it would be a lot more user friendly and interactive if the user was able to choose which supergroup, they are from using buttons or a drop-down menu, instead of having to enter it manually. This functionality could not take that long to implement so given more time we would have been able to add this. This is not an essential feature, as it would not make a difference to the underlying functionality.

7.6 SG Import + Export

190003657

The implementation process for import and export was idealistic and realistically not a very good approach. The idea was to have it running against ourselves before then testing it against another group to ensure it was working as intended. However, due to a misunderstanding of how the protocol worked and what each part of it truly meant, it lead to development issues which could likely have been circumvented with greater communication between the developers, super group representative, and the super group overall.

Initially, in order to meet the MVP, an implementation of the two was done following our understanding of the spec, however it was untested. Continuing onto the next semester, this lead to issues with having to debug code that the developers had not seen for a while, add in an ever changing api, development stalled on feature for a while.

Over time though, as we realised that we needed to communicate more with the super group in order to clarify parts of the api which we weren't sure on, development progressed. With the addition of postman and the understanding of the yaml structure of requests and responses in addition to how to structure them using axios, a fully implemented version of the two was completed.

A point that helped speed up development was communication between our team and others in our supergroup, as this allowed a better understanding of the protocol, speeding up our development, and helping each other debug our sites when ever errors occured. Another would be the use of the startup script, as it allowed development when the site had crashed to speed up tremendously.

The main difficulties faced were the transition from testing against ourselves and then against other groups, as the realisation that we hadn't actually perfectly followed the protocol as we though we had was understood then. Leading to much of the two needing to be rewritten. The other would be learning the yaml file structure of requests and responses, understanding errors and why they were occuring, and fighting the javascript interpreter as it would occasionally not bother to show you an error even though there was one. Th initial point required us to rethink how we approached the protocol, and the latter required much learning through trial and error.

Finally, a problem that probably could have been circumvented was the need to configure our sites structure to fit the protocol, and vice versa. For example, the site not being able to read zip files. Again, more communication and more care in the design of the site would have been able to solve this.

In the future, if we were to come back to this project, I believe fully defining the main parts of the api and then having our group build our site around it, rather than building the site and then the api, would have helped smooth the development cycle immensely. As in this way we could make sure we satisfy all the features of the protocol and then build extra features that we want on top of it.

7.7 Recommender System

200013403

To evaluate and optimize the strategies we build the recommender system, below experiments are set up with the MovieLens dataset.

- **Depth**
  Inspired by Paul Covington et al [1], we set up experiments with depth. Adding too much width and depth were added will cause the incremental benefit to diminish and convergence to become difficult.
  - Config 0 (Depth 0): A linear layer simply transforms the concatenation layer to match the softmax dimension
  - Config 1 (Depth 1): 32 ReLu
  - Config 2 (Depth 2): 64 ReLu $\rightarrow$ 32 ReLu
  - Config 3 (Depth 3): 128 ReLu $\rightarrow$ 64 ReLu $\rightarrow$ 32 ReLu

Therefore, complex models also have their disadvantages. The first is computational cost, as larger models require both more memory and more computation to fit and serve. The second is the requirement for more data: in general, more training data is needed to take advantage of deeper models. With more parameters, deep models might overfit or even simply memorize the training examples instead of learning a function that can generalize.

