FindMyTradie

Title	Technical Guide
Project Name	FindMyTradie
Names & Student ID	Christian McAuley - 19353261 Bryan Rohan - 19432556
Programme	Computer Applications (Sft Eng.)
School	School of Computing
Year	4th
Emails	christian.mcauley6@mail.dcu.ie bryan.rohan3@mail.dcu.ie
Module Code	CA400
Supervisor	Hossein Javidnia

Table of Contents

1 Overview & Motivation

1.1 Explain the importance and relevance of the project

FindMyTradie is an innovative iOS mobile application designed to connect customers with tradespeople in their local area. The app streamlines the process of finding, hiring, and managing tradespeople for various projects. Customers can search for tradespeople based on their location and specific trade, such as electricians, plumbers, or carpenters. Each tradesperson's profile contains detailed information, including their skills, ratings, reviews, and verification status. This empowers customers to make informed decisions when selecting a professional for their job.

Customers can easily contact tradespeople through in-app messaging or by phone, and they can submit job requests directly through the app. Tradespeople, on the other hand, can view these requests, accept or reject them, and manage their ongoing projects. FindMyTradie also simplifies the payment process, tracking payment history and providing valuable analytics for tradespeople to monitor their business performance. This app is an excellent tool for tradespeople to grow their client base and gain new jobs without solely relying on word-of-mouth referrals.

1.2 Gap in the Market

FindMyTradie was inspired by the realisation that people frequently search for skilled tradespeople to complete various tasks but often struggle to find reliable professionals. Traditional methods, such as asking for recommendations or searching online, can be time-consuming and may not always yield the best results. FindMyTradie aims to eliminate the need for a middleman by providing a convenient and reliable platform for customers to directly connect with tradespeople. The app's user-friendly interface and extensive selection of tradespeople enable customers to easily find and hire professionals based on their specific requirements and preferences.

Currently, there is no application in Ireland that offers the comprehensive set of features provided by FindMyTradie. Existing solutions may only offer limited options for finding tradespeople, and they often lack the ability to manage payments and job requests efficiently. FindMyTradie addresses this gap by offering a one-stop solution for customers to find, hire, and manage tradespeople while also providing tradespeople with a platform to showcase their skills and grow their business.

By empowering customers to choose their own tradespeople based on ratings, skills, and past reviews, FindMyTradie ensures that they receive the best possible service for their specific needs. Furthermore, the app's integrated payment management system offers a secure and transparent way for customers to handle payments and track their payment history. This unique combination of features makes FindMyTradie a game-changer in the Irish market, bridging the gap between customers and tradespeople and revolutionising the way people hire professionals for their projects.

2 Research

2.1 Market Research:

We carried out extensive market research to see if there was a need for an app that bridges the gap between customers and tradespeople. We both come from a background where someone in our family works in the trade business so we felt there was a missing opportunity to make an app to help connect people with tradespeople. We both found that someone is always looking for a tradesperson (e.g. An Electrician), but they don't know where to look or who to ask. We felt people were still too heavily reliant on "word of mouth" rather than having an application to cut that out.

This research helped us identify the target audience and essential features needed to create a user-friendly app.

2.2 Competitor Analysis:

We reviewed existing competitor applications to determine their strengths and weaknesses.

We found some faults in their applications and websites. One major issue with some of the competitors we looked at such as an application called YourPro where you enter in the details of what you're looking for and it gives you a tradesperson, rather than allowing you to pick the tradesperson you want. Seeing issues like there helped us narrow down what we felt was important in this application.

This helped us figure out which features and functionalities to include, so our app would stand out.

2.3 Technical Research:

We researched the latest technologies, tools, and frameworks to ensure our app is made to a high standard. As a result, we decided to use React Native Expo for mobile application development.

Based on the research we conducted, we made the following design decisions:

2.3.1 User Interface:

We went with a clean, minimalistic, and easy-to-navigate user interface, enabling both customers and tradespeople to use the app effortlessly. This was important to us to have a good clean modern user interface because we believed that the first impression you get of an application is always how it looks first.

The application is aimed for age groups 18+ which means that the application will be used by older generations so it was important to keep it minimalistic and easy to navigate for users.

2.3.2 Sign-up and Authentication:

Our application has two distinct user types: tradespeople and clients. To provide a seamless and efficient experience, we've implemented separate account creation and authentication processes for both user groups.

Tradespeople's sign-up process collects essential information such as contact details, expertise, service areas, and credentials, allowing clients to evaluate their suitability for jobs. The authentication process for tradespeople may also involve verifying professional licenses or certifications.

Clients have a user-friendly sign-up process, providing their contact information, location, and necessary details for tradespeople to understand their service needs. Client authentication ensures identity and contact information validation, maintaining platform security and trustworthiness.

By tailoring sign-up and authentication processes to each group's needs, tradespeople can effectively showcase their skills, while clients can easily find and connect with suitable service providers, streamlining the experience for all users.

2.3.3 Search and Filter

Our research highlighted the need for an efficient search and filtering system in our application, allowing customers to easily find the right tradesperson. This feature is vital for boosting user satisfaction and meeting both clients' and trades people's needs.

The search and filter feature lets customers refine search results using criteria such as location, trade, and verification status. Location filtering helps customers find tradespeople within their area, ensuring availability and proximity, particularly for urgent or location-specific services.

Trade filtering allows customers to identify suitable tradespeople based on their expertise and required service, ensuring connections with skilled and experienced professionals.

The verification filter enables customers to search for verified tradespeople with valid professional licenses, certifications, or credentials. This feature builds trust and confidence, assuring clients that hired tradespeople are qualified and reliable.

2.3.4 Job Management

In our research, we determined that incorporating job management features in our application is crucial for enabling tradespeople to efficiently handle their jobs and effectively communicate with customers. These features are essential for streamlining work processes, enhancing user experience, and fostering positive client relationships.

The "accept job" functionality allows tradespeople to confirm their availability and commitment to a specific job, establishing a clear understanding of responsibilities and timelines for both parties. The "decline job" feature enables tradespeople to refuse jobs they cannot accommodate due to scheduling conflicts or other limitations, thus assisting them in managing their workload and maintaining open communication with clients.

Furthermore, integrating payment references facilitates tradespeople in tracking the financial aspects of their jobs, ensuring precise records of completed tasks and corresponding payments. This feature not only simplifies financial management for tradespeople but also builds trust between them and their clients by providing transparent transaction records.

2.3.5 Reviews and Ratings

We need a review and rating system that offers customers valuable insights into the quality of work provided by the tradespeople. We felt this was essential because it’s important that the clients are aware of the tradesperson's previous work and what people thought of them.

2.3.6 Analytics

Including an analytics page for tradespeople to view their statistics, such as past jobs, current jobs and monitor their total income processed through the app will allow Tradespeople to track their business performance.

3 High-Level Design (Flow / High-L Design Diagram)

3.1 Frontend (React Native Expo):

3.1.1 User Interface Components:

These are the visual elements that users interact with, such as forms, buttons, and lists. We made use of modals in particular which allowed us to give the frontend a professional look when users interact with certain features that cause a modal to appear prompting them to make a decision.

3.1.2 Navigation:

We used a mix of bottom Nav bars and Drawer navigation for both sides of the application this allowed us to ensure that the user could navigate to any part of the application with ease.

3.1.3 Customer Side VS Tradie Side:

We split frontend into two parts, one for the Customer and one for the Tradie, this was essential because the functionalities for Customer and Tradie differ.

3.2 Backend (Firebase):

We decided to use Firebase as the backend for our application as it offers several advantages compared to creating a backend from scratch with something like Node.js.

Firebase provides a ready to use backend service which allowed us to focus more on the application features rather than the complexities of configuring servers etc. This saved us a lot of time and was a crucial decision as it allowed us to create a really nice User Interface.

