Walkie Talkie - A car-pooling taxi app

-Tinder for taxis

Website link: [Walkietalkie-436fd.web.app]
GitHub repository: [Walkie-Talkie]

Stakeholders:
Cathal Lawlor - 21325456
Paulius Zabinskas - 20120267
Stephen Roe - 21355786
Charlie Quinlan - 21327731

1. Introduction

1.1 General Overview

The Walkie Talkie web application is designed to help users split the cost of a taxi fare by carpooling with other users who are travelling to the same location. To use the app, you would create an account and input your location, a pseudonym or username, and a method of contact. The app then logs this information into its database, and when the "findMatchingUsers" function is called, it returns a list of usernames that are also travelling to the same location, along with confirmation of their origin and destination locations.

While this app was only created as a demo, there is potential to expand on its functionality. For example, rather than having a predefined list of locations, the app could use the user's current location via the Google Maps API and calculate if another user is in close proximity with a similar destination. This could improve the app's usability and increase the likelihood of successful carpooling. Additionally, the built-in chat feature could make the app more convenient for public use.

1.2 Statement of Problem & Goal

During the development of the Walkie Talkie app, several challenges were encountered. One of the main challenges was implementing real-time synchronisation of data between the app and the Firestore database. This required careful consideration of data modelling and database schema design, as well as testing to ensure that changes made by one user were immediately reflected in the user interface of other users.

Another challenge was integrating Firebase Auth to manage user authentication and authorization. This required careful planning to ensure that user data was stored securely, and that only authorised users had access to the data stored in Firestore.

Additionally, designing a user interface that was both functional and user-friendly was a significant challenge. This required careful consideration of layout, colour, typography, and user interaction design. CSS and SCSS were used to create a pleasing transitioning sidebar, which made the user interface more intuitive and engaging.

Overall, the development of the Walkie Talkie app required a multi-disciplinary approach that incorporated knowledge of front-end development, database management, and user interface design. Despite these challenges, the Walkie Talkie app was successfully developed and provided a solution to the problem of high taxi fares for individual travellers.

1.3 Project Success & Evaluation Criteria

Project Success Criteria:

1. Successful deployment of the Walkie Talkie web application on a demo server.
2. The application meets the functional requirements specified in the project brief, including the ability to allow users to find other travellers who are travelling to the same location and share the cost of the taxi fare.
3. The application is user-friendly and intuitive, with a responsive and visually appealing user interface.
4. The application is scalable and can handle numerous users and requests.
5. The application is secure, with user data stored safely and securely using Firebase Auth and Firestore.
6. The application is reliable and performs consistently under normal usage scenarios.

Evaluation Criteria:

1. User engagement: Measure the number of users who sign up for the application and the frequency with which they use it.
2. User satisfaction: Gather feedback from users to evaluate their satisfaction with the application, including its functionality, user interface, and overall usefulness.
3. Conversion rate: Measure the number of successful carpool matches made using the application and evaluate the effectiveness of the algorithm used to find matching users.
4. Performance: Measure the application's response time and load time under various usage scenarios and ensure that it is scalable and can handle numerous users.
5. Security: Evaluate the security of the application, including user authentication, data storage, and data transmission, to ensure that user data is stored and transmitted securely.
6. Reliability: Evaluate the reliability of the application, including its ability to perform consistently under normal usage scenarios, and ensure that any bugs or errors are quickly identified and fixed.

2. USER REQUIREMENTS

The registration and login process for the Walkie Talkie web application is straightforward and user-friendly, and the app's main page makes it easy for users to enter their information and find potential carpool matches.

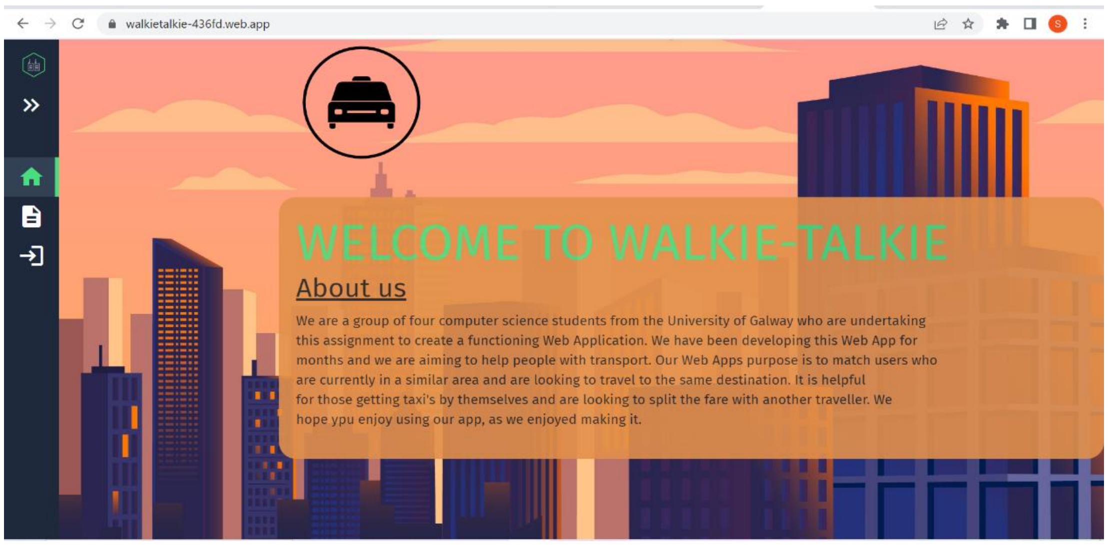
a) To register for the Walkie Talkie web application, the user would click on the "Register" link located on the login page.
b) This would redirect the user to the registration page, where they would be prompted to enter their email address and create a password.
c) Once the user has entered their email and password, they would click the "Register" button to create their account.
d) After registering, the user would be redirected to the login page, where they would enter their email and password and click the "Log In" button to access the main app page.
e) Once logged in, the user would be automatically navigated to the main app page.
f) On the main app page, the user would be prompted to enter their prefix, contact information, current location, and destination.
g) After entering this information, the user would click the "Mach me" button to submit their information to the database.
h) The user would then be able to view a list of other users who are travelling to the same location and are willing to share the cost of the taxi fare.
i) If the user finds a potential carpool match, they would be able to communicate with the other user using the contact information provided.