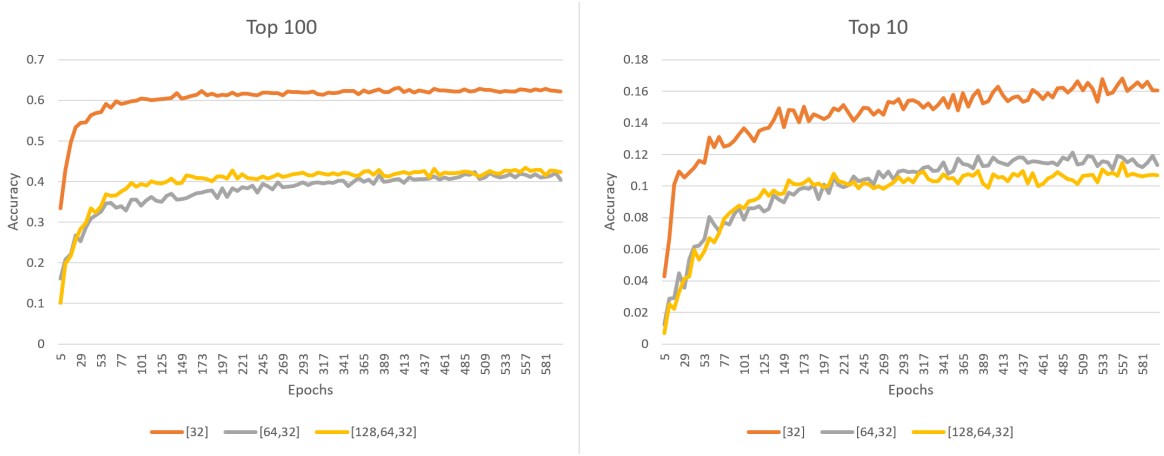


Figure 4: We found that introducing depth could improve precision on holdout data and often have higher expressive power.

- **Embedding Dimensions**
  We found that higher values will correspond to models that may be more accurate, but will also be slower to fit and more prone to overfitting.
- **Learning Rate**
  A higher learning rate can quickly improve accuracy but causes over-fitting. A small learning rate costs more to make the model convergent.
- **Overfitting**
  We found that before the accuracy reaches its peak, the increase of accuracy with time is rapid but once we cross the peak, the rate of accuracy stalls or even slightly decreases. We think this is evidence of overfitting because the loss is increasing. The validation process detects the problem and then tries to trade the accuracy for generalizing the model.

In order to balance the performance and costs, we set hyperparameters of our recommender system as below.

```
SPLIT_RATIO          = 0.8
BATCH_SIZE           = 1024
MAX_TOKEN            = 10000
EMBEDDING_DIMENSIONS = 32
LINSPACE             = 1000
MODEL_DEPTH          = [32]
LEARNING_RATE        = 0.1
EPOCHS               = 600
```

We think 62% accuracy for the top 50 recommendations is good enough given such a short time to implement. We can see the rest 38% as some kind of feature that allows users to explore what they are not currently interested in.

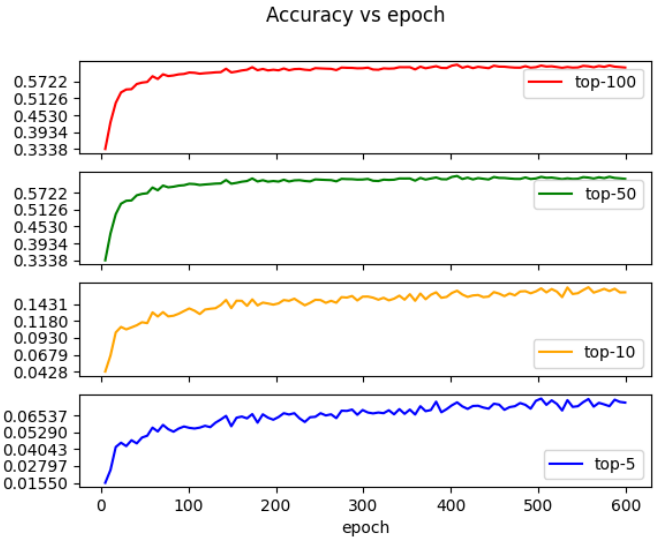


Figure 5: shows the accuracy of the model we trained for MovieLens: 63% (top 100), 62% (top 50), 15% (top 10), 7%(top 5). We then applied the same strategies to build another RS for the CS3099 Journal system.

7.8 DevOps

190003657

The implementation of the scripts helped immensely with the development of the site. Let us compare it to how it was previously:

If you choose not to run the startup script and wanted to do things manually. You would first need to activate the virtual environment in the root of the repository. Then you would need to navigate into the folder for the backend and start that up, from which you will need to create a new terminal to run the front end as the output of the backend will take over your first terminal. This is also assuming you have already set up the required node modules etc. for your machine. Only then would you be able to begin development on the site.

