



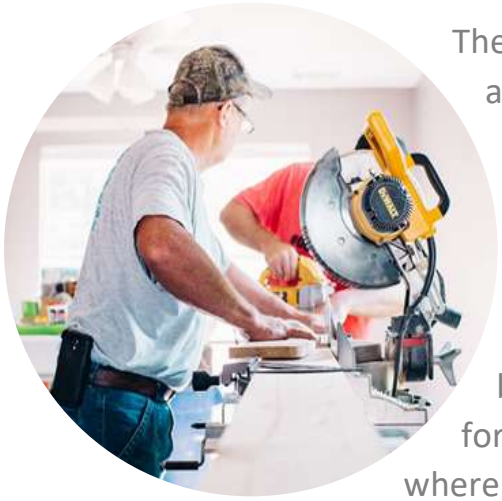


Group 16 | Final Report

Asad Khan, Amandeep Bharj, Daryas Hussein, Shakil Ali, Tanveer Hossain, Ye-Seul Kim

Labora

Abstract



The demand for services is increasing day by day. We are in a world where people have become reliant on other individuals doing jobs for them. The concept of 'D.I.Y' is becoming obsolete due to people being busy majority of the time. Finding services urgently on websites such as Checkatrade.com or even Yellow Pages is difficult because workers are often busy due to their heavy schedules. A competitive job for workers making it difficult for start-ups. This is where our idea of workers (contractors) finding customers would help these situation. We want to develop an app that is catered to solve these problems.

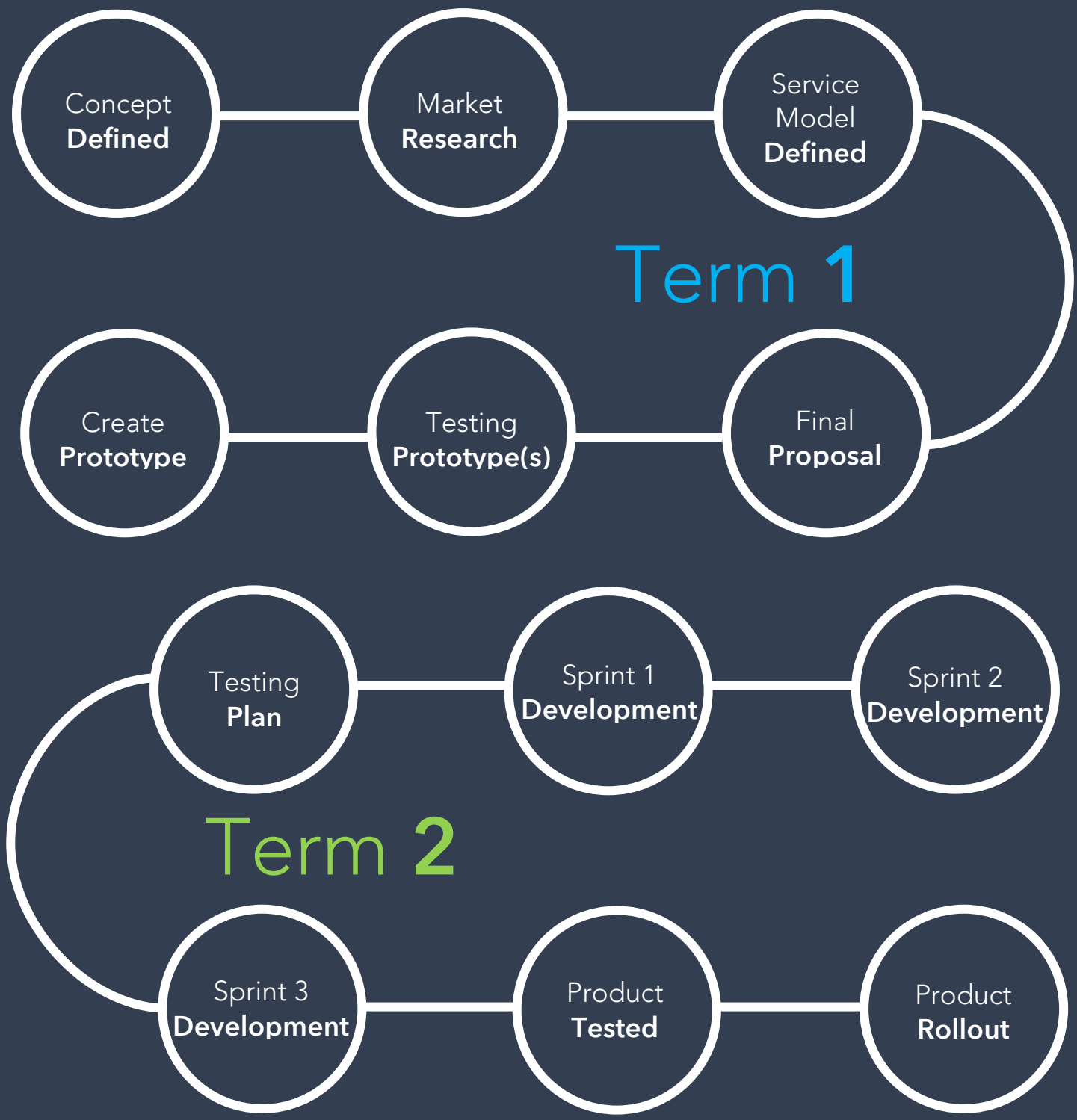
Acknowledgements

We would like to thank our supervisor Esben Sørig for his guidance and knowledge during the course of this project. We appreciate the time taken during meetings with our supervisor and our peers.

We would also like to thank the Department of Computing at Goldsmiths, University of London for giving us the opportunity to work on this project and assigning us into groups.

And finally special gratitude go to our group members collectively have worked hard to ensure we have made a success of this project.

Timeline



Labora

Contents

1. Introduction 5
1.1 Incentives 5
1.2 Purviews & Aims 5
1.3 Report Structure 6
2. Literature Review 6
2.1 Background Research 6
2.2.1 Android Studio 7
2.2 Application Scheme – Android Studio, Firebase 7
2.2.2 Firebase 9
3. Development Process 10
3.1 Kanban - Trello 10
3.1 Sprints & Scrum 11
3.2 Management of Group Members 12
3.3 Methodologies 12
3.4 GitLab and Local Storage 13
4. Analysis 14
4.1 Requirements (Functional and Non-Functional) 14
4.1.1 Functional 14
4.1.2 Non-Functional 15
4.2 Switch to two applications 16
4.3 Adapted Technical Architecture 16
4.4 Stakeholders – Contractor and Requesters 18
5. Design 18
5.1 Activity Diagrams 18
5.2 Sequence Diagram 21
5.3 Database Design 23
5.4 Prototype 23
5.5 Code Design 27
6. Implementation 28
6.1 Front End 28
6.1.1 Login Page 29
6.1.2 Register, Profile & Confirmation 30

6.1.2 Services Page 32
6.1.3 Map Page 33
6.2 Back End 33
6.3 Unanticipated Complications & Resolving Solutions 34
7. Testing, Evaluation & QA 35
7.1 Formative Evaluation – (Introduction) 35
7.2 User Testing 36
7.2.1 Test Cases 36
7.2.2 Black Box 36
7.2.3 White Box 37
7.3 Unit Testing 37
7.4 Integration Testing 38
7.5 Systematic Testing 39
7.6 Robo Testing 39
7.7 How well it conforms 39
7.8 Constraints 40
7.9 Functional Requirements Evaluation 41
7.10 Non-Functional Requirements Evaluation 41
8. Conclusion 42
8.1 Summative Evaluation 42
8.2 Future Developments Strategy and Plan 44
8.3 Overall Conclusion 44
9. Appendix 45
10. Bibliography 133

Link to **Gantt Chart, Tracking Sheet and User Guide**:

https://drive.google.com/drive/folders/1avz04rd1nrcGhxv_cibi6T2Nh8DJuIbF

1. Introduction

41% of self-employed are **freelancers in 2017**

Office for National Statistics, IPSE

1.1 Incentives

The UK has seen a substantial rise in self-employment, current figures indicating an estimated 5,000,000 self-employed individuals. Freelance workers have reached around 2,000,000 in the UK since 2001, the largest growth seen by the country in its history. In addition, the unemployment level in the UK has fallen to 4.1%, this is the lowest recorded since 1975

As a result, this paints the picture that the number of self-catered employment opportunities has risen in the UK and is very likely to continue doing so. Consequently, we also see that we, as a nation, are becoming much more technologically advanced. Therefore, almost certainly freelance employment and technology will cross paths, and will need to synchronise to achieve maximum output. Contractor's in other words, to exploit modern day technology to their advantage, will need a platform whereby they can gain access to work or publicise themselves seeking work. This is where we believe our project idea can seal the gap in the market. It consists of two applications: Labora Service Requester Application, for people requiring a service to be completed and Labora Contractor Application, for people providing the service.

1.2 Purviews & Aims

Our main aims for both Labora applications is to cater for and allow efficient exchange of services between two sets of users: People with a service they need carried out and people with the skills to carry out those jobs. The focal point of our operation was to make sure that both applications were visually identical in terms of layout, yet, still accessible and usable to people in said user groups. Our software for the service requester app must ensure that it can easily register/login users and take them to an area where they can submit a form for their desired service. Correspondingly, the contractor software must allow users to login/register as well, however, take them to a map wherein they can select the jobs that appeal to their expertise.

1.3 Report Structure

Section 1 (This section) - Introduction: Introduction to our project

Section 2 - Literature review: Reviews the background work already conducted on our project and the application schema

Section 3 – Development Process: The process of development described for our project

Section 4 – Analysis: Analysis of the functional and non-functional requirements, as well as description of technical architecture adaptation

Section 5 – Design: Aims of this section are to go through how we designed each component of our application from the application to the database

Section 6 – Implementation: The stages we went through when implementing our front-end and backend. Includes complications experienced

Section 7 – Testing, Evaluation and Quality Assurance: All various tests conducted, to ensure quality, and the evaluation we came to from this testing

Section 8 – Conclusion: Summarises the accomplishments of the project as well the future development plans

2. Literature Review

2.1 Background Research

Once research began on our project, we had to investigate whether prior work had been conducted in the field of service requesting (requiring a service), and service providing (contractors' providing a service). We used Google scholars to pinpoint any research into service requesting applications and found that no considerable academic research has been done regarding it (**refer to pg 45 appendix**) we hypothesised that there may be a technological barrier to entry. To get an insight into who our target audience is and what technology people of certain age groups are used to, we created the table below.

We see how business directories are very outdated, and today in some places inexistent. However, websites (1990-99), and more recently mobile applications (2000-09) have taken over. Consequently, we found that if we decided to create our service request and contractor application, we will need to develop it in a form which is accessible, whilst still functional. This may ensure that our potential stakeholders (service requesters and contractors) can utilise the app both effectively and efficiently. Thus, possibly providing a solution to service requester and contractors.

Before 1910	1990-99	2000-09
Vacuum Cleaner	Scanning Systems	MP3
Air Conditioning	Website	Wikipedia
Electrocardiogram	MEGA 1	BitTorrent
Radar	Linux OS	Kindle
Tea Bags	VoIP	Mobile Apps
Business Directories	Iris Scanning	Touchscreen
Plastic	Online Radio	OLPC
Electric Washing Machine	eBay	Electronic Voting
Radio Broadcasting	Wi-Fi	Vacuum Robot

2.2 Application Scheme – Android Studio, Firebase

2.2.1 Android Studio

Android Studio is the official integrated development environment (**IDE**) for Android application development. It is based on the **IntelliJ IDEA**, a **Java** integrated development environment for software, and incorporates its code editing and developer tools (Rouse, 2019). Android Studio utilises a Java programming language, but in the form of Android. Also, due to it being specialised for building android applications, it had certain syntax which catered towards GUI components e.g. buttons, spinners and input-text field boxes. Below are examples from our LoginActivity (Service Requester Application) of how we initialised and assigned values to our GUI features:

```
// Create all the UI features we wil
private Button buttonSignIn;
private EditText editTextEmail;
private EditText editTextPassword;
private Button buttonSignUp;

// Create the database variables
private FirebaseAuth firebaseAuth;
```

Creating variables to our GUI features and database variables.

Figure. 1

Initialise the objects

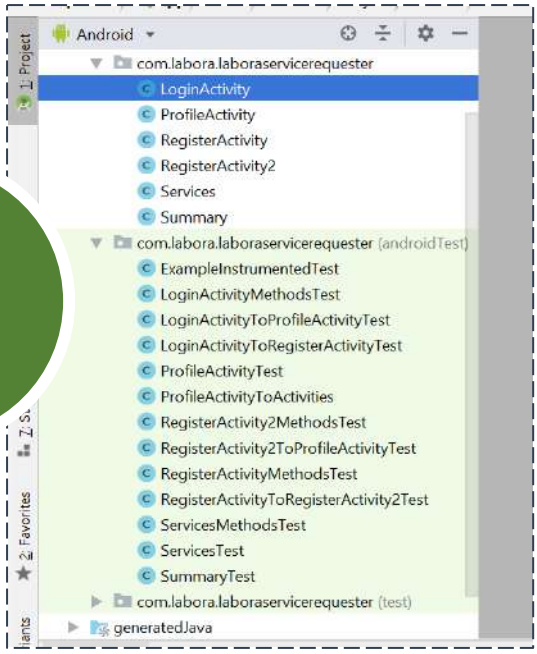
Figure. 2

```
{
    //profile activity
}

// Initialise the objects
editTextEmail = (EditText) findViewById(R.id.editTextEmail);
editTextPassword = (EditText) findViewById(R.id.editTextPassword);
buttonSignUp = (Button) findViewById(R.id.buttonSignUp);
buttonSignIn = (Button) findViewById(R.id.buttonSignIn);

progressDialog = new ProgressDialog( context: this);

// Set on click listeners
buttonSignIn.setOnClickListener(this);
```

The platform also enabled us to separate out our class files. This meant we were able to code different pages of the app, on different files. As a result, we could simultaneously work on the project, without having to suffer conflicting code. Consequently, we made use of the in-built Git features. This enabled us to connect our project to Goldsmiths GitLab repository (Gitlab, 2019). We created two independent repositories, one for our service requester application and one for our contractor application. Therefore, the use of Android Studio as our development environment, enhanced our development experience, as well as easing functions, such as committing, merging, pulling and pushing code. Thus, improving the efficiency of the whole group.

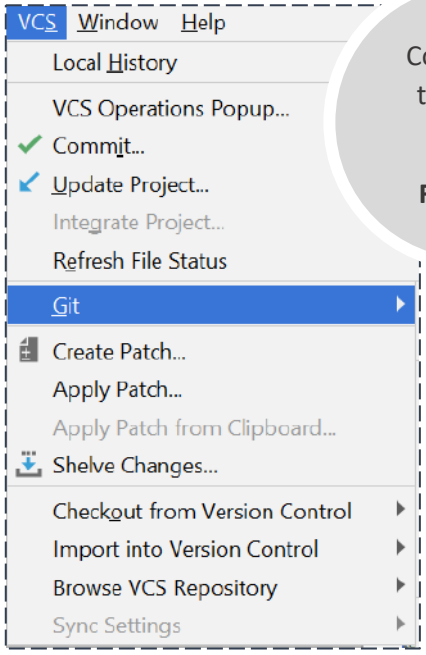
Initialise the objects

Figure. 3



Commiting to GitLab process

Figure. 4



Repo 1 – **Service Requester**

Figure. 5

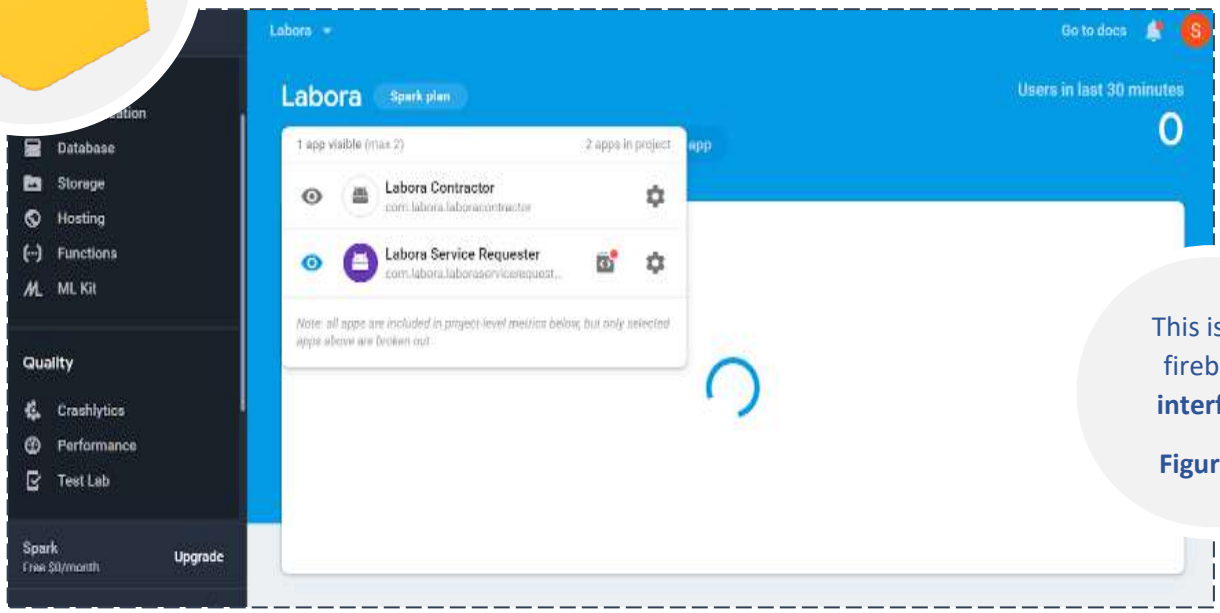
Repo 2 – **Contractor**

Figure. 6

2.2.2 Firebase

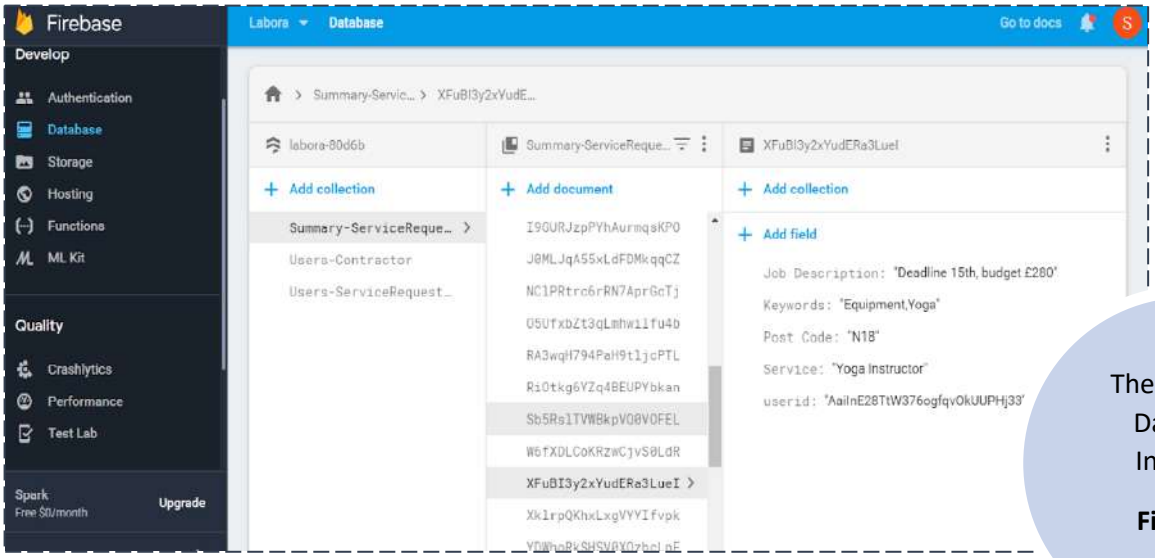
Firebase was used for the backend component of our application(s). Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in Realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data. (Firebase, 2019).

Initially, we planned to use MongoDB, however, due to the benefits that a Firebase backend could bring, we switched and adapted our technical architecture. A few of the advantages that Firebase encompassed was that it had a testing environment, called 'Test Lab', wherein we were able to test our apk via an automated Robo Test. In addition, Firebase integrated very well with Android Studio, which meant the overall applications(s) would functional more fluidly. Also, when it came to testing, we were able to change the rules of the database, so we could type into the field we were testing in real time and check if it worked. Therefore, enabling us to troubleshoot instantaneously.



This is the firebase **interface**

Figure. 7



The Firestore Database Interface

Figure. 8

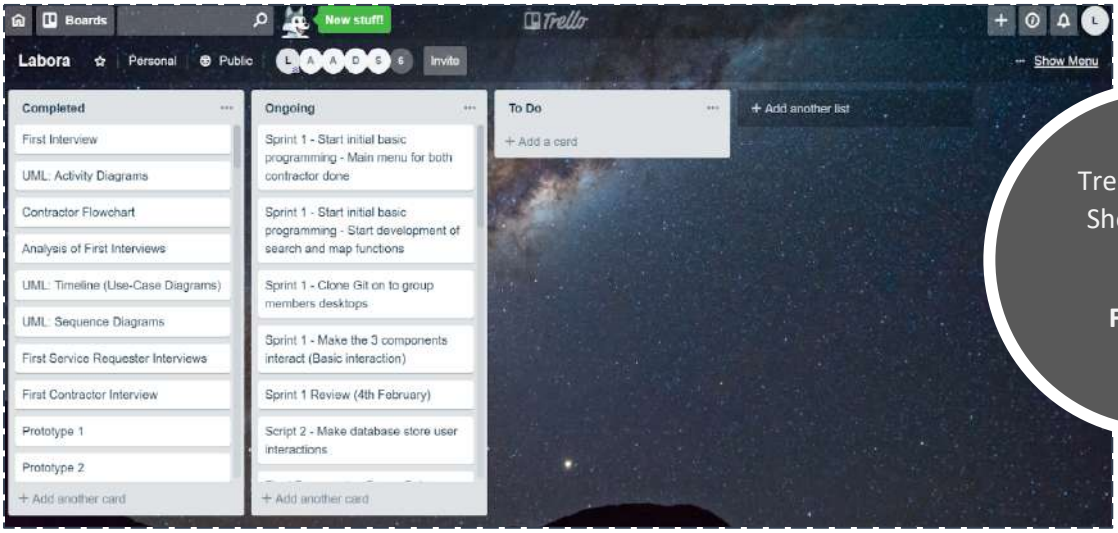
3. Development Process

3.1 Kanban - Trello

We began our development process by outlining the tasks we needed to conduct in our sprints and scrums. We did this by utilising the online website Trello (Trello, 2019).

Trello employed a Kanban style-approach to creating boards. A Kanban is based on 3 different principles. Visualise what you have to complete today, in the present time. Set a limit on the amount of work currently being progressed, so as to not overload the workforce (team). And lastly, enhance the workflow by selecting the next highest priority job to be completed (CollabNetVersionOne, 2019). We already set a target of creating our minimum viable product (MVP) for the end of this project and development stage. In order to get an idea of what our MVP contained us initially made a backlog (**refer to pg 48**) which enclosed all the components we aimed to build for first, second and third release of our application(s).

As we were interested in the first version (MVP), we concentrated on features such as search for jobs, request button and see location (general area) of service requester. Following on from this, we got a good idea for what we envisioned our final MVP would look and function as. As a result, we were then able create a Kanban on Trello's boards section. We started off by creating three separate sections: Completed, Ongoing and To Do. Originally the To Do list had quite generic and basic tasks e.g. build the front end for the service requester. We found that this would be quite difficult to interpret by group members, as it was too broad. As a result, we decided to break general tasks down, for example instead of having one task for the whole front-end, we would say create login button for home page, create text fields for service requester, add images for profile page etc. Therefore, we were able to create a more specific task list. Consequently, it meant each member of the group could select a task for the sprint and complete it, once they did, put it in the 'Completed' section and pick another card.



Trello Board – Showing our logs

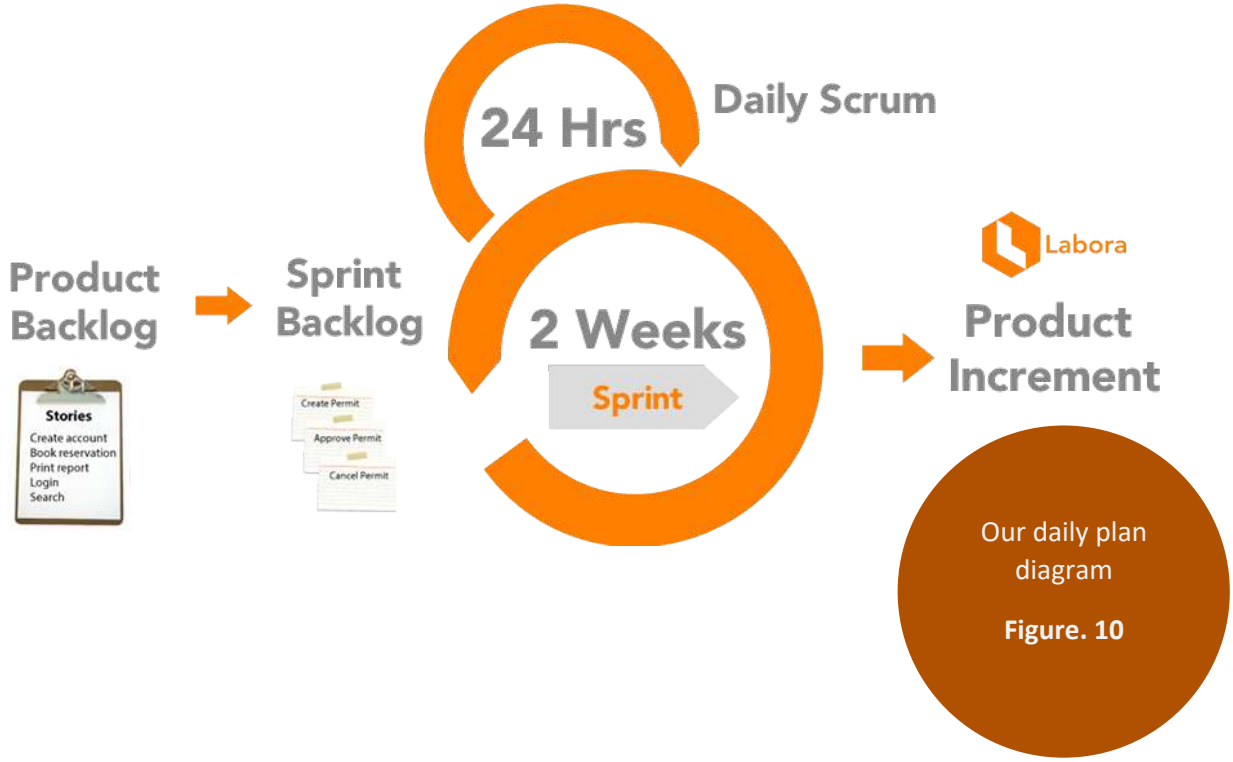
Figure. 9

3.1 Sprints & Scrum

Adjoining the Kanban, we needed to set up the structure of our weekly sprints and daily scrums. As for the Kanban, we used Trello, we needed a platform whereby group members could access their tasks for the sprints and what they had to achieve by the end of a scrum duration. Due to the frequent periodic nature of the sprints (two-week sprints) and scrums we had to adopt a software which was easily accessible and accommodated instant messaging. The reason for this was so that any discrepancies in the tasks any members had could be immediately answered. In addition, although Trello was good for displaying the tasks, we need a service which allowed us to immediately make it evident that a task is being changed or assigned to a different member.

We concluded that we would use WhatsApp (WhatsApp, 2019). WhatsApp brought many benefits to our group and development process. All members of the group already had WhatsApp, which meant we already had the skills required to use such a platform. In addition, we created a group chat, including all members of our group. We had an agreement, to alongside our constant weekly meetings, keep constant contact throughout the process, every day. We made use of the group description feature. We asked members for the card (task on Trello) they wanted to complete, and we outlined them on the description. As a result, everyone knew their job, and everyone knew what others had to do.

Consequently, any issues which would arise we could directly contact every individual immediately. Therefore, we kept up a record of what each individual's daily progress was. Also, because our two-week sprint's consisted of many daily scrums, if we found members who were falling behind or needed help, as their level of progress was limited, we would assign available resources (group members with free time) to that task. Therefore, ensuring that the structure of our development was kept up.



Our daily plan diagram

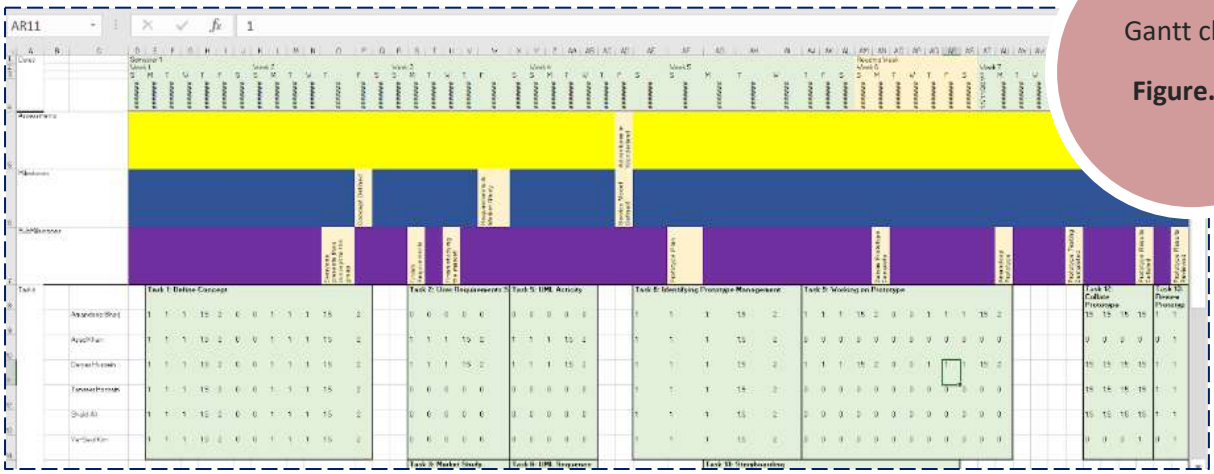
Figure. 10

3.2 Management of Group Members

Additionally, after setting up areas for development tasks and communication, we had to establish the management of resources (group members) system. Since term 1, we were making use of a Gantt Chart and Tracking Sheet. The Gantt chart was catered more towards the sprints and scrums, due to the nature of its layout. It had dates across the top border, and underneath, 3 sub-lists: Assessments, Milestone and Sub Milestones. Underneath we had each group members name (3 sets of these).

Each week we would outline the week and the name of the tasks the members performed. Using our Google Drive and WhatsApp conversations, we were able to determine how many hours a day a member performed. Therefore, we were then able to output their hours for a week and determine if they needed extra support or whether they were available to assist elsewhere. Throughout our development stage, we found that initially many members were always available, as tasks were short and basic. As the application(s) grew and we moved into the third sprint, we found that tasks were maximising group members time. As a result, if we need extra support on a certain component, we had to do overtime. Consequently, this was all noted on the Gantt chart. Therefore, the useful accumulator at the bottom of the spreadsheet meant we could check how many resource hours each member used and calculate which members possible required a change of task or support. In addition, the tracking sheet served more as a recording document. It outlined our tasks, both in term 1 and term 2. We had a total resource hour each of 232 hours.

Within the document, we placed in a table the name of the activity carried out by the group and individuals, as well as the hours the inputted for that task. Thus, this enabled us to keep a robust record of development and contribution per member. Therefore, it ensured management was transparent and well organised, minimising the chance of mismanagement.



Gantt chart

Figure. 11

3.3 Methodologies

Furthermore, after concluding how we would manage our resources (group members) and track tasks required to be completed, we set out to find a methodology, which we would base the principles of our development upon. We found the agile methodology was most commonly adopted in terms of software engineering in industry or academically (Stackify, 2019).

Agile methodology consisted of working in an iterative manner. Incrementally developing features and ensuring they work, then proceeding. Primarily, this concept is seen in the application of our sprints and scrums. Sprints represent the sequential long-term goal (in a two-week duration), and scrums are the iterative structures that enabled us to work towards those objectives gradually.

There were significant leverages when applying these principles. One of which was that it allowed members to become owners of their own task. As a result, this meant micromanagement became a major theme amongst members and their tasks. Consequently, this meant that the workload was spread. Therefore, this reduced overall stress of the group and meant that group members specialised in the areas and task they had to perform. For example, we had group members initially struggle building front-end features such as buttons and ensuring they were clickable. However, by the end of the first sprint, they were accustomed to this code, and efficiently assisted others with it.

3.4 GitLab and Local Storage

Furthermore, we instantly discovered that there was a major issue with simultaneously coding and trying to merge work. We initially thought it would be better for the group if we each coded a different part of the application each week. This would resolve the issue of having to work on the same part of the app at the same time, whilst ensuring no conflicts arose. However, when it came to stage where we needed to check whether a certain action would register a certain response, problems reared. For example, from the login screen, we needed to check if the profile page would open up (see below).

LoginActivity.java

```
// Checking username & password against database
firebaseAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener( activity: this, (task) -> {
                progressDialog.dismiss();

                // If their account details exits, log them in
                if(task.isSuccessful())
                {
                    // Finish current activity
                    finish();
                    // Instantiate ProfileActivity
                    startActivity(new Intent(getApplicationContext(),  ProfileActivity.class));

                }
                // Else they dont have an account
                else {
                    // Toast to inform the user
                    Toast.makeText( context: LoginActivity.this,  text: "Please enter valid login credentials or register an account", Toast.LENGTH_SHORT)
                }
        });
}


// On click method
@Override
public void onClick(View view)
```

LoginActivity.java **Code**

Figure. 12

Our solution to this was using the system software, Git, which is a distributed version control, system (Sridhar, 2018). It tracks changes in source code in software development. As aforementioned, we used Goldsmiths GitLab Repository, which provided a user-friendly interface to Git. Due to members of the group not being familiar with Git, we needed to provide extra time and resources of learning. As a result, this benefitted us in the long term, as it meant we could work independently on code, confirm via messenger the specifics regarding the work and then push. Consequently, it made development much more practical and efficient, as everyone could pull any individual's code and make adjustments were necessary. Thus, development progressed adroitly.

4. Analysis

4.1 Requirements (Functional and Non-Functional)

We will now discuss the functional and non-functional requirements for our application(s). This section is very important for outlining what our software must meet in order to be accepted as the minimum viable product (MVP). We have configured a Systems Requirements Specification for our software and will use it when determining if our project is a success (**refer to pg.71 appendix**). Functional requirements define the specific behaviours and functions the software should perform. Whereas, non-functional requirements describe how a system should behave and what limits there are on its functionality.

4.1.1 Functional

The main objectives of this project included creating an application(s) which served the user group it was intended for. In our case, this meant creating two application: Service Requester Application and Contractor Application. As a result, we needed to develop an application(s) which ensured user accessibility and usability. Consequently, we could achieve this through constructively criticizing our application against functional requirements that it should have. Therefore, if we met these functional requirements, we would get the understanding that most, if not all, users have an acceptable level of User Experience. Thus, rendering our application utilitarian. Functional requirements table for our MVP is below:

FUNCTIONAL REQUIREMENTS		
ID	**DESCRIPTION**	**GENERAL OR SPECIFIC TO SERVICE REQUESTER/CONTRACTOR APP**
FR001	Should be able to login to their account using their login credentials	General
FR002	Can register an account via the registration page	General
FR003	Profile page which displays their details and has buttons to go to their desired pages	General
FR004	Service request page, where users can enter their preferences	Service Request App
FR005	Map page, where users can view jobs in their desired location	Contractor App
FR006	Confirmation page, for when a transaction (in terms of action) has been successful	General

FR007	User credentials are stored securely in the database	General
FR008	Markers on the map are clickable	Contractor App
FR009	Once marker clicked, job details appear	Contractor App
FR010	Details are stored in the database, in correct fields e.g. integers in an integer field	General
FR011	Markers must indicate the status of a service request	Contractor App
FR012	Application correctly extracts and displays data, previously submitted e.g. personalised welcome message	General
FR013	User can request more than once	Service Requester App
FR014	User can request same types of services	Service Requester App
FR015	Application should scale itself, thus not limiting certain users with different devices	General

4.1.2 Non-Functional

Similarly, in order guarantee that the software we create is of an acceptable standard, we must also meet non-functional requirements. Non-functional requirements give us a good idea of whether our project is limited or can gratify on diverse scales. Below are the non-functional requirements for our MVP:

NON-FUNCTIONAL REQUIREMENTS		
ID	**DESCRIPTION**	**GENERAL OR SPECIFIC TO SERVICE REQUESTER/CONTRACTOR APP**
NFR001	Users should experience fast submission when submitting forms	General
NFR002	UI should tailor to the width and height of user device	General
NFR003	Users should find spinners and filling forms relatively easily	Service Requester App
NFR004	Users should find clicking the markers on the map easy	Contractor App
NFR005	Confirmation page should appear quickly after action and form submission	General

NFR006	Database should store data relatively quickly	General
NFR007	Database should be easy to query	General
NFR008	Database should scale and be readily-available, without dropping user experience	General

4.2 Switch to two applications

Moreover, in terms of the "physical" application itself, we initially proposed to have a single application, which catered for both service requester and service providers (contractors). In addition, we believed we would have the application share the same MongoDB database. However, once we researched further, we realised our application was quite similar to Uber's ecosystem (Ashlock, 2014). It was similar in terms of service providing and service requesting. Benefits of having two applications meant users didn't have to see anything which didn't regard their interests. Also, it makes it easier to monitor the two sets of users. Consequently, we altered our original decision and opted to have two separate applications.

