Community Social Media Website

Author: Sean Hiseman

Supervisor: Dr Daniela Tsaneva

10/05/2024

BSc Computer Science

School of Computer Science and Informatics

Cardiff University

Abstract

This project aimed to build a new form of social media website, one that combines community and individual interaction as well as offering new ways to chat and post. Problems faced by users in the current social media landscape are researched and addressed in the features of this website. The final product is a prototype that can serve as a foundation for future work, allowing new and innovative features to be developed.

Acknowledgements

I thank Dr Daniela Tsaneva for her supportive and insightful supervision, providing me with valuable guidance that greatly helped me to thoroughly tackle the subject to a high quality. I am also grateful to all the users who provided me with insights and ideas by completing my surveys, as well as those who took part in product evaluations, providing essential feedback and ideas for improvement.

Table of Contents

Table of Figures

1 Introduction

Despite being such a large and thoroughly used part of most people's lives, few would claim that the current array of social media sites is perfect. Low-quality attention-grabbing content, data harvesting, algorithms focused on maximising advertising views and misinformation are just some of the problems we often come across. Although there are many sites in the current market, none bring together all the features that users want. Discord offers engaging communities, but only for discussions. Reddit does this too, but only with posts. This shows a market that lacks a comprehensive and meaningful solution to a variety of customer problems with social media, which is what the project will address.

1.1 Objectives

1.1.1 First objective

The primary objective is to build a functional website prototype that facilitates the creation of engaging discussions and content in both personal and group contexts. This includes registration and login, secure data handling, connecting with other users, posting content, group and profile customisation and search functionality. These basic features form a foundation upon which the second and third objectives could be built.

1.1.2 Second objective

The second aim is to develop the site with original features not yet present in other sites. This includes balancing group and profile functionality, as well as combining post and chat features. Finding ways to improve content serving is valuable to users too, since current sites focus on maximising advertisement revenue. New features do not need to be complicated. Instead, they should be an attempt at solving a problem from a new perspective.

1.1.3 Third objective

The third aim is to develop algorithms for aggregating and serving content, including for search results, channels and a feed with recommended posts. The focus of these algorithms is to return the most relevant, useful and high-quality results to the user. Many algorithms for content recommendation already exist, so simply coding them to fit with my site is enough to meet this objective. However, adding my own changes helps improve the experience for the user as well as further meet the second objective.

1.2 Scope

Given the time limits, it is not possible to build a complete social media site in every way. However, the final product is fully functional in all the features it implements. The website is also not hosted on the open internet or open publicly to user registration. Instead, the project consists of a prototype and a database hosted on Cardiff University servers, allowing remote access for testing and user demonstrations. Considering that the recommendation algorithm is an objective alongside the main website, rather than the entire focus of the project, there was no need for it to employ any overly complicated techniques like machine learning.

1.3 Intended audience

Young people, especially those disaffected by current social media options, are the primary focus for the site, although it could still be open for any age group to use. As a research demonstration, the tested users will be entirely students at Cardiff University. Despite this, I specifically ensured that the site did not contain any features that would make it too niche, instead preserving its potential to be used by a wide audience of users.

2 Background

2.1 Current social media landscape

2.1.1 Why don't some people use Facebook?

to better understand what the project should contain and focus on, attitudes towards current social media sites need to be researched first. As the world's largest social media website[1], Facebook is a natural place to begin. The study, titled “Why don’t some people use Facebook”[2] offers insight into the distrust that many users feel towards the platform, which included the feeling that their wellbeing as a user was not as important as business interests. This effect is emphasised by strong privacy concerns where users disliked the lack of control over how their data was used and distributed. The strong presence of family members on the platform was off-putting to some since it restricted what they were willing to share. Despite the large amounts of personal data collection, the quality of content was sometimes undesirable, resulting in an overall feeling that the platform lacked usefulness and that other alternatives were better. It's important to note its limited sample size of only 17 users, 14 of whom were female. Despite this, the study offers a useful insight into user opinions and sentiment.

2.1.2 Students’ Acceptance of Discord as an Alternative Online Learning Media

The target audience of young people, especially students, is important to understand specifically. Their use of Discord, with its communities of channels, is therefore particularly helpful to understand. A paper looking into its use among students during the COVID-19 pandemic[3] found that having different text and voice channels greatly aided the learning experience. Despite initially being designed for gamers, the platform has found a much wider audience, showing that useful features can broaden the appeal of a previously niche platform. The positive reception to combining multiple channels of different types in a group is especially important for my project, which seeks to expand upon this idea both with posts and apply it to individual users. However, Discord’s interface was noted as being quite complicated for beginners, a challenge that I must therefore be aware of too. The study consisted of 44 students from one university. Although this is a small and narrow audience, their views are likely somewhat representative of the wider university population.

2.1.3 Instagram as a Tool for Study Engagement and Community Building among Adolescents

Instagram is particularly popular among young people[4], so understanding its use in community building will help guide my project. Despite its focus on individual user profiles, there was a study that aimed to evaluate the creation of a community around a profile dedicated to increasing awareness of healthcare advice[5]. This study found that Instagram is already a promising tool for influencing and creating positive impressions of their content with users, although user engagement and retention could be improved. Since users often follow a large variety of accounts, posts by the study have been missed by being too mixed in with other posts. If users had the option of joining a group for healthcare advice, potentially compromising smaller and more specialised groups, the posts could have had a larger impact. The study used 599 participants which is a reasonable sample size that adds legitimacy to their findings.

2.2 Content algorithms

2.2.1 Netflix Recommendation System based on TF-IDF and Cosine Similarity Algorithms

To build a recommendation system for this project, it was essential to first research the implementation and effectiveness of recommendation algorithms for already existing platforms. Netflix must serve content to a diverse and global audience, so its dataset is ideal for testing the effectiveness of recommendation approaches. A research team created an approach that combines TF-IDF and Cosine Similarity algorithms and applied it to 7,787 Netflix data entries to test its effectiveness[6]. TF-IDF stands for Term Frequency-Inverse Document Frequency and first requires data to be tokenized into individual words, as well as having any punctuation removed. Term Frequency (TF) is the ratio of a term's occurrences in a document to the total number of words in that document. Inverse Document Frequency (IDF) measures the importance of a term relative to the entire set of terms, with a lower IDF indicating that the term appears in fewer documents. Each document can then be represented as a vector of TF-IDF scores. This allows the Cosine Similarity of documents to be calculated, which is done by finding the angle between vectors where more similar vectors will have a smaller angle between them. This was applied to the titles and descriptions of Netflix films and TV shows to find the similarity of different content, resulting in the 10 most similar items being recommended. Although this approach applies in a different context to my community social media site, the underlying principle of recommending media based on the similarity of its text content is essential when building my own system. Such algorithms are general-purpose, allowing them to be modified to include user upvotes and analysis of the text content of posts. They also do not need to be limited to just 10 posts, instead being called continuously as a user scrolls down a feed.

$$tf_i = \frac{n_i}{\sum_k n_k} \quad (1) \qquad idf_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|} \quad (2) \qquad \cos\theta = \frac{A.B}{\|A\| \, \|B\|}$$

[Figure 1: Calculating the TF, IDF and cosine similarity]

2.2.2 A Survey of Collaborative Filtering Techniques

The approach described in Section 2.2.1 is an example of content-based filtering, but there is also collaborative filtering. This means predicting a user's preferences based on those of other users on the assumption that similar likes in the past are an indication of future tastes. A challenge of this approach is data sparsity, since some users may only rate a few items. Showing the user random items to assess an initial preference is a possibility, but risks disengaging the user from the platform if they see too much irrelevant content. As the user and content base grows, the efficiency of collaborative filtering must scale too, which can be done by clustering similar users and items. Increased dataset size will also result in similar items having multiple different names, which can be overcome by building indexes of synonyms. One of the methods mentioned for implementing collaborative filtering is by using Pearson Correlation. This is the linear measure of correlation between two variables, ranging from –1 to +1. The paper[7] mentions that collaborative filtering can work especially well when combined with content-based filtering in a hybrid approach. Since one of the main aims of the project is to develop a high-quality recommendation algorithm, this approach needs further exploration

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Where,

r = Pearson Correlation Coefficient

x_i = x variable samples $\quad$ y_i = y variable sample

$\bar{x}$ = mean of values in x variable $\quad$ $\bar{y}$ =mean of values in y variable

[Figure 2: Finding the Pearson Correlation]

2.2.3 Automatic recommendation system based on hybrid filtering algorithm

Further understanding of Hybrid Filtering is needed to implement it effectively. A paper[8] describes the approach in more detail. Although focusing on web mining for recommendation, the algorithm's fundamental features are the same. User profiles similar to the active user are found and for each similar user, items are selected based on content vectors and user profiles. Hybrid Filtering was assessed for effectiveness using Precision, Accuracy, Recall and F-Measure when applied to datasets of users rating books. In all these metrics, Hybrid Filtering scored higher than Content-based and Collaborative when applied individually, which suggests that it is a useful approach to apply to help achieve the goal of building an algorithm to improve content quality, especially since it also uses Cosine Similarity and TF-IDF. It was also found to be quicker to adapt to new users than each of its constituent algorithms. However, the filtering is only applied to text content, so the algorithm would have to be modified for use in a multi-modal social media website [Section 10.3.3]. The rating system for books is also simpler than the voting I intend to implement, so the algorithm will have to be modified even further.

3 Methodology

3.1 User testing

3.1.1 Initial survey

To be informed about what my site should focus on, I needed to understand the opinions and ideas of users, which I did by conducting a short survey on Microsoft Teams. This consisted of 8 anonymous multiple-choice questions about current social media use, content preferences and privacy, as well as text boxes for user's biggest problems with currently existing sites and ideas for new features. I also asked about what potential uses users could have for an Artificial Intelligence 'co-pilot' since adding such a feature to assist users could be an area for future research. Users were found by being contacted by me directly and most were willing to participate upon being given a brief explanation of the research. To ensure ethical standards were followed, the survey began with a participant information sheet and consent checkboxes that had to be filled for the survey to be submitted.

3.1.2 User evaluation

Once the site was fully developed, I conducted in person user experience studies to evaluate my product, finding out which features they found important and well-implemented, what needed to be improved or fixed, and what could be added in future development. 5 users took part, each having already filled in the initial survey. Before starting, they were shown the participant information sheet and asked to confirm their consent to participating in the research. The study consisted of describing the features of the site to the user and performing a demonstration myself, before handing over to the user to observe and discuss the site while they used it, noting down any relevant comments. While they were using the site, I asked a series of open-ended questions and noted their responses in Microsoft Forms. Questions were about the site's appearance, its features, what the user believed should be added or removed and what they overall liked and disliked.

3.2 Product development

3.3.1 Development practice

I followed Agile development practices[9] where each week I set myself specific tasks to be implemented to completion. I scheduled to begin coding from the first week and continue to the final week to ensure adequate time could be dedicated to finishing, testing and fixing each of the large number of features. Although already familiar with the languages and frameworks to be used, there were inevitably pieces of information that I didn't know or new knowledge I had to learn. Whenever I was stuck with my code, I first turned to the online documentation for the frameworks I was using. If this wasn't helpful, I searched for my problem on Stack Overflow to see if anyone has had a similar issue before.

3.3.2 Limitations

Since the final product is a locally hosted prototype, no users from the open internet are intended to use it. This means that only a limited amount of content can be uploaded for purposes of testing, all either by me or provided to users for testing. The recommendation algorithm can therefore not be thoroughly tested for how many different users subjectively feel about the quality of the posts it serves. Instead, the scores that the algorithm outputs will be measured directly since the numerical values of each post can be changed arbitrarily to represent different levels of quality.

