Department of Computer Science

Loughborough University

BSc COMPUTER SCIENCE
23COC251

Developing a University-based Social Media Web Application

Author:

Anant Shah

Supervisor*:*

Dr. Parisa Derakhshan

May 2024

Abstract

Starting university is a critical but challenging transition for students, many of whom move away from home for the first time. The majority of students start university directly after finishing school, marking not only the initiation of their university experience, but also their transition into young adulthood. This significant change brings about the necessity for students to meet new people and make friendships in an unfamiliar environment, a task that can be overwhelming and daunting. Students currently rely on a wide array of social media platforms, most commonly consisting of Facebook, Instagram, and WhatsApp to build and maintain these connections. However, upon starting university there is a lack of information for new students on which Facebook and WhatsApp Group chats to join or which Instagram accounts to follow. This lack of information can cause many students to miss out on early opportunities and events posted on these groups and social media accounts to meet other students. This leaves some students struggling to form connections and establish friendship groups.

This project aims to develop a dedicated social media platform for university students. It will allow them to form specific groups based on their university courses, halls of residence, sports teams, or societies. The goal is to provide a more streamlined process for students to connect with their peers and to help with the making of new friendships. The website will serve as a single point of contact for students to engage with their university's community, reducing the need to navigate amongst several other social media apps. Features, such as groups, private messaging, and event planning are all integrated into a user-friendly interface. All these features are clearly displayed and always accessible from any page via the navigation bar, simplifying the process and eliminating any confusion, which enhances the process of students navigating to the area that they desire.

Contents

List of Abbreviations

- **MVC** – Model View Controller
- **HTML** – Hypertext Markup Language
- **SQL** – Structured Query Language
- **DRY** – Don’t Repeat Yourself
- **CoC** – Convention over Configuration
- **ACID** – Atomic, Consistent, Isolated, Durable
- **RDBMS** – Relational Database Management System
- **SNS** – Social Networking Site
- **XP** – Extreme Programming
- **IDE** – Integrated Development Environment
- **VS Code** – Visual Studio Code
- **CSS** – Cascading Style Sheets
- **REST** – Representational State Transfer
- **ORM** – Object Relational Mapper

List of Tables

List of Figures

Chapter 1

Introduction

Contents

1.1 Problem Domain

Incoming first year university students often struggle with connecting with their peers to form friendships both prior and during university. Universities currently lack an official method for students to get in contact with their peers living in the same halls of residence, studying the same degree, and members of sports teams and societies that students intend on signing up to. Instead, students settle with using various third-party social media platforms to find the specific accounts/groups associated with the relevant group they were looking for, for example their hall of residence, or degree course. Furthermore, these platforms aren't dedicated to helping students connect and therefore have mediocre features which aren't the best to streamline this process.

1.2 Aims and Objectives

My proposed solution is to create a dedicated platform, in the form of a website, that is tailored towards helping students connect with other students during their time at university.

The website will provide two primary services: helping students meet other students with common interests to establish new friendships; and be the platform that students use as a messaging/social media service for their halls of residence, course, as well as the societies and sports clubs they are members of.

I intend for my solution to be the official service used by universities in the UK, so I will ensure that students must go through a verification process to verify their attendance at a university via their student email, along with the potential in the future to provide each university customisation of their university's portal on the platform.

My overall aim is to help students overcome their nerves and prevent loneliness whilst at university, as the process of a young adult moving away from home to a new environment of other students that they don't know is an overwhelming and intimidating experience.

The objectives of the project are to:

1. Conduct a literature review to find well-suited technologies and methodology to complete the project.
2. Carry out a gap analysis to find the strengths and weaknesses of the existing solutions, and then determine how I can use the key findings from this research to improve my solution.
3. Create designs to produce a clear and easy-to-use graphical user interface.
4. Develop a web application that will help students create friends at university.
5. Research testing methodologies and use them to assist in producing a robust product.

1.3 Motivations

The primary motivations for undertaking this project are:

- To create a platform that students can use to help find other students at their university, particularly a product that I wish existed for me when I started university.
- To utilise the web development and software engineering skills that I learned from my university course.
- To learn new skills in web development, specifically in JavaScript libraries.

1.4 Project Plan and Timeline

My workplan starts off by planning the project and how I will develop it over the year. This timeline (as shown in the Gantt Chart below) allows a break in January to accommodate the Semester 1 exams. Additionally, I've given myself an earlier deadline to allow for extra time and flexibility in case any project development issues arise and cause delays.

Before I start writing any code, I will focus on the design of my project. As part of this, I will list out all the features and elements I want the project to include and specify where. For example, in the user settings, I want a user to be able to change their password, profile picture, etc. and also be able to see their account details. Once I have established this list, I will then create page designs on how I want each page to look like. I will likely use the software draw.io in order to create these designs. My aim is to make the user interface user-friendly, bright, and colourful, include light and dark modes, and designed in a way that allows the interface to be easily adaptable based on the screen size of the user. For some of the page designs, I will make more than one design in order to figure out which design layout is better.

I also want to complete some research and testing on the various web technologies that exist, to determine which I would like to use as part of my development (e.g. node.js or react.js). The testing part of this will include some coding, but it will mostly be to test the features of each technology, and this code will likely not be used in my final project as it won't be clean, efficient coding.

Alongside this, I will start creating the database format for my website. I will draft up the database tables using Microsoft Excel, and research into which database management system I want to use (for instance SQL Server, MySQL or Oracle). As you can see in the Gantt chart, I intend to initially create the database, but then later optimise it once I have started coding, by adding/removing tables or columns that I initially missed. Additionally I intend to make the database tables more efficient, as the first draft of the tables will consist of first and second normal form and over time, I will transition the tables to be third normal form.

In terms of developing and coding the website, I will aim to split this up into frontend and backend development. Throughout development, I will bounce back and forth between the two developments, to integrate them together to solve any migration errors, as well as get early versions of the website, which I will use to frequently review what I have developed to my initial plan and design.

In both the website design and while I code the frontend, I will account for the website's mobile view. However, I will focus on actually developing the mobile view towards the end of the project as it is not a necessity, so if the project becomes delayed, it is the main feature I will have to postpone or potentially not include.

Simultaneously, since at this stage the website itself is finished in development, I will review the whole project to ensure that I achieved everything that I initially wanted to and give a reason as to why I didn't include certain features or added other features that weren't initially planned.

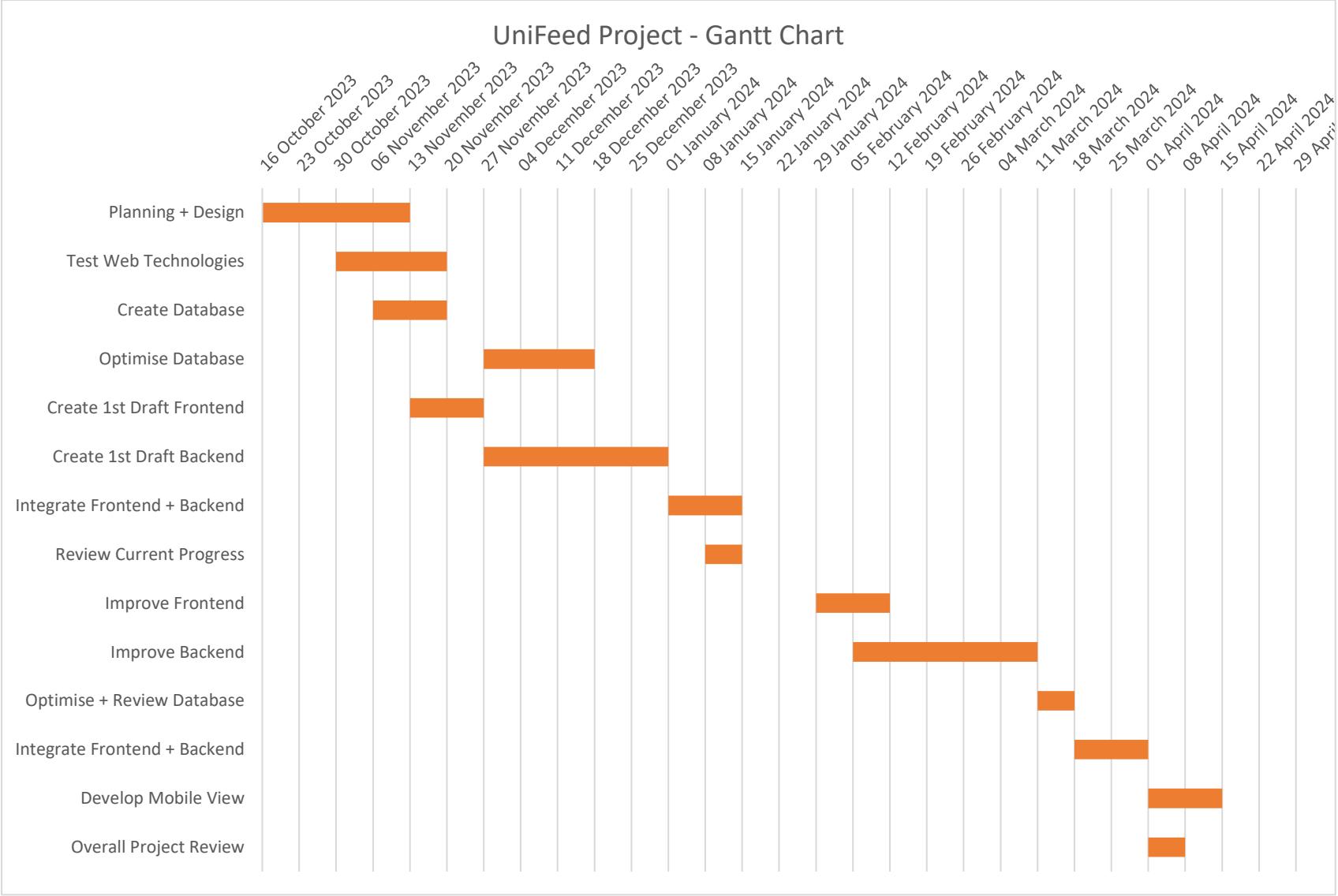

Figure 1.1: Gantt chart showing my planned project timeline

Chapter 2

Literature Review

Contents

2.1 Introduction

In this section, I will demonstrate the research I have conducted in multiple aspects relevant to my project. This research will then be analysed with respect to my project, to assist with the initial plan and design, and carry onto the development phase as I develop the project.

In this chapter, I will outline all the popular existing solutions that university students currently use to connect with others at university, and find the flaws and gaps in these solutions, which the flaws and gaps will be analysed to evaluate how I can provide solutions to these issues in my website.

2.2 Miniature Gap Analysis

I will be conducting a miniature gap analysis, as through both my research and knowledge as a university student myself, I have concluded that Facebook is the only significant existing solution to the problem that I am attempting to solve. Therefore, I will only do a gap analysis on Facebook.

2.2.1 Criteria to be Assessed

Here are the criteria that I will use, to assess the existing solutions and determine the pros and cons of each solution.

1. Students associate the solution as a platform that can be used to create friends at their university.
2. This solution has sufficient features that help students meet other students.
3. It is clear and easy for students to use this solution to connect with other students at the same university.

2.2.2 Facebook Gap Analysis

Facebook is a social networking site, that was created for Harvard University students (February 2004), and later expanded to allow membership to students at other North American universities. The original purpose of Facebook was to allow students to connect with other students at the same university. Students would sign up to Facebook, using their university's email address as verification, and upload photos of themselves to help connect with other students.

However, as Facebook grew to more and more universities, it eventually became public in 2006 and allowed anyone over the age of 12 to become a member. In October 2007, Facebook grew to 50 million members and hit 500 million members in July 2010. As of the third quarter of 2023, Facebook now has just over 3 billion monthly active users, which also makes it the largest application worldwide.

With such a large number of members, Facebook has clearly shifted from its original purpose of helping university students connect, to now making it easy for family and friends to connect with each other.

Criterion 1 - *Students associate the solution as a platform that can be used to create friends at University.*

For this criterion, Facebook certainly meets it. Whilst Facebook has been mainly associated with being used by people to keep in touch with their family and friends for the past 15 years, it remains the most popular platform used by first-year university students in the UK to meet other students and create friends.

According to the results of a 2008 study that researched first-year students at a chosen university about how they used social media for integrating into the university and socialising, 55% of students were already members of Facebook before starting university and 38% of the remaining 45% of students joined Facebook either just before or after starting university. In addition, of the 55% of

students who were already members of Facebook, 97% said they had more Facebook friends than they did before starting university, along with 75% of respondents stating that they used Facebook more than they did before starting university (Madge, Meek, Wellens & Hooley, 2009).

These percentages highlight that the majority of students from the study used Facebook to meet other students, with only 3% having never used the platform. This is further backed up by the '75% of students used Facebook more after starting university' statistic, which clearly shows that the majority of students used Facebook for university-associated reasons, whether it is because they made Facebook friends and continued to use the platform to keep in touch with those friends, and/or used the Facebook groups for information about their halls of residence, societies, course and sports club.

Criterion 2 - *This solution has sufficient features that help students meet other students.*

Facebook only achieves this criterion to a small extent, as upon assessing the platform's capability to allow first-year students to facilitate connections with other students, it is apparent that Facebook is inadequate at providing students with optimal features that are essential for the process of helping students meet and form friendships at university.

On the positive side, Facebook as a platform does have some positive attributes that aid in easing the process of students making friends. The user profiles and friend request features, provide a baseline for students to initiate connections and friendships with other students. Additionally, Facebook Groups is a feature that allows for a dedicated space for students to be in a virtual environment with the other students at the university. These groups can further be harnessed as students can make more specific groups for all the university courses, halls of residence, sports clubs, and societies, allowing students to narrow down the number of members in a group from in the thousands (e.g. a group for Loughborough first-year students) to in the double/triple digits (e.g. a group for all Economics students). These 'subgroups' with fewer members make it easier and less intimidating for a first-year student to reach out to other students by sending friend requests and/or sending an 'about me' message to the whole group.

On the contrary, the extent to which Facebook meets the criterion is limited. The positive features offered by Facebook are mostly lacklustre, on top of the fact that the platform is also missing crucial features and aspects. Facebook Groups are groups that can be created by any Facebook member, with no verification method to authenticate these groups. This means that someone could create an imposter group named '2024/25 Loughborough University freshers', and students about to start their first year might join this group. On top of that, the current method for a student to become members of all the groups they desire (e.g. Loughborough University Freshers group, Computer Science group, Robert Bakewell group, etc.) is to first join their university's main group for all students, and then post a message in the group asking to be added to each of the specific groups that they desire. This is inefficient, and more importantly, frustrating, and annoying for students, as a significant number of students are nervous and aren't fond of the idea of sending a message publicly to a group with thousands of members, alongside the fact that this method is not at all clear and obvious to new students. As a student myself, I only discovered this by having the initial initiative to join Loughborough's main group and then once I was a member, I saw other students post these messages.