4.3 Adapted Technical Architecture

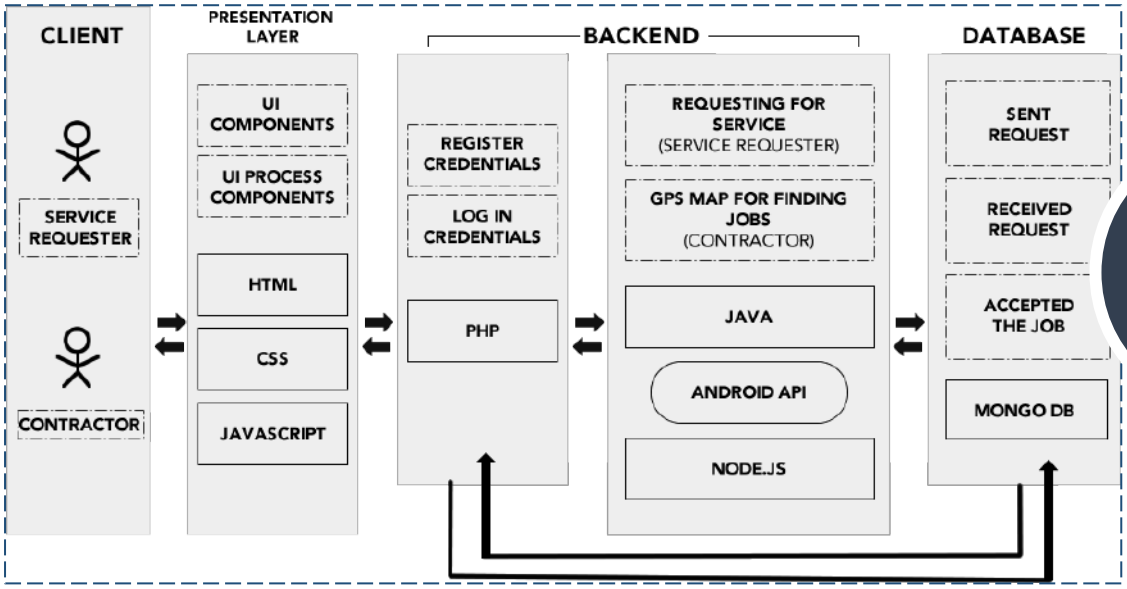
Additionally, a change in the applications structure, lead to a change in the technical architecture originally intended. We had to modify the technologies formerly chosen, and then re-plan a new adapted technical architecture. At the beginning of our project, during the proposal phase, we researched how we wanted to structure our front-end and backend, and what we needed in order to ensure they worked synchronously. Although we researched and were open to new ideas, at the time, we were biased in our judgement as a group and were methodologically reluctant to seek out new platforms and software that would aid us in the development of our application(s). Instead, we chose architecture technology such as HTML, CSS, Java and MongoDB. The sole reason was due to the fact all members of the group were primarily exposed to only these languages. Consequently, this led us to make the "secure" decision on choosing the aforementioned languages as our foundations for this project.

However, towards the end of the term, the group insisted on carrying out more research. We had the idea of exploring and collating how different individuals over the globe approached application production. This derived the information on Android Studio and how dominantly it was being used for building Android-based software. This led us to consider changing our technical architecture. In order to consolidate our interest and materialise utilising Android Studio (Java front-end), we undertook extra meetings. During these meetings we discussed the advantages of Android Studio development over more traditional approaches.

For example, instead of coding vast amounts for a button and its style using HTML and CSS, we could do all of that in less lines of code, whilst getting a better visualisation of how the application would look like on an actual android device (emulator). Consequently, utilising Android was a step in the right direction for the group, as well as the project.

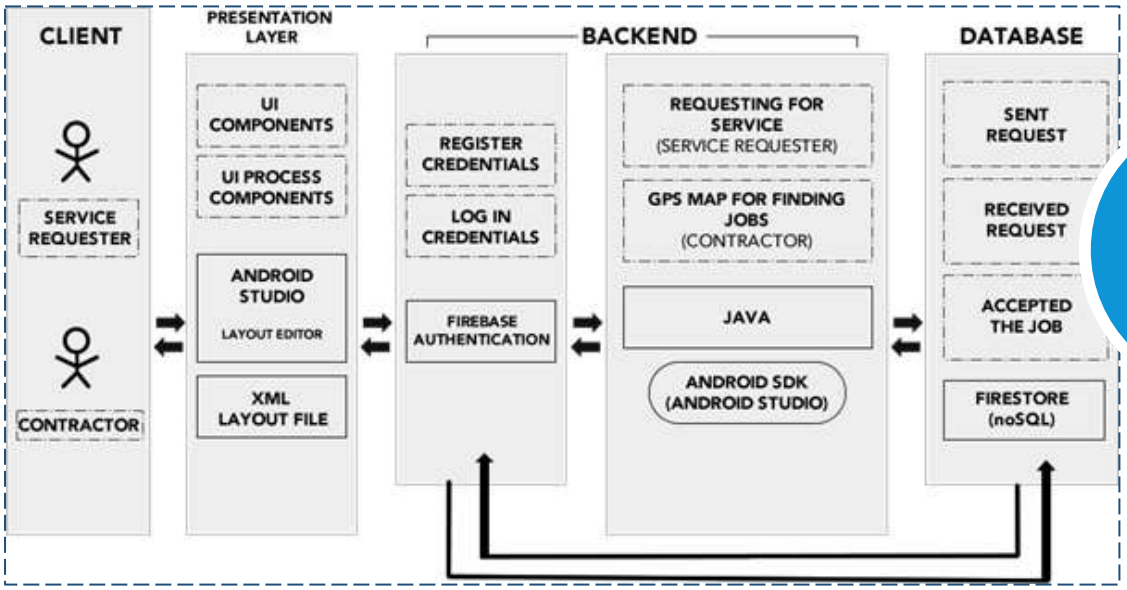
The next significant change in our technical architecture came about when we were discussing how to plan tasks for the implementation of the MongoDB-based backend. Similarly, we as a group were technically limited, especially for backend development. We understood the basics for administering MongoDB features, however, not to the level where we could entirely constitute an interactive backend.

Therefore, we took the same approach as for the front-end and investigated other routes we could take in order to achieve database required for the MVP. Eventually, we discovered Google's Firebase (Firebase, 2019). Before we made the crucial decision of tuning our technical architecture, we weighed the possible benefits and possible drawbacks of Firebase against MongoDB (**refer to pg.52 appendix**). Finally, after careful consideration and group acceptance. We came to the decision we should adapt our architecture and opt for Firebase over MongoDB.



Old **Version**

Figure. 13



New **Version**

Figure. 14

4.4 Stakeholders – Contractor and Requesters

A significant factor of our project involved correctly identifying who the target market of our application was. In our own experience, we had members of the group who had trouble finding trustworthy and experienced freelance workers to carry out services. Online research showed us that our primary two user groups were service requesters (**refer to pg.53 appendix**) and contractors (IPSE, 2019). To further validate if these were our target audience, we set both groups a questionnaire each (**refer to pg.55, 56**).

The questionnaires' answers and analysis helped us confirm that the major user groups for our applications are people who need a service carried out, and people who have the skills to provide those services. Therefore, we were able to envision our market and start to tailor our planning and development process towards them.

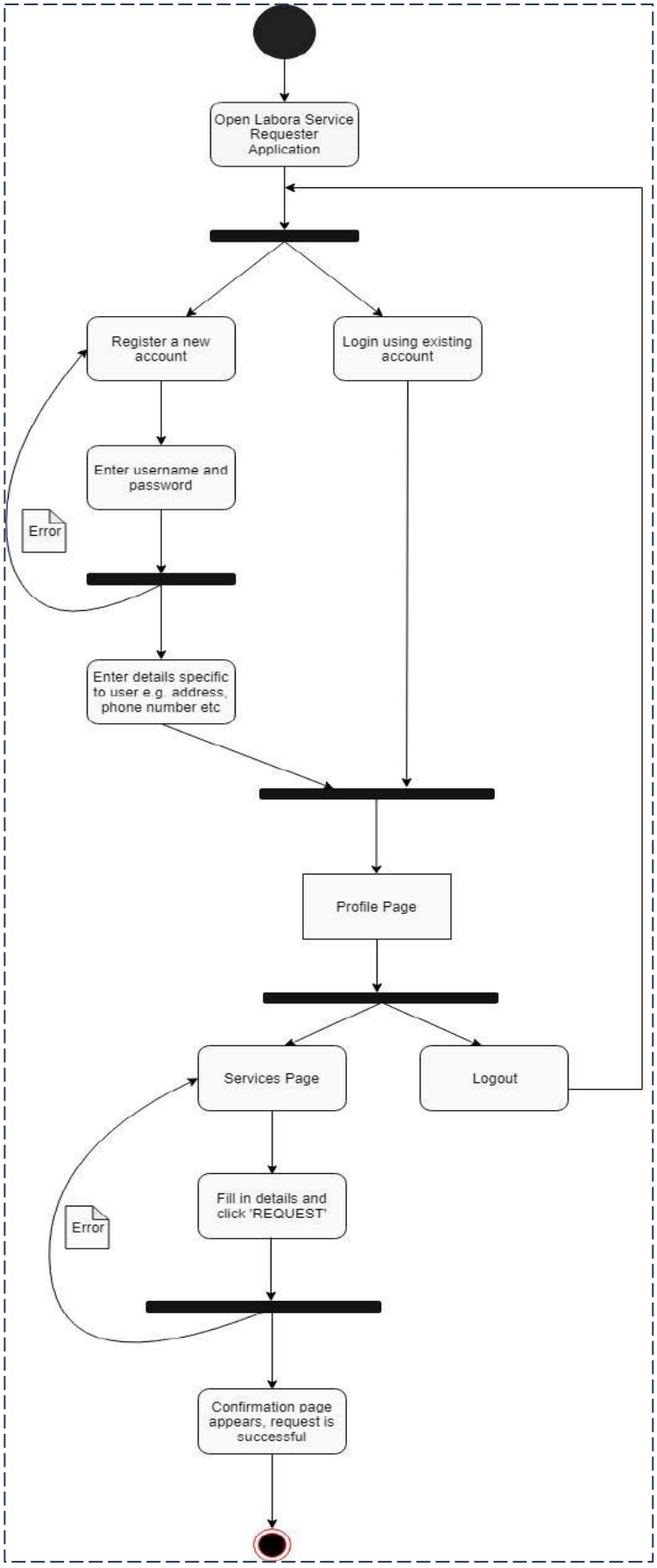
5. Design

The following passage will express, as well as illustrate, how we constructed design models, based on how we believed the application(s) to function and behave. A variety of models have been used to portray not only the functional aspect of the application, but the visual outlook also. Additionally, we have overseen user questionnaires on the models. As a result, we were able to refine our design. Therefore, getting closer to optimum user accessibility, usability and appeal.

When discussing diagrams to best represent our systems functionalities and stages, we decided to utilise UML diagrams (Paradigm, 2019). Unified Modelling Language (UML), is a standardized modelling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artefacts of software systems, as well as for business modelling and other non-software systems.

5.1 Activity Diagrams

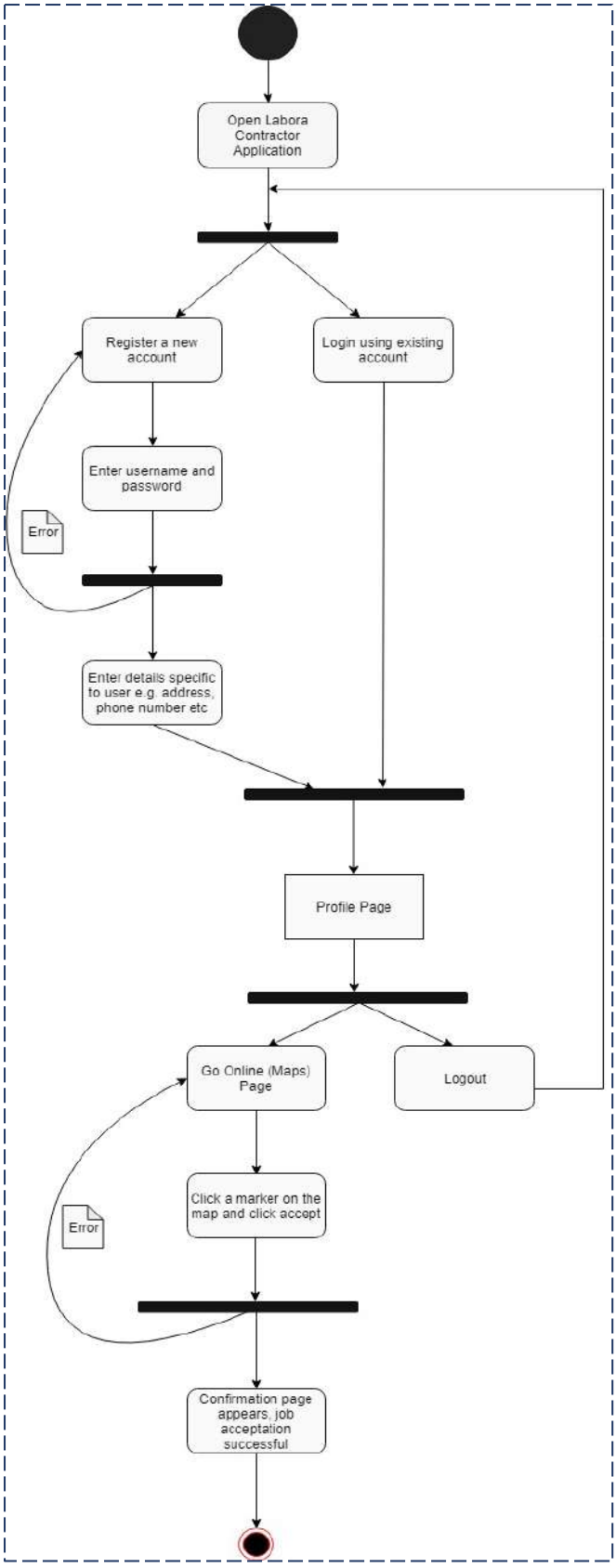
The first type of UML diagram we used were activity diagrams. The activity diagram for both service requester and contractor embodies the different processes users can face when going through the applications. It is clear at each step the options available and demonstrates how the system flows. Thus, we were able to depict the various functionalities, as well as the paths that could be taken, when progressing through our application(s).



Activity **Diagram**

Figure. 15

For **Service Requester**



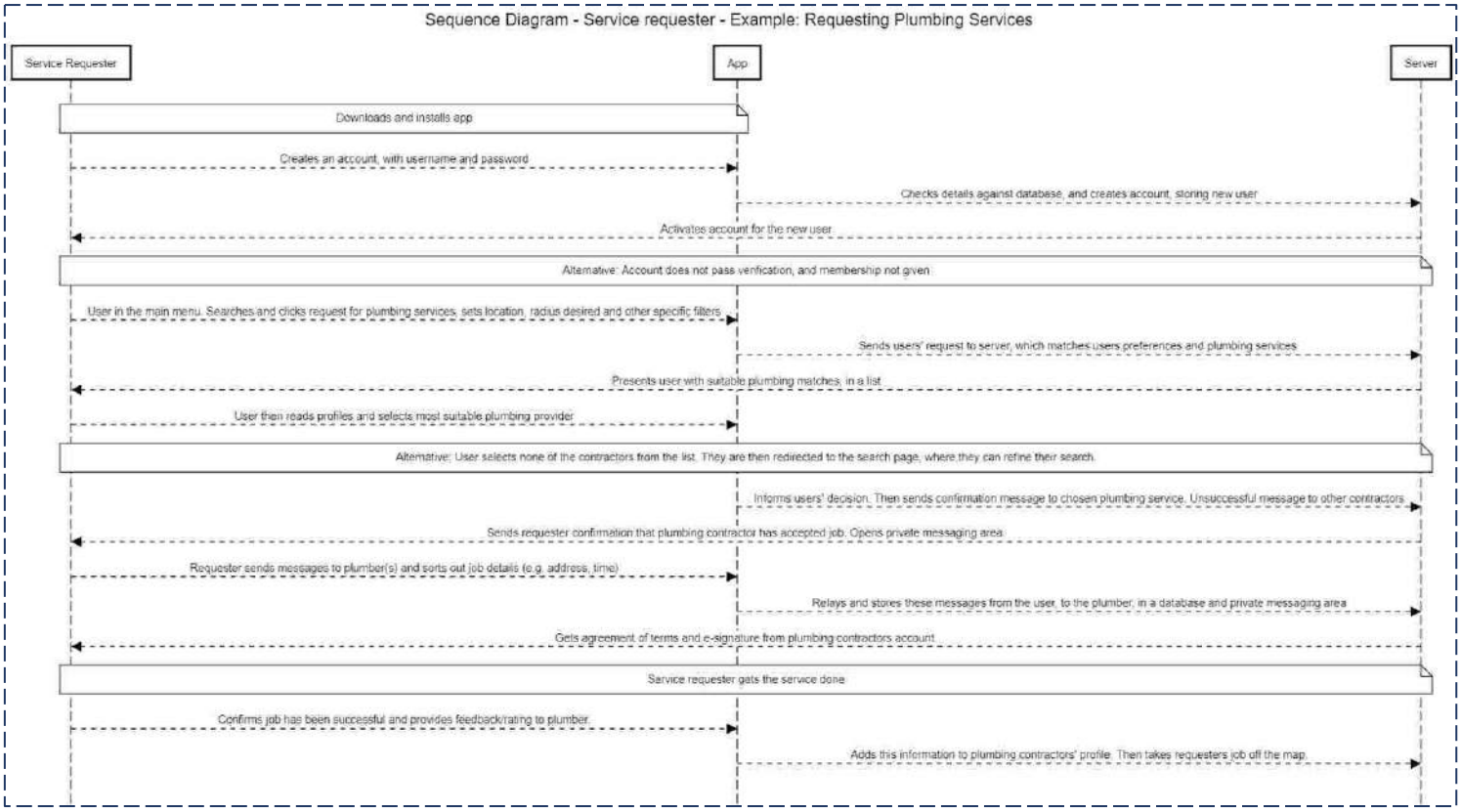
Activity **Diagram**

Figure. 16

For **Contractor**

5.2 Sequence Diagram

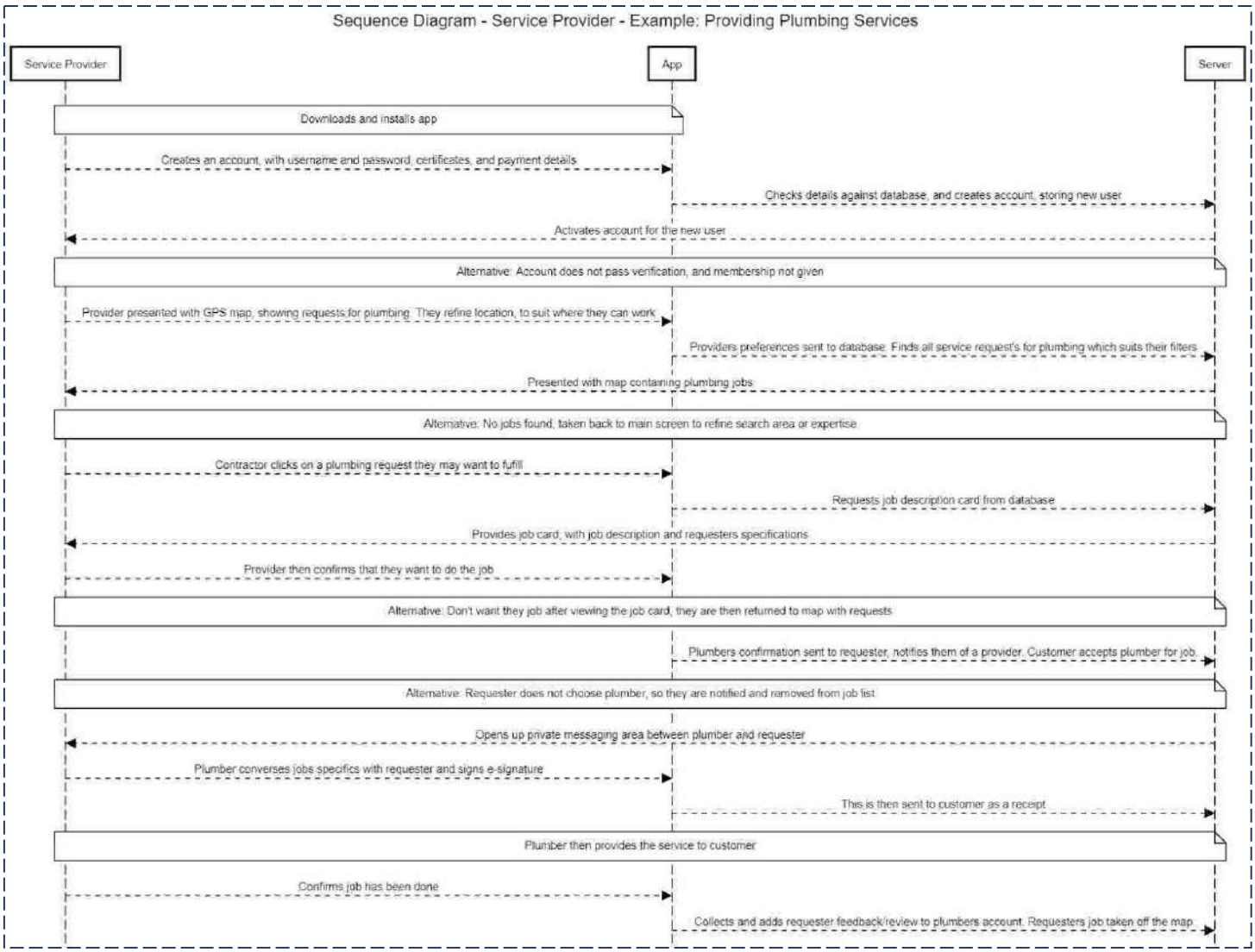
Subsequently, we delineated our applications' process via UML sequence diagrams. Sequence diagrams are similar to activity diagrams in the context of representing a system's process and flow. We outlined, in the diagrams below, the typical procedure a user would take in our service requester and contractor applications. In addition, we have provided alternatives that would could occur at each step. Thus, entirely demonstrating our software.



Sequence **Diagram**

Figure. 17

For **Service Requester**

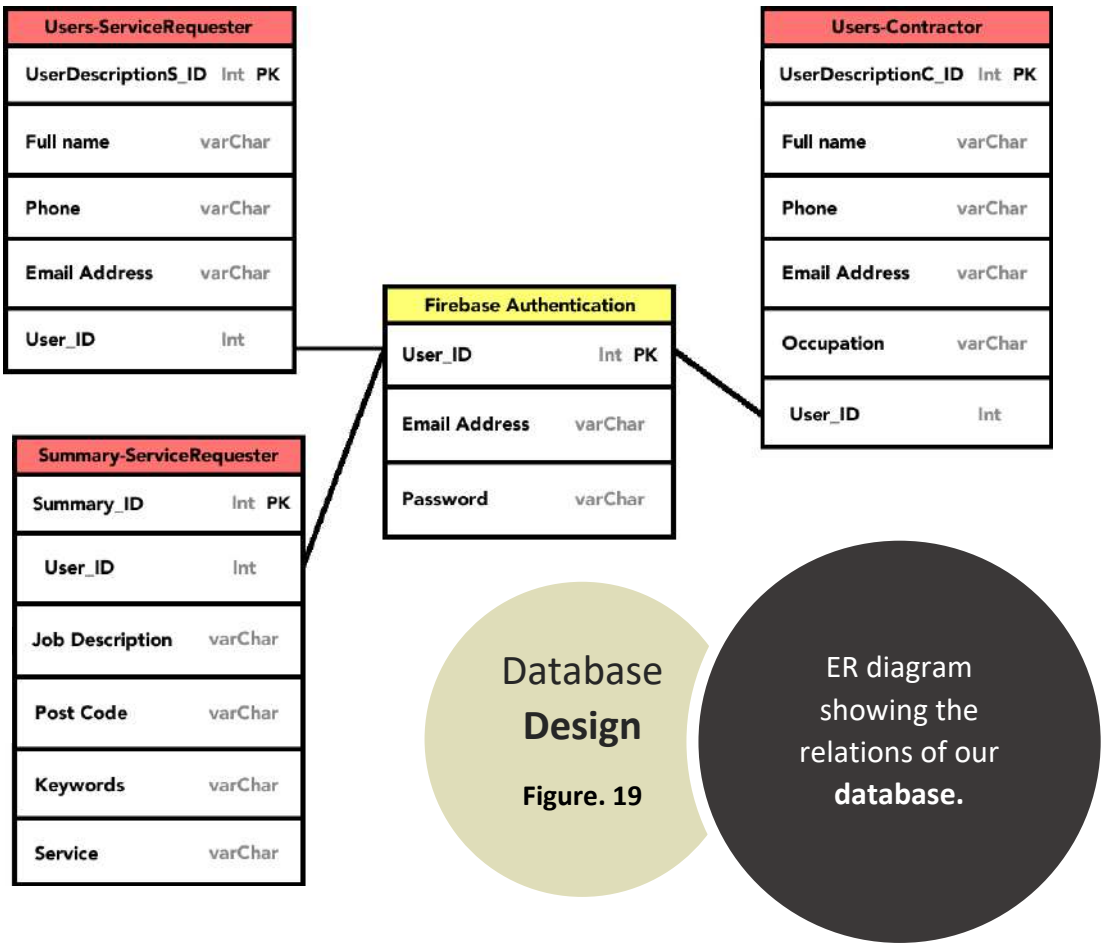


Sequence **Diagram**

Figure. 18

For **Contractor**

5.3 Database Design



5.4 Prototype

Before we started to prototype, we had to create storyboards to showcase our ideas of the product to our end users. This showcases a journey the users will experience when using the app and gives us initial ideas on what we want to have on our app. Once we finalised how our products were meant to look like, we created 2 prototypes for both contractor and service requester on Adobe XD (refer to figures below).

Our intention was to create two groups of users from the service requester's side and show one group prototype 1 and the other prototype 2. This was part of our A/B testing as it would enable us to gain an unbiased feedback. However, with the contractor we showed them the both prototypes. It proved difficult to find contractors who were willing to partake in user testing in such short amount of time. We conducted two types of research. An interview was conducted straight after they interacted with the prototype. We also conducted a questionnaire online by posting both prototypes online and received feedback which corresponded with what we heard from the interview. We consider prototype 1 and 2 as two different designs rather than two models.

The results (**refer to pg.55, pg.56, pg.60, pg.63, pg.66 appendix**) gave us indication on where we can improve our quality when we build our final application. The emerging pattern of users saying prototype 1's function is the strongest point whereas prototype 2's visuals of the menu to the search function is better. They said prototype 1 should be improved in terms of design whereas prototype 2 should be improved with functionality and ease of use. When we asked a select few of users (both contractor and requesters) through a meeting, we asked them:

"Where would you lean most when the *final* outcome is developed?"

Majority of our users preferred the function over form (**refer to pg.55, pg.56, pg.60, pg.63, pg.66**). We took this in to serious consideration and brokered the idea of developing an app that involves some form of aesthetics whilst putting function at the centre core of our final product. We found out that people would use the app rather than going on online websites. As a whole, the results helped us to gauge the importance of taking user information in to consideration in order for us to appeal to our desired market.

The underlying theme from the individual feedback pointed towards an application, which needed to be attractive i.e. making use of warm, inviting colours, such as red, yellow, orange, which are also the colour theme of our Labora logo (Gremillion, 2019). As well as being easy on the eye, the application(s) had to be intuitive. This was in order to facilitate a broader range of users, with different usability and accessibility requirements. Consequently, we integrated these changes, by adding larger buttons, an easier navigation menu and intuitive features (e.g. back button in the top left-hand corner).

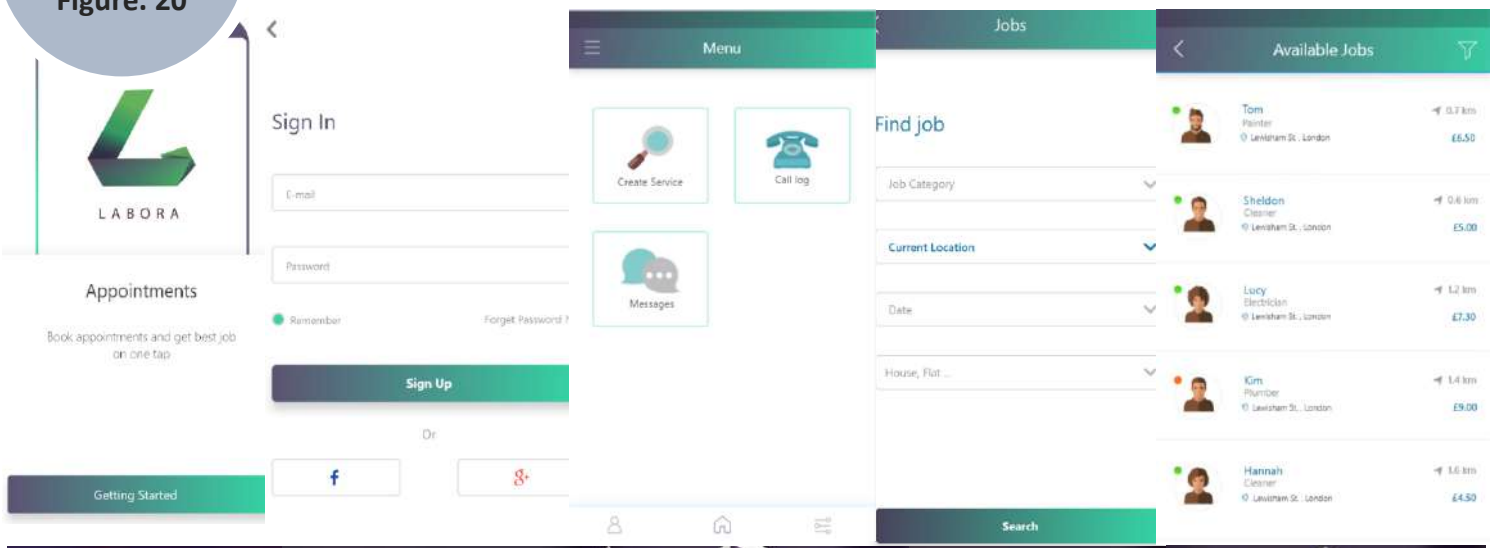
User Experience is an integral part in developing our product. From research, we've found out that users lean towards functionality over the aesthetics of an application. This feedback has had great influence on how we further developed the product when we release it to the mass market. Accessibility was a criterion that we initially set out to meet in our prototypes. However, in prototype 2, we found out that it wasn't the case (**refer to pg.66**) so we decided to opt for prototype 1 which was more suitable for our users in terms of functionality and accessibility. We looked into the ideology of Fitts Law (Foundation, 2019).

This takes into consideration the importance of size, and predicts the time taken to select a button. We want to reduce the time taken to make it easier for the user. The contractor page has a large button which tells users to 'Go Online'. We made the menu page as minimal as possible so that contractors can get straight to finding work from a touch of a button.

Other features that will be added beyond our MVP will have its separate menu page which the user has to slide up to get. The service requester has a large button on the screen where they can start searching for jobs. We've made sure buttons and information fields are big enough for the user to see, thus making it accessible for all.

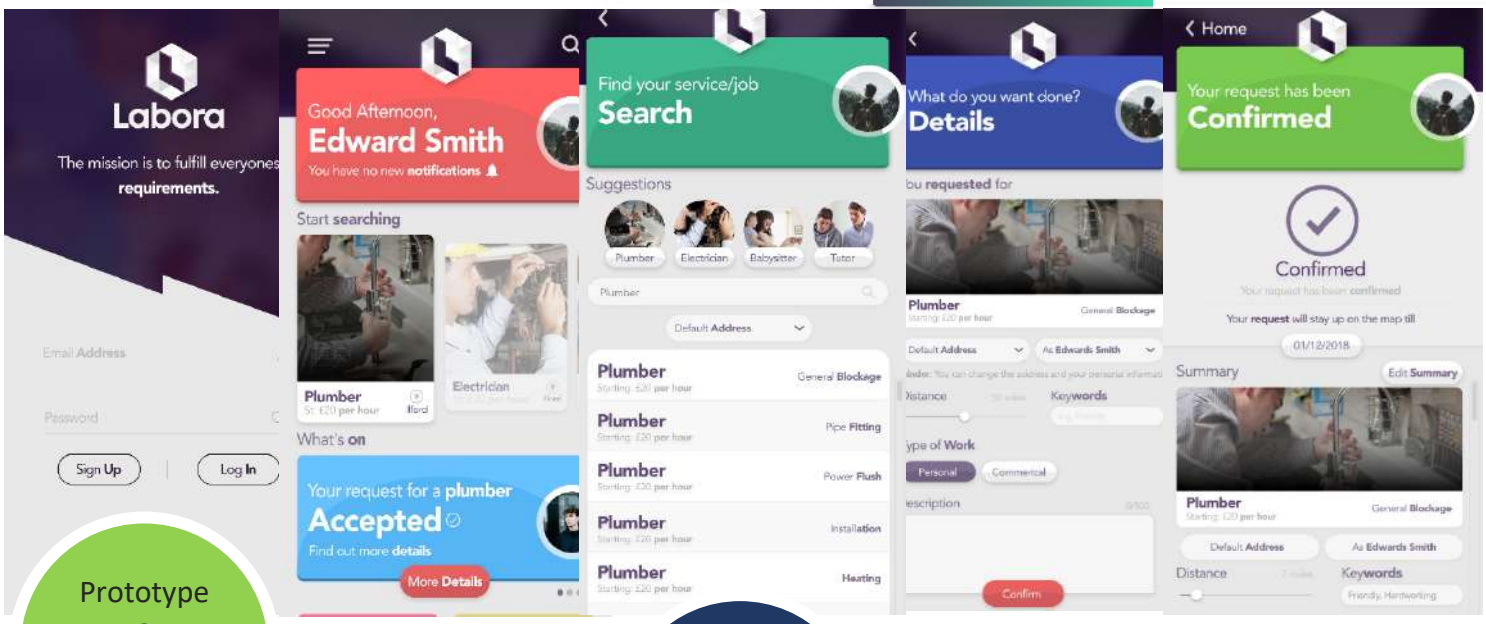
Prototype 1

Figure. 20

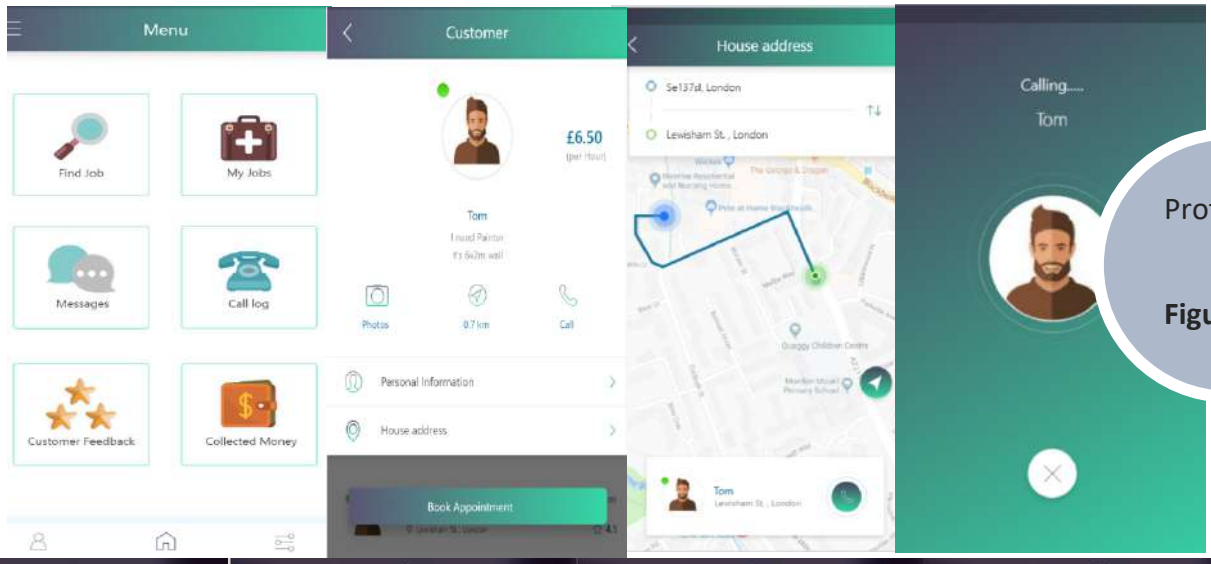


Prototype 2

Figure. 21

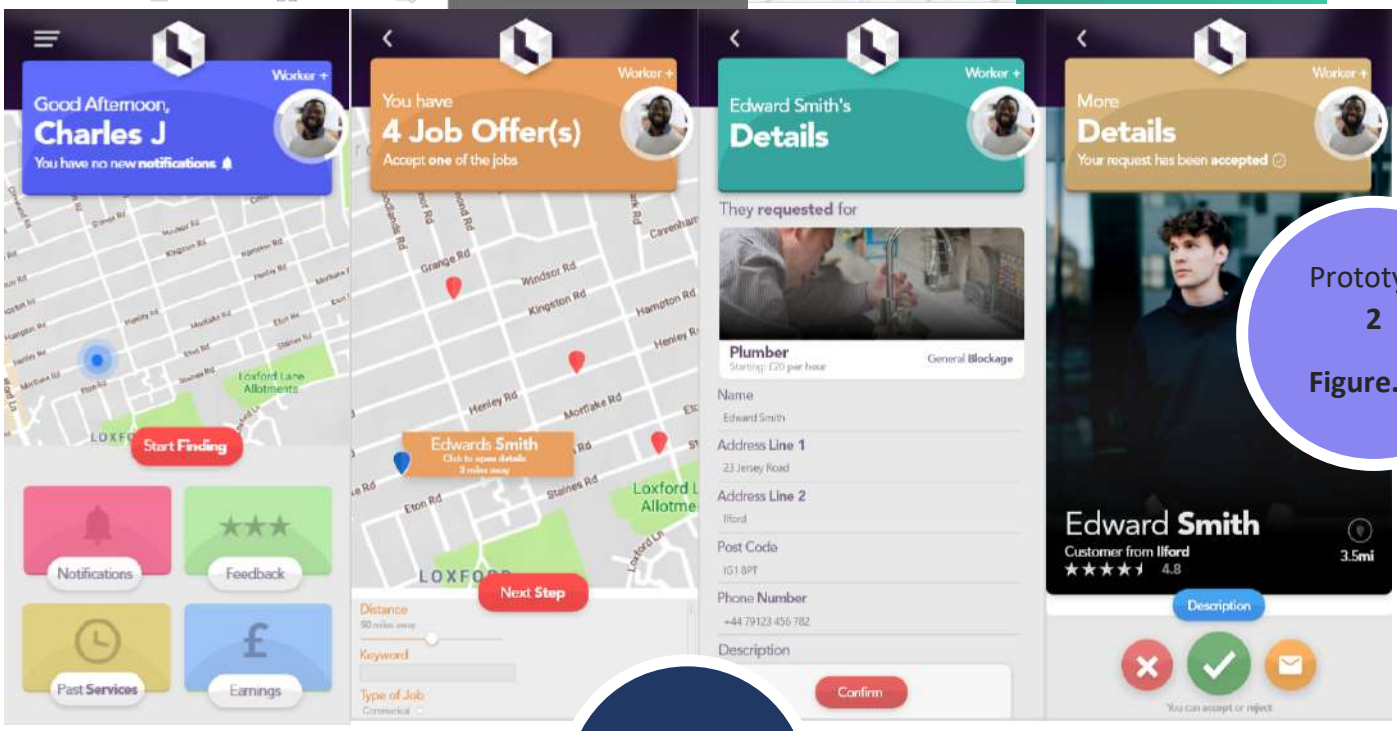


Service Requester **Prototypes**



Prototype **1**

Figure. 22



Prototype **2**

Figure. 23

Contractor **Prototypes**

5.5 Code Design

As well as planning and polishing the design for the front-end, we started scheming the structure of our code. Although code would not be visible to the user in terms of User Experience (UX), we believed we needed to construct it in a manner such that it would aid us in the long term. For example, if we coded each activity of the application in the main ("onCreate") method, then it would function the same. However, this would be quite inefficient if we desired to edit in the future. Also, at an industry-level (similar to an academic standard) it would be unacceptable, as collaborative work would become disorganised. As a result, considering this, we revised our code so that large code blocks were separated.