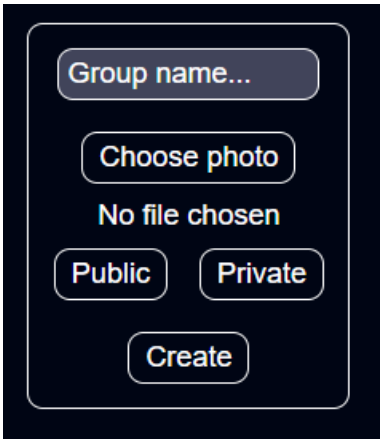
[Figure 3: Project development chart]

4 Specification and Design

4.1 Groups

4.1.1 Group design

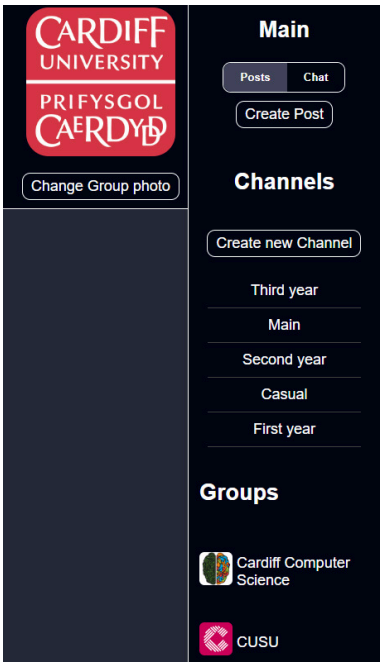
Groups are one of the two primary content locations that the site centres around, alongside profiles. Any user can create a group by using the dropdown menu in the left sidebar. They can choose a name and group photo, as well as set the group to private or public. Upon creating the group, they are taken to the main channel, including the header where they can write a description for the group. The contents and channels of private groups are not visible to non-members, instead displaying a join request button that can be accepted or rejected by moderators or administrators.



[Figure 4: Creating a group]

4.1.2 Nested groups

Smaller groups can join larger groups in a nested fashion, where sub-groups appear in the sidebar below the channels. This was implemented because groups focused on broader topics will likely have smaller topics that are still too extensive to be covered by a single channel. For example, a group for UK universities could have each university as a sub-group. Within an individual universities group, each school could be its own sub-group too. This was inspired by Discord's Student Hubs[10], but my feature is much more general purpose and allows for nesting at an unlimited depth, making interactions between communities much more direct. An admin can request for their group to join another group by clicking 'Add to group' and typing in the name of the group to join. Admins in the other group can see this request and accept or reject it in the same way as individual users.



[Figure 5: Sub-groups within a group]

4.2 Profiles

4.2.1 Profile functions

Since not all users may wish to post their content in a community setting, I implemented similar functionality to profiles too. Posts can be made in the same way to different channels, although only the user can post to their own profile. Like groups, profiles can be either private or public, with private profiles only visible to friends and not having followers. To everyone else, only the name, bio, profile picture and an option to send a friend request are visible. On all content that a user posts across the site, as well as alongside any messages they send, their profile picture and username are visible, redirecting to their profile when clicked on.

4.2.2 Account control

To ensure user control over their privacy, they can toggle their account between public and private at any time. They can also change their username, bio and profile picture, as well as delete any post, reply or message they make across the site. To further guarantee user control over their data, there's an option to delete their account which permanently removes all entries across all database tables associated with that user.

4.3 Channels

Posts and chats are the two content types that the site focuses on, and each can be made in dedicated channels. Post channels are ranked by an algorithm, whereas chat channels have messages that are shown chronologically, each with the profile picture of the user who sent the message. In groups, each channel can be either posts or messages or both, with an option on the sidebar to switch between the two. Profiles only display post channels, since each user has a section for direct messages with their friends. Channels exist because content posted to groups often has many different topics and themes related to the group's subject, so keeping track of all content of all different types can be challenging. Also, not all users may be interested in every type of post made to a group, so splitting posts into separate channels lets them more carefully curate the content that they consume.

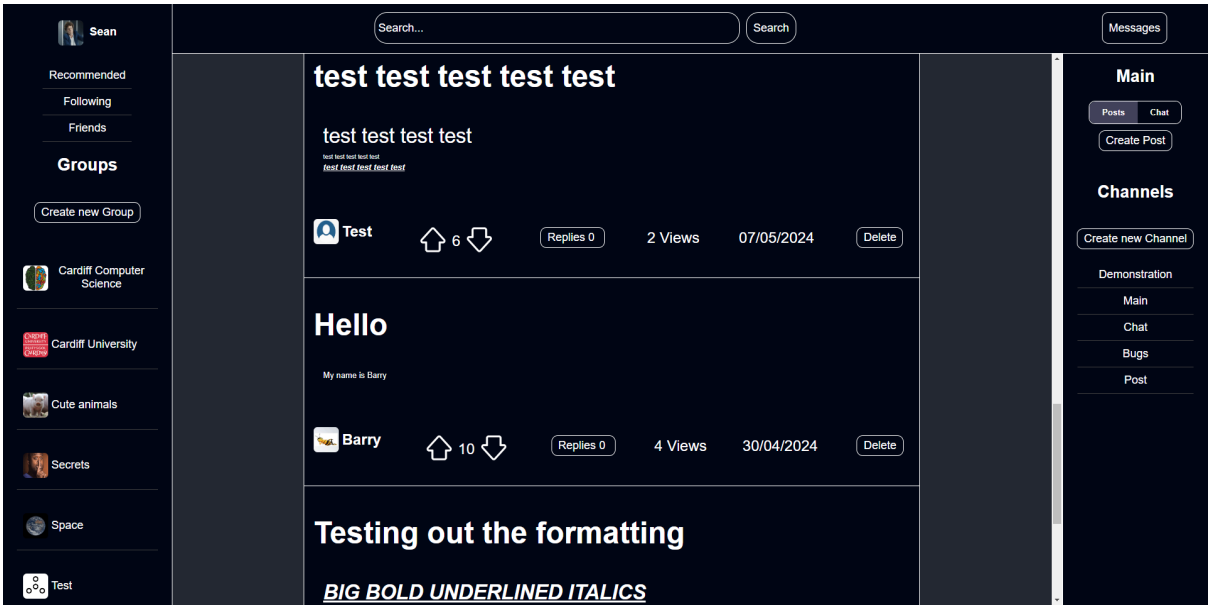
[Figure 6: Chat channel within a group]

4.3.1 Channel modes

Initially, the intention had been that a channel can either be for posts or messages, which is decided when creating the channel. However, it was realised that this could result in 2 different channels for the same topic, increasing clutter in the channel list. Therefore, the approach was changed to channels being for both posts and messages by default, with a toggle to switch between the two modes in the sidebar. Since not all channels need both modes, there's still the option to have only one or the other. I had also originally planned for profile channels to work in the same way. However, an individual's profile is a different environment to a group discussion, so there was little need for each channel to have a messages section where multiple users can participate. Also, the ability to chat with a user is already fulfilled by the messages page.

4.3.2 Main channels

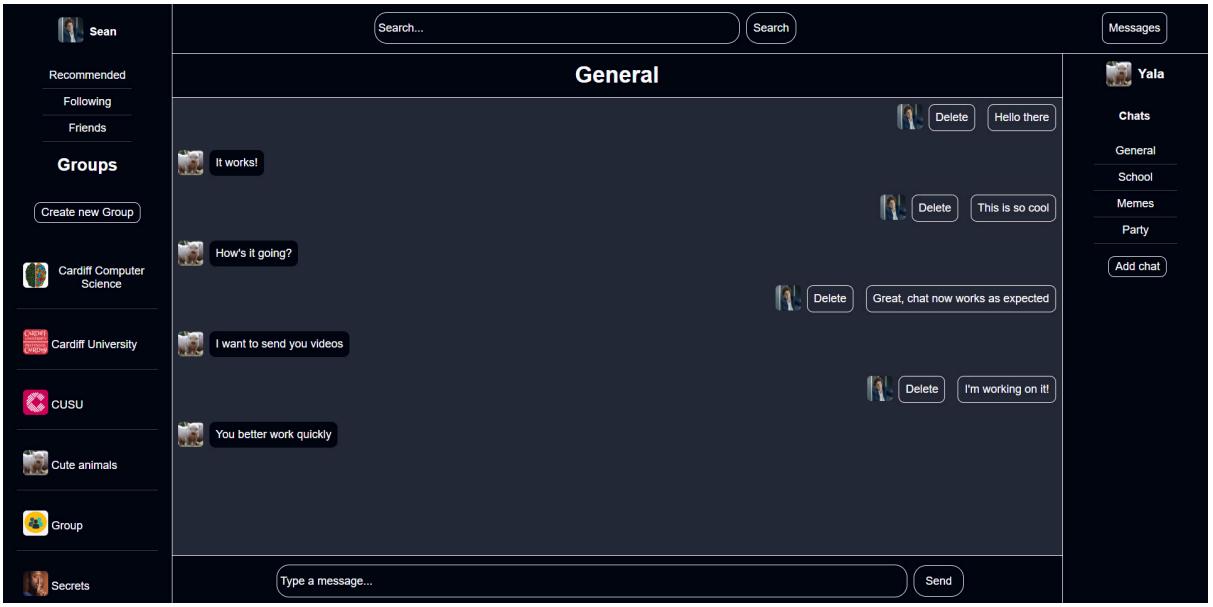
Each group and profile has a main channel that cannot be deleted. The purpose of this feed is to aggregate the content posted to all the different channels in one place, although posts can still be made directly to the main channel. This channel exists because a group or profile may have many different channels which would be time consuming to check individually, especially if a user is browsing many different groups, so viewing all of a groups content in one location is much quicker.



[Figure 7 Test posts in a group main channel]

4.4 Direct messaging

Due to the large number of social apps that many users now have, the phenomenon of ‘multi-communicating’ has emerged, where two users engage in multiple, separate conversations, especially in collaborative workplace environments[11]. Currently, users must keep track and manage these conversations across multiple different apps, which can be confusing and annoying, especially for students who may discuss several different modules, as well as social events or general conversation. This inspired me to simplify the process of multi-communicating within my site by allowing users to create multiple chat channels with the same friend, each with its own title. Users can only send direct messages to their friends, rather than any user of the site, as a privacy feature to prevent unwanted messages from unknown users.



[Figure 8: Direct messages page, with multiple chat channels]

4.5 Content ranking

To sort content by quality, there needs to be a way for users to vote on what they see, most sites do this with like/upvote buttons, with some including dislikes/downvotes to manage poor quality content. However, this results in a binary voting system, where the vote is either given or not, with no way to express how strongly the user likes or dislikes the content; it may be satisfactory, good or exceptional (or the opposite for negative sentiment). This is why I created the ability for each post to be upvoted or downvoted multiple times by a user, with a maximum net vote of either –10 or 10 to prevent excessive voting. Replies can be added too, which can be nested (i.e. replies can be made to replies) and are sorted by net upvotes.

4.5.1 Views

Raw numbers of upvotes and downvotes are not much use on their own. They need to be combined with viewing data to find ratios of upvotes and downvotes per view, since higher/lower quality content will be voted on more often. However, considering the content as being viewed just because it was scrolled past on a feed is a poor indicator of quality, since the user may ignore or be disengaged with the content. To solve this, I only considered a piece of content as being viewed if it is upvoted, downvoted, the replies are opened, or the uploader's profile is viewed. Once one of these criteria has been met, the view is incremented once. Doing any of these actions again will not increase the view count unless the page is reloaded. This approach is taken to avoid any unhelpful views, only counting those where the user has actively engaged in the content.

```
//Adds a view to the post
const incrementViews = async (postId) => {
    try {
        if (hasViewed === false) {
            await axios.post('/api/increment_views', { postId, isGroup });
            setHasViewed(true);
        }
    } catch (error) {
        console.error("Error incrementing views:", error);
    }
};
```

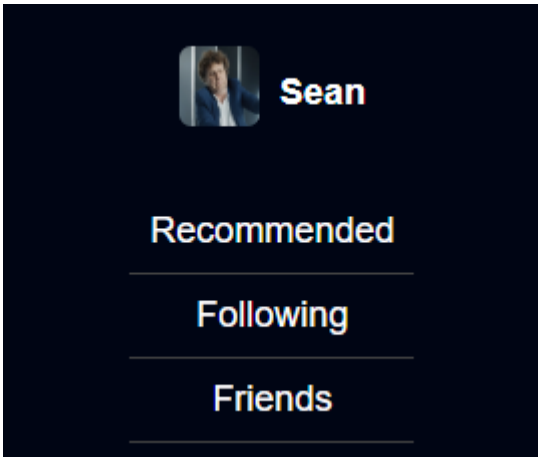
[Figure 9: Incrementing views by one]

4.6 Content serving

Since my content sorting algorithm is intended to deal with large numbers of posts from different users, profile posts do not have an algorithm applied to them, instead being shown chronologically to ensure no posts are hidden by the algorithm. The main profile feed still shows a time ordered aggregation of content posted to the channels.