In conclusion, it is clear that the features offered by Facebook are inadequate for providing students with an optimal environment to make the friendship-making process easy and stress-free. Facebook's

lack of verification for groups, along with a lack of critical group features (such as links to join subgroups) makes it frustrating for a student about to start university to meet other students. However, there are some positives to take away from this, the user profiles coupled with the friend request system tie together nicely with groups to allow students to make individual friendships with other students.

Criterion 3 - *It is clear and easy for students to use this solution to connect with other students at the same university.*

Assessing whether Facebook meets the criterion of being clear and easy for first-year university students to use to connect with others and foster friendships involves examining both its positive attributes and potential drawbacks.

One positive attribute of Facebook is its user-friendly interface, that makes it easy for its users to create and customise their own profiles, join their desired groups and make friends through messaging and friend requests. Furthermore, the platform's popularity means that many students are already familiar with the user interface and navigating on the platform beforehand, with 55% of students being members of Facebook before starting university in the 2008 study mentioned earlier (Madge et al., 2009). Facebook has Groups always pinned on the left-hand side of the site, making it easy to access, and in this section, it orders all of your groups by last used, so it is easier to select your most frequently visited groups. There is also a 'Your Feed' option, which shows you posts from every group when a user just wants to receive any new information.

However, these attributes come with flaws. The 'Groups' button being pinned on the left, is clustered with over 20 other pages, such as Marketplace, Events, Memories, etc. This highlights how Facebook Groups is just one of the features that Facebook offers, rather than being a primary feature of the platform. With 38% of students in the 2008 study joining Facebook just before or after starting university, a large proportion of students will be new to the platform, and a cluttered user interface with many different options is both visually unappealing and more importantly, confusing.

It's important to note that the percentage of 38% was taken in 2008 when Facebook was the biggest SNS (Social Networking Site), but since Instagram's creation in 2010, it has slowly gained more and more popularity over Facebook with younger audiences. A recent study conducted in September – October 2023 that surveyed 13 to 17-year-old American teenagers on which social media they use frequently. 59% of these teenagers said Instagram, whilst only 33% said Facebook. This study also compared this to data taken from 2014, where Facebook was used by 71% of teens, compared to 41% who used Instagram [2]. This clearly outlines that Facebook used to be the dominant SNS in the youth, and now the majority of teenagers don't use the platform, which backups my original argument that a significant number of new students aren't familiar with Facebook's overly cluttered user interface, with new features and layout changes being added regularly and therefore overcomplicating it for a new student that just wants to use Facebook to join their university's associated groups to meet others and make friends.

2.3 Technology Research

This section will consist of my research into various technologies and frameworks that will be used in my project.

2.3.1 Client-side Framework

Angular

Angular is a JavaScript framework that is maintained by Google, it has comprehensive tooling and offers a full MVC (Model-View-Controller) architecture. Angular also supports TypeScript (a superset of JavaScript) and it is considered a 'heavyweight' framework, but this also means there is quite a steep learning curve to it.

React

This is a framework that was developed by Facebook and is used to make user interfaces. It utilises a component-based architecture, and with it being the most popular of the frameworks discussed, React has a large community that can be used for support and guidance. Whilst this also has a fair learning curve, it is significantly more lenient than Angular.

Vue

Vue is a JavaScript framework used to constructing user interfaces and single-page applications. It uses a one-way data flow model and overall is relatively simple to learn as it has similarities with HTML and JavaScript. Despite this, Vue.js has a small plugin library size, and a notable number of plugins only have documentation in Mandarin.

Conclusion

Considering the attributes of each client-side framework, React is my chosen framework. This decision is supported by the fact that as React has the largest community out of the three frameworks discussed, it will help me to learn how to use it quickly and easily, as I haven't developed with any client-side framework before. Additionally, it is a good between of Angular and Vue, as it is more lenient to use than Angular, and more plugins and better documentation than Vue.

2.3.2 Server Framework

Express

Express is a web application framework that provides many useful features for web applications in a robust manner. Express supports middleware to ease the processing of requests/responses and synergises well with a front-end framework (e.g. React), enabling an application to be full stack developed entirely in JavaScript.

Django

Django is a high-level Python web framework, renowned for being flexible, scalable, and simple. Django has very detailed documentation, along with a large community. This framework also contains strong security features that protect against many security threats, such as cross-site scripting, SQL injection and session security. Django also has great scalability for handling large amounts of site traffic.

Ruby on Rails

Ruby on Rails (also known as just 'Rails'), is a full-stack web application framework, which is written in the language Ruby. This framework follows two principles: DRY (don't repeat yourself) and CoC (convention over configuration), which both work hand-in-hand to assist the development process by increasing code efficiency and simplifying coding decisions, respectively.

Conclusion

In conclusion, Django is my chosen framework. This is due to several reasons, including its strong security features that will help improve the security and robustness of my website, as well as the vast number of components available within the framework. Django also has great scalability, which is great for my project as if I make it public, the more popular my website gets (i.e. the more traffic it receives), Django will be able to scale easily to handle the demand.

2.3.3 Database Types

There are two main types of databases to choose from, to use for my website that are applicable, with the choice being between SQL and NoSQL databases.

An SQL database is a relational database management system that requires data to be stored in tables, with the tables having column headings to indicate what the data below it refers to (e.g. date of birth), and rows that link all the information in one row together. Due to the table format, SQL databases need a predefined schema. A key advantage of SQL databases is that they are ACID compliant, to help ensure that all data transactions are smooth and reliable.

A NoSQL database, as the name suggests, is a non-relational database system that mainly handles unstructured or semi-structured data. The benefit of this is that there is much greater flexibility in the data that can be stored in a NoSQL database. This also comes with a drawback as the data formatting isn't strict, so there may be temporary discrepancies and different nodes in the database might reply with data that is different to what another node would respond with.

After evaluating the advantages of both SQL and NoSQL databases, I have chosen to use an SQL database. The structured data and relationships between entities are well-suited for my website, to link user account information with their posts, comments, etc. In addition, the ACID properties ensure that all transactions within the database are consistent and reliable. The partial con of SQL databases requiring to be organised with a predefined schema, actually has a benefit to it, as it forces me to structure the database tables towards the start, rather than leaving it unstructured and dealing with it later, like in a NoSQL database.

2.3.4 Database Choices

MySQL

MySQL is an open-source RDBMS (Relational Database Management System). It is owned and managed by Oracle, the third-largest software company. MySQL is also the most popular SQL database management system worldwide. Key features of MySQL are that it is very robust and scalable (easily sufficient in size for my requirements), has fast read/write speeds, and is compatible with many programming languages and operating systems.

Microsoft's SQL Server

This is another popular RDBMS, owned and managed by Microsoft. A unique feature of SQL Server is that it is entirely cloud-based and cannot be used as a production server. As it is a cloud database system, it auto-downloads any available updates automatically. Overall, it is easy to use with all the industry standard features.

PostgreSQL

This is the youngest of all the database management systems I have mentioned, being published in 1997. The main benefit compared to the other DBMSs is that it has the most features, whilst it is still reliable and fast. PostgreSQL also stands out in terms of its extensibility and customisation, as it allows users to create their own functions and data types. Despite all these advantages, PostgreSQL doesn't have good quality documentation, in comparison to the other database management systems, so debugging issues might be tough.

Conclusion

Taking into consideration the characteristics of each DBMS discussed, I decided to use MySQL. This is because it is very reliable, has great documentation and is very robust. I don't require the additional features and customisation options available with PostgreSQL, at the expense of poor documentation making the development phase harder.

2.4 Conclusion

I have conducted this literature review thoroughly and it has provided me with valuable guidance on how to proceed with this project. Prior to undertaking this project, I had little knowledge and experience of JavaScript frameworks and modules, and therefore had to do a significant amount of research into the most popular technologies to learn the benefits and drawbacks of each in order to identify which is most suitable for my project.

Whilst my personal experience of having to use SNS to meet other students and eventually struggle to make friends at university as a student myself was motivation for choosing this project, the gap analysis helped me to discover why the existing solutions are subpar at solving the problem at hand from a third-person perspective. Of all the components required for this project, I considered myself to be the strongest in the database aspect of it. In the past, I have previously used Microsoft SQL Server, SQLite, MongoDB, and MySQL, and from that experience, I thought the database decision would be straightforward to just select SQL Server as I thought it was sufficient enough to handle all my requirements. SQL Server indeed has the capabilities I desired, but this literature review eventually made me discover that MySQL would make the development process easier, as well as give me a better final product with its fast read/write speeds.

Chapter 3

Specification

Contents

3.1 Introduction

As this is the specification section, I will clearly outline all of the requirements and expected results for the project. This starts with a project scope to help segment the rest of the stages of the project. I will refer to the requirements defined in this section, in the stages after my implementation, with those being the testing and evaluation section.

3.2 Project Scope

The scope of the project consists of the following four separate sections:

- **Requirements** – A thorough and concise set of requirements will be created, based on the features and functionality mentioned in the literature review. Each individual requirement will become a user story, with the user stories/requirements being categorised by its page.

- **Design** – Wireframes will be drafted for each web page/view in order to make the front-end development process smoother and faster. Each wireframe will then be presented to a small

set of users to perform user testing on the designs, to be able to eliminate any potential weaknesses and overall improve the website's appearance and design.

- **Implementation** – In this section, the website will be developed on both the front end and back end. This development will follow the wireframe designs created in the design section, as well as utilise the user stories from the requirements section to determine the order in which I implement each requirement. I will also make use of the web technologies, frameworks, and libraries discussed in the literature review.

- **Testing** – Rigorous testing will be conducted to identify any issues or detects in the code so that they can be fixed before the product is finalised.

- **Evaluation** – The entire project will be evaluated against the initial set of requirements, to identify the successes and failures of the final project, with an overall conclusion being drawn based on this. The potential next steps for future development/improvement will be stated, with appropriate justifications being provided.

3.3 Requirements

In this section, I will state the requirements of this project in three tiers.

- **Minimum** requirements – the bare minimum that is needed for the final product to be useful and at least partially solve the problem stated in my problem domain.
- **Core** requirements – these are requirements that ensure the platform sufficiently solves the problem at hand and is overall functional and easy to use.
- **Additional** requirements – additional features/functionality to further ensure that the product is different to the existing solutions.

Minimum:

- Users are able to join their university's respective group, as well as sub-groups for their course, halls of residence, as well as the sports and societies they are members of.
- Users are able to send messages to the other members in a group via the chat, and group admins are able to post upcoming events and group-related information in the group feed.

Core:

- Users are able to send other users friend requests and send direct messages to their friends.
- Users are able to receive notifications for friend requests, along with new posts in group feeds/chats, and they are able to customise which notifications they receive and don't receive.
- When requesting to join a group, users are able to submit messages, including images, to

Additional:

- Users are able to create their own groups and sub-groups and is able to customise their group to their liking.
- Users are able to receive notifications via email.
- When users create their account, their university email address is officially verified instead of just verifying the email domain.
- Enhance the messaging functionality, both direct messages and group chat messages, by allowing users to send links that navigate to other site content, such as a specific event post, or another chat.

3.4 User Stories

This section will outline the user stories for each page/view of the website, which can also be used as a more specific and detailed version of the requirements. I used Trello to log all of these requirements onto a Kanban board, with each requirement being tagged with its associated page, along with its tier of requirement (minimum, core and additional).

3.4.1 Login/Sign Up Page

This page should allow the user to either login to their existing account by entering their email address and password or allow the user to sign up and create an account.

Functionality:

- Users can login to their account by entering their email address and corresponding password.
- Users are able to sign up and create an account, which takes them to an account creation menu, where they have to enter in required information, as well as being able to enter in any optional information. All information must be verified before a new account can be created.
- Users are able to click on a link to navigate between the login and sign up pages.

3.4.2 Home Page

This page should act as the main feed page for a user to view the latest posts both for their university, and for the groups that they are members of.

Functionality:

- Users are able to view the feed of messages and events posted in the main university group.

- Users are able to view the feed of messages and events posted in the groups that they are members of.
- Users are able to scroll through the feeds to view all of the posts.
- Users should be able to click on a post to open the post and view the contents of the post.

3.4.3 Groups Page

This page should allow the users to view all of the groups that are available to them and give them the option to view each group and join it.

Functionality:

- Users should be able to see all of the groups that are available to them, in a neat and organised list.
- Users should be able to filter the list of groups based on group type.
- Users should be able to click onto a group to view more information about the group, as well as have the option to join that group.

3.4.4 My Groups Page

This page should allow the users to view all of the groups that they are members of and access each individual group.

Functionality:

- Users should be able to see all of the groups that they are members of, in a presentable manner.
- Groups should have a feed channel that only group admins can post messages in, a group chat channel that any group member that any group member can post messages in.
- Users should be able to click on another user to see their profile.

3.4.4 Messages Page

This page allows users to view their messages from their friends and other students, as well as sending messages.

Functionality:

- Users are able to view all of the messaging chats they have had with their friends.
- Users are able to send messages to their friends and read messages from their friends.

3.4.5 Profile Page

This page allows users to view their profile and customise it, as well as access their account settings.

Functionality:

- Users are able to preview their profile that is displayed to other users.
- Users are able to modify and customise the content displayed on their profile.
- Users are able to view and modify their account settings.

3.5 Site Map

Figure 3.1 is a site map of the website, highlighting the individual web pages and demonstrating on a simple level, how the navigation flows between those web pages. Once a user is logged in, all pages can be accessed at any time via the navigation bar.

