The Neighbour Hub - Documentation

Introduction to the Neighbour Hub project & background

The selection of our project followed a collaborative process where we ensured that every member of the group had the chance to independently draft a proposal for an app concept. To determine which we would choose for the final project we conducted various rounds of voting before reaching a consensus to create an app that creates connections between neighbours within a specific geographical area, such as a street or wider community.

The concept of The Neighbour Hub was chosen for its potential to connect communities and facilitate interactions between those living near each other. The app would enable users to create profiles linked to their address, connecting them with neighbours who share similar interests or goals. Additionally, we intended on the app providing features such as a community message board for discussions, tools to organise and promote local events and spaces for sharing recommendations, offering services of looking for assistance within the neighbourhood.

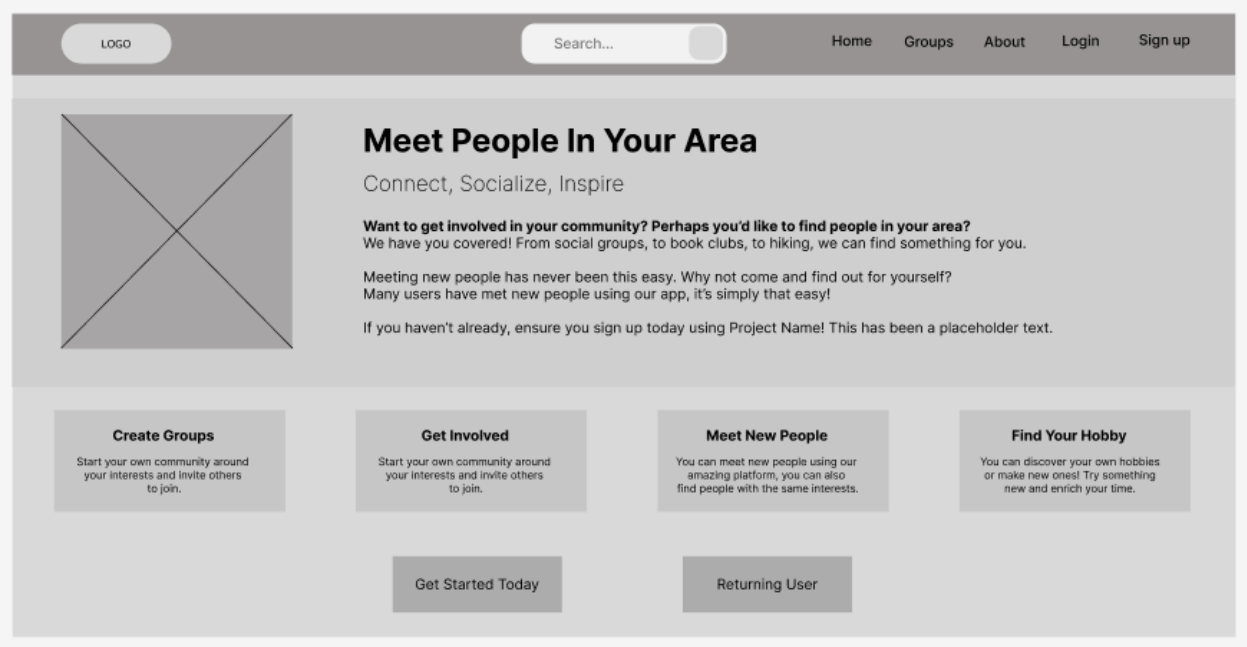
While the scope of the project initially presented challenges, particularly compared to some of the simpler proposals, as a group we addressed these concerns by developing a detailed plan. This plan outlined the core features to prioritise, such as the message board and events page, while identifying optional features that could be added if time permitted. We used this plan to form a task list and then delegated responsibilities based on each team member's skills and interests. In doing this we ensured that all team members felt confident in their roles. By using the individual strengths within the group, we began to lay the foundations for a successful app.

