Junior Honours Project

Federated Social Media Site

University of St Andrews



Team Members

Katy Cluckie
Erin Erturan
Kitty Milnes
Kamran Sharma - [redacted]

Supervisor: Dr Susmit Sarkar
Group: B4

Date of Submission: 15/04/2021

Abstract - [redacted]

This federated social media site allows users to interact with others across different implementations. The project allows users to view and create forums and subforums and subscribe to those of interest to them, as well as posting and commenting within the subforums, all within an easy to use interface. The site can be accessed by other groups, to get information, or make new posts as and when requested, provided the correct security checks are passed. This report highlights the aims and objectives for this project, alongside the design and implementation used. It also highlights the agile methodology, and how we as a group communicated and worked together remotely, alongside focusing on the intergroup communication and protocol.

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 **12,171 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 Page

1 Introduction - **4**

1.1 Overview of the Project and Task 4

1.2 Our Aim for our Implementation 4

1.3 Initial User Stories 4

1.4 Decisions on Technology 4

1.5 Key Aspects of the Project 5

2 Software Development Methodology - **5**

2.1 JIRA 5

2.2 Scrum Document 6

2.3 Reflection on Scrum Sprint Cycles 6

2.4 What we would do Differently 6

2.5 Communication as a Group 7

2.6 Scrum Meetings 7

2.7 Unexpected changes to our team 7

2.8 Version Control 8

2.9 Supergroup Communication 8

2.10 Server Continually Running 8

3 Design and Implementation **9**

3.1 System Architecture - 9

3.2 Technologies used - 10

3.3 Interface design and Usability - 11

3.4 Design and Implementation Decisions - 12

3.5 Unusual design features - 13

3.6 Special algorithms and data structures - 14

3.7 Local Features - 14

3.8 Intergroup Features - 17

4 Intergroup Collaboration - **20**

4.1 How the Meetings Worked and how Decisions were Made 20

4.2 Protocol 20

4.3 Issues with the Protocol 21

4.4 Digital Signatures 21

5 Discuss Changes - **22**

5.1 Major Changes Throughout our Implementation 22

5.2 Protocol Changes 23

6 Ethics and Security - **23**

6.1 Encryption 23

6.2 Preventing Harassment and Abuse 23

6.3 Testing with Non-Sensitive Data 24

6.4 Digital Signatures 24

6.5 Cookies 24

6.6 Prepared Statements 24

6.7 Client-Side Security 24

7 Testing - **25**
7.1 Overview of our Testing Approach 25
7.2 Using Mocha and Chai 25
7.3 Testing the API 26

8 Evaluation **26**
8.1 Evaluating against Our Aims and the Task set to us - 26
8.2 Evaluating in Comparison to Other Groups - 27
8.3 Evaluation Overall - 27

9 Conclusion **27**
9.1 Long Term Improvements 27
9.2 Summarise Project with Key c evemen s and Key Drawbacks - 28
9.3 What we would do Differently Again - 29
9.4 What we Enjoyed and Learnt - 29
9.5 Acknowledgements - 30

10 Appendices **30**
10.1 Protocol 30
10.2 User Stories from the First Submission 31
10.3 Scrum meeting summary document 31
10.4 Test Cases 31

11 Bibliography **34**

1 Introduction - [redacted]

1.1 Overview of the Project and Task

The task set for us was to create an online community that is federated and suited to a university environment. The project aimed to create a site similar to Reddit, that allows users to create different communities and sub-communities, whilst being able to create posts, and have conversations with other users in the comments. The project being federated means our implementation has to be able to connect to other implementations within our supergroup and allows us to share data within these groups. A basic implementation would allow a user to register, create different communities and posts. It was suggested to us that we take into account the moderation of the site, as well as putting a focus on security and privacy. The aim was to implement all of this alongside a web-based interface. After these basic features were implemented, the project scope was increased to try out some extension features of an exceptional standard. The suggestions of these ranged from private messaging to search features and gave us the scope to make the implementation our own.

1.2 Our Aim for our Implementation

We wanted our implementation to be easy and clear to use. We also wanted the implementation to be similar in functionality to other discussion sites out there, but with our own twist on it. We decided to first focus on all of the basic requirements of the minimum viable product before focusing on the usability and design of the website. We aimed to allow users to be able to view, post, edit and delete within forums from other groups. These updates are also reflected in other implementations within the supergroup. We also want users to be able to personalise their account so that they can subscribe to forums and see their associated subforum post feeds. Users should also be able to like and unlike posts and comments. The aim was to create something that would allow those at university to communicate with each other and have dedicated places to post for help for different classes and to talk to people with similar interests. We felt this was especially important in the current circumstances due to the Coronavirus pandemic forcing everything to be online.

1.3 Initial User Stories

We planned our user stories at the start so that we could get an idea of what we wanted to do and what steps real users would want to take. We did this by imagining we were the end-user and made a list of how we wanted to carry tasks out. This allowed us to envisage the end product and gain a better idea of how to implement different functionality from the server and client-side perspectives. The user stories from our first deliverable are attached in the appendices, showing what our initial idea was for different tasks that the user may do, and what we would need to do to execute these as expected. (Section 10.2)

1.4 Decisions on Technology

Initially, we had set out to use Java for our backend alongside JDBC for working with the databases, however, after starting to implement these, we struggled to set it up. We then concluded that using Express.js for our backend seemed a lot more straightforward as our front end was also in Javascript, so the two of them could easily communicate with each other. We decided to use MySQL / MariaDB so that we could run the database on the school servers. We also decided to run our program on the school servers, as this was recommended to us and we felt there would be support in the event of any server failures. The CS Systems Wiki [12] was also very helpful in guiding the set up of our server and pseudo-user.

1.5 Key Aspects of the Project

To create a successful program, we followed the agile software development methodology, so that we could decide what needs to be done and when. The project focuses on a local and a federated social media site and has many different design features which will be explained later on in this report. The local and federated side are joined together with users being able to carry out similar actions both locally and globally with other groups. The testing of this project is thorough and robust with automated testing. We collaborated well with the supergroup and managed to follow the constantly changing protocol. We have met all of the basic requirements set out to us at the beginning to a high standard, while also implementing some additional features which will be explored as we go further into the report. Our site can be accessed on: https://cs3099user-b4.host.cs.st-andrews.ac.uk .

2 Software Development Methodology - [redacted]

2.1 JIRA

Throughout our project development, we made use of Jira, an online project tracking software [1]. This enabled us to add items to our backlog, assign story points and plan sprints. It provided a clear visual representation of what issues were left on the backlog and what points were in active sprints. At the end of each sprint, Jira generated several graphical representations of the sprints progress; the most beneficial for us being the burndown charts to show when during our sprints we achieved each item and the amount of story points this correlated to. On top of this Jira can also generate charts representing our progress for the whole year, this makes looking back on our journey throughout the project very easy to do.

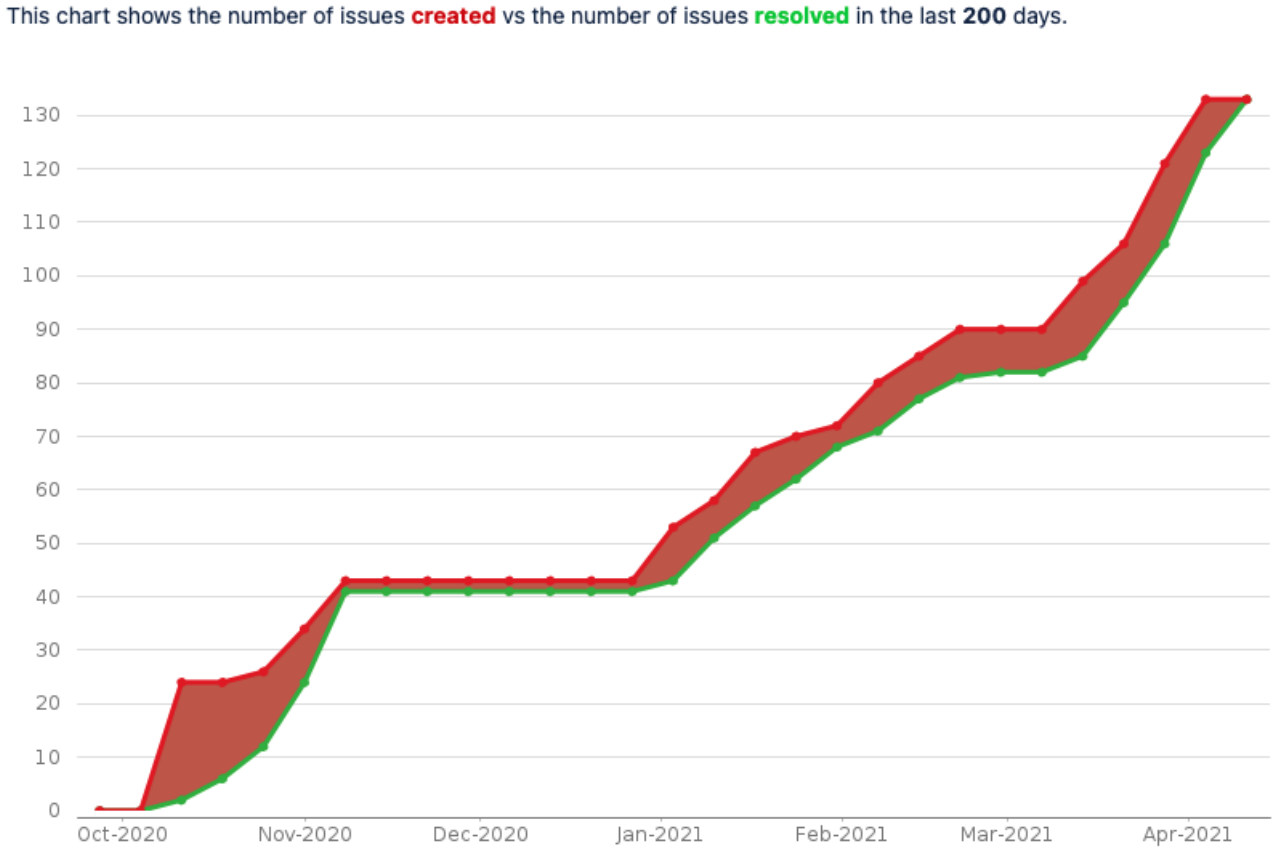


Figure 2.1 - A graph to show items added to the backlog vs items completed over time

Jira enabled us to see an accurate representation of the current sprint in progress; seeing which items were still outstanding, and which ones were in progress or completed. We could add to the backlog whenever there was a new functionality we wanted to implement or a bug that needed fixing.

2.2 Scrum Document

We ensured that we kept an up to date log of all the sprints we completed. We updated this at the end of each sprint, usually during our scrum meeting. There is a summary for each sprint included in the document listing the date, reflection on the previous sprint, items achieved during the sprint, any items we didn't achieve, new items added to the backlog and all items taken from the backlog for the upcoming sprint. We also discussed our plan for the next sprint, assigned a scrum master and decided on the duration of the sprint (Section 10.3).

This document provides an in-depth summary of all the sprints we have completed from the start of our project. It gives a good outline of our progress and any challenges or changes in circumstances we faced.

2.3 Reflection on Scrum Sprint Cycles

The majority of our sprints were productive, we aimed to follow the sprint structure introduced to us at the start of the project [11]. We found that we had sometimes underestimated the story points and that some aspects proved more difficult than we originally expected, in this case, we would reassess and alter the story points as necessary. The graph from Figure 2.1 suggests we were fairly consistent in terms of completing issues put on the backlog, and the steady rate demonstrates we were able to work consistently throughout the year, and planned our time such that we didn't have any crunch time at the end of the project.

Occasionally, other module deadlines prevented us from achieving everything we had hoped. We aimed to have a good number of tasks in a sprint so that we felt challenged whilst still being able to achieve most items.

In some cases undergoing shorter sprints worked better for us; this was often when other modules were less demanding. Over the Christmas break, we could spend more time on the project, so a reduction in the sprint duration kept us focussed and on task. We found that when deadlines were approaching we preferred shorter sprint durations, as we found they kept us more driven and provided some extra momentum.