4.6.1 Feeds

The site has 3 separate feeds: Recommended, Following and Friends, giving the user the option to separate which types of content they wish to view. Recommended consists entirely of new content, and following only has posts from the profiles followed and groups that the user is a part of. Posts from friends are from people the user knows personally and are likely to be different to posts from profiles that the user follows, which is why the friends' posts are in their own feed, separate from followers. This feed is also chronological, again to avoid any posts from being missed.



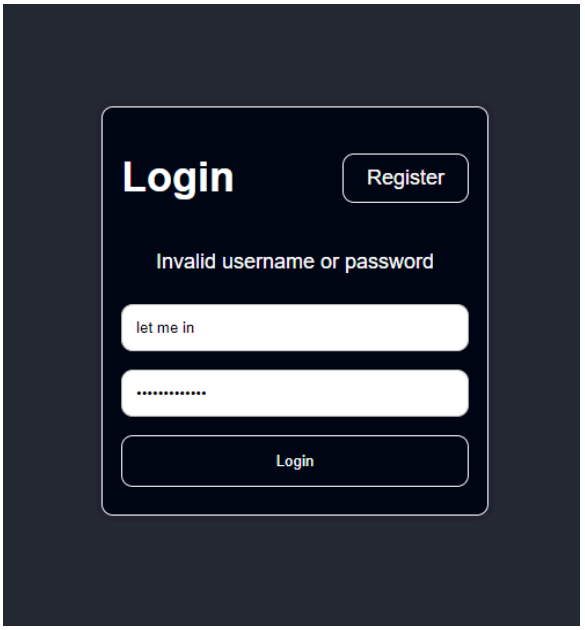
[Figure 10: 3 different post feeds]

5 Implementation

5.1 Site entrance

5.1.1 Login

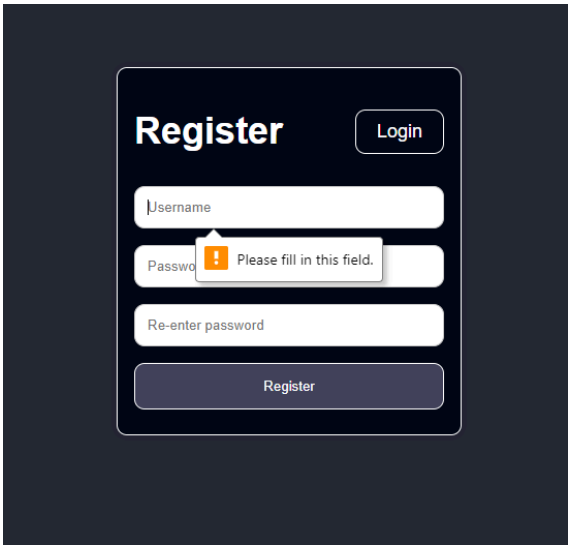
The default view of the site is the login page, which includes a link to the registration page. The login page is where a user is redirected to whenever they logout or try to load a page when not logged in. A username and password are required to login, with the password converted to a hash using the bcrypt JavaScript library, which is compared with the encrypted entry into the database. The final hash is independent of the characters and length of the password, which prevents anyone with database access from viewing or calculating the password. Password entry shows each character as a dot, preserving privacy when entering around others. A small notification appears if either box is not filled in and error text appears above if the username or password hashes do not match those in the database. Upon successful login, the user is redirected to the recommended feed.



[Figure 11: Login page with incorrect details]

5.1.2 Registration

To login, a user must first register a username and password. The username is checked against currently existing usernames to prevent duplication, and the password is re-entered (again with each character as a dot) to ensure no mistakes are made. The password is immediately hashed straight from the request body and inserted into the Users table, alongside the username, account creation time and a randomly generated 36-character id that is used as a primary key to identify the user throughout the site. An entry into the Profiles table is created next containing a unique profile_id, the user_id foreign key, a default profile photo and a blank profile. Upon completing registration, the user is redirected to the login page.



[Figure 12: Registration page with prompt]

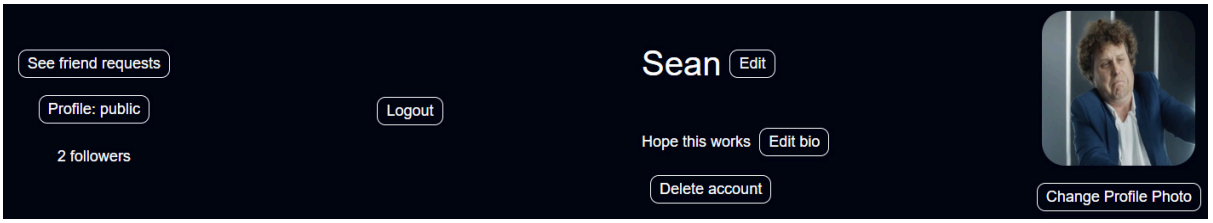
5.2 Page design

5.2.1 Base layout

Almost all pages in the site are rendered within the base layout. This consists of a header with a search bar and link to the messages section, as well as a left sidebar with links to the user's profile, their content feeds and all the groups they are a member of, as well as a form to create a new group. The content area of this base is where the specific pages are loaded, such as for profiles, groups and messages. Each of these pages consists of a right sidebar that lists the different channels of the page.

5.2.2 Headers

At the top of each page is a header containing information and options specific to the page, including name, bio/description, profile picture and follower/member counts, as well as buttons for friend/join requests. To avoid cluttering the user's view of the feed, this header disappears as the user scrolls down the channel's feed.



[Figure 13: Profile header of logged in user]

5.2.3 Channel feed

Within each specific page is a multipurpose channel feed. Viewing a page usually means being in a specific channel (Main by default), so this is the area where the posts or messages of that channel are displayed. However, if the button to create a post is clicked, this channel feed will instead display the form for creating posts. If an admin wishes to view the members of a group, or the join requests, then the channel feed shows these instead. This simplifies the viewing of a page by having one area where most of the dynamic content can be accessed.

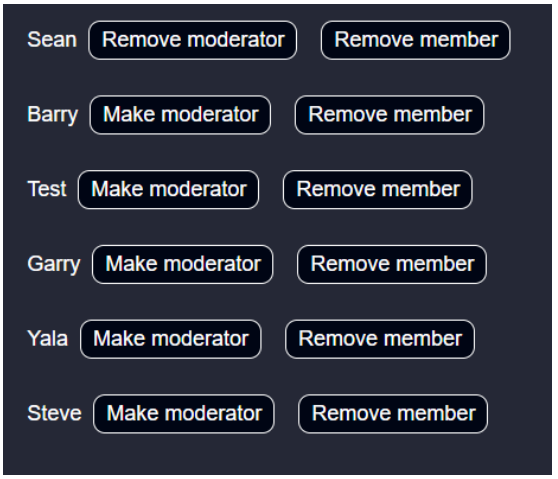
5.3 Permissions

5.3.1 Personal profile

Not all pages appear the same to all users. A different profile layout appears if the user is viewing their own profile, allowing them to access options for customisation and control. These consist of changing the username, bio and profile picture, viewing friend requests, logging out and deleting the account, as well as creating and removing channels and posts. Friend requests can also be managed, with the content area being used to list the requests, each with an accept/reject option.

5.3.2 Administrators and moderators

Administrators too have a different view of their groups, with the ability to view group members, appoint them moderator or remove them from the group. They can also add, delete and edit the group channels, as well as changing the group name, description and profile picture of the group. Moderators don't have this ability but can still remove posts and messages, as well as accept new members. To modify the page depending on the permissions, the id of the logged in user is checked against either the user id of the profile being viewed or the list of members for a group. The user who created the group is automatically an administrator, although they lose this position if they leave. For moderators the 'delete' button is only rendered if 'is_mod' is true.



[Figure 14: Admin view of group members]

5.3.3 Members and friends

A single button is used for managing the status of friendships, either displaying 'send friend request', 'cancel friend request' after it has been sent, or 'remove friend' once the request is accepted. To display the correct text, this 'manageFriendship' button checks the database tables for both friends and friend requests. The 'memberChange' button in groups works similarly, although it must also check if the group is public or private, since public groups do not need permission to join and can simply display either 'join' or 'leave'.

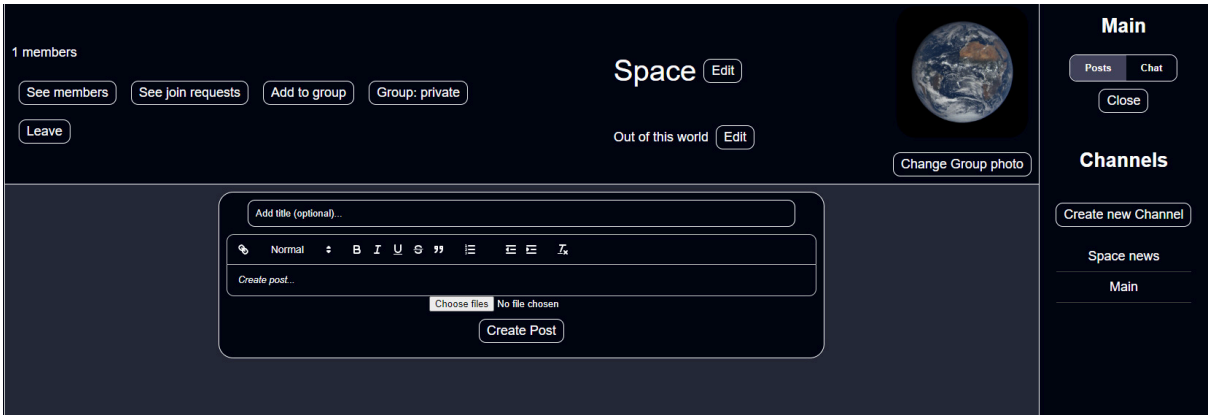
5.4 Posts

5.4.1 Creating a post

Since some current social media sites often only allow each specific post to be one content type (e.g. Instagram allows images and videos, but only one at a time), I wanted posts to my site to be more multimodal. Posts are made using a customisable form that allows for a variety of ways to format text, including size changes, italics, bold, underline and listing. Multiple images and videos can be attached and inserted into the post upon upload. This whole form is then stored in the database as an HTML file, so that it can be displayed in a widget as a single post.

5.4.2 Post widget

In a content feed, posts are displayed in a vertical list. To minimise scroll time, there is no gap between each post, only a white dividing line. Alongside the post are buttons for upvoting and downvoting, a link to the profile of the poster, as well as a button to view the replies to the post. The post display widget is the same in both profiles, groups, and search results, including a delete button that only appears if the ‘canRemove’ variable is set to true by the logged in user either viewing their own post or being a moderator or admin. React Quill is again used to display the content in the same format as it was submitted, with videos and images automatically resized to avoid taking up too much of the screen.



[Figure 15: Blank post creation form in a group]

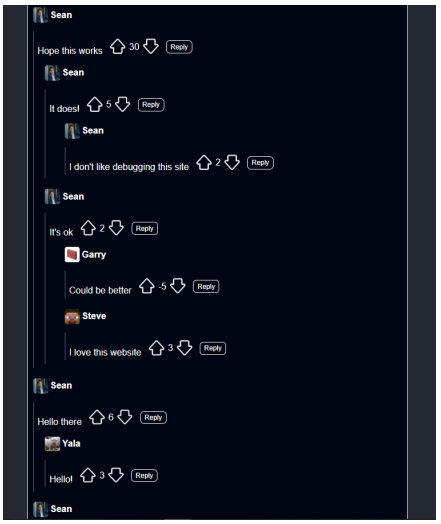
5.4.1 Content votes

Upon loading the post, a request is made to the ‘content_vote’ route to check if the user viewing the post has reached the up/down vote limit. If they have, the class of the up/down vote buttons is set to inactive, changing the background to grey and preventing any more of that vote from being cast, which also happens when the user hits the limit while voting. The same is done for replies.