This has therefore overall saved many hours of development time for out team over the last semester as we no longer need to spend that time starting up the site, and can instead have it up and going within seconds. If we were to be generous and say that it took 20 seconds to do the above manually, that's still at least a 4 fold increase if it were to only take 5 seconds now. Spread that over 5 people and then over an entire semester, time is now used more so where it matters.

We planned on also continuing onto implementing continuous integration and deployment. However, due to needing to focus on more important features such as super group import/ export, we were unable to implemenet this. If we were to come back to this project, this is definitely where we would head into next, as it would also speed up our development and smooth the process over for everyone.

7.9 History

190013199

Our implementation for the history feature matches the objectives we set out to achieve. Our viewing history is easy to use and offers useful and fully functioning tools for helping users find what they are looking for. Every button has an intuitive purpose and most users should find the features and overall design familiar and comfortable to use. Given our current plans, the History section has been 'completed' and all parts have been shown to work as expected.

There is however a minor issue with how user history works, and the issue has been found to be fundamentally rooted in our current database design. In our system, history elements are relationally linked to the posts within our database. Therefore, if a post gets deleted, then any associated history elements get deleted as well. This can be problematic as the user may be looking for a deleted post which no longer exists and thus any reference to it has been pernamentally deleted from their history aswell. Importantly however this issue is not a bug nor unexpected as we must delete all database entities that reference a post if we are to allow deleting the post itself. Currently we have not addressed this problem given that history was one of the last features we implemented, and more discussion would be required regarding the best course of action taking into account both systematic database integrity and the user experience.

Most likely, the best solution for this would involve ending the FK relation linking both relations while providing functionality for accessing deleted posts via history. Perhaps having deleted posts be 'archived' instead of pernamentally deleted could solve this but we are yet to implement any archive functionality in our system nor was it ever officially in our product backlog. Hence, as the ideal solution for this problem requires additional development we cannot afford at this time, the issue must remain unresolved until further notice.

Some additional functionality that we could introduce to improve our viewing history feature includes some form of search by date or by time period (Today, last 7 days, This month, etc.) similar to the browser history tab of firefox. This functionality is justifiably useful to the user given how people utalise history features in other contexts and it would not overcomplicate a feature that we intend to keep relatively straight forward. Additionally, we do not currently allow the user from clearing their history or removing individual history elements. We did not consider this to be high priority given the context of academic code, however, the global popularity of being able to remove ones history for a variety of reasons merits it's existence within our viewing history and thus is a valid extension for this feature.

7.10 Notifications

200012696

Upon evaluation of this feature, we have concluded that it would be much more beneficial if the notifications were displayed to the user on the sidebar. This conclusion was reached as the user might find it a nuisance if they are constantly alerted through email. This can also be seen on other social media applications like Facebook, as it is more convenient for the user. We had made steps to make this happen but due to the lack of time we had not been able to fully implement it.

We had also encountered an issue with sending emails. While testing locally, the system was able to send emails seamlessly and notify the user. After the system was deployed the system was unable to establish connection with the Gmail server. After some debugging, we have concluded that it was due to the settings on the cs3099user03 server that the issue arose. We tried to fix this but due to the restrictions on the user profile we were unable to make any changes.

8 Human Computer Interaction

190020048

Jakob Nielson's 10 general principles of usability heuristics for interactive design was applied in an original analysis of the current code journal's user interface. Below shows the list of heuristics and the identified features and issues which are explained with brief justifications. Suggestions for improvements were also provided for the issues identified.