2.4 What we would do Differently

On reflection, we believe we worked well together and have completed this project with good communication and teamwork throughout. We worked mainly as a team on group calls to make use of each other's ideas, skills and knowledge. We found this approach to be very effective for us as we felt it was efficient and ensured everyone was up to date with all aspects of the implementation.

We also tried an approach of splitting up and assigning different tasks to individuals to complete, this however provided some difficulties as many tasks, especially in the early stages of implementation, relied on other tasks being completed. This was difficult to ensure as we each had different timetables and commitments.

Our current and most effective approach is a mix of the two methods outlined. We don't assign tasks to individuals but each person works on something that is still to be completed on the sprint and

communicates this to the team. We still meet regularly and complete some aspects as a group or to discuss any difficulties and help each other with implementing features. This method has come from a natural evolution throughout the year and we have developed how we work together to best suit both our individual and team requirements whilst allowing us to still work efficiently and progress as a team.

In an ideal situation, we would have developed this technique for working together sooner however as we had no experience of working together and had never met each other before this project it is unlikely this would have been possible. We became a very different team with a different dynamic halfway through the project when we lost a team member, so have had even less time to create a method that works well for our current group of 4 members.

2.5 Communication as a Group

As mentioned briefly above for group communication we made use of a Microsoft Teams channel to chat and video call each other. One person would usually share their screen during this and edit the code whilst others offered advice, tested code and researched approaches, frameworks and fixes for any bugs we found. This worked well and we all had a chance to be involved and kept up to date in the progress of the project.

For day to day communication, we also had a Facebook group chat, where we would inform others of what we were working on, give quick advice and reveal any findings or issues we had. We also used this to plan any meetings or suggest that we may want to add to the backlog.

Our members study 3 different degree paths so have very different timetables and deadlines. We worked well as a team around this, finding times to suit everyone and using different members' expertise to help in different areas of implementation. We have always been honest and understanding of our members' module deadlines and how much each person thinks they will be able to contribute week to week. Overall we believe our communication has been frequent, and in-depth meaning that we have been able to work very well as a team and complete what we've planned to do on time, and to a good quality.

2.6 Scrum Meetings

We had at least one scrum meeting a week consistently from the start of the project (excluding some from over Christmas break). We often tied these weekly sprint meetings in with our supervisor meetings, weekly, on Fridays at 2pm, via Microsoft Teams.

All of our scrum meetings went well and as we had planned, following the scrum outline provided in the lectures at the beginning of the year [10]. All members were heavily involved in the project and therefore had a very good understanding of where our progress was at and how much we had achieved before any scrum meetings. This allowed us to have very efficient scrum meetings and allowed us more time to tackle any problems we were facing on implementing new features.

2.7 Unexpected changes to our team

We had one major change to our group and hence the group dynamic when one of our team members unexpectedly dropped Computer Science and subsequently left our team. He informed us of this on 14/01/21 meaning he had completed half a year working on the project with us. His departure from our group naturally led to some large adjustments for the rest of the group, the biggest of these being the need to discover how we worked best together with one less teammate. Teamwork and communication among our remaining members has never been an issue but naturally, we still needed to adapt to our new situation.

2.8 Version Control

We used the school's GitLab service to track and store all the changes made throughout our project. We could commit and pull any changes to ensure we were editing the most current version of our project. This was particularly useful in the earlier stages of the project as we were working on different devices and had no other way of sharing the work we had done with one another.

Gitlab also provided us with a record of each commit, meaning that if for any reason we needed to revert our code to a previous version this could be easily done. Later on in the project, we began working from our groups' pseudo-user, this enabled us all to have shared access to the same files stored on the school's servers. Despite now having access to shared files, we still made sure to commit regularly from the pseudo-user to ensure we had regular copies of versions in case we ever needed to revert to these.

2.9 Supergroup Communication

As mentioned in section 2.5 we relied heavily on Microsoft Teams for communicating within our group and with our supervisor. Our supergroup B also used teams throughout the year. We used it to attend supergroup meetings to discuss the protocol, share files, track other groups' progress and to ask questions or inform other groups regarding any changes.

One of the most useful features of Microsoft Teams for aiding communication within the supergroup was the use of file sharing features. We had access to a feature tracker allowing us to view what teams had completed different aspects of the protocol. This was supposed to provide a clear guide on which groups we could test certain aspects of the protocol against. We were also able to access the formal protocol documentation which was needed to refer to when implementing the protocol.

2.10 Server Continually Running

Once some groups had got a large amount of the protocol implemented it became important to have our server constantly running so that teams could test against our implementation, whenever they were working on it. This makes sense to do from a user viewpoint too, as users would expect to be able to have 24-hour access. To do this, we used a Node module called Forever [13]. One person needs to start the server with the command "npx forever start websiteV3.js".

When the server is running any group members using the pseudo-user can see the live logs by typing the command "npx forever logs websiteV3.js -f", this keeps all members up to date and enables us to all to work on the code separately whilst observing what others are doing on the site. We can see other groups accessing our data, requests responses sent from other groups when we retrieve their data and the logs for everything we have implemented. This allows us to continue accurately troubleshooting and debugging our code. Another advantage of using forever is that it automatically restarts the server whenever it crashes. This, combined with the logs means we can see whenever there has been an issue and what has caused it; while our site is immediately brought back online, hiding the crash from the users. This has proved to be an extremely useful debugging tool as we can utilize other groups within the supergroup accessing our site as a normal user would and ensure that there are no issues that didn't get discovered during our testing.

3 Design and Implementation

3.1 System Architecture - [redacted]

The first part of our system architecture is our database. Our database has different tables to store the data from our website in the most efficient way. The tables have the necessary fields and primary keys so that we can uniquely identify any necessary information. There are foreign key links between tables, creating explicit links between posts and comments for example. The design of our database allows us to query for different information and at times we carry out complex queries utilising inner joins [9]. We needed the userID in most tables, and it appears as a foreign key in all but two tables.

The figure below shows the Entity-Relationship diagram of our database and highlights the relationships between different tables.

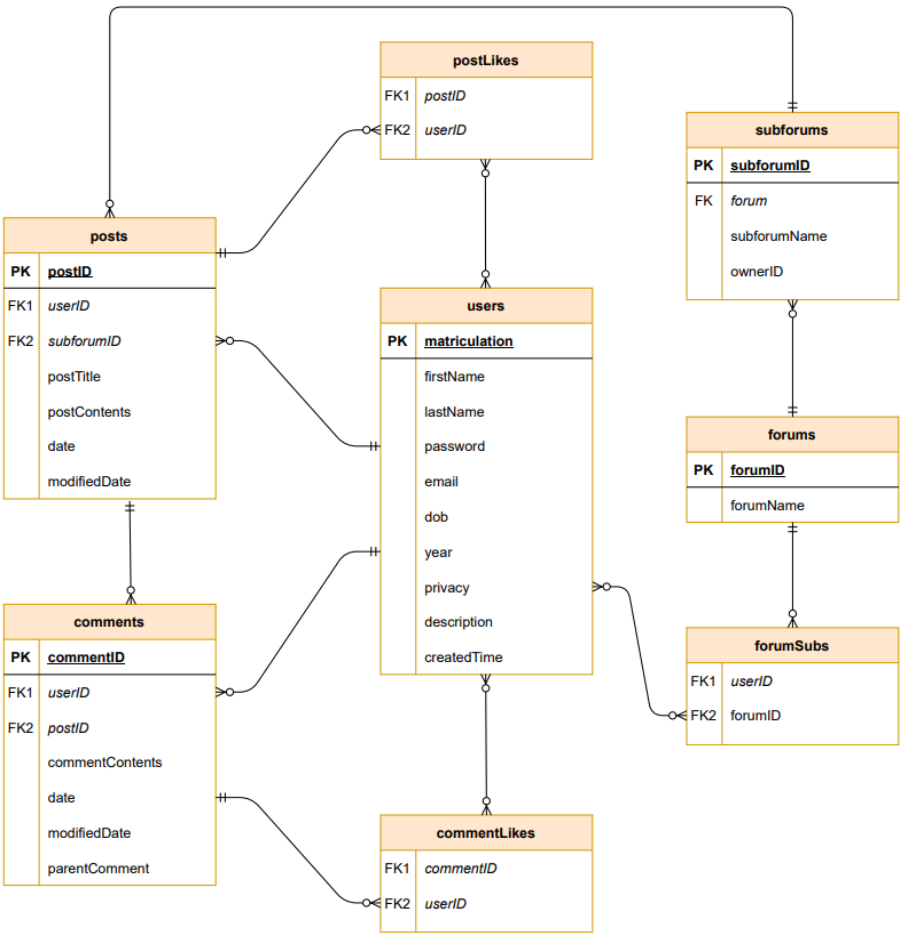


Figure 3.1 - ER Diagram showing the structure of our database (created by [redacted] using ER Diagram.drawio - diagrams.net)

We use a server that is constantly running and is listening for requests from different clients and the supergroup. Many different clients can access our site by connecting to the same server to request information. If the information requested is for local data, our system will fetch the resources from the database and then return the correct HTML to update the client's site via the server.

The client will be receiving information constantly and parsing it to display on their user interface. If one of our users wishes to request information from a group, the request will go through our server and then

will be passed on to the digital signature verification process, and if this passes, our server will process this request and send a JSON object back to the user who will parse the received information.

The diagram below highlights the key main points mentioned above and shows the different requests that clients can make, and how our server, alongside the other groups in the supergroup deal with different requests.

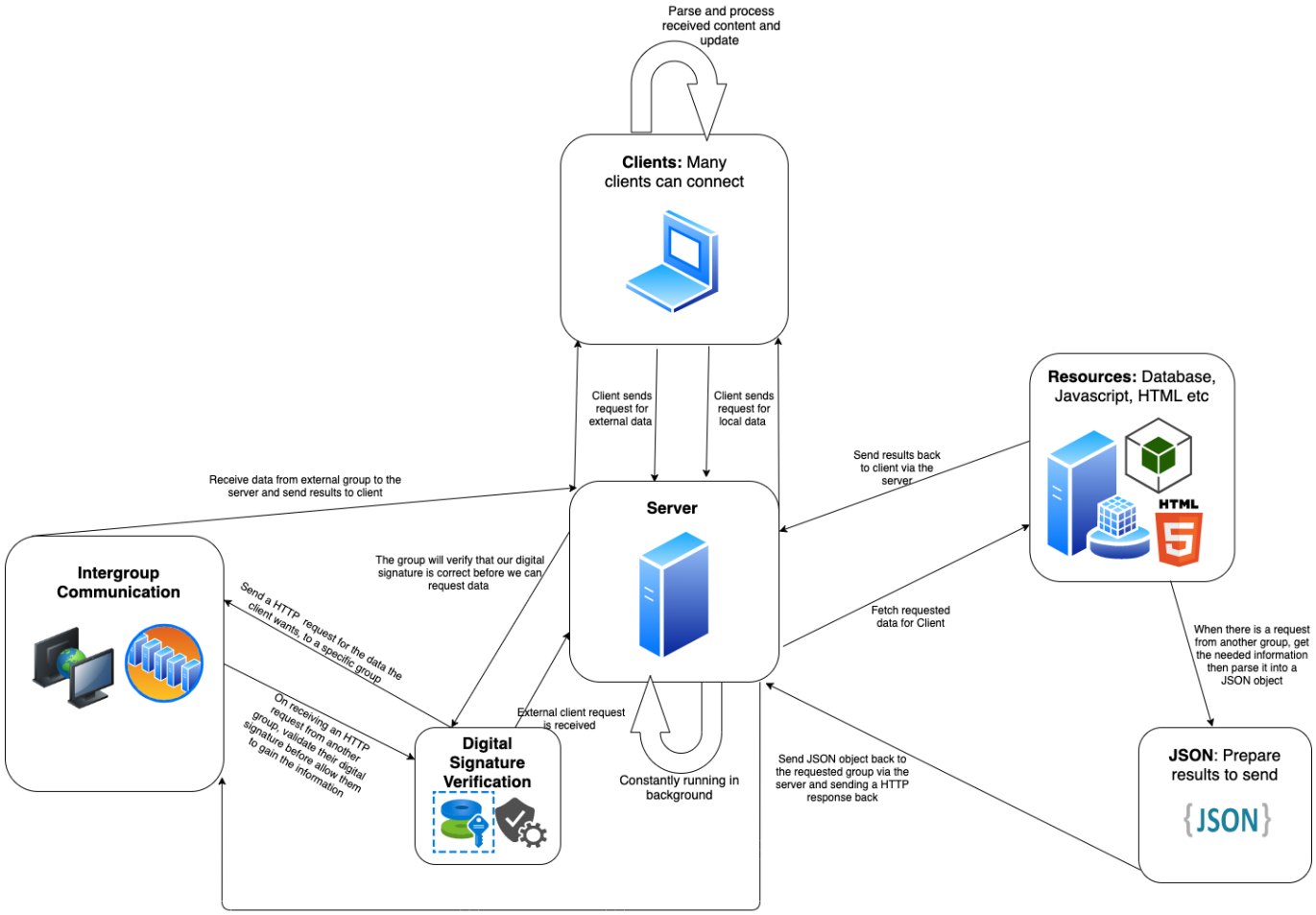


Figure 3.2 - Diagram of our System Architecture (created by [redacted] using ER Diagram.drawio - diagrams.net)

3.2 Technologies used - [redacted]

Throughout our implementation, we made use of many current technologies to achieve the end product we have today. Our backend was coded using Express.js as this felt like the most straightforward approach and allowed us to use routes to different pages for get and post requests. Node allows us to query our database using the SQL functionalities available through the MySQL module.