Firebase offers a few integrated features which we made use of:

Authentication: This component manages user authentication, including registration, login, and password management. Firebase looks after the Authentication process and keeps users' information safe and secure.

Firestore: Firestore was used to store and manage all user data, jobs data, payment references, and reviews.

Storage: Firebase Storage is used to store profile images and verification documents. Each profile picture or document is linked with the user's ID.

3.3 Key Features:

3.3.1 Sign Up:

Signup for both Customers and Tradespeople. The Tradespeople sign up is more in depth and interactive as we require more information in order to get them set up. We require information about their trade and skills and also their location and travel distance.

3.3.2 Search Functionality:

Allows customers to search for tradespeople based on their specific trade and location and also filter the results by rating or verified tradespeople. This allows users to narrow search based on preferences, they can then view each individual tradies profile to decide who they would like to get in touch with.

3.3.3 Job Management:

Tradespeople can accept and decline jobs, while customers can track the status of their jobs.

3.3.4 Payment Management:

Tradespeople can create payment references with the payment method and value, and customers must approve them upon job completion in order for the payment to be saved.

3.3.5 Reviews and Ratings:

Customers can leave reviews and ratings for tradespeople once they have been a customer of that tradesperson, these reviews are visible to other users when searching for tradespeople.

3.3.6 Analytics:

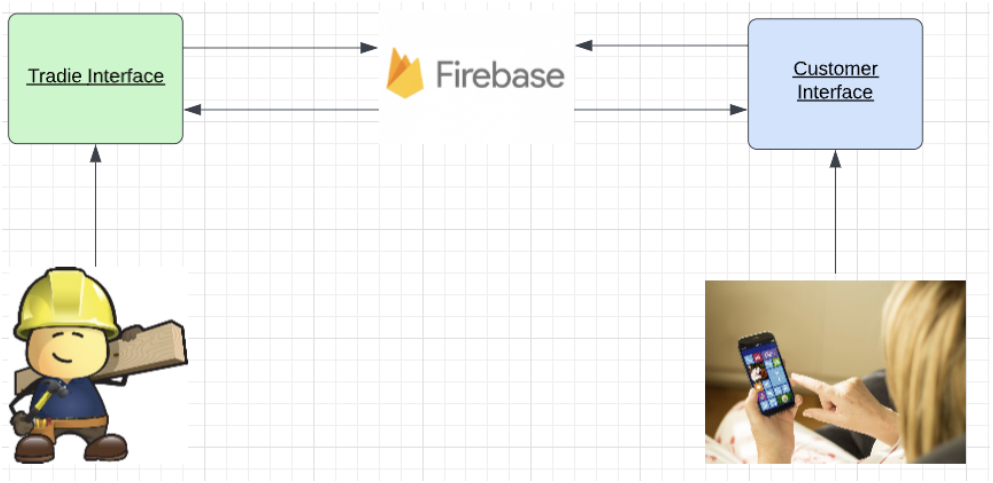
Tradespeople can access an analytics page to view data on their total jobs, ongoing jobs, and total income processed through the app. The Tradespeople can also see a line graph of the past five months of their “Jobs per Month” showing what months had a higher average than others.

3.3.7 Tradesperson Verification:

When a tradesperson signs up they will be prompted to upload a document proving their qualifications, upon a review from an admin they're account will either be verified or not which will be indicated by a blue tick beside their name.

3.4 High Level Diagrams

The system operates by communicating with firebase from the user interface specific to their account type. Each Account type has a separate set of screens that are unique to that account type. They communicate with firebase to update/add/delete anything on the application. Example: If a user requests a job from John Smith, the users Request screen will automatically update and contain the requested job and on John Smiths Tradie home screen he will see the request come through.



Basic System Architecture

3.4.1 Logging In (Sequence Diagram)

Fig 1 is a sequence diagram showing how the users of the application log in, users a prompted to enter their email address and password and then press log in, when login is pressed Firebase will check if the user has an account and If its a success they will be navigated to the Customer Home Screen if "isTradie == False" otherwise they will be navigated to the Tradie Home Screen. This process is also described in the Login Flow diagram (Fig 8).

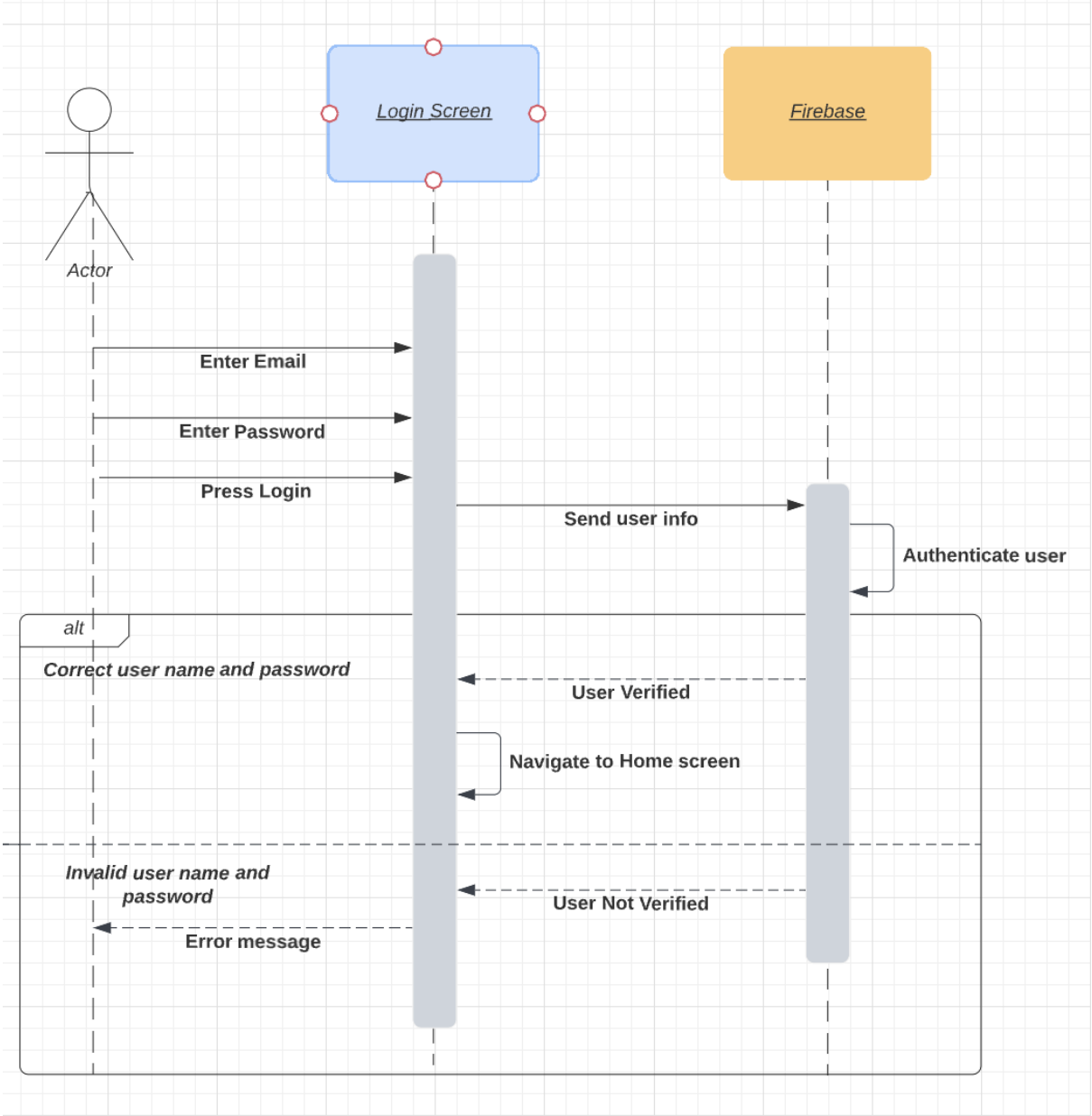


Fig 1 Login Screen (Sequence Diagram)

3.4.2 Search for Tradespeople on Customer Home Screen (Sequence Diagram)

Fig 2 shows how customers interact with the home screen to search for Tradespeople and request a job from them after selecting a Tradesperson.