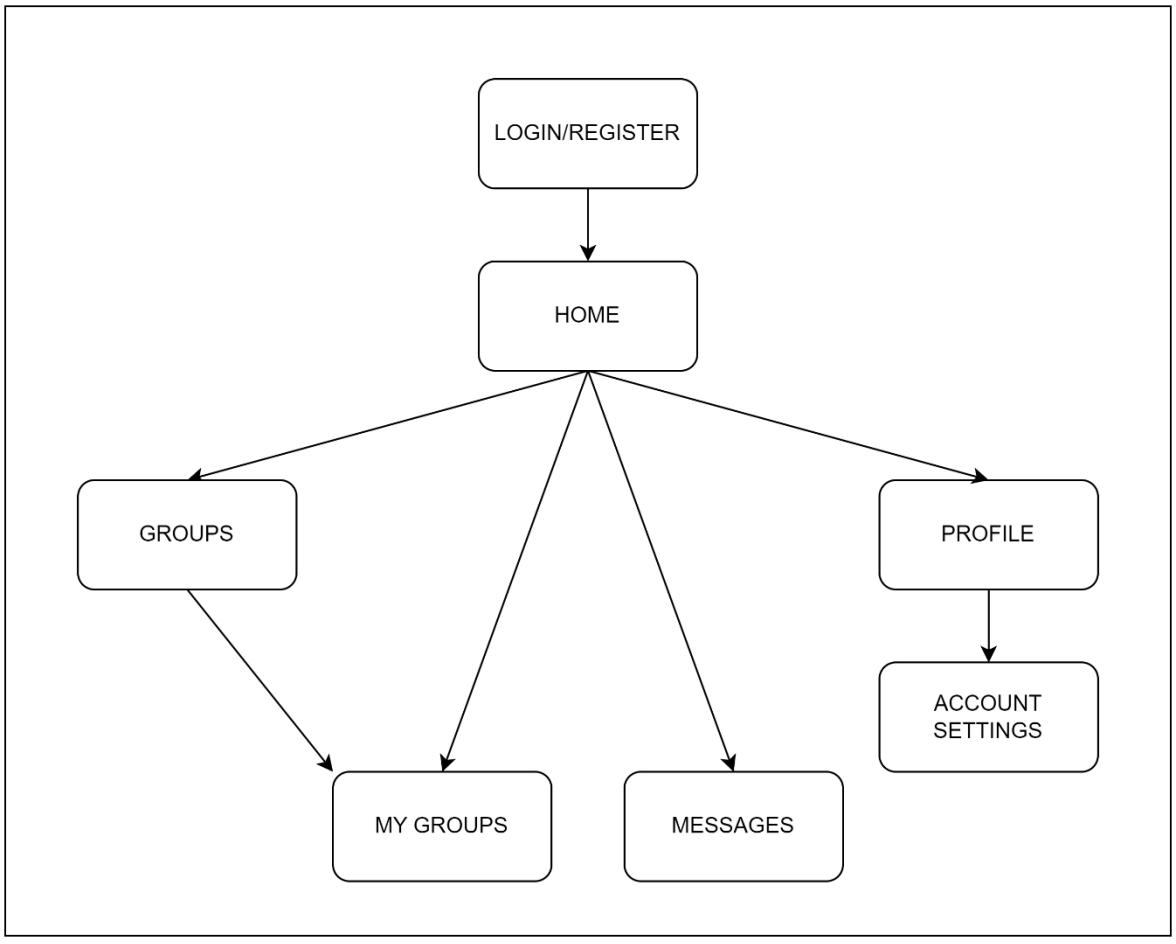

Figure 3.1: A simple site map for the web application

3.6 Conclusion

In this section, I stated all the requirements and expectations for the rest of the project. This information will be used in the design section to help ensure that the wireframe prototypes that I design capture all the requirements that I planned to meet, of which the prototypes will be used during the implementation stage.

In addition, I am going to use this section to create testing tables in my testing section, to process if my final product passed or failed each step of my specification.

Chapter 4

Methodology & Planning

Contents

4.1 Development Models

In this section, I will evaluate three distinct development models: Waterfall, V-Model, and Extreme Programming, focusing on their features, as well as highlighting their advantages and disadvantages, in relation to my project. I will use this to assess and select the most appropriate model to use for the implementation stage.

4.1.1 Waterfall

The Waterfall model is the most traditional software development model, consisting of a sequential series of stages that are to be completed in order. These stages are requirements analysis, specification, design, implementation, testing and deployment.

Advantages:

- The linear, chronological path provides a clear framework, with most (if not all) of the planning being done before any implementation of the project has even started, streamlining the implementation with clarity and guidance.
- Each stage of the process requires thorough documentation, which ensures that all the necessary information and details of the project are well-documented and accurate.
- The waterfall methodology allows for more precise planning and execution for projects with stable requirements that aren't likely to change over the development duration of the project.

Disadvantages:

- It is inefficient and time consuming to revisit and modify an earlier stage of the project, meaning that this method isn't suitable for projects with insecure requirements, or longer projects.
- As the requirements are defined very early into the project, it is likely that the requirements don't remain completely appropriate towards the end of the project's development, due to any unforeseen issues.

4.1.2 V-Process

The V-Process model, also known as the Verification and Validation model, enhances the traditional Waterfall model. Just like the waterfall model, it follows the same set of steps that I mentioned above, in chronological order. It adds to the waterfall model by integrating testing phases for each stage of development. The V-Process model emphasises a systematic approach to every stage of development.

Advantages:

- The model has quality assurance integrated into it, which will increase the final product's quality and reliability.
- The early testing phases will help to reduce the overall time spent by identifying problems before coding starts.

Disadvantages:

- This model shares the primary weaknesses of the waterfall model, with its lack of adaptability to change the project requirements.

4.1.3 Extreme Programming

Extreme programming (XP) is an agile framework that emphasises on iterative development with frequent releases, with the goal of achieving high levels of customer involvement over the course of the project's development, in order to adapt to changes in requirements quickly.

Advantages:

- XP has high flexibility to adjust and accommodate for any changes at any time during the project.
- The continuous feedback from customers ensures that the product meets the users' requirements.

Disadvantages:

- The main disadvantage of extreme programming for my project, is that it requires consistent involvement from customers, which also relies on the customers feedback to hopefully represent the majority of my target audience.
- The scope of the project can increase rapidly and uncontrollably based on the customer's feedback if are constantly asking for more features.

4.1.4 Kanban

Kanban is another agile framework, unique with its highly visual workflow management method, created by Taiichi Ohno, an engineer at Toyota, to improve their manufacturing efficiency. In software development, Kanban helps teams visualise their workflow to limit the number of work-in-progress tasks active at once to maximise efficiency. This is achieved through Kanban boards, that display the tasks in a project, organised by the current stage of each task, with colour coding to highlight the person assigned to each task. However, in a solo project, I can instead use the colour coding to tag the tasks and categorise them by types.

Figure 4.1: Example of a Kanban Board

Advantages:

- The visualisation makes it easy to view the status of each piece of work, to help track progress.
- Limits can be set for the number of tasks in each column at once, to prevent multitasking and overcommitment, maintaining the focus on completing the active tasks first.
- The methodology is simple and easy to use, making the methodology easy to implement and follow without the use for training or any complex tools.

Disadvantages:

- Kanban relies on personal discipline to regularly update the board to account for the latest progress.
- It is easy to neglect long-term planning, as the focus of Kanban is to segment the project into small tasks and complete the tasks one by one, which doesn't provide good enough insight for future phases.

This is the model I decided was the most appropriate for the project, as the methodology is easy to follow, and has little downside for being used for a solo project, other than the self-discipline required to frequently update the Kanban board. The visualisation aspect provides great clarity on the current progress of the project, as over time the backlog of tasks will reduce and the number of completed tasks will increase, effectively showing me how far I am through the implementation of the project at a glance.

4.2 Tools and Environments

In this section, I am going to outline the various tools and environments that I have chosen to use for the implementation stage of my project, as well as justifying my reasoning for each choice.

4.2.1 Integrated Development Environment / Text Editor

I have chosen to use a text editor instead of an integrated development environment (IDE), with that text editor being Microsoft's Visual Studio Code (VS Code). VS Code will be the sole text editor used throughout the development of the project.

VS Code is a highly versatile and widely acclaimed text editor, offering comprehensive support for a wide range of both programming languages and technologies, making it ideal for developing my web-based application. It has a lightweight design and built-in support for web development, including a terminal, debug console, and a large library of community extensions that can be installed to assist my coding. This is the primary reason I chose VS Code, as the extension marketplace enhances what VS Code is out-of-the-box, offering languages, debuggers, and tools to support my development workflow. In addition, this has been my primary text editor that I have used for the past 5 years, so I am experienced with the application, as well as already familiar with many useful extensions that I can utilise for my project.

4.2.2 VS Code Extensions

VS Code has a big, versatile library of extensions, created by companies, open-source communities, and individual developers. My main purpose for using extensions for my project, is to aid in writing code, to ensure I write code neatly and efficiently so that it is easy for other people to read, as well as help with debugging, to overall make the final project be completed quicker, and at a better quality.

Here are the extensions that I used for my project:

Auto Complete Tag

This extension is a pack that contains 2 extensions, Auto Close Tag and Auto Rename Tag. These extensions are both simple but effective extensions that now feel second nature to me, to the point that I find it tedious to write code without them.

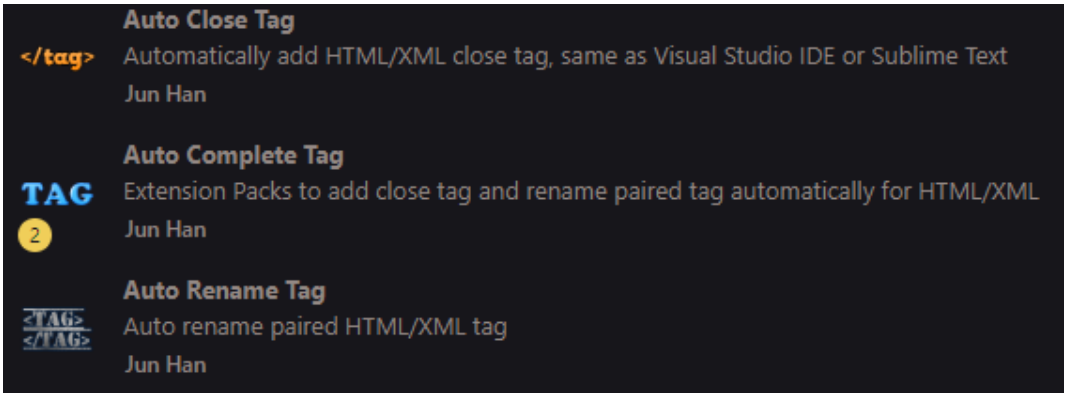

Figure 4.2: Auto Complete Tag Extension

- The Auto Close Tag extension automatically writes a closing tag, whenever I write a tag, both saving me time, as well as reducing errors to ensure that I never have an open tag without a closed tag.
- The Auto Rename Tag extension ensures that if I rename one half of a tag (either the open or close tag), it will according rename the other tag to match it.

Live Server

To run my website, I use the terminal provided in VS Code to localhost my code. This extension changes the localhost to now act as a live server, so any changes I make, are automatically uploaded to the locally hosted website once the file is saved.

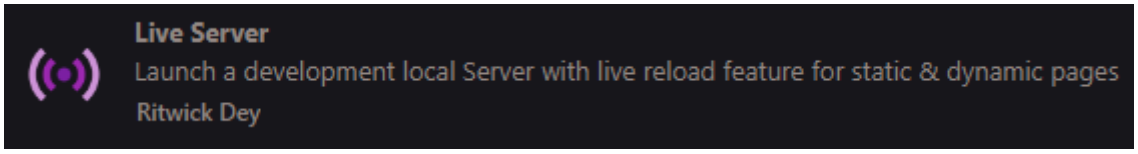

Figure 4.3: Live Server Extension

This extension is incredibly useful, especially for front-end development, as it makes it very easy to instantly see the results of any new code I've written, which helps to do some testing directly after making changes. When developing the front end, this extension encourages me to save my code frequently as I go along, so that on my second monitor, the website instantly refreshes to show the newly implemented code, and if there is anything wrong with the change, I can instantly attempt to resolve it, rather than continuing to write more code, with that code potentially having to be changed as well, depending on the issue.

Prettier – Code formatter

Prettier is an opinionated code formatter, that enforces a consistent style of code by parsing my code and re-formatting it, taking into account the maximum line length, and wrapping lines of code where applicable.

Figure 4.4: Prettier Code Formatter Extension

This extension ensures the tab indent, line wrapping and spacing across my code is both organised optimally and consistently, making the code clear and readable to myself, and others too. The formatter supports JavaScript, TypeScript, JSX (JavaScript with React), which is ideal for my project as I am using JavaScript, along with React.

Oh Lucy

Oh Lucy is a simple extension that allows me to apply a customised colour code and theme to my code. Whilst VS Code out-of-the-box already has its own colour scheme, the separate colours to differentiate code is limited in the colour variation (as the colours are mainly shades of green, blue, yellow). In addition, it adds clear colour boxes for CSS code to show each colour, making it simple to choose and change colours when I style the website.

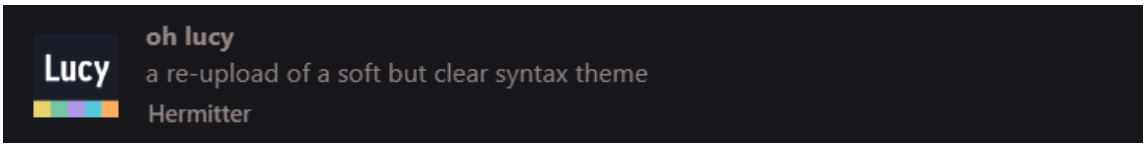

Figure 4.5: Oh Lucy Extension

4.2.3 Kanban Board & Project Management

As mentioned in the specification section, Trello is the tool that I am using for my Kanban board. Trello is a web-based project management tool that is used to organise tasks of a project on a Kanban-styled board, that utilises cards for each task, and each task can be labelled (e.g. the yellow tags are for the database-related cards).

The columns allow me to break up the individual tasks of my project into sprints, and each card is tagged with its sprint number. When I start a new sprint, I drag all of the tasks for that sprint into the ‘To Do’ column, and the sprint is completed when all of those tasks are moved along into the ‘Completed’ column. I split up the workload so that each sprint on average would take 2 weeks to complete, but I didn’t follow this strictly as sometimes tasks would in reality take longer to complete than I had originally anticipated, and some weeks I’d spend longer working on my project, and others I’d spend less (e.g. during the exam period).

Despite this, throughout the development of the project, after each sprint was completed, I compared the actual date I finished each sprint to the date I planned for that sprint to be finished, which helped me to show if I was on track or falling behind.