We chose to run our program using Node so that we could make use of the extensive amount of libraries that can be installed and executed with it. One node module that we made extensive use of was Forever, this allowed us to have our implementation constantly running and would update whenever any changes were made to our code. Our front-end was coded in Javascript, Node.js and HTML, with CSS to give the user the best experience using our website. Using Javascript on the front-end allowed us to connect to our Express.js backend with ease.

For our database, we chose to create it with MariaDB on the school servers. This meant we could query it using Node's MySQL module which was very convenient for us. For our testing, we investigated a framework specially made for testing Node.js code and client-side browser. This enabled us to write many tests which are well documented and easy to change, and at the same time to see the output of the results alongside code coverage of the tests.

3.3 Interface design and Usability - [redacted]

Throughout the development of new features in our website, we constantly found subtle aspects to add to the user interface to improve usability. For the MVP we primarily focused on getting the required functionality working with basic navigation, saving the design for semester 2. Thereafter we incorporated CSS to add design and colour to our pages and thought of features to add to improve the user's experience.

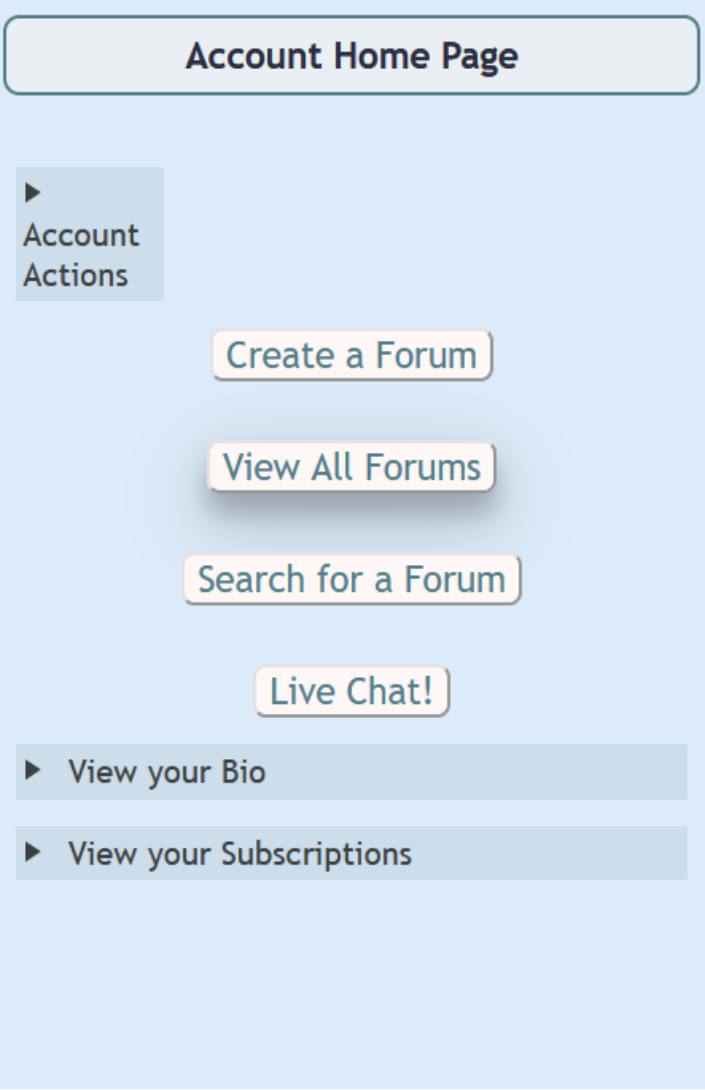
We incorporated several additional features to give the user a more personalised experience, including giving them a profile page with their subscriptions and post history, as well as an associated privacy option. We felt that it was important for the user to have easy access to the content they were interested in and had created on the website. We also added a search feature for forums to allow the user to narrow down the content they want to see, instead of having to look through a long list of forums across the supergroup. We also added collapsible sections to several pages to make the page look less cluttered and to give the user control over what they want to look at. We added links to the title of posts to help navigation since posts could be displayed on multiple pages, and these would link to the subforum page with the posts and automatically scroll down to the specific post.

We also spent a considerable amount of time on the comments interface, due to the complexity of displaying nested comments. We decided to display nested comments in an indented tree-like manner to make clear the nested comment level. Also, when replying to a nested comment, we display the whole thread up to the comment so that the user does not have to go back if they want to see the whole thread again.

Given the current pandemic, we were not able to perform user interface testing to the extent we would've liked. However, we asked those within our households, some with a Computer Science background and some without, to use our website in whatever way they would like. We watched them carry out tasks and made notes of errors that were occurring and also took some feedback from them on usability issues. One improvement we made to our interface design based on our user testing feedback was to change the cursor to a pointer on buttons and summaries, to make the elements allowing user interactions more intuitive. We also discovered a bug within one of our redirects, however, this was quickly solved after a user pointed it out to us.

In ideal circumstances and with more time to plan, we would've liked to have organised a few sessions with test users where we'd give them a list of specific tasks to do on our website, and then get feedback on their experience. We appreciate this would've been valuable input to receive as it would've been an unbiased reflection of how well the website was designed.

Another feature of our interface is our responsiveness. Since a lot of university students primarily use smartphones for social media we felt it was important to format our website nicely for smartphones as well as for larger screens which we primarily tested on. For this purpose, we had separate CSS for screens smaller than a specified amount, which was 600px. We added this responsiveness to large areas of our website. We learnt how to do this through the use of a tutorial on W3 schools [7].



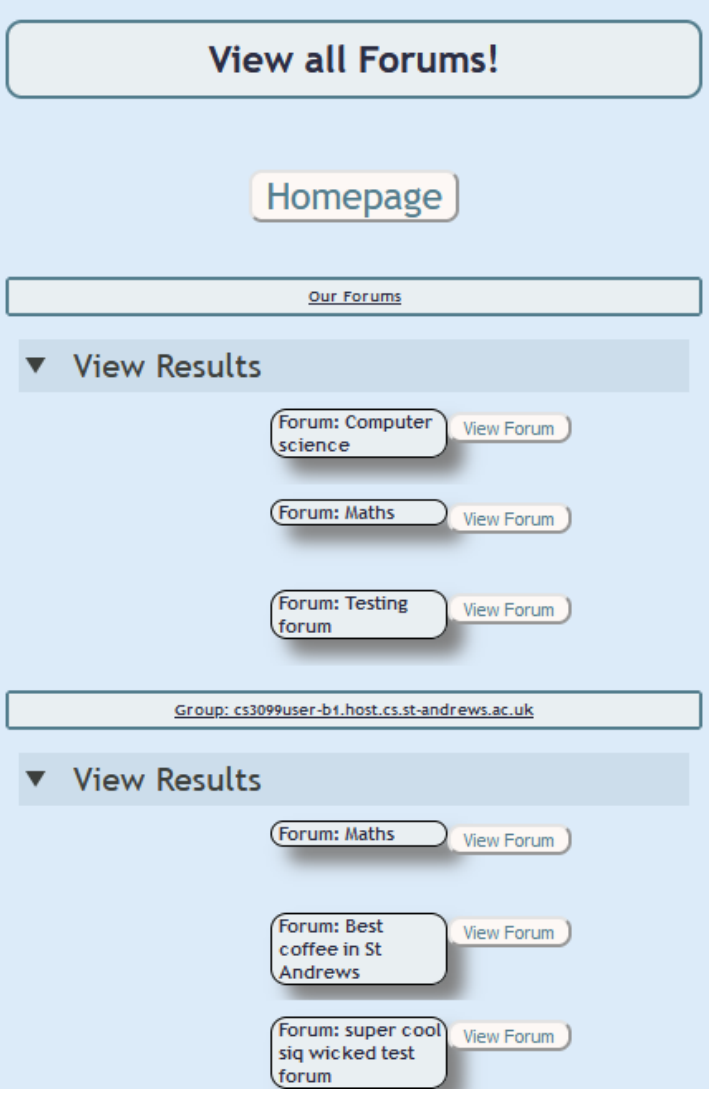


Figure 3.3 & 3.4 - Screenshots showing examples of our site being responsive to smaller screens like smartphones

3.4 Design and Implementation Decisions - [redacted]

The structure of our final database design can be seen in detail in Figure 3.1. Originally we did not make use of foreign keys, mostly because our knowledge on databases and SQL was limited. As we went through the year we were able to update and develop the design of the database. This allowed us to utilise more complex queries and access data from different tables within a single query, compared to nested queries, which often required callback functions.

To keep track of who was logged in to the site we decided to use cookies to keep track of the user's matriculation number. These cookies are assigned on logging in, before doing so a new user will have to create an account. We use validation methods to ensure they are filling in th0eir details correctly, this is done using regex patterns on matriculation number, email and date of birth. In particular, we have a verification step ensuring the new user is at least 17 years old, which is the minimum age of any university students.

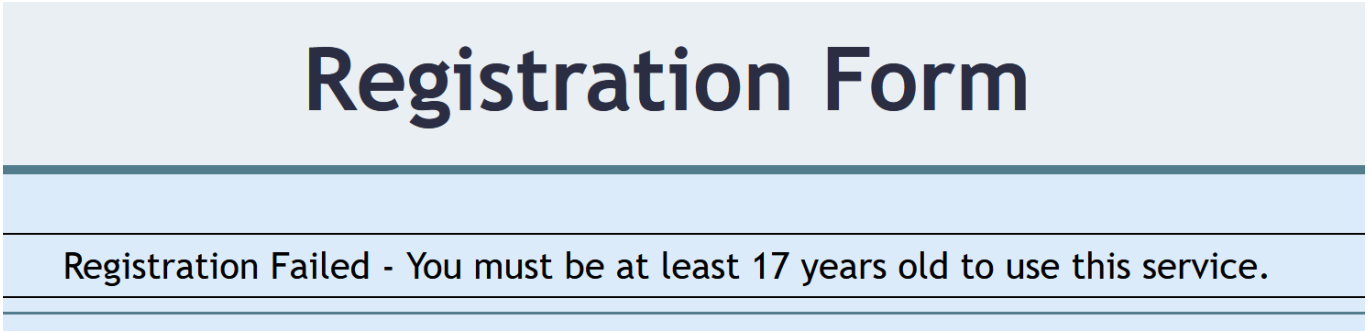
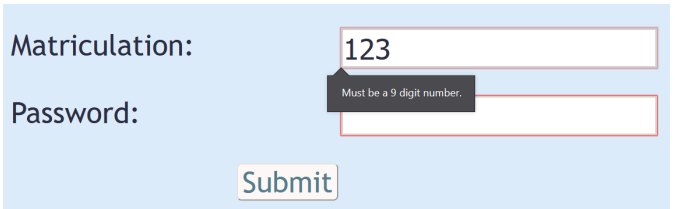


Figure 3.5 - Screenshot after a user younger than 17 has tried to sign up

As well as this we have checks on password and names to ensure they are a suitable length; displaying a useful message to the user as to what the expected input is if it doesn't match. This has also been implemented in posts and comments to ensure the user does not attempt to post an empty post or comment or one that is too long.