3. System Design
3.1 UI Design:

For the UI Design the focus was on formatting and styling the web application to ensure that users would find it visually appealing and easy to navigate. To begin the styling process, there had to be close collaboration with the team members who were responsible for developing new functionality and attributes for the website. Additionally, there was regularly sought feedback from Charlie and other group members to ensure that the web app was aesthetically pleasing from the user's perspective.

To gain insight into designing user-friendly and visually appealing applications, we researched several popular apps. We observed that many modern applications use sleek and seamless styling, which, in our opinion, enhances their overall attractiveness. Furthermore, the simplicity and user-friendliness of these apps often lead to returning users, largely due to their layout. For instance, a navigation bar that provides access to all the app's functions with a single click is an effective feature. We instead implemented a sidebar using CSS and SCSS. This sidebar was built as a component so it could be implemented in each ‘page’.

Landing page
Upon launching the web app, the user is directed to the homepage, which features a brief introduction to our group and an overview of our app. At the top of the page, there is an eye-catching orange box that serves as the header. To achieve its rounded-off appearance, we applied the border-radius style tag. The header contains headings and paragraphs that provide further information about our app.



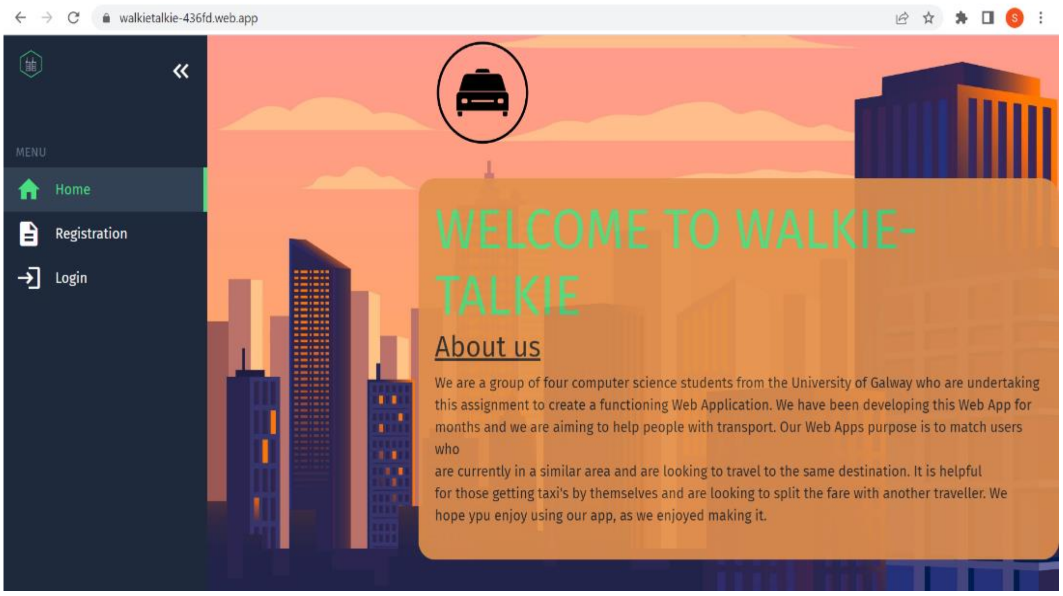
WELCOME TO WALKIE-TALKIE

About us

We are a group of four computer science students from the University of Galway who are undertaking this assignment to create a functioning Web Application. We have been developing this Web App for months and we are aiming to help people with transport. Our Web Apps purpose is to match users who are currently in a similar area and are looking to travel to the same destination. It is helpful for those getting taxi's by themselves and are looking to split the fare with another traveller. We hope ypu enjoy using our app, as we enjoyed making it.