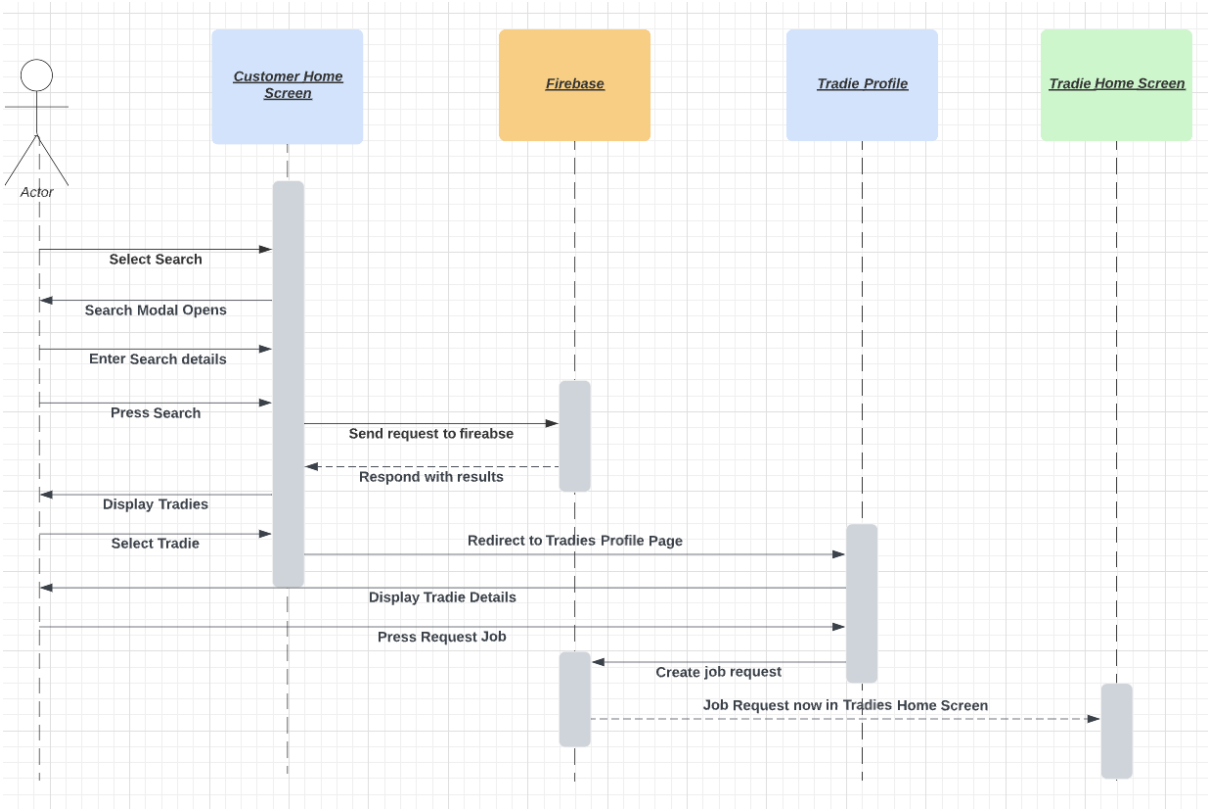


Fig 2 Customer Home Screen (Sequence Diagram)

Once the customer is satisfied with the search results and has visited the profiles of the tradies they can select a tradie and request a job, this will send a request to the tradie which they can then accept or decline the job (assuming that prior communication has occurred through our message or call buttons).

3.4.3 Tradie Home Screen (Sequence Diagram)

Fig 3 is a high level sequence diagram that illustrates the Tradespersons interaction with their side of the application like viewing job requests, current jobs and previous jobs. It shows how they interact with accept and decline buttons and also how they can finish the current jobs and record a payment reference.

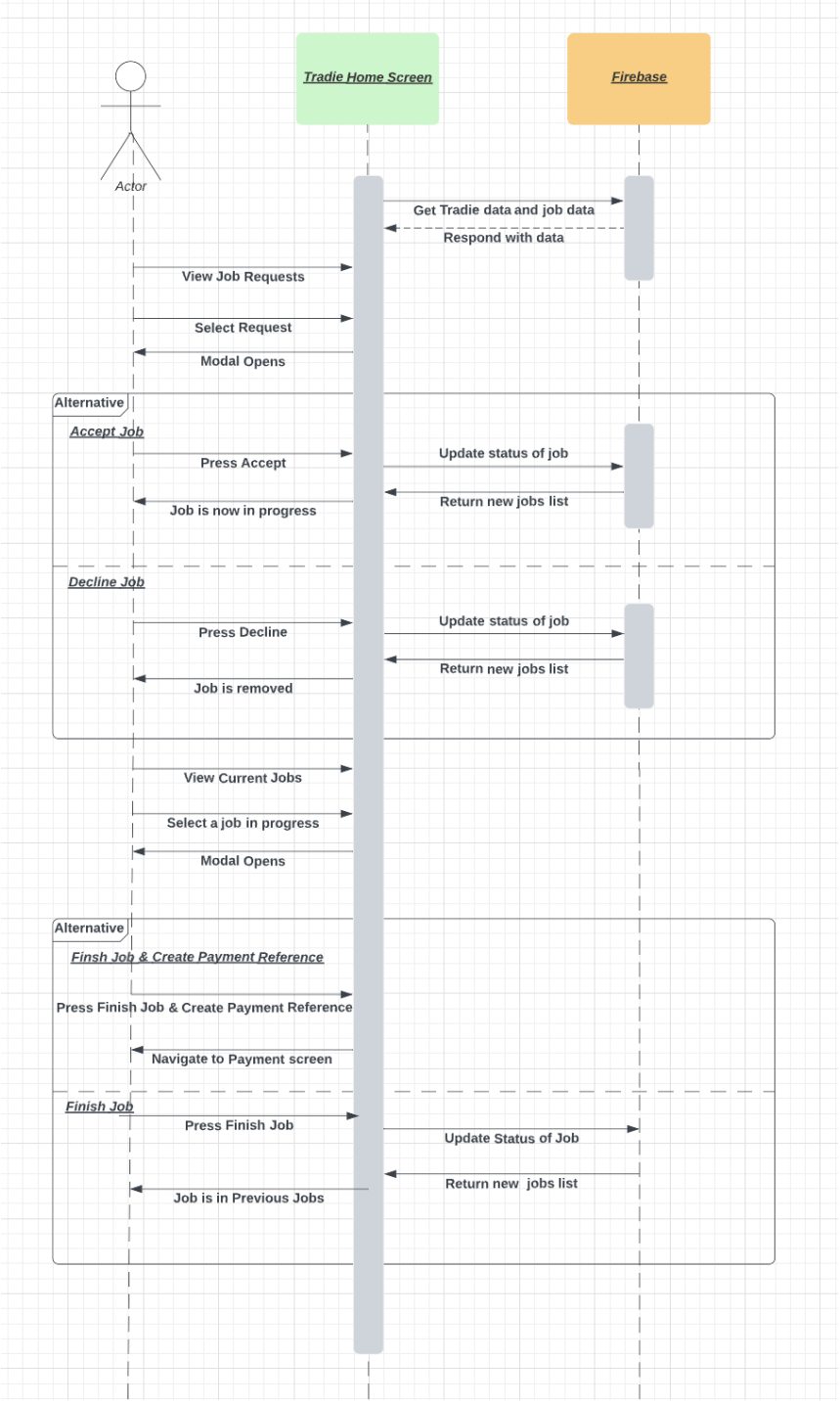


Fig 3 Tradie Home Screen (Sequence Diagram)

3.4.4 Tradie Payment Reference Creation (Sequence Diagram)

Fig 4 is a more detailed diagram showing how Tradespeople interact with the payment section of the application. They can create a payment reference with the amount, the client and the payment method, the payment is not saved until the client confirms the payment has occurred.