List of Heuristics	F/P	**Identified Problem or Feature**	**Brief Justification**
#1: Visibility of system status	F	User reminders display the access permission – e.g. 'You may not see the Upload section if you are not an author'.	Users can know the permission they have in the journal.
		User statistics are displayed in the side bar on the 'Home' page.	The number of posts, journals, comments, and likes are reported to the user statistics on the side bar on the 'Home' page so users can get detailed reports on engagement and comprehensive insights into publishing behaviour.
		Loading icon for page transitions and time to query from the recommendation system.	Indicates that the system is in a transition state and is loading a new page (loading folders and video). Reassures the user that a longer wait is normal, and that the system is still working.
		Button changes colour on hover – clear feedback on the user's action.	The colour change of a button communicates that the system is working and reduces uncertainty. It prevents users from tapping the same button multiple times.
#2: Match between system and the real world	F	User-centric language such as 'Your Posts – relates to the user in first-person pronoun.	The user of first-person pronouns such as 'my' and 'me' relates to the user on a personal level. This immediately connects the features with the user.
		Recognisable terms and familiar language/instructions, e.g., 'Title' or 'Description' in input fields	The use of recognisable and familiar language allows users to intuitively understand the instruction to enter a title of a post and the input functionality.
		Navigation bar is in a place that is familiar to the user – navigation is at the top of the page.	The navigation bar is designed with the users' habits in mind.
#3: User control and freedom	F	Users have the option to 'dismiss' reminders that stick to the bottom of the screen.	Users have the option to dismiss reminders at the bottom of the screen if it is blocking their workflow.
		Users can choose the number of posts that are displayed in the list of viewing history	Users can control the amount of information that is displayed on the interface.

List of Heuristics	F/P	**Identified Problem or Feature**	**Brief Justification**
	F	Users can exit the Help Menu using the '×' button to return to the main page.	Users can escape the Help Menu and resume their work with a clear exit button.
		Users can delete posts and comments	Users have the option to delete posts and comments
	P	Cannot restore a deleted post or comment – a post's data is permanently deleted and cannot be recovered.	Users cannot undo the deletion of a post because the action is permanent and unrecoverable.
		Cannot delete user account or edit user information	Users cannot delete their accounts or edit and update their information.
		Authors cannot change the access and sharing permission of the posts	Authors might want to keep some of their posts private. However, all posts published are public.
		Users cannot opt-out of notifications	Notifications of other users commenting on their posts are notified via email but there is no option to unsubscribe notifications.
#4: Consistency and standards	F	Common website components e.g., notifications, modals, search bar lower the learning curves of users – user-familiar and consistent components increase user learnability and ease of interaction.	Beginner users can easily understand how to interact with common website components, and they do not need to learn new interactions. Users can therefore focus entirely on browsing post content.
		User account login, home page and features are well-design and follow industry standards	Users are familiar with the layout and instructions for signing up or logging into an account.
		Most icons, buttons and page layouts are visually consistent	Icons and buttons used on the website closely align with how other sites represent the same concept, e.g., submit and delete buttons.
		Common utility tools, e.g., Save, Submit, Delete are externally consistent	Visual placement of utility tools is designed in compliance with ruling web conventions.
		Home page and field titles remain a necessity and are externally consistent	There is an explicit link called 'Home' to minimise home page navigation confusion. Users can go back to the home page as a common task. It gives users a starting point if they are lost when exploring the website.
#5: Error prevention	F	User confirmation to permanently delete a post	Prevent users from accidentally deleting a post
		Prevents the user from uploading unsupported file formats	There is guidance on the type of file formats that the code journal supports.
	P	No auto-complete search suggestions for the search bar (#6: Recognition rather than recall)	There is no auto-complete to prevent user slips from performing queries with typographic errors.