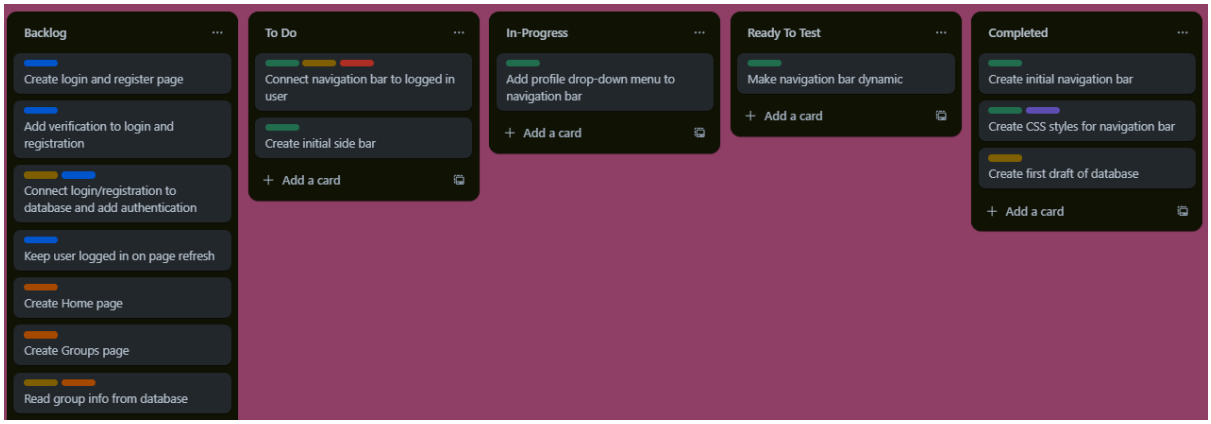

Figure 4.6: Trello 'Kanban' board towards the beginning of development

4.2.4 Database Management Software

MySQL Workbench is a powerful, versatile tool that was made for developers, as well as database architects and administrators. The software provides a robust GUI that simplifies many database tasks, including schema creation, management, and maintenance. This eliminates the need for me to directly deal with raw SQL commands, which is a slow and dull process. According to the official MySQL website, MySQL Workbench provides "advanced data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, and much more" (MySQL, no date).

With regards to my project, a significant benefit of MySQL Workbench is that it is developed by Oracle, who acquired MySQL in 2010. This makes MySQL Workbench the official MySQL software, ensuring seamless compatibility, coupled with robust support for MySQL databases. This is especially useful when it comes to connecting my database to my Django server, as all I need to do in Django is provide the database credentials (database name, user, password, host, and port) and the connection is successfully established.

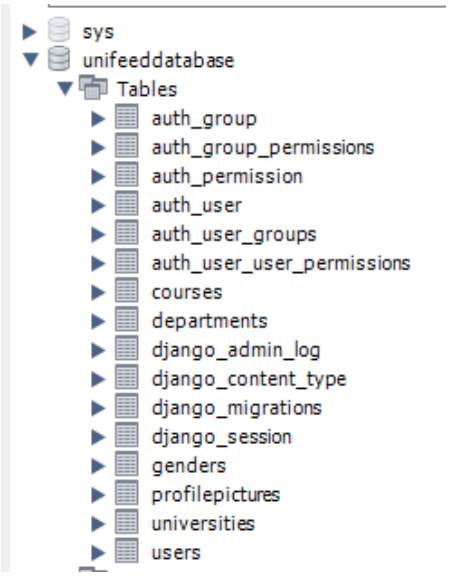

Figure 4.7: Early draft of database in MySQL Workbench (after integrating with Django)

4.2.5 Diagram and Design Software

Draw.io is my chosen software for creating diagrams and designs. Draw.io (also known as diagrams.net) is a web-based application specialised for creating diagrams. As you can see in **Figure 4.4**, this software has a basic layout, where you simply drag in shapes from the left onto the canvas. On the right-hand side, you can customise the shapes by changing their colour, thickness, opacity, etc.

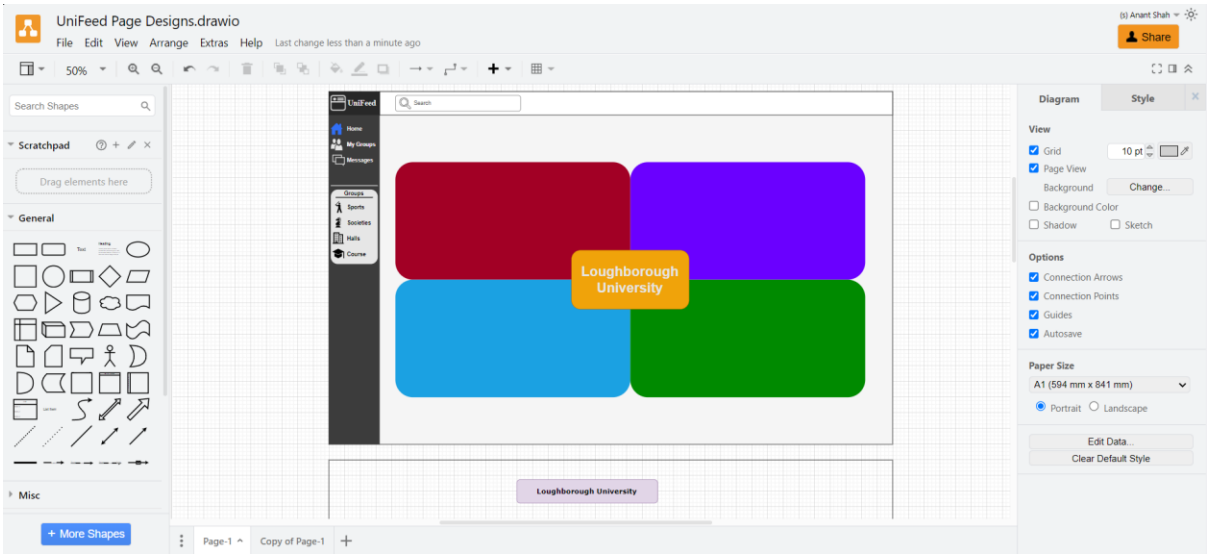

Figure 4.8: My first draft of creating the home page using draw.io

Chapter 5

Design

Contents

5.1 Design Approach

My initial objective for designing the platform, was to create a distinct colour scheme that can be used throughout the pages of the website. Accomplishing this first saved time in designing the wireframes for each page, as I didn't need to dedicate as much time to picking the colours and selecting the ideal hex colour codes. I wanted the colour palette to consist of black, white, several shades of grey, along with a primary blue colour, that would act as the primary colour of the platform. I also chose a darker and lighter tone of this primary blue, to be used when certain attributes of the website are hovered over or active.

For my colour palette, I opted for mainly simple greyscale colours, with a bright blue being the primary accent colour used across the website. There is a slightly darker version of that blue, to be used when various buttons or components are hovered over or clicked. In addition, upon progressing through designing the wireframes, I also realised it would be a nice touch to have a bright bar across the top of each page directly below the navigation bar, to help add more colour and diversity to the website, as using only blue made the website's colour palette appeal slightly muted. These are what the soft and gradient colours are used for.

```
/* Colour Palette for the Website */
:root {
  /* The main greyscale colours used in the website */
  --white: #FFFFFF;
  --black: #000000;
  --light-grey: #E6E6E6;
  --dark-grey-1: #282828;
  --dark-grey-2: #3C3C3C;

  /* Blue is the primary colour used throughout the website, with a
```

```
    darker blue used when hovering over components */
  --primary-blue: #007BFF;
  --dark-blue: #0056B3;

  /* Colours used to create the gradient effect for the title banners */
  --soft-green: #D5E8D4;
  --gradient-green: #97D077;
  --soft-yellow: #FFF2CC;
  --gradient-yellow: #FFD966;
  --soft-pink: #E6D0DE;
  --gradient-pink: #D5739D;
  --soft-red: #F8CECC;
  --gradient-red: #EA6B66;
}
```

5.2 Wireframes

A critical part of the design stage of developing a project, is to create prototypes of what the user interface design will look like for the final product. I am going to be doing this in the form of wireframes. Wireframes are basic blueprints that show the layout of various elements on each web page, whilst abstracting the more detailed elements, such as graphics and colours. The abstraction allows me to focus on constructing the web page structure and placement of components, rather than the aesthetic details.

In section 4.2.5, I mention that draw.io is my chosen application for creating designs, along with demonstrating that the software is simple and straightforward to use. As this software has no complicated, advanced features, it makes it perfect to create the wireframes, because the main focus with constructing wireframes is to apply abstraction and not focus on the smaller details.