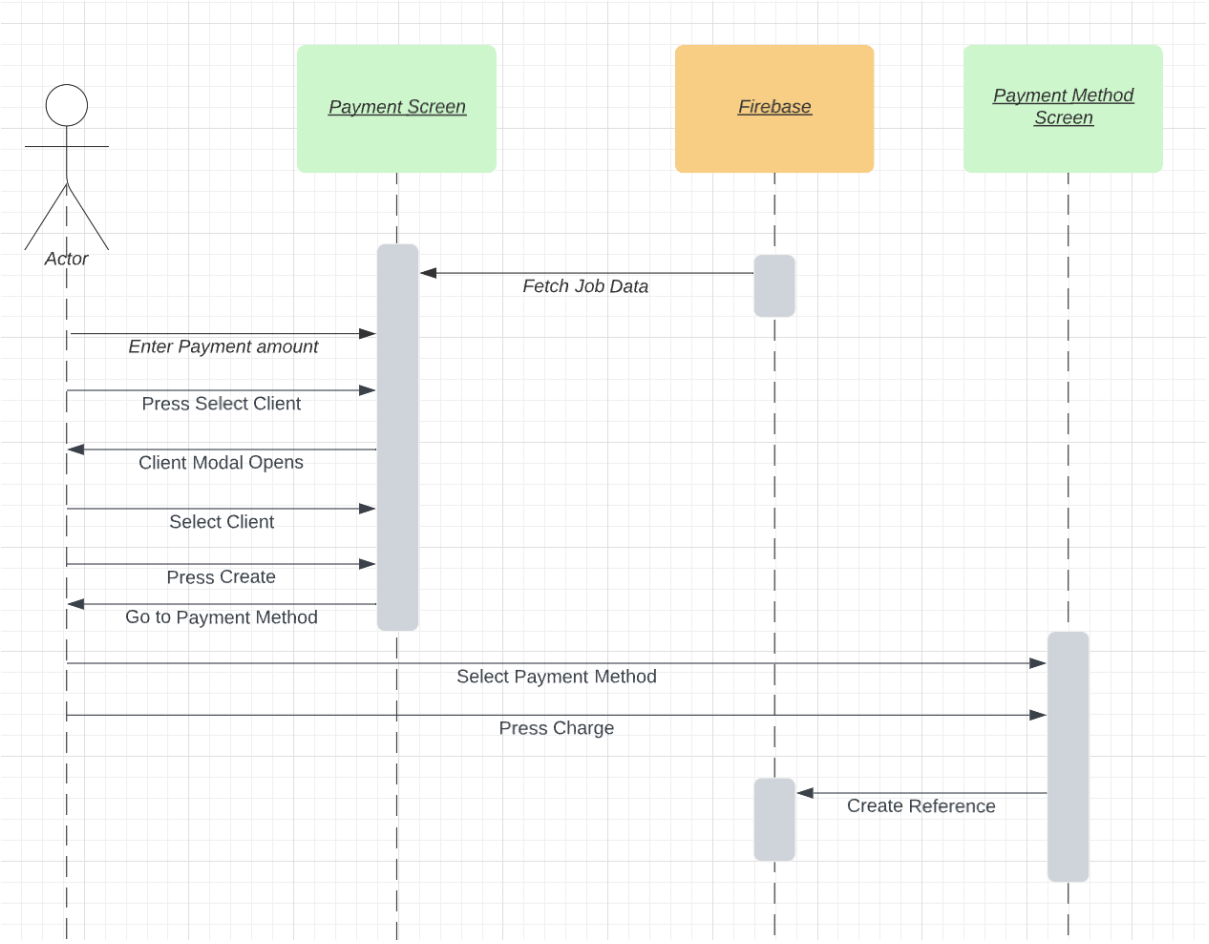


Fig 4 Tradie Payment Reference Creation (Sequence Diagram)

3.4.5 Customer Search & Request Job (Data Flow)

Figure 5 is a high level data flow diagram showing the customer interacting with the processes that allow customers to search, select and request different tradespeople.

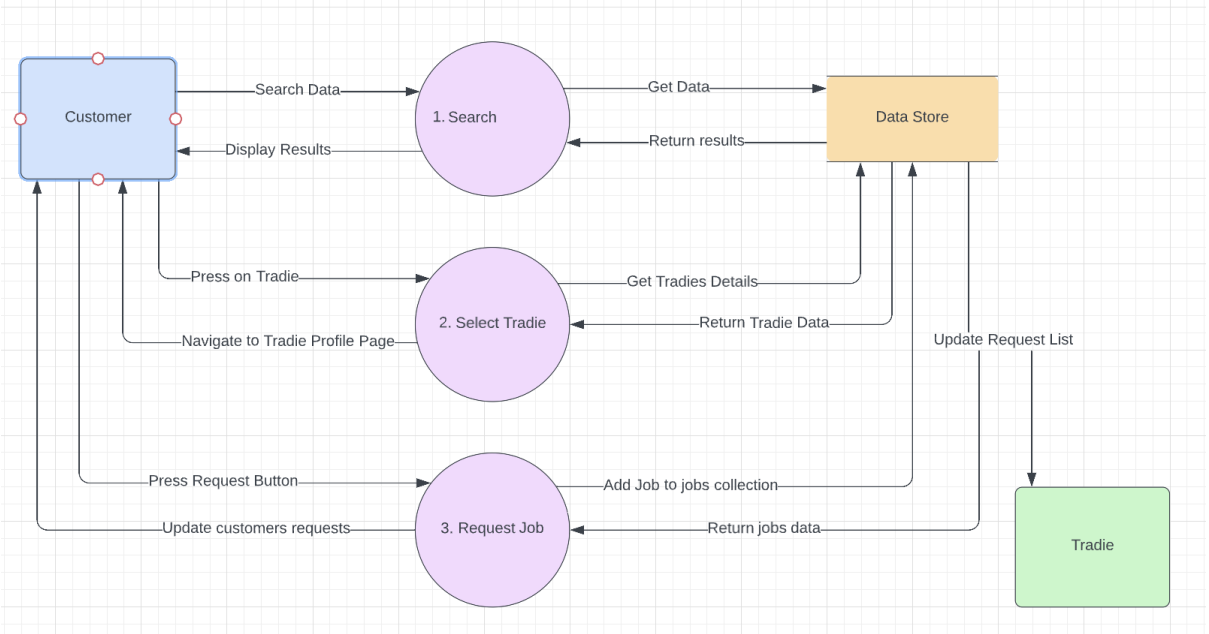


Fig 5 Customer Search and Request Job (Data Flow)

3.4.6 Tradie Accept

Figure 6 is a high level data flow diagram that shows how the Tradesperson interacts with the job handler process in order to accept or decline job requests made by customers.