Consequently, it made our program much more modular. Therefore, this minimised the hours spent on locating and eradicating bugs, as we knew which component must be responsible for any issues. In addition, incorporating comments in our work were of great significance. They allowed each member of the group to get a basic idea of what each segment outputted or contributed. Thus, the inclusion of code design, enhanced our technical product(s).

```
// Public class login activity
public class LoginActivity extends AppCompatActivity imple

    // Create all the UI features we will be using
    private Button buttonSignIn;
    private EditText editTextEmail;
    private EditText editTextPassword;
    private Button buttonSignUp;

    // Create the database variables
    private FirebaseAuth firebaseAuth;
    private ProgressDialog progressDialog;


    // On create method
    @Override
    protected void onCreate(Bundle savedInstanceState,
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        // Initialise fire base
        firebaseAuth = FirebaseAuth.getInstance();

        // Conditional to check if current user exists
```

Comments **Exemplar**

Figure. 24

An example of the LoginActivity showing comments in our code.

We have done this for all **activities**.

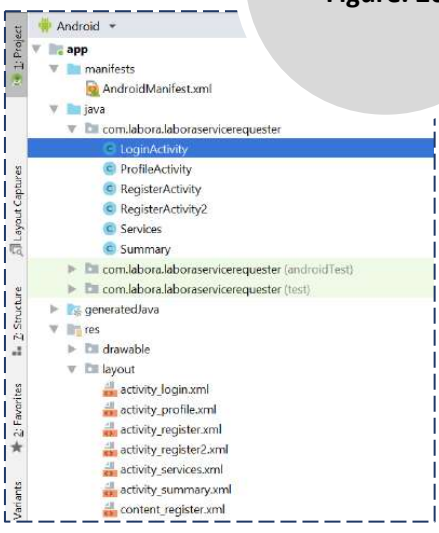
```
// Method for logging in
private void userLogin()
{
    // Create strings to store username and password
    String email = editTextEmail.getText().toString().trim();
    String password = editTextPassword.getText().toString().trim();

    // Check if email entered
    if(TextUtils.isEmpty(email))
    {
        //email is empty
        Toast.makeText( context: this,  text: "Please enter a valid email address", Toast.LENGTH_SHORT).show();
        // Return
        return;
    }

    // Check if password is entered
    if(TextUtils.isEmpty(password))
    {
        //password is empty
        Toast.makeText( context: this,  text: "Please enter a valid password", Toast.LENGTH_SHORT).show();
        // Return
        return;
```

Modular Code **Exemplar**

Figure. 25

6. Implementation

6.1 Front End



Activity Files

Figure. 26

As expected, the front-end, visual aspect of any application is an integral component when looking to build a successful application. When users of our software first experience it, the most apparent and noteworthy feature would be the design and how well the app is laid out. As mentioned in the design section of this report (above), we exhausted a lot of our initial research time in order to get the design element of the application as acceptable as possible. We found that most users really welcomed the idea of having an application which was simplistic in nature yet engaging from afar (**refer pg.55, pg.56, pg.60, pg.63, pg.64 appendix**).

Once we finalised the prototype we would be basing our front-end on, we initiated the implementation stage in Android Studio, using the Java programming language. Using our front-end code design we created the activity (".class") files that were required for each page of our application(s). We then assigned each activity page to different members of the group.

6.1.1 Login Page

Creating the login page was the starting point for our application(s). We used a variety of online videos, which gave us a greater idea of how to create a login system, without yet having any form of storage (backend) for the user credentials (DK, 2017).

Android Studio had a range of widgets which made front end application building more intuitive. For example, we knew we had to have two text fields for the username (password) and password, and a button, in order to submit the credentials and log the user in. We applied the “EditText” (which is an extension of the component “TextView”) and “Button” widgets from the “palette” (Development Tools) provided. This was all conducted in the “activity_login.xml” file, which is automatically generated, when a class file is created. The “.xml” files gave us an idea of how the widgets appeared on the application, as it had two sections: “Design” (which physically displayed a preview of the page) and “Text” (which is commonly used to set id fields of widgets, similar to CSS id’s, and also provide styling to components).

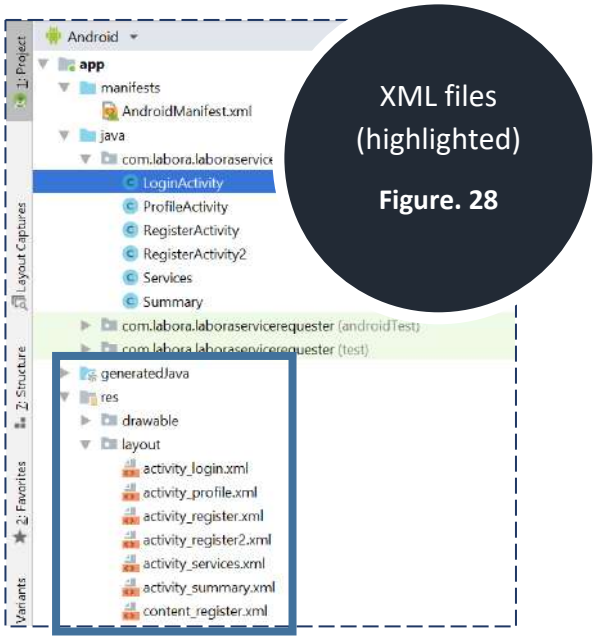
Once we got the login page to resemble our desired outlook, we needed to add dynamic functionality to the cluster of features. In our “LoginActivity.class” file, we initialised and assigned the widgets we implemented, to take the values passed by the user. As we were at the beginning phase of implementation, we did not have a database. Therefore, in order to test our login logic, we needed to set a “dummy” username and password. This was easily handled, as we simply set the “EditText” fields to take the strings we were going to test with (e.g. “test@labora.co.uk” for username and “123456_password” for password). In addition to this, we also had to create a new class file, as the login functionality required a profile page to open up. Due to different members working on different pages, we already had a “ProfileActivity”.

As a result, the logic was adjusted and we supplemented code to enable the application to open the profile page, if the entered credentials matched those set in the “LoginActivity”. Eventually, we had a database, so the credentials were checked against stored values, instead of hard-set values.

```
android:background="@color/white"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".LoginActivity">

<ImageView
    android:id="@+id/imageView"
    android:textAlignment="center"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="44dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.441"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/labora1" />
```

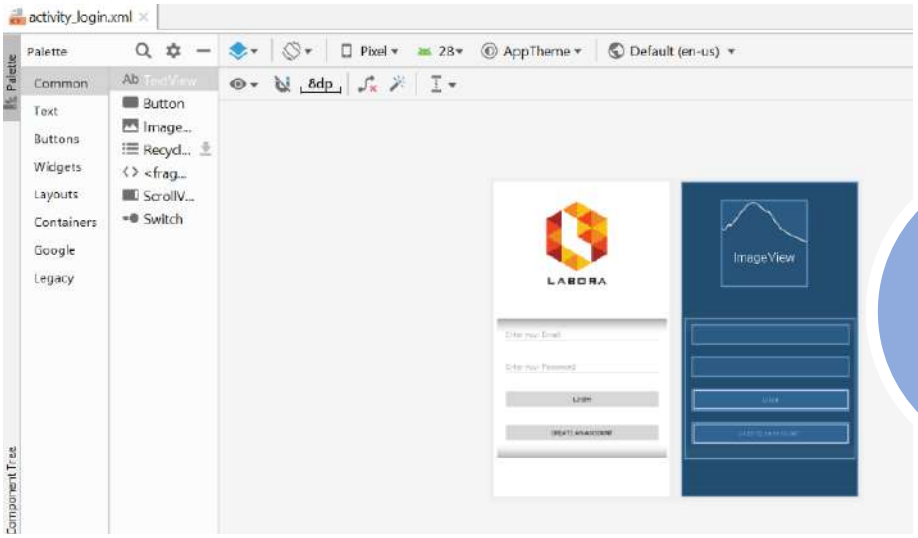
XML code view

Figure. 27



XML files (highlighted)

Figure. 28



Design view of App

Figure. 29

6.1.2 Register, Profile & Confirmation

Similarly, we built the register (“RegisterActivity” and “RegisterActivity2”), profile (“ProfileActivity”) and job confirmation (“SummaryActivity”) pages via the same means as login. They generally consisted of “TextView”, “EditText” and “Button” components. The register page was completed alongside the database, as we had to ensure usernames and passwords entered, would then be able to login in once the application was closed and connection disabled.

The code consisted of conditional logic (if and else statements for error handling), which assessed whether the user’s input has previously been used e.g. in terms of the username. As a result, we completed register page with identical code to that of the login page, which facilitated the opening of the “ProfileActivity”, upon successful registration. The profile page, for both applications operated as our main menu, it had two buttons “Search” (for service requester)”Go Online” (for contractor) and “logout”, which we added using the “Button” widget. It was relatively simple, as the buttons performed the simple task of logging the user out and opening up the “Services”/”Map” activities respectively.

```
private FirebaseAuth firebaseAuth;


// On created method
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);

    // Initialise firebaseauth
    firebaseAuth = FirebaseAuth.getInstance();

    // Initialise progressDialog object
    ProgressDialog = new ProgressDialog( context: this);

    // Initialise register button
    buttonRegister = (Button) findViewById(R.id.buttonRegister);

    // Set variable values
    editTextEmail = (EditText) findViewById(R.id.editTextEmail);
    editTextPassword = (EditText) findViewById(R.id.editTextPassword);

    // Set onClick listeners
    buttonRegister.setOnClickListener(this);

}
```

Register Activity.java **Code**

Figure. 30

```java
// Public class Profile Activity
public class ProfileActivity extends AppCompatActivity implements View.OnClickListener{

    // Initialise variables
    private FirebaseAuth firebaseAuth;
    private TextView textViewUserEmail;
    private Button buttonLogout;
    private Button buttonSearch;

    // On create method
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);

        // Assignment operations
        firebaseAuth = FirebaseAuth.getInstance();

        // Check if current user exists
        if(firebaseAuth.getCurrentUser() == null)
        {
```

ProfileActivity.java **Code**

Figure. 31

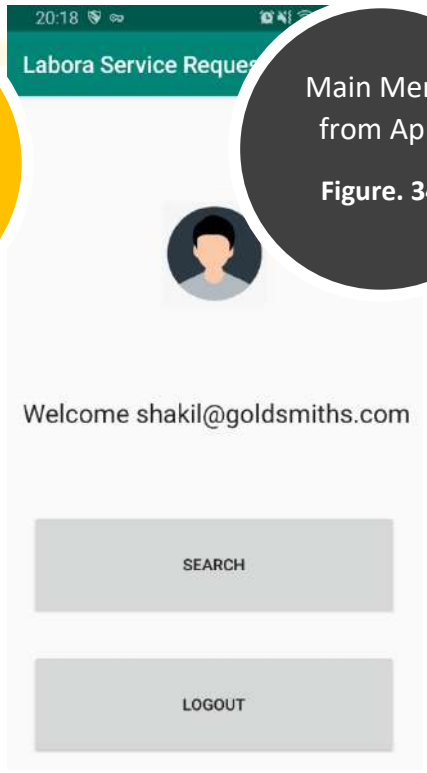
```java
package com.labora.laboraservicerequester;

// Import statement
import ...

// Public class Summary
public class Summary extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_summary);

        // For the back button
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }
}
```

Summary.java **Code**

Figure. 32

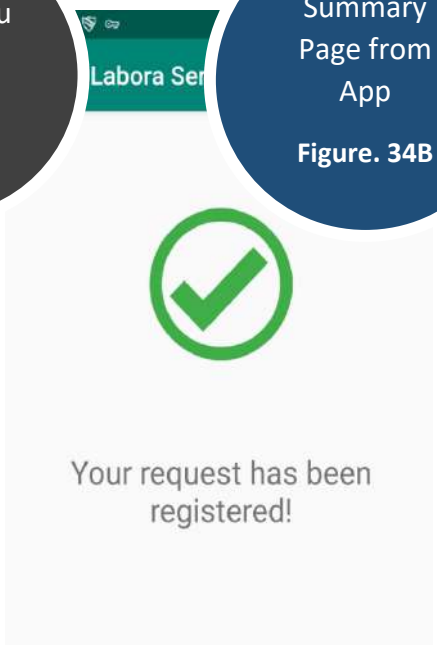
Login Page from App

Figure. 33



Main Menu from App

Figure. 34



Summary Page from App

Figure. 34B

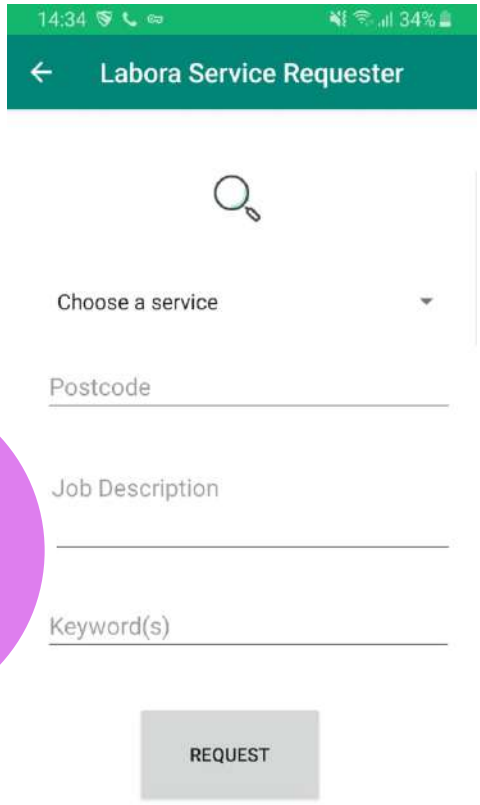
6.1.2 Services Page

The service requester and contractor applications have analogous structures, aside from their primary feature page. In terms of service requester, it's the "ServicesActivity" file. We have implemented this with "EditText" fields, so users can enter details such as their phone number, address and job description. In addition to the text fields, we have used a "Spinner" widget. This acts as a scroll down menu. This feature is suitable for the search area we wanted to implement, as users can select a service they want from a variety of options. The addition of an "onclicklistener" on the spinner meant we could store what the user selected.

Services.java **Code**

Figure. 35

```
//String requesterName = name.getText().toString();
//String requesterPhone = phone.getText().toString();

// Get the id of the user
String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();
//String requesterEmail = FirebaseAuth.getInstance().getCurrentUser().getEmail();


// Conditional to check for empty fields
if(TextUtils.isEmpty(requesterService) || requesterService.equals("Choose a service"))
{
    Toast.makeText( context: Services.this, text: "Please choose a service", Toast.LENGTH_LONG).show();
}
else if(TextUtils.isEmpty(requesterPostCode))
{
    Toast.makeText( context: Services.this, text: "Please fill in a post code", Toast.LENGTH_LONG).show();
}
else if(TextUtils.isEmpty(requesterJob))
{
    Toast.makeText( context: Services.this, text: "Please fill in a job description", Toast.LENGTH_LONG).show();
}
else if(TextUtils.isEmpty(requesterKeyWord))
{
    Toast.makeText( context: Services.this, text: "Please fill in keywords", Toast.LENGTH_LONG).show();
}
else if(!TextUtils.isEmpty(requesterPostCode) && !TextUtils.isEmpty(requesterJob) && !TextUtils.isEmpty(requesterKeyWord) && !TextUtils.isEmpty
```



Services Page from **App**

Figure. 36

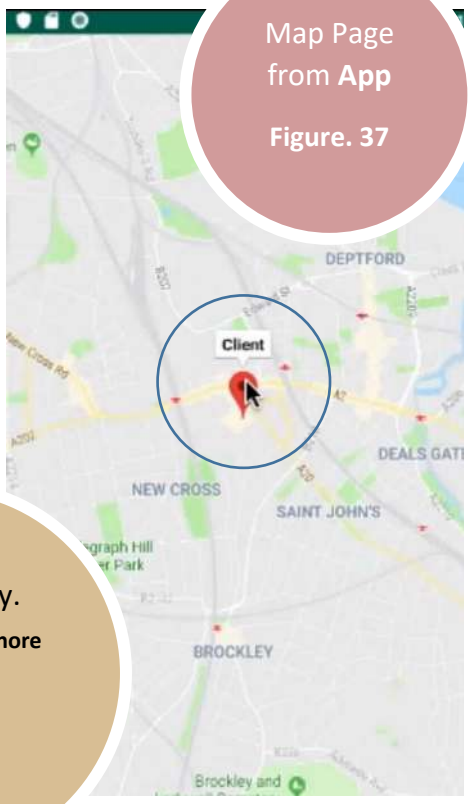
6.1.3 Map Page

As stated above, the “MapActivity” is specific to the contractor application. We have incorporated a Google map API for this page (Google, 2019). At the point of implementing the map, we had already structured our Firebase database. We have implemented the map activity so that it displays the jobs on a map, in the contractor’s general location. Also, we changed the colour depending on the status of the job, this was enforced by editing the colours in the java file. Furthermore, another “onclicklistener” was added to the markers on the map, in order to let the system, know when a user (contractor) has pressed on a job.

```
public void onMapReady(GoogleMap googleMap) {

    //Initialising the object
    map = googleMap;

    //Coordinates hard coded as example for the map function working
    LatLng coords = new LatLng(51.474144, -0.035401);
    LatLng coords2 = new LatLng(51.476557, -0.036270);
    LatLng coords3 = new LatLng(51.472204, -0.034126);


    //Adding the coordinates to the map
    map.addMarker(new MarkerOptions().position(coords).title("Jane Doe, SE14
6NW").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_ORANGE
)));

    //The camera is focused on the points when the user clicks on the 'go online'
button from the previous activity
    map.moveCamera(CameraUpdateFactory.newLatLng(coords));
    map.animateCamera(CameraUpdateFactory.newLatLngZoom(coords,14));

    //Adding the coordinates to the map
    map.addMarker(new MarkerOptions().position(coords2).title("Job has been
accepted").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_G
REEN)));

    //The camera is focused on the points when the user clicks on the 'go online'
button from the previous activity
    map.moveCamera(CameraUpdateFactory.newLatLng(coords2));
    map.animateCamera(CameraUpdateFactory.newLatLngZoom(coords2,14));

    //Adding the coordinates to the map
    map.addMarker(new MarkerOptions().position(coords3).title("Job has been
rejected").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.H
UE)));

    //The camera is focused on the points when the user clicks on the 'go
button from the previous activity
    map.moveCamera(CameraUpdateFactory.newLatLng(coords3));
    map.animateCamera(CameraUpdateFactory.newLatLngZoom(coords3,14));


    //This finds the title and starts the next activity
    map.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener()
        @Override
        public boolean onMarkerClick(Marker marker) {
            if (marker.getTitle().equals("Jane Doe, SE14 6NW"))

            {

                startActivity(new Intent(getApplicationContext(),
SummaryActivity.class));

            }
```

MapActivity.
java **Code** (Ignore red lines)

Figure. 38



Map Page from **App**

Figure. 37

6.2 Back End

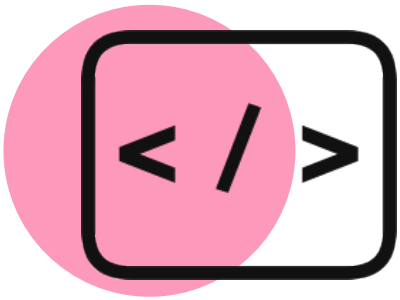
Following on from our front-end development, we also had to develop a backend for our applications(s). Dissimilar to the two-application approach we undertook for the front-end, we built our database in a manner to serve both the service requester and contractor applications. Before we started any code for our database, we needed to integrate it with Android Studio. This was easily achieved by selecting ‘Firebase’, under the tools option. Once we successfully connected our database with our Android development environment, we were then able to start storing objects of data. The first step was to initialise a ‘firebaseAuth’ object (Google, 2019), as we would then use this authentication specific object to save user inputs to our database.

In the “LoginActivity” we used it extract and cross-examine the value of the username and password. If they matched, we used the method “getInstance()”, which pulls the instance of the user, and logs

them into their account. Else, we would inform the user, with a toast message, about incorrect sign in credentials.

Likewise, in “RegisterActivity”, when a user attempted registering a new account, the details were checked against the database for matches. If none were identified, Firebase created an instance of the user and logged them in, whilst storing their credentials, for future sign-ins. Moreover, the complexity of implementation intensified once we reached the “Services” and “Maps” pages of the application(s). Antithetical to the other pages, instead of initialising a ‘firebaseAuth’ object of type ‘FirebaseAuth’, we initialised a ‘FirebaseFirestore’ object called ‘nFirestore’ for “Services”. We then used a map data structure, to map key-value pairs between the service requester’s inputs and the fields on the app. Following on from this, we now needed a method to extract these values from the database and display them on the map.

We tested many online resources and methods, although, we were unable to successfully implement this feature. However, to resolve this issue, we placed multiple markers on the map, which contained information on previously submitted requests. Therefore, we believe we have managed to make it viable for the MVP.



6.3 Unanticipated Complications & Resolving Solutions

As anticipated, throughout our implementation process there were many phases where we experienced complications. We found that many complications arose in the final sprint and towards the last few scrums. Mostly, many of the bugs in the code were easily fixed by additional conditionals or better error handling. For example, we noticed that the phone number field accepted all values. We fixed this by making sure the entered data had to be of type integer. On the other hand, we did experience major issues. For instance, we did not foresee that we would not be able to extract GeoPoint coordinates from the database. We tried to overcome this through utilising resources available to us on Firestore and obtaining values from certain fields. However, to no avail, we could not render our application functional in this manner.

Consequently, we did manage to overcome the problem. We decided in order to meet the MVP standards, we could place markers on the map, which contained data from previously submitted requests. It served a brute-force method, nonetheless, it was an option we took considering the time frame for development. Therefore, we were able to complete our application to a viable level and focus on other features.

```
    //The camera is focused on the points when the user clicks on the 'go online'
button from the previous activity
    map.moveCamera(CameraUpdateFactory.newLatLng(coords));
    map.animateCamera(CameraUpdateFactory.newLatLngZoom(coords,14));

    //Adding the coordinates to the map
    map.addMarker(new MarkerOptions().position(coords2).title("Job has been
accepted").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_G
REEN)));

    //The camera is focused on the points when the user clicks on the 'go online'
button from the previous activity
    map.moveCamera(CameraUpdateFactory.newLatLng(coords2));
    map.animateCamera(CameraUpdateFactory.newLatLngZoom(coords2,14));

    //Adding the coordinates to the map
    map.addMarker(new MarkerOptions().position(coords3).title("Job has been
rejected").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_R
ED)));
```

Markers on the Map Code

Figure. 39

```
//    public GeoPoint getLocationFromAddress() {
//
//        DocumentReference strAddress =
FirebaseFirestore.getInstance().collection("Summary-
ServiceRequester").document("Post Code");
//
//        Geocoder coder = new Geocoder(this);
//        List<Address> address;
//        GeoPoint p1 = null;
//
//        try {
//            address = (List<Address>)
coder.getFromLocationName(String.valueOf(strAddress),5);
//            if(address==null){
//                return null;
//            }
//
//            Address location =address.get(0);
//            location.getLatitude();
//            location.getLongitude();
//
//            System.out.println();
//            p1 = new GeoPoint((double) (location.getLatitude() * 1E6),(double)
(location.getLongitude() * 1E6));
//
//            return p1;
//
//
//        }
//
//        catch (IOException e)
//        {
//            e.printStackTrace();
//            return null;
//        }
//
//    }
```

Conversion of GeoPoint (Attempted)

Figure. 40

7. Testing, Evaluation & QA

7.1 Formative Evaluation – (Introduction)

Our development process involved two of our main stakeholders (contractors and service requesters). We did this by staying in contact with them and found new testers so we can get a contrasting perspective on the software we designed. We understood the significance of testing as it ensures we develop a high-quality product. To fulfil the quality requirements, we looked at the user's needs. We worked collectively with our stakeholders in order to prove a product that is in working order and has a user experience that is feasible for their requirements. In order for us to achieve this, we needed to do rigorous testing. However, before we started the process, we conducted a research on the types of testing which would provide us with the best possible results. The principle of following the agile approach was implemented to testing.

We initially carried out unit testing **(refer to pg.80, pg.84 appendix),** then from there we did integration testing **(refer to pg.124, pg.125 appendix), system testing (refer to pg.126, pg.128 appendix)** and acceptance testing**.** We also conducted robo (automation) testing **(refer to pg. 129 appendix)** as well as user testing **(refer to pg.90, pg.96, pg.104, pg.110 appendix),** as extra forms of testing so we can ensure a product that is fit for purpose.

To accomplish our aim of high-quality end product, we also conducted non-functional testing. These tests were reliability testing**,** usability testing and scalability testing**.** This was all achieved so we can guarantee a high-quality product to our end users.

7.2 User Testing

Communication with our stakeholders increased as we kept in touch to ensure we deliver to the best possible product. We had a select few users who we decided to either meet in person or video chat during development at the end of every sprint to showcase our current progress of our app. Both white box/black box testing were undertaken at the end this period. This feedback would come useful as it influenced the outcome of the final product and to find out whether we needed to make any changes to make the product to the highest degree.

7.2.1 Test Cases

Test cases were designed prior to testing our product. Our users/testers need to understand what parts require testing. It verifies whether the parts of the application is functioning as required to ensuring excellent quality with our frontend and our backend. Users were able to use the test cases to test each part of our application and answer the questionnaire we provided them for further black box testing. Here's an example test case: (Show an example test case from a table)

7.2.2 Black Box

Black Box Testing involves testing the software with our users. It was carried out at the end of sprint 3. We gave users a questionnaire to fill out whilst testing the product. With the users testing it, they would tick the boxes which conforms to the test they have carried out. They would give a mini-feedback to give us a better insight on the outcome of each test they undertook thus ensuring a better-quality product.

The feedback was largely positive and all mentioned that the app works in terms of user experience, functionality and reliability. We quickly resolved the issue of the form being saved in the database correctly with 50% of our users mentioning it doesn't save initially. Minor adjustments in terms of design is required. We had 75% of users who agree the menu was intuitive however, the other 15% were undecided on this. This is something we will take into consideration beyond our MVP because we want to cater our app for all stakeholders and meet our user requirements of a high level of quality end product. (**refer to pg.103, pg.110**). Improvements beyond our MVP also include allowing the user being capable to accept more than However, judging by the feedback as a whole, majority either agreed or strongly agreed with all the questions that was asked ensuring that both apps passes the minimum criterion.

7.2.3 White Box

White Box Testing is testing the software with our software group peers. As software developers, we want to ensure the product is made to the maximum quality and bug free essentially. We undertook the same tests as our users did with our peers filling out questionnaires and being asked the same questions from the test case. We generally found the same pattern where no one from our group faced any major issues with the service requester app and pretty much escaped bug free. (**refer to pg.90, pg.96 appendix**) However, when we tested the contractor side, we found that the map page didn't load the jobs on to the screen. There were issues with GeoPoint coordinates being queried from the Firebase Database to Android Studio. It involved a lot of time and effort in which we were finding difficult to find due to our other commitments and only having four weeks. However, to give the contractor a perspective on how it would look, we temporarily hard-coded it so we could test it for black box testing. We also found out 50% disagreed with the fact that you can apply for more than one job. We seek to resolve these issues beyond our MVP (**refer to pg.90 appendix**).

Our application at this point has been in the hands of many individuals and they tested both contractor and service requester apps rigorously to ensure we get the highest possible outcome in terms of quality. With many opinions, it gives us assurance that our app will appeal to our desired market and users will be satisfied with the overall outcome.

7.3 Unit Testing

Unit Tests was integral in achieving a high-quality outcome. JUnit was crucial in aiding us with unit testing. This was our initial testing strategy as we wanted to test whether the code actually worked. We adopted the test-driven method in which we would note down any errors we found from the bugs and then work on fixing it until it passes. For example, the table below contains a segment from our unit testing, which provides a paramount example. The figure shows that once we identified the fault with the scrolling and user being unable to type in the address field, we worked in a test-driven manner in order resolve the issue at hand. We fixed the issue by having a '*next*' button on the keyboard, this eliminated the scrolling problem. Overall, incremental unit tests allowed us to discover minor or major bugs. Therefore, once we alleviated them, we were assured quality was raised to an exceptional standard.

Test	Test File Name (.java)	Status (Failed/Passed)
Test 1 Test if 'registerUser' function is working (e.g. does it check for empty fields and turn editTexts into strings)	RegisterActivity2Test	X

Check if 'onClick' works and calls 'registerUser'	RegisterActivity2Test	✔
Check if 'buttonRegister' is clicked and registers user	RegisterActivity2Test	✔
Check if after registering, Main Menu page opens up	RegisterActivity2Test	✔
Test 2 Check if name entering field is appearing and working	RegisterActivity2Test	✔
Test 2 Check if phone number entering field is appearing and working	RegisterActivity2Test	✔
Test 2 Check if occupation entering field is appearing and working	RegisterActivity2Test	X
Test 2 Check if address entering field is appearing and working	RegisterActivity2Test	✔
We resolved this by moving the contents within the activity_register2.xml upwards as the test wouldn't scroll downwards and thus wouldn't detect address field. This was just an easy fix – however, in the future we will fix this problem ensuring users can scroll.		

7.4 Integration Testing

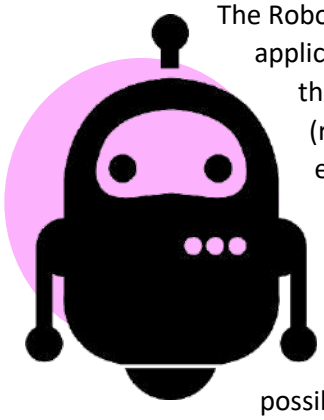
Integration followed on from our unit testing. We required this type of testing, as it meant we could get a clear idea on whether all the individual components we built, functioned together or not. We set out a variety of criteria, mainly from our systems requirements specification (**refer to pg. 71, 72 appendix**). Next, we worked through our application, systematically, ensuring every point of comparison is tried and tested. If successful, we would tick it off, else we would take a similar test-driven approach, like unit testing, and reform the code. As a result, this meant we could ensure that our application worked from the bottom-up. Consequently, we would be more likely to have a robust application(s), faultless app(s).Therefore, improving overall usability and accessibility, which complements our initial objectives and user-requirements, as well as warranting higher quality. We found no significant anomalies with our components and thus, all functioned well together.

7.5 Systematic Testing

Following on from integration testing, we carried out systematic testing (**refer to pg.126, pg.128**). Systematic testing was the largest testing factor which consolidated the quality assurance of our service requester and contractor applications. Utilising a similar structure to integration testing, we carried out the same test cases, however with various different inputs. This was in order to check if we would get unexpected or unwanted results, especially on boundary values. As a result, once we found changes in output, from testing certain values, we inferred that there was a possible defect in our code. Consequently, we would deconstruct the code, and analyse what the problem was, going line by line through the code. Therefore, we would be able to replace the previous effort, with new and superior code. Thus, enabling us to establish an improved level of quality.

7.6 Robo Testing

Furthermore, in addition to step by step waterfall methodological testing, we did automation testing (in the form of Robo tests). As stated in our updated technical architecture, we decided to go with Google's Firebase database backend. This was due to many reasons, outlined in this appendix (**refer to pg.52**), but the main being due to its scalability, flexibility and ease of integration with Android Studio. Firebase had test lab, which is a section for testing an uploaded APK (Google, 2019). Once an APK was uploaded to the testing area, it immediately underwent an automated test run (Google, 2019).



The Robo test went through the application, in essence acting as a user, interacting with our application(s). As a result, we got to test our application from another perspective, other than our and users views. Also, test lab provided us with screenshots and statistics (regarding the speed of activities), which allowed us to get a better insight into how efficiently the application was working. (**refer to pg.129 appendix**). Consequently, we were able to set a benchmark for how well our application should run and what the desired output behaviour should be. Therefore, this meant every time we uploaded the latest version of our APK (Android Application Package), we were able to see how it contrasted with a preceding form.Hence, this meant we were constantly looking to upgrade our quality and provide software at its finest state possible

7.7 How well it conforms

We believe our product meets our customer high standards. Rigorous testing has only led to a product that is made to the best of quality. From Unit Testing down to User Testing our product has been designed to already be released to mass market and hopefully, our product would appeal to many other customers.

Almost certainly, our application will never be bug free; there could be potential bugs that users spot. We hope to have a system where users can report this issue directly. We will try our best to fix it right away ensuring the best experience is found within the app.

7.8 Constraints

As expected in any long-term process, consisting of multiple members simultaneously using software, we faced many constraints. Firstly, in order to begin the technical aspect of our project (coding), all team members were required to download the following software: Android Studio (v3.3 with emulator API Google Pixel XL, running Pie 9.0 operating system) (Summerson, 2018) and Chrome Remote Desktop (v14) (Raphael, 2019). Android Studio was intended for the purpose of providing an android-based coding environment. This would aid us when implementing the application(s), as it meant we could run the software in real time and test whether it suited the outputs we were looking to achieve. In addition, in order to make sharing code more productive and simpler, we linked Android Studio to the Goldsmiths GitLab repository. As a result, any code written which functioned, could be pushed to the repository and pulled/merged by all the members.