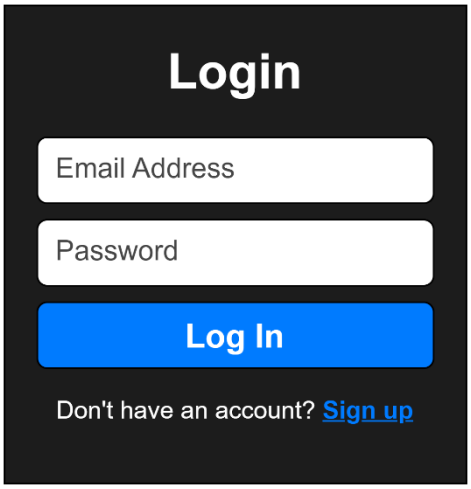

Figure 5.1: Wireframe of Login Page

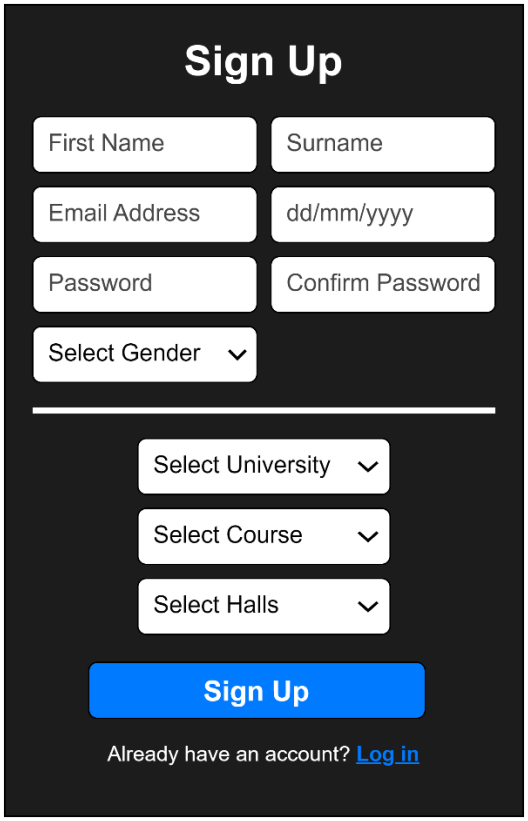

Figure 5.2: Wireframe of Sign Up Page

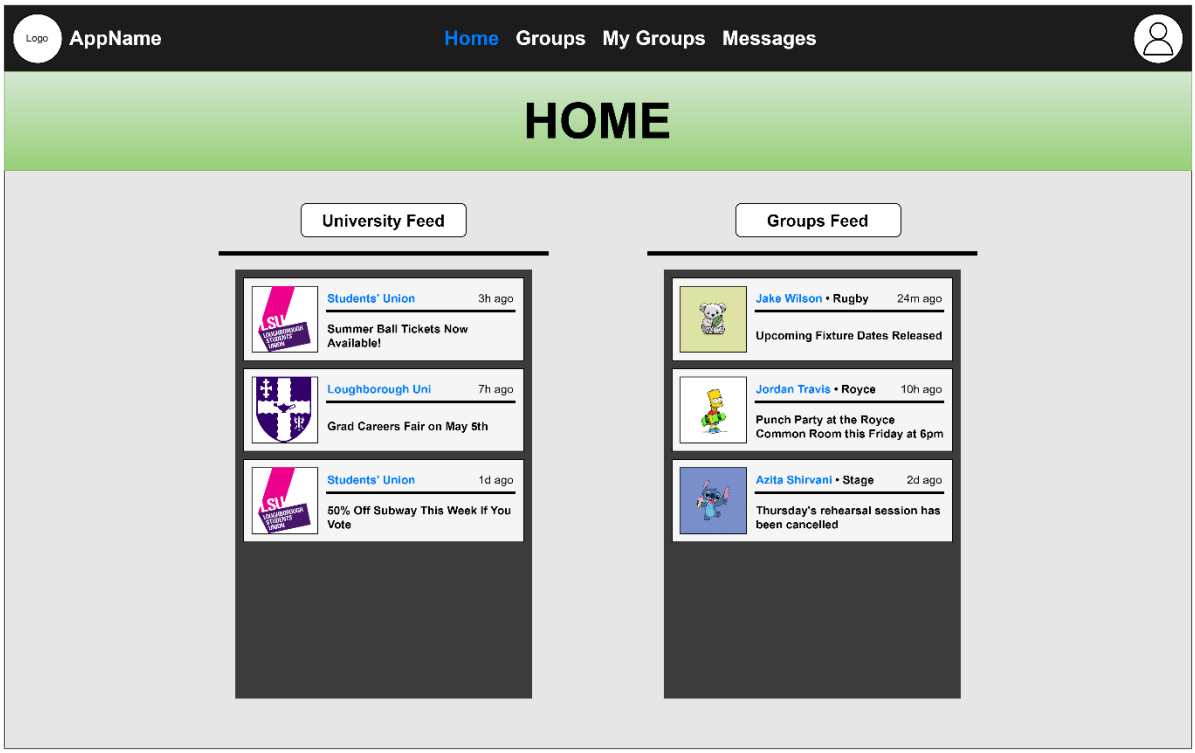

Figure 5.3: Wireframe of Home Page

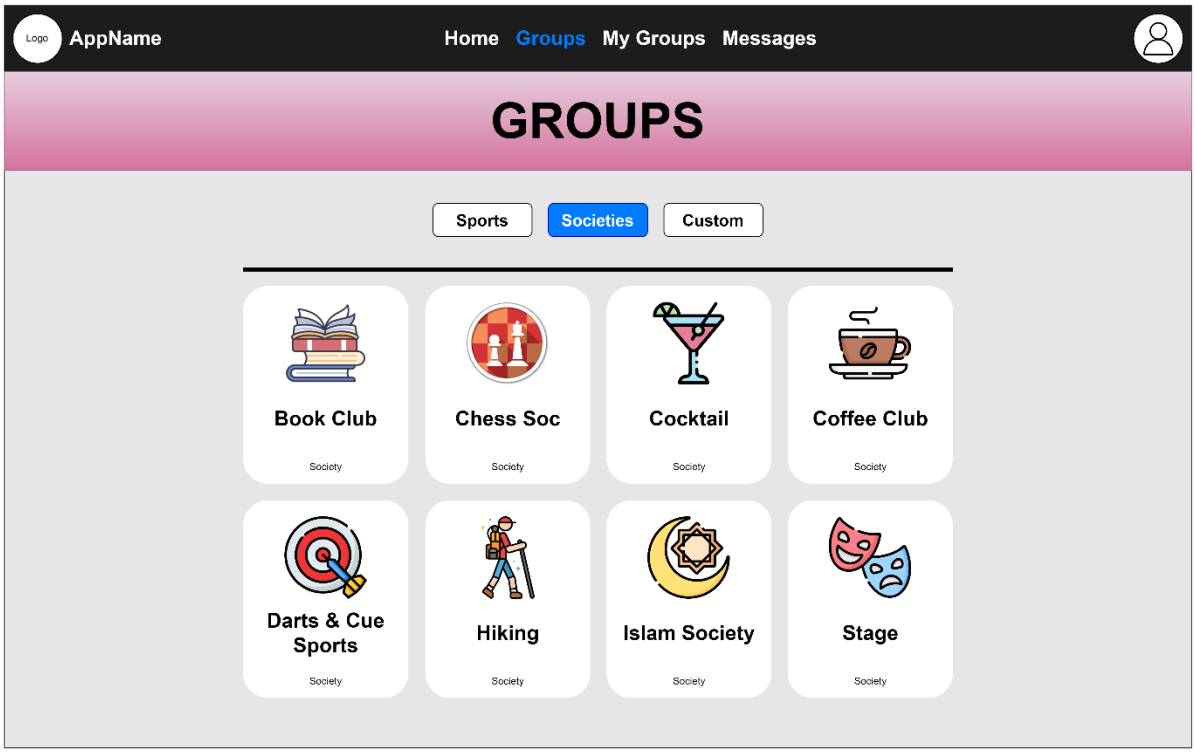

Figure 5.4: Wireframe of Groups Page

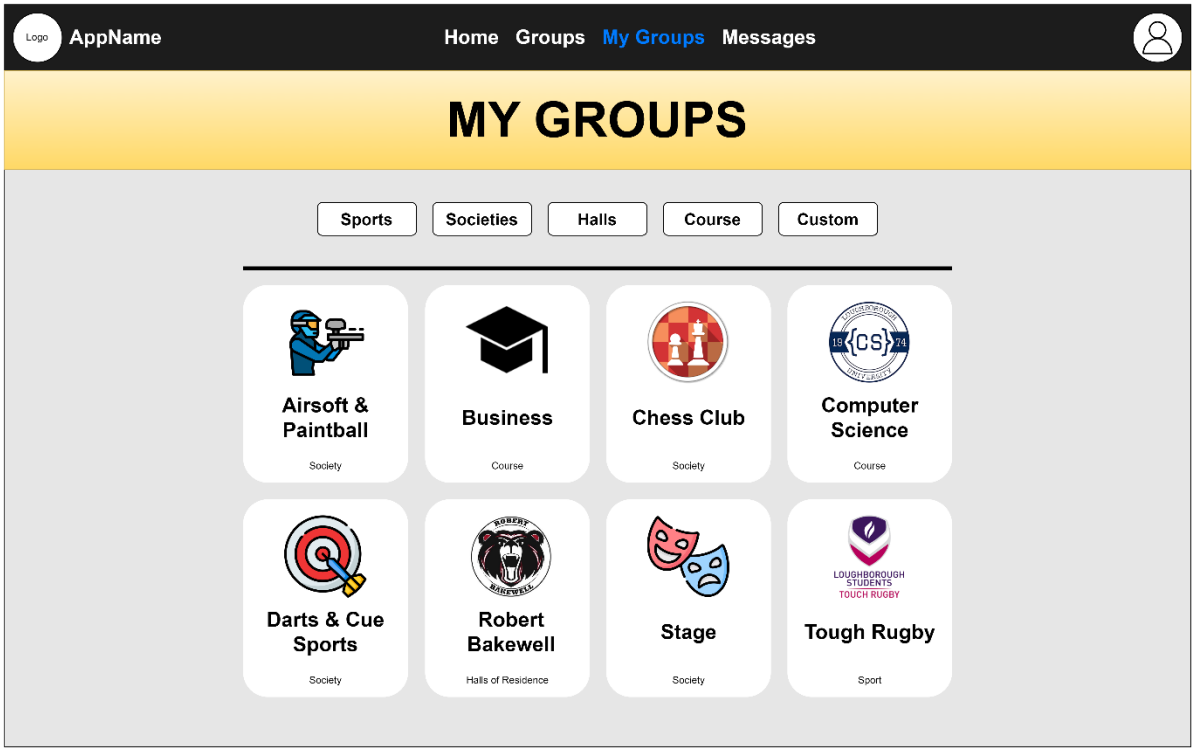

Figure 5.5: Wireframe of the My Groups Page

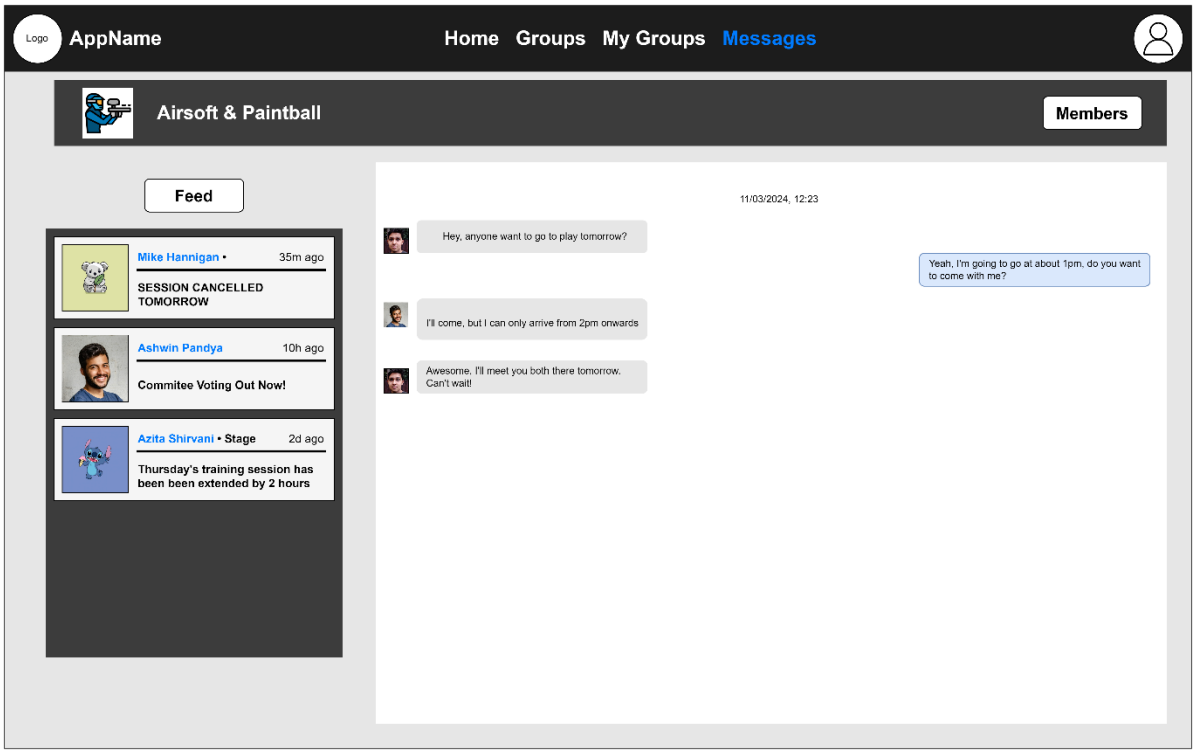

Figure 5.6: Wireframe of Specific Group Page

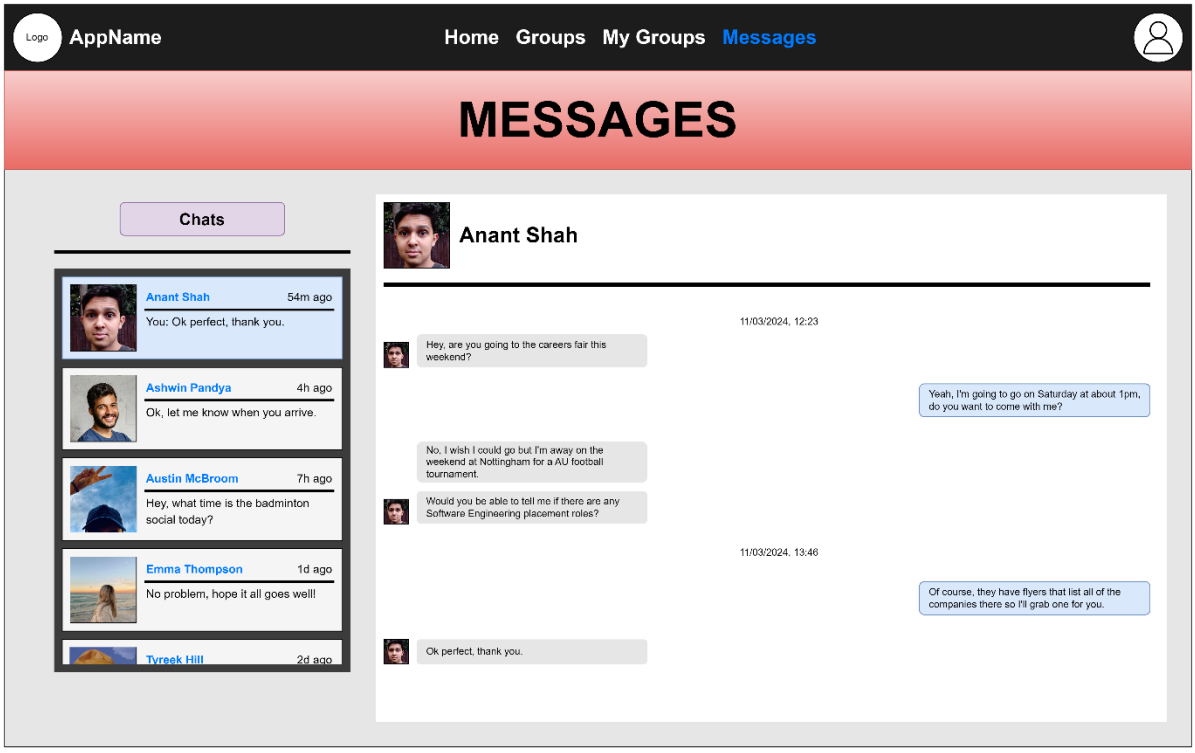

Figure 5.7: Wireframe of Messages Page

Chapter 6

Implementation

Contents

6.1 Setup

To initiate the implementation stage of the project, I first started with the setup to allow me to begin coding.

6.1.1 React with Vite

Project Initialisation:

My first step was to create a React project to development the project locally. I opted to use Vite to create my React project, instead of the more traditional approach of using Create React App. I set this up by following the clear documentation provided in the 'Getting Started' section on Vite's website.

Once I had installed Node.js and React.js, I used the command '***npm create vite@latest***' to initialise a new project. I was then prompted to input the following information:

- **Project name:** UniFeed-Frontend
- **Framework:** React

- **Variant:** TypeScript

Installation:

After providing that information, the react project structure was successfully generated. The next step in the documentation was to navigate into the directory of the project that I just created, with '***cd UniFeed-Frontend***', and then run '***npm install***' to install to install all the necessary dependencies to setup all the node modules required for the package configuration.

Development Environment:

The project was now in a state ready for development. Running '***npm run dev***' starts the Vite development server. Vite's server leverages fast HMR to update the website almost instantaneously without reloading the page, allowing for a highly responsive and interactive development experience.

Building for Production:

Vite also has a dedicated command, '***npm run build***', to prepare the project into a production-ready build for when I want to deploy the production. Vite has a very efficient building process that gives a fast deployment, along with the live application having excellent performance.

6.1.2 React Router

React Router is one of the most popular libraries within React, and it is used to handle the navigation and rendering of components in web applications. Using React Router enables me to deliver the project as a single-page application, which ensures that the pages are not refreshed whilst users navigate across the various components and views of the website. React Router achieves this by utilising dynamic routing, which means that the routing occurs as the app is rendered. This results in quick and effortless transitions from view to view, as well as updating the web URL accordingly based.

Integrating React Router was a straightforward process, as React's documentation is clear and comprehensive. The bulk of the code for React Router resided in my App.tsx file:

```
1  const AuthenticatedRoutes = () => {
2    const { user } = useAuth();
3
4    if (!user) {
5      return <Navigate to="/login" replace />;
6    }
7
8    return (
9      <>
10       <Navbar />
11       <Routes>
```

```
12          <Route path="/home" element={<Home />} />
13          <Route path="/groups" element={<Groups />} />
14          <Route path="/mygroups" element={<MyGroups />} />
15          <Route path="/messages" element={<Messages />} />
16          <Route path="/profile" element={<Profile />} />
17          <Route path="*" element={<PageNotFound />} />
18        </Routes>
19      </>
20    );
21  };
22
23  const App = () => {
24    return (
25      <Router>
26        <AuthProvider>
27          <Routes>
28            <Route path="/login" element={<Login />} />
29            <Route path="/signup" element={<Signup />} />
30            <Route path="*" element={<AuthenticatedRoutes />} />
31          </Routes>
32        </AuthProvider>
33      </Router>
34    );
35  };
```

In this code, App is the default function, with AuthProvider being a custom function that I created to ensure that the user can only navigate to the main routes of the website if they are logged in. The only routes provided are the login and signup routes, with all other routes being re-directed to the AuthenticatedRoutes function, which first authenticates if the user is currently logged in or not before accessing the rest of the router routes. Both the AuthProvider and useAuth functions are located in the Authentication.tsx file, which I show below in section 6.3.1.

With React Router implemented in my code, I can now integrate Link elements into all the button clicks on the pages of the website, mainly on the navigation bar, which are associated with navigation to other areas of the application.

6.1.3 Django with Django REST Framework

In my literature review, I established that I would be using Django as the server framework for managing all backend requirements of my project. Django serves as a robust and scalable server-side framework that manages HTTP requests, data interactions and application logic.

For my project, Django's primary role serves as acting as an interface to the database through its ORM (Object-Relational Mapping) system, which abstracts database operations into higher-level (Python) code, eliminating the need to write raw SQL. This setup allows for secure and efficient data transactions between the backend and the database. In addition, ORM's high-level syntax makes the code easier to write, which speeds up the rate of development, as well as making it simpler to debug and read.

On top of that, Django REST Framework is a powerful enhancement to standard Django, by providing a flexible toolkit for building Web APIs, which I will use for communication between the frontend and backend (Django REST Framework, no date). The REST framework has request parsing, which simplifies request data handling, allowing for easy parsing of various content types. It also has built-in support for many authentication schemes (such as TokenAuthentication,OAuth2), ensuring that no unauthorised users can access any data via the API.