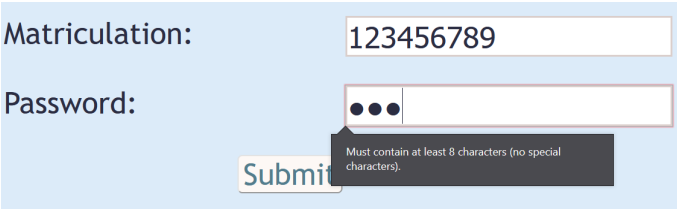


Figure 3.6 & 3.7 - Screenshots to show examples of error messages, on the login page

We have made an effort to ensure all our paths are clear and unique for each forum, subforum, post, comment and user. Our local routes all start "https://cs3099user-b4.host.cs.st-andrews.ac.uk/path" and then follow the structure forumName/subforumName/posts/:postID, to have the names of the forum and subforum in the path we had to ensure the subforums within a forum were uniquely named. We felt this was logically justified from a user's point of view and also enabled the path to be more readable than it would have been, had we opted to use forumID and subforumID instead.

3.5 Unusual design features - [redacted]

With a lot of our pages being personalised and constantly updated, we needed to generate these pages dynamically. To do this we built up our HTML in a string, adding results of database queries from the server and then serving it. We were able to refactor out standard features that we wanted to use across multiple pages, such as navigation buttons and page headers in a htmlGenerator.js file, to improve the consistency of the HTML design across the website. We then send this string of HTML as our response to the front-end.

Another unusual design choice that the supergroup decided to implement was digital signatures to access and send data in external requests. For this, a signature standard was agreed upon in terms of

the hashing function, algorithm and salt length. This was tricky to implement as none of us had experience in implementing such a security feature but we were able to use the Node Crypto module to get it working.

Our database design started unusually as we were initially unsure of good database design practices. At first, we decided to have each forum as its own table to store its associated data. However, we realised this was unnecessary and bad practice, and replaced this with a single ‘forums’ table to store all forums. We also didn’t take advantage of the relational nature of MySQL databases as we didn’t incorporate foreign keys until later in the development. Once we had incorporated the foreign keys we were able to perform more complex querying and included constraints such as ‘ON DELETE CASCADE’ which ensured that when we deleted posts, their associated rows in comments and postLikes would be deleted automatically.

3.6 Special algorithms and data structures - [redacted]

A notable algorithmic design choice involved creating a recursive function to display nested child comments. For this, we needed to iterate through every parent comment on the page to find the child comments and then recursively go through each of those child comments and get their children. This was tricky due to the asynchrony of the MySQL query function which required careful use of callback functions. We used an online resource [8] to introduce the concept of callbacks and then implemented them ourselves. We felt having child comments was quite inefficient, but it was a required feature of the protocol.

We hadn’t had much experience with asynchrony before the project, but we gained much more confidence in dealing with these functions as the development went on; making use of online resources to help our understanding [6]. We were able to reduce our reliance on callbacks for database queries once we incorporated foreign keys. This allowed us to do more complex querying – rather than nesting one query at a time – so that we could access more data in one query and simplify our code. To deal with asynchrony with external HTTP API requests, we learnt how to use promises, which had the main benefit of being more readable than our nested callbacks.

For the forum search implementation, multidimensional arrays were used to store search matches for each group. Arrays were a natural choice to store this data with them being dynamically sized and allowing various data types to be stored. We were able to store string forum names along with their integer IDs. We were able to push matches onto the array, iterate easily over it, and check if they already contained an identical match to avoid displaying duplicate results.

3.7 Local Features - [redacted]

Below is a summary of what local features we have implemented, which we expand on and discuss our reasoning for implementing them.

- Sign Up
- Log In/Out
- Profile Page - Post History, Bio, Subscriptions, Privacy Settings
- Home page - Post Feed, View Subscriptions, Edit Bio
- Create Forums, Subforums, Posts and Comments
- Posts can have Links, Images and Embedded Videos
- View Forums, Subforums and External Forums
- Search for Forums - both Local and External
- Live Chat

- Subscribe and Unsubscribe to Forums
- View Posts and Comments with a button to view the Author's Profile Page
- Like and Unlike Posts and Comments
- The Author can Edit and Delete Posts and Comments
- Nested Comments have the same functionality as Comments
- Admin Roles

Some of our basic local features include the ability to log in and out. Without this feature, there would be no way of distinguishing between different users on the site. We decided to use matriculation numbers as the unique identifier for each user. This decision was made as the idea behind the project was to create a platform for university students to discuss and share thoughts, resources and knowledge. This made matriculation numbers the obvious choice for identifying students as all students have one, and they are unique within a university. Once logged in the user can create and view forums, subforums, posts and comments.

On top of the basic features we also have many additional local features. Locally we have allowed Youtube videos and image embedding in posts, so they can share an even wider range of content. We have added as much personalisation as possible to create a more bespoke feel for each user.

The user has a personalised home page containing their feed. We allow users to subscribe to forums. The personalised home page then displays posts from the forums they've subscribed to, similar to how on Reddit the front page contains content from subreddits you're subscribed to. Here the user can also create and update their bio. Nobody else can see a user's home page.

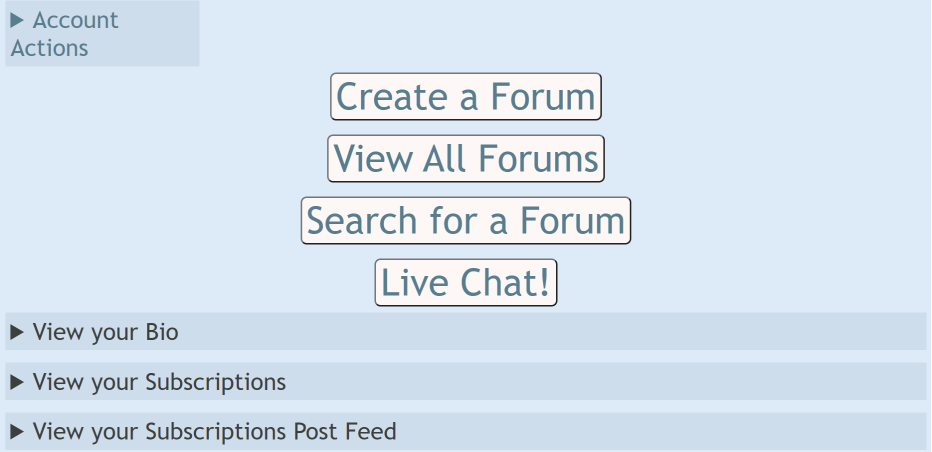


Figure 3.8 - Screenshot to show what a User's Homepage will look like

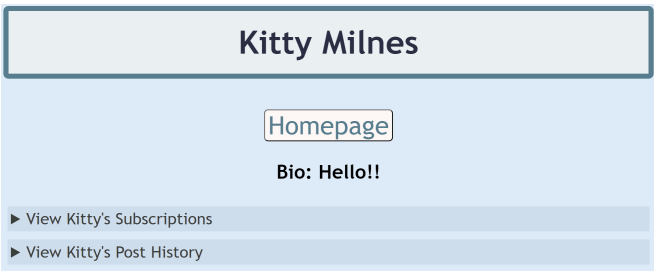


Figure 3.9 - Screenshot to show what a Profile Page will look like when Public

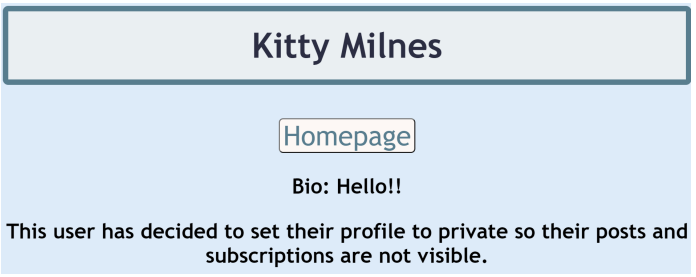


Figure 3.10 - Screenshot to show what a Profile page will look like when Private

In addition to the home page, we have also implemented a profile page showing the post history and bio of a user. This is intended as a public page allowing other users to see your post history, similar to clicking on someone's Facebook profile. Users can set their profile to private mode to prevent other users from seeing their post history.

We have allowed the liking of posts and comments and have enabled authors of posts and comments to be able to delete or edit these. We also have nested comments so that different comment chains can be created under one post. Additionally, we decided to give the creator of a subforum the ability to act as admin, this gives them the power to delete all posts and comments within that subforum. We felt this was an important feature to add a method of self-moderation within the subforums. Originally we also enabled the "admin" to edit any comments within their subforum however we worried this may allow them to abuse their power on the site and therefore removed this feature; now admins can only delete posts.



Figure 3.11 - Screenshot to show how Nested Comments appear. Note: Kitty can only edit and delete the comments that she has authored.

Finally, we believe we have some distinctive features that make our site stand out. We have an option to search for different forums, bringing up any results not only stored locally but on forums hosted by other groups. This means if another group has what you are looking for, you can quickly and easily look at their forum. Additionally, our search works with incomplete and multiple words so even if the user is not quite sure what they are looking for they will still be able to utilize this search function.

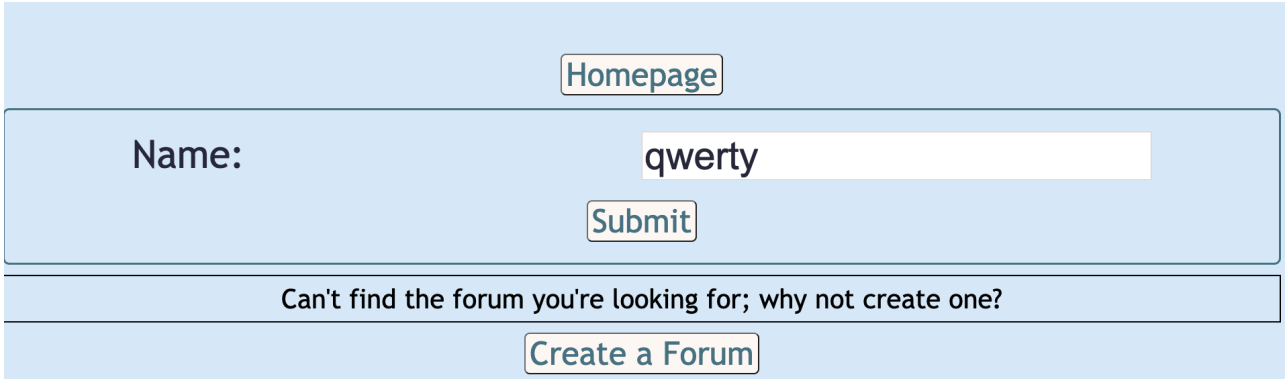


Figure 3.12 - Screenshot to show an example of would appear when there are no search results found.

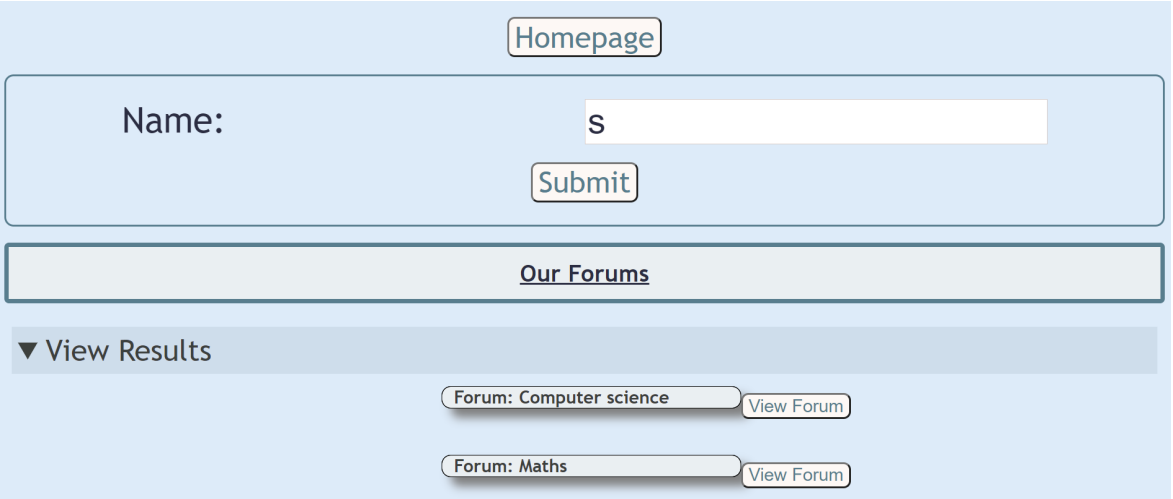


Figure 3.13 - Screenshot to show an example of what would appear when “s” is searched

A live chat feature is available as well, enabling users to chat with others that are currently online. We used the help of a tutorial to get this feature started [3] however built upon it with our own CSS styling and additional functionality, incorporating it into our website. Our chat displays how many people are currently active, shows when users enter and leave the chat and display timestamps of when each message was sent and by whom. We also assign each user a random colour when they join the chat to further help distinguish everyone on the chat.