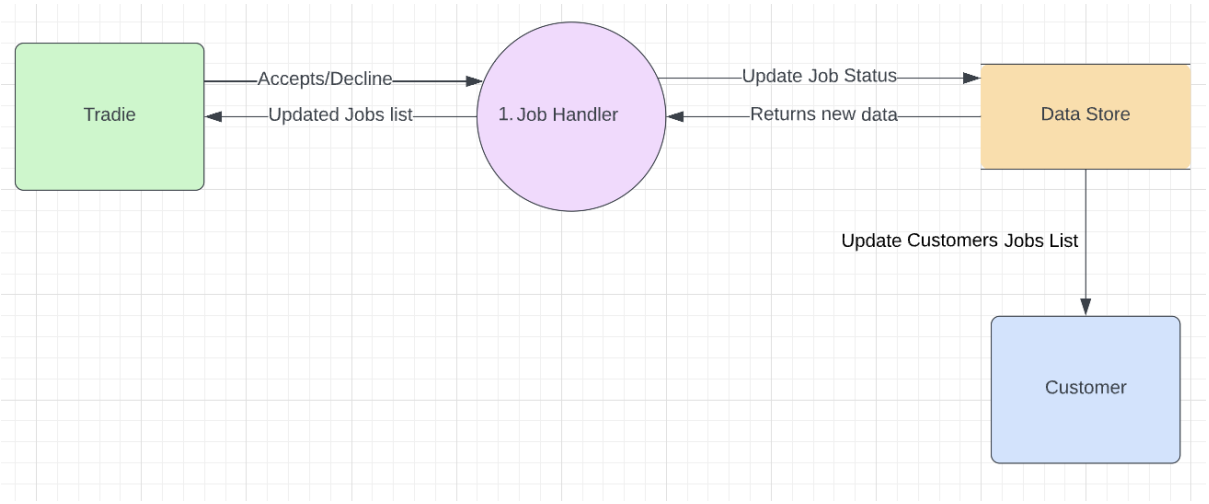


Fig 6 Tradie Accept/Decline Job (Data Flow)

3.4.7 Customer & Tradie Sign-Up (Flow Diagram)

Fig 7 shows the steps involved in registering for Customers and Tradespeople in the Sign up section.

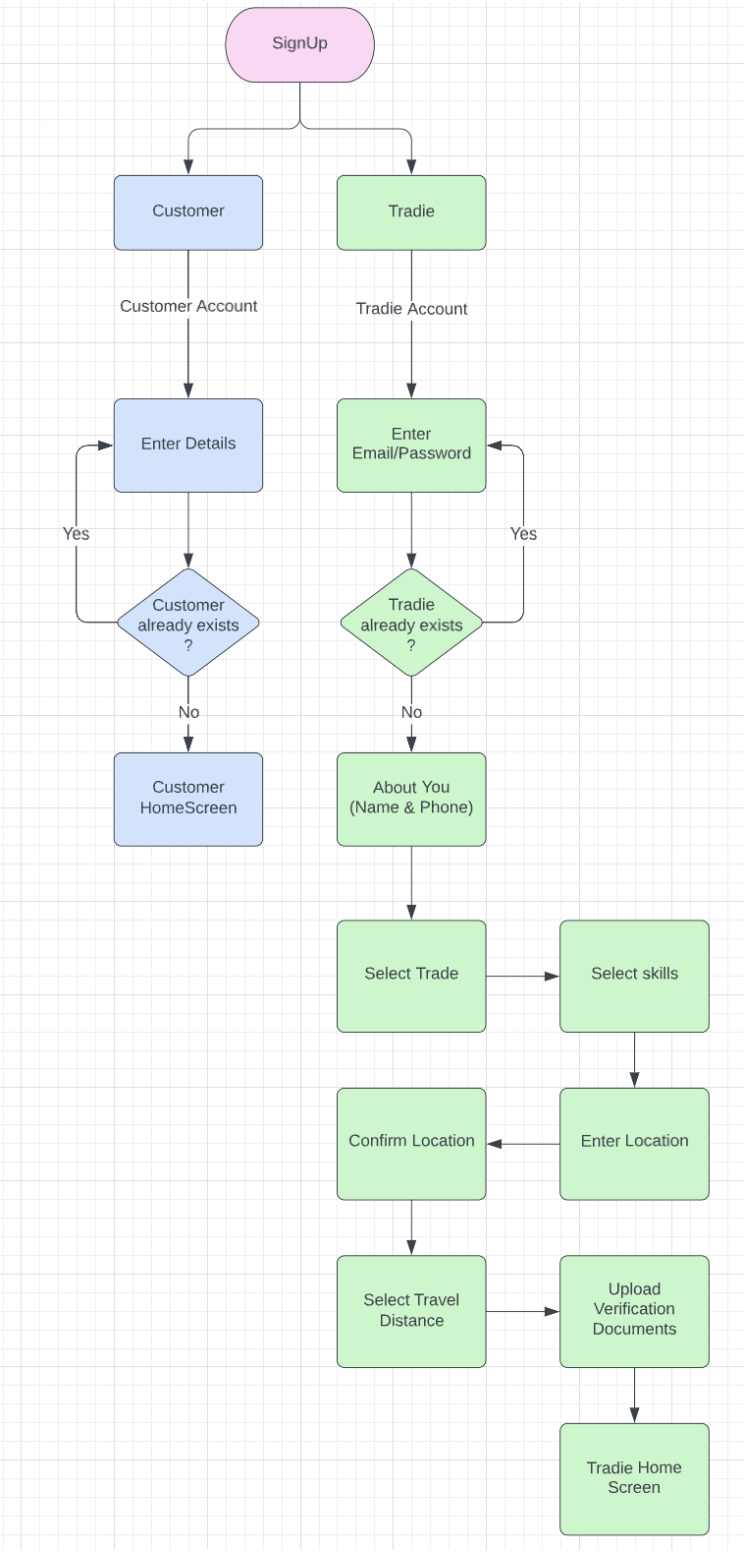


Fig 7 Sign Up (Flow Diagram)

3.4.8 Logging In (Flow Diagram)

Fig 8 shows how users interact with the login screen in order to gain access to their relevant accounts.

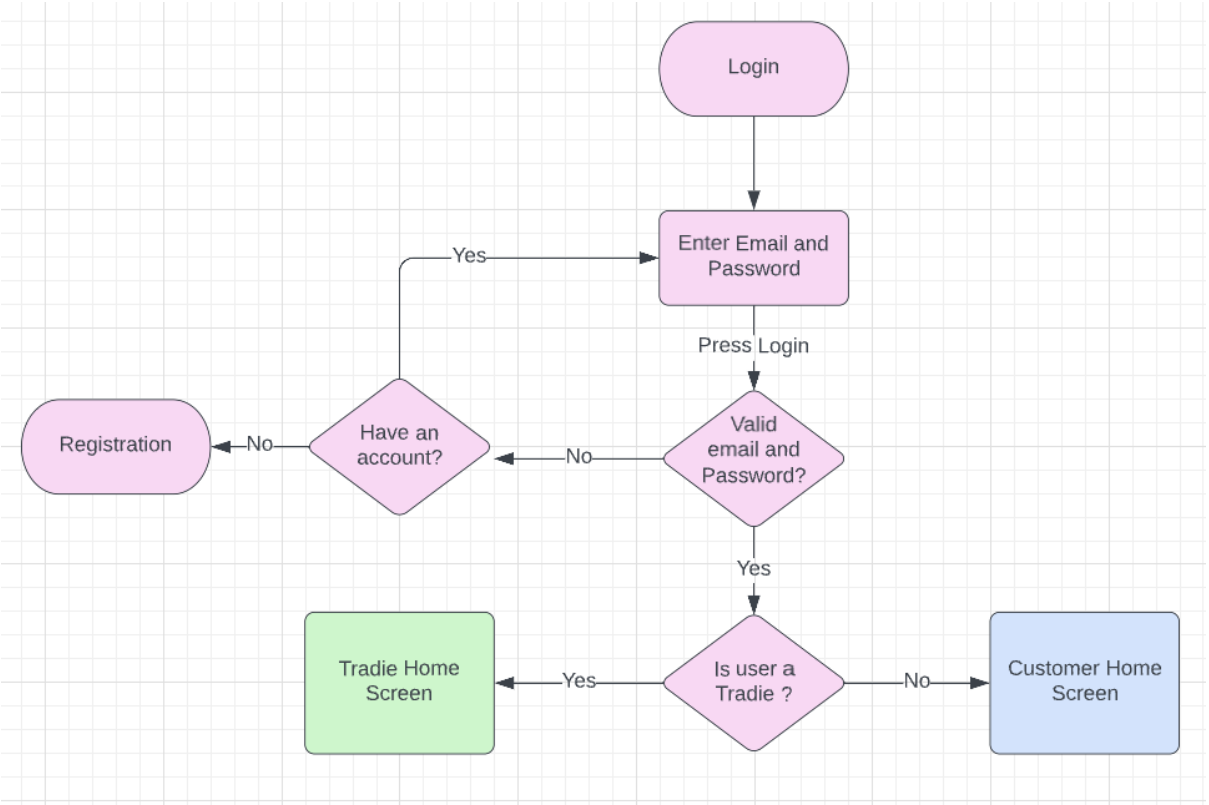


Fig 8 Login (Flow Diagram)

3.4.9 Use Case 1: Customer Search

The use case diagram below shows the relationships between different processes that a customer uses when searching for tradespeople.