[Figure 16: Upvote limit reached]

5.4.2 Post replies

All posts have a reply section that can be opened, along with a text area to write a reply, by clicking a button that also indicates how many replies have been made. Replies are nested, where one reply can be made to another in an unlimited chain. These replies are ranked according to net upvotes, with the same voting rules as the main post. Each reply has the profile picture that links to the profile of the user.



[Figure 17: Nested reply section]

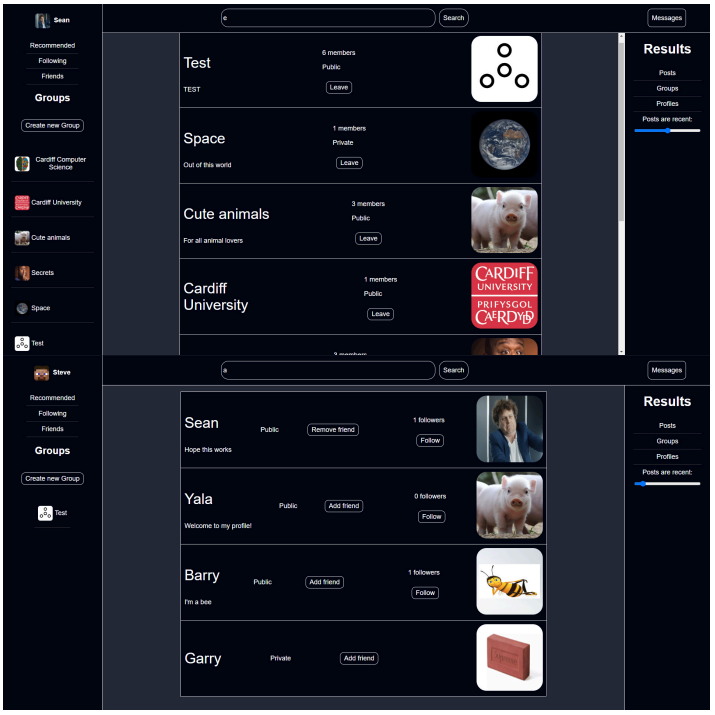
5.5 Search results

5.5.1 Display

When entering a search term, the user may be searching for posts, profiles or groups. To accommodate this, the right sidebar on the search results page includes a toggle between these types of results, with posts being the default option. The results for each option are returned in their own specific widget, with a button to join/leave for groups and friend/follower buttons for profiles. Clicking on the name or profile picture redirects to the profile/group page. Each post widget also includes the profile of its uploader. Switching between the options sends a request to retrieve a different set of results, which is quicker than fetching all 3 types at once upon the initial search, since the user may only be looking for one type of result. When searching posts, both group and profile results are returned combined, each using the standard content widget. Both the title and the body of the post are searched.

5.5.2 Search components

Like when loading an individual profile or group, the widgets need to check if the user is a member/follower and if a join/friend request has been sent, since the same button components are used. This requires the backend code to return the isMember and isRequestSent properties, which are found by checking the userId against the database tables for members/friends and requests and appending the result to the returned group/profile object. Widgets also display the public/private status of their profile or group. If no entry in the database matches the query, 'no results' is displayed.



[Figure 18: Group and profile search results]

5.5.3 Search function

The fundamentals of the search functions are quite simple, since developing a highly complex search algorithm is outside the scope of the project. The input string is converted to lowercase and compared against entries in the database. For posts, this means searching both the title and content columns of the profile and group posts tables, then combining the results, alongside the profile information of the uploader. These posts are then passed to the sorting algorithm, assigning each a score and returning results to the user with the highest scoring

posts shown first. Groups and profiles each have their own search function where only the user/group names are searched and information about the user's friendship and following of a profile, or membership of a group as well as and requests made, is returned.

5.6 Content feeds

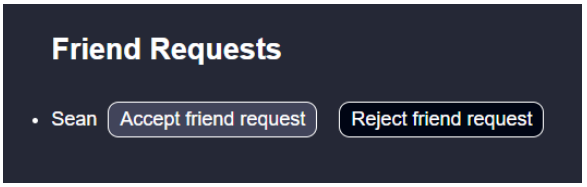
5.6.1 Post sorting

Post channels, search results and following feeds all have their posts ranked by the same algorithm, where posts with a higher ratio are ranked higher. Both the ratio of upvotes to downvotes and the upvotes per view are used, which are weighted by the time since the post was made. This approach was taken to display the highest quality content as voted on by the users, since better content will be upvoted more often than downvoted, as well as being upvoted more often per view (since a post may still have a good up/downvote ratio even if it is poor enough quality to not be voted on often). The time weight is 1 divided by 1 plus the number of minutes since the post was made, multiplied by the time_preference for each user (with a higher constant resulting in more recent posts being favoured), which results in the overall ratio becoming smaller over time. The ratio is then multiplied by the number of views, both because higher quality content will be viewed more often and to prevent content from being punished by being viewed multiple times by the same user (since the number of votes is limited but views are unlimited).

5.7 Messaging

5.7.1 Adding friends

Every profile and user search result has a button that checks if the logged in user is friends with the viewed user, showing ‘Add friend’ if not. Upon clicking this button, a friend request is sent that the receiving user can accept or reject. Upon accepting, a new ‘General’ chat channel is created automatically. In the friends database table, user1 is always the user that sent the request and user2 is always the user that accepted.



[Figure 19: Accepting a friend request]

5.7.2 Socket.io

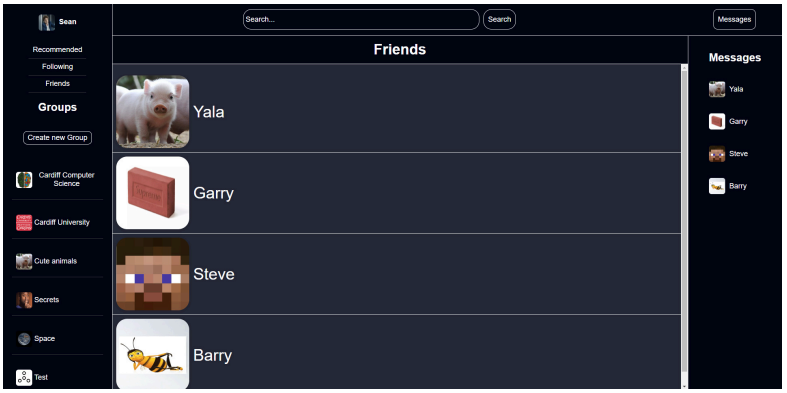
Direct messaging between users is handled by Socket.io, a library for real-time, bi-directional messaging. Upon loading each conversation, the user connects to a socket, which is used to listen for and send messages. Every time a message is sent, it is emitted to the socket so that it can be 'heard' by the other user. Each message is checked for its length to ensure it has neither zero characters or more than 1000. Messages sent by the user each have a delete button. When clicked, the message_id is sent to the backend and a 'destroy' request is made.

```
export const directMessagesSocket = (socket) => {
    try {
        socket.on('join_conversation', (conversationId) => {
            socket.join(conversationId);
        });

        socket.on('send_direct_message', async (message) => {
            const messageLength = message.message_content.length;
            if (messageLength === 0) {
                socket.emit('error_message', { error: "Message too short" });
                return;
            } else if (messageLength > 1000) {
                socket.emit('error_message', { error: "Message too long" });
                return;
            }

            const newMessage = await Messages.create({
                message_id: v4(),
                conversation_id: message.conversationId,
                sender_id: message.senderId,
                message_content: message.message_content,
                timestamp: new Date()
            });

            socket.to(message.conversationId).emit('message_confirmed', {
                ...message,
                message_id: newMessage.message_id,
                timestamp: newMessage.timestamp
            });
        });

        socket.on('leave_conversation', (conversationId) => {
            socket.leave(conversationId);
        })
    } catch (error) {
        console.log("Socket error:", error);
    }
};
```

[Figure 20: Socket.io code for sending a message]

5.7.3 Messages page

Since users must be friends to message each other, the base messages page displays a list of friends that directs to their profile upon being clicked. Friends are displayed according to the length of the friendship, with newer friends shown first. This is achieved by using order: [['FriendSince', 'DESC']] when fetching the friends. On the sidebar is a list of conversations that directs to the 'General' channel when clicked. From here, messages in the 'General' chat channel are viewed using the same chatChannel component as is used in groups, with messages sent by the user appearing on the right with a delete button, and the friend's messages on the left. This is achieved by checking the 'sender_id' against the logged in 'userId' and passing this prop to each message component, allowing the CSS class to be set to outgoing and 'canRemove' to be true. Having each message container that spans 70% of the screen width, then setting the message as having the 'flex-start' or 'flex-end' properties causes the messages to appear on the correct side.



[Figure 21: Friends page with conversation list]

5.8 Recommendation algorithm

5.8.1 Collaborative Filtering

A user may be more likely to prefer posts that are like those that their friends have upvoted. Sorting posts this way is Collaborative Filtering, calculated using the Pearson Correlation [Section 2.2.2]. This method is particularly well suited to the website, since it uses a voting range from –10 to + 10 resulting in larger variations between users compared to if only an individual upvote or downvote were counted. Votes of the user and votes of each friend are retrieved from the database and stored in lists. The Sequelize operator Op.or is used to ensure all friends are accessed, since a friend could be identified by either user1_id or user2_id. Lists of each friend are input into the Pearson Correlation function alongside the users vote list, resulting in a similarity score with each friend.

5.8.2 Content-based Filtering

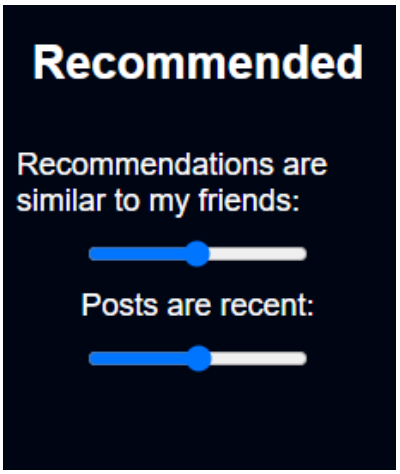
Recommendations are also filtered based on the user's past interactions with content, as determined by what they have previously upvoted. This is done by first finding the TF-IDF (Term Frequency-Inverse Document Frequency) of each post in the dataset, including the upvoted posts. This is found using the 'TfIdf' function imported from the 'natural' node library used for natural language processing. Once all these vectors are computed, the cosine similarity between the set of upvoted posts and posts to be recommended can be found, which measures the angles between vectors to determine their similarity [Section 2.2.1]. A smaller angle indicates more similarity, allowing a similarity score to be assigned to each post to be recommended. Posts are then sorted according to this score, with larger scores ranked higher. A difficulty of this approach is that the content of my posts is stored in HTML tag format. If the raw content was processed, these tags would introduce unfair similarities. Before processing, these HTML tags therefore had to be stripped. However, some image and video posts only contain a link to the media in their content column, meaning that stripping this leaves nothing to compare. Therefore, the title of each post is also included, both to ensure there is something to compare as well as providing more data to increase the accuracy of the recommendation.

```
//HTML tags are present in raw content
Codiumate: Options | Test this function
const stripHtmlTags = (content) => {
    return content.replace(/<[^>]*>?/gm, '');
};

//Ensures text is in same format
const normalizeText = (text) => text.trim().toLowerCase();
```