List of Heuristics	F/P	**Identified Problem or Feature**	**Brief Justification**
	P	The text editor does not support undo and redo to prevent mistakes (#3: User control and freedom)	Support undo and redo of edits can prevent user mistakes. Providing the ability to undo the most recent action helps users to feel more secure and more confident to experiment with unfamiliar features, since they are aware that a mistake is low cost and can be easily fixed.
#6: Recognition rather than recall	F	Viewing history on the user account page	The viewing history is a record of the addresses of the posts a user has recently visited, and data associated with those websites. The saved data helps make the website load faster if users revisit it.
		Posts tags are categorically organised in the search results. Tags help users to browse and explore specific content.	Posts with similar tags are categorically organised in the search results. Users can easily identify similar and relevant posts by clicking on the tags.
		Placeholders of input fields serve as prompts and instructions (#10: Help and documentation)	Help users to recognise what information is required in the input fields. For example, 'Title' for the video title, 'Press ENTER to add a new tag' for users to add a new tag when uploading a new post.
		Buttons on the navigation bar are recognisable and follow the user's mental model (#2: Match between system and the real world)	For example, the recognisable 'Home' page button allows users to easily locate featured and posts that are available.
		User account information provides a stronger cue for user control (#3: User control and freedom)	User account information reminds the user is signed in and provides a stronger cue for user control and video management.
	P	No auto-complete search suggestions for the search bar (#5: Error prevention)	The search bar does not provide users with relevant search results while typing in keywords. Users might need to recall more relevant keywords for better search results. This increases the user's memory burden.
#7: Flexibility and efficiency of use	F	Users can post multiple files for a post	The code journal supports multiple files for a post to the group and organises the files in categories.
		Keyboard shortcuts serve as accelerators for expert users – speed up frequent switching of pages.	Keyboard shortcuts are available as an alternative way to switch pages for expert users. For example, 'alt + b' switches to the 'Home' page. Accelerators improve the efficiency of repetitious page transitions.
		Users can upload and download files from posts	There is flexibility in accessing the files in posts.

List of Heuristics	F/P	**Identified Problem or Feature**	**Brief Justification**
	F	Users can upload images and videos. The website shows a preview of images and video files.	Users can preview an image or a video before downloading a file.
		Users who are authors can edit the source code of files.	The text editor allows authors to edit and change the content in source files. Users can then save new versions of such changes.
		Hierarchical roles allow access to different user features – allow personalisation: users have different needs at different times.	Well-designed roles allow personalisation of user features – i.e., 'Viewer', "Author', 'Reviewer' of posts.
		'Upload' supports a broad range of file formats – allows user flexibility.	Users have multiple methods to accomplish the same task. The file to be uploaded can be in different formats, so users can select a preferred format that works for them.
	P	Users cannot customise the web interface to suit their needs	Expert users might want to customise the interface to suit their needs of peer-reviewing posts or organising posts in folders. The interface should adapt to the user's dynamically changing needs. For example, a user can set up multiple window arrangements (workspaces).
#8: Aesthetic and minimalist design	F	Organised interface components: the Home, Upload and Account pages all adhere to the Gestalt proximity principle and have high usability.	The white space between fields makes the user's uploading and logging in process intuitive and convenient. Input fields are legible and easy to read.
		Syntax highlighting is a feature in the text editor of posts that displays text, especially source code, in different colours and fonts according to the category of terms.	Syntax highlighting focuses on language-specific keywords, operators, and elements that have the same meaning in any given piece of code. The code is more legible and easier to understand.
		Contrasting colours of buttons, icons and page layouts are consistent. • Colour scheme enhances the contrast between the text and background	The journal adheres to a consistent colour scheme: orange, black, white, grey are prominent colours in the user interface. The website's text contrast increases the legibility and usability. It also makes the design memorable.
		Essential information for decision making is displayed on most pages High signal-to-noise ratio – e.g. novice users can use the 'Show Tour' guide in the Help Menu.	There is no unnecessary noise or information on most pages and users have a clear mental model of the actions required to achieve the desired outcome.