Sidebar

The left-hand sidebar draws the user's attention, as it displays icons that allow them to access different components of the web application. If the user is unable to decipher the meaning of the icons or where they lead, they can expand the sidebar for additional information. We believe that this approach simplifies navigation and makes the web app more accessible to a wider audience.



WELCOME TO WALKIE-TALKIE

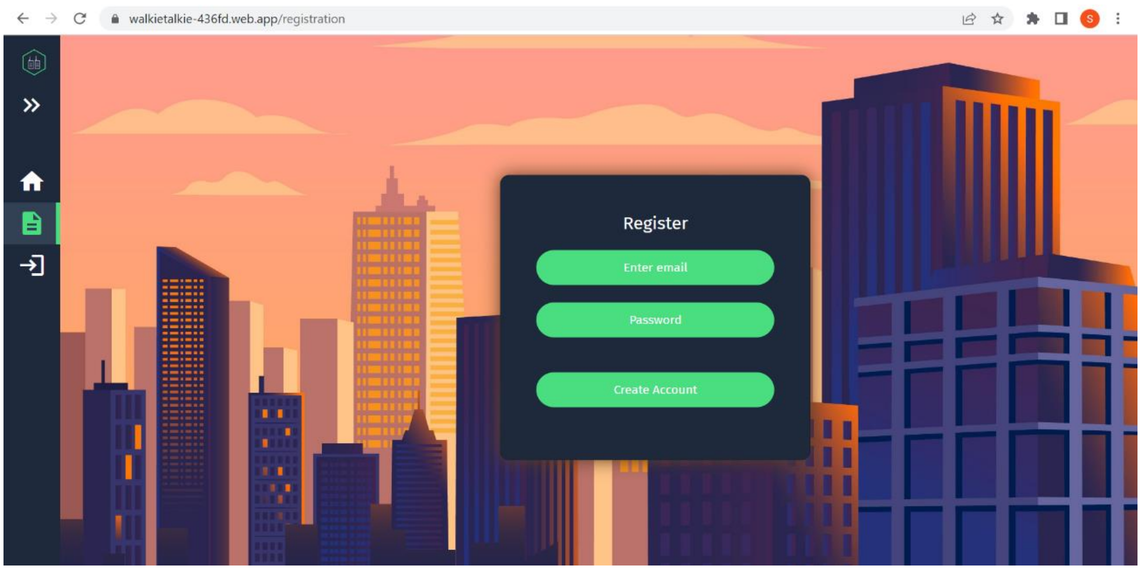
About us

We are a group of four computer science students from the University of Galway who are undertaking this assignment to create a functioning Web Application. We have been developing this Web App for months and we are aiming to help people with transport. Our Web Apps purpose is to match users who
are currently in a similar area and are looking to travel to the same destination. It is helpful
for those getting taxi's by themselves and are looking to split the fare with another traveller. We
hope ypu enjoy using our app, as we enjoyed making it.

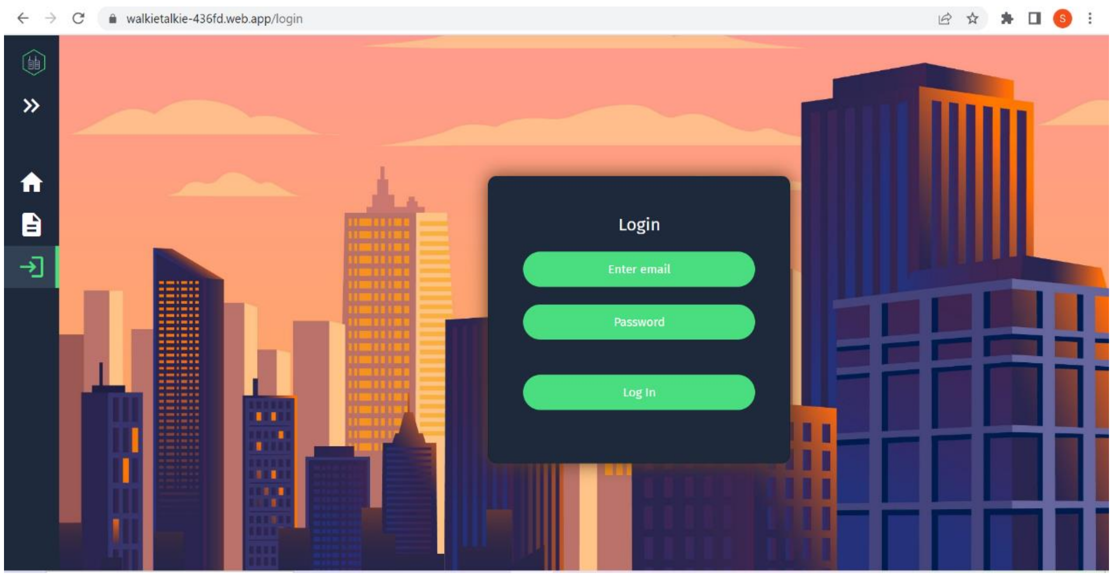
This feature is crucial for ensuring a smooth and satisfying user experience, as it offers additional information to those who may require it. In selecting the colour scheme for our web app, we aimed to achieve a modern and seamless look that would complement our design. To enhance the aesthetic appeal of our app, we incorporated a logo that we found online, but adjusted the colour to match our chosen scheme.