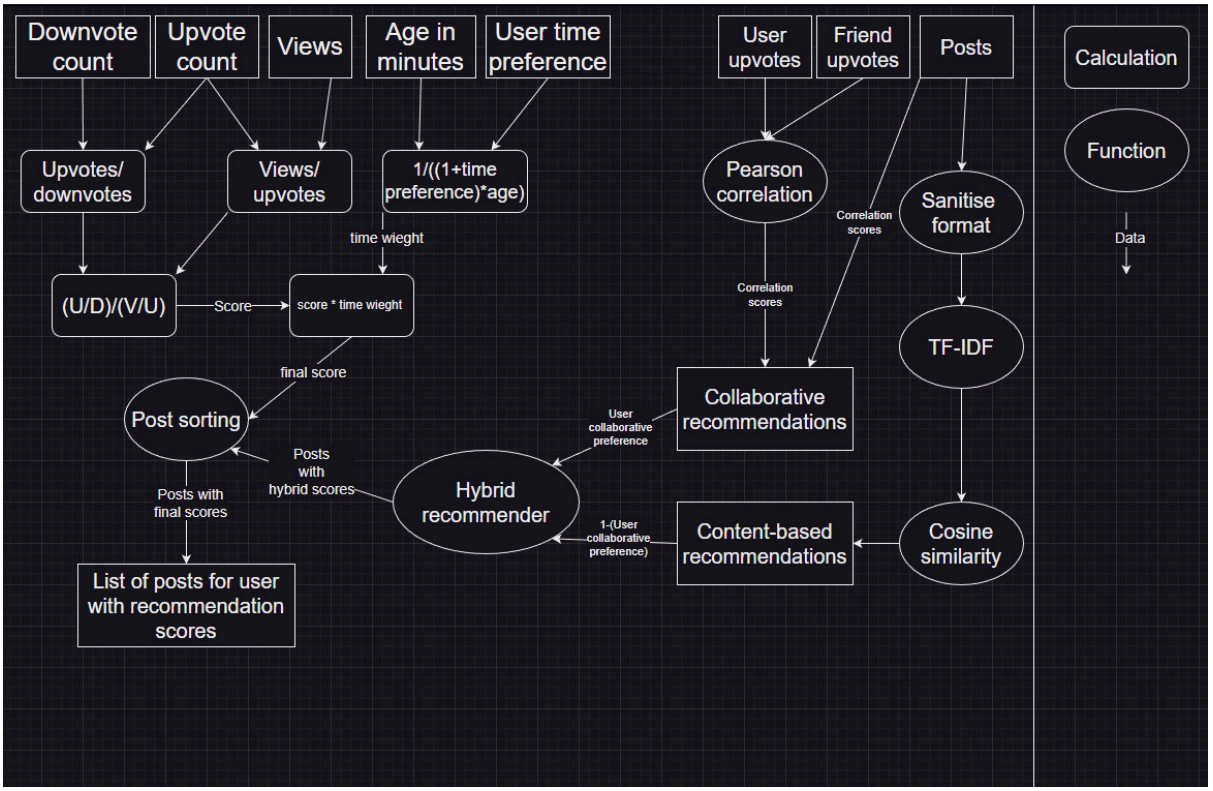
[Figure 22: Processing text input]

5.8.3 Hybrid filtering

Collaborative and content filtering are then combined in a hybrid algorithm, with each of the two methods having a different weighting on the result. This was done because each algorithm has weaknesses as discussed, so combining them creates a fairer algorithm overall. Also, different users will use the site in different ways. Some may have many friends that are similar to them, suiting collaborative filtering more. Others may use it alone, making content-based filtering better for them. By using a hybrid approach, the website not only becomes more effective but more inclusive too. This also allowed the implementation of a slider on the sidebar that saves user preferences for the extent that Collaborative or Content-based filtering is used. Since a regular user is likely to be unfamiliar with these terms, the label "Recommendations are similar to my friends" is used. A collaborative_preference value between zero and one is saved to the Users table, with one indicating an entirely Collaborative approach. One minus the collaborative_preference gets the Content-based preference, allowing the weights for the hybrid function to be set. Once the hybrid score for each post is calculated, the standard content ranking algorithm is applied so that posts are still influenced by upvotes, downvotes, views and recency.



[Figure 23: Algorithm customisation sliders]



[Figure 24: Flow of data for hybrid recommendations]

5.9 Styling

Since the focus was on feature implementation, I also couldn't have styling that was too time-consuming to implement. Since my target audience of young people and students are more likely to prefer websites in dark mode[12], a dark blue (hex code #000514) is the primary colour used across the site, with other colours being lighter shades of dark blue. Elements are separated using a one-pixel white border, resulting in a minimalistic yet professional looking appearance. These borders also create a clear contrast between elements which improves visibility for visually impaired users. Setting the default body font size is set to large also helps with this. To further increase the clearness of navigation, clickable elements are indicated with rounded corners (set by changing the border radius) except for list elements, which are separated by a grey bottom border. When clickable elements are hovered over, as well as input areas when selected, the background colour transitions over 0.3 seconds to a lighter shade of blue to cleanly indicate interactivity to the user. Most elements use a padding and margin of 10 pixels to avoid looking too compacted or too close to other elements.

6 Architecture

6.1 Technology stack

The site is built using the SERN (SQL, Express, React and Node) technology stack, resulting in all code being written in JavaScript. This allows for extensive use of libraries, resulting in enhanced features, cleaner code and quicker development. Although MySQL is used for storing data, it is accessed using the Node library Sequelize, removing the need to implement SQL queries in my JavaScript. Integrating these technologies together provides an essential boost in performance over other web frameworks like Flask[13], which is especially important when handling image and video media.

6.1.1 MySQL

MySQL is an open source database system written using Structured Query Language (SQL). The database itself was created and edited using MySQL workbench, hosted on a localhost connection. MySQL was chosen because of its easy-to-use syntax that is similar to natural-language, simplifying the process of database creation, as well as its ability to simply interact with JavaScript through Sequelize. Although a more lightweight database system like SQLite would've been suitable for the project too, MySQL was preferred due to its better scalability that can handle high traffic and many simultaneous connections[14], making the website more compatible with future development.

6.1.2 Node

Node is a JavaScript runtime that allows for websites to be scalable and run efficiently. I chose it because of its performance, since my website contains many pages and media elements. A key advantage of Node is its ability to host many connections simultaneously, performed using its event-driven and asynchronous Input/Output architecture[15]. This lays the foundation for a prototype to be scaled to many users. It also uses Node Package Manager (NPM), which provides me with easy access and quick installation of a vast collection of libraries and modules, both increasing the number of possible features and reducing development time.

6.1.3 Express

Express is a JavaScript framework that works with Node to provide Application Programming Interfaces (API's). The website consists of several dozen data routes, many accessed simultaneously, so a lightweight routing framework is crucial to prevent lag. I used Express because of its high performance, as well as simple and high-speed routing of data between frontend and backend. It also has close integration and common use with Node.js, since there is extensive documentation and many resources on their use together.

6.1.4 React

React is another JavaScript library often used alongside Node.js which allows for the building of single-page, component-based frontend applications. I chose it for its use of reusable components that data can be passed to, greatly increasing the speed and simplicity of development. Each component manages its own state, making the user interface fast and reactive. It also has a large ecosystem of libraries and tools, allowing my site to take on greater complexity and be developed quicker. Instant updates to on-screen components is done by setting a variable as null with useState e.g. [channels, setChannels] = useState([]), then setting the state upon the initial receiving of data when loading the page. When a component is updated, it sends an Axios request [Section 7.2.2] to the Express backend. This then sends a response with the updated data, allowing the variable to be newly set.

6.2 Libraries

6.2.1 Quill

To create and display posts, I utilised React Quill, a rich text editor that allows its text contents to be highly varied with a large number of modules, as well as the ability to add custom modules if needed. I chose this to make text content on my site more varied and engaging. It also automatically embeds external videos and keeps formatting of pasted text, making it easier for outside content to be shared. It does not natively support video and image upload, so I had to customise my code to support this. Although its forms have a default theme, the CSS for this can be edited, allowing their style to fit in with the rest of the site.

6.2.2 Axios

Axios is a library for simplifying HTTP requests[16]. It uses promises, allowing easier handling of responses and errors. It also includes interceptors, which can intercept requests and responses before they are handled, increasing the flexibility of middleware. Automatic JSON parsing is included too, alongside support for HTTP methods like ‘GET’, ‘POST’, ‘PUT’ and ‘DELETE’. All of this increases the reliability and reduces the complexity of the frontend routing code.

6.3 Code structure

6.3.1 Backend

Due to the large number of lines of code and files involved in the project, clean organisation and structure is essential. Most of the code is split between folders for frontend and backend, brought together by an app.js file. This is where an Express app is created, which is then told to use the /media directory for content uploads and /frontend/build for the React app. The website icon is used here too. To differentiate between routes for pages and backend data, all data handling routes are prefixed with /api. Socket.io is activated here too, followed by Sequelize and finally the running of the website on a server. Backend files are mainly split between the routes folder, where data is handled, and the functions folder, which contain reusable code to assist with data handling.

```
//api prefix prevents clashes with React app
app.use('/api/', authentication);
app.use('/api/', directMessages);
app.use('/api/', groups);
app.use('/api/', replies);
app.use('/api/', routes);
app.use('/api/', profileDataRouter);
app.use('/api/', profiles);

app.use(history('index.html', { root }));

app.get('*', (req, res) => {
    if (req.headers.accept.includes('text/html')) {
        res.sendFile(path.join(__dirname, './frontend/build', 'index.html'));
    } else {
        res.status(404).send('Not found');
    }
});

io.on('connection', (socket) => {
    directMessagesSocket(socket);
    groupChatChannelSocket(socket);
});

sequelize.authenticate()
    .then(() => console.log('Database connected...'))
    .catch(err => console.log('Error: ' + err));

//Start server
const PORT = process.env.PORT || 7000;
http.listen(PORT, () =>{
    console.log(`Server running on port ${PORT}`);
});
```

[Figure 25: app.js setup for Node]

6.3.2 Frontend

To further organise the frontend, JavaScript files are split between pages and components, and CSS has its own folder that all files import from. The pages are those that are referred to by URL, such as different feeds, groups, profiles or the login. The components are imported into their relevant pages, allowing data to be passed to them. These pages are imported into the React app.js file (separate from the Node app.js) and assigned a URL using react-router-dom. Except for the login and register pages, all routes beginning with '/' are rendered within the BaseLayout. Further route paths are then specified, such as profile and group pages defaulting to viewing the 'Main' channel. All elements of the frontend are compiled together automatically by React into single JavaScript and CSS files. Index.js mounts the app on a 'root' element in every page.

```
const App = () => {
    return (
        <Router>
            <Routes>
                <Route path="/login" element={<Login />} />
                <Route path="/register" element={<Register />} />
                <Route path="/" element={<AuthCheck><BaseLayout /></AuthCheck>}>
                    <Route path="recommended" element={<AuthCheck><RecommendedPage /></AuthCheck>} />
                    <Route path="following" element={<AuthCheck><FollowingPage /></AuthCheck>} />
                    <Route path="friends" element={<AuthCheck><FriendsPage /></AuthCheck>} />
                    <Route path="search/:tab?" element={<AuthCheck><SearchResults /></AuthCheck>} />
                    <Route path="messages/:username" element={<AuthCheck><MessagesPage /></AuthCheck>} >
                        <Route path=":friend_name" element={<MessagesPage />} >
                            <Route index element={<Navigate replace to="Chat" />} />
                            <Route path=":title" element={<MessagesPage />} />
                        </Route>
                    </Route>
                    <Route path="group/:group_name" element={<AuthCheck><GroupWrapper /></AuthCheck>}>
                        <Route path=":channel_name/:channel_mode?" element={<GroupHome />}>
                            <Route index element={<Navigate replace to="Main/post" />} />
                            <Route path=":channel_name/:channel_mode" element={<GroupHome />} />
                        </Route>
                    </Route>
                    <Route path="profile/:username" element={<AuthCheck><ProfileWrapper /></AuthCheck>}>
                        <Route index element={<Navigate replace to="Main" />} />
                        <Route path=":channel_name" element={<Profile />} />
                    </Route>
                </Route>
            </Routes>
        </Router>
    );
};

export default App;
```

[Figure 26: app.js setup for React]

6.3.3 Code standards

To avoid variable conflict and increase code clarity when data is passed between the front and back ends, I used different variable naming conventions. In the frontend, variables are declared using camel case (e.g. profilePhoto) whereas an underscore is used in the database and backend (e.g. profile_photo), although there is some mixing when data is accessed directly from the other end. To increase code readability, variables, routes and imports are listed in alphabetical order where possible. Variable names are intended to clearly, uniquely and concisely describe their function. Comments are also added to further clarify the purpose and functioning of some code, reducing confusion and development time.