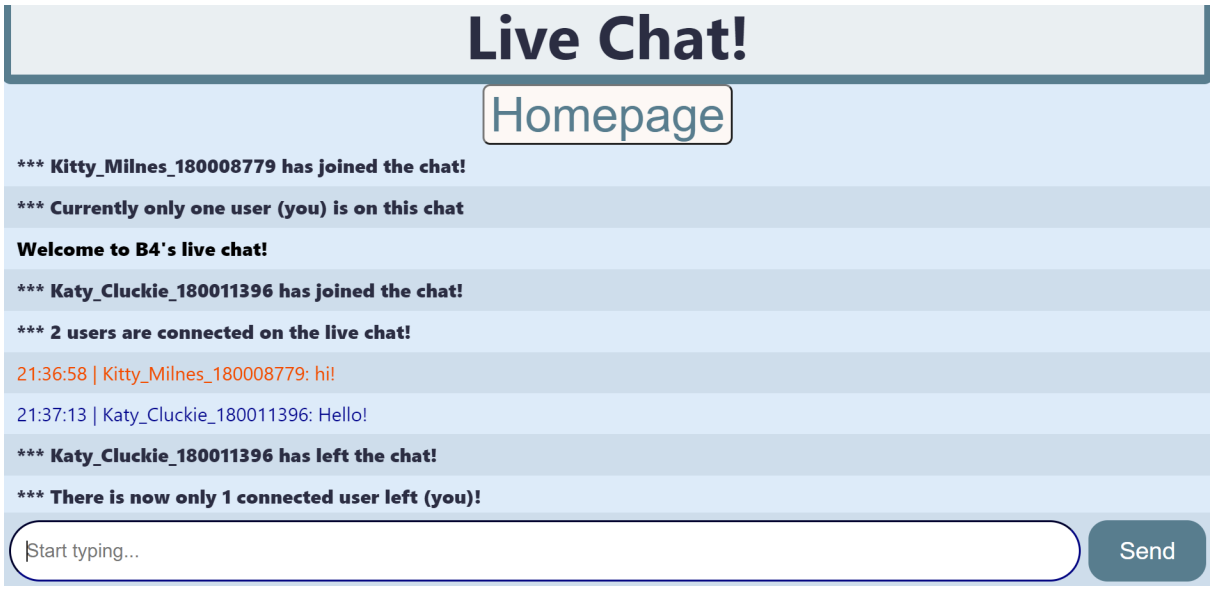


Figure 3.14 - Screenshot to show our Live Chat feature

3.8 Intergroup Features - [redacted]

Below is a list of implemented intergroup features as described by the supergroup protocol.

Intergroup Features

- Search for External Forums
- View all External Forums, Subforums/Posts/Comments
- Like Posts, Comments and Nested Comments by External Users
- An author can Edit and Delete Posts, Comments and Nested Comments
- Create Posts and Comments on External Subforums and Posts

- Receive Posts, Comments, Nested Comments made by other groups for our implementation
- Process getting Edit and Delete requests and Likes from other groups for posts and comments
- Digital Signatures and Verification

A large part of our implementation covered intergroup features, that is, features allowing our website and backend to interact with other groups' implementations, accessing their forums and other data, as well as allowing other groups to access ours. The purpose of this is to create a seamless experience, where, as far as possible, someone on our website can interact with a different group's forums, the same way that someone on that group's front end website would be able to. Notable limitations to this include the creation of forums and subforums, which aren't included in the protocol.

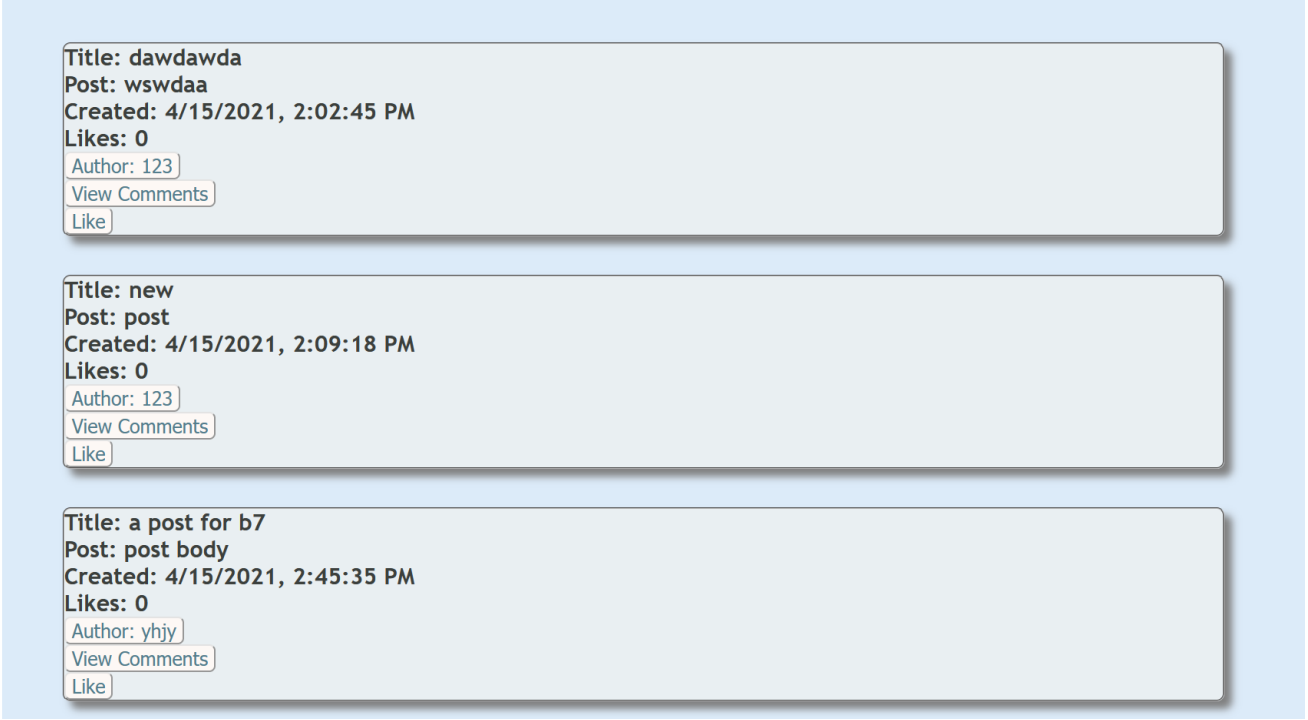


Figure 3.15 shows that on an external subforum, the posts appear almost identical to how they'd appear if they were on one of our own forums.

The protocol (Section 10.1) requires that each group sends a digital signature with any request for sending or receiving data, as well as each group verifying that when we receive a request, that the digital signature matches the required format. This is to ensure that only those authorised can access the data. When we receive a request for data we have a robust process to ensure that the digital signature is correct and they can access our information. More detail on digital signatures can be found in section 4.4

We have our API endpoints (Section 10.1) set up allowing other groups to get various parts of our data. Other groups can access all our users, get a specific user and get the posts or comments of a specific user. They can access individual forums and subforums, as well as all the subforums belonging to a forum. They can get a post as well as all of the posts in a subforum. Similarly, they can get the comments of a post, or an individual comment, with the addition of getting child comments.

We also have post requests set up, allowing other groups to create posts and comments (both parent and child) on our database's forums. A post from an external user appears the same as one from a local user would, with the exception that after displaying their username we mention that they're an external user. We also do not display local post history for external users as we do for local users.

In addition to this, we have patch and delete requests, allowing users from other groups to externally edit and delete posts and comments they wrote on our forums. Deletion works the same way as it does locally, so a stub is retained such that child comments are preserved. There is validation checking the user has the same id as the author of the post/comment.

Our protocol also allows for external voting requests on posts and comments (although we've only implemented upvotes). One thing to note about this is that having votes requires keeping track of users who've already liked a post, so we had to store references to external users in our database, to keep track of this.

In addition to letting other groups make requests on our implementation, we also have code in place to access and make requests on other groups' content. These requests can be made from our front end, and are presented very similarly to how we present our local content.

When searching for a forum, our code iterates through the groups, getting all their forums, and then searching the results to produce the matches for that search. So searching produces results for each group that has matching results, appearing with our local search results, but separated by group. A forum can then be opened to view its subforums.

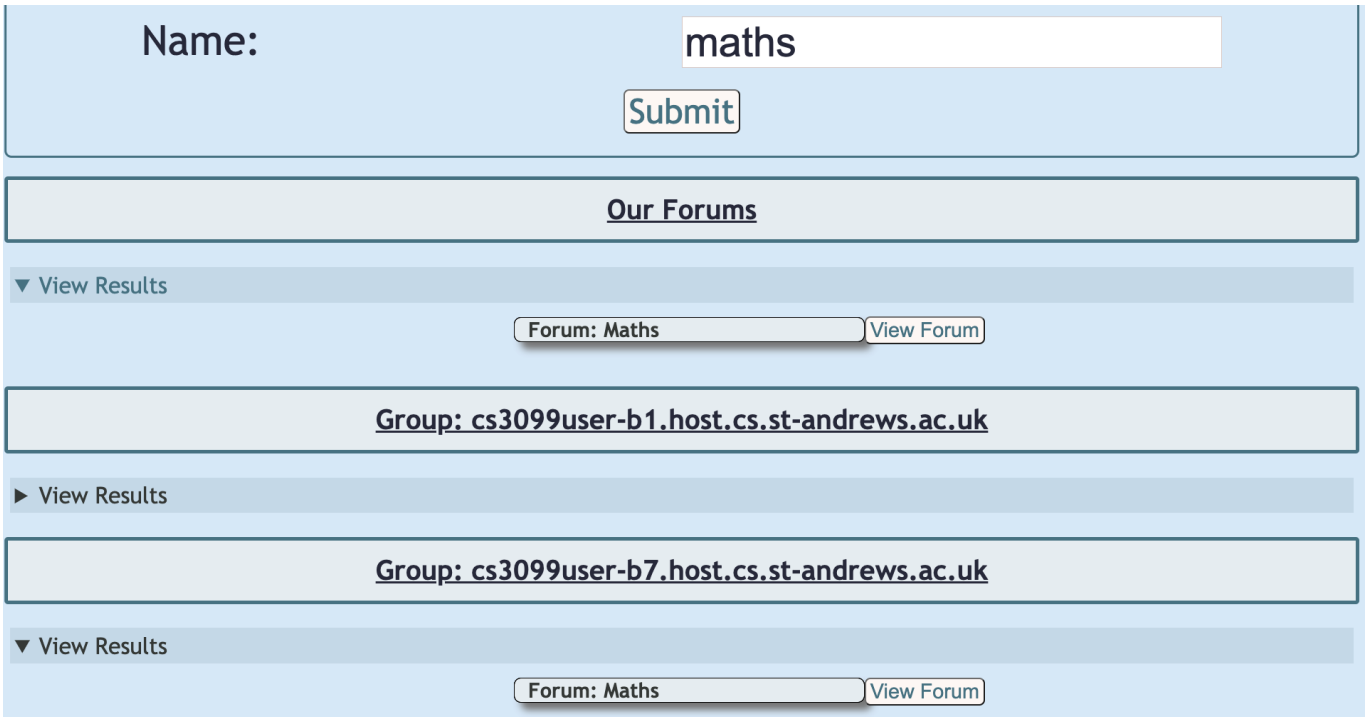


Figure 3.16 - Screenshot to show searching on all forums and getting results from external groups

Aside from our search feature, we also have a "View All Forums" button, which loads the list of all of the forums belonging to other groups, which can be expanded and opened.