Background image / colour scheme notes

We also included a background image that creates a subtle contrast to the dark sidebar, directing the user's attention to the content of the page. This careful attention to detail in our design approach is aimed at creating a user-friendly and visually appealing web app that can cater to the needs of all users.



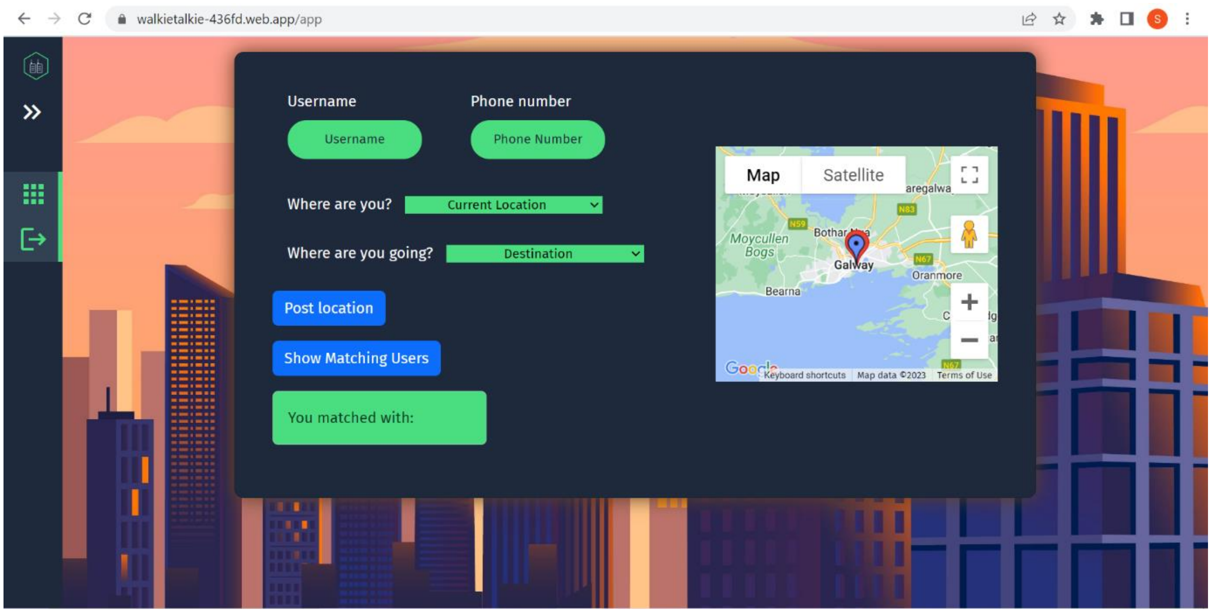
To ensure maximum user-friendliness during the registration and login processes, we kept the layout of both components identical. The inputs were neatly placed in a centred card, with the placeholders also centred. Maintaining a consistent design and layout across all components was essential for creating a cohesive and intuitive user experience.



Once the user has successfully created an account or logged in, they are automatically redirected to the main application page by clicking the button located within the card. This streamlined process helps minimise user frustration and promotes ease of use.

To ensure our users' privacy and security, access to this page is restricted to those who have successfully logged into their accounts. We have maintained consistency in layout and colour scheme across all pages for a seamless user experience.