6.3.4 Site security

Across the front and backend, all routes are wrapped in an authentication check to ensure that only the logged in user is accessing their data. Node handles session management, allowing the user_id of the session to be checked and an error returned if the user_id is not present. For the frontend, each route is wrapped in the 'AuthContext component', which again checks the session user_id. If successful, the username and user_id is returned for use by the frontend. If there is any error, the user is redirected to the login page. This has the effect of preventing the user from viewing any page, even by changing the url, unless they are authenticated. To prevent the website from crashing if an error is encountered, all const's and routers have their code contained within try/catch statements to ensure the site continues to function if an error occurs.

6.3.5 UUID

All data that requires a unique identifier is assigned a UUID v4, generated with the 'uuid' JavaScript library. This is a random 36-digit alpha-numeric code where the chances of collision (two items having the same ID by chance) is immensely small[17]. I chose this because, although the size of my project is limited, it is essential to use a reliable ID technique for larger applications, giving my site the potential to scale. The alternative was to generate IDs sequentially, but this has the security flaw of making ID's more guessable as well as greatly increasing the risk of ID clashes and limiting the ability of the database to be sharded.

7 Results

7.1 User research survey

20 people anonymously responded to the survey in an average time of 6 minutes each, similar to the recommended time of 5 minutes. Everyone completed the multiple-choice questions, and a large majority gave useful answers to the two text response questions. See Appendix A for the quantitative results and Appendix B for the text answer results.

7.1.1 Current social media views

Almost all respondents were users of YouTube, Instagram, WhatsApp and Snapchat. Several other sites were commonly used too, further showing how widely spread the current social media landscape is and the need for a site that can combine their main features. Most users were moderately satisfied with the content they receive from their current social media sites. However, it was particularly notable that nobody was very satisfied with their content, reinforcing the need to improve recommendation quality. Virtually all users were concerned about privacy when using their sites which they reiterated when asked about what their most common social media problem was. Advertisements, poor-quality short-from content and echo chambers were also mentioned.

7.1.2 Content preferences

Across both text answer questions, ‘content’ was the most commonly used word. A large majority preferred viewing posts and discussions equally, as well as having an equal preference for a focus on both communities and individuals, suggesting that building a site that balances all four was justified. This was further backed up by most users intending to use communities for viewing both posts and discussions. There was a notable difference in wanting to participate in public discussions and discussing only with friends, implying that adding private groups was an important feature. One user wanted the ability to view only their friends' content, separated from content provided algorithmically, showing that the dedicated friends feed was a useful addition.

7.1.3 New features

Most users would use an Artificial Intelligence co-pilot in some way, though not all. By far the most important use would be to help identify and check misinformation. Content generation, especially video, was not widely popular. Although such features would primarily be intended for content creators (and therefore not be relevant to many users), this could also be due to a distrust in the quality and necessity of AI generated content. Customisation of content recommendation was mentioned multiple times when users were asked what feature they would like to add, as well as more reliable fact checking. One user mentioned the lack of ability to express negative preferences, which the site addresses with its voting system.

7.2 User evaluation study

5 users who completed the initial research survey were asked to take part in an in-person evaluation of the finished prototype [See Section 3.1.2]. Each study took no longer than 10 minutes and was performed using my desktop prototype. See Appendix C for the full interview notes.

7.2.1 Appearance

Users generally commented about small details when asked about appearance, both positive and negative. Profile picture design was received well, since the site uses squares with rounded edges as opposed to the circles that are standard for most sites, which omit large amounts of the image. The layout too received praise for appearing clean and lacking clutter. Although the dark mode colour scheme was intended to suit the preferences of most users, some commented that the specific colours used could be adjusted. Some users would like to add light mode, or even the ability to change the scheme to a variety of colours. Each user emphasised that these were small considerations, and that the design overall was satisfactory. However, their comments still encouraged me to consider how future development could improve the appearance further [See Section 10.4].

7.2.2 Features

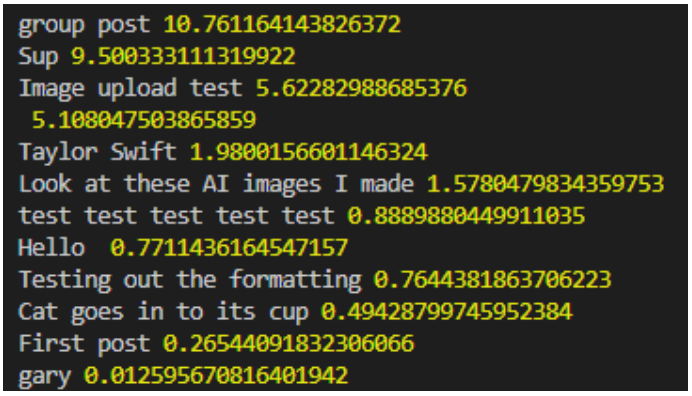
Most features received a positive response, especially those that were newly implemented. The voting system was very well appreciated for their ability to allow the user to more thoroughly express their content preferences. Multiple direct-message chats were also warmly received, since most users expressed their frustration with only having one chat per person on other sites. Customisable algorithms were welcomed too since they address user concerns about obscure and privacy violating content recommendation. A separate friends feed resonated well with Instagram users who were tired of their friend's posts being combined with those from other accounts that they follow as well as recommended content, justifying the creation of the 3 separate feeds. Those who used Discord and Reddit enjoyed groups having multiple channels, with each being able to switch between chats and posts, since it combined features they were already familiar with into a new format. Overall, users said that they liked the site and would be interested in seeing it developed further, including as a mobile app for them to use.

7.2.3 Improvements

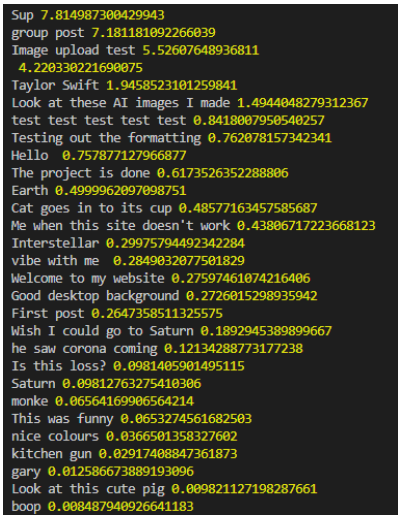
Since the product is a prototype, the users were still able to identify many areas for improvement. Due to some of the features being new, it was noted that the site may take time for a new user to become familiar with. It was suggested that an inbuilt tutorial or automatic tour of features upon sign-up would help make the user experience more friendly. The account delete button would be better suited if hidden within a menu, rather than being directly on the profile header. Delete buttons for messages could be better placed too. Overall, the message for improvements was that further customisation should be built, especially for the recommendation algorithm and visual appearance. Collapsible sidebars would be a small quality of life improvement too, since currently they are fixed in place. Sharing images, videos and other files would make the users even more likely to use the multiple chat channels, as discussed in Section 10.1.4.

7.3 Algorithm results

A successful algorithm result means that it assigns higher scores to posts that have more views, are more recent and have higher ratios of upvotes per downvote and views per upvote, as well as posts that are similar to those already upvoted by the user and their friends. Both the content ranking algorithm (used in group post channels, search results and the following feed) and the recommendation algorithm (used in the recommended feed) achieved the expected results, with more recent posts having especially high scores if the user time_preference was set to a high value. Each algorithm not only assigned a score to each post, but also fetched all relevant post data such as uploader profile information, votes, replies and views, allowing the results to be passed directly to a relatively simple /recommended_posts route.



[Figure 27: Content ranking algorithm sample data]



[Figure 28: Hybrid algorithm sample data]

```
//Accesses recommendation algorithm to provide content
router.get('/recommended_posts', authenticateCheck, async (req, res) => {
    try {
        const userId = req.session.user_id;
        const user = await Users.findByPk(userId);
        const recommendations = await hybridRecommendations(user);
        const sortedPosts = await sortPostsByWeightedRatio(recommendations, userId);
        //Logs scores
        //sortedPosts.forEach((post) => {
            //console.log(post.title, post.score);
        //});
        res.status(200).json({ success: true, recommendations: sortedPosts });
    } catch (error) {
        res.status(500).json({ success: false, message: error.message });
    }
});
```

[Figure 29: applying algorithms to recommended posts]

8 Evaluation

8.1 First objective

The first objective was met successfully, as all the intended features of a basic social media prototype were implemented. A lot of time was put into ensuring that the site functioned well, without any glitches, large delays or errors. If any of these were present, the user experience would be noticeably harmed. The website achieved this well, with all components successfully retrieving and accessing the database, as well as updating their state instantly. Although response times were usually fast, it is important to note that this was with a very limited database of content, as well as being hosted locally. Evaluating performance when used by many different users simultaneously, all remotely accessing much larger amounts of content, would be a useful subject for future work. Functionality was the primary focus during development, although care was still taken to make the site clean and presentable. The result was a user interface that serves its purpose well, but still has potential for further improvement [Section 10.4]. Posts are one of the most essential features to any social media site and were a primary focus of the first objective. They had originally been intended to allow the user to place videos and images anywhere within them, similar to a Word document, rather than being attached on the end. Although this specific feature turned out to be too complicated to implement in time, the final post form is still a success by supporting multiple file uploads and dynamic text formatting. My experience working with posts inspired ideas for future work [See Section 10.1.4].

8.2 Second objective

Several new or innovative features were created for the website, so this objective was met successfully too. Having many channels per group, each combining posts and chats, created a new way to enrich community interaction that balances user needs [Section 7.1.2]. This is further enhanced by the creation of nested groups, making group interaction even more dynamic. Applying this feature to profiles too helps users to more carefully curate how their personal content is posted. Multiple votes per post and reply, rather than simply ‘liking’ the content, provides a much clearer way for users to express their preferences, both positive and negative, helping to meet their need for higher quality content. Allowing users to multi-communicate by having multiple direct message channels between them helps reduce the need to spread themselves across many different social media apps, although adding the ability to share content [Section 10.1.4] would enhance this effect further.

8.3 Third objective

The website includes functional algorithms for aggregating and serving and content, used by both content gathering and recommendation feeds, so this objective was also met. The purpose of this objective was to help address user concerns about content quality. Some users in my research expressed a desire for more customisable algorithms and many were concerned for privacy [Section 7.1]. Collaborative filtering only applies to friends so that a user's votes are only compared to people they know, rather than anonymous members of the site at large, to protect privacy. The slider for the extent that collaborative filtering is applied enables further privacy (allowing comparisons to other users to be turned off completely) as well as enabling customisation. This user control is enhanced further by the slider for determining how recent posts are. Since these are new features, they help to meet the second objective too. Although the algorithms work well on my limited database of content, all posts made to the website are currently fetched to be scored. For future work they'd need to be adapted to quickly and efficiently assess greater numbers of posts [Section 10.3.3].

9 Conclusions

9.1 Improving social media

This project started as an exploration of the issues faced by users in the current social media landscape and what can be done to help them. I had many of my own thoughts, so it was enlightening to see not only some of my own sentiments echoed by others but also entirely new perspectives that I hadn't considered. Some of the features I'd already had ideas about building such as a friends feed, changes to content voting and algorithm changes were also independently thought of by the surveyed users, reinforcing that the project was headed in the right direction. Hearing directly from users also inspired me to spend more time thinking about how the project's progress could be expanded upon in the future [See Section 10]. The most important conclusion from my research is that social media websites in general still have much work to do to satisfy user needs, although fully satisfying users may directly conflict with the need to make money through advertising.