List of Heuristics	F/P	Identified Problem or Feature	Brief Justification
	F	Gestalt proximity principle is applied to user input design • Field labels are close to their corresponding input fields.	Field labels are visually closer to the user input fields which increases usability and proximity.
		Visual hierarchy of website components is proportional • Design elements are organised according to the order of importance.	The sizes of the website buttons, icons, title, subtitle, field labels are proportional to their importance. The scale visually controls the delivery of the user experience. Small, medium, and large components in the design are easily differentiable.
		Scalable and responsive web interface across mobile devices	Chrome's inspection developer tool shows that the web application is responsive on mobile devices.
		List view of post recommendations in 'Home is clear and there is information of posts.	Clear layout of posts in 'Home'. It is easy for the user to navigate from folder to folder. The meta information of posts helps the user to easily browse the posts that they are interested.
#9: Help users recognise, diagnose, and recover from errors	F	Error message of denying the user access to a deleted post – redirects users back to the home page.	The error message of denying the user access to a deleted post is complemented with a solution to explore other posts.
		Error messages are written in a consistent style and tone of voice: •Readable plain language •Polite tonality •Precise descriptions	The code journal does not use technical jargons to overwhelm users and error messages are written in precise descriptions to assist the user in troubleshooting.
		Tips and guidance messages stay away from user criticism: • Constructive advice • Explicit indication of an issue	Rather than criticising the user's error, error messages explicitly indicate the issue and offer users constructive advice.
		Error messages are displayed as pop-ups on the browser	The code journal clearly informs users when an error has occurred with an error message displayed as a pop-up. A combination of visual treatments and a message indicates that an error has occurred.
	P	The code journal clearly informs users when an error has occurred with an error message displayed as a pop-up. A combination of visual treatments and a message indicates that an error has occurred.	If no search hits were found, the original query terms do not allow users to revise and search a wider scope without retyping queries.
#10: Help and documentation	F	Users can start of skip onboarding 'Show Tour' tutorials from the beginning (proactive help)	Familiarise new users with the interface at the first launch.
		Confirmation before permanent deletion of data (proactive help)	onfirmation to prevent user mistake of permanently deletion.

List of Heuristics	F/P	Identified Problem or Feature	Brief Justification
	F	Online documentation and help are readily available for users in the Help Menu	Users will be directed to the Help and Documentation page by clicking on the 'Browse Support' button.
	P	Alternative text is not displayed on hover over buttons	Push revelations to assist users is a suggestion for improvement.
		'Show Tour' in the Help Menu provides a limited walkthrough for beginner users	Real-time user support is not available in case of an emergency.
		No live support on the website	Real-time user support is not available in case of an emergency.

9 Conclusion

Everyone was satisfied with their roles in the team, and we believe we have all be able to contribute with the best of our ability. By using the SCRUM Organisational method, we were able to assign tasks which best suited our skill set. Also, through the user of a group chat we were able to coordinate well and offer and ask for help. We were able to utilise the sprint backlog to schedule ourselves and to ensure we were on schedule to implement the necessary features. We have also been able to utilise the user stories to decided on the features which the user desired.

We have been able to make the most out of the weekly supervisor meeting by taking minutes and paying attention to the advice and guidance which was given. We have also seen the supervisor as a potential user and been able to ask questions about features which they would like to see in the system. Some of these features is the notification feature and scripts to start-up the system with ease.

We believe we have made substantial progress since our last deliverable and implemented several new and complex features: Super group Login; Super group import/export; recommender system; Enabling Login and registration; having user profile and statistics (such as number of journals published by the user); site wide search; interactive guide; keyboard shortcuts; viewing history and many other features.

One of the greatest drawbacks of our implementation concerns security, which is an area in which we lack both knowledge and experience. For example, the passwords of the users are stored without encryption. This could be a potential security issue. Furthermore, the password that is being sent to the backend by the client is sent as plaintext as well.

For future improvements, security was one of the key things we would focus on. We could implement hashing and salting on the backend when storing the passwords and other confidential data. This would mean even if there happened to be unauthorised access to the database, the essential data will not be compromised. Also, we could be very keen to implement the inline commenting feature. Currently our system can only accept comments for a whole file or post, it can not take comments for each line in the code. We believe this feature would be very beneficial for the user as it helps them to interact with the code in a more specific manner.

A Testing summary

Testing was generally done manually, with the help of sites like postman in cases where they proved to be usable. This would be things such as uploading test posts to the site, testing buttons, using posts to import and export between other journals etc. In addition to this, logging results in the backend to make sure everything is as intended and using functions to do work for us when possible - e.g. validation and generating id's.