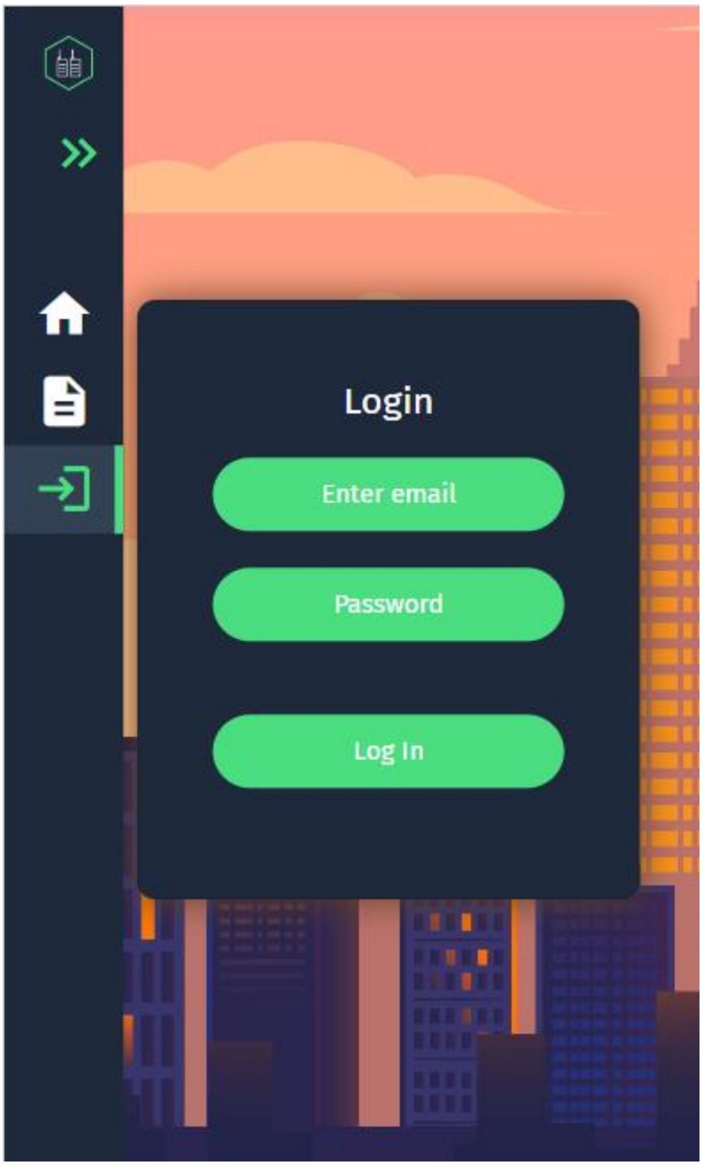
Main application page



Our main application page features multiple inputs and distinct components. The first component enables users to share their location information with the database.
It requires users to input a temporary username, phone number for communication with other users, current location, and intended destination.
The second component facilitates the matching of users who are in the same area and heading to the same destination. When the matching user button is pressed, the matched users' information is displayed in a div that updates in real-time.
Lastly, our map component allows users to view locations they have selected in our select tags, providing real-time feedback and information about their destination.

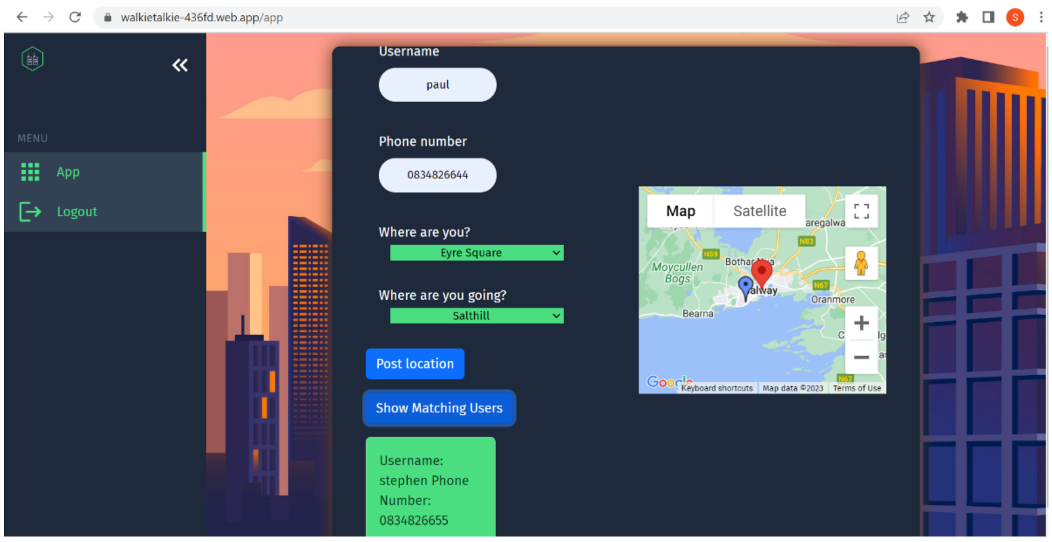
Bootstrap side note

By incorporating Bootstrap, a popular front-end framework, and utilising media queries, we were able to optimise our web application to be responsive and easily accessible on a wide range of devices, including smartphones, tablets, and desktops. This responsive design not only enhances the user experience by providing a consistent and seamless interface across different devices, but also increases the accessibility and reach of our application. The use of responsive design can lead to increased user engagement and satisfaction, which ultimately translates into higher usage rates.



Matching user result

After matching users, their information is displayed on the page along with the locations of each user on the map. The users can communicate with each other outside of our application and make arrangements for travelling together. Once the user has finished using the matching, they can log out to complete their session.



walkietalkie-436fd.web.app/app

Username: paul

Phone number: 0834826644

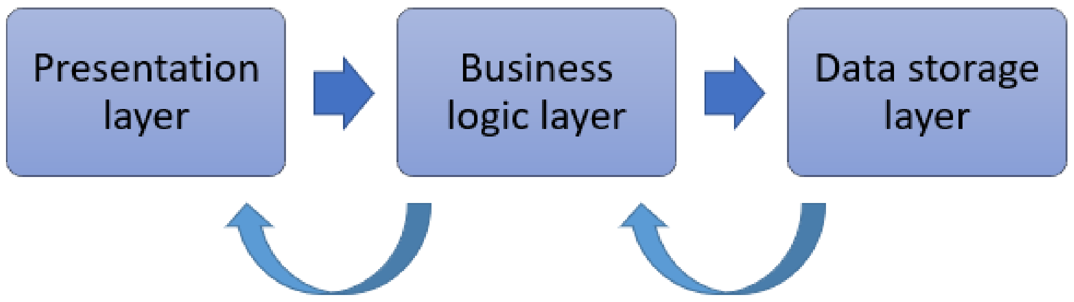
Where are you?: Eyre Square

Where are you going?: Salthill

Username: stephen Phone Number: 0834826655

3.2 Program Design:

- **Architecture**



The Walkie Talkie web application follows a three-tier architecture, which consists of a presentation layer, a business logic layer, and a data storage layer. The presentation layer is responsible for handling user input and displaying information to the user, while the business logic layer is responsible for processing data and implementing application logic. The data storage layer is responsible for storing and retrieving data used by the application.