From this point, the interaction with a remote forum is the same as a local one, provided the other group has set up their endpoints correctly. We can view and open subforums, look through posts, like posts, view their comments, and then like, write and reply to comments. As discussed above one notable exception is that we're not able to create new forums and subforums on other groups' implementations.

4 Intergroup Collaboration - 

4.1 How the Meetings Worked and how Decisions were Made

As members of supergroup B, we had to design and implement a protocol in collaboration with other groups in the supergroup. Clearly, since different groups were designing this together, this required supergroup meetings, to discuss potential changes as well as any new ideas. Meetings were held via a shared Microsoft Teams channel.

On top of this, some consensus was required between the groups to approve the changes. The consensus was reached by a simple majority, with each group getting one vote. In hindsight, this probably wasn't a good idea, as enough groups were happy to just vote for anything, that it essentially became a rubber-stamping process.

4.2 Protocol

The purpose of the protocol was to allow the different groups to communicate via a common interface. This required the setting up of common API endpoints allowing other groups to access our data in a JSON format. This means different groups will know where to access relevant data required for showing information from another group's database.

The protocol returns data through the use of JSON objects. This makes it quite accessible between the different groups that are using different technologies, as JSON objects can be parsed by many different languages. As a semi-structured data format, it also makes the parsing of the objects easier, as relevant data can be accessed more easily. The protocol contains error codes that are sent as a response to erroneous requests.

One of the main decisions of this protocol is to have forums and subforums. The idea of this is that forums will cover broad topics, whereas subforums are for more specific topic areas. Posts are then contained in subforums and have associated comments. This is best described as a series of one-to-many relationships from forums to subforums, subforums to posts and posts to comments.

Another feature of the protocol is child comments, allowing comments to be replied to. This is powerful as it allows threads of comments to be made. One disadvantage of this approach is that it is quite computationally complex, as, at each comment level, the comments table has to be queried again to find its children.

The protocol contains editing and deleting requests for posts and comments. There is a verification process involved here, as only the author should have permission to take these actions. It is up to the individual groups how to handle deletion of children - whether to cascade the deletion downwards through the comment chain or to leave a stub in, preserving the child comments. We felt it was more appropriate to leave stubs, as it means child comments don't get deleted.

Another feature of the protocol was that it contained different links linking to different parts of the database. For example, a comment would contain links to its children, its post, its subforum, its forum, and its author. These could in theory be used to obtain extra information about a post's forum, for example, although they didn't strictly add any functionality as everything can still be obtained through requests. While we included these links, we personally didn't make use of these when making external requests for other groups' data and felt that the requirement for these links was cumbersome and unnecessary.

The protocol also contained optional features for profile pictures, post types and downvotes. These features were strictly optional so not required, and we did not implement these as they were added to the protocol so late that they would have derailed our plans for finishing the project.

4.3 Issues with the Protocol

For us, the most glaring issue with the protocol was the fact that it was so complicated, which contrasted with our initial vision for what the protocol should contain. In our view, the protocol should have been a simple mechanism allowing groups to share data using the features described in the MVP. The final protocol felt bloated and unnecessarily complex, enforcing that certain features, eg. digital signatures, child comments, must be implemented, which we felt should have been individual design decisions for each group.

Another issue was how late changes kept getting added to the protocol. We felt this was already an issue in late February, and mentioned it in the third deliverable. Unfortunately, this persisted, and major changes kept getting added to the protocol right up to the end of March. We felt that having less than two weeks of term time to implement and test completely new features, and then having to write the report was not feasible in the time we had left.

The above two issues resulted in us having to scale back the scope of our project. We were forced to remove features such as private messaging from the backlog to account for the new protocol sections that were pushed on us. Still, compared to other groups we seemed to have implemented more functionality, with some groups seeming to struggle to implement any of the protocol.

Another problem with the protocol being so complicated is that it meant very few groups had it implemented fully and correctly. This meant we had a lot of difficulties when testing our protocol features with other groups, as we were often one of the first to add a given feature.

A further issue was the spreadsheet that groups were supposed to update to state which parts of the protocol they'd implemented. Although we kept our section up to date, we found that many groups didn't maintain their sections, or claimed to have completed sections that they hadn't. This made our testing more difficult as we had to guess which groups had implemented a feature.

One problem with the protocol is that its design slowed down our server. Since it required child comments such that any comment could be replied to, the operation of loading the comments became very computationally expensive, and there was little we could do about this.

As discussed some optional features were added to the protocol very late, namely voting, post types, and profile pictures. We implemented voting as we felt it was a good idea to have votes between groups, although we did think it was poorly thought out in the protocol as there is no requirement for userIDs to be unique between groups. As explained we didn't implement post types and profile pictures, although we did implement a local version of post types, which embeds media when it detects youtube or Imgur URLs as the contents of a post.

4.4 Digital Signatures

When sending a request for data to another group, we have to construct a digital signature to pass the verification. The first part of the signature is the request string. This includes the target of our request, the current date and the id of the user making the request. We also have a signature input string that has the request string, alongside the URL to access our public key and the algorithm that we as a supergroup

decided to encode the data with. We sign these together, using our private key, padding and salt length, in base 64. Rehashing this signature with our public key, which is publicly available without the need for a signature, verifies that we are indeed the source of the request. To help our understanding with digital signatures referred to this paper [14].

The next step is to send the correct headers so that other groups can parse and verify our data alongside our digital signature. We would send the signature input, which is the format of the signature, alongside the actual signature, current date, user ID and username for certain requests. With the headers, we send the required data for an HTTP request including the hostname of where we are requesting our data, the path of the data we want and the method. This ensures that the other groups can verify that our path and method match with those sent in the encoded request string and signature. The security benefits of a digital signature are discussed in more detail in section 6.4.

When groups send requests to get data or send data to us, we have a robust verification process to ensure that their digital signature is correct. This includes trying to parse all the required JSON objects from the protocol so that if they did not include one, or have one by a different name, this would be an invalid signature and we would return a 403 error. From this, we split the signature into the different sections to verify that these are expected. If for any reason, the splitting of strings by certain characters were to fail, it means that the group has not followed the correct format specified in the protocol and therefore we cannot pass the validation.

We validate the public key that the group sent in the signature, alongside the public key we fetched from their public source for it. If these 2 match, alongside the request-target strings we can be safe to fulfil their request, and if they don't the signature is not valid. We store an array containing the addresses of every other group's backend and only match against those. This adds to security, as otherwise, anyone with a digital signature would be able to get data, making our data vulnerable, and allowing the possibility for unknown sources to repeatedly make complex API requests, potentially overloading the server.

Once validated, the specified request is processed. Successful requests return JSON objects containing the results of that request, whereas unsuccessful requests, eg. specifying a user who doesn't exist, return a 404 error indicating the request failed. A 403 code error is used when the requester's digitally signed request doesn't match their public key. When an API request is successful, the 200 code is returned along with the JSON object.

5 Discuss Changes - [redacted]

5.1 Major Changes Throughout our Implementation

One major change from previous submissions is our database structure. We decided to make our database structure more robust, by including foreign keys to link between different tables. This then allowed us to improve our querying to be more specific and more efficient as we were able to carry out queries with inner joins to retrieve the information we wanted, instead of having to do multiple queries to get information.

Another change to our project plan was the languages and tools we wanted to use for implementing this program. Initially, we had thought about using Java for the backend but as we started getting things set up, we realised Express.js was easier to use and link in with our front end code. We also changed from storing our database, locally on MySQL, to storing it on MariaDB so that we could host it on the school servers, this is a change from our second hand in, as at that point everything was hosted locally, as the website wasn't hosted on the internet.

One change was in terms of the wording. Initially, we had set out to use communities and subcommunities, but we decided that forum and subforum was better wording for this as a supergroup. This wasn't a massive change, it just changed our references to different cases.

In our initial plan and previous submissions, we hadn't thought of or considered child comments. This was a large change to our overall picture. We had to change how we reply to comments and get comments to display them as nested, alongside changing our database structure to allow for storing parent and child comments. Although this was a fairly big change, we feel that it makes our program more in line with other social media sites in use today and improves the overall usability.

We also had not thought about editing and deleting posts and comments in our plans or our MVP. We have now implemented this both locally and with the federation.

5.2 Protocol Changes

Throughout this practical, the protocol has changed substantially. We feel this is to be expected, as the supergroup comes to terms with what the task is. However, our protocol was still changing 2 or 3 weeks before the deadline, which then meant we had to edit our implementation of the protocol very late on in the project and this caused us to almost fall behind in improving our local implementation of the project. The protocol also seemed very complex in comparison to Supergroup A. The changes also relied solely on 2 or 3 groups doing what they wanted, and sometimes it would include features that we felt should not be in a protocol and should maybe be an extension to the protocol. Despite these difficulties, we managed to keep up and adapt our API calls to work alongside these. As a change to our MVP submission, our API calls have become more advanced and now feature patch requests for editing posts and comments, alongside delete requests for removing posts.

6 Ethics and Security - [redacted]

6.1 Encryption

To ensure that the data of any users are kept safe and secure, we decided to encrypt any data that would be deemed to be sensitive. When a user signs up, or logs in to our site, their password is encrypted using a Node library called Crypto, using 'aes-128-cbc' and utf8. This allowed us to encrypt and decrypt any piece of information that we wanted. We chose to use the Advanced Encryption Standard 'aes-128-cbc', alongside utf8 to encrypt the data. To carry out this encryption we followed a tutorial on w3schools [4], which highlighted how to use this Crypto library for this particular purpose. Once the password has been encrypted, we then decided to store it in the database in its encrypted form. We then decided to use the same method to encrypt the matriculation number cookie. This means, that if someone was to edit their cookie for the matric number, it would not pass our attempt to encrypt/decrypt it and it would alert us that someone had changed their matric number, to try to pretend to post as someone else.

6.2 Preventing Harassment and Abuse

As the creators of this website, we felt some responsibility to prevent harassment and abuse from occurring on our platform, as these are serious issues that plague many similar websites. This can be especially problematic in a university community where many people will know each other personally and exactly the sort of environment where harassment would occur if our website was anonymous and unmoderated.

An obvious way to solve this was to include the user's matriculation number in their username, meaning a user cannot remain anonymous. The reason for this is that we feel that in an anonymous website, some people could harass others and say things they might not otherwise say if their identity were known. While anonymity is often a good thing, it can be detrimental in this sort of environment where the users will know each other. We feel that taking away the aspect of anonymity helps to reduce online harassment, as users cannot hide behind a fake identity.

Another feature we added to prevent this was subforum moderation. When creating a subforum, the user creating it automatically becomes the owner, allowing them to delete posts and comments made on that subforum, meaning content that is deemed inappropriate can be removed.

6.3 Testing with Non-Sensitive Data

During our period of testing and general use of the program. We decided it would be best to use 'fake' data, not real passwords or email addresses etc, to protect our safety and those testing our site as we were improving the security.

6.4 Digital Signatures

In terms of our API calls to the other supergroups, we decided to use digital signatures, to prevent anyone from being able to access our forums, subforums etc. This is important as it means we only accept requests from our list of verified sources. Using this across the supergroup meant that all users on each implementation had their information protected from unauthorised use. This implementation is discussed in more detail in section 4.4.

6.5 Cookies

We use cookies to store the current user's matric number throughout our site as they are logged in, encrypted for security. We also use it to store the current forum subforums or posts that they are on, to make querying for different information more straight forward. We do not need to ask for consent to store cookies because the ones we store fall under an exemption for being necessary for the provision of service. [2]

6.6 Prepared Statements

Throughout our implementation, we carry out many queries on our database from user inputs and requests. This can leave us open to SQL injection attacks whereby users enter SQL commands in input fields [15]. This could damage our database structure if not properly prepared for. We decided to use prepared statements to protect us against any form of SQL injection attack. As well as using prepared statements, we also validated for keywords specific to our database, like the names of our tables, or keywords that could not be entered as values in fields.