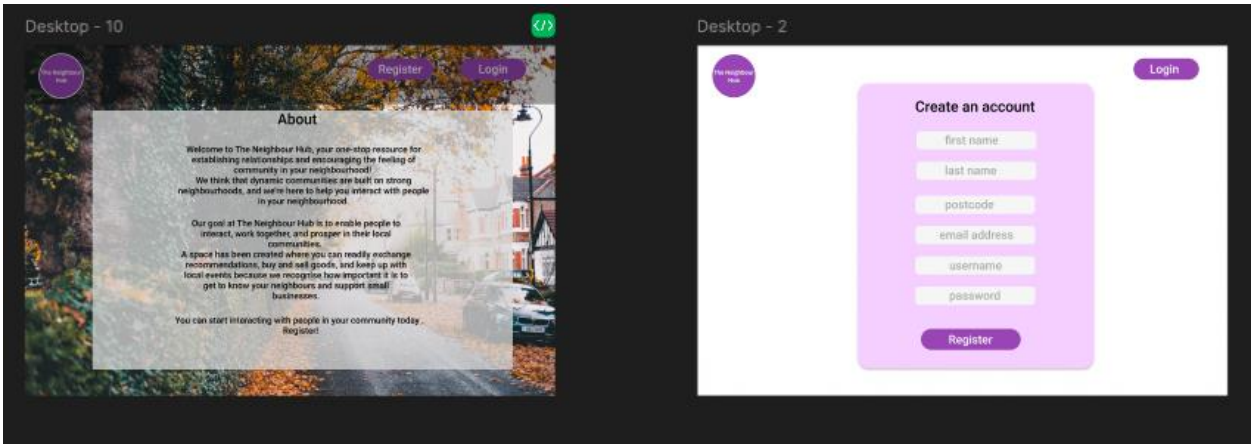
Specifications and design

To start we researched different existing meetup websites such as Bumble Friends, meetup.com etc. During this process we analysed;

- **Branding and colour schemes:** We noticed and observed how colours and layouts were carefully chosen to align with the theme of the website and with its target audience.
- **UI and design:** The use of clean responsive and interactive layouts.
- **Naming and concepts:** The naming, logo and branding of the websites and how closely it was tied to its concept.

Drawing inspiration from these websites we made the first design wireframing on Figma and made some mock-ups. These designs were then refined and carefully picked our colours for the outcome of our website to ensure a user-friendly and welcoming website.





Implementation and Execution

Development Process

Frontend development tasks were divided between Filsan and Yusra. First, we designed the structure of the website using HTML, CSS, and JavaScript on Visual Studio Code. Each group member was assigned to a particular section of the design, such as navigation, user registration, and the homepage layouts. Once the designs were complete, it was time to implement those designs in React.

Database design

Following the creation of the wireframe designs, it became much easier to visualise and plan the layout and functionality of each page of the web app. This process also helped to clarify the type of information that needed to be stored or provided by the database. We discussed the relevance of certain data points and discussed whether some were necessary, especially if they wouldn't be used at this stage. For example, while we initially considered connecting users on a street-by-street level, we decided that using postcodes would be more practical and user friendly. As a result, we removed the requirement for users to input their full street address during sign-up, streamlining the registration process for the user and the database design.

The users table was structured to store user details such as first and last name, email address, username and password. Constraints like UNIQUE and NOT NULL were placed on those fields that needed to be unique and were required to be completed. The general posts table allows users to share information or requests across predefined categories and a foreign key connects the post back to the user that created it. In addition to this the events table allows users to create events, and store details including event name, category, date, location and pricing. Again, these events are connected with a foreign key back to the user. The design of the database allows for its current function, and leaves room for future expansion.