In the future, if we were to come back to this project or advise a new team who were to take it over - using

a test driven development methodology would likely help smooth development immensely as it would reduce the number of errors by having the developers focus just on having the correct inputs and outputs for each of their components. This could be done by using things such as unit tests and factories to generate new test cases to run against various methods.

A.1 UI Testing

Testing elements of the UI was a very manual process and probably the most time consuming part of development. Usually, UI testing would involve both the f12 console of a browser and the logging output of the server to determine whether UI elements were performing the expected functionality, or if not, where the error was occuring. Each and every button, tag, icon and any other component of our UI that offered interactivity was manually tested at multiple stages on both group account and localhost hosting. Additionally, we performed edge and exceptional case testing by trying to use our UI elements in an unintended manner. for example using an email address as a phone number during registration, or intentionally trying to go out of bounds using the viewing history functionality.

An additional part of our UI testing involved checking how easy to use some element are. As we had to carefully adhere to good academic practice, external testing was not something we could conduct at a useful scale, at least not without putting ourselves in a gray area and perhaps compromising our project.Therefore, we had to rely on one another for testing new features. This was deemed acceptable in this context, so long as the tester was not responsible for the development of the given feature. Of course, this has given us a testing sample of extremely computer-fluent users and thus not reflective of the average computer owner. However, since the target audience of our site consists of programmers and computer scientists, the testing sample should still accurately reflect the target demographic of our product, so this isn't a problem.

Overall we were able to discover and fix multiple UI related issues and believe that our UI offers both effective and simple solutions which cover their intended use case, whilst keeping the system resilient against unintended use across all components of our user interface.

A.2 Supergroup Testing

We had a lot of problems in regard to supergroup functionality and testing. In general, the problem was that there were minimal testing opportunities until the end of the timeline for project development. From our perspective, we had a mostly working solution for a long time, however needed to iron out some edges before our implementation could support the full protocol. Testing was needed for this, but we had nothing to test against.

To resolve a lot of our issues, we used postman to send requests both to our journals and the journals we were testing against to get greater insight into what was happening behind the scenes and how requests where being handled. In addition, we used a text compare tool to directly compare jsons sent by us and other groups to identify problems and irregularities in our implementation of the protocol and by combining the tools offered by text comparison and postman we were able to fix a lot of issues and achieve working functionality.

Once we achieved a correct protocol implementation, we were able to fully test it by attempting to use it with other groups that currently had their supergroup functionality live on the group accounts. If we could get our super group functionality to fully complete a protocol cycle against another group, then that could be seen as indication that our protocol is correctly implemented (assuming of course that the success is repeatable). We aimed to show the functionality working against at least 2 groups where possible, although our ability to perform this final stage of testing was limited by factors outside of our control, and given the limited time, we had to ensure that we manage to finalise, proof read, and submit our project before the deadline. Overall, based on the testing we were able to conduct, it appears that we have managed to implement the protocol succesfully, as defined by the supergroup documentation. Given more time and testing opportunities we would aim to continue testing as to better affirm the status of our implementation.

A.3 Other Testing

Other methods of testing not mentioned above include getting the yaml file of responses and requests and comparing them with the expected input/output to ensure that we were using the correct headers, parameters etc. Logging the results of tests and then verifying if they were as expected. A more manual approach to unit testing if you will.

Console logs were also used frequently when responses were received and sent off. This way we could see exactly what was occuring and decipher the problems that we had in our process. e.g. if an export request was inititiated but never ended up sending a file as the forward file success message never occured.