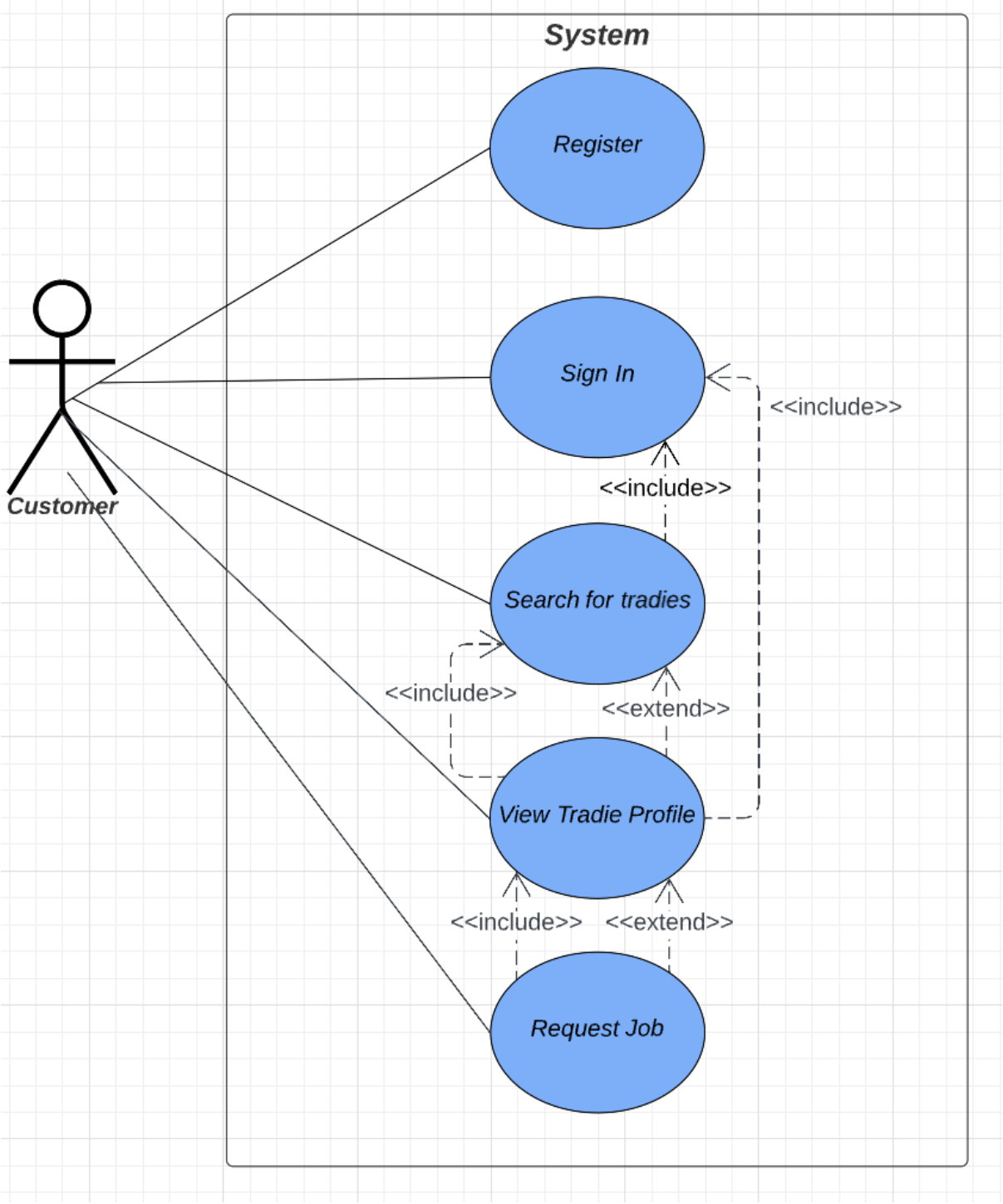


Fig 10 Use Case 1 (Customer Search)

3.4.10 Use Case 2: Tradie Monitoring Jobs

The use case diagram below is a high level depiction of the Tradesperson using their side of the application.