6.7 Client-Side Security

The design of our implementation means that on the client-side, users can view and edit our HTML, but as we have prepared for this with server-side validation, if the users were to change any of it there would not be any compromise to the security and robustness of our program. One possible downside to our implementation is the security of our live chat, as there is a script that is visible to the client if they were to inspect the code. They could change the code as we do not perform as many checks on this. We decided that as it did not contain any sensitive information, that we would leave it visible on the

client-side, and if we had more time would work on ways of improving this. One possible issue with the security of our site is that we don't sanitise our data for protecting against cross-site scripting attacks. It is something we would like to improve on in the future.

7 Testing - [redacted]

7.1 Overview of our Testing Approach

In the first semester, we focused on familiarising ourselves with new technologies including Node.js and its libraries and implementing the MVP's standard requirements. During this process, we relied on manual testing to verify that functionality worked as expected. This primarily involved interacting with the website from the user's perspective where we tested usability features of the website as well as input validation. For the relatively small set of requirements of the MVP, and with our primary focus being to set up the base foundation of the program, we felt this was sufficient.

Following the submission of the MVP, we reflected on feedback from our supervisor to incorporate automated testing to make our code more robust. We understood that as our program became more complex, we required more accurate and reusable tests as well as better documentation for our testing. We decided to take on a Behaviour Driven Development (BDD) approach with the Mocha and Chai frameworks. These frameworks provided tools to submit data to the website and check the response against the expected behaviour as described by our user stories.

We immediately understood the benefits of automated testing when we retroactively tested already implemented features and found bugs and issues that we had not picked up with our manual testing and reflected on how setting up automated testing from the start would have resulted in a more robust MVP. A notable benefit was seen as we worked towards refactoring our code to make our code more reusable before moving towards implementing more complex functionality. With automated testing set up, we were able to refactor code with confidence, as we could run the automated tests after each change to check that everything was still working. From the second semester, we worked towards integrating automated testing with the development of new features.

7.2 Using Mocha and Chai

When researching automated testing, we came across the Mocha and Chai frameworks and found that they were easy to use and had a lot of documentation. To set up the frameworks with Node applications, we followed a helpful tutorial on Dev.to [5]. Our Mocha and Chai tests involved testing our GET and POST methods by passing in data and checking the response of the website.

We set up our tests to run on a test database so that the test data didn't clutter our main database. The test database is created at the start of the run and dropped after all tests have completed. We also grouped our tests by functionality, such as for signing up or making posts, and checked test cases for each feature. This ensured that the tests were well structured and with the output more readable. A breakdown of our tests can be seen in the appendices (Section 10.4).

We also incorporated test coverage with a framework called Istanbul. With this, we were able to see our statement, branch, function, and line coverage. This helped us identify more test cases to check as we ensured we covered as much of the written code for each function as possible. Our test coverage can be seen in the figure below.

File	% Stmts	% Branch	% Funcs	% Lines
All files	71.43	63.23	65.6	71.4
CommonFunctions.js	84.76	68.33	100	84.76
CreateViewSubforum.js	77.78	67.74	100	77.96
SignUpLogIn.js	88.46	86.36	100	88.46
commentFunctions.js	63.1	52.63	94.44	62.9
htmlGenerator.js	58.46	0	80	58.46
postFunctions.js	66.44	67.74	88.24	66.44
userPages.js	80.46	65.56	92	80.46
websiteV3.js	56.3	44.44	37.27	56.3

Figure 7.1. Test Coverage Output generated by Istanbul.

7.3 Testing the API

Intergroup communication was the most difficult aspect of the project to test. This mainly came down to two factors: the complexity of the protocol and other groups' progress towards their intergroup communication.

Initially, we had created an apiTest.http file to store manual tests for external API calls with a VS Rest Client extension. This allowed us to write test API GET and POST calls and check the responses. However, the protocol became more complex, requiring authorisation and verification with digital signatures to access external data. Testing with this added verification was tricky as we would need to get and process the digital signatures in the test file, and we were unsure about how we could go about doing this.

The other significant factor limiting a more structured testing approach was the fact that a lot of groups, even a week before the deadline, had not fully got their intergroup communication set up, meaning we had only a few implementations to test against. Our main problem was not being sure if an issue came from our end or the external group's end with this limited number of groups to test against. Debugging at this stage was time-consuming and, at times, uncertain as a result.

8 Evaluation

8.1 Evaluating against Our Aims and the Task set to us - [redacted]

The sections listed under the minimal viable product were as follows:

- Ability to register users
- Posting new articles
- Posting comments on articles
- A simple, web-based user interface
- An implementation of a protocol for propagating content between instances

We had already surpassed these requirements in some areas by the MVP submission in November, and feel that we now have built considerably on this to produce a fleshed-out, feature-rich website. We can access and share data with other groups, meeting the group protocol

By implementing various features such as live-chat and profile pages, we think we've created a website that is well designed and offers various usability features. Overall we're very satisfied with the amount of work we've managed to do, especially given how complicated and chaotic the supergroup was. Based on this, we think we've succeeded in achieving our aims for this project.

8.2 Evaluating in Comparison to Other Groups - [redacted]

We have achieved a good level of functionality compared to other groups within our supergroup. In terms of structure, the fact the protocol was so in-depth left little room for freedom, however, we were able to design our site as we wanted, adding our own features and styling, setting us apart from the other groups.

We are happy with our progress and the rate at which we managed to implement the protocol. We found ourselves ahead of many groups at this stage which also made it difficult to test our API requests as we had limited groups to do this against. On the flip side, it means our own request endpoints have been extensively tested against as we have been used as the standard of a working implementation by some groups. Generally, our development was ahead of a lot of groups in the final stretch of this project. We saw that some groups still did not have their local functionality working currently in line with protocol less than a week before submission.

8.3 Evaluation Overall - [redacted]

Overall, we feel that this has been a very successful project in terms of meeting all of our aims and the goals of the specification. Our website is very reliable and this can be seen by other groups using ours to help test their intergroup communication. We found it suited our group best to test on groups B1 and B5 predominantly, as at most points they were making similar progress to us. We feel we achieved an excellent standard of the website whilst dealing with all the intergroup and protocol issues. Given the fact that we sadly lost a member of our group halfway through the project, we're very happy with the amount we've managed to do, despite this relative disadvantage to other groups.

9 Conclusion

9.1 Long Term Improvements - [redacted]

Given more time we would have liked to continue adding additional features to our project. Some ideas we have include implementing the rest of the optional parts of the protocol, namely likes and markdown. We would also have liked to have some of the features we have implemented locally available for other groups data in our supergroup. For example, we would have liked to look into subscribing to another group's forum from our site, adding this to our user's subscriptions feed and displaying it on their post history within their profile page.

Locally we would have liked to build on our live chat feature. We would have aimed to add this to individual forums or subforums so that people can chat between just one chosen forum. Storing these messages within our database would also be a nice touch so that a user can read back messages sent prior to them joining the chat. We would also have liked to have a direct message feature so that individuals can message other users making it more like other social media sites we use day to day. Although these features would have been nice to implement we are very happy with our current progress and what we have achieved.

9.2 Summarise Project with Key Achievements and Key Drawbacks - [redacted]

Overall, we feel that we have made many significant achievements in this practical. Some of our more notable achievements that we are proudest of would range from our live chat feature, editing, liking and deleting posts, as well as being able to embed videos and images in posts. Alongside these, we also are happy with how our nested comments work, as well as being able to search for forums, locally and externally. The decision to make a profile page for local users which shows their subscriptions and post history is also something we are pleased with. In general, we feel that one of the biggest achievements is the federated aspect of this project. There were many difficulties that we had to overcome to achieve our end product. The ever-changing protocol and the protocol being very in-depth and possibly too in-depth for some groups were the main difficulties. All of this aside, we do feel that we achieved a lot with the protocol and intergroup communication and have an end product that is easy to use and carries out the correct tasks when a user requests it.

Thankfully, we didn't have too many drawbacks throughout this project. The main issue was the intergroup communication and protocol. As mentioned in previous sections, the protocol was very complex and changed a lot, even in the lead up to the final submission. This meant that we had to spend more time on this, rather than focusing on our implementation. Another drawback was the lack of other groups that we could test against, as at the time of testing our API calls only 2 or 3 groups had a working implementation. We did, however, manage to get both our implementation and our intergroup implementation up to a standard that we were happy with. Another area that was an issue was getting our program to run on school servers, there was no real reason for this, apart from the fact that it took a while to get the pseudo-user and all the Nginx files sorted. This allowed us to have our project constantly running. Having never coded with promises and callbacks, we struggled initially to get our heads around the idea of asynchronous code and the best way to execute this.

The figure below summarises our progress on the project from start to submission.

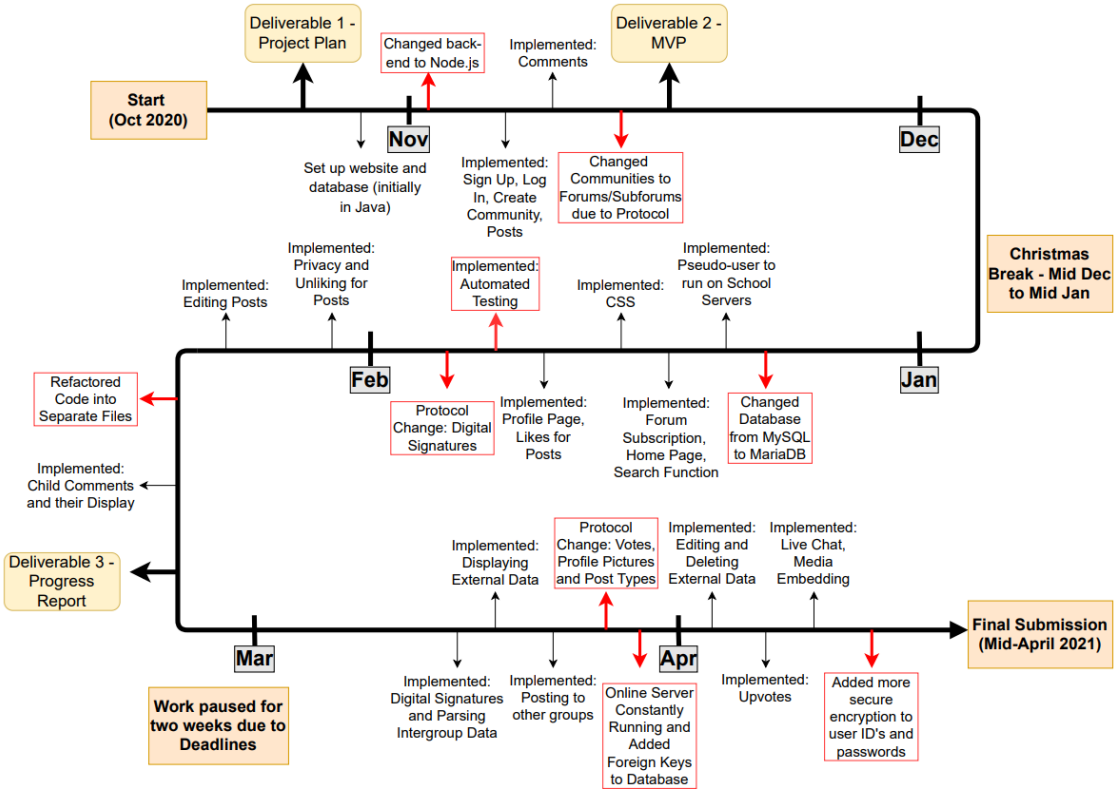


Figure 9.1 - A diagram to show the Timeline of our Project Development. The red arrows show notable changes to our project and protocol - (Created by [redacted] using ProjectOverallRoadmap.drawio - diagrams.net)

9.3 What we would do Differently Again - [redacted]

We agreed in our reflection that the main focus for improvement would be the group to intergroup communication. As we were working on intergroup communication, the requirements became more complex as the supergroup decided to implement digital signatures. This made it hard to make a start on testing our external API request, as we needed to pass the external group's verification before accessing data and testing our functions. While we were able to get this working before most groups, we noticed that some other groups had difficulty implementing it even a week before the deadline. What added to the difficulty was the fact that more requirements were added to the protocol such as voting on posts and modifying them, when most groups hadn't got their digital signatures or the required external requests working in the first place.