9.2 The final product

Building any form of social media website is an incredibly complex task, so the final product must be evaluated in the context of being a technology demonstrating prototype. For this purpose, it has succeeded beyond what I had expected at the beginning of the project. Starting with many goals and ideas, I had expected that the project would be hard to complete in time. However, I was surprised that most of my features could be successfully implemented in the end and that I didn't have to scale back compared to what was proposed in the Initial Plan. Creating a successful hybrid recommendation algorithm was particularly important, since during the initial research I assumed that a choice would have to be made between Collaborative and Content-based filtering, which would have decreased the quality of recommendations. Adapting both of these algorithms to work together therefore greatly improved the final product.

10 Future Work

10.1 Expanded features

10.1.1 Improved channels and feeds

Further spreading the functionality of a social media site into different specialised channels could be applied to a wider range of settings. For example, the user could split the users and groups they follow into different feeds, allowing them to be viewed separately in different ‘playlists’ that are continuously updated. Instagram already has a separate feed for followers, but this does not allow the accounts to be split and lacks functionality for separate friends. An option for a user's posts in groups to also be displayed in their profile could be added too. The post channels on a user's profile could also be assigned different permissions, allowing only selected users to view them. Posts are stacked vertically, so another option to view two or even three posts side-by-side could be useful, especially for search results. Adding voice chats and video calls, both in groups and between users, would make channels more multimodal and greatly improve the experience and ease of communication between users. Adding a main feed, combining recommendations, followers and friends, could avoid the user having to switch between different feeds to view all their content.

10.1.2 Multiple profiles

The code already separates users from their profiles, which was intended to allow for the future work of a feature where each user can create multiple profiles. Some may be public, others private. Like channels, which friends each profile is visible to could be specified. This could be useful when a user's friends include family members or professional connections, since a different profile can be made visible to each type of friend. Each profile having a different name and profile picture, with the user not shown, would allow for each user to manage their entire online identity from one account, since many people have different personas on all their various social media platforms. There would have to be a limit on such a feature (e.g. 10) to prevent spamming of different identities, as well as managing all profiles from a single page to reduce complexity.

10.1.3 Enhanced search

A channel to view replies could be added to the search results. Various filters such as upload time ranges, views, net upvotes, location and topic would be greatly useful to users, although may result in slightly longer response times. The similar filters could be added to group and profile channels too. Search currently is quite simple, matching the input string with the title and contents of the database post tables. Further development and research could apply machine learning techniques to enhance the relevance of results, as well as use computer vision to directly analyse visual content for relevance to the query. Performing such analysis once upon content upload (rather than every time it is accessed by the user) would massively save on computation cost, as well as provide data for the recommendation algorithm.

10.1.4 Content upload

Currently, posts are the only part of the site where non-text content can be uploaded. Future work could allow replies and chats, both between users and in groups, to include different forms of media too. These don’t need to just be images and video, but can also include audio, PDF’s and any other type of downloadable files. Such upgrades would be particularly useful to students and project collaborators, as well as providing a much richer media environment for all users. Post quality can be improved further by including these files within the upload form, rather than attaching them at the end. Also, multiple pages could be made per post, in the same way as uploading multiple images on Facebook or Instagram, rather than all content being included in a single form. This would result in much more engaging and in-depth posts, since many long-form collections of text and media uploaded together allow topics to be explored in more detail. Adding the ability to share these posts in chats, as well as crossposts between profiles and groups, could even further enrich the interactions between content and users.

10.1.5 Remote hosting

Hosting the prototype using a cloud service, such as one provided by the university, would have expanded its functionality by allowing users to access it remotely. However, this was optional and not one of the main goals of the product, so my time was instead better spent on building a locally hosted prototype that worked well. Although I tried to upload the website to Cardiff University's OpenShift, my VPN repeatedly would not allow me access. Since the prototype itself functions well, this was ultimately not an important issue and does not make the project any less successful, since all 3 objectives were still met. Remote hosting will be highly useful for future work to be done, since larger numbers of users can be tested.

10.2 Artificial Intelligence copilot

10.2.1 Features

My initial research included a question about the uses of an AI-copilot, since I was curious about how this could apply to future research. As the results showed, almost all users would use such a feature, with help countering misinformation being particularly important. This could come in the form of the search bar having two buttons: one for searching and one for talking with the copilot. This copilot could have access to some tables in the site's database, allowing it to include relevant posts, replies, groups and profiles in its response, allowing the user to discuss results with the copilot to find the best content for them. When viewing a post or reply, there could also be a 'copilot' button that checks the contents of the post, providing additional information and context or countering any misinformation.

10.2.2 Considerations

Choosing a language model for this feature would involve trade-offs between speed, size and accuracy, since models with fewer parameters are faster and may even run locally, but more often produce strange and inaccurate outputs. A hybrid approach of a small local model assisting with using the site, such as asking about features or finding posts, and a larger model when checking for misinformation would help maintain trust (since inaccurate results would quickly be ignored) while reducing cost and maximising speed.

10.3 Algorithm improvements

10.3.1 Content ranking

The current content ranking scores for group channels, the following feed and search results are generated only using upvotes, downvotes, views and upload time, resulting in the same feed for all users in a group. Since preferences for content within a group are likely to be quite similar, there is unlikely to be a need to customise the results to each user in the same way as the recommendation algorithm. However, more sophisticated techniques could still be used. The factual accuracy of each post could be considered [Section 11.3.2], as well as its similarity to other highly ranked posts, measured by collaborative filtering [Section 6.8.1]. Information about the uploaders themselves may be useful to include, such as their follower count, so that the performance of the content can be compared against the total audience size of the user.

10.3.2 Tackling misinformation and harmful content

One of my user's main concerns is recommending content that contains misinformation and not being able to tell if it is real, as well as the presence of hateful and abusive posts. Although addressed by the copilot, not being recommended the content in the first place would be preferable. Further work on the recommendation algorithm could address this concern by adding an 'accuracy' weighting to each post, as well as using Natural Language Processing to assess the sentiment of each post. Using a large language model (LLM) connected to the internet is one possible way to assess misinformation, although there are the same trade-offs as mentioned with the co-pilot [Section 10.2.2]. To avoid performing computationally expensive analysis on every post, it could be done only when a user presses a 'check' button on a post or reply, both producing a note for the user and an accuracy weighting that can be saved and applied to the recommendation algorithm. It would be essential that the design and results of such a system be transparent and available to users, as well as having human overrides to overcome any language model mistakes (e.g. a community of human fact-checkers and site-wide moderators). To not do so would risk an obscure and sometimes inaccurate system, resulting in low trust and potential unfair censorship of posts. Transparency and reversibility are also essential when adjusting the ranking of potentially hateful posts since some, especially satire or quotations, could be unreasonably suppressed.

10.3.3 Recommendation

Only text content is compared in the current algorithm, meaning that the recommendation of posts only containing images and video is solely based on their titles, potentially producing recommendations that do not fairly assess the post's content. Future work could overcome this by producing video transcriptions and using alt texts to describe images, or even using computer vision to directly assess the media's content. Algorithm scalability would have to be improved too [see Section 2.2.2]. Since no user viewing history is implemented, the same content is recommended when the page refreshes, only changing if the data used to score the content is changed. Adding such a history would ensure only new posts are shown. Privacy must be ensured when implementing viewing history, with its data encrypted and the option to delete history given to the user. Since privacy is a primary concern for my users, any algorithm that recommends them content cannot rely too heavily on personal data. This is why only the votes of friends are considered, rather than anonymous users. However, some may be more open to their votes being compared to others, so an option to compare with followers or even with any user could be added. Since this feature would have access to more data, its recommendations may be more accurate and could also be used to recommend new friends too, as well as people and groups to follow.

10.4 User interface

Further work on the user interface would streamline and refine the experience for the user. This may include a toggle between light and dark mode, since some may prefer a lighter background. Grouping some buttons together into drop-down menus would reduce clutter, such as on the profile and group administrator pages. Other buttons could appear when an element is clicked, such as the ‘delete’ option only showing when a message is selected, rather than always being next to it. Creating icons instead of words (such as with the voting arrows and post form) would simplify the interface more, although where exactly this is appropriate should be tested with users first to avoid confusion.

11 Reflection on learning

Completing this project has been a challenging yet fruitful learning experience. It contains by far the most code I have ever created for a single application, as well as the longest report I've yet written. Both products have been an essential test and development of my skills, preparing me for future success.

11.1 Planning

The timeline set out in the initial report was ambitious, but achievable. This alone was a good exercise in aiming high. Almost all the features outlined in Section 11 were ideas to be implemented, so showing restraint and focusing on what was necessary to complete in the time given was a challenge. I feel that I laid out the project structure well, focusing first on completing a basic website where more complicated features could be built on top, followed by writing. My main lesson was how to not overestimate what could be done in limited time while still maximising what was achieved.

11.2 Research

To know what to build, I had to be familiar both with what already exists, its limitations and what users are looking for. Although I'd created online surveys before, this exercise was good practice for carefully selecting a limited number of relevant questions, since length surveys are less likely to be completed. Researching the wider social media landscape was a useful learning experience too, since I already had ideas about what was wrong and what needed to be improved but I had to be careful to only pursue ideas that had evidence. Despite my goal to build a recommendation algorithm, I initially wasn't familiar with how this could be done. Learning complicated technical concepts from academic papers and implementing them in code was the most difficult aspect of the project, so this was a crucial skill to learn. First, writing pseudocode derived from the techniques of each paper made it much easier to translate the ideas into production code.

11.3 Code development

I already had experience working with my technology stack, greatly speeding up the coding learning curve. However, the project objectives were much more ambitious than my previous work, still requiring me to learn new techniques [Section 3.2]. The result is bringing my relatively beginner JavaScript coding skills much closer to industry standard, paving the way to even bigger work in the future. Coding the algorithms was the most difficult part of the project, taking the most amount of time and being finished close to the deadline. Although I scheduled to start them with plenty of time remaining, other development of the site resulted in algorithm development being delayed. If I were to do the project again, I would start with

developing the recommendation algorithm alongside the basic prototype, rather than waiting until it was almost finished.

11.4 Testing

This was the most time-consuming part of the project, which is a valuable insight for future work. Although I took a significant amount of time to carefully write the code, it was almost impossible to write everything perfectly the first time. This taught me to include logs for errors and to track the flow of data whilst writing the initial code, rather than add them in later to try and find a bug. Testing the algorithms was challenging and time consuming, since I had to repeatedly make modifications to get a correct score. Most often each piece of content would be given a score of zero, indicating that the algorithms were not functioning correctly. I tested each individually and only created the hybrid algorithm once both were working which emphasised the importance of having the time to thoroughly test, since delays will affect the implementation of software further development. Deploying the website beyond the desktop where it was developed took much longer than expected due to difficulties with duplicating and running the database on my laptop. This was necessary to make the website portable for user testing and I incorrectly assumed that it would be relatively straightforward and therefore left it until the end of development. The result was that I wasn't able to demonstrate the prototype during development, which would have created more actionable feedback compared to only demonstrating the finished product. In future, I will ensure that seemingly quick tasks are tested further in advance of deadlines, since unforeseen errors can cause serious delays.

11.5 Writing

It was essential not to leave too little time to produce the report, both to ensure enough time to write in sufficient detail and to edit my work. However, given the number of technical tasks I had to complete, this was difficult. I overcame this by regularly writing small amounts, around 100 words, in between writing and testing code. This was especially useful whenever a feature was difficult to debug, since taking my mind away and focusing on another task allowed me to come back with a fresher perspective, sometimes making it easier to spot the mistake. I can apply this style of working in the future to enhance my productivity, particularly when a deadline is approaching.