Establishing Connection to the MySQL Database

I followed the installation process on the Django REST framework website to successfully install Django with Django REST Framework. After that, I configured the settings.py file to enable the connection to the MySQL database.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'UniFeedDatabase',
    'USER': 'root',
    'PASSWORD': 'password',
    'HOST': 'localhost',
    'PORT': '3306',
  }
}
```

Whilst the connection to my MySQL database was established, I still needed to have Django models to reflect the structure of my database tables, as this is necessary for me to be able to use Django's ORM full set of capabilities. Instead of writing the models myself, I utilised the command line command '***python manage.py inspectdb > models.py***' to automatically generate the models from my existing database into a models.py file within my project directory.

Django Model for Departments Table

```
class Departments(models.Model):
    departmentid = models.AutoField(db_column='DepartmentID',
primary_key=True)  # Field name made lowercase.
    departmentname = models.CharField(db_column='DepartmentName',
max_length=100)  # Field name made lowercase.
    universityid = models.ForeignKey('Universities', models.DO_NOTHING,
db_column='UniversityID', blank=True, null=True)  # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'departments'
```

Once the models were created, I then defined serialisers for each model in a '***serializers.py***' file. Serialisers are necessary to translate the Django models into a format that can easily be converted into JSON, so that it can be used for API responses to the client-side React server.

Serialiser for the Model of the Department Table

```
class DepartmentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Departments
        fields = '__all__'
```

The next step in using the Django REST framework to process the data received from the database, is to create view sets in order to provide a standard set of CRUD operations for my models. This standardised structure allows my web application to handle user inputs along with store and retrieve data across the server-side, client-side and database system within my project. This creates smooth communication and interaction between the components that construct my web application.

View Set for the Department Serialiser

```
class DepartmentViewSet(viewsets.ModelViewSet):
    queryset = Departments.objects.all()
    serializer_class = DepartmentSerializer
```

The final step on the Django side is to create URL routes. The URL routes are in relation to all the pages of my website, and provide a set of instructions telling Django what to do whenever a user visits each page.

In relation to my project, I make use of URL routes to handle database requests to fetch the necessary data from the database that is needed as part of loading each page. For example, on the messages page, the URL routes fetches the chats and messages for that user.

URL Routes

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import CourseViewSet, DepartmentViewSet, UserViewSet,
UniversityViewSet, GenderViewSet, ProfilePictureViewSet

router = DefaultRouter()
router.register(r'courses', CourseViewSet)
router.register(r'departments', DepartmentViewSet)
router.register(r'users', UserViewSet)
router.register(r'universities', UniversityViewSet)
router.register(r'genders', GenderViewSet)
```

```
router.register(r'profile_pictures', ProfilePictureViewSet)

urlpatterns = [
    ##All API endpoints are prefixed with /api/
    path('api/', include(router.urls)),
]
```

Connecting React to Django with API Requests

I also updated the Vite configuration file (vite.config.ts) to configure Vite to proxy API requests to Django, so that I can forward requests from the frontend React server, on port 3000, to the backend Django server, on port 8000.

```
export default defineConfig({
  plugins: [react()],
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:8000', // Django is running on port 8000
        changeOrigin: true,
        secure: false,
        ws: true,
        rewrite: (path) => path.replace(/^\/api/, '')
      }
    }
  }
```

6.2 Features

This section demonstrates each of the features that I implemented in my final product, and explains how I implemented each feature, including any libraries or tools used. I also provide screenshots showing the end result of each feature.

6.2.1 Navigation Bar

User Interface

The first step to creating the navigation bar was to create the user interface. Following the design I made, as shown in Figure 5.3 in the Design section, the navigation bar was dark grey bar that spanned across the top of the page, with three distinct sections: the left section that displayed the app name and logo; the middle section that provides the primary navigation, with buttons to the four main pages of the website (Home, Groups, My Groups and Messages); and the right section, showing the user's profile picture and also acting as a button to access the profile page.

These three sections are reflected in my Navbar function, as inside the Nav element, there are three division elements with the respective classes of ‘left-section’, ‘middle-section’ and ‘right-section’, with those classes having the necessary CSS styling to ensure that each section is positioned correctly on the navigation bar.

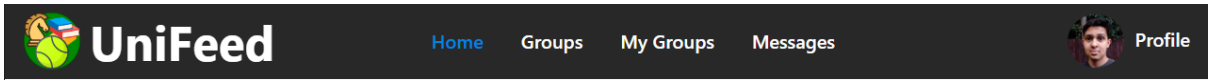

Figure 6.1: View of the Navigation Bar, with the active page ‘Home’ highlighted in blue

Navigation and Linking

Once the user interface for the navigation bar was created, the next task was to add functionality to it so that the components on the bar take users to the chosen page when users click on them. To start this, I imported the Link, useMatch and useResolvedPath utilities from the React Router DOM library (React Router DOM is React’s specialised package for web-based applications).

The ‘Link’ component is the central component used throughout the Navbar function to navigation links that change the route path to a specific page, without causing a complete page reload. Every component on the navigation bar is inside a Link element, with the app name and logo taking users back to the home page and the central navigation links taking users to the page they describe. For the main navigation links in the centre of the bar, I created a custom function called ‘NavLink’.

This function serves two purposes. The first purpose is that it re-uses code by shortening the list elements of the unordered list from four lines to one line per list element, increasing durability and efficiency. However, the major benefit is that it makes the list elements dynamic. I use this by dynamically updating the class name of the link so that if a link is pressed, it will update the class name, changing the colour of the text on the Navbar for that link to highlight the current page that the user is on.

```
1  export function NavLink({ to, children, ...props }: Props) {
2    const resolvedPath = useResolvedPath(to);
3    const isActive = useMatch({ path: resolvedPath.pathname, end: true });
4    return (
5      <li>
6        <Link to={to} className={isActive ? "active" : ""} {...props}>
7          {children}
8        </Link>
9      </li>
10   );
11 }
12
13 //Defines the interface for Props that are used in the NavLink function
14 interface Props {
15   to: string;
16   children: React.ReactNode;
17 }
```

Profile Drop-Down Menu

Unlike the rest of the components, clicking on the profile text or the profile picture won't navigate directly to the profile page. Instead, it opens a drop-down menu. This menu displays the user's full name, along with two buttons, one to take the user to the profile page, and the other logging the user out and taking them to the login page.

I created the drop-down menu by originally creating the menu user interface, but setting the max height to 0 and the overflow to hidden (as highlighted in the code snippet below). I then use in-line JavaScript to toggle the class name assigned to the profile drop-down div element, toggling the height of the menu, which opens and closes the menu.

Profile Div

```
8     const [isOpen, setIsOpen] = useState(false);
9     // To toggle the dropdown menu
10    const toggleDropdown = () => {
11      setIsOpen(!isOpen);
12    };

...

29 <div className="profile" onClick={toggleDropdown}>
30    <img src="/src/images/Profile-Picture.jpg" alt="Profile" />
31    <h2>Profile</h2>
32    <div className={`drop-down-wrap ${isOpen ? "open-menu" : ""}`}>
33      <div className="drop-down">
```

Drop-down Menu CSS

```
1 .navbar .profile .drop-down-wrap {
2     position: absolute;
3     top: 100%;
4     right: 0%;
5     width: 250px;
6     /* Max height of 0 coupled with the overflow being hidden keeps the drop-
7        down menu hidden, and when clicked, the max height will be changed. */
8     max-height: 0px;
9     overflow: hidden;
10    /* To add a transition animation to make the menu look like it actually
11       drops down from the navbar. */
12    transition: max-height 0.7s;
13 }
14
15 .navbar .profile .drop-down-wrap.open-menu {
16    max-height: 400px;
17 }
```

Navbar.tsx

```
import "./styles/NavbarStyles.css";
import { Link, useMatch, useResolvedPath } from "react-router-dom";
import React, { useState } from "react";
import { MdLogout, MdSettings } from "react-icons/md";

export default function Navbar() {
  //State to manage the toggle of the dropdown menu
  const [isOpen, setIsOpen] = useState(false);
  // To toggle the dropdown menu
  const toggleDropdown = () => {
    setIsOpen(!isOpen);
  };
  return (
    <nav className="navbar">
      <div className="left-section">
        <Link to="/home">
          <img src={Logo} alt="UniFeed Logo" className="logo" />
        </Link>
        <h1 className="title">UniFeed</h1>
      </div>
      <div className="middle-section">
        <ul className="nav-links">
          <NavLink to="/home">Home</NavLink>
          <NavLink to="/groups">Groups</NavLink>
          <NavLink to="/mygroups">My Groups</NavLink>
          <NavLink to="/messages">Messages</NavLink>
        </ul>
      </div>
      <div className="right-section">
        <div className="profile" onClick={toggleDropdown}>
          <img src={ProfilePicture} alt="Profile" />
          <h2>Profile</h2>
          <div className={`drop-down-wrap ${isOpen ? "open-menu" : ""}`}>
            <div className="drop-down">
              <div className="user-info">
                <img src={ProfilePicture} alt="Profile" />
                <h2>{Name}</h2>
              </div>
              <hr />
              <Link to="/profile" className="drop-down-link">
                <MdSettings className="icon" />
                <p>Account Settings</p>
                <span>{">"}</span>
              </Link>
              <Link to="/login" className="drop-down-link">
                <MdLogout className="icon" />
                <p>Logout</p>
```

```
48                 <span>{">"}</span>
49               </Link>
50             </div>
51           </div>
52         </div>
53       </div>
54     </nav>
55   );
56 }
57
58 /*This function creates a custom Link function, to re-use code so that each
58 link in the navigation bar is only 1 line in the Navbar function. */
60
61 /* It also dynamically updates the className of the link so that if a link
62 is pressed, it will update the className, changing the colour of the text
63 on the Navbar for 64 that link to highlight the current page that the user
64 is on. */
65
66 export function NavLink({ to, children, ...props }: Props) {
67   const resolvedPath = useResolvedPath(to);
68   const isActive = useMatch({ path: resolvedPath.pathname, end: true });
69   return (
70     <li>
71       <Link to={to} className={isActive ? "active" : ""} {...props}>
72         {children}
73       </Link>
74     </li>
75   );
76 }
77
78 //Defines the interface for Props that are used in the NavLink function
79 interface Props {
80   to: string;
81   children: React.ReactNode;
82 }
```

6.2.2 Home Page

User Interface

As the home page is the first page the user sees when they access the website, it serves as a 'feed' page that shows the user all of the latest content, to help keep the students up to date with the latest news and information. I divide this feed of posts into two categories; on the left side is the feed for all of the posts posted by official university accounts (such as the university's account and the student's union account), and the right side is the feed of all the posts in the groups that the user is a

member of. Also, there is a large title banner across the top of the home page just below the navigation bar, emphasising that the user is on the home page, as well as adding some more colour and vibrancy to the page. This title banner feature is consistent across the main pages of the website, each with their own unique colour, to help differentiate the pages from each other.

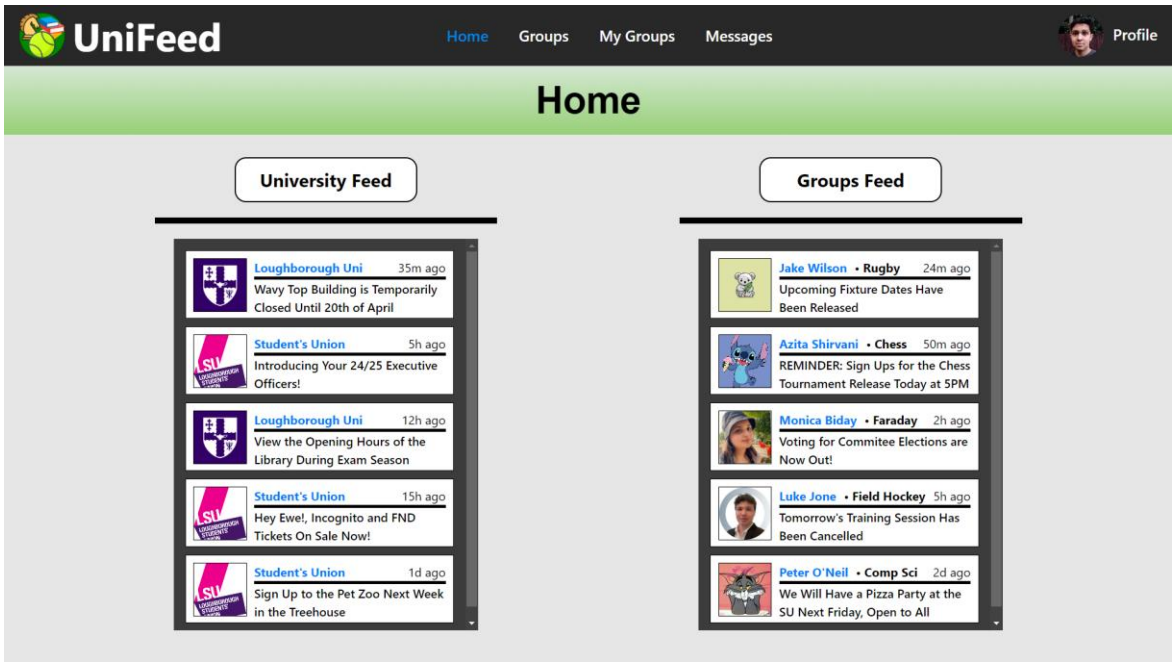

Figure 6.2: View of Home Page

Feed Container Sizing

Both of the 'feed' boxes height scale with the page size, to prevent the feed from becoming an endless stream of posts going down the website. Instead, the boxes themselves are scrollable. This functionality is achieved by using a calculation to set the height of the feed container to 100% of the viewable height, minus 350 pixels (to account for the content above the feed, including the navigation bar, title banner, etc.) I also set the overflow in the y-axis to automatic to enable scrolling within the container. I also account for particularly short and large web page sizes by defining a minimum and maximum height for the feed box, to ensure that the feed is still viewable in a presentable manner.

```
.homepage .row .column .feed-container {
  margin-top: 20px;
  padding: 15px;
  color: var(--white);
  background-color: var(--dark-grey-2);
  width: 400px;
  height: calc(100vh - 350px);
  min-height: 300px;
  max-height: 800px;
  display: flex;
  flex-direction: column;
```

```
  overflow-y: auto;
}
```

Adding Posts to Feed Container

To display the posts from the database in the feed containers, I created two functions, called UniFeedPost and GroupsFeedPost, which can be used to display a post in the 'University Feed' and 'Groups Feed' containers, respectively.

I then utilise the .map feature to loop through the contents of the database tables, which have been pre-formatted to only contain the relevant information (e.g. convert time from the time of post to how long ago it was, for example '54m ago'). Inside the map feature, I call the custom functions I created to display all the posts in the table.

The Groups Feed div (with the loop of displaying posts highlighted)

```
<div className="column">
  <div className="textbox">Groups Feed</div>
  <div className="bar"></div>
  <div className="feed-container">
    {GroupsFeedTable.map((post, index) => (
      <GroupsFeedPost
        key={index}
        Name={post.Name}
        ProfilePicture={post.ProfilePicture}
        Time={post.Time}
        Message={post.Message}
        GroupName={post.GroupName}
      />
    ))}
  </div>
</div>
```

The GroupsFeedPost function (used to display a post in the Groups Feed container)

```
function GroupsFeedPost({
  Name,
  ProfilePicture,
  Time,
  Message,
  GroupName,
}: GroupsFeedPostProps) {
  return (
    <div className="post">
      <div className="left-section">
        <img src={ProfilePicture} alt={Name} className="picture" />
      </div>
      <div className="right-section">
```

```
        <div className="row">
          <p className="name">{Name}</p>
          <p className="groupname">• {GroupName}</p>
          <p className="time">{Time}</p>
        </div>
        <div className="row">
          <div className="line"></div>
        </div>
        <div className="row">
          <p className="message">{Message}</p>
        </div>
      </div>
    </div>
  );
}
```

6.2.3 Groups Page

User Interface

To create the user interface for the groups page, I used a rows and columns approach to position the content as I designed in the design section.