The architecture of the Walkie Talkie web application includes double-sided communication between the presentation layer, business logic layer, and data storage layer. This allows for efficient data flow and processing, enabling the application to provide a responsive and dynamic user experience.

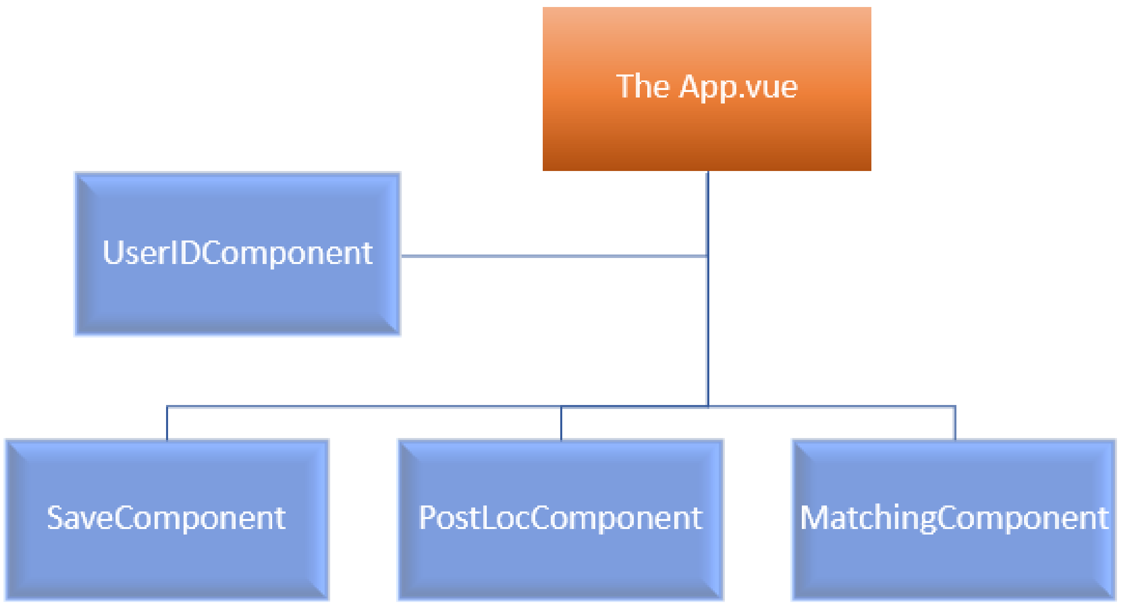
The presentation layer of the Walkie Talkie web application consists of the user interface, which includes all the web pages and forms that the user interacts with. The user interface is designed to be responsive and user-friendly, with a navigation bar that allows the user to interact with all the web pages. However, the user can only access app functionality if they are recognized as a logged-in user.

The business logic layer of the Walkie Talkie web application is implemented using Firebase Cloud Functions, which handle the back-end logic for the application. The cloud functions are used to process user input, handle data storage and retrieval, and implement the custom matching algorithm used to find potential carpool matches. The business logic layer communicates with both the presentation layer and the data storage layer to retrieve data, process it, and return the results to the presentation layer.

The data storage layer of the Walkie Talkie web application is implemented using Firebase Firestore, which is used to store and retrieve user data. Firestore is a NoSQL database that allows for easy scaling and provides real-time data synchronisation, making it an ideal choice for a real-time application like Walkie Talkie. The data storage layer communicates with both the presentation layer and the business logic layer to store and retrieve data as needed.

Overall, the architecture of the Walkie Talkie web application is designed to be modular, scalable, and easy to maintain. The use of Firebase Cloud Functions and Firestore allows for easy implementation of back-end logic and data storage, while the user interface is designed to be responsive and user-friendly. The double-sided communication between the layers of the application enables efficient data processing and flow, providing a dynamic and responsive user experience.

- **Component Based Design & Implementation**



The Walkie Talkie web application was designed and implemented using a component-based approach. This approach allowed the application to be broken down into smaller, more manageable components, each with its own functionality and purpose. Multiple components were used to create the web pages of the application, including UserIDComponent.vue, SaveComponent.vue, PostLocComponent.vue, and MatchingComponent.vue.