However, we came across an issue instantly. We found that multiple group members were concurrently editing parts for their component of the application. Although, this was not an issue, until the *'build.gradle'* (Vogella, 2016) file was edited. The gradle command searches for a file called '*build.gradle*' in the directory you are currently in. It is sometimes called a build (configuration) script. The build script defines a project and its tasks. Some features of the software required the build script to be edited. Therefore, we had to come up with a solution to overcome this.

We eventually decided to take 2 routes if an issue such as this occurred. Route 1 would be to find a different approach to implementing the feature desired e.g. when designing the home page, instead of editing the build file to import a certain design package, you can edit the .XML file. Route 2 would then be to inform the other group members, either during our frequent meetings or via a communication path (e.g. WhatsApp, Call or Email).

This leads onto the other constraint we faced. We discovered that due to the variation in group member abilities when carrying out coding tasks, we would experience situations where we needed to view and debug code in real-time. During our meetings, we were able to solve these shortcomings, however, problems did occur outside of meeting hours.

This meant we needed a means of looking at individual members' code from home, without git pushing/pulling an error. We concluded using Google's Chrome Remote Desktop. This allowed any group member (with permission in advance) to remotely control another member's computer through a proprietary protocol, created by Google. As a result, we could review each other's code and make suitable adjustments, as well as test them in present time. Consequently, this improved our workflow and was time efficient. Therefore, adding to our resolute agile approach.

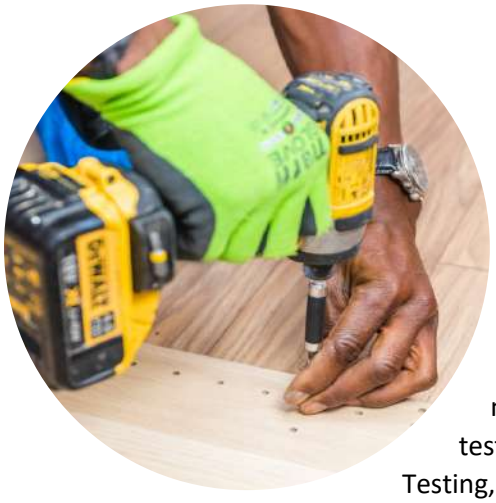
7.9 Functional Requirements Evaluation

Furthermore, once we concluded the various testing methods we applied, we analysed and evaluated the results attained. We found that all functional requirements stated in our functional requirements table (see section 'Analysis'), were completed and satisfied to an adequate standard. Our service requester and contractor applications both had separate criteria for their main functionalities. The service requester app having to be able to submit requests and contractor app having to be able to apply to jobs. We checked at each criterion whether we met them. Consequently, we found our applications performed the functionalities required. Thus, we believe we have successfully implemented the MVP for our product(s).

7.10 Non-Functional Requirements Evaluation

Similarly, for our non-functional requirements, we assessed our application(s) against the non-functional requirements table (see section 'Analysis'). The testing methods we used (as specified in this section above), helped prove that our software met the minimum requirements to be considered an initial version for our app. Unit testing, user testing and systematic testing strengthened our belief in both our application's functionality. As a result, due to the vigorous testing applied, we were able to see the limits of our product. Thereby, we can confirm that the basic requirements have been withheld.

8. Conclusion



8.1 Summative Evaluation

When we had started to develop testable components of our application(s), we needed various testing methods in order to analyse the integrity of each feature. This meant researching methods for testing and putting them in practise (**refer to pg.88 appendix**). Once we established the procedures we wanted to place our code under, we began testing with the identified methods, straight away. As previously mentioned in this document, we required functional testing methods. For the functional aspect we decided to use the following: Unit testing, User testing (black and white box), Integration testing, System Testing, Acceptance testing.

Furthermore, once we carried out the testing methods described above, we were able to collect both qualitative and quantitative data about our service requester and contractor applications. The results were collated, and we visually displayed them on various graphs. Also, we were able to analyse the results to get a general consensus on specific parts of our application(s). Firstly, the unit tests due to their fundamental nature, enabled us to get a clear idea of which vital features we may need to correct (**refer to pg.80, pg.84 appendix**). The appendix extract for unit tests show that we had, by the third set of unit tests, successfully implemented the features required for the MVP. In addition, the user tests (black and white box) (**refer to pg.90, pg.96, pg.104, pg.110 appendix**), gave us a great insight into how users interacted with our application. In addition, it allowed us to get an unbiased view of the app. We had similar criteria for user testing, as we did for the unit testing. The generic view we got was that the application did as it was supposed to and passed the success criteria. However, the overall comments all seemed to suggest our app's lacked certain features. The main two were lack of design and scarce complex functions e.g. rating system or payment area.

Unit tests and both forms of user tests show correlating results, this could be due to the similar structure both methods use. In addition, the integration testing results portray that the separate components we created are able to function together, whilst also outputting satisfactory results (**refer to pg.124, pg.125 appendix**). We found that our front-end and backend (Firebase database) operated accordingly i.e. all data inputted by users was stored, retrieved and displayed accurately. Moreover, systematic testing displayed results which leaned towards a positive outlook (**refer to pg.126, pg.128 appendix**). We found that there were a few bugs on boundary values. For example, when we tried to submit an empty username, and it worked. We changed it with the following code.

```
// Conditional to check for empty fields
if(TextUtils.isEmpty(requesterService) || requesterService.equals("Choose a service"))
{
    Toast.makeText( context: Services.this,  text: "Please choose a service", Toast.LENGTH_LONG).show();
}
else if(TextUtils.isEmpty(requesterPostCode))
{
    Toast.makeText( context: Services.this,  text: "Please fill in a post code", Toast.LENGTH_LONG).show();
}
else if(TextUtils.isEmpty(requesterJob))
{
    Toast.makeText( context: Services.this,  text: "Please fill in a job description", Toast.LENGTH_LONG).show();
}
else if(TextUtils.isEmpty(requesterKeyWord))
{
    Toast.makeText( context: Services.this,  text: "Please fill in keywords", Toast.LENGTH_LONG).show();
}
else if(!TextUtils.isEmpty(requesterPostCode) && !TextUtils.isEmpty(requesterJob) && !TextUtils.isEmpty(requesterKeyWord) &&
    // Initialise Hash Map
```

Checks whether Text is **empty** in the app

Figure. 41

This then resulted in a toast (message on screen) being displayed, informing the user that they need to insert data for the username. As a result, we were able to iteratively solve our issues. Consequently, getting our application(s) closer to a deployable stage.

This brings us onto the conclusion we made after we collated our results and analysed the different data. On one hand we found that our application met most of the requirements that we defined in our system requirements specification. However, on the other hand, we found that there were elements of our application we needed to fix instantly, or else it would reduce user usability as well as render our applications' main purposes, to request and provide services, void. Unit testing, systematic testing and acceptance testing all portrayed analogous results. These testing methods are specifically designed to find faults and bugs which may not necessarily be syntactical but semantically. For example, the figure below taken from one of our unit tests shows how within our '*Service.class*' file we had a spinner. In android studio, there is no hint for the spinner object. As a result, we had to set the first element in the array to a different colour and give it the value our hint would have taken i.e. choose a service. Syntactically everything was correct, nonetheless, when we ran the service request application, we immediately noticed that 'choose your service' was selectable as a service option. We then corrected this issue with the code in the diagram below:

```
final Spinner spinner = findViewById(R.id.spinner1);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource
        ( context: this, R.array.services, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
spinner.setOnItemSelectedListener(this);
```

```
// For the spinner (This is required by Android Studio)
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
{
    // Conditional if statement
    if(parent.getItemAtPosition(position).equals("Choose a service"))
    {
        // do nothing
    }
    // Conditional else statement
    else {
        String text = parent.getItemAtPosition(position).toString();
        //Toast.makeText(parent.getContext(), text, Toast.LENGTH_SHORT).show();
    }
}

// For the spinner (This is required by Android Studio)
@Override
public void onNothingSelected(AdapterView<?> parent)
{
    // do nothing
}
```

Spinner Code

Figure. 42

8.2 Future Developments Strategy and Plan

What's more after collecting, analysing and evaluating our progress so far in this project, we have been able to arrive at many conclusions. We have spared a thought for future development plans, and we believe, given a longer time frame and development period, we would be able to broaden our existing software into something which is more of a polished, end-article product (**refer to pg. 131, 132 appendix**).

From the appendix provided, you can see how we plan to use our backlog (**refer to pg. 48 appendix),** to cumulatively bring in the additional features, at every new instalment. Features include the predictive search, video calling service and payment area. These can all be achieved over multiple sprints and scrums. Additionally, due to our initial planning of development languages (Android) and database choice (Firebase), we have found that we will benefit in terms of scalability, as both environments support an increase in activity from users. Furthermore, using the DevOps philosophy, we will aim to build our application methodologically. Initially planning the features, then coding small components. We will ensure it is the right amount, so that it is testable. After that we need to integrate it with our previously built elements. Once this is done, we can then test the application and judge whether it is at a releasable state or not. Therefore, following an agile work process whilst maintaining a constant rate of progress.

In addition, another improvement we could make would be transposing the contractor application into an online web application. This is due to the fact we have problems implementing "GeoPoints" in Android Studio. We could build a flask application, which would incorporate restful and simple API's. As a result, this would improve our current contractor application. Consequently, users may have a better experience, visually and interactively. Therefore, raising the standard of our software. In addition, we have made a deployment plan in order to demonstrate how we would distribute our application (**refer to pg. 130**)

8.3 Overall Conclusion

Overall, from the results we were able to get a clear understanding of the deficiencies our software had. Also, it granted us the ability to view which elements we succeeded in building without flaw. Most ultimately, we can see from the general user feedback and statistics, that users were content with our application being of at least Minimum Viable Product (MVP) standard. In spite of this, we still received beneficial constructive criticism about the application such as the design element (leading onto usability and accessibility for different user groups e.g. colour-blind people) and more features are desired (most popular being rating system and payment area).

As a result, we can infer that users believe our current app(s) is at a satisfactory standard, however, they would like to see the feedback be put in action, in order to increase their preference towards our software compared to others. Consequently, we can build an extension of our current MVP. Therefore, incrementally developing, via agile workflow, and thereby steadily developing an entirely deployable rendition of software.

9. Appendix

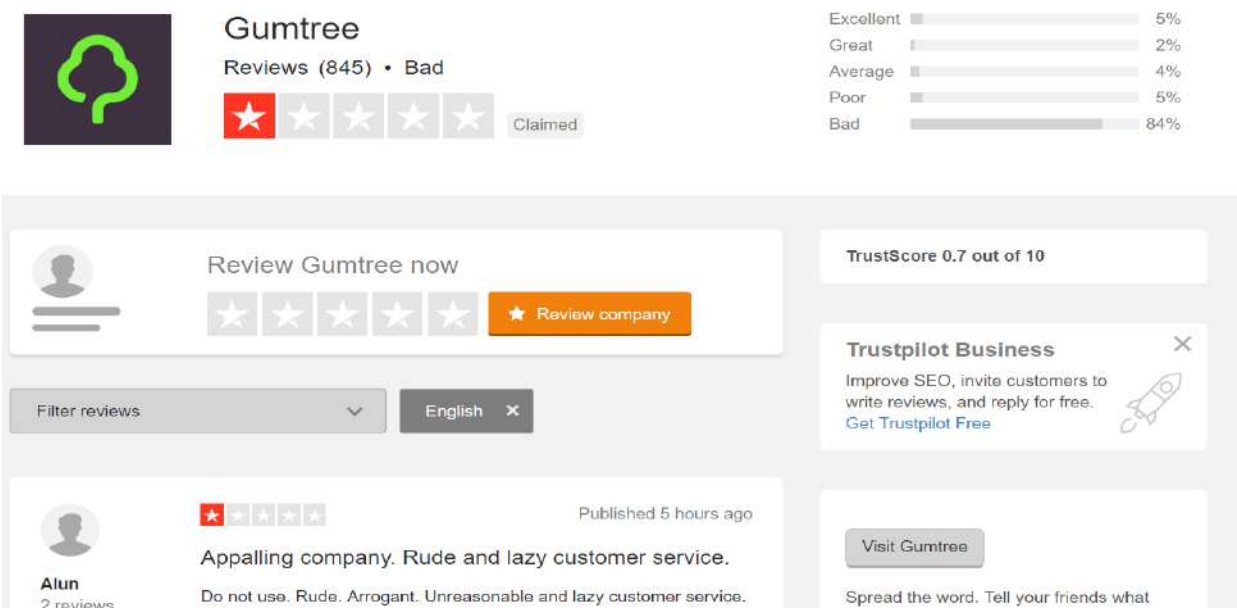
Milestone 1B: Software Project: Extensive Market Research

Introduction

Our product/service is a general services app which could be used within a local community or on a national scale. It will allow users to post a need they have, for example emergency plumbing. This will then show up on the apps' system and people qualified or with the right skill set, will have access to view the job. There will be two types of user accounts, one for people requesting a service and one for people providing it. Our app is based around the same principle as uber and uber eats, which works by making the consumer advertise their needs.

Market & Competition

We are entering a market where it is not fully saturated, and there is competition. Our competition includes websites such as Gumtree, council websites and apps on various app stores (e.g. NextDoor). Although there are not many websites, we have a major competitor in Gumtree. They provide services ranging from tutoring to childcare, and their brand name gives them a slight upper hand. However, although they provide a place for providers to advertise their services, they are not operating in the same way that we wish to operate. We are a request services app, whereas they are a more search-and-pick oriented platform. Therefore, they are not a direct competitor for out app.



Furthermore, we researched Gumtrees' reviews online, and we got the overall feeling that Gumtree lacked the simplicity and customer support that was needed, when providing a platform for services online. Most of the negative reviews for services came from the fact that service providers, who paid more for their ads, got their company or service advertisement posted at the top of the searches. Although this seems reasonable in business terms, this does not ensure customers who are looking for contractors, are finding the most suitable contractors for them. This practically narrows the

customers options, especially if the service requestor has limited technical and research skills. The use of our app would mean customers receive a full list of contractors who are willing to do the job, and at the customers budget and specifications. As a result, this means the customer can then save time, instead of having to spend hours looking for a suitable person to do the job. Consequently, this eases the process for both providers and requestors, as they have a platform wherein, they do business under compatible circumstances.

Regulations & Standards

We have researched what rights consumers and providers have when exchanging money for services online. The minimum information to be provided by the contractor includes:

- Name of service provider(s)
- Geographical address
- Email address
- Registration number
- VAT number
- Details of any professional body/institution with which service provider registered
- Prices must be clear and unambiguous (also including whether prices are inclusive of tax and delivery costs).

The Consumer Contracts (Information, cancellation and Additional Charges) Regulations came into force 13th June 2014. This regulation has been put in place to regulate most contracts made between a 'trader' and a 'consumer'. Within our app, we will also have a terms and conditions, which both types of users (service requestor and service provider) must accept if they want to create an account. Therefore, this ensures that we are not liable for any damages or issues between consumers and contractors, and we are merely an intermediary for bringing two parties together.

Theory and Research: Academic Search

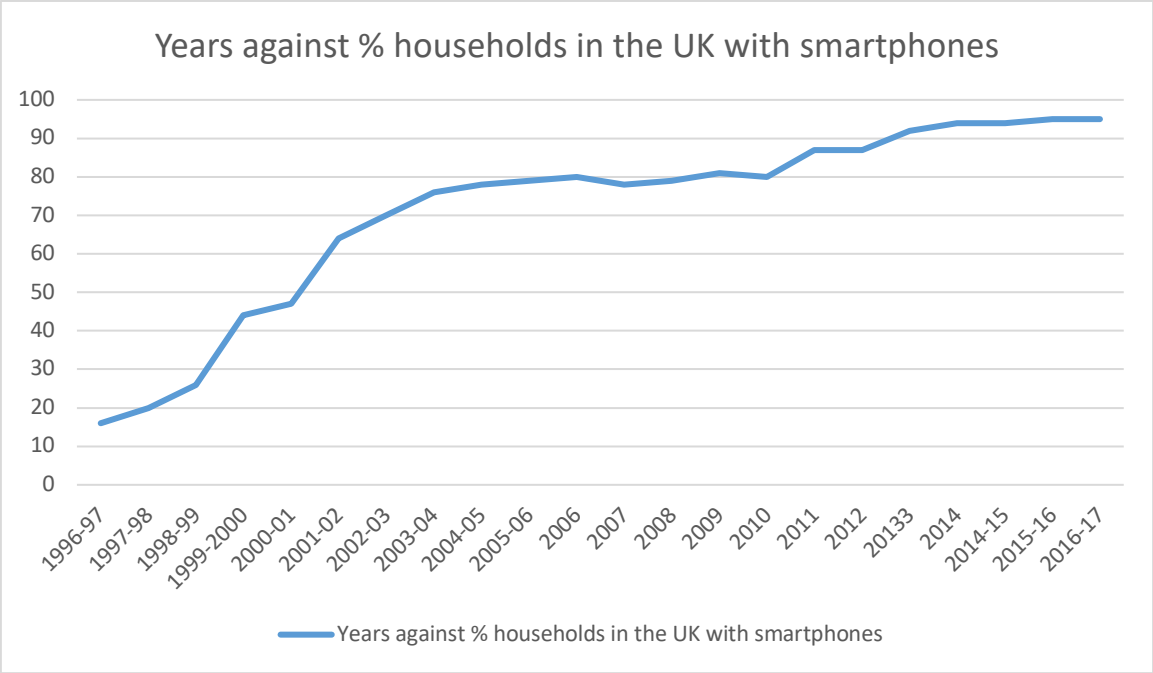
Moreover, we did additional research into whether the apps' general idea has been investigated. We used Google scholars to find any research into whether communication systems have been built to link contractors and service requestors. Although we did not find anything specifically related to our general service request app, we did find research regarding matching mobile users and service providers. The research is based on users pulling profiles from a data base. This is like how we will be storing contractors with a profile, and requestors choosing who they want to carry out the task from a list of profiles.

We believe the research conducted is good as it has been patented. The inventor of the technology is Hirohisa A. Tananka, who has a Ph.D. from Stanford University (2002) and an A.B from Harvard University (1997). He has been employed across many universities as a professor from 2003 up till the present day, where he is now a professor in the University of Toronto. There has not been extensive or specific research into a general service request platform/application. Therefore, we still strongly feel that there is a market for our app and that there is no direct competition.

Analysis

The service request app we are aiming to create is quite convenient for modern times, as almost everyone nowadays has access to the internet or owns a device which enables them to get online. It completely gets rid of the need for business phone books (e.g. Yellow pages). Also, it may even become the better alternative to service locator websites (e.g. Gumtree). This is because, our app doesn't require the customer to have 'good' technical skills. Instead, they are just a click of a button away from getting a whole list of contractors that suit their budget and specifications. However, we do accept the fact we must adapt out app to accommodate for the groups of different users (e.g. older users). For instance, doing research on the average age of app users, we found that 18-24-year

olds spend 93.5 hours or 12.5% of their time using apps, compared to 55-64-year olds, who spend just over half of that same time, using apps (55.6 hours). We believe this is due to the older



generation have less technical skills, which are needed to utilise apps. We have created the graph below which shows how the percentage of households in the UK which own smartphones increased from 1996 to 1997 (data was provided by Statista**):**

From the line graph above we see that practically every household in the UK has a smartphone. However, the problem doesn't lie in not having access to apps, it comes down to the accessibility. Users may not be able to see, hear, move, or not be able to process some types of information easily at all. There are many reasons why users can find it hard to efficiently access and use an app.

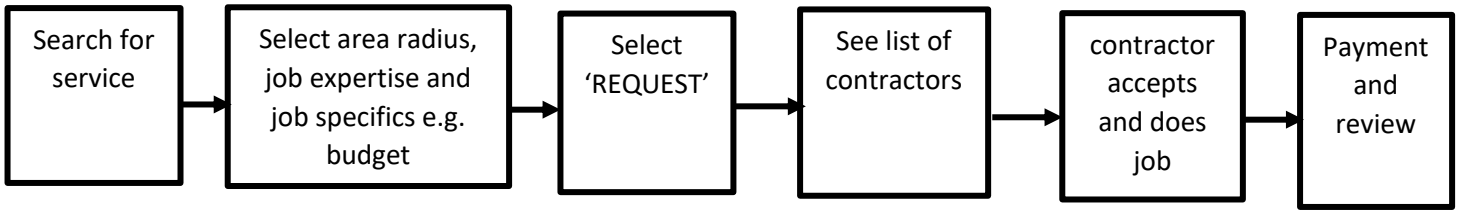
Before 1910	1990-99	2000-09
Vacuum Cleaner	Scanning Systems	MP3
Air Conditioning	**Website**	Wikipedia
Electrocardiogram	MEGA 1	BitTorrent
Radar	Linux OS	Kindle
Tea Bags	Iris Scanning	Touchscreen
Business Directories	VoIP	**Mobile Apps**
Plastic	Online Radio	OLPC
Electric Washing Machine	eBay	Electronic Voting
Radio Broadcasting	Wi-Fi	Vacuum Robot

Furthermore, we have created a table of inventions from before 1910 all the way up to the present year. Above are 3 columns from the full table (find the full table under 'Extracts'). They each contain an invention which is/was used to contact contractors in that era. We have highlighted them in red. We believe, once we have conducted initial interviews and further questionnaires, the younger age groups will respond that they have used applications or websites, compared to the older aged categories, who will probably have used business directories to contact service providers. This leads us to believe certain age categories have a preferred method. As a result, we must build our application in a manner which caters and facilitates all age groups. One way we can do this is by changing the interface of the app depending on the age of the user (e.g. if it's a user, aged 52, the interface must use large print, as well as less complex looking navigation system). Most ultimately, we can only create an effective app once our research has been conducted and analysed. Until then we can only assume and infer what users prefer.

Backlog

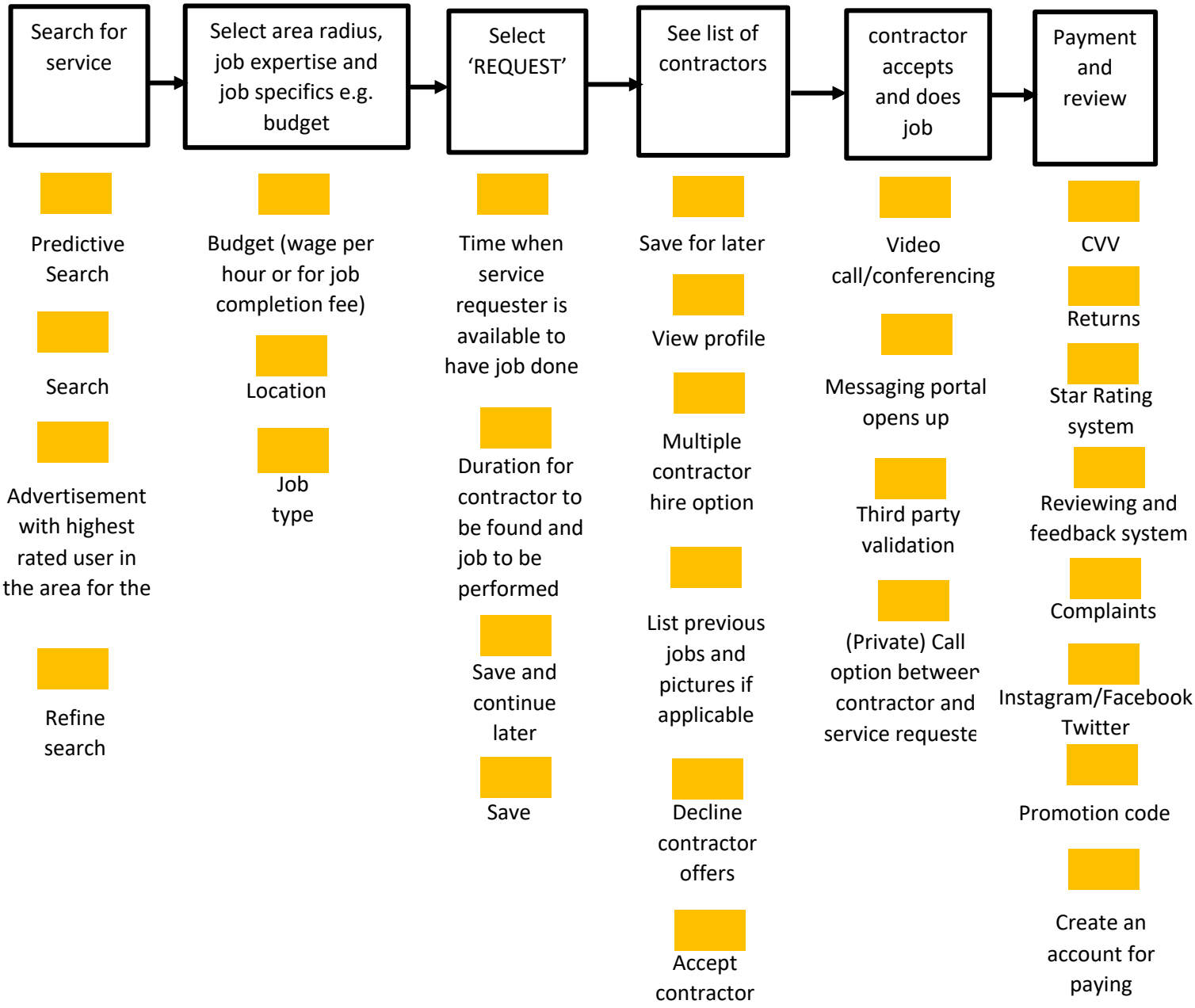
Backlog (1):

- Detail the core functionality as a work flow



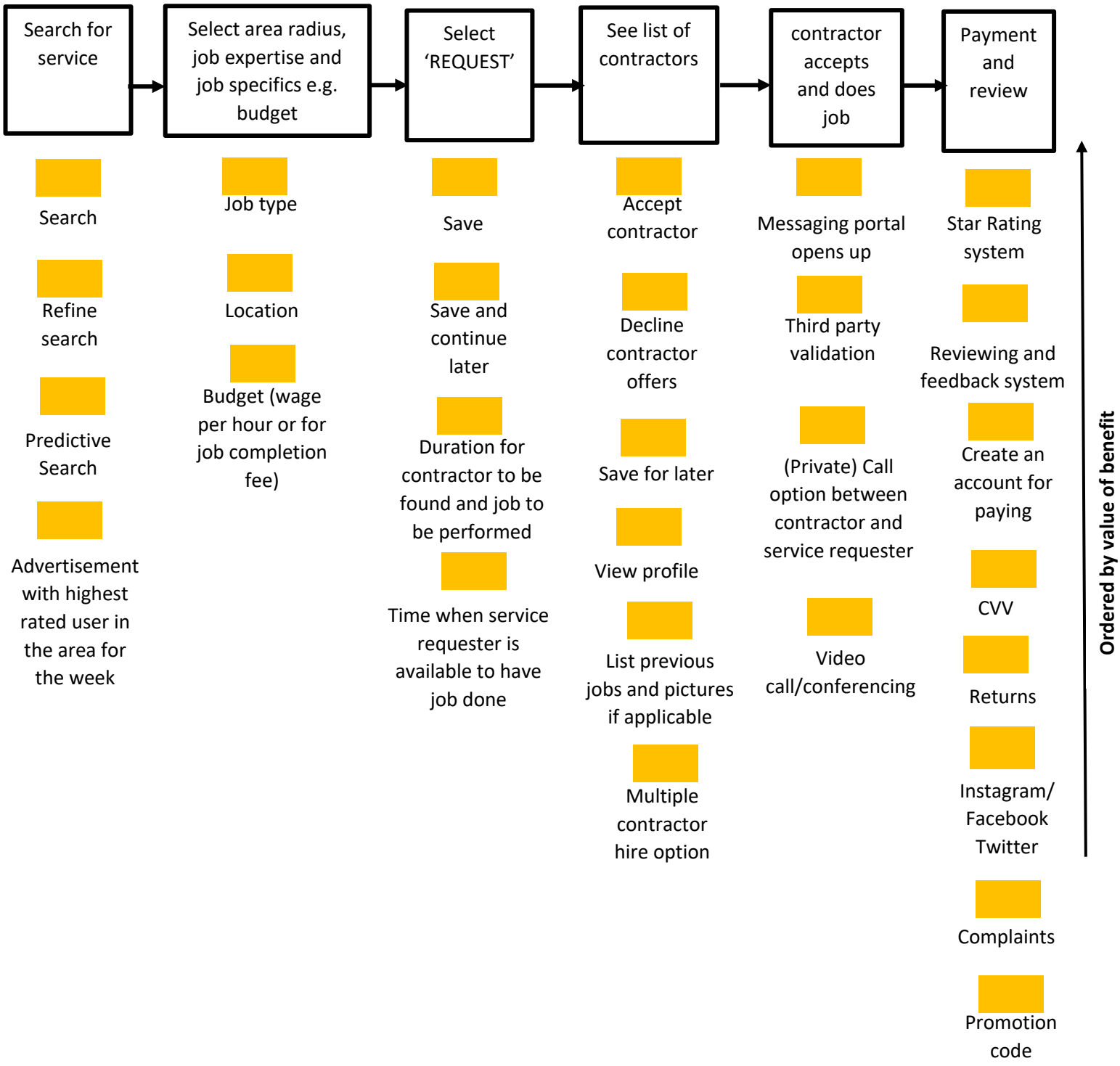
Backlog (2):

- Product Feature Group (what is done at each step)



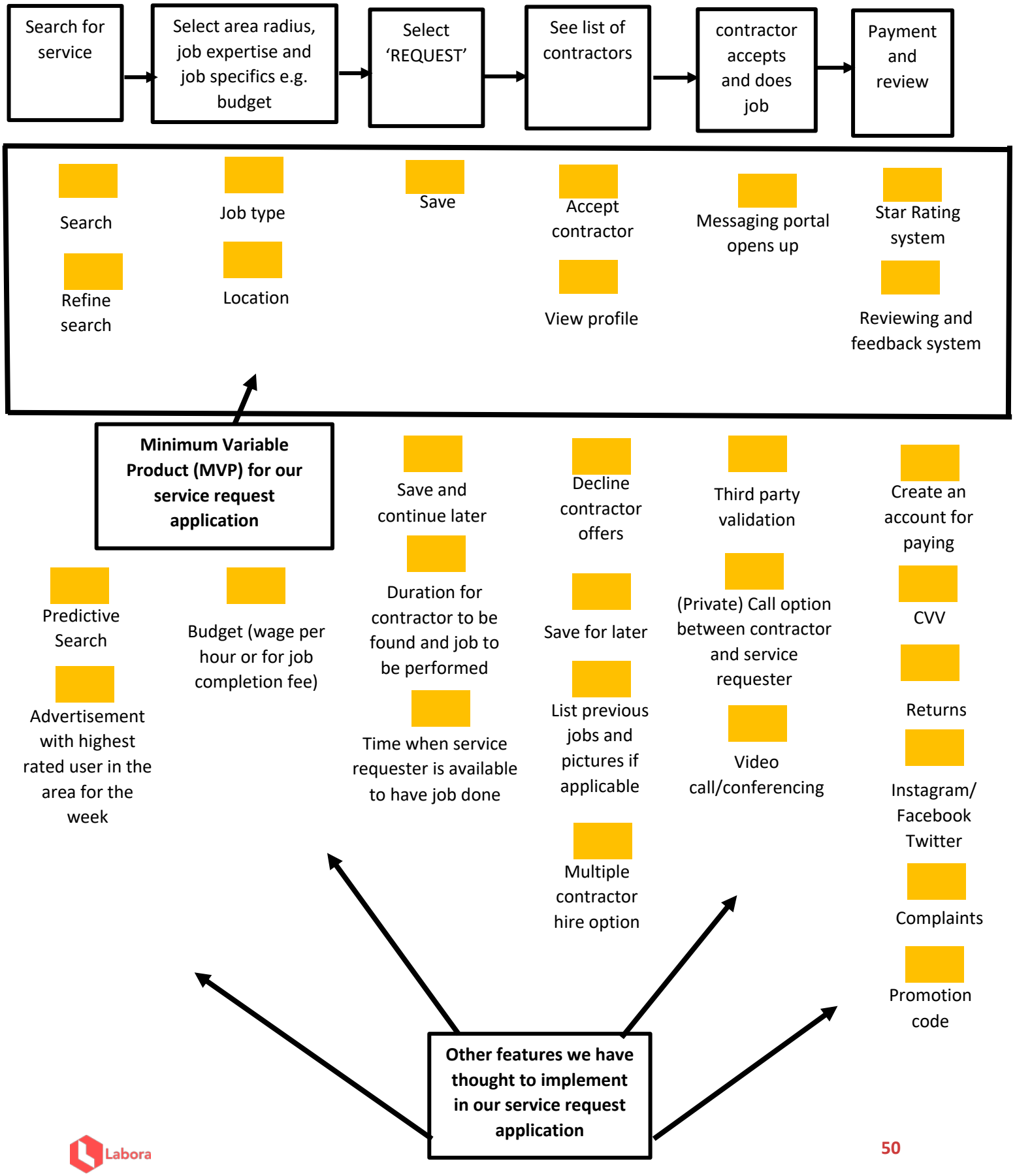
Backlog (3):

- Prioritise Features



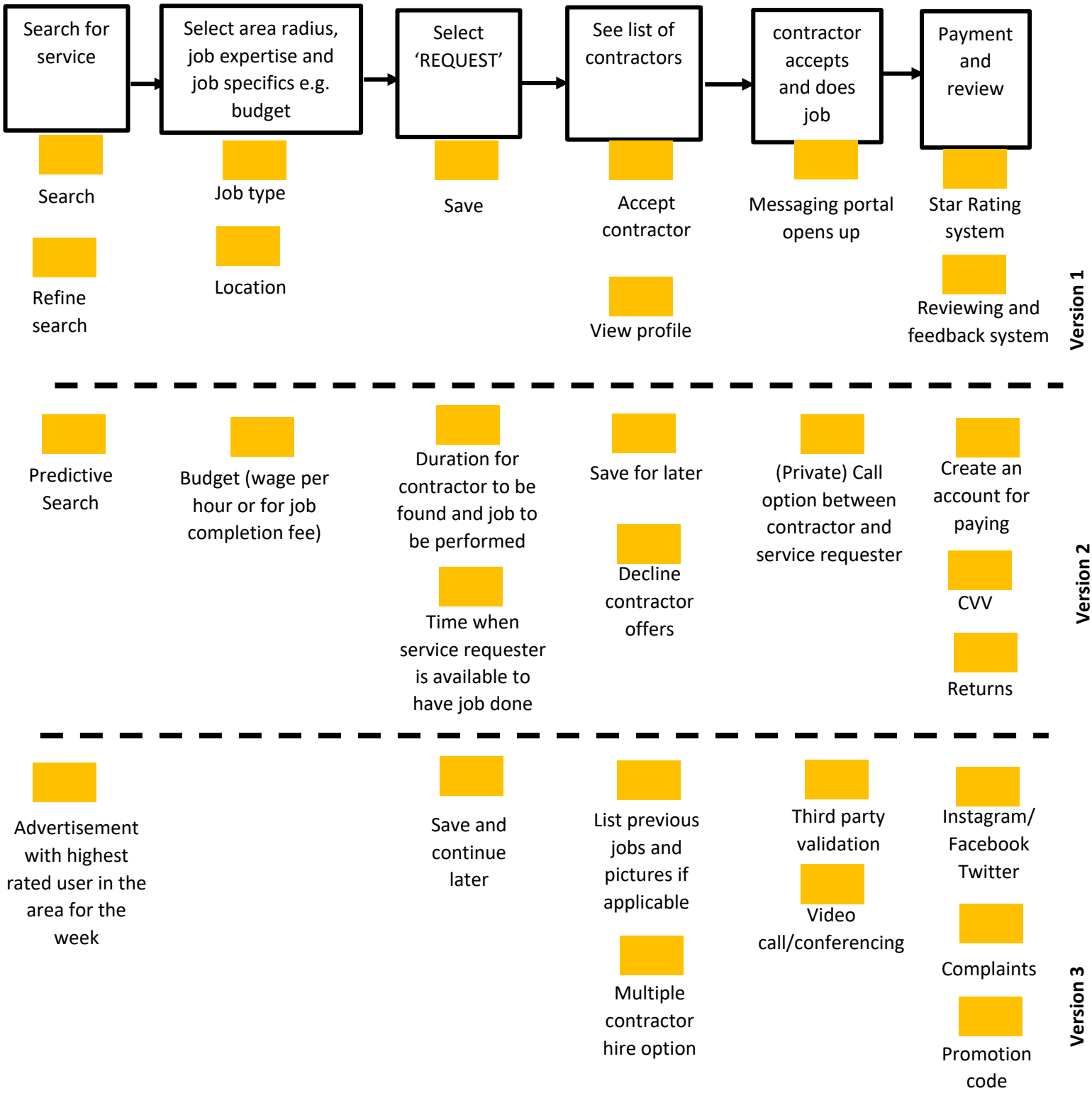
Backlog (4):

- Identity First Delivery



Backlog (5):

- Identify Subsequent Delivery



MongoDB VS Firebase (Firestore)

Initially, we decided to pick between SQL (Structured Query Language) and MongoDB . Our initial research saw us conclude that we would implement MongoDB backend. After further research during our project development, we discovered a platform, Google Firebase, which was more suited and efficient to our android java-based application. This document will outline the advantages of both backend languages and then come to a summary of why we decided to adapt our technical architecture.