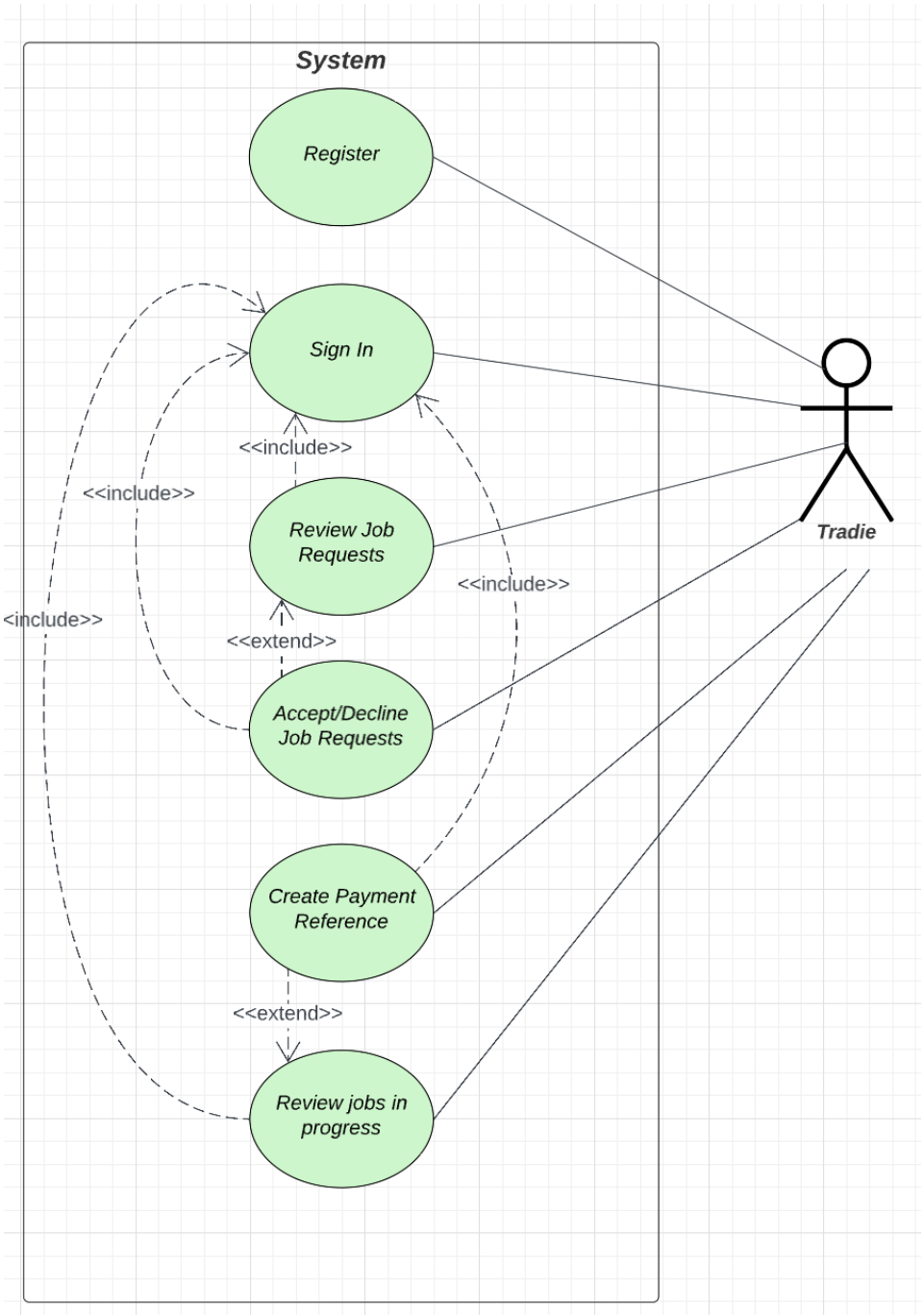


Fig 11 Use case 2 (Tradie monitoring jobs)

The diagram below shows the different processes available to Tradepeople when they want to edit their account details.

3.4.11 Use Case 3: Tradie Editing Profile

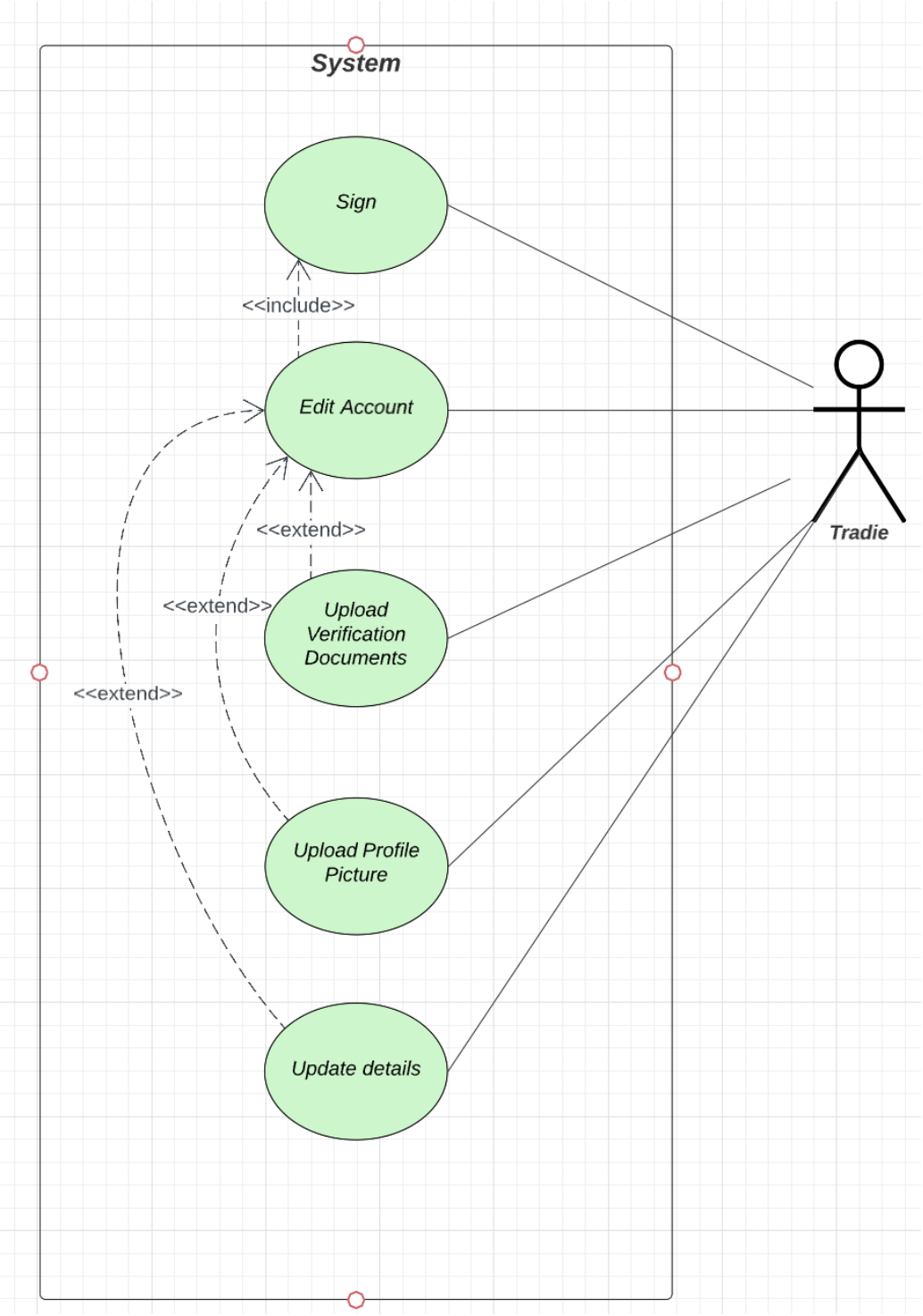


Fig 12 Use case 3 (Tradie Editing Profile)

4 Implementation

4.1 Programming Languages and Tools

4.1.1 Programming language(s):

JavaScript: We used this as it is the primary language used in developing React Native applications.

4.1.2 Libraries:

1) React Native, This open-source framework, developed by Facebook, enabled us to build a mobile application using JavaScript and React. React Native allowed for faster development times and consistent user experience.

2) Expo, This is a set of tools and services built around React Native that simplifies the development process. With Expo, we had access to a variety of built-in components and APIs that extended the functionality of the application without needing to deal with native code.

3) React Navigation, a library for managing navigation and routing in React Native applications, React Navigation helped us create navigators, such as stack navigators and tab navigators, handle screen transitions, and manage navigation state.

4) Firebase SDK, a set of libraries that provide access to various Firebase services, such as Authentication, Firestore, and Storage. These services enabled user authentication, real-time data storage and retrieval, and file storage capabilities in the application which was essential for our project.

5) React Native Google Places API, this library allowed for easy integration with the Google Places API, which helped us create features like searching for places, retrieving place details, and displaying maps within the application.

4.1.3 Services:

1) Firebase Authentication - This was used to authenticate users when signing into the application. It simplified the authentication process and user management within the application.

2) Firebase Firestore - This NoSql database service enabled real time data synchronisation and storage for the application, it allowed us to retrieve data easily and was a suitable choice for a dynamic application like FindMyTradie.

3) Firebase Storage - this file storage service enables us to store and retrieve user-generated content, such as user profile pictures and verification documents.

4) Google Places API - this API provided detailed information on locations entered by users allowing us to accurately calculate distances between Tradies and Customers, which was essential for our search functionality.

4.1.4 Dev Tools:

1. Node.js
2. Npm
3. Expo CLI
4. VS Code
5. Gitlab

4.2 Sample Code

4.2.1 Calculating Distance between Customer and Tradies Location Code

This function is in CustHomeScreen.js and it queries the Firestore database for Tradies with the selected trade it then calculates the distance between the customers entered location and the tradies location, and filters the results based on the tradesman's work radius. The filtered data is then passed to the SearchResultsScreen for display.