For example, the Walkie Talkie signature function page was created using four components: PostLocComponent.vue, MatchingComponent.vue, UserIDComponent.vue, and SaveComponent.vue. The PostLocComponent.vue was used to display the form for the user to enter their location and destination, while the UserIDComponent.vue was used to display the user's identification. The MatchingComponent.vue was used to display the potential matches found by the matching algorithm, and the SaveComponent.vue was used to save the user's information and trigger the matching algorithm.

By using a component-based approach, the Walkie Talkie web application was able to be developed more efficiently, with each component responsible for a specific part of the application's functionality. This also allowed for greater flexibility and modularity in the design, making it easier to add new features or modify existing ones.

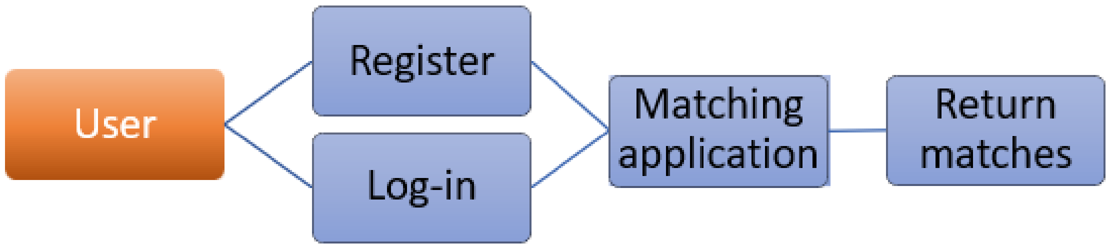
- **Data Model**

The Walkie Talkie web application uses Firebase Firestore to store and retrieve user data. User data is stored in a collection, which is made up of auto-generated IDs that contain information provided by the user. This information includes the user's name or pseudonym, contact details (email or phone number), current location, and destination. Each data point in the collection is associated with an ID linked to the logged-in user, which functions as a recognition of the user and prevents matching the user with him/herself.

Data retrieval is performed using Firebase queries, which ensure performance and adaptability to a larger user base. The queries allow for filtering and sorting of data to find potential carpool matches. This approach is more efficient than downloading the entire database, as only the relevant data is retrieved, reducing the amount of data transferred over the network.

The use of Firebase Firestore as the data storage solution provides real-time data synchronization and easy scalability, making it an ideal choice for a real-time application like Walkie Talkie. The data model is designed to be efficient, secure, and adaptable, ensuring that the application can handle a growing user base while providing a responsive and dynamic user experience.

- **Control Flow**



When a user enters the Walkie Talkie web application, they are directed to the home page, which contains information about the service. From here, the user can choose to either log in or register. If the user chooses to register, they are directed to the matching page. If the user logs in, they are automatically directed to the matching page.

On the matching page, the user is presented with a form that includes text boxes to enter their name or pseudonym and contact information, as well as select boxes to choose their current location and destination. Once the user has entered their information, they can click the "Match Me" button, which triggers a Firebase query to return a list of users traveling in the same direction from the same location within the last 30 minutes.

The control flow of the Walkie Talkie web application is designed to be simple and intuitive, with a clear sequence of actions for the user to follow. The matching algorithm is triggered by the user's input, ensuring that matches are based on up-to-date information. Overall, the control flow is designed to provide a seamless and responsive user experience, making it easy for users to find potential carpool matches.

- **Application testing**

We attempted to implement GitHub actions, specifically trying to use firebase's default github actions settings, intending to have automated testing upon deployment of the app. Unfortunately, we never got this working due to time constraints, with it always breaking our application without fail.

In the future, we would definitely implement this feature, as it would be key for maintaining the application into the future, and to avoid new deployments breaking features.

4. Core Technologies

VueJS UI & Component Design (Stephen & Charlie)

VueJS was used extensively in the development of the Walkie Talkie web application to create an interactive and responsive user interface. To implement the UI system, the application used a combination of VueJS components and CSS.

VueJS components were used to create modular, reusable pieces of user interface functionality. These components were organised into a folder structure under the src/components directory, and included components such as "Navbar", "Sidebar", "UserForm", "MatchList", and others.

CSS was used to style these components and create a cohesive visual design for the application. In addition, VueJS directives such as "v-model" were used to enable two-way data binding, allowing for the creation of interactive user interfaces. For example, when a user entered their location or destination, the application used VueJS to dynamically update the list of carpool matches.

To manage user interaction and navigation between pages, the application used VueJS Router. VueJS Router was used to create a dynamic routing system that allowed the application to display different pages based on the user's actions. For example, when a user submitted their information, the application used VueJS Router to display a list of potential carpool matches.

Firebase database (Paulius & Charlie)

Firestore was used extensively in the Walkie Talkie web application to store and manage user data. We used Firestore to host a collection consisting of individual posted data, each given a random ID. Each data point had several fields of information stored, including the name or pseudonym of the user, their location, their destination, a timestamp, and a unique user ID. This information was used to filter individual users by their location, destination, and time limit, in order to identify potential carpool matches.

The use of Firestore allowed us to store and retrieve data in a structured format, which made it easy to filter and sort the data as needed. Additionally, Firestore's real-time synchronisation capabilities ensured that changes to the data made by one user were immediately reflected in the user interface of other users.