In the end, we thought that digital signatures acted more as a detriment to intergroup communication and should've only been implemented once every group had their basic external functionalities working. Communicating better with the supergroup, we could have made more of an effort to suggest this change to the protocol. We would have also pushed the supergroup towards agreeing to finalise the protocol earlier, as in our view it was unacceptable how late some of the major changes were.

We also would have liked to incorporate more end-user testing. The only feedback we received outside of our group was from individuals in the supergroup testing intergroup communication through our site and from members of our households. Opening testing to a wider audience would have given us unbiased feedback in regards to our design and ease of use of the website, giving us a better idea of what additional usability features to add.

9.4 What we Enjoyed and Learnt - [redacted]

Throughout this year-long group project, we have learnt many different skills that will help us in the real world. We feel that having to work together as a team has greatly improved our team skills, alongside being able to take individual responsibility and manage our time. As we all came from different degree paths, we had to balance our time and deadlines and as a group, we always understood when members of the team had upcoming deadlines. This practical has helped us all understand the level of organisation needed in a task like this; in terms of splitting up small tasks while having your eyes set on the final product. Our team as a whole had great communication throughout and we were very open and honest with each other, which we feel is very important when working on a large project like this.

However, we did learn that when you put 11 groups together in a supergroup, there will always be communication flaws and certain individuals feeling as though they are taking charge and making decisions for themselves. That being said, we also feel that we learnt a lot about how protocols work and the decisions that have to be made and agreed upon behind the scenes to get 11 groups on the same side so that our projects all work with each other seamlessly.

The scale of the project meant that we had to consider using many different languages and skills for coding, such as promises and callbacks and dealing with asynchronous code, as well as using databases and Javascript. It was the first time for most of our team using these languages and skills, and therefore provided a great learning experience for becoming more proficient in teaching ourselves these skills and languages which are newer to us.

The final learning curve for us was the fact the project was done fully remotely. The pandemic restricted our access to labs and meeting as a group and supergroup to discuss things. This has taught us how to adapt to learning online and communicating over online platforms such as Microsoft Teams.

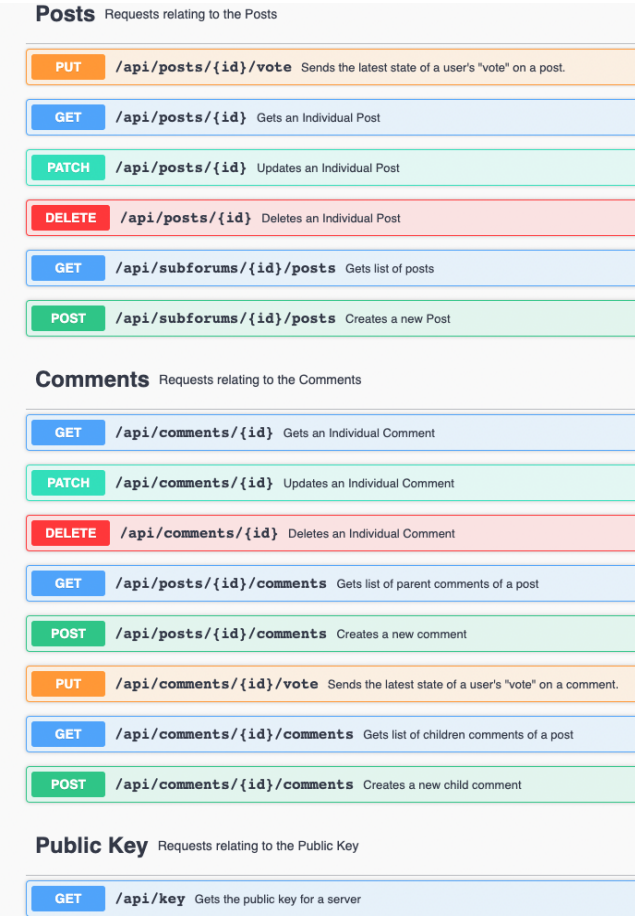
9.5 Acknowledgements - [redacted]

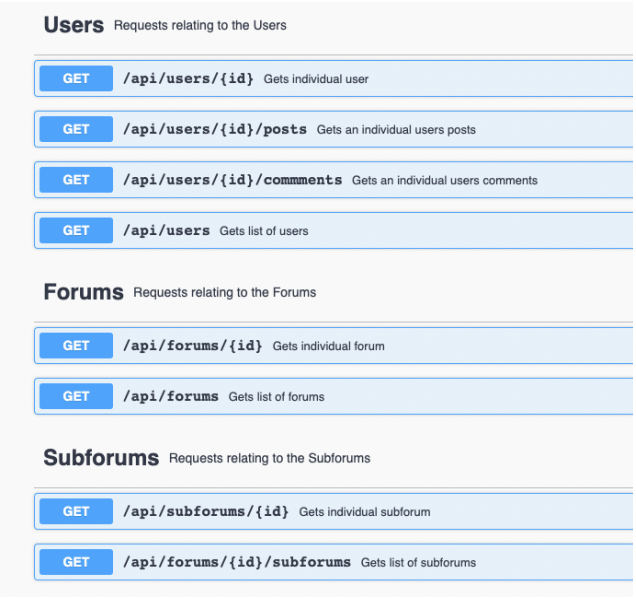
We would like to thank our supervisor, Dr Susmit Sarkar for his continued assistance throughout the year and project duration. We would also like to thank Node for enabling us quick and easy access to all the tools required to develop our project through npm, and also to GitLab for allowing us to use version control and store backups of our code. Lastly, we would like to acknowledge our use of the socket.io tutorial we adapted to create our live chat feature [3].

10 Appendices - [redacted]

10.1 Protocol

The protocol is publically available at https://github.com/FindlaySloan/3099_Protocol_Group_B
It has also been included in our submission as a pdf document.
Below are the API endpoints for the protocol.





10.2 User Stories from the First Submission

- As a user I will want to be able to create an account unique to myself so that I can tailor the platform to show only things I am interested in, I will therefore want to log in every time I use the site as I will then see things I am interested in.
- Leading from this I would need a way to demonstrate what I am interested in, I can do this by subscribing to communities that I find interesting allowing me to see and interact with the posts within these.
- To get the most out of these communities, I would want to post things related to the topic and that others may find useful. As these will only apply to particular communities I will want to be able to post in just one specific community.
- I may want to interact with others about a particular post, therefore I will want to be able to comment on a post and view others' comments.
- On top of this if I think there isn't a community that covers my interests I would like to be able to make my own that others can subscribe to.
- I want my private information to be secure and kept safely, this means it is important to me that my personal details and password is encrypted.

10.3 Scrum meeting summary document

This document can be found within the submission folder as a PDF.

10.4 Test Cases

Test Server
- ✓ Displays front page

Test Sign-up
- Valid Details
  - ✓ Valid details (unique matric)
  - ✓ Valid details user 2 (unique matric)
- Invalid Details
  - ✓ Empty email
  - ✓ Invalid email
  - ✓ Empty first name
  - ✓ Empty last name
  - ✓ Empty password
  - ✓ Invalid password
  - ✓ Empty dob
  - ✓ Empty matric
  - ✓ Invalid (Length) matric
  - ✓ Not unique matric

Test Log-in
- ✓ Empty matric
- ✓ Empty password
- ✓ Valid matric but invalid password
- ✓ Invalid matric but password in table
- ✓ Valid log in details

Creating Forums and Viewing Subscriptions

Valid Forums
- ✓ Create test forum 1
- ✓ Viewing created forum
- ✓ Create a test forum 2
- ✓ Punctuation in name

Invalid Forums
- ✓ Duplicate Forum, capitalisation
- ✓ Already created Forum
- ✓ Empty Forum name
- ✓ Key word name

Subscriptions
- ✓ No Subscriptions on Home Page
- ✓ No Subscriptions on Profile Page
- ✓ Subscribe To Forum
- ✓ Subscription showing on Home Page
- ✓ Subscription showing on Profile Page
- ✓ Unsubscribe from Forum

Creating SubForums

Valid Subforums
- ✓ Create a valid subforum
- ✓ Viewing subforum
- ✓ Punctuation in name

Invalid Subforums
- ✓ Invalid forum while searching subforums
- ✓ Same words as created Subforum, but capitalised
- ✓ Already created SubForum
- ✓ Invalid Forum ID
- ✓ Empty SubForum name
- ✓ Key word name

HomePage
- ✓ Show no subscribers
- ✓ Show subscribers
- ✓ Update Bio

Search for Forums
- ✓ Forum that does not exist
- ✓ Empty Forum
- ✓ Random Cases of letters from a valid Forum
- ✓ Multiple words and results

Posting Tests

Invalid Posts
- ✓ post with empty title
- ✓ post with empty contents
- ✓ post with both empty title and contents

Valid Posts
- ✓ Valid post
- ✓ Duplicate post
- ✓ Valid post, by 2nd user

Posts appear in correct pages
- ✓ Post shows up on homepage
- ✓ Post shows up on profile page

Deleting Posts
- ✓ Admin trying to delete post
- ✓ Another post by 2nd user
- ✓ Delete own post

Embedding
- ✓ Link in post
- ✓ Youtube video
- ✓ Imgur

Liking Posts
- ✓ Liking Post
- ✓ UnLiking Post

Editing Posts
- ✓ Edit Post Page
- ✓ Editing post
- ✓ Editing post with empty contents

Commenting Tests

Invalid Comments
- ✓ Empty Comment

Valid Comments
- ✓ Submit Comment
- ✓ Valid comment, 2nd user
- ✓ Another valid comment by 2nd user
- ✓ Get to Reply to Comment Page
- ✓ Reply to comment
- ✓ Reply to reply of comment (nested comment)
- ✓ Link in comment

Edit Comments
- ✓ Get to Edit Comment Page
- ✓ Editing Comment with empty contents
- ✓ Editing Comment

Liking Comments
- ✓ Liking Comment
- ✓ Unliking Comment

Deleting Comments
- ✓ Test admin trying to delete comment
- ✓ Delete own comment

Profile Page
- ✓ Get to Profile Page
- ✓ Making own profile private (toggling privacy on)
- ✓ Nothing shows on Private Profile Page
- ✓ Get to own Private Profile Page
- ✓ Making own profile public (toggling privacy off)
- ✓ View other users profile page

Database Issues
- ✓ Too long posts (>1024 characters)

✓ Too long post title (>1024 characters)

Generating All Forums
✓ View All forums

Operations when matric cookie incorrect
✓ Create a forum
✓ Create a subforum
✓ Profile Page
✓ Homepage
✓ Posting
✓ Commenting

11 Bibliography - 

[1] Atlassian. https://www.atlassian.com/software/jira

[2] Iubenda. https://www.iubenda.com/en/help/5525-cookies-gdpr-requirements

[3] Socket.IO. https://socket.io/get-started/chat

[4] W3 Schools. https://www.w3schools.com/nodejs/ref_crypto.asp

[5] Dev.to. https://dev.to/easybuoy/testing-node-api-with-mocha-chai-248b

[6] Javascript Info. https://javascript.info/async-await

[7] W3 Schools. https://www.w3schools.com/css/css3_mediaqueries_ex.asp

[8] NodeJS. https://nodejs.org/en/knowledge/getting-started/control-flow/what-are-callbacks/

[9] Mr Michael Torpey.

https://studres.cs.st-andrews.ac.uk/CS3101/Lectures/L10_SQL_Joins_to_Constraints.pdf

[10] Dr Edwin Brady - Scrum.

https://studres.cs.st-andrews.ac.uk/CS3099/Lectures/Week2a-Scrum.pdf

[11] Dr Edwin Brady - Sprint.

https://studres.cs.st-andrews.ac.uk/CS3099/Lectures/Week2b-Scrum.pdf

[12] St Andrews CS Department. https://systems.wiki.cs.st-andrews.ac.uk/index.php/Pseudo_accounts

[13] Geeks for Geeks. https://www.geeksforgeeks.org/how-to-run-a-node-js-application-permanently/

[14] Backman et al. Signing HTTP Messages

https://datatracker.ietf.org/doc/draft-ietf-httpbis-message-signatures/?include_text=1

[15] W3 Schools. https://www.w3schools.com/sql/sql_injection.asp