MongoDB advantages:

- Schema less – document database, holds different documents
- Structure of single object clear
- Query ability
- Ease of scale-out, easily scalable
- Maps application objects to database

MongoDB disadvantages:

- Joins in MongoDB are not supported
- High memory usage
- Limited data size
- Limited nesting

Firebase (Firestore) advantages:

- Specifically created for mobile application development
- Integrates well with android studio
- Stores data into documents and organised collections
- Scalable
- Single object in hierarchy

Firebase (Firestore) disadvantages:

- Storage format different to other languages (Firebase uses JSON), so migrating is harder
- Reporting tools not same standard as other backend languages
- Costs (Limited to 50 connections and 100MB of storage)

The comparisons above show the few pros and cons of both database platforms/languages. In order to come to an overall conclusion and decide which would maximise our output, we decided to go with the backend provider which worked most competently with android studio (our front end). We found that Firestore (Firebase) was most appropriate in this situation. As well as effectively storing our application(s) data and transactions, it also has additional features, which would aid us in future. For example, there is a section called 'test lab'. Test lab enables users to upload an APK (Android Package Kit) and test it. It then runs an automated robo test on the application. Once the test is complete, test logs, screenshots, crawl graphs and videos are outputted. These all portray how successful (or unsuccessful) the test has been.

Overall, most ultimately, a combination of the aforementioned points and additional services provided by Google's Firebase, result in us choosing Firestore (Firebase) as our backend. We intend to use it in a way which will benefit our application development as well as improve the quality of testing we do against our application(s).

Milestone 2A: Software Project: Stakeholders Requirements

In order to build a successful application, we need to take into consideration all stakeholders when trying to develop this platform. We looked at the potential stakeholders and their needs.

Our **main** stakeholders will revolve around two user bases

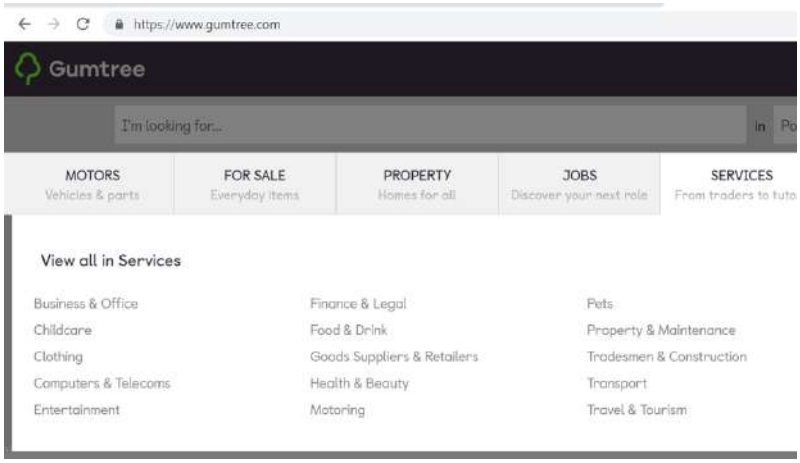
- Freelance Workers
- Customer: Individuals who request for the services for the freelance workers.
  - An example of a usual customer would be a homeowner

Finding stakeholder requirements

- Internet research
  - Problems people face when finding workers or finding customers
- Questionnaires
  - Making a Google Form which our stakeholders will fill in which will give us a clearer picture on what problems people are facing and how we can meet their needs.

The way we've found out about our stakeholders is to look at real-life issues people face. For example, we found out what issues freelance workers face when trying to find work as they're often unnoticed as they may be start-ups. Also, websites such as checkatrade.com are often hard to advertise for start-ups. In terms of customers and their problems, we found out through internet research that they often can't find workers who can find their time frame.

We have conducted our initial external market research activities. We discovered that there are some competitors (e.g. NextDoor app). However, our biggest competitor was Gumtree (https://www.gumtree.com/), a British online classified advertisement and community website. They posed the biggest threat to our app out of all the other applications/websites. Although they allowed people to post their services online, they did not have an area where consumers can post their needs.



From the screenshot above you can see Gumtree have an area to browse the services people are offering. However, no area for anyone looking to request a service. In addition, when looking at customer reviews I have noticed that there were many negative reviews regarding hoax advertisers. We believe with our app we can eliminate the problem of fraud. This can be achieved by making sure all those who are signing up to provide a service, create a profile, including certification and documentation of their profession. We also found out that the Consumer Contracts Regulations

came into force 13th June 2014, which protects the rights of the consumer when requesting services online. In addition, when trying to research 'local service request systems/application' on google scholars, the results were quite vague. This led us to believe that our market is equally quite dubious, possibly because most providers are making a platform for providers. Whereas, we intend to put consumers (service requesters) at the centre of our operation. We do believe our app is viable and if we enter the market early and build a good brand name, we can become successful. However, due to the current nature and speed of the technology industry, eventually many more competitors, including existing large firms can implement a similar system to ours.

Draft a short paragraph that succinctly describes your requirements and market activities.

With the way of the freelancer approaching the customer, it will allow them to find work easily because it will be the customer requesting; the freelancer can easily fulfil that request by touching a button. This also solves the problem for the customer as they can find workers who fit their timeframe and won't need to go through the excruciating process of calling every worker to see they're busy or not. Also, a benefit for the customer is that freelancers are cheaper and completely flexible.

As a result, both stakeholder needs are being met.

Our market activities included initially searching for competitors which either platformed the same service as us or similar services. We found through our opening search online and on app stores (e.g. Google Play store) that there were a few companies providing general services to consumers (e.g. Skillvo Local Services Network and Gumtree). However, we quickly realised that not a lot of them created it in a way which allowed a user to advertise their need. After conducting further market research in the form of customer reviews and google scholar, we found that there are regulations in place to protect consumers when purchasing services/good online. Therefore, we have concluded that there is a requirement for our app in the current market and that it is also viable, due to the lack of direct competition.

The relation between stakeholders and application

Although there will be one app for the users to download, the app itself will be catered to both of our main stakeholders. The freelance worker will have special privileges and will be assigned to a membership that will allow them to find customers who request for the services through the GPS map system that we will be incorporated. They can also add information about their job. The customer will have a different interface. For example, they will be given options on what service they require and how much they're willing to pay. They don't have access to the GPS map system. Both users will have a messaging system as a means of communication.

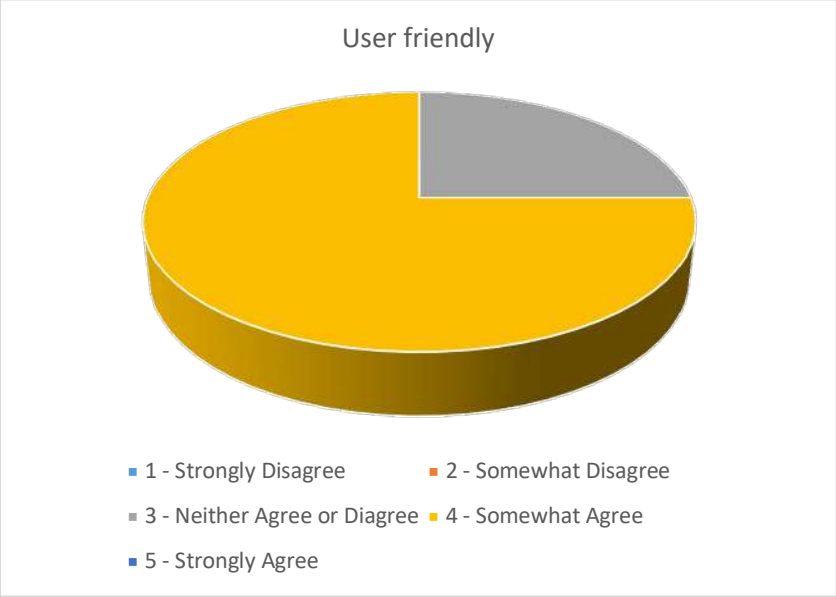
Analysis Prototype 1 - Interviews

After we finish the first prototype, we interview people and let them use the application to get feedback from them. ¾ of the people agreed that the application is simple to use. All of them agreed that it is very easy to navigate. They liked the logo and the design of the application. From the people we interviewed we figured out that the application need improvement most of the people did not find it enjoyable. They found the application attractive however it needs some improvement. Most of them said the application is in good content and they would like to use it in the future and recommend it to their families and friends. Most of the customer would like to use the application rather than spending so much time searching on websites and we need to improve the security of the application and find a way to verify the contractor's certificate. However, they recommend it to improve the application more before using it.

Analysis - Prototype 2 Interviews

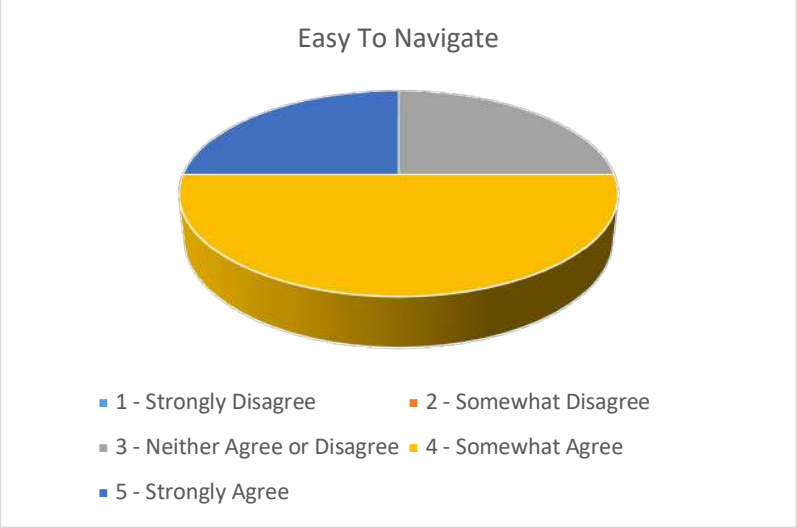
We started our project with an initial 'First' interview and 'First' questionnaire. We collated these results and analysis and created two versions of our service request applications, via Adobe XD. We then needed to test these two prototypes. We did this with the 2 methods: Face-To-Face Interviews and Questionnaires. This document will aim to analyse the feedback interviews, conducted for prototype 2 testing, of our service request application.

The interview has a combination of closed and open questions. This is since we would like to collect quantitative and qualitative data. The questions are related, so we can make direct comparisons and correlations between results. Below are graphs for each criterion, and the number the interviewee awarded them. It is based on how strongly they agreed/disagreed. Along with this you will find a summary of the feedback provided by the interviewee and analysis of what the results portray.

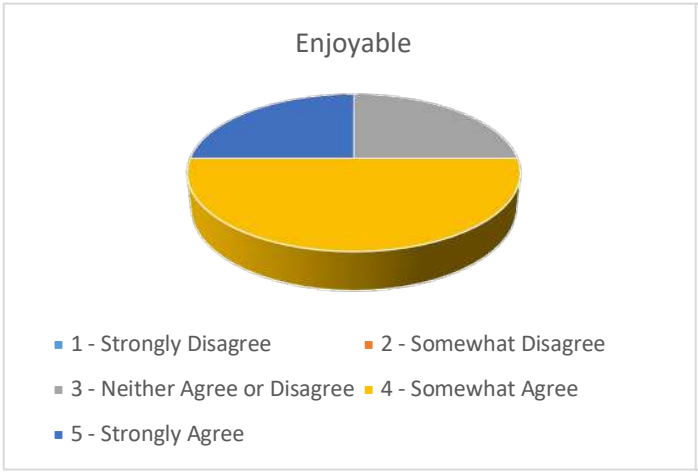


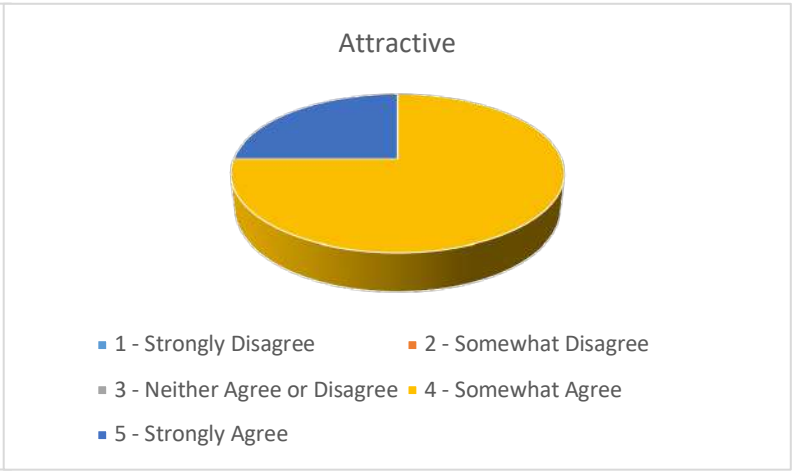
This was the first criteria we had on our feedback table. As you can see from the pie chart, 25% of the testers somewhat agreed that the prototype was user friendly. This leads us to believe the interface is relatively easy to use and can be used by many different age groups. Also, from the longer, open questions after the feedback table, the general comments we got were similar, in the sense that the prototype was quite user friendly. They found that once the menu was found, it became easy to use. Therefore, we have concluded that we shouldn't change the friendly aspect of the interface too significantly, as it is seeming successful.

The next question and feedback criterion were regarding how easy it is to navigate. From the graph below, you can see that half of the interviewees thought that it was easy to navigate. The comments correlated with these scores. Interviewees stated that once they found the menu, they were able to freely navigate the application. As a result, this means we must improve the display for the initial menu drop down icon. Consequently, this will make it easier for users of our application to fully utilise it. Therefore, we have made the decision to research and develop our existing menu icon for the future increase in accessibility for users.

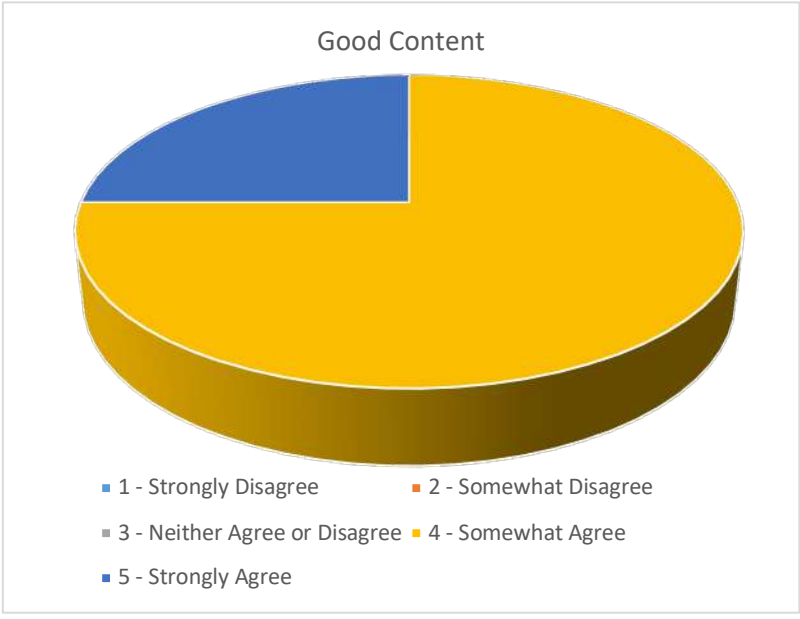


Furthermore, we come to the criterion and question about the enjoyable aspect of the application. As the graph below shows, 50% of the interviewees somewhat agreed. This tells us that half of the testers found this prototype either unpleasant or had no 'fun' when testing. However, interviewee number 4 stated 'It is more enjoyable than searching on Gumtree'. As a result, we find that our app's enjoyability is befitting, but there is space for improvement. Therefore, although it is easier for some to use when attaining services, we can do more e.g. animations, default filters which will make it enjoyable, yet usable, for all our users.



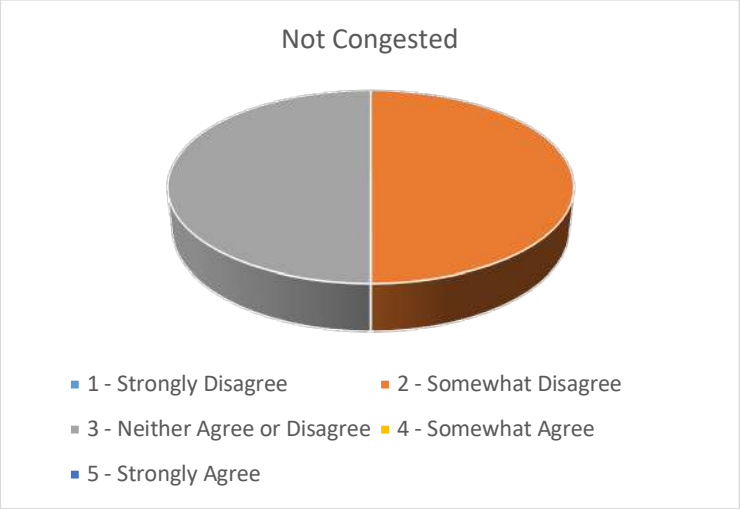


The graph above displays the results for if the testers found our prototype 2 attractive. 75% of testers agreed that the prototype was attractive. We also got feedback from an individual who believed this prototype was very professional. This leads us to believe that our prototype is quite alluring. As a result, because we didn't receive too much criticism for the visual aspect of our prototype, we will try to keep most features in our final service request application. Consequently, this means that we will have more time to develop other parts of our application. Therefore, we made the decision not to significantly change anything appearance wise.

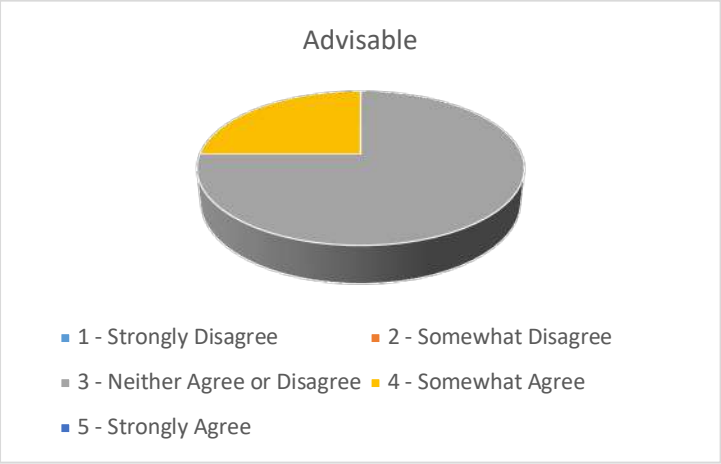


Moreover, we move onto analysing the results for good content. The graph above shows that 100% agreed that our content on the prototype was good and related. The feedback also suggested that the content we provided is complementary. We can see from this that no changes need to be for the content section of our application. As a result, we will pass this on when creating our final version of the app. Therefore, we have agreed that no changes, unless necessary, will be done to our service request related content.

The next criterion on the list is in relation to whether our prototype version 2 is not congested. The pie chart below demonstrates how testers found our application to be quite congested. No one agreed that our prototype was not congested. As a result, we can infer that because previously the same testers agreed the app was attractive and user friendly, the factor holding them back from fulling agreeing was the fact that our home page was quite congested with content. Consequently, we will reduce the number of items on a page at a time. Therefore, this may cause our application to become easier on eye. Thus, improving accessibility and usability.

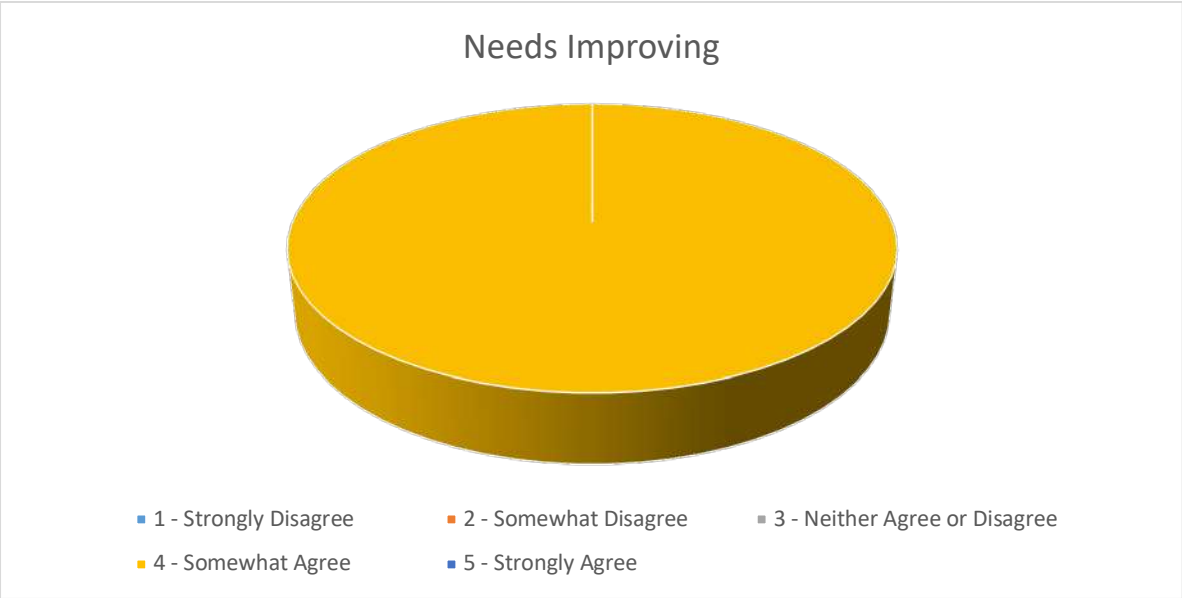


The penultimate criterion and question were regarding advisability. The pie chart quite clearly reflects that the testers are not sure whether they would recommend our application or not, based on their initial experience with our application. 75% of users neither agreed nor disagreed that they would recommend our app. As a result, this shows us that we have not built something which can ultimately replace their existing methods for requesting a service. Consequently, this means we must improve and develop our current prototype and eventual application, so that it not only caters to all potential users, but also takes the place of their current processes. Also, from the open question relating to advisability, many of the testers believed they would recommend it to friends and family, however, to selected members, as they thought the prototype to be too complex for some individuals. Therefore, this section of feedback will be pivotal in achieving our target, of creating an application which is user friendly, yet effective in requesting and receiving a service.



The final criterion and question were perhaps the most significant and most important for the future and direction of our service request application. The pie diagram below shows how all testers of our prototype believed there was a need for improvement. 100 % of prototype testers somewhat agreed we had to evolve our current version. The feedback for the questions had a general similar trend amongst them.

The feeling was that the prototype needs some adjustments before it can be used as the final piece. For example, an answer an individual gave was regarding the usability and accessibility and how it lacked this. This leads us to believe we can expand our research and look for ways to simplify interfaces. As a result, we can create a more inviting and easier to use application. Consequently, this will mean our subsequent users will find it more straight forward and uncomplicated than other means of service requesting e.g. Gumtree website. Therefore, based on these results and constructive criticism we can further advance our prototype.



Overall, most ultimately, we conducted this interview and feedback form in order to collect results which could either support our progress and direction or inform us where we are falling short. After analysing our results, we see how we must add (or take away) from this prototype 2 so that it becomes more accustomed to our target audience (service requester). However, we did receive some positive aspects of our, for example, the general concept of a general service request application was liked by all. This can be seen in the feedback at the end of the interviews. As a result, if we can incorporate the conditions the testers advised e.g. tutorials, easier to locate menu button and less congested pages, then we can certainly find users more willing to swap their existing service request methods to adopt ours.

Analysis for the Prototypes Interviews - Contractor

Total interviews conducted: 4

Number of interview questions: 8 (Contractor Questions)

Two prototypes

As part of our user testing, we asked the same two contractors whom of which we conducted the first interviews with to test our product. We wanted to find out whether the concept work and, if the designs that we created runs smoothly as we anticipated. We want to make the best possible design so therefore, we gave the contractor two designs and decide from there, what is the best in terms of user friendliness and aesthetics. We will take this feedback into consideration and make the final product based on the feedback to make the 'perfect user experience'. The way we conducted was different. We gave the contractor a list of what they rate each aspect of the app from user friendliness to whether the design needs improving. We first conducted the prototype 2 interview first, then after that, we conducted the prototype 1 version.

Prototype 2:

User Friendliness:

3/5 (Contractor 1)

- They were on the fence about the user friendliness. They believe that the buttons are big, so it is very good for accessibility but however, they don't like the fact that there is too much going on in the main menu.

5/5 (Contractor 2)

- They say it is very simple to use. They like the feature of finding customers through the app and, the messaging feature which allows them to talk to potential customers without meeting them face-to-face.
-

From this we can gather that the opinions are mixed. There is one contractor who thinks it's not as user friendly whilst the other thinks it's extremely simple to use. We need to take contractor 1's views on board and try to make it simple as possible. We will make the main menu just have the GPS as the main menu so that the user is already immersed into finding a job already. As soon as the user drags the menu up, they will see a list of buttons making it easy for the contractor as everything will look organised.

Easy to Navigate:

4/5 (Contractor 1)

- They said that the buttons are were clear and big enough to navigate through the app. This means that our app is capable for people who can't see clearly as compared to others. We want to make accessibility a key feature, so a voice-command feature would make the app easy to navigate.

5/5 (Contractor 2)

- They also said it was easy to navigate and it was clear for the user to know what pages lead to what pages on the app.

The navigation side is most important element in the user experience side. This is because we want to make the user have the best possible experience when going through different pages in the app. If it is not clear, then it makes it hard and users will be put off from using the app. We also want to make the app cater for all users, therefore, we are implementing voice features, so it will be suited to visually impaired users as well.

Enjoyable:

5/5 (Contractor 1)

- They said that it was enjoyable because it would be useful for themselves. They also said that it suits both parties when it comes to finding work and jobs. They also think the GPS map makes the application feel more hands on and the animations/transitions is very cool when you are switching pages.

5/5 (Contractor 2)

- This contractor said it will be a better alternative to using websites such as checkatrade.com. They also mention it will be a brilliant platform to find customers for themselves.

We are committed to provide an enjoyable user experience so that users can use it. The reviews we got here has fulfilled that as we want to make the app to feel interactive and feel more hands on. We want to make the user feel fully immersed into finding a customer by implementing the GPS map on the menu.

Attractiveness:

5/5 (Contractor 1)

- They like the colours that I have implemented in the different pages and think the app looks contemporary and would blend in the current market of apps.

5/5 (Contractor 2)

- They also said the same as contractor 1 by saying they like the different colours in different pages and that it just generally looks visually pleasing.

We don't want to make the app look like an eyesore. We want the app to look upmarket and blend in with the contemporary feel and the brand of our application. However, looking at the reviews on the user friendliness, people would prefer if we maintained having a user-friendly look whilst having the same attractiveness. We will be discussing this with our contractors in the coming weeks as they will be part of our development team, so we can make the best possible design. We think the app design should hold the principle of 'form follows function'.

Advisable:

5/5 (Contractor 1)

- They said that they will recommend this application to their friends and family because they would need their services in the future. They also mention that they will recommend this app to their fellow workers.

5/5 (Contractor 2)

- They also said that they will recommend this application to their friends and family. They believe that it will be a great platform for customers to find workers and workers getting jobs.

We want the whole concept to work. Therefore, we want to make it so that everyone passes down the message and recommend our app to others. This will give us confidence that if we develop this to be how we vision it, then I think we can see people sending positive messages about our app.

Needs Improving:

3/5 (Contractor 1)

- Their main concern is the fact that it's not as user friendly as they expected. They believe the main menu and other pages on the app feel too congested. They think that we should simplify this and whilst keeping the same theme. They also think it is not ready to be used as a final design.

1/5 (Contractor 2)

- A completely different perspective as they said it doesn't need improving at all.

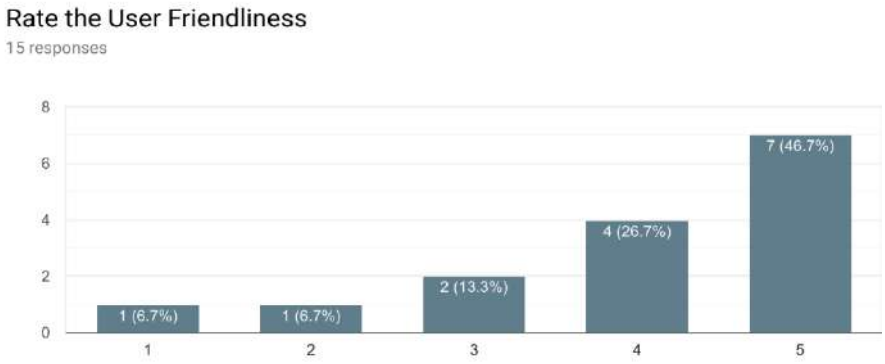
We have two different opinions once again. The contractor 1 would like to see the app redesigned so it's more user friendly whilst keeping the attractive element. We will take this feedback on board. It does correspond to what people have been saying in the service requester side so this is something we will closely working with the people we interviewed.

Concept:

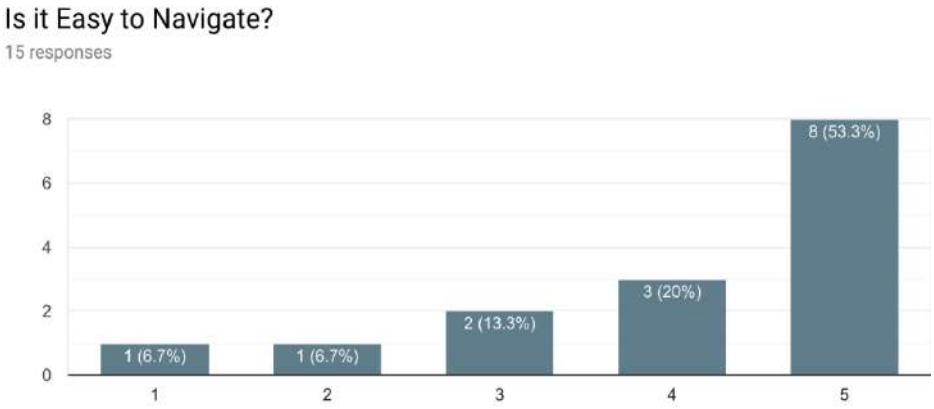
We asked two extra questions about the concept. We asked will the contractor use this app once it is released and whether concept works. Both contractor's responses were that they would use the app because it enables them to enhance their experience in the field that they're working in as they don't often find jobs as easily. It also allows them to find jobs without paying large fees which helps with trying to maintain a healthy profit for their business. When asked whether the concept works, they both responded with yes. These interviews have enabled us to get a bigger picture of our app. We now know a lot more which allows us to develop one app that considers all users.

Labora Prototype 1 – Questionnaire – Analysis

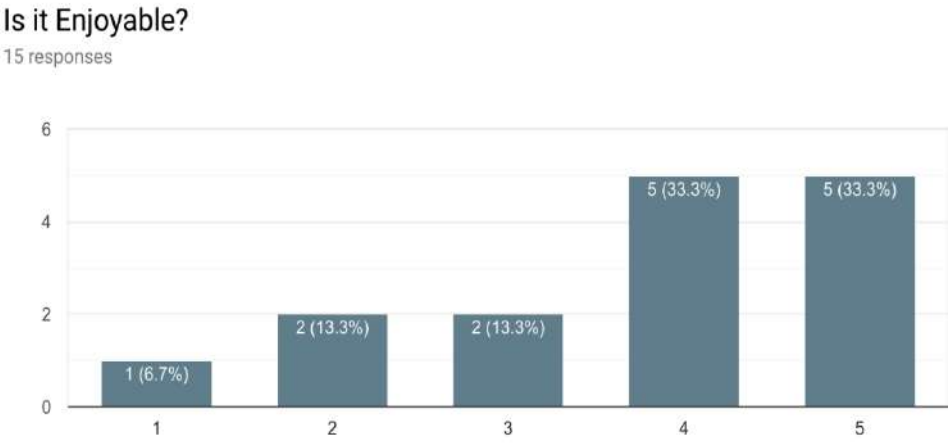
After creating prototype 1 we distributed a questionnaire and received feedbacks from 15 people, who responded and allowed us to understand where our work is good and where we can improve.



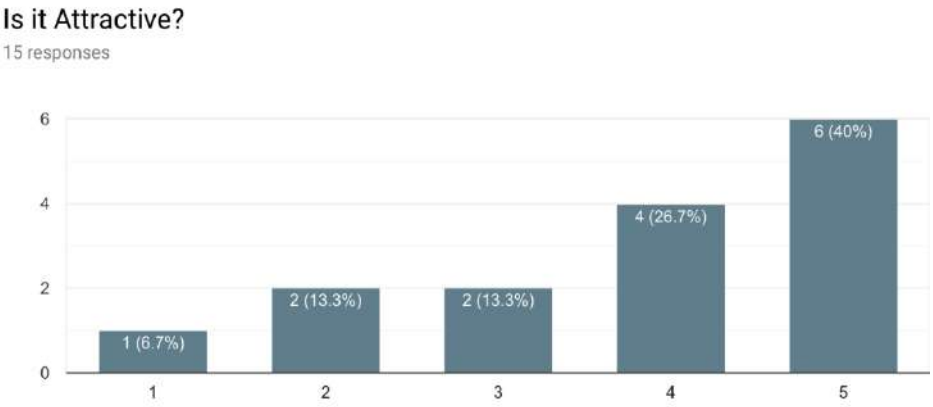
From the answers of our interviewees, the question on user friendliness shows that they found the interface easy to understand and easy to use, 7 out of 15 people voted with maximum points, but there are also 8 other people who did not give the maximum, so we can understand from the chart that some improvements can be made.



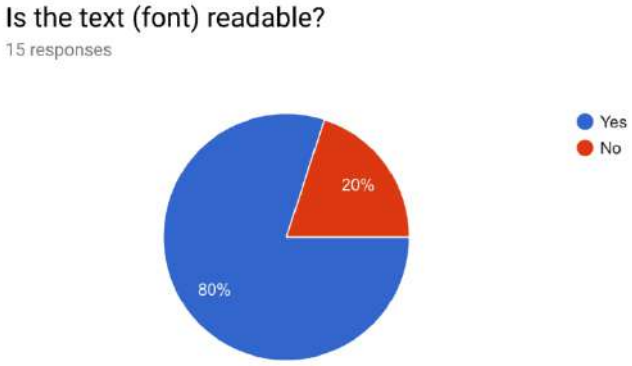
We asked if the prototype we showed was easy to navigate, and the answer was very positive.



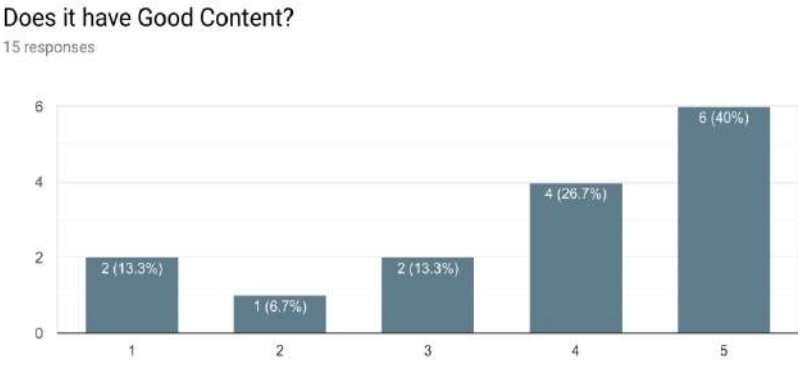
When asked if they found the application enjoyable, the results show that the majority agreed.



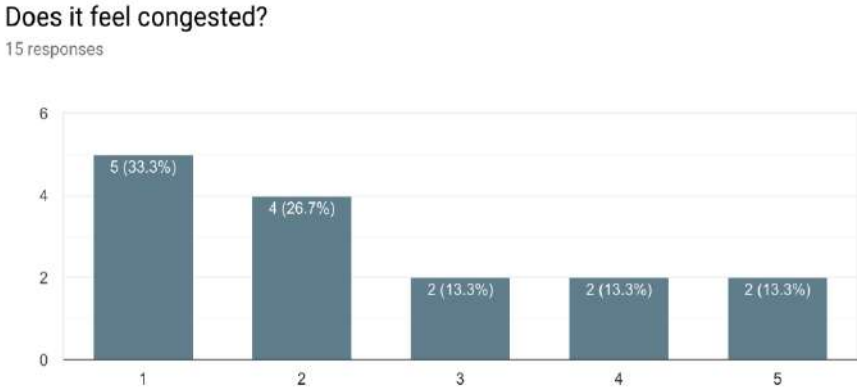
The platform is also quite attractive for users, this is supported by the interviewee's comments in the interview questions feedback.