Weekly Progress Report

Week	SEMESTER 2 What did you do this week?	What will you do next week?	Anything blocking your progress?	Would you like t	Preferred meeting time
5	Bug fixes, work on the front end, fixing UI problems and making the website more aesthetically pleasing	May not get much done due to deadlines but User Metrics and home page side bar are my main priorities	deadlines	Yes	15:00:00
5	Wrote report, got export working, working on import, brought project in line with fixit	finish import and other tasks in planner	deadlines	Yes	15:00:00
5	Fix bug;Search	History Table in Database; post ranking	No	No	15:00:00
5	Working on Deliverable 3	try to start to get my head around nginx and using that to implement functionality for super group login	practical that are due next week	No	15:00:00
5	Downloading posts and researched syntax highlighting for post source code	Refine front end pages and UI elements	Deadlines	Yes	15:00:00
6	Tags; Advanced searching for users posts journals and tags;	Start RS	No	No	15:00:00
6	unable to do work due to deadlines for all other modules	continue working on import export	previously it was deadlines	No	15:00:00
6	I wasnt able to do a lot of work this week due to incoming course work but still managed to look at how the nginx site would work	try to fully implement other users logging into our journal	no	No	15:00:00
6	nothing worthy of mentioning for the group projects, deadlines were king this week	continue where I left off	deadlines	Yes	15:00:00
6	Refined front end pages and UI elements	Help and documentation pages for better user onboarding	Deadines	Yes	15:00:00
7	Reading RS paper	Start to implement RS	No	No	15:00:00
7	Fixed a few bugs with registration and implemented functionality that lets users from other super groups login into our system	try to implement notification feature	making a start on Networking practical	No	15:00:00
7	side bar, bug fixes, maintenance, covering of edge cases and things we forgot to cover	a basic footer plus work on additions to sidebar. probably going to do more maintenance as well	no	Yes	15:00:00
7	startup script now accommodates group account, import is now receiving posts, general debugging	continue working on import export	the protocol	No	15:00:00
7	Keyboard shortcuts, manual testing and bug fixes for comments	Pop-up modal for user accessibility and instant support	No	Yes	15:00:00
8	Recommendation System part I	RS part II	No	No	15:00:00
8	started on history, footer + css, started on stats, nav bar improvements plus fixes, other bug fixes and css fixes	viewing history	no	Yes	15:00:00
8	Implemented notification functionality so the user would receive an email every time a comment is made on their post. Could try to extend it further so they can view it on their account as notification instead of just emails.	Implementing Password recovery	nothing	No	15:00:00
8	Rewrote the pathing in nginx to accommodate the supergroup protocol as previous implementation would not have allowed it, reflected changes in relevant files. Progressing on export	Continue working on import export	Yes	No	15:00:00
8	Help menu modal, keyboard shortcuts, user accelerators	User walkthrough and onboarding tour with tool tips	No	Yes	15:00:00
9	Connect RS to Server; Video; Image supports; Syntax highlighting	Report and evaluation	No	No	15:00:00
9	User onboarding tour, help and documentation page	Tooltips and human computer interaction review	No	Yes	15:00:00
9	Created more comprehensive logs in the supergroup files to help better understand how things are flowing. setup script now also sets up python libraries for the recommender system, startup now works properly with no errors, exported files working as intended	continue working on import export	deadlines are back yet again	No	15:00:00
9	Viewing History, started on report, user statistics, export option expansions, css fixes, viewing history css and formatting, bug fixing	report and finalize project	my brain being out of juice	Yes	15:00:00
9	Implemented and tested forgotten password functionality on the local host. need to migrate it to nginx	Allow our users to login to other journal using their login	nothing	No	15:00:00
10	Report and Evaluation	Demo if there is one	No	No	15:00:00
10	Import export now works fully with another team, users are now also exported if asked for, filled in my sections of the report	demo	Yes.	No	15:00:00
10	Read me, report, helped with finishing import export, finished stats, comments, cleanup	presentation to alex, also going to struggle through complexity	no	No	15:00:00
10	Implemented functionality that enables users to login to other journals using their home login and worked on the report : super group login, user , notification and conclusion.	prepare demo	other practicals	No	15:00:00
10	Human computer interaction review and heuristics analysis	Demonstration, supergroup and review meeting	No	Yes	15:00:00

References

[1] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16). Association for Computing Machinery, New York, NY, USA, 191–198. DOI:https://doi.org/10.1145/2959100.2959190

[2] MovieLens Dataset https://grouplens.org/datasets/movielens/100k/

[3] MERN stack https://www.mongodb.com/mern-stack