Frontend

We mainly used React for our frontend development to create a great responsive website, there were talks at the beginning of using CSS Tailwind and Bootstrap, but to keep things as simple as possible, CSS was used for the styling. The front-end was first structured in a way that JSX files were in the same folders as their CSS files to keep organised, however a file reshuffle saw that it was just as good to keep all CSS files in one place and JSX in another. The routing was the main issue to take care of and some CSS changes didn't appear straight away until some tweaking had been made. It was also important to ensure login/registration forms were structured properly and contained the right information to allow verification.
One problem noted was the background images and displaying them correctly, we also had issues with responsiveness – this with different devices and with different browsers.
This is something we can do to improve the app in the future and allow for good functionality across mobile devices too.

Back end

Actions performed by Ameenah and Libby. The backend of this project is made up of a server.js file which holds the API functions and the database connectivity. There is also a schema file for the database.
Different dependencies were trialled (for example bcrypt), the code is hard coded to port 5000 although this can be changed in a remote setting.
Ameenah did a fantastic job with much of the API calls.

APIs were quite problematic, especially the login APIs. There were issues with retrieving events and posts, and there were some database issues. The login was more difficult as there seemed to be various ways and dependencies used, e.g. AuthO, Appwrite, Axios etc. Initial we wanted to keep things simple and use a simple password comparison code, however, things were a bit more complicated than initially thought. Two different methods were used when writing the login code, firstly a separate database.js file was made that attempted to capture the data and compare with the database, this code was not effective at all. A second attempt was made by using more simplified code in the server.js file which also aimed to get the username and password, perform an SQL query and if the account existed, there would be direction to the homepage, for some reason this code was ineffective, and it was difficult to figure out why. A third attempt was made to figure out the login, this time with JWT tokens and a little Axios, this was the most code used and affected the homepage, login page and server file. Unfortunately, the code couldn't be corrected in time.

e.g. of login API code:

```
app.get('/', (req, res) => {
   res.sendFile(path.join(__dirname, '.src/pages/Login.jsx'));
});


app.post('/login', async (req, res) => {
   //get username and password
   const { username, password } = req.body;
   if (username && password) {
   //sql query to throw error if user or password doesn't exist
      connction.query('SELECT * FROM users WHERE username = ? AND password =
?', [username, password], function (error, results, fields) {
         if (error) throw error;


   //to find existing accounts
         if (results.length > 0) {
            req.session.loggedin = true;
```

```
                req.session.username = username;
    //direct user to home page
                res.redirect('/HomePage');
            }
            else {
                res.send('Incorrect Username and/or Password!');
            }
            res.end();
        });
    } else {
        res.send('Please enter Username and Password!');
        res.end();
    }
});
```

Problems were generally fixed by team problem-solving, search online within forums, and using debugging techniques and programs such as Postman.
For example, one problem faced was using the port 5000 on a mac, even when the port was not used in the terminal, the port was always in use somewhere else. Troubleshooting online gave the answer that an airdrop feature in the mac settings was using this port when activated. Turning this off solved the port issue and it was able to be used.

Conclusions

Overall, I think we gave this a good go. It's not perfect in functionality but it's a work in progress. The issues we've had with the backend has been a huge learning curve, and we have discovered potential problems that can occur with node and general folder set up.
In the future it would be useful to incorporate other website features and security -this is imperative for a website like this. For example, it would be useful to have a 'change password' function and a user dashboard page so that account editing can take place.
Another idea to implement would be an infinite scrolling feature rather than clicking through pages. This would allow good compatibility with mobile phones and tablets.
The positives to this project were that everybody worked extremely hard and put a lot of time into figuring out code, learning the various dependencies and getting to grips with React and Node. The different teams worked well together to sort out issues and although

there was apparent friction when choosing the correct idea, or figuring out which JavaScript library, the situations were dealt with diplomatically.
It's interesting to work in a team where we live in different places and have never met in person.
Across the team we believe the idea for a neighbourhood meetup app is great because it can help to bring communities close together without the necessity of joining a bigger website like Facebook. We had some great ideas, and we implemented as much as we possibly could with everyone's time restrictions. All in all, we are proud with the overall result and find ourselves in a much more empowered position when it comes to building full-stack applications.