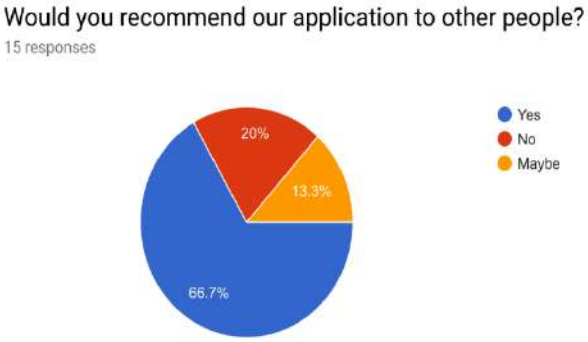
To the answer if it is readable, 80% have confirmed 'yes', so the fonts have been chosen well.



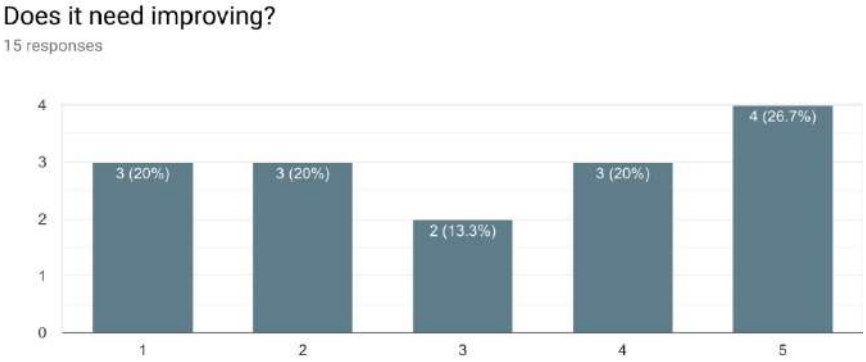
Out of 15, 5 individuals told us that there is good content, but the others' response was different, nearly 1/3 said there isn't enough content, and they advised us to add more of it. Most likely, our interviewees would have liked to see more services and more options on the interface of our service application prototype.



From this graph we understand that the people who tested our prototype version 1, they found it quite uncongested, nearly 60% founds the platform simple to use and easy to navigate. Therefore, we will try to keep this simple nature present in our final version.



Many would suggest our application to friends and family (66.7%), because this many people have told us they might need it. The other 20% said they said they will never suggest our app because there is already something similar or because they will never use it. The rest, at 13%, are in doubt.



Our feedback to the question of whether we can improve our prototype have got various comments. Some told us that the prototype version 1 is already good, whilst others instead advise us to work harder and improve the functionality and the design.

Any suggestions on what we can improve?

3 responses

Great design! Maybe add a filter for price range on the find job page.

improve the content

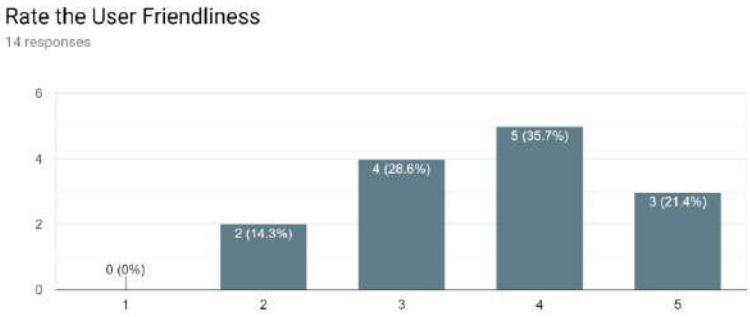
you must improve navigability

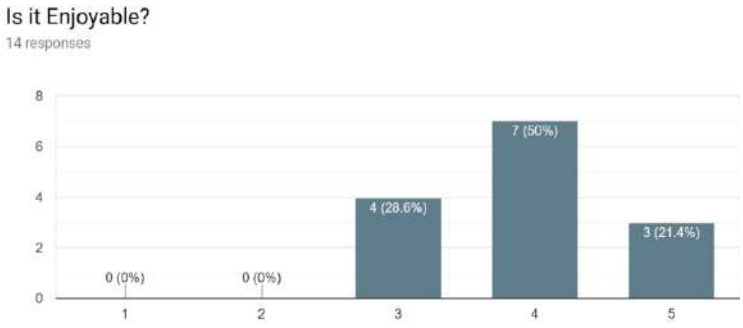
Our interviewees at the end of the questionnaire gave us some tips to improve our platform, they advised us to add a search filter, for example for the price, while others told us to increase the content, while others told us to improve navigability, especially between pages.

Labora Prototype 2 – Questionnaire - Analysis

We completed the prototypes. We then needed a way to test and get feedback for them. This was achieved in the form of face-to-face interviews and questionnaires, so that we can collect both quantitative and qualitative data. In this document, we will analyse the responses for the prototype 2 questionnaire.

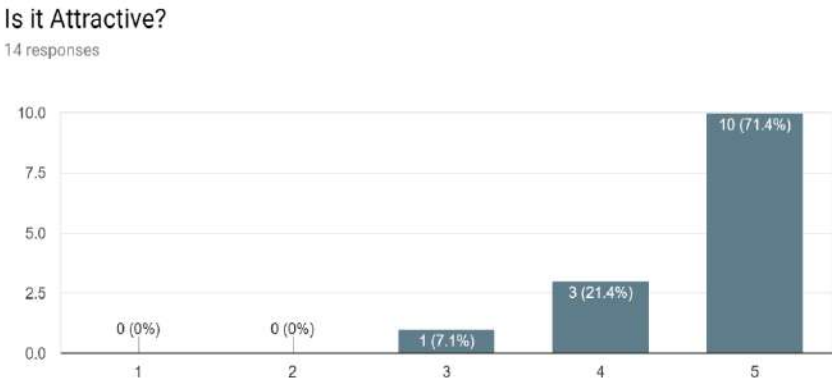
We started off by asking how user friendly and enjoyable the service requester application prototype is, also how easy it is to navigate. The graphs for these questions can be found below:





Overall, from these initial few questions we see that the app is generally quite usable. However, due to the mixed nature of the results, and how spread they are, we can infer that there are improvements to be made. For example, although majority of the responses (8) for user friendliness is rated either 4 or 5, we do find that 2 (14.3%) individuals out of the total responses, found our prototype 2 unfriendly. Similar case for navigation. Although, 7 people thought navigating the prototype was quite easy, another 7 either weren't sure or didn't find it easy. As a result, this leads us to believe that we must improve our navigation and overall user interface. Consequently, this may be vital in making sure the application is accessible. Therefore, we may ensure service requesters/contractors find our application easier and more effective to use than their current methods of service requesting/providing.

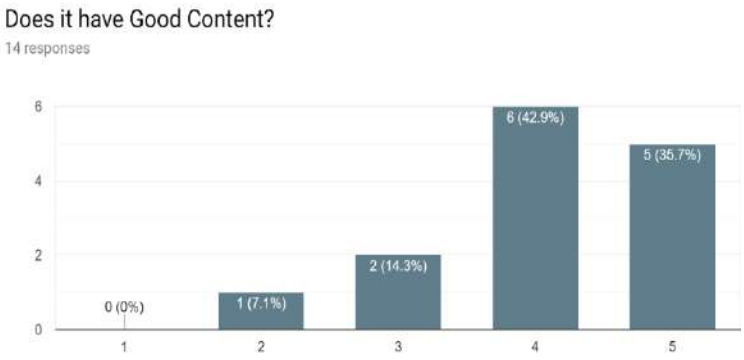
The next three questions are regarding the visual aspects of the prototype. The results for the questions are below:

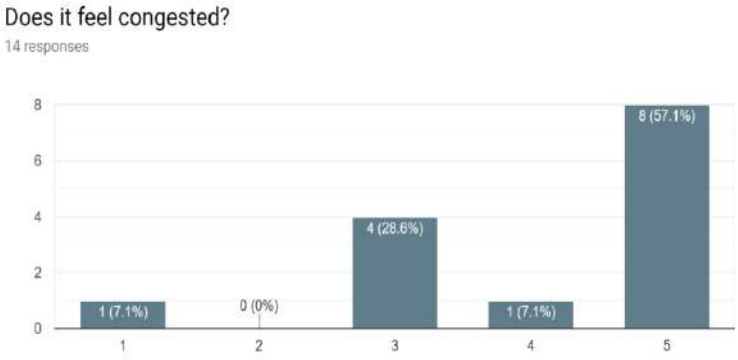


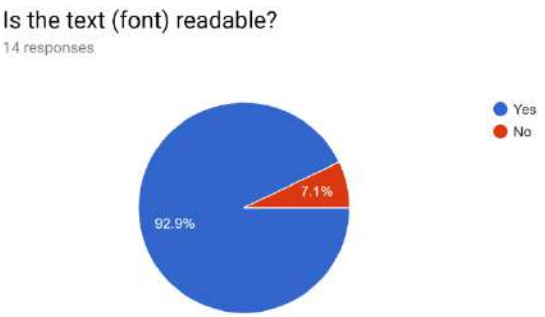
The first question asked whether the initial impressions of the prototype was attractive. 92.8% of responses agreed that this was the case. Also, no responses disagreed that it was attractive. Therefore, we agreed the display was appealing. The next question was about font, 92.9% agreed that the font was readable. We assumed this would be the case, as we did investigate which fonts were most readable and incorporated one. In addition, when developing our app, we aim to create a system where interface changes with user requirements (e.g. more font options

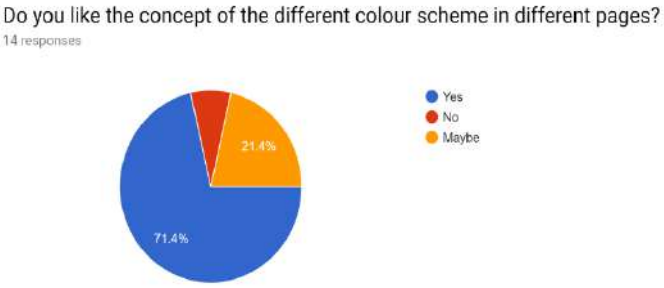
or text-size setter). The last question was in relation to the different colour scheme used on separated pages. 71.4% of the responses said they did like this concept. As a result, this portrays to us that our display features are correct. Therefore, we can spend more time on developing other features of the app e.g. accessibility, as we believe we have been successful in fulfilling this requirement.

The next two questions are about how good the content is and how congested it feels. The bar charts for the responses are below:



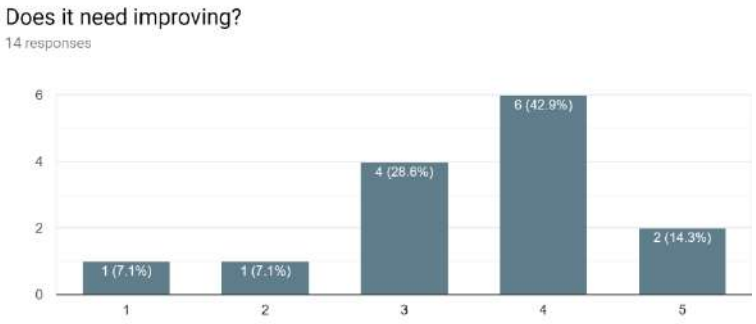






The results above do have quite a spread distribution. The question on good content had 9 people agreeing, however, 3 people were unsure and 2 didn't believe the content was 'good'. This leads us to believe our content is either very vague or quite confusing. As a result, we must ensure in our final product we include suitable, clear content directly related to either service providing or service requesting. The next question is whether the prototype pages feel congested. Most users agreed this was the case (57.1%). Although 4 users weren't sure, and 1 user disagreed, we knew we had to decrease the number of items on a single page at a time, especially the home page. We discussed and investigated how we could give users who wanted to access extra features, the chance to, and we concluded that we will make a pull up bar, like Instagram's. This means, users get the essential parts of the application presented in a clear manor, and any extra features they wish to view, can be seen if they swipe up. As a result, this reduces our issue of congestion, whilst maintaining our auxiliary features.

The final three questions were some of the most significant questions, as they let us know whether individuals would recommend our app and what we could do to advance it. Below are the results:



Any suggestions on what we can improve?

7 responses

Accessibility

Change user interface

Improve display for visually impaired. Also, I didn't know that was a side menu when you click the icon.

Less congestion

There are too many colours on the first page, feels like too much to take in at once but the design of it is very good

I think that app does need a bit of improving. There's a lot going on. I'd appreciate it if it was simpler.

Less congestion, because it looks packed. Tutorial.

As stated above these are the most important questions, as they let us know how successful this prototype has been. Also, they show us how we can improve and progress for the final application. The first question asked the individual whether they would recommend our app, 50% said yes. However, a large proportion were unsure (42.9%). This leads us to believe we must improve with the user's feedback if we wish to increase the number of recommendations, as our current prototype is not yet a suitable replacement for service requesters/contractors. The second question is regarding whether we need to improve. Out of 14, 8 responses indicated we need improvements and the following question gave them a chance to tell us how. We received various comments. The general output we took was that the interface was not as friendly and accessible as it could be. As a result, we need to work and focus on the user interface and implement helpful new features e.g. separate display for visually impaired users. Consequently, this will allow more individuals to utilise our application. Therefore, this may increase the viability of our product and make users of our application want to substitute their existing methods for receiving/providing services.

Overall, most ultimately, we believe that the conduction of this questionnaire for our prototype 2 was essential for us to gather quantitate data. We did also receive feedback for our application, which is a bonus. As mentioned before, we have now analysed this data and along with the analysis of our prototype interviews'. We will now be confidently able to produce one final prototype which shall incorporate all changes and modifications, including user recommendations. Our target is to create an efficient and useful application which facilitates all users in our target market and completing this testing phase may allow us to do so.

User Stories

Contractor (Detailed Scenario):

As an electrician (contractor), I want to find customers who require my service and find jobs, so I can make a living out of this. Jobs could include fixing their wirings or installing new electric power systems.

I will use the application as a platform to help me make money. I will firstly find potential customers via the GPS map the app provides who want my service. I will then find out more details about my potential customer and talk to them via the messaging system the application has. I will talk to them regularly and fit a schedule in which suits both parties. If the customer requirements and my requirements have been met, they will accept the job and I will meet with the customer as we planned. After the job has been executed, I will ask my customer to give me a review based upon how I have executed the job. I will also have to give a review based on how they were as a customer by giving them a rating and general feedback.

In turn, this will help me achieve a living out of this.

Service Requester (Detailed Scenario):

As a service requester, I want to find contractors to fulfil a service that I require. For example, I need to install a new sink, I would call a plumber to help me with that.

I will use the application to find workers that I require by tapping a button. I will pay them what will be owed for undertaking the service. First, I will go on the app and press the search button to find a service. Then, the search will give me a list of services and its speciality. I will click to proceed then I will have to confirm my personal information which includes my address, the description of the work they must carry out and the dates. I will click to confirm, and the details will be transferred to the contractor and from there they will have the opportunity to look at my details. Then, I would expect a message from one of the contractors and we can discuss further information regarding how they will execute the job and confirm our dates. I will also be given the opportunity to accept the job and from there, the contractor will come on the date that they promised. After the job has been executed, I will give them a review on how good their job is and upload any pictures on how well they did the job.

In turn, the result would mean that this app would help me find workers as fast as possible as compared to other websites.

As a/an	I want to…	So that….
Service Requester	Find a Plumber	They can help me fit my sink.
Service Requester	Recommend this app to my friends and family	They can help them whatever service they need.
Service Requester	Recommend this app to people in the trade	They can use the app to make money for themselves.

Service Requester	Find a Baby Sitter	They can manage my child whilst I am away.
Service Requester	Message contractors regarding information about the job	I can get a greater insight on the details e.g. what time the contractor has confirmed, what job they will execute.
Service Requester	Add a review for the contractor	Other customers can see how well/bad they executed the job.
Service Requester	Accept the contractors job confirmation	The job is confirmed and is given to the contractor. The request will be taken off the GPS map.
Service Requester	Change address	The contractor will know where to undertake the job.
Contractor	Find a Customer	I can start earning money and making a living off it.
Contractor	Recommending the app to friends	They can use the app and make a living out of it. Also, it will help people find workers if they require it.
Contractor	Use the GPS map to find customers	I will know where the customer is based and thus, find out my travelling options when I will be working there.
Contractor	Message the customer	To inform them when they are willing to start the job and whether they fit my time schedule.
Contractor	Add review on the customer	Other contractors can get an idea on the rating of the customer.

Systems Requirement Specification (SRS)

Project Glossary - Definition

We created a project glossary to give us the understanding what words and phrases meant. We wanted to give these meanings in the context of the project.

Term	Explanation
Service Requester	A user who uses the app to request for services they require.
Contractor	A user who works in a specific field of service. They will have access to the GPS map.
Request a Service	The service requester will post a request through their end and it will be posted on the GPS map which the contractor will pick up.
Accept the Job (Contractor)	The contractor will accept the job so that they can communicate with the service requester via the message system.
Confirm the Job (Service Requester)	The service requester will confirm the job so that the job is finalised and all information regarding the meeting time is confirmed.
Job	The contractor will undertake jobs at the service requester's house.
GPS map	The GPS map is only exclusive to the contractor where they can find customers on there once, they requested the job.
Cancel the request	The request from the service requester can be cancelled before their job gets accepted by the contractor.
Cancel the job	The job can be cancelled but only if both parties request to cancel the job.
Message System	A method of communicating between both stakeholders.
Labora's Database	The database will hold information about the service requesters actions and the contractor's actions. (E.g. request for service, confirming jobs and GPS map information).
Search	The service requester can search for services.
Category	The service requester will have a list of categories of services they can choose from.

Use Case

A use case listing was created for both users which helped us to identify all the activities that will take place when the users use the application.

Use Case Specifications

USE CASE ID	USE CASE NAME
UC1	LogInToSystem
UC2	SearchForService
UC3	ConfirmRequest
UC4	RequestSent
UC5	CancelRequest
UC6	FindCustomersGPS
UC7	ContractorHandlesRequest
UC8	AcceptContractor
UC9	Communication
UC10	JobConfirmed
UC11	EmailConfirmation
UC12	UpdatedToDatabase
UC13	JobExecuted
UC14	ReviewJob

USE CASE	LogInToSystem
ID	UC1
BRIEF DESCRIPTION	The users attempt to log into the application
PRIMARY ACTORS	Service Requester, Contractor
SECONDARY ACTORS	None
MAIN FLOWS	1) The user enters username and password 2) The system checks the details if the information is correct 3) The system validates if it's the contractor or the service requester log in details. 4) The system matches and allows access to the application. 5) The user will be introduced to the main menu
ALTERNATIVE FLOWS	InvalidLogin

USE CASE	LogInToSystem:InvalidLogin
ID	UC1.1
BRIEF DESCRIPTION	The system doesn't allow the user access as login details are not correct

PRIMARY ACTORS	System
SECONDARY ACTORS	Service Requester, Contractor
MAIN FLOWS	1) The user enters username and password 2) The system checks the details if the information is correct 3) The system matches, and it is the incorrect information 4) The user will be allowed to retry 5) The system will only allow 5 tries before it locks the user out
ALTERNATIVE FLOWS	None

USE CASE	SearchForService
ID	UC2
BRIEF DESCRIPTION	The service requester searches for a service
PRIMARY ACTORS	Service Requester
SECONDARY ACTORS	None
MAIN FLOWS	1) The user will go on the search bar and find potential services 2) The user will be directed to a list of services 3) They have to choose one service and its speciality the contractor undertakes.
ALTERNATIVE FLOWS	None

USE CASE	ConfirmRequest
ID	UC3
BRIEF DESCRIPTION	The service requester confirms the request
PRIMARY ACTORS	Service Requester
SECONDARY ACTORS	None
MAIN FLOWS	1) The system will confirm to the user if the information is correct. 2) The service requester will go through the information and check. 3) The service requester will press the confirm button.
ALTERNATIVE FLOWS	None

USE CASE	RequestSent
ID	UC4
BRIEF DESCRIPTION	The system sends the request
PRIMARY ACTORS	System
SECONDARY ACTORS	Service Requester
MAIN FLOWS	1) The service requester presses the confirm button 2) The system will send the request to the contractor and will appear on their GPS map
ALTERNATIVE FLOWS	None

USE CASE	CancelRequest
ID	UC5
BRIEF DESCRIPTION	The user cancels the request
PRIMARY ACTORS	Service Requester, System
SECONDARY ACTORS	Contractor
MAIN FLOWS	1) The service requester cancels the request before the contractor has accepted the job 2) The system acknowledges this and cancels the request. It will be taken out of the GPS map
ALTERNATIVE FLOWS	None

USE CASE	FindCustomersGPS
ID	UC6
BRIEF DESCRIPTION	The contractor finds the customer on the GPS
PRIMARY ACTORS	Contractor
SECONDARY ACTORS	Service Requester, System

MAIN FLOWS	1) The contractor will use the GPS map on their main menu to find potential customers 2) They will press on the waypoint button where it will give details on the customer's information 3) Contractor will handle request from there
ALTERNATIVE FLOWS	None

USE CASE	ContractorHandlesRequest
ID	UC7
BRIEF DESCRIPTION	The contractor will handle the request by accepting or rejected
PRIMARY ACTORS	Service Requester, Contractor
SECONDARY ACTORS	System
MAIN FLOWS	1) The service requester will be expected for a response from one of the contractors 2) The contractor will accept the job 3) The system will acknowledge this and allow access for the contractor to send messages. 4) The system notifies the service requester.
ALTERNATIVE FLOWS	CReject

USE CASE	ContractorHandlesRequest:CReject
ID	UC7.1
BRIEF DESCRIPTION	The contractor will reject the service requester's request.
PRIMARY ACTORS	Service Requester, Contractor
SECONDARY ACTORS	System
MAIN FLOWS	1) The service requester will be expected for a response from one of the contractors 2) The contractor will reject the job 3) The system will acknowledge this and remove the pinpoint out of their own GPS map
ALTERNATIVE FLOWS	ContractorReject

USE CASE	AcceptContractor
ID	UC8
BRIEF DESCRIPTION	The service requester accepts the contractor's request to communicate with them
PRIMARY ACTORS	Service Requester
SECONDARY ACTORS	Contractor, System
MAIN FLOWS	1) The service requester will get a notification that a contractor has accepted their request and would like to message them 2) The service requester will accept 3) The system acknowledges this and allows the contractor to talk to the user
ALTERNATIVE FLOWS	rejectContractor

USE CASE	AcceptContractor:rejectContractor
ID	UC8.1
BRIEF DESCRIPTION	The service requester rejects the contractor's request to communicate with them
PRIMARY ACTORS	Service Requester
SECONDARY ACTORS	Contractor, System
MAIN FLOWS	1) The service requester will get a notification that a contractor has accepted their request and would like to message them 2) The service requester will reject 3) The system acknowledges this and won't allow the contractor to message the user. 4) The system will notify the contractor.
ALTERNATIVE FLOWS	None

USE CASE	Communication
ID	UC9
BRIEF DESCRIPTION	The service requester and the contractor will communicate through the messaging service the app has
PRIMARY ACTORS	Service Requester, Contractor

SECONDARY ACTORS	System
MAIN FLOWS	1) If the service requester has accepted, they can communicate on the app to discuss the pricing information and when to start
ALTERNATIVE FLOWS	None

USE CASE	JobConfirm
ID	UC10
BRIEF DESCRIPTION	The service requester confirms the job
PRIMARY ACTORS	Service Requester
SECONDARY ACTORS	System, Contractor
MAIN FLOWS	1) The service requester will proceed to confirm whether they want the contractor to undertake the job or not. 2) They confirm, and the contractor will have the job secured. 3) The system will update this on the database that a job has been confirmed
ALTERNATIVE FLOWS	JobReject

USE CASE	JobConfirm:JobReject
ID	UC10.1
BRIEF DESCRIPTION	The service requester rejects the job
PRIMARY ACTORS	Service Requester
SECONDARY ACTORS	System, Contractor
MAIN FLOWS	1) The service requester will proceed to confirm whether they want the contractor to undertake the job or not. 2) They reject, and the contractor will be notified via the system that the service requester did not want to go further 3) Contractor has to find another customer
ALTERNATIVE FLOWS	None

USE CASE	EmailConfirmation
ID	UC11
BRIEF DESCRIPTION	The service requester and the contractor will get an email confirmation
PRIMARY ACTORS	System
SECONDARY ACTORS	Service Requester, Contractor
MAIN FLOWS	1) The system will confirm the details by sending an email to both service requester and contractor.
ALTERNATIVE FLOWS	None

USE CASE	UpdatedToDatabase
ID	UC12
BRIEF DESCRIPTION	The information that has been confirmed will be updated to the database
PRIMARY ACTORS	System
SECONDARY ACTORS	Service Requester, Contractor
MAIN FLOWS	1) The system will update the database regularly once both the service requester and contractor have confirmed the job
ALTERNATIVE FLOWS	None

USE CASE	JobExecuted
ID	UC13
BRIEF DESCRIPTION	The contractor will execute the job at the service requester's house
PRIMARY ACTORS	Service Requester, Contractor
SECONDARY ACTORS	None
MAIN FLOWS	1) The contractor will start undertaking the job at the service requester's house 2) The contractor will be paid depending on how the service requester made the payment plan (e.g. weekly or daily) 3) The service requester's service is fulfilled
ALTERNATIVE FLOWS	None

USE CASE	ReviewJob
ID	UC14
BRIEF DESCRIPTION	Both contractor and service requester will review each other
PRIMARY ACTORS	Service Requester, Contractor
SECONDARY ACTORS	System
MAIN FLOWS	1) Once the job has been executed, the service requester has to review the contractor's job. They will give a rating and comment on how they executed the job. They will also upload images. 2) The contractor can give a review to tell other contractors how they're as customers.
ALTERNATIVE FLOWS	None

Unit Testing – Service Request

All test files can be found on the repository of group 16 service requester

LoginActivity.java Test Cases:

Test	*Test File Name*	*Status (Failed/Passed)*
Check if 'LoginActivity' launches without crashing	*LoginActivityMethodsTest*	✔
Check if 'LoginActivity' launches and is able to go to 'ProfileActivity' after signing in	*LoginActivityToProfileActivityTest*	✔
Check if 'LoginActivity' launches and is able to go to 'RegisterActivity'	*LoginActivityToRegisterActivityTest*	✔
Check if email entering box is appearing and working	*LoginActivityMethodsTest*	✔
Check if password entering box is appearing and working	*LoginAciivityMethodsTest*	✔
Check if on create function is working (on click listeners for each button is working)	*LoginActivityMethodsTest*	✔
Test if 'userLogin' function is working (e.g. does it check for empty fields)	*LoginActivityMethodsTest*	✔
Check if 'buttonSignIn' works	*LoginActivityMethodsTest*	✔
Check if dialog box shows up when logging in	*LoginAciivityMethodsTest*	✔
Check after logging in, if ProfileActivity opens up	*LoginActivityRegisterActivtityTest*	✔
Check if 'buttonSignUp' opens RegisterActivity	*LoginActivityRegisterActivtityTest*	✔

RegisterActivity.java Test Cases:

Test	*Test File Name*	*Status (Failed/Passed)*
Check if email entering field is appearing and working	*RegisterActivityMethodsTest*	✔
Check if password entering field is appearing and working	*RegisterActivityMethodsTest*	✔
Check if on create function is working (on click listeners for each button is working)	*RegisterActivityMethodsTest*	✔
Test if 'registerUser' function is working (e.g.	*RegisterActivityMethodsTest*	✔

does it check for empty fields)		
Check if 'buttonRegister' works	*RegisterActivityMethodsTest*	✔
Check if dialog box shows up when signing up	*RegisterActivityMethodsTest*	✔
Check if after signing up, RegisterActivity2 opens up	*RegisterActivityToRegisterActivity2Test*	✔
Check if 'buttonRegister' opens RegisterActivity2	*RegisterActivityToRegisterActivity2Test*	✔

RegisterActivity2.java Test Cases:

Test	*Test File Name*	*Status (Failed/Passed)*
Check if name entering field is appearing and working	*RegisterActivity2MethodsTest*	✔
Check if phone number entering field is appearing and working	*RegisterActivity2MethodsTest*	✔
Check if on create function is working (on click listeners for each button is working)	*RegisterActivity2MethodsTest*	✔
Test if 'registerUser' function is working (e.g. does it check for empty fields and turn editTexts into strings)	*RegisterActivity2MethodsTest*	✔
Check if 'onClick' works and calls 'registerUser'	*RegisterActivity2MethodsTest*	✔
Check if 'buttonRegister' is clicked and registers user	*RegisterActivity2MethodsTest*	✔
Check if after registering, ProfileActivity page opens up	*RegisterActivity2ToProfileActivityTest*	✔

ProfileActivity.java Test Cases:

Test	*Test File Name*	*Status (Failed/Passed)*
Check if activity opens without crashing	*ProfileActivityTest*	✔
Check if user's username/email address is appearing on screen, when activity is currently open	*ProfileActivityTest*	✔
Check if on create function is working (on click listeners for each button is working)	*ProfileActivityToActivitiesTest*	✔
Test if menu buttons are displayed (most importantly, 'search')	*ProfileActivityToActivitiesTest*	✔
Check if 'onClick' works and calls the specified functions	*ProfileActivityToActivitiesTest*	✔

Check if 'search' button works ('openSearch' function) and opens up the 'Services' class file	*ProfileActivityToActivitiesTest*	✔
Check if 'logout' button works and takes the user back to 'LoginActivity' class	*ProfileActivityToActivitiesTest*	✔

Services.java Test Cases:

Test	*Test File Name*	*Status (Failed/Passed)*
Check if activity opens without crashing	*ServicesTest*	✔
Check if fields and a button appear on screen when activity is run	*ServicesTest*	✔
Check if on create function is working (on click listeners for each button is working and that fields are being turned into strings)	*ServicesTest*	✔
Test if "Service" spinner will give an error if empty or if hint selected	*ServicesMethodsTest*	✔

We resolved this test case by adding a conditional to check input is empty or the "Choose a service" option
`if(TextUtils.isEmpty(requesterService) ||`
`requesterService.equals("Choose a service"))`
`{`
`Toast.makeText(Services.this, "Please choose a service",`
`Toast.LENGTH_LONG).show();`		

Test if "Post Code" field will give an error if empty	*ServicesMethodsTest*	✔

We resolved this test case by adding a conditional to check input is empty
`else if(TextUtils.isEmpty(requesterPostCode))`
`{`
`Toast.makeText(Services.this, "Please fill in a post code",`
`Toast.LENGTH_LONG).show();`
`}`		

Test if "Job Description" field will give an error if empty	*ServicesMethodsTest*	✔

We resolved this test case by adding a conditional to check input is empty

```
else if(TextUtils.isEmpty(requesterJob))
{
    Toast.makeText(Services.this, "Please fill in a job description",
Toast.LENGTH_LONG).show();
}
```

Test if "Keywords" field will give an error if empty	*ServicesMethodsTest*	✔

We resolved this test case by adding a conditional to check input is empty

```
else if(TextUtils.isEmpty(requesterKeyWord))
{
    Toast.makeText(Services.this, "Please fill in keywords",
Toast.LENGTH_LONG).show();
}
```

Check if 'onItemsSelected' function works and the spinner stores the correct service	*ServicesMethodsTest*	✔
Check if 'onNothingSelected' function works when nothing is selected (e.g. with a toast)	*ServicesMethodsTest*	✔
Check if 'request' button works ad opens the 'Summary.class' activity	*ServicesMethodsTest*	✔

Summary.java Test Cases:

Test	*Test File Name*	*Status (Failed/Passed)*
Check if 'Summary' activity loads without crashing	*SummaryTest*	✔
Check if a completion message is displayed	*SummaryTest*	✔

Unit testing – Contractor (MVP) – Test Cases

All test files can be found on the repository of group 16 contractor

LoginActivity.java Test Cases:

Test	Test File Name (.java)	Status (Failed/Passed)
Check if 'LoginActivity' launches without crashing	LoginActivityTest	✔
Check if 'LoginActivity' launches and is able to go to 'ProfileActivity' after signing in	LoginActivityTest	✔
Check if 'LoginActivity' launches and is able to go to 'RegisterActivity'	LoginActivityTest	✔
Check if email entering box is appearing and working	LoginActivityTest	✔
Check if password entering box is appearing and working	LoginActivityTest	✔
Check if on create function is working (on click listeners for each button is working)	LoginActivityTest	✔
Test if 'userLogin' function is working (e.g. does it check for empty fields)	LoginActivityTest	✔
Check if 'buttonSignIn' works	LoginActivityTest	✔
Check if dialog box shows up when logging in	LoginActivityTest	✔
Check after logging in, if Main Menu opens up	LoginActivityTest	✔
Check if 'buttonSignUp' opens RegisterActivity	LoginActivityTest	✔

RegisterActivity.java Test Cases:

Test	Test File Name (.java)	Status (Failed/Passed)
Check if email entering field is appearing and working	RegisterActivityTest	✔

Check if password entering field is appearing and working	RegisterActivityTest	✔
Check if on create function is working (on click listeners for each button is working)	RegisterActivityTest	✔
Test if 'registerUser' function is working (e.g. does it check for empty fields)	RegisterActivityTest	✔
Check if 'buttonRegister' works	RegisterActivityTest	✔
Check if dialog box shows up when signing up	RegisterActivityTest	✔
Check if after signing up, RegisterActivity2 opens up	RegisterActivityTest	✔
Check if 'buttonRegister' opens RegisterActivity2	RegisterActivityTest	✔

RegisterActivity2.java Test Cases:

Test	Test File Name (.java)	Status (Failed/Passed)
Test 1 Check if name entering field is appearing and working	RegisterActivity2Test	X
Test 1 Check if phone number entering field is appearing and working	RegisterActivity2Test	X
Test 1 Check if occupation entering field is appearing and working	RegisterActivity2Test	X
Test 1 Check if address entering field is appearing and working	RegisterActivity2Test	X
Check if on create function is working (on click listeners for each button is working)	RegisterActivity2Test	✔

Test 1 Test if 'registerUser' function is working (e.g. does it check for empty fields and turn editTexts into strings)	RegisterActivity2Test	X
Check if 'onClick' works and calls 'registerUser'	RegisterActivity2Test	✔
Check if 'buttonRegister' is clicked and registers user	RegisterActivity2Test	✔
Check if after registering, Main Menu page opens up	RegisterActivity2Test	✔
Test 2 Check if name entering field is appearing and working	RegisterActivity2Test	✔
Test 2 Check if phone number entering field is appearing and working	RegisterActivity2Test	✔
Test 2 Check if occupation entering field is appearing and working	RegisterActivity2Test	X
Test 2 Check if address entering field is appearing and working	RegisterActivity2Test	✔
We resolved this by moving the contents within the activity_register2.xml upwards as the test wouldn't scroll downwards and thus wouldn't detect address field. This was just an easy fix – however, in the future we will fix this problem ensuring users can scroll.		
Test 2 Test if 'registerUser' function is working (e.g. does it check for empty fields and turn editTexts into strings)	RegisterActivity2Test	✔
We resolved this test case by adding code to the RegisterActivity2 which tells the user they have to enter a string in order to proceed with registration For example:		

```
if(TextUtils.isEmpty(phone)){
    //phone number is empty
    Toast.makeText(this, "Please enter a valid phone number",
Toast.LENGTH_SHORT).show();
    return;

}
```

Test 3 Check if occupation entering field is appearing and working	RegisterActivity2Test	X

MenuActivity.java Test Cases:

Test	Test File Name (.java)	Status (Failed/Passed)
Check if on create function is working (on click listeners for each button is working)		✔
Test if menu buttons are displayed (most importantly, 'search')		✔
Check if 'onClick' works and calls the specified functions		✔
Check if 'go online' button works ('go online' function) and opens up the 'map' class file		✔
Check if 'logout' button works and takes the user back to 'LoginActivity' class		✔

MapActivity.java Test Cases:

Test	Test File Name (.java)	Status (Failed/Passed)
Check if fields and a button appear on screen when activity is run	MapActivityTest	X
Check if on create function is working (on click listeners for each button is working and that fields are being turned into strings)	MapActivityTest	X
Test if "Service" field will give an error if empty	MapActivityTest	X

Test if “Job Description” field will give an error if empty	MapActivityTest	X
Test if “Keywords” field will give an error if empty	MapActivityTest	X
Check if ‘onItemsSelected’ function works and the spinner stores the correct service	MapActivityTest	X
Check if ‘onNothingSelected’ function works when nothing is selected (e.g. with a toast)	MapActivityTest	X
Check if ‘request’ button works ad opens the ‘Summary.class’ activity	MapActivityTest	X

Testing Environment

Dependent Variables (Affected by other values):

- User testing the device

Independent Variables (Constants which do not depend on other values):

- Mobile Phone (Samsung Galaxy S9)
- Environment (Room)

Setting up a satisfactory testing environment is important, as it could impact the users decision making and thought process. Creating a good environment leads to the user making voluntary decisions and unprovoked decisions. As a result, this leads to unbiased decisions. Consequently, the data we collect in terms of feedback (qualitative) and ticked-boxes (quantitative), will be more representative of the overall target audience (in this case people who want to receive and provide services). Therefore, we can adjust our application specifically, which means the final product is closer to achieving maximum accessibility and usability for users.

Step1: Firstly, we set up the testing, so it is carried out in a quiet room, void of any distractions (visually or audible). This was an important step, as before mentioned, results can become skewed.