To manage the data in Firestore, we used the Firebase console, which provided an easy-to-use interface for viewing and editing data. We also leveraged Firestore's querying capabilities to search for, and filter user data based on specific criteria, such as location and destination. The data stored in Firestore was also used to run the "findMatchingUsers" function, which returned a list of usernames that were also travelling to the same location.

Firebase auth (Paulius & Charlie)

Firebase Auth is a user authentication and identity management service provided by Google Firebase. In the Walkie Talkie web application, we used Firebase Auth's email and password authentication function to authenticate and authorise users. This functionality was integrated into the app's functionality, such that if a user was not authenticated, they would not be able to use the app's features.

When a user signs up for the Walkie Talkie web application, their email and password are securely stored in Firebase Auth, and the user is then authenticated upon logging in. This allowed us to ensure that only authorised users had access to the data stored in Firestore.

To manage the authentication process, we used Firebase's Authentication API, which provided an easy-to-use interface for managing user accounts and passwords. We were also able to set up security rules that ensured that only authenticated users could access and modify the data in Firestore.

Firebase Cloud Functions (Paulius & Cathal)

Firebase Cloud Functions were used to handle the back-end logic for the Walkie Talkie web application. The functions were implemented using JavaScript and were triggered by events such as the creation of a new user, the submission of user information, and the deletion of user data.

The cloud functions were used to perform various operations on the Firestore database, including creating new documents, modifying existing documents, and deleting documents. For example, when a user submitted their information, the cloud function would be triggered to create a new document in the Firestore database with the user's information.

The cloud functions were also used to integrate asynchronous functions to handle data retrieval and modification. For example, when a user submitted their information, the cloud function would use an asynchronous function to retrieve the list of other users who are travelling to the same location and are willing to share the cost of the taxi fare. The cloud function would then use this information to create a list of potential carpool matches.

Matching Functionality & Database Querying (Cathal & Stephen)

The matching functionality of the Walkie Talkie web application was implemented using a custom algorithm that used the user's location, destination, and time of travel to find potential carpool matches. To create this algorithm, the application used data stored in the Firestore database, including the location, destination, and time of travel of other users.

To retrieve this data from the database, the application used Firestore queries. These queries allowed the application to filter and sort the user data to find potential matches. For example, the application could query the database to find all users who are travelling to the same location and are willing to share the cost of the taxi fare.

The locations collection was indexed by the timestamp, allowing significant gains to be made performance wide, for accessing the database. The benefits of this approach were significant. By using queries to filter and sort the user data, the application was able to find potential matches quickly and efficiently. This reduced the amount of time and resources required to find a carpool match and improved the overall performance of the application.

Note – research and further development

At the start of the project / planning to develop past this demo project, we envisioned using regions to match users if they're mapped on a grid of 1x1km squares. Cathal tried using latitude and longitude to calculate this but then ran into the problem of how the earth is a globe.
The issue of 0.1 degrees changes in longitude at the equator being nowhere near what 0.1 degrees of change is at our latitude in Ireland is significant and gets quite severe the closer to a pole you get.
For example, at latitude of 53° and longitude of -9° (somewhere in Clare), a 0.1 change in longitude is 7 kilometres, versus a change of 11.2 kilometres at a latitude of 0° at the equator. This quickly gets a lot worse the closer to either pole you go.

Conclusions & Key Challenges
Roles & Teamwork

In September 2022, we formed as a team, named ‘Dumplings’. From there we began brainstorming ideas for our project.

We assigned rough roles based on our interests, with Paulius working on backend development, Stephen focusing on UI/UX design, Cathal working on front-end functionality, and Charlie on backend functionality.

As the project progressed, our roles changed significantly. Paulius took on a larger role in the backend, setting up the base app and producing testing functions. Cathal built off of Paulius's sample code, working on the location functions and getting the queries to work with the database. Charlie, with Stephen's help, worked on converting the pages into VueJS components and implementing a single-page design. Meanwhile, Stephen focused on implementing CSS for the app and using Bootstrap in certain places.

Cathal acted as our team lead and scrum master, while Charlie took on the role of product owner, providing input as a customer, helping us design desired features based on feedback. We aimed for biweekly sprints with weekly scrum meetings to check on progress, which usually took place on Mondays or Wednesdays. Although we didn't strictly adhere to this schedule, it worked well for us. Charlie provided input as a customer, helping us design the app based on desired product feedback.

For the final product, we only began building the app in week 6, laying the foundations for what was to come. It wasn't until weeks 8/9 that we really kicked into gear with the final project pieces. Reflecting, we should have all pushed forwards with the project a lot earlier as this put undue stress on all of us.
Throughout the project, we communicated effectively using WhatsApp to organize meetings and GitHub for version control. We tried using Jira and GitHub issues as tickets, but we found them too formal for our app development structure. We didn't use Discord for any file sharing at all 😀.

In conclusion, we worked well together as the Dumplings team, adapting to changes in roles and focusing on biweekly sprints to make progress. We communicated effectively using various tools and were able to build a successful app by the end of the project.