4.2.1 Job Request Query Code

RequestsScreen.js

```
useEffect(() => {
  const jobRequestQuery = firebase
    .firestore()
    .collection("jobs")
    .where("customerId", "==", firebase.auth().currentUser.uid)
    .where("status", "==", "Requested")
    .orderBy("createdAt", "desc");

  const jobCurrentQuery = firebase
    .firestore()
    .collection("jobs")
    .where("customerId", "==", firebase.auth().currentUser.uid)
    .where("status", "==", "accepted")
    .orderBy("createdAt", "desc");

  const jobPreviousQuery = firebase
    .firestore()
    .collection("jobs")
    .where("customerId", "==", firebase.auth().currentUser.uid)
    .where("status", "==", "completed")
    .orderBy("createdAt", "desc");

  const unsubscribeRequest = jobRequestQuery.onSnapshot((snapshot) => {
    const jobRequestData = snapshot.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    setRequestJobs(jobRequestData);
  });

  const unsubscribeCurrent = jobCurrentQuery.onSnapshot((snapshot) => {
    const jobCurrentData = snapshot.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    setCurrentJobs(jobCurrentData);
  });

  const unsubscribePrevious = jobPreviousQuery.onSnapshot((snapshot) => {
    const jobPreviousData = snapshot.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    setPreviousJobs(jobPreviousData);
  });

  return () => {
    unsubscribeRequest();
    unsubscribeCurrent();
    unsubscribePrevious();
  };
}, [firebase.auth().currentUser.uid]);
```

This hook in RequestsScreen.js initialises three queries for job requests, current jobs, and previous jobs, sets up real-time listeners for each query, and updates the app's state with the fetched data ensuring that the Customer can see the correct information regarding their jobs at any time.

4.2.3 Analytics Code

Analytics.js

```
React.useEffect(() => {
  const userId = firebase.auth().currentUser.uid;
  const jobsRef = firebase.firestore().collection("jobs");

  const fetchJobCounts = async () => {
    const snapshot = await jobsRef.where("tradesmanId", "==", userId).get();
    let completed = 0;
    let accepted = 0;
    let requested = 0;
    const uniqueClients = new Set();

    snapshot.forEach((doc) => {
      const status = doc.data().status;
      const customerId = doc.data().customerId;
      uniqueClients.add(customerId);

      if (status === "completed") {
        completed++;
      } else if (status === "accepted") {
        accepted++;
      } else if (status === "Requested") {
        requested++;
      }
    });

    setCompletedJobs(completed);
    setAcceptedJobs(accepted);
    setRequestedJobs(requested);
    setUniqueClients(uniqueClients.size);
  };

  fetchJobCounts();

  const unsubscribe = jobsRef
    .where("tradesmanId", "==", userId)
    .onSnapshot(() => {
      fetchJobCounts();
    });

  return () => unsubscribe();
}, []);
```

4.3 Integration

JavaScript served as the core programming language for developing the application. It is used to write components, handle logic, manage states, and interact with external services.Several libraries and frameworks are used to provide essential functionality. React Native serves as the foundation for building mobile applications, while other libraries like React Navigation and Firebase SDK enhance the app with navigation, authentication, and data storage capabilities.The application relies on external services, such as Firebase Authentication, Firebase Firestore, and Google Places API, to handle user authentication, real-time data storage, and location-based features. Integration with these services is achieved using their respective SDKs, libraries, and APIs.

Gitlab was used as our Version Control System, we made sure to develop separately from the master branch, for each sprint we would branch off the master and work on our respective issues. Once the sprint was complete and our issues were complete, we would push to Gitlab and create a merge request and after careful reviews the request is approved and merged into master.

4.4 Pair Programming

Despite living far away from each other, we successfully relied on pair programming over Zoom to complete the project. Pair programming is a technique where two developers work together on the same task, with one person writing the code while the other reviews the code in real-time. This collaborative approach increased code quality,problem-solving, and aided in faster development.

Zoom allowed us to effectively communicate, share our screens, and collaborate in real-time as if we were sitting side by side.

We followed a structured approach for our pair programming sessions. Before each session, we would discuss the tasks to be completed and set clear objectives. During the session, one of us would share our screen and write the code, while the other would review the code and provide suggestions or help troubleshoot any issues. We would periodically switch roles to ensure that both of us were actively engaged and contributing to the project. We made sure to commit our changes regularly and update the branches on gitlab.

4.5 Agile Development

4.5.1 Sprints

We met with our project supervisor every 2 weeks, so we set our sprints to start on the day of a sprint and end on the day of the next meeting. This allowed us to have a focus for the 2

week period, with certain tasks that must be completed within that time frame. Once we finished the meeting with our supervisor we planned our tasks for the next sprint and also reflected on the previous sprint and prioritised any issues that did not get completed on time.

4.5.2 Jira Tasks

During our Sprints we created Jira tasks that needed to be completed by the next sprint. This allowed us to see what we were both doing during each sprint and it kept us on track throughout the project.

5 Problems Solved

5.1 Challenges Encountered

5.1.1 Initial Set Up

The initial setup of the project was difficult, originally only one of us had a mac and we wanted to create an IOS app so it was difficult to organise the development environment on Windows, thankfully we both ended up with MacBooks which gave us the opportunity to work with identical environments throughout the project.

5.1.2 TailWind

At the start we wanted to use TailWind CSS but we spent a lot of time trying to resolve dependency issues so we decided to move on without it, although TailWind CSS is great, time was being wasted so the decision to move on was crucial.

5.1.3 MongoDB

We didn't plan on using Firebase from the start, the original plan was to use MongoDB but the complexity involved in setting up a backend etc made Firebase stand out as it is much easier to use and setup, especially since the main focus for this project is the UI.

5.1.4 Search Feature

The calculation for measuring if a user is within the set travel distance of a Tradie was challenging but after plenty of research and trial and error we were able to get it working correctly, this functionality is important for the application as there is no point in a plumber from Mayo getting job requests from customers in Dublin.

6 Future Work

6.1 Short-term Goals

Although we are thrilled with the final product we've created, we know there's always room for improvement. We found a few small details that could be improved to make the application even better.

1) One significant improvement would be the implementation of push notifications. Currently, tradies are not immediately notified when they receive a job request. By integrating push notifications, tradies would be alerted instantly on their phones, enabling them to respond more promptly to potential clients. Similarly, customers would receive notifications when their job requests are accepted or when a payment reference is awaiting approval. This feature would enhance the user experience by keeping both tradies and customers informed and up-to-date on the status of their jobs.

2) Another area that could be improved is the loading speed of profile pictures. At times, there is a noticeable delay in loading images, which could be frustrating for users. By optimising the code and employing techniques such as image compression, caching, and lazy loading, we could significantly reduce the loading time and provide a smoother experience for users.

3) Currently, communication between tradies and customers occurs outside of the application. Introducing an in-app messaging system would streamline communication and make the entire process more seamless and efficient. By developing a chat platform within the app, users could easily send messages, share files, and discuss job details without having to switch between different applications. This would not only improve the user experience but also help maintain user engagement within the app.

4) Another feature we would like to include in future work is that tradies can upload photos of jobs they have completed onto their profile page to show the client their work. This would give the client a better idea of the tradies' quality of work.

6.2 Long-term Goals

1) We would like to have the application running on all platforms not just IOS, an Android and Web Application would increase the possible target audience and increase the overall use of the application in a business context.

2) Create the same application but configure it for different countries, at the moment that application only works for the Republic of Ireland.

3) Currently, the application allows Tradies to create a payment reference but does not directly handle payments. Adding an integrated payment processing feature would improve the user experience allowing Tradies to handle transactions within the application. To achieve this, we could integrate with payment gateways such as Stripe or PayPal.

7 Conclusion

7.1 Lessons Learned

Developing functional mobile applications is difficult and takes a lot of planning before development. The logistics behind creating an application that can allow different users to interact with ease can become quite complicated and confusing. It's important to take note of everything you do during development because it's easy to forget why you added a certain piece of code.

7.2 Final Thoughts

Completing this project was by no means easy, but it gave us an opportunity to try something new and build a mobile application of our own from scratch. We feel like this project is a great way to showcase all of our abilities and strengths gained over the course of this degree. We learned valuable lessons throughout which will help in future endeavours.