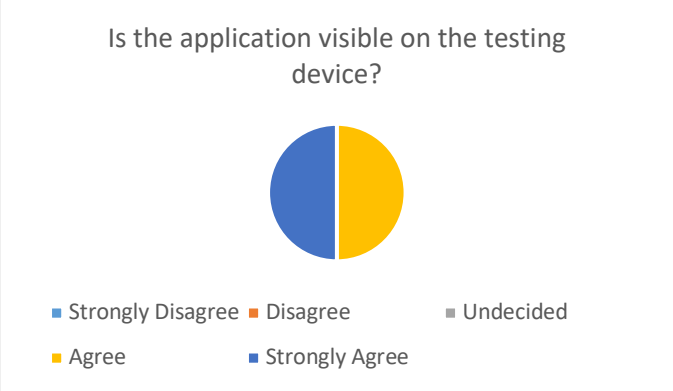
Step 2: Next, because our labora application is a mobile app, we used android studio to download the APK (Android Package Kit) to a phone. This then enabled us to display the application as it would be in real-life. We also ensured the mobile device being used (Samsung Galaxy S9) was constant throughout. This is so that all independent variables throughout testing were kept the same.

Step 3: After, we also set up screen recording on the phone (includes audio). This enabled us to keep track and store users movements. Also, we could analyse and possibly find trends made by all users. In addition, social ques made by users would be seen and noted by us (testers).

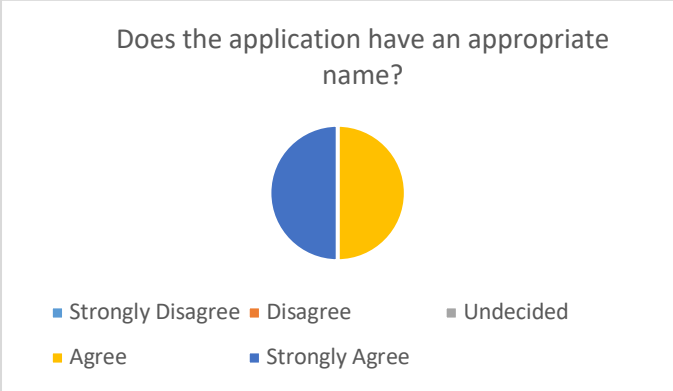
Step 4: Identifying our target audience. We kept in contact with our original contractors throughout, which meant that we could actually test our application with real contractors. We do have to find a new contractor, as black box testing requires users to be unknown to the product. Furthermore, we can find service requester, by contacting our original testers, and also finding new individuals.

Step 5: Carry out the tests using the user testing sheet. Allow users to interact with the application and decide whether it meets each individual criterion. Provocation or advising the user will affect results, so this will not be allowed.

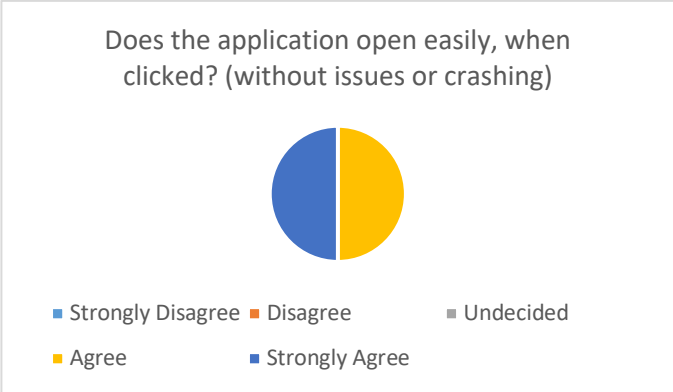
Contractor – User Testing White Box – Analysis



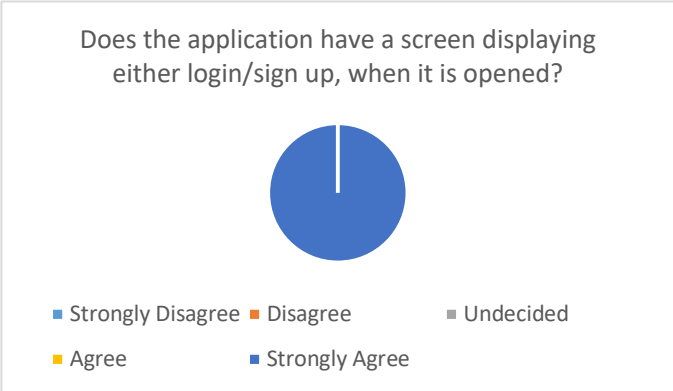
The diagram displays the results for whether the application is visible on the device it is being tested on. All the testers have agreed that it is. As a result, this means we do not have to work on this part of the application (or we have less to improve on).



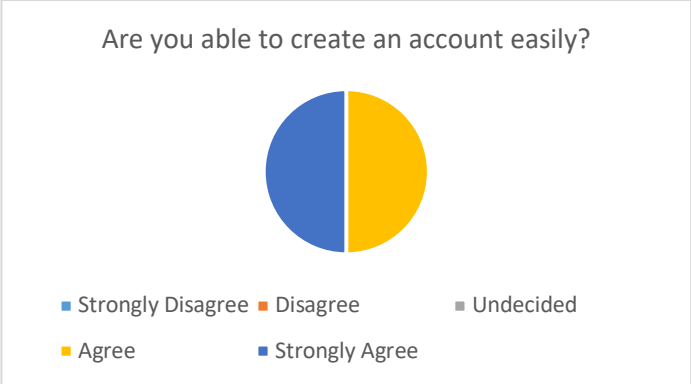
This diagram displays the results for whether the application has an appropriate name. According to the results, all the testers strongly agree that the name is appropriate. As a result, this means we do not have to focus on this part of the application (or we have less to improve on).



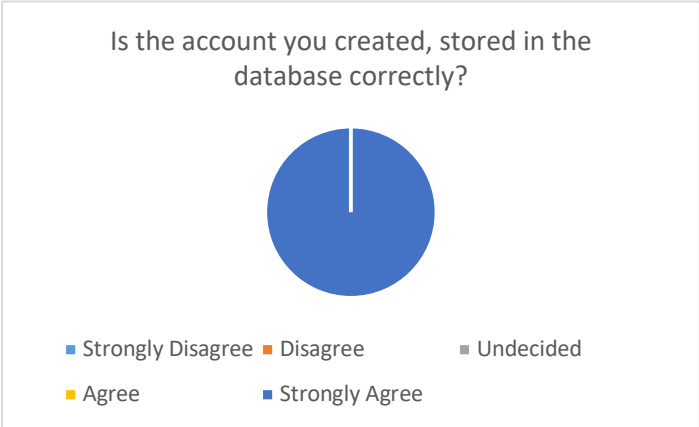
The diagram display the results for whether the application opens with ease, once clicked on. From the results we see an overall agreement that it does open safely, with no crashes. As a result, we do not have to focus our time on this part of the application (or we have less to improve on).



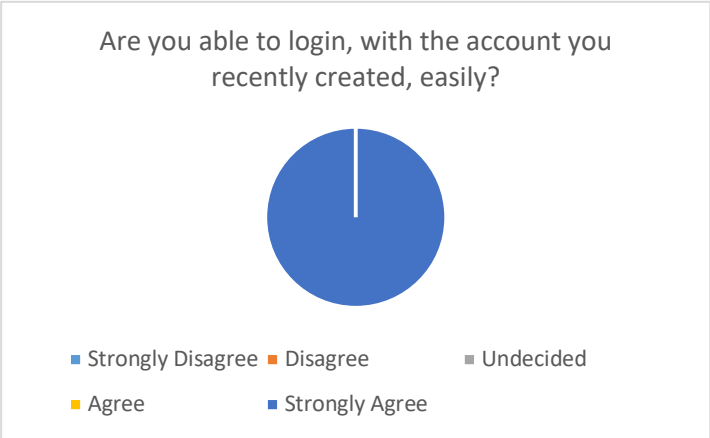
The diagram displays the results for whether the login/sign up page is shown on start up. The results portray that the start-up screen does display login/sign up. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



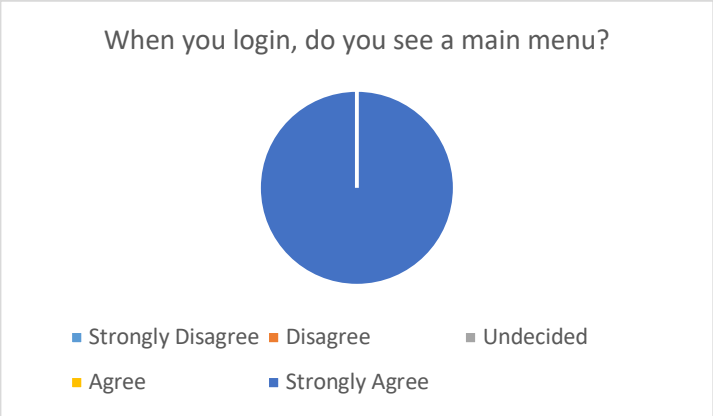
The diagram displays the results for whether an account can be created easily. The results portray that users can make an account fairly easily. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



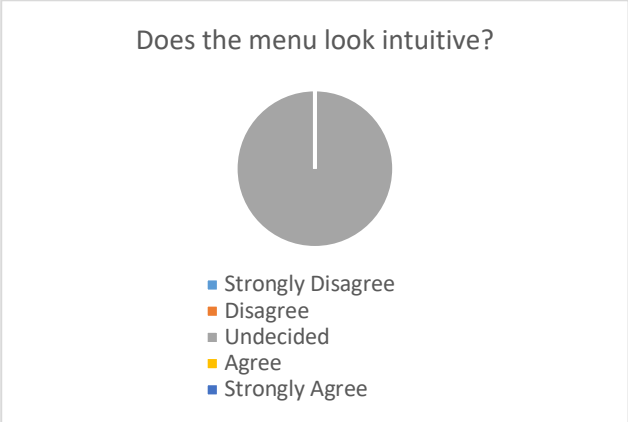
The diagram displays the results for whether the created account is stored in the database correctly. The results portray that the new account is stored in the right destination. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



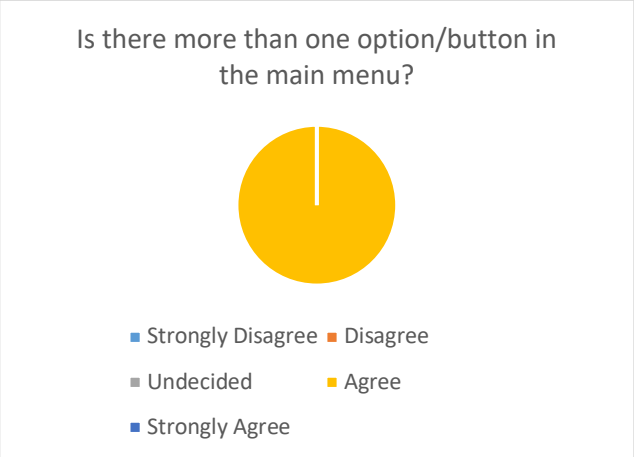
The diagram displays the results for whether the user can sign in using the recently created account. The results portray that the users are able to login easily with new login credentials. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



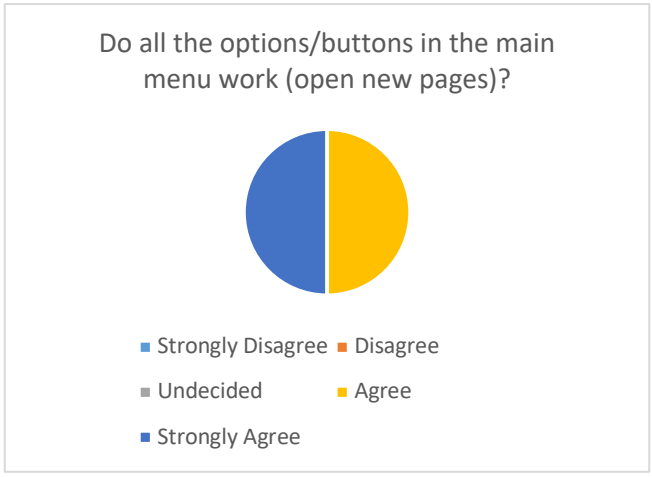
The diagram displays the results for whether after login, you can see a main menu. The results portray that users do see after login, a main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



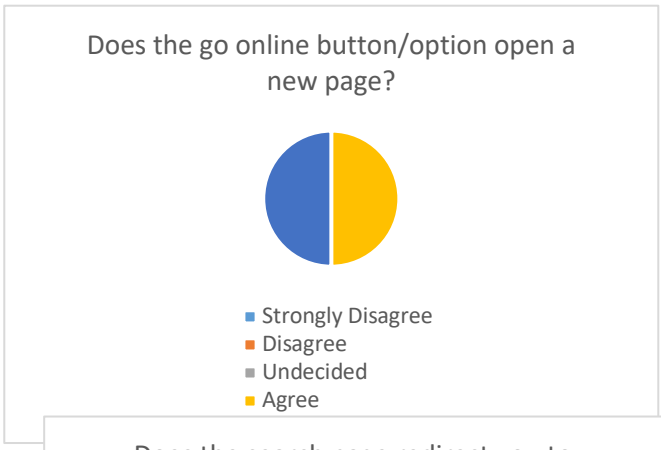
The diagram displays the results for whether the main menu looks intuitive. The results are varied between agreement and undecided. As a result, we need to ensure that we allocate time on this part of the application, so that in the next user tests, we have more users agreeing that the main menu is intuitive.



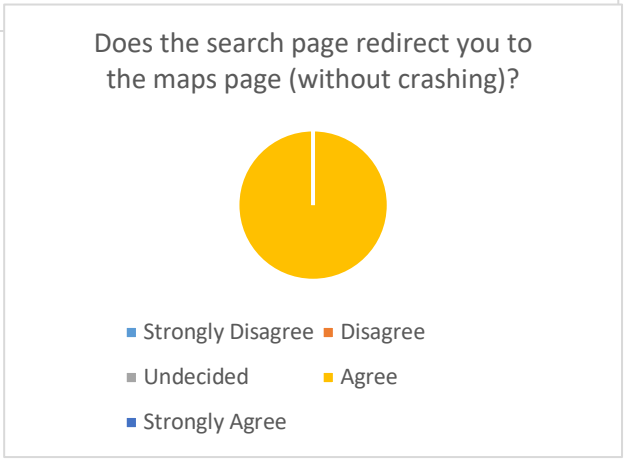
The diagram displays the results for whether there are multiple buttons in the main menu. The results portray that users do see more than one button in the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



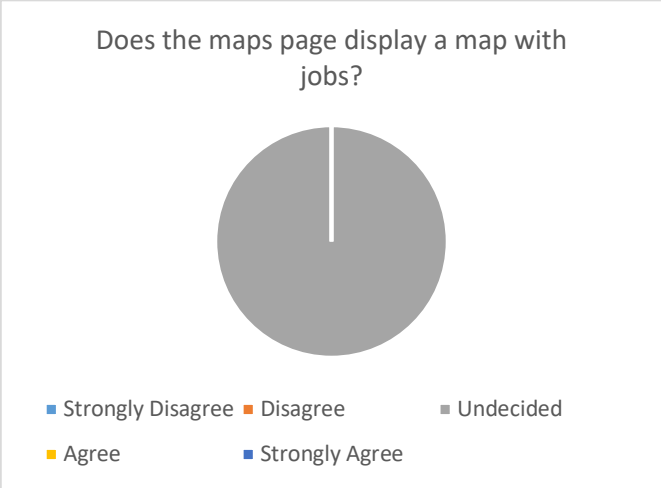
The diagram displays the results for whether the option buttons in the main menu function and open new pages. The results portray that the option buttons do open new pages. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



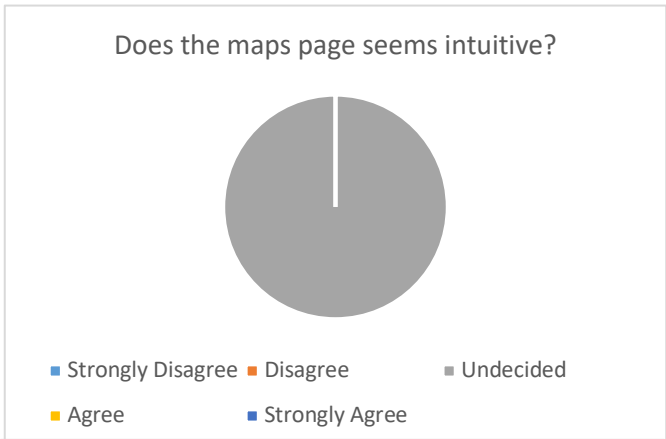
The diagram displays the results for whether the ‘go online’ button functions and opens the map. The results portray that the button does work as purposed. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



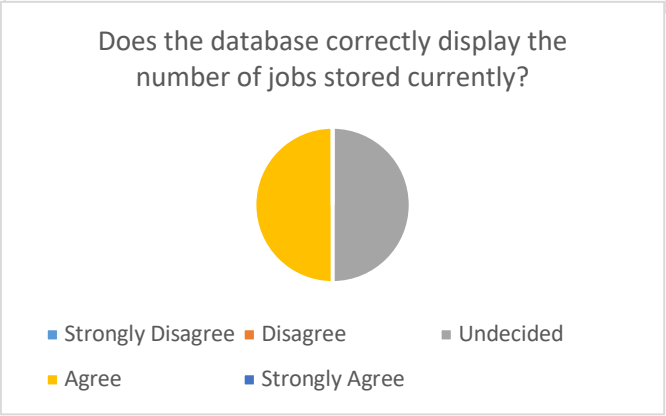
The diagram displays the results for whether the search (go online) button opens maps and doesn’t crash. The results portray that the page opens successfully without fail. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether the map page displays the jobs. The results portray that users found the map page either doesn't show potential clients or doesn't show many clients. As a result, we must focus more time on this part of the application as it is a major section of the application.

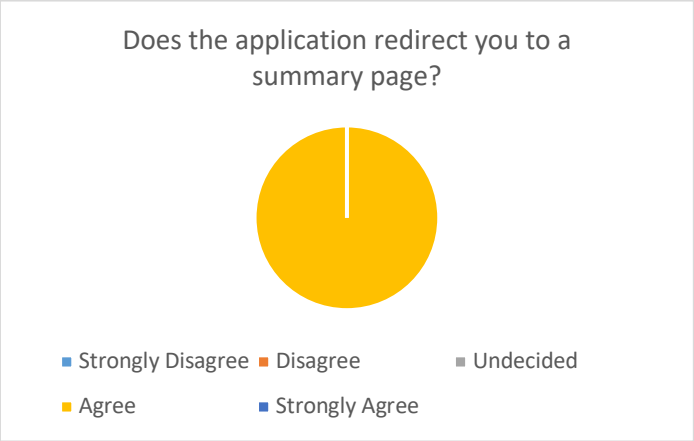


The diagram displays the results for whether the maps page seems intuitive. The results portray that the map page is not as intuitive as could be. As a result, we do have to focus time on this part of the application, and perhaps add hints to improve the intuitiveness.

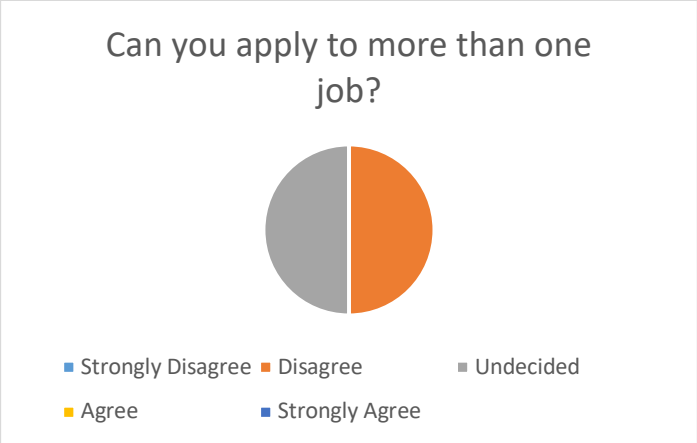


The diagram displays the results for whether the database correctly stores the number of jobs. The results portray that the jobs are either not stored in the right way or are partially stored correctly. As a result, we must focus more time on this part of the application, as storing jobs of is significant.

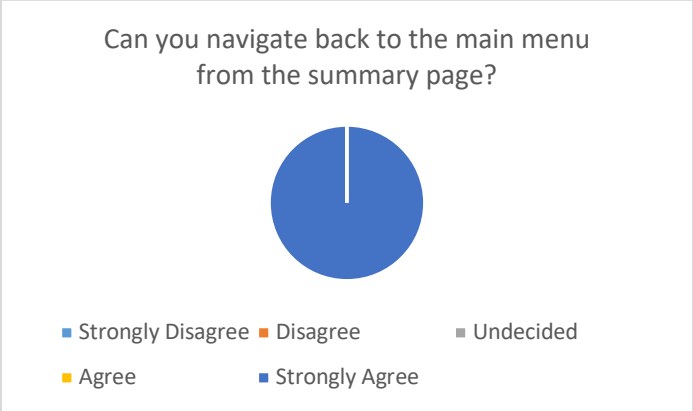
The diagram displays the results for whether you can click the job on the map and apply for it. The results portray that you can. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



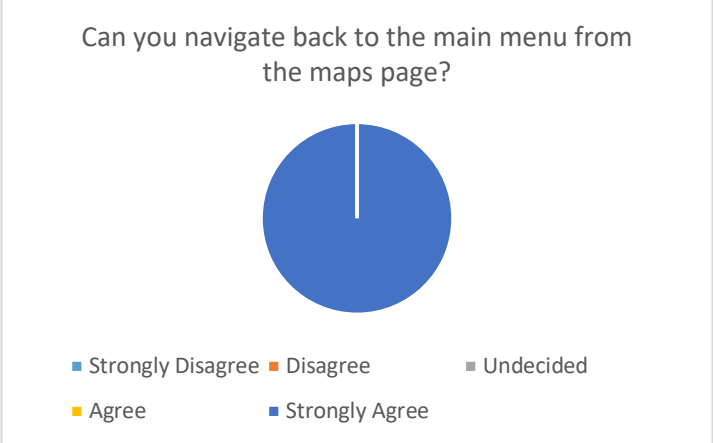
The diagram displays the results for whether the application does redirect to the summary page. The results portray that the app does redirect to the summary page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



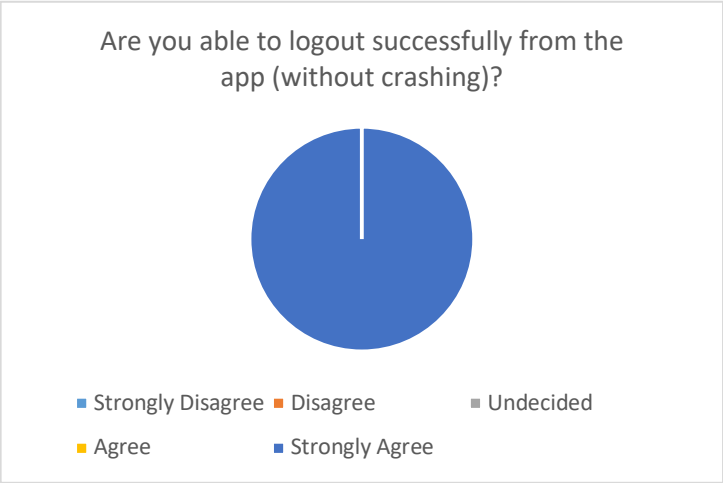
The diagram displays the results for whether you can apply to more than one job. The results portray that you cannot apply for more than one job, or it is difficult to do so. As a result, we need to focus time on this part of the application (or we have less to improve on). We can fix this by placing more jobs on the map.



The diagram displays the results for whether you can navigate back to main menu from the summary page. The results portray that you can navigate back to the summary page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether you can navigate back to main menu from the maps page. The results portray that you can navigate back. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

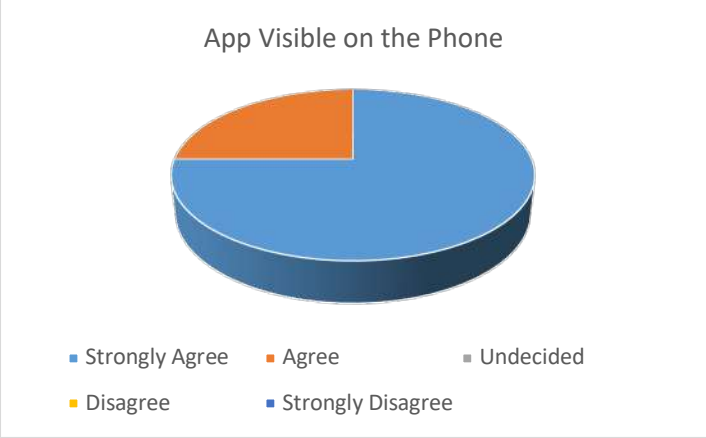


The diagram displays the results for whether you can logout from the application successfully. The results portray that you can . As a result, we do not have to focus time on this part of the application (or we have less to improve on).

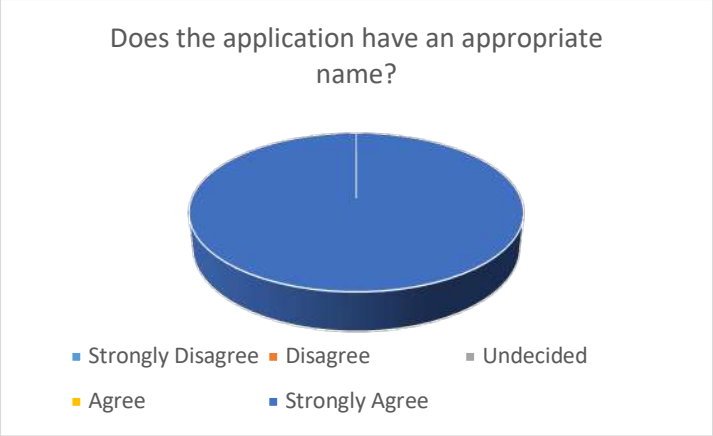
The diagram displays the results for whether you can sign out the app. The results portray that you can log back in again. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

Overall, most ultimately, we believe the user tests have been successful in terms of finding out which parts of the application function as expected and which parts need improvement. This is in order to create an ending application which suits users' needs and is accessible. We can do this by using the feedback from the contractors. The overall consensus was that we need to improve design, which could in turn improve our intuitiveness. Also, the application has reached the MVP (minimum viable product) and needs more advanced features to take it to the next level.

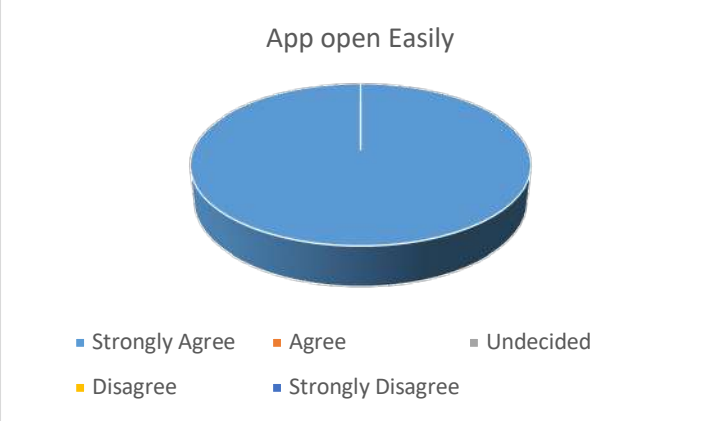
Service Requester – User Testing White Box – Analysis



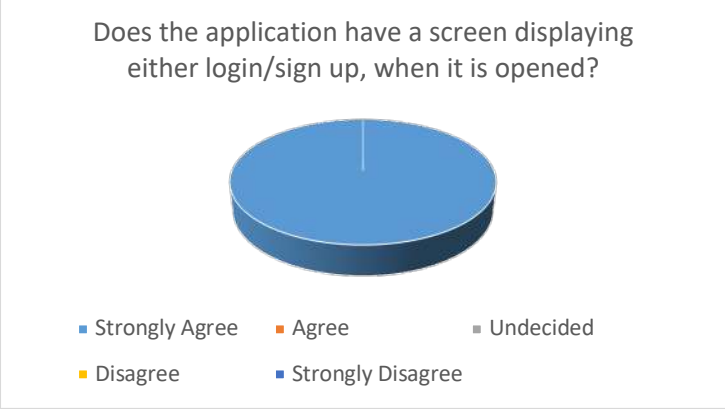
The diagram displays the results for whether the application is visible on the device it is being tested on. All the testers have agreed that it is. As a result, this means we do not have to work on this part of the application (or we have less to improve on).



This diagram displays the results for whether the application has an appropriate name. According to the results, all the testers strongly agree that the name is appropriate. As a result, this means we do not have to focus on this part of the application (or we have less to improve on).

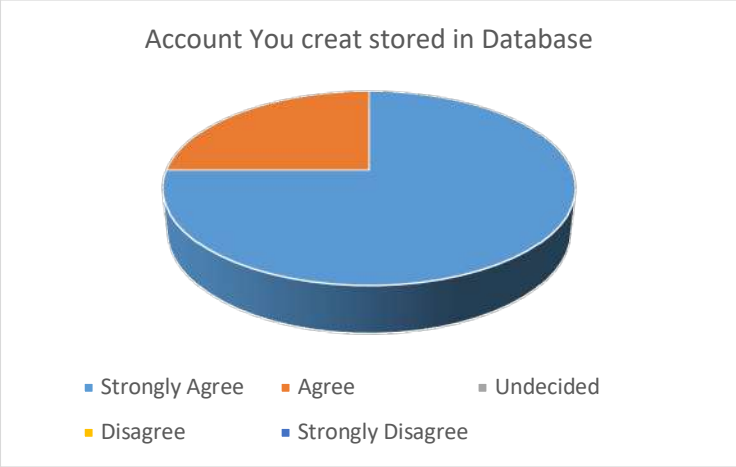


The diagram display the results for whether the application opens with ease, once clicked on. From the results we see an overall agreement that it does open safely, with no crashes. As a result, we do not have to focus our time on this part of the application (or we have less to improve on).

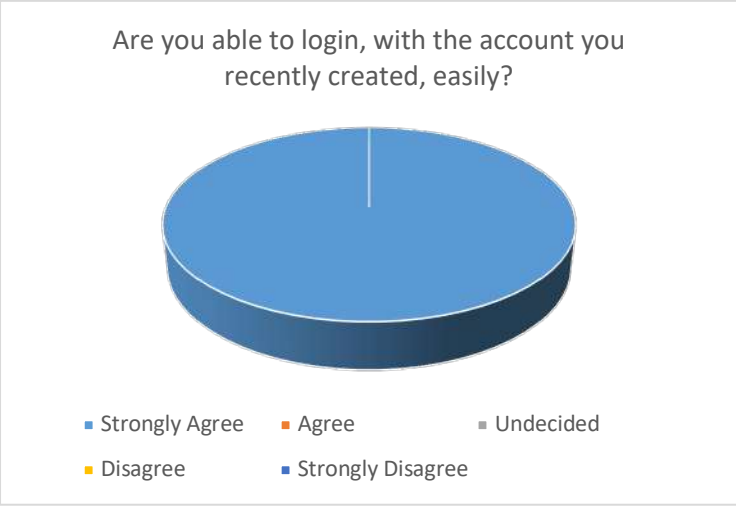


The diagram displays the results for whether the login/sign up page is shown on start up. The results portray that the start-up screen does display login/sign up. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

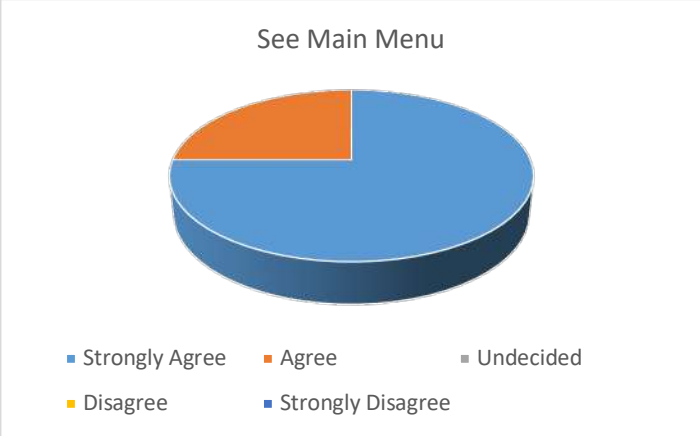
The diagram displays the results for whether an account can be created easily. The results portray that users can make an account fairly easily. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



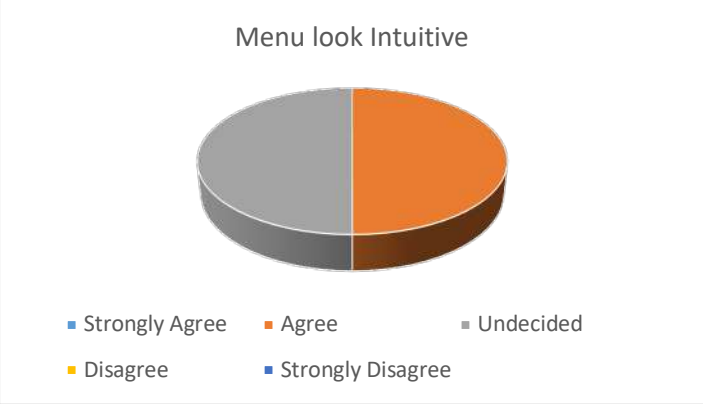
The diagram displays the results for whether the created account is stored in the database correctly. The results portray that the new account is stored in the right destination. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



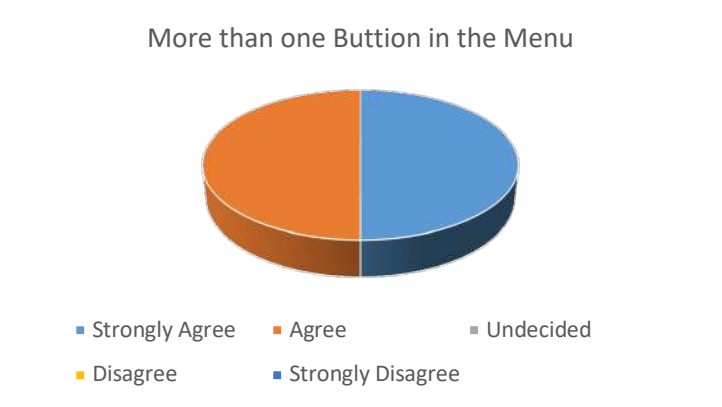
The diagram displays the results for whether the user can sign in using the recently created account. The results portray that the users are able to login easily with new login credentials. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



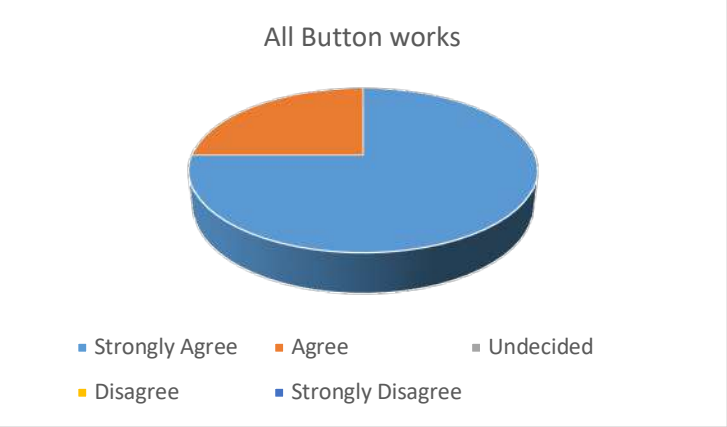
The diagram displays the results for whether after login, you can see a main menu. The results portray that users do see after login, a main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



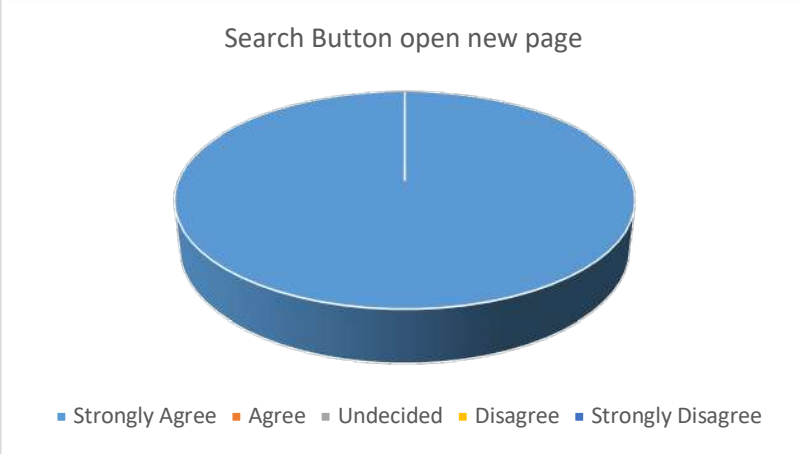
The diagram displays the results for whether the main menu looks intuitive. The results are varied between agreement and undecided. As a result, we need to ensure that we allocate time on this part of the application, so that in the next user tests, we have more users agreeing that the main menu is intuitive.