The whole page is within '***groupspage***' container, with the filter buttons for sports, societies and custom all being in column containers within a parent row container. On the second row, is the black bar that goes across the page. Then on the third row, is the groups container.

Just like above, in the '***groups***' container, I use sets of rows and columns division containers to place four groups per row. I also defined a height constraint to the groups container, along with overflow functionality in the y-axis to allow for scrolling within the groups section, eliminating the need to add scrolling to the whole page instead.

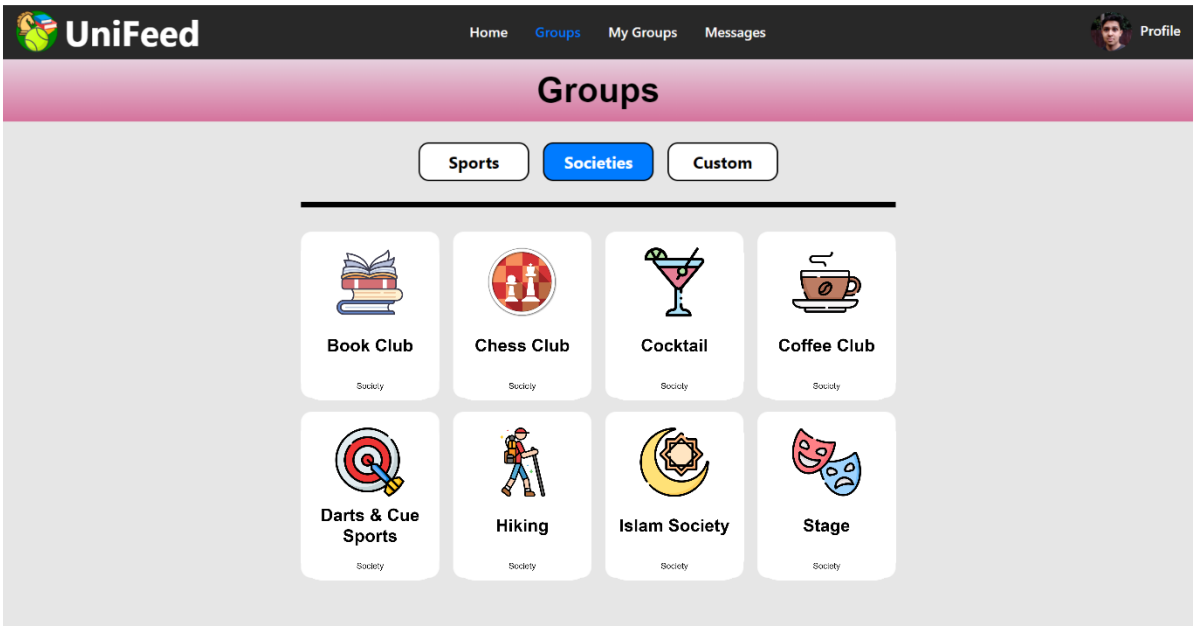

Figure 6.3: View of Groups Page

Functionality

The main functionality within the Groups page is the filter feature. At the top of the page, are three buttons (called “Sports”, “Societies” and “Custom). Clicking these buttons can be used to toggle the filters to choose which groups are displayed to the user.

I used TypeScript coupled with boolean logic to toggle the status of the buttons’ class name, changing the class name between ‘visible’ and ‘hidden’. When the class name is ‘hidden’, that group type is not displayed.

Groups function

```
export default function Groups() {
  var ActiveFilter = "Sports";

  const [isSportsActive, setIsSportsActive] = useState<boolean>(true);
  const toggleSportsActive = () => {
    setIsSportsActive(!isSportsActive);
    ActiveFilter = "Sports";
  };

  const [isSocietiesActive, setIsSocietiesActive] = useState<boolean>(false);
  const toggleSocietiesActive = () => {
    setIsSocietiesActive(!isSocietiesActive);
    ActiveFilter = "Societies";
  };

  const [isCustomActive, setIsCustomActive] = useState<boolean>(false);
  const toggleCustomActive = () => {
    setIsCustomActive(!isCustomActive);
```

```
  ActiveFilter = "Custom";
};

return (
  <div className="groupspage">
    <div className="title-banner">
      <h1 className="title">Groups</h1>
    </div>
    <div className="row">
      <div className="column">
        <button
          className={`textbox ${isSportsActive ? "active" : ""}`}
          onClick={toggleSportsActive}
        >
          Sports
        </button>
      </div>
      <div className="column">
        <button
          className={`textbox ${isSocietiesActive ? "active" : ""}`}
          onClick={toggleSocietiesActive}
        >
          Societies
        </button>
      </div>
      <div className="column">
        <button
          className={`textbox ${isCustomActive ? "active" : ""}`}
          onClick={toggleCustomActive}
        >
          Custom
        </button>
      </div>
    </div>
    <div className="row">
      <div className="bar"></div>
    </div>
    <div className="row">
      <div className="groups-container">
        {GroupsArray.map((post, index) => (
          <Group onClick={DisplayGroup}
            className={ActiveFilter === {GroupType} ? "visible" : "hidden"}
            key={index}
            Logo={post.Logo}
            GroupName={post.GroupName}
            GroupType={post.GroupType}
          />
        ))}
```

```
        </div>
      </div>
    </div>
  );
}
```

I also created a custom Group function that is called to display each group. Within this function, I make use of modals to change the display of a particular group's box when it is clicked. A modal is a versatile user interface element that provides me with additional features to customise the user interface. I am using the modal to toggle between two different views within the group box.

In the division there are two model elements, one is visible when the 'isModalClosed' variable is set to true (which is the default view as seen in Figure 6.3 above). When the group is clicked, the boolean variable is changed and the alternate view displays instead, displaying the group's description, along with a 'Join Group' button.

Group function

```
function Group({
  Logo,
  GroupName,
  GroupType,
  isModalOpen,
  GroupDescription,
}: GroupProps) {
  var isClosed = !{isOpen};
  return (
    <div className="group">
      <Modal isClosed={isModalClosed} close={() => setIsModalOpen(true)}>
        <img src={Logo} className="logo" />
        <h2>{GroupName}</h2>
        <p>{GroupType}</p>
      </Modal>
      <Modal isOpen={isModalOpen} close={() => setIsModalOpen(false)}>
        <p>{GroupDescription}</p>
        <button onClick={JoinGroup}>Join Group</button>
      </Modal>
    </div>
  );
}

interface GroupProps {
  Logo: string;
  GroupName: string;
  GroupType: string;
  isModalOpen: boolean;
  GroupDescription: string;
```

```
}
```

To setup the modal, I also created a modal.tsx file that handles the operations of the modal:

```
import React from "react";

interface ModalProps {
  isOpen: boolean;
  close: () => void; // Function to close the modal
  children: React.ReactNode; // Content inside the modal
}

const Modal: React.FC<ModalProps> = ({ isOpen, close, children }) => {
  if (!isOpen) return null;

  return (
    <div className="modal-overlay" onClick={close}>
      <div className="modal-content" onClick={(e) => e.stopPropagation()}>
        {children}
        <button onClick={close}>Close</button>
      </div>
    </div>
  );
};

export default Modal;
```

6.2.4 My Groups Page

User Interface

The user interface of this page is mostly the same as the Groups page, sharing most of the same code and features. What is unique about this page, in comparison to the Groups page, is that clicking on a group will open the group.

This is the code for the OpenGroup function, which is run when a group is selected.

```
import { useState } from "react";
import "../styles/OpenGroup.css";

type Message = {
  id: number;
  text: string;
  sender: "you" | "other";
};
```

```
export default function OpenGroup(GroupID) {
  const [activeTab, setActiveTab] = useState<string>("Feed");

  return (
    <div className="open-group">
      <div className="tabs">
        <button onClick={() => setActiveTab("Feed")}>Feed</button>
        <button onClick={() => setActiveTab("Chat")}>Chat</button>
        <button onClick={() => setActiveTab("Members")}>Members</button>
      </div>
      <div className="content">
        {activeTab === "Feed" && <div>Feed content goes here...</div>}
        {activeTab === "Chat" && (
          <div className="chat-box">
            {messages.map((message) => (
              <div key={message.id} className={`message ${message.sender}`}>
                {message.text}
              </div>
            ))}
          </div>
        )}
        {activeTab === "Members" && (
          <ul>
            {members.map((member, index) => (
              <li key={index}>{member}</li>
            ))}
          </ul>
        )}
      </div>
    </div>
  );
}
```

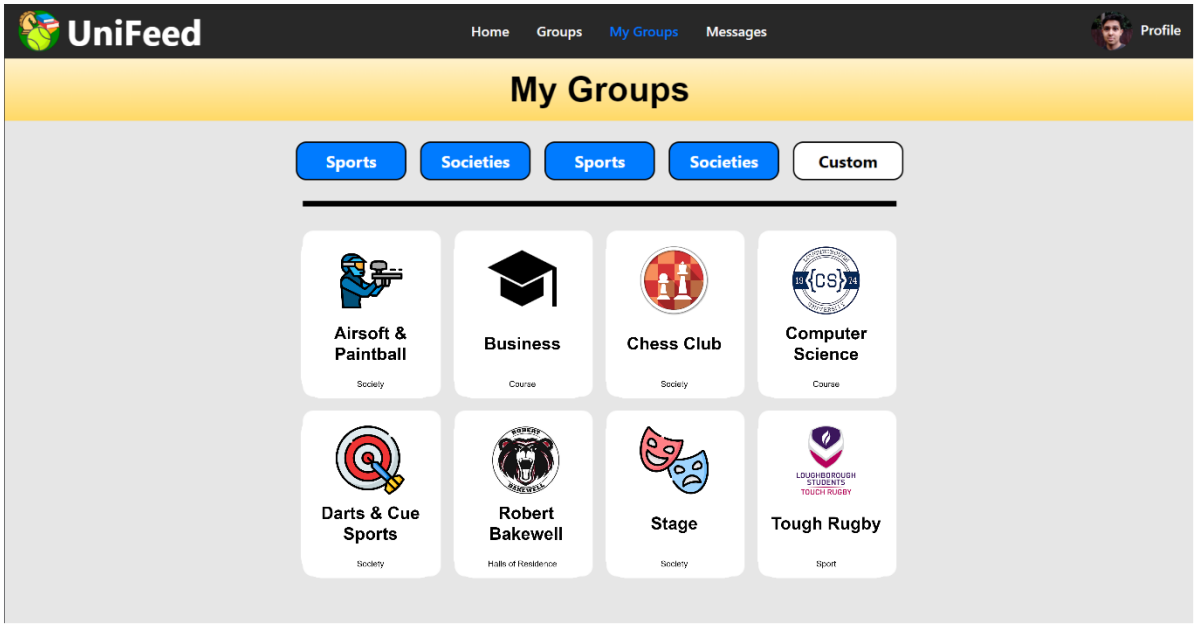

Figure 6.4: View of My Groups Page

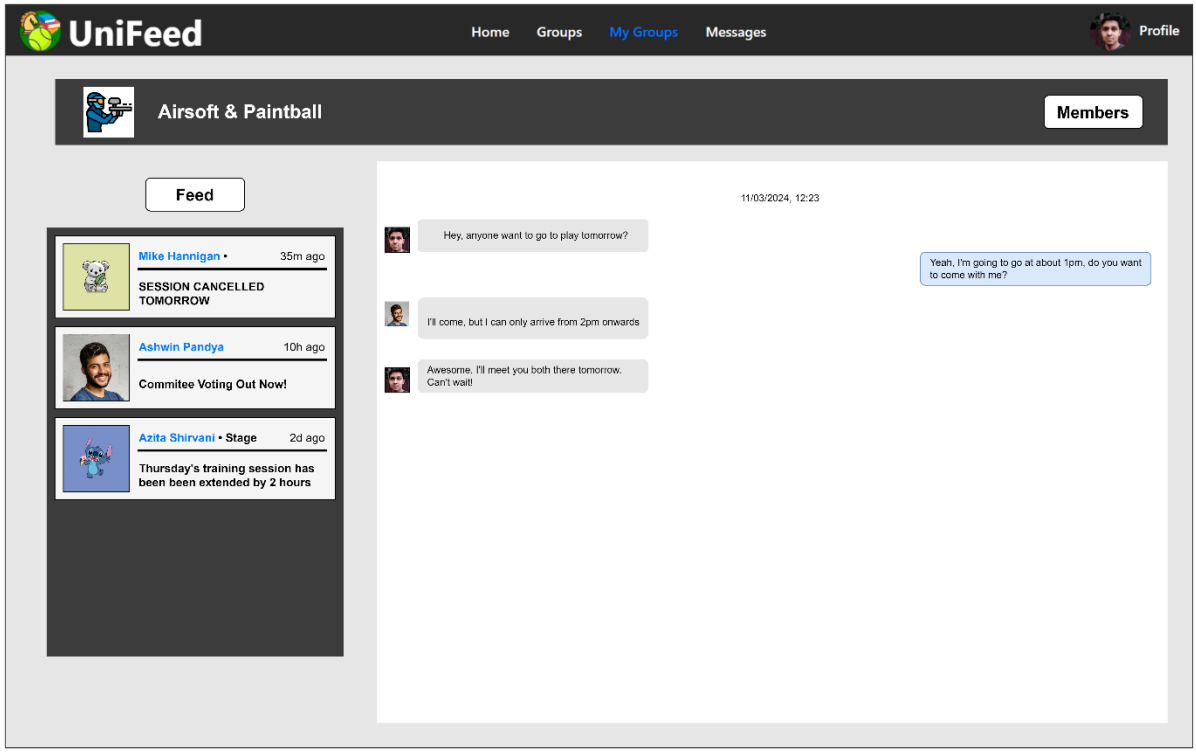

Figure 6.5: View of Specific Group Page

6.2.5 Messages

The messages page consists of re-used code from the previous pages. This includes the feed container from the home page, but re-configured into a chat-box, as well as the messaging feature from the Groups code.