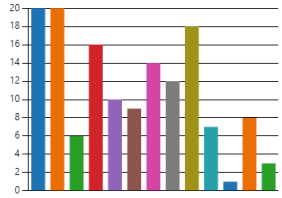
Appendices

Appendix A

2. Which of the following sites do you use?

More Details

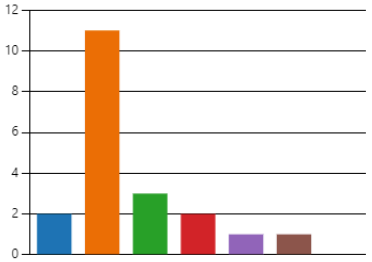
YouTube	20
Instagram	20
TikTok	6
Snapchat	16
Facebook	10
Twitter/X	9
Discord	14
Reddit	12
WhatsApp	18
Pinterest	7
Tumblr	1
Quora	8
Others	3



3. To what extent are you concerned about your privacy when using these sites?

More Details

Very concerned	2
Moderately concerned	11
Slightly concerned	3
Neither concerned nor unconce...	2
Slightly unconcerned	1
Moderately unconcerned	1
Very unconcerned	0



5 respondents (**28**%) answered **content** for this question.

6. Do you prefer viewing posts or discussions on social media?

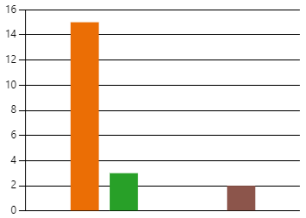
More Details

Posts	6
Discussions	1
Both equally	13

5. Overall, how satisfied are you with the content quality on the sites that you use?

More Details

Very satisfied	0
Moderately satisfied	15
Slightly satisfied	3
Neither satisfied nor dissatisfied	0
Slightly dissatisfied	0
Moderately dissatisfied	2
Very dissatisfied	0



7. Do you prefer sites that focus on communities or individuals?

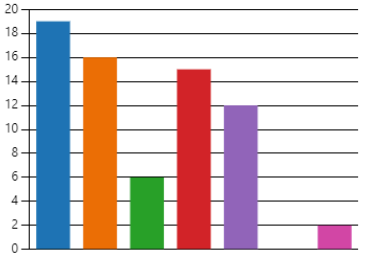
More Details Insights

Communities	5
Individuals	4
Both equally	11

8. What would you/do you use online communities for?

More Details

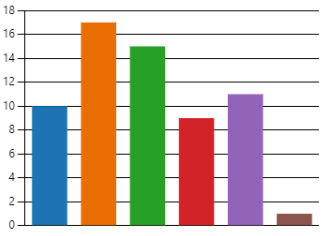
Viewing posts	19
Viewing discussions	16
Participating in discussions	6
Talking with people you know	15
Sharing content	12
None of these	0
Other	2



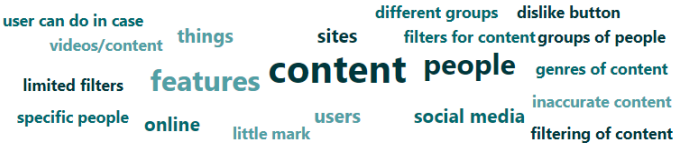
10. Which of these are important to you when you are recommended content?

More Details

I see posts from people I follow	10
I see posts from my friends	17
The content don't contain any ...	15
The content is similar to posts I'...	9
Being regularly exposed to a var...	11
None of these are important to ...	1



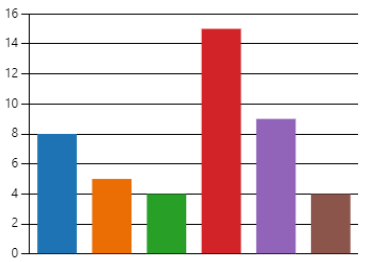
4 respondents (**27**%) answered **content** for this question.



9. Which of the following would you use an Artificial Intelligence 'co-pilot' for on a social media site?

More Details

Generating text	8
Generating photo-realistic images	5
Generating photo-realistic videos	4
Fact-checking	15
Assisting with using the site's fe...	9
I wouldn't use it	4



Appendix B

What is your biggest problem when using your social media sites?

Low quality, short-form content intended for grabbing attention rather that being accurately informative
How many ads there are
Lots of unfiltered and inappropriate content that can be accessed by anyone without context
Echo chambers
Ads
New monetisation is at the expense of website quality.
Privacy. I want my data to be protected.
Usage of personal information
Large amounts of advertisements.
I kind of feel watched because of how extremely specific the content and advertisements I receive are
Lack of self-hosted alternatives
How much misinformation there is and how emotionally draining it is when scrolling through the posts. Also doom scrolling.
They seem to keep track of all my activity and data accessed
Content reach / communication - there is an ability for bias, as certain voices and viewpoints may be amplified over others due to how content is shared / how certain content sources may be more significant in user feeds than other sources.
Doom scrolling and can't control the content I'm seeing easily
That they are defo selling info to anyone willing to pay
Drama between influencers, difficult to determine if media is true or false, spending long amounts of time on social media
Scrolling for longer than intended as a way of procrastinating.

If you could add one feature to any social media site, what would it be?

Having different profiles for different groups of people to see e.g. friends, family, public
A very in depth recommendation filter which could include flags such as: choosing specific people you follow, genres of content, recency of posts, adjustable bias towards new or similar content. This would be mainly for sites like Instagram or twitter where there are limited filters for content; usually only sensitive and if you follow or not
Filter out ai generated results
Any social media that doesn't have a gif search
Remove cookies for more privacy.
More reliable fact checking.
I would add the ability to mark certain videos as either comedy or factual/informing videos. It would be nice if some videos/content had such a mark, put on by other viewers as I sometimes get confused as to whether the information shared is true or not. I also think this could help people who easily are tricked by things online to avoid problems. An icon or a simple little mark would help with this
Option to customise your own feed better
Restore the dislike button on YouTube (/ for any social media site, ensure it has like and dislike features), allowing inaccurate content to be more quickly identified by users.
Easier filtering of content, seems like expressing disinterest doesn't do much
The feature which cracks down harder on racism and abuse in these apps
When it comes to online safety, have a feature about what a user can do in case of things like hacking and cyber bullying (unsure of how this may be incorporated, but it gives users ideas of how they can help themselves or who to speak to in these sorts of situations)
The ability to disable scrolling through short videos and posts whilst keeping the ability to see friends' content and direct message. This would allow me to use social media for communication whilst not getting distracted by or addicted to scrolling

Appendix C

What are your thoughts on the appearance?

List dividing lines look nice, profile picture shape better than circle, background colour could be improved, font looks a little basic. Only small changes, ok overall
Dark is nice but background looks strange, Discord and Tumblr vibes
Easy to navigate, not too much clutter
Better to have a light mode but appearance is nice, not too cluttered
Dark mode is nice, overall quite clean, like the fade effect on buttons, adding menus might be good

What are your thoughts on the features?

Upvoting system is 'so cool', nested comments are nice, nice that shows message not deleted
Really cool, like multiple channels, voting system interesting (but feeling is neutral)
Cool features, good quality of life features
Like that combines features for different site, good that no stories
Seems innovative, algorithm customisation is good idea, groups channels are useful

What did you like most about the prototype?

Multiple chat with same person
Split channels in groups
Customisable recommendations
Customisable algorithm, separate friends feed
Friends and followers separated from recommendations, multiple chats with each user

What did you dislike most about the prototype?

Background colour
Delete button should be moved
Can't change to light mode
Quite a lot to learn for a new user, no tutorial
Not being able to send content to friends

What features, if any, do you think should be added?

Share videos/images with other users in direct messages
Make mobile app, collapsible sidebars, multiple modes of content interaction, notifications, search chat for messages
Built in games in chats, dedicated games channels
Add more algorithm customisation options, including do not recommend for certain content types
Much more detailed customisation of feeds (more than 3), algorithm and colour scheme

What features, if any, do you think should be removed?

Delete account button not so easy to press
Delete button next to messages changed
Nothing
Wouldn't remove anything
None, happy with everything

Anything else you'd like to add?

I like it, I'd use it
Make site overall much more customisable
Could have disappearing posts. Temporary live chats too
Think it's great
Would use the site once further developed

References

[Figure 1] Chiny, M., Chihab, M., Bencharef, O. and Chihab, Y. (2021). Netflix Recommendation System based on TF-IDF and Cosine Similarity Algorithms. *Proceedings of the 2nd International Conference on Big Data, Modelling and Machine Learning*. [online] doi:https://doi.org/10.5220/0010727500003101.

[Figure 2] Patil, A. (2021). *Pearson's Correlation Coefficient - A Beginners Guide*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/01/beginners-guide-to-pearsons-correlation-coefficient/.

[1] Stacy Jo Dixon (2024). *Most popular social networks worldwide as of January 2024, ranked by number of monthly active users*. [online] Statista. Available at: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/.

[2] Aloudat, A., Al-Shamaileh, O. and Michael, K. (2019). Why some people do not use Facebook? *Social Network Analysis and Mining*, 9(1). doi:https://doi.org/10.1007/s13278-019-0564-z.

[3] Arifianto, M. and Izzudin, I. (2021). Students' Acceptance of Discord as an Alternative Online Learning Media. *International Journal of Emerging Technologies in Learning (iJET)*, [online] 16(20), pp.179–195. Available at: https://www.learntechlib.org/p/220539/.

[4] McLachlan, S. (2023). *Instagram Demographics in 2023: Most Important User Stats for Marketers*. [online] Hootsuite Social Media Management. Available at: https://blog.hootsuite.com/instagram-demographics/.

[5] Thomas, V.L., Chavez, M., Browne, E.N. and Minnis, A.M. (2020). Instagram as a tool for study engagement and community building among adolescents: A social media pilot study.DIGITALHEALTH,6,p.205520762090454. doi:https://doi.org/10.1177/2055207620904548.

[6] Chiny, M., Chihab, M., Bencharef, O. and Chihab, Y. (2021). Netflix Recommendation System based on TF-IDF and Cosine Similarity Algorithms. *Proceedings of the 2nd*

International Conference on Big Data, Modelling and Machine Learning. [online] doi:https://doi.org/10.5220/0010727500003101.
[7] Su, X. and Khoshgoftaar, T.M. (2009). *A Survey of Collaborative Filtering Techniques*. [online] Advances in Artificial Intelligence. Available at: https://www.hindawi.com/journals/aai/2009/421425/.
[8] Sharma, S., Rana, V. and Malhotra, M. (2021). Automatic recommendation system based on hybrid filtering algorithm. *Education and Information Technologies*. doi:https://doi.org/10.1007/s10639-021-10643-8.
[9] Tessem, B. (2014). Individual empowerment of agile and non-agile software developers in small teams. *Information and Software Technology*, 56(8), pp.873–889. doi:https://doi.org/10.1016/j.infsof.2014.02.005.
[10] discord.com. (n.d.). *Student Hubs*. [online] Available at: https://discord.com/student-hubs.
[11] Reinsch, N.L., Turner, J.W. and Tinsley, C.H. (2008). Multicommunicating: A Practice Whose Time Has Come? *The Academy of Management Review*, [online] 33(2), pp.391–403. Available at: https://www.jstor.org/stable/20159404 [Accessed 13 Apr. 2024].

[12] Virtanen, J. (2023). *Dark Mode Preferences: Exploring User Motivations in Interface Theme Selection*. [online] Available at: https://www.utupub.fi/bitstream/handle/10024/176173/Virtanen_Julius_opinnayte.pdf?sequence=1.

[13] K. Lei, Y. Ma and Z. Tan, "Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js," *2014 IEEE 17th International Conference on Computational Science and Engineering*, Chengdu, China, 2014, pp. 661-668, doi: 10.1109/CSE.2014.142.

[14] www.redswitches.com. (2023). *Understanding SQLite Vs MySQL: Comparing Databases For 2024*. [online] Available at: https://www.redswitches.com/blog/sqlite-vs-mysql/.

[15] Chaniotis, I.K., Kyriakou, KI.D. & Tselikas, N.D. Is Node.js a viable option for building modern web applications? A performance evaluation study. *Computing* **97**, 1023–1044 (2015). https://doi.org/10.1007/s00607-014-0394-9

[16] Axios (2023). *Getting Started | Axios Docs*. [online] axios-http.com. Available at: https://axios-http.com/docs/intro.

[17] Hall, J. (n.d.). *What are the odds?* [online] Jonathan Hall. Available at: https://jhall.io/archive/2021/05/19/what-are-the-odds/.