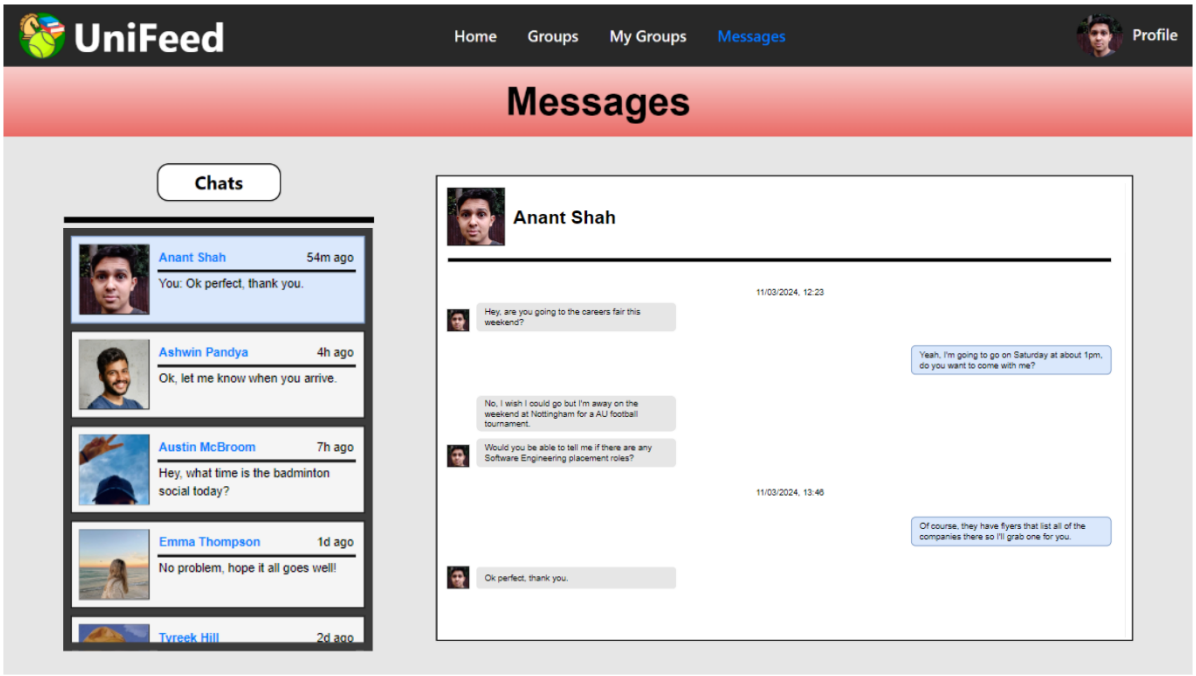

Figure 6.6: View of Messages Page

Chapter 7

Testing

Contents

7.1 Introduction

This section will demonstrate the testing process that I went through to test the application to find out if it meets the requirements that I set in the specification section.

7.2 Functionality Testing

Functionality testing is a form of software testing that validates the software system against the functional requirements pre-defined in the specification. The objective of this process is to ensure that the development application fulfils the goal that I originally set out to achieve. In this context, I decided to develop my web application with the aims to produce a solution to the problem I identified in my abstract and problem domain.

I went about carrying out these tests by creating testing tables, which a separate table for each of the components of the website. Each table has the following headers: element, expected result, status and comments.

- The ‘element’ column states what the element is that I am testing.
- The ‘expected result’ column states what the result is that I initially expected that element to produce.

- The ‘status’ column indicates whether the expected result was fulfilled or not, in binary form (i.e. pass or fail).
- The ‘comments’ column is a non-mandatory column, that I can use to add any relevant extra information for a specific test. This involves the tests that were either not successful, partially successful or elements where the expected result was no longer applicable due to any unforeseen discoveries during the development of the project.

7.2.1 Login

Element	Expected Result	Status	Comment
Email Input	A user is able to enter their email address.	Pass	
Password Input	A user is able to enter their password.	Pass	
Login Button	A user is able to login to the website using the login button.	Pass	
Sign Up Page Link	A user is able to navigate to the sign up page using the link.	Pass	

Table 7.1: Login Component Testing

7.2.2 Sign Up

Element	Expected Result	Status	Comment
First Name Input	A user is able to enter their first name.	Pass	
Surname Input	A user is able to enter their surname.	Pass	
Email Input	A user is able to enter their email address.	Pass	
Date of Birth Input	A user is able to enter their date of birth.	Pass	
Password Input	A user is able to enter their password.	Pass	
Confirm Password Input	A user is able to enter their ‘confirm’ password.	Pass	
Select Gender Input	A user is able to select their gender from the drop-down menu.	Pass	
Select University Input	A user is able to select their university from the drop-down menu.	Pass	
Select Course Input	A user is able to select their course from the drop-down menu.	Pass	
Select Halls Input	A user is able to select their halls of residence from the drop-down menu.	Pass	
Sign Up Button	A user is able to sign up and create an account using the sign up button.	Pass	
Login Page Link	A user is able to navigate to the login page using the link.	Pass	

Table 7.2: Sign Up Component Testing

7.2.3 Navigation Bar

Element	Expected Result	Status	Comment
Logo Button	A user is able to click on the logo to navigate to the home page.	Pass	
App Name Button	A user is able to click on the app name to navigate to the home page.	Pass	
Home Link	A user is able to click on the 'Home' link to navigate to the home page.	Pass	
Groups Link	A user is able to click on the 'Groups' link to navigate to the home page.	Pass	
My Groups Link	A user is able to click on the 'My Groups' link to navigate to the home page.	Pass	
Messages Link	A user is able to click on the 'Messages' link to navigate to the home page.	Pass	
Profile Drop-Down Menu	A user is able to click on the profile picture/ profile text to open/close the drop-down menu.	Pass	
Profile Drop-Down Menu	A user is able to click on the 'Profile' link to navigate to the profile page.	Pass	
Profile Drop-Down Menu	A user is able to click on the 'Logout' link to log the user out and navigate to the login page.	Pass	

Table 7.3: Navigation Bar Component Testing

7.2.4 Home

Element	Expected Result	Status	Comment
University Feed	A user is able to view all the posts made by university-affiliated accounts in the University feed.	Pass	
University Feed	A user is able to click on a post in the University feed to open and view the contents of the post.	Pass	
University Feed	A user is able to scroll within the University feed to navigate through the stream of posts.	Pass	
Groups Feed	A user is able to view all the posts made in Groups that the user is a member of in the Groups feed.	Pass	
Groups Feed	A user is able to click on a post in the Groups feed to open and view the contents of the post.	Pass	
Groups Feed	A user is able to scroll within the Groups feed to navigate through the stream of posts.	Pass	

Table 7.4: Home Page Testing

7.2.5 Groups

Element	Expected Result	Status	Comment

Groups View	A user is able to view all the groups available to them, in a neat and organised list.	Pass	
Group Type Filter	A user is able to select the group type filters to filter the list of groups displayed to them.	Pass	
Groups Display Button	A user is able to click on a group to open an information display about that group.	Pass	
Groups Join Button	A user is able to click the join button on a group to become a member of that group.	Pass	

Table 7.5: Groups Page Testing

7.2.6 My Groups

Element	Expected Result	Status	Comment
Groups View	A user is able to view all of the groups that they are a member of, in a presentable manner.	Pass	
Open Group	A user is able to click on a group to open that group.	Pass	
Group Features	Groups should have a feed channel that displays the feed of posts.	Pass	
Group Features	Groups should have a chat channel that allows group members to send and read messages.	Pass	
Group Features	Groups should have a member's channel that lists all the members of that group.	Pass	
Group Features	A user should be able to click on another user to view their profile.	Fail	Profile feature was not implemented

Table 7.6: My Groups Page Testing

7.2.7 Messages

Element	Expected Result	Status	Comment
Chats History	A user is able to view the list of users that they have had chats with.	Pass	
Open Chats	A user is able to click on a chat to open that chat channel.	Pass	
Messaging	A user is able to send and read messages in the chat channel.	Pass	

Table 7.7: Messages Page Testing

7.2.8 Profile

Element	Expected Result	Status	Comment
View Profile	A user is able to view their profile.	Fail	This feature was not implemented due to time constraints and
Edit Profile	A user is able to edit the information on their profile.	Fail	

Customise Profile	A user is able to customise the content displayed on their profile.	Fail	its low priority. My time constraints are mentioned in my mitigating circumstances claim.

Table 7.8: Profile Page Testing

7.4 Conclusion

From this testing process, I can conclude that the majority of the elements met their expected result. As I had personal life problems (as stated in my mitigating circumstances claim), I didn't have sufficient time to fulfil all the requirements of the project, and because of this the profile page wasn't created and therefore it failed all of its tests. However, this is one of the tasks that I will mention in the evaluation chapter, under the future works and improvements section.

Despite these challenges, I was still able to meet the expected result for most of the tests.

Chapter 8

Evaluation

Contents

8.1 Introduction

In this section, I will conduct a complete analysis of the final results of the project, with respect to the initial goals and objectives that were declared at the beginning of the project. I will then go on to explain the future actions I could take to further enhance this project.

8.2 Analysis

In my analysis, I will compare the final product to the aims and objectives that I set in section 1.2.

Objective 1:

Conduct a literature review to find well-suited technologies and methodology to complete the project.

In my literature review, I conducted thorough research for the state-of-the-art technologies that I could utilise in my web application, which helped me confidently pick the web technologies and tools that I would use for my project.

Objective 2:

Carry out a gap analysis to find the strengths and weaknesses of the existing solutions, and then determine how I can use the key findings from this research to improve my solution.

I would say that I met this objective to a good extent. I conducted an in-depth gap analysis in the existing market, which the focus being on Facebook, as I deemed that there were no other solutions out there that were viable. I took key takeaways from my analysis on Facebook, that helped to influence and shape the structure of my application.

Objective 3:

Create designs to produce a clear and easy-to-use graphical user interface.

I used the software draw.io to produce high quality, visually appealing wireframe designs for my user interface. These designs all followed a common thematic of being minimalistic whilst still delivering all of the features that I intended to.

Objective 4:

Develop a web application that will help students create friends at university.

This was the largest portion of the objectives. To achieve this objective, I first began with dedicating time towards learning the programming languages, web technologies and tools that I had chosen to use for my project. This led to me producing a web application that satisfied most of the specification that I declared in chapter 3.

However, due to unfortunate and unforeseen circumstances regarding my personal life, I had a time-consuming personal matter that occurred during the majority of this stage, which resulted in the quality of the work produced to not be satisfying enough as a product that I would personally use.

Objective 5:

Research testing methodologies and use them to assist in producing a robust product.

For this objective, various testing methodologies were researched, with me concluding on using functionality testing to carry out the final testing once each component was finished being developed. In addition, I also conducted a lot of testing during the implementation stage, as this was in theme with my agile kanban methodology.

8.3 Future Work and Improvements

There is a large number of paths that I could take to improve and enhance my product. I will divide this into two categories, improvements and additional features.

8.3.1 Improvements

The current application that I have produced lacks some of the minimum features for it to be an application that the average person would reasonably consider using. Currently, users have no way of actually creating groups, so implementing this feature would be the first improvement I would add.

Then, I would go back and review the first three chapters that I wrote, ensuring to capture all of the features that I said I would implement, and ensure that each feature is implemented is an efficient, presentable way. Once I completely meet all of the requirements set in the specification, I would then move on to adding additional features.

Hosting Website on a Cloud Service

Another significant drawback of my current product is that it still is only available if I locally host the front-end and back-end. Integrating my website with a cloud service, such as Google Cloud Platform or Amazon Web Services means that I can put my website online for other people to actually visit and use.

8.3.2 Additional Features

Enhanced Chat Experience

As this is a social media website targeted towards young adults, this app needs to be trendy and packed with the latest features to rise above the competition.

This is primarily achieved by adding emojis, reactions and gifs to the chatting interface, allowing students to communicate with each other using more than just words.

Mobile User Interface

The majority of students nowadays primarily use their phone for social media, as opposed to laptops and computers. For this app to be successful, it is essential that the web application has a dedicated mobile view. Once this is implemented, the app could then be made available as a mobile web application, rather than just a website.

8.4 Reflection on Learning

Undertaking this project has provided me with the opportunity to develop and further advance my Computer Science development skills. I have utilised a lot of the knowledge, information and skills that I gained through the modules on my university course, my placement year, personal projects, and more. In this section, I will reflect on everything that I have learnt throughout the development of this project.

Prior to this project, I had gained an interest in web development, specifically from my experience during the Web Programming and Team Projects modules in my first and second year of my course. However, during my placement year at TJX Europe, I was given the role of platform engineer, does not involve any web coding, or any coding for that matter. As a result of this, I had forgotten a significant amount of my development and engineering skills due to lack of maintaining the skill. This brought a steep learning curve when I began this project, as I had very little experience in web development beyond the basics of HTML, CSS and JavaScript.

I went about tackling this issue of rediscovering my coding abilities, by reading the documentation and resources provided for the tools I required for my project on their relevant websites(e.g. Node.js, React.js, Django, MySQL, etc.). I also made use of online forums and YouTube videos to learn how to use specific features among the tools.

The aspect of the project that I enjoyed the most was the design stage, as I have a passion for aesthetics and a slight issue in perfectionism, meaning that I often spend too much time tweaking the minor details. However, discovering wireframe prototypes was particular useful for me as it was a method of design that allowed me to draw and create various iterations of designs, whilst focusing on the primary details and abstracting away the less significant attributes, such as the exact styling of elements. In addition, draw.io is a design tool that I have dealt with a lot in the past, so I was already experienced in the software.

This project was also my first hands-on experience at complete full-stack development, creating the client-side, server-side and database using three separate programming languages with their respective tools. I learnt incredibly valuable knowledge on the precise details of each stack, and more importantly, how to setup the configuration to allow the stacks to communicate with one another. This was a very challenging issue at first, and even in my final product I haven't completely handled this configuration the correct and intended way, so I still have more to learn on full-stack development. The key takeaway is that I have beaten the first hurdle in this area of computer science, and now view it as an exciting adventure that I will continue to learn about, rather than a daunting and overwhelming approach to software engineering.

References

1. Madge, C., Meek, J., Wellens, J., & Hooley, T. (2009) 'Facebook, social integration and informal learning at university: "It is more for socialising and talking to friends about work than for actually doing work"', Learning, Media and Technology, 34(2), 141-155, DOI: 10.1080/17439880902923606
2. Anderson, M., Faverio, M. and Gottfried, J. (2023) 'Teens, social media and technology 2023', Pew Research Center: Internet, Science & Tech. Available at: https://www.pewresearch.org/internet/2023/12/11/teens-social-media-and-technology-2023 (Accessed: 16 December 2023).
3. MySQL. (no date). MySQL Workbench. Retrieved from https://www.mysql.com/products/workbench/
4. Django REST Framework, (no date), Django REST Framework. Available at: https://www.django-rest-framework.org/ (Accessed: 3 February 2024)