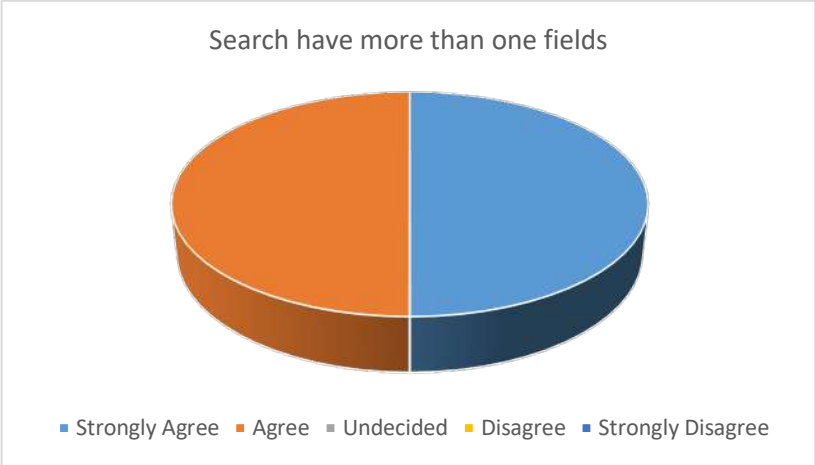
The diagram displays the results for whether the main menu looks intuitive. The results are varied between agreement and undecided. As a result, we need to ensure that we allocate time on this part of the application, so that in the next user tests, we have more users agreeing that the main menu is intuitive.



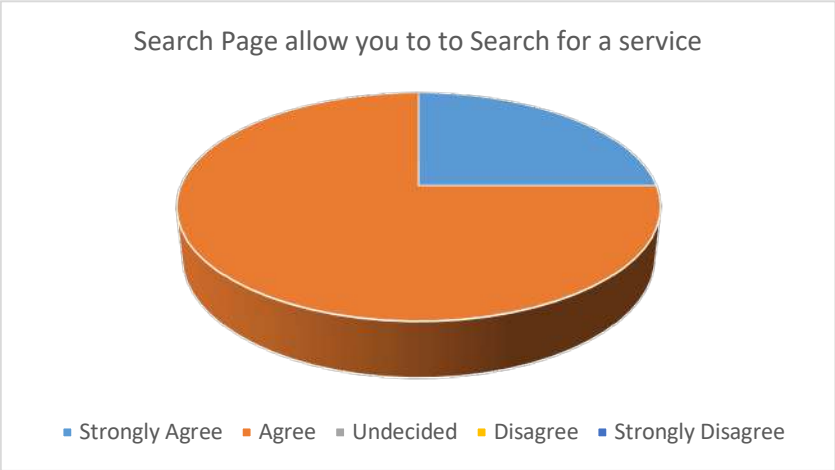
The diagram displays the results for whether the option buttons in the main menu function and open new pages. The results portray that the option buttons do open new pages. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



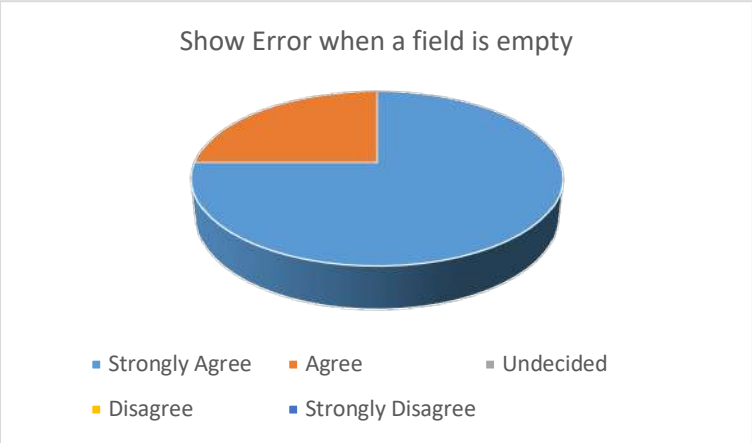
The diagram displays the results for whether the search button functions and opens the services list and details filling-in form. The results portray that the button does work as purposed. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



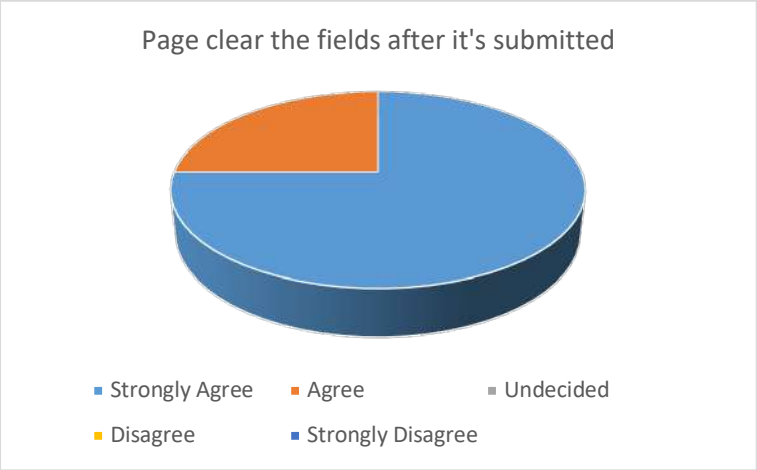
The diagram displays the results for whether the search (services page) has more than one field. The results portray that there are more than one field on this page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



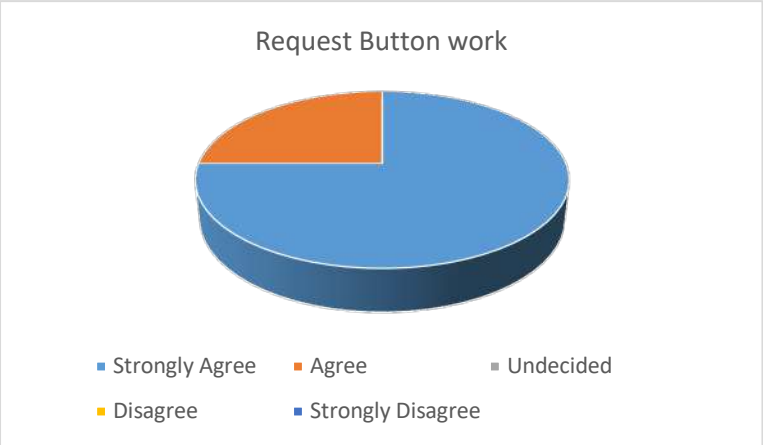
The diagram displays the results for whether the search (services page) allows you to search for a service. The results portray users are able to search (choose) a service. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



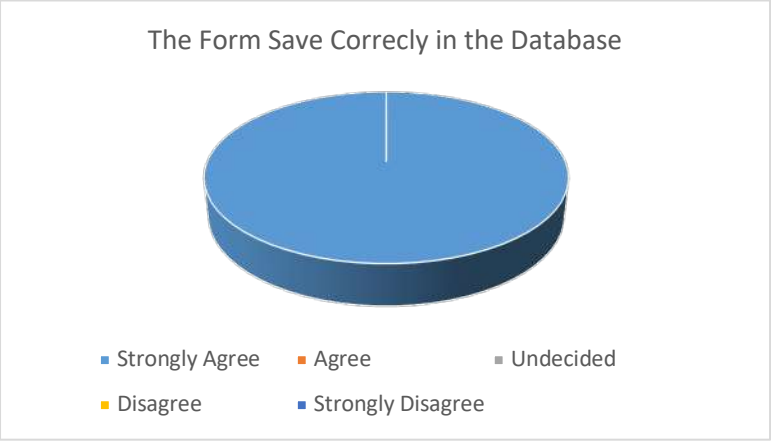
The diagram displays the results for whether the fields on the search (services page) produce an error message when left empty. The results portray that an error message is shown. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



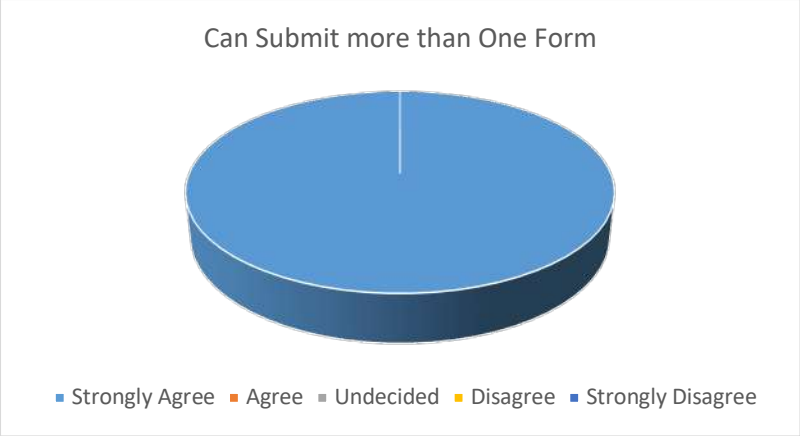
The diagram displays the results for whether the search (services page) clears the fields after being submitted. The results portray that fields are emptied after the request button is clicked and fields are filled. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



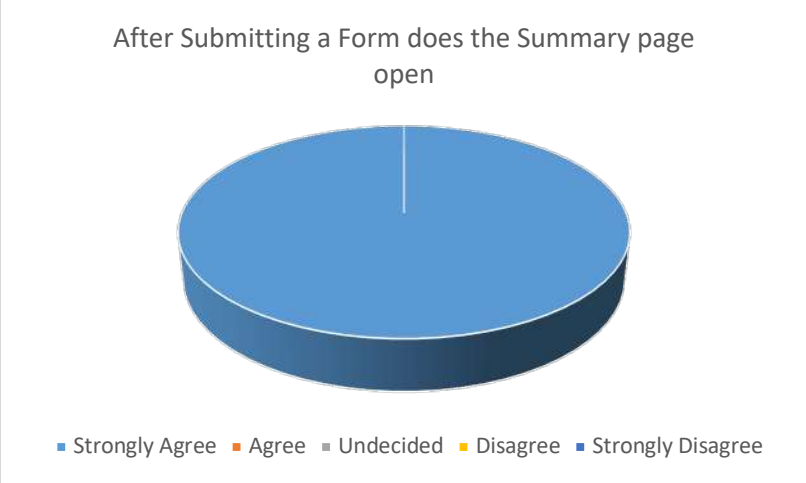
The diagram displays the results for whether the request button works. The results portray that once clicked the request button functions, and the summary page is displayed. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



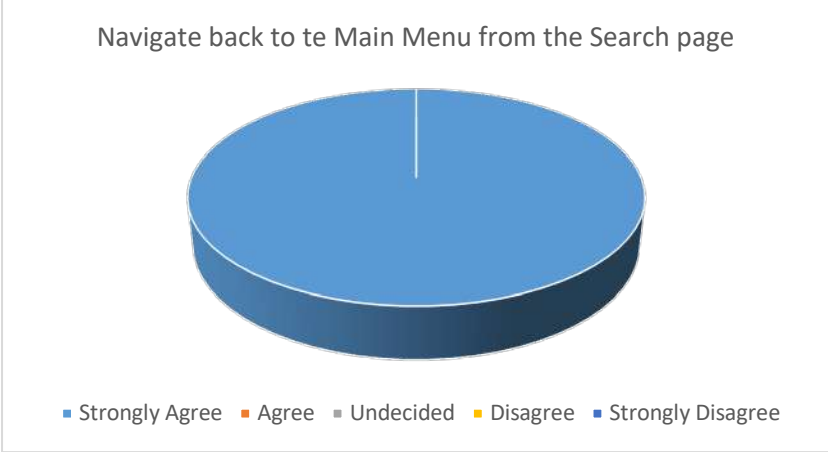
The diagram displays the results for whether the form is correctly stored in the database. The results are show users believe they are stored in the right destination. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



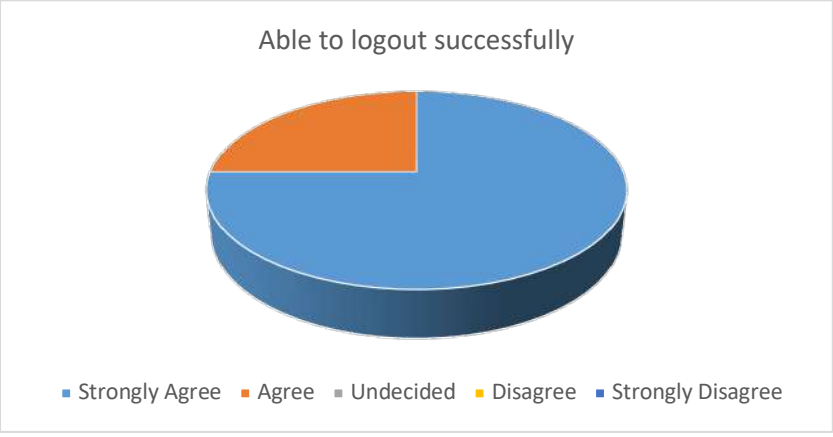
The diagram displays the results for whether the application allows more than one submission of the form. The results portray that the app does allow multiple request submissions. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



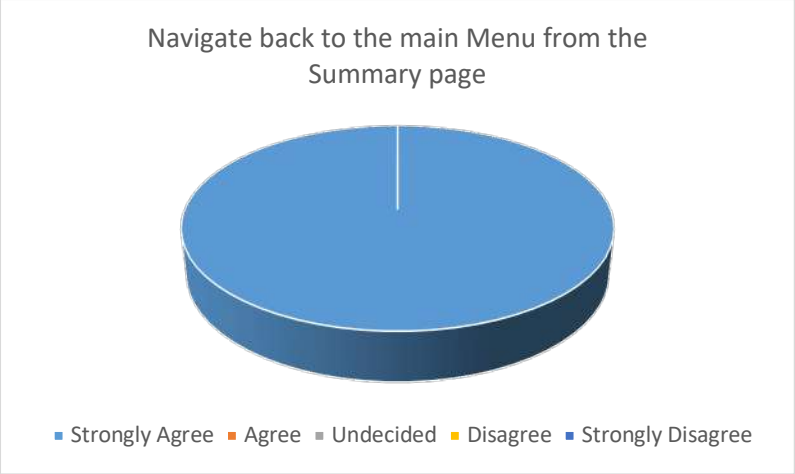
The diagram displays the results for whether the application does redirect to the summary page. The results portray that the app does redirect to the summary page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



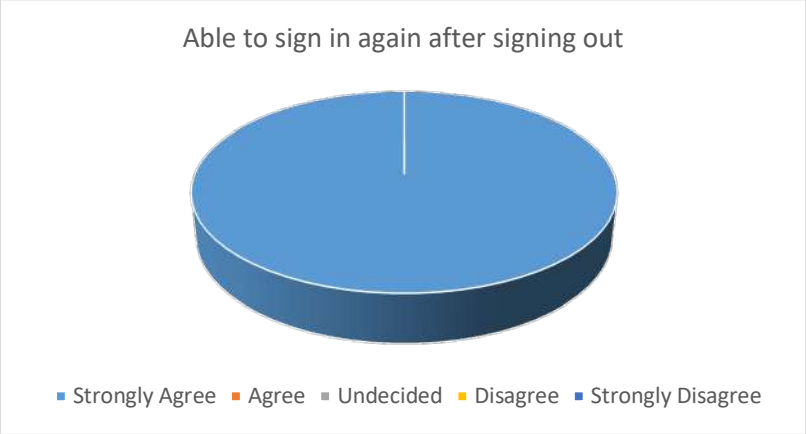
The diagram displays the results for whether you can navigate back to main menu from the search (services) page. The results portray that you can navigate back to the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether you can logout from the application successfully. The results portray that you can . As a result, we do not have to focus time on this part of the application (or we have less to improve on).



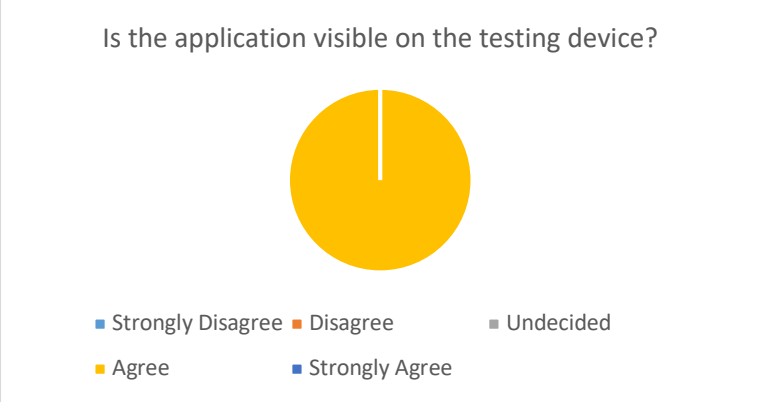
The diagram displays the results for whether you can navigate back to the main menu from the summary page. The results portray that you can go back to the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



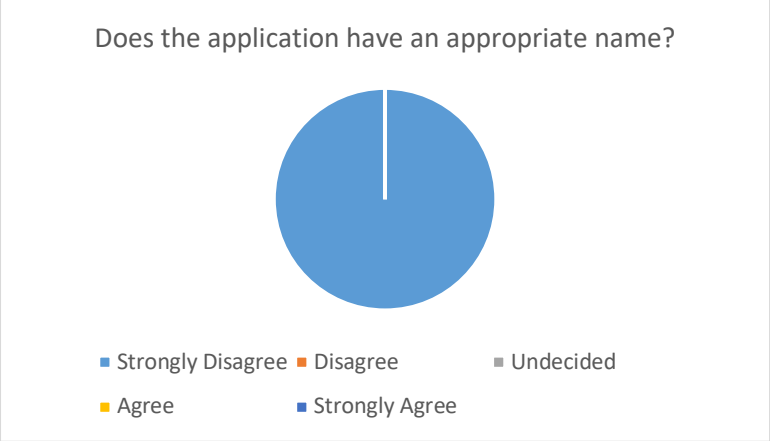
The diagram displays the results for whether you can sign out the app and log back in. The results portray that you can log back in again. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

Overall, most ultimately, we believe the user tests have been successful in terms of finding out which parts of the application function as expected and which parts need improvement. This is in order to create an ending application which suits users' needs and is accessible. We can do this by using the feedback from the contractors. The overall consensus was that we need to improve design, which could in turn improve our intuitiveness. Also, the application has reached the MVP (minimum viable product) and needs more advanced features to take it to the next level. Furthermore, we will also need to go over some of the more basic functionalities of the app to ensure they also pass the minimum criterion.

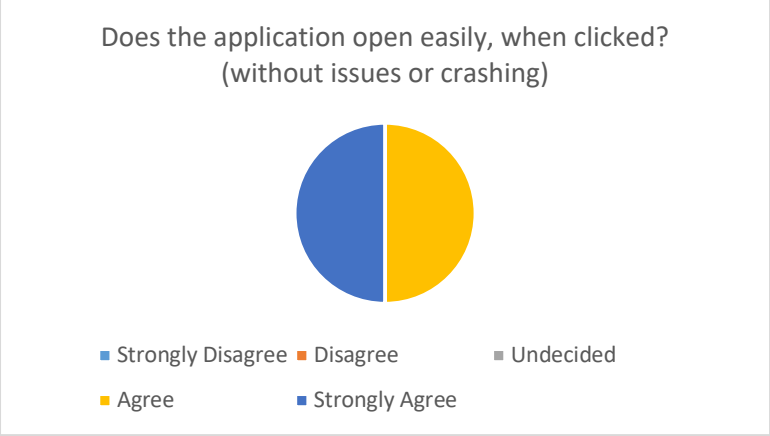
Contractor – User Testing Black Box – Analysis



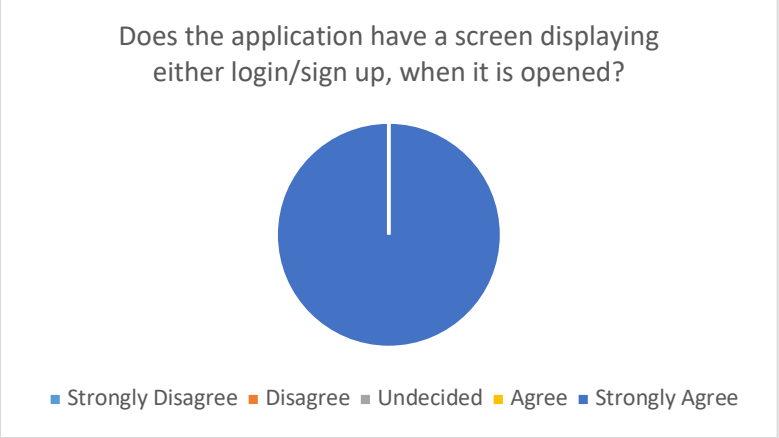
The diagram displays the results for whether the application is visible on the device it is being tested on. All the testers have agreed that it is. As a result, this means we do not have to work on this part of the application (or we have less to improve on).



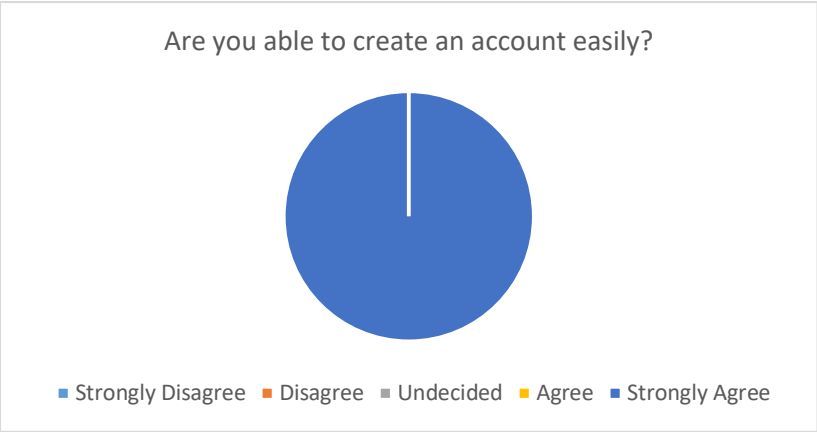
This diagram displays the results for whether the application has an appropriate name. According to the results, all the testers strongly agree that the name is appropriate. As a result, this means we do not have to focus on this part of the application (or we have less to improve on).



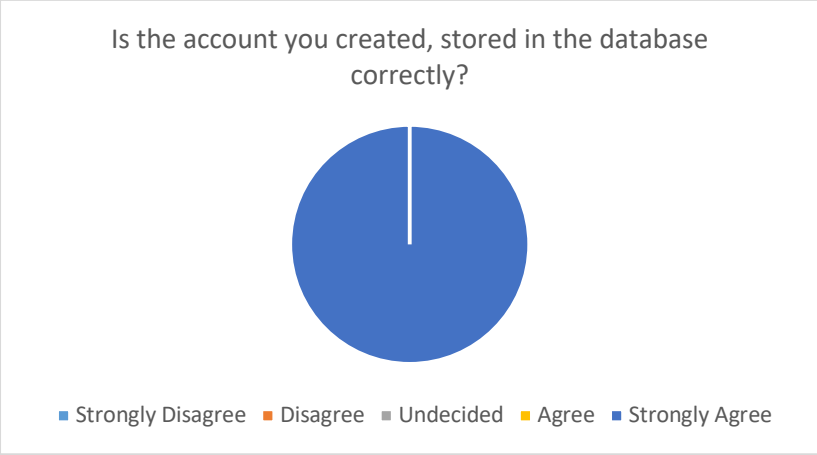
The diagram display the results for whether the application opens with ease, once clicked on. From the results we see an overall agreement that it does open safely, with no crashes. As a result, we do not have to focus our time on this part of the application (or we have less to improve on).



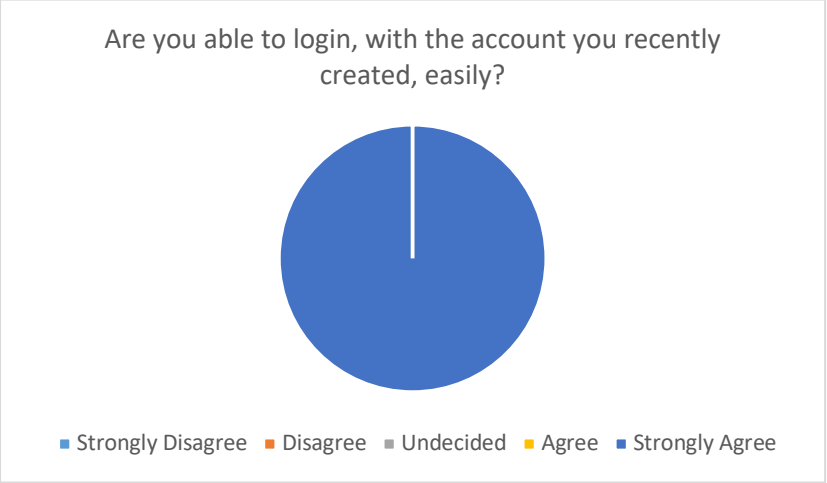
The diagram displays the results for whether the login/sign up page is shown on start up. The results portray that the start-up screen does display login/sign up. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



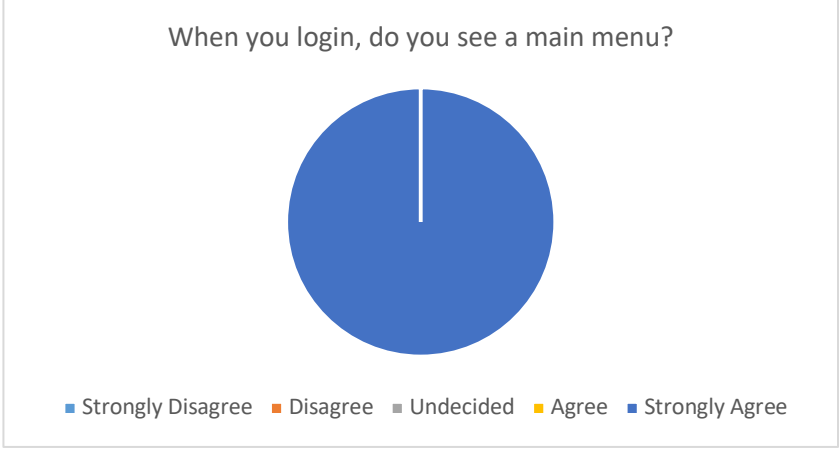
The diagram displays the results for whether an account can be created easily. The results portray that users can make an account fairly easily. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



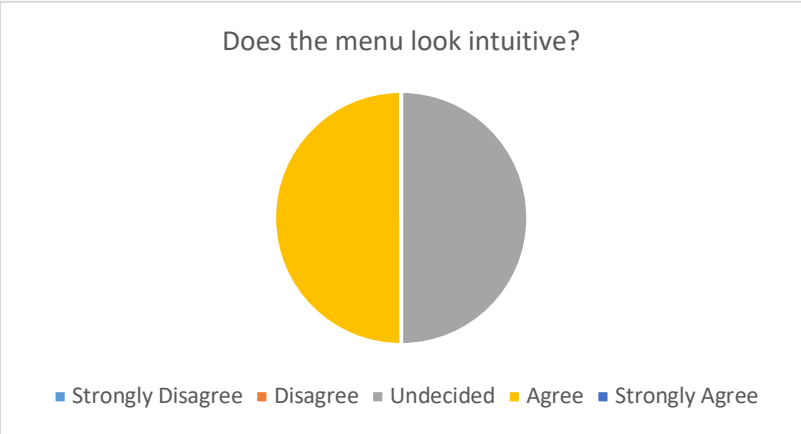
The diagram displays the results for whether the created account is stored in the database correctly. The results portray that the new account is stored in the right destination. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



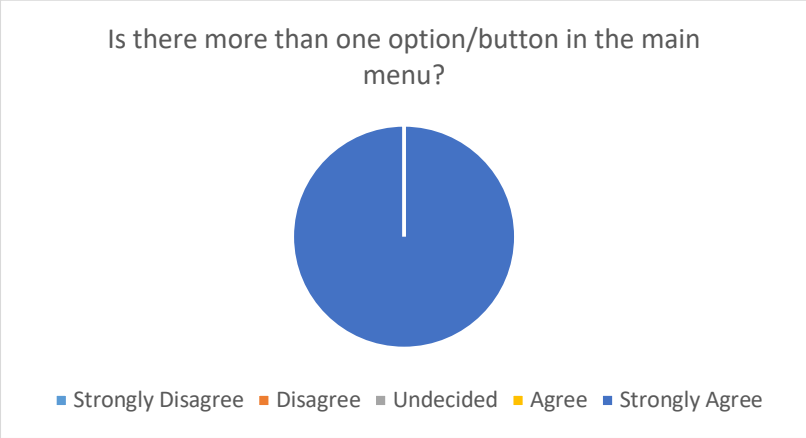
The diagram displays the results for whether the user can sign in using the recently created account. The results portray that the users are able to login easily with new login credentials. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



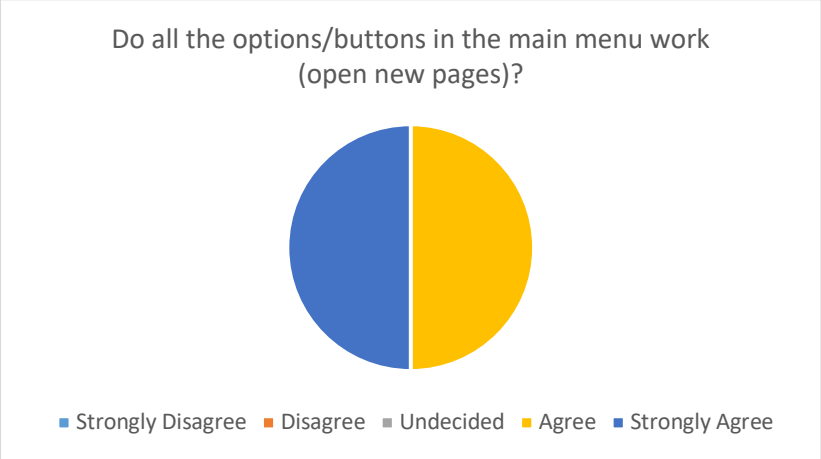
The diagram displays the results for whether after login, you can see a main menu. The results portray that users do see after login, a main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



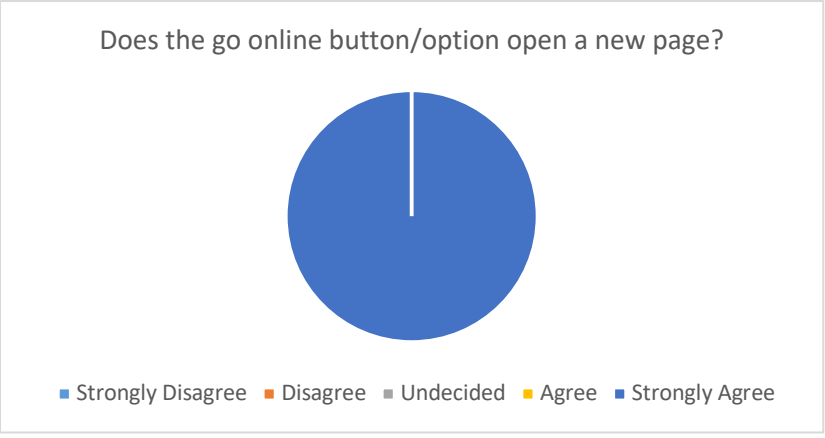
The diagram displays the results for whether the main menu looks intuitive. The results are varied between agreement and undecided. As a result, we need to ensure that we allocate time on this part of the application, so that in the next user tests, we have more users agreeing that the main menu is intuitive.



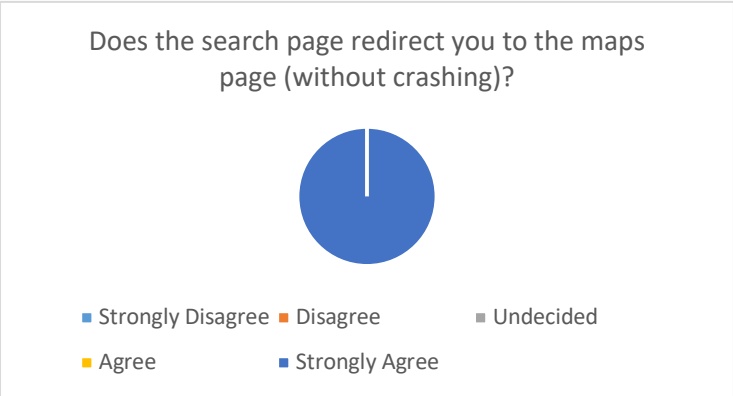
The diagram displays the results for whether there are multiple buttons in the main menu. The results portray that users do see more than one button in the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



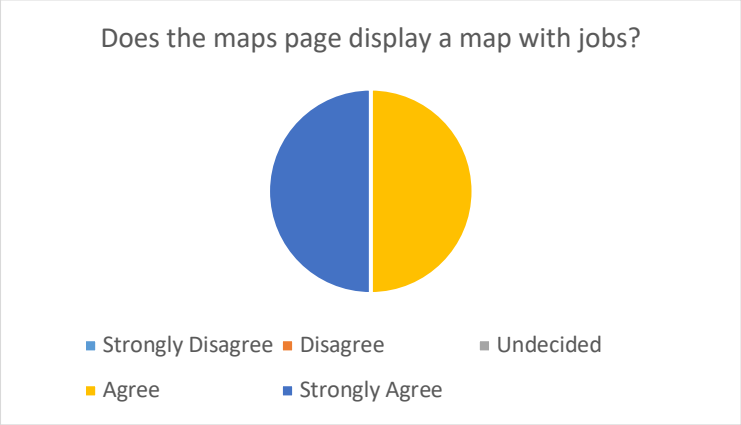
The diagram displays the results for whether the option buttons in the main menu function and open new pages. The results portray that the option buttons do open new pages. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



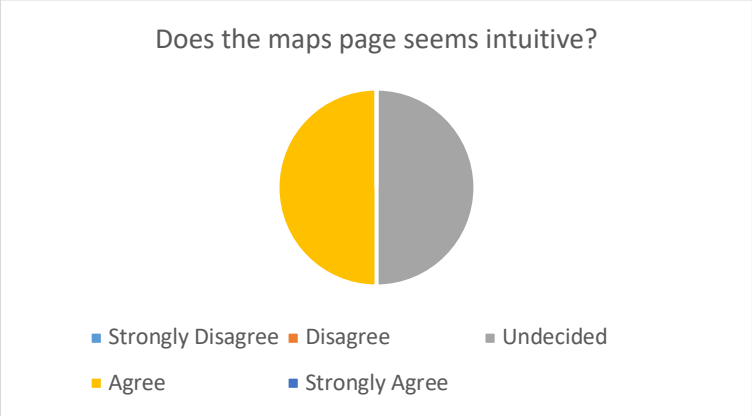
The diagram displays the results for whether the ‘go online’ button functions and opens the map. The results portray that the button does work as purposed. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



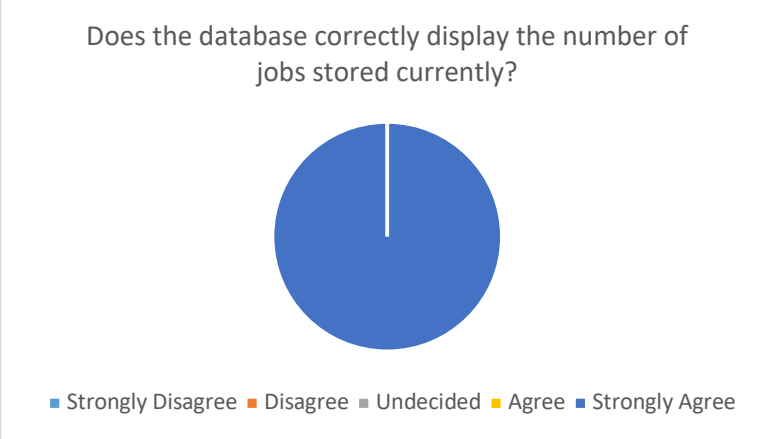
The diagram displays the results for whether the search (go online) button opens maps and doesn't crash. The results portray that the page opens successfully without fail. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



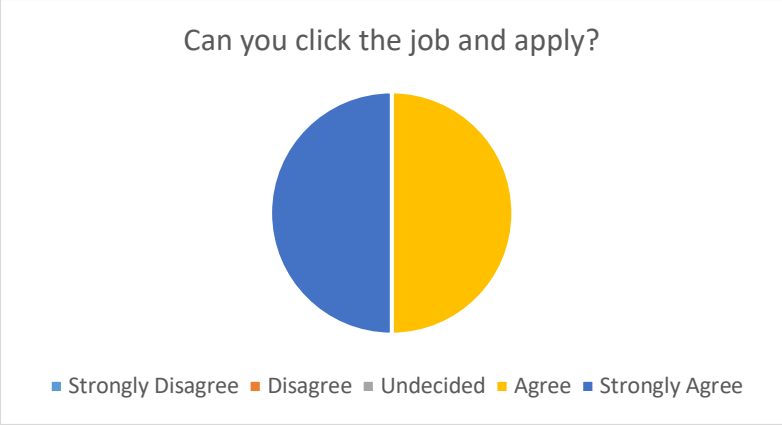
The diagram displays the results for whether the map page displays the jobs. The results portray that the map page does show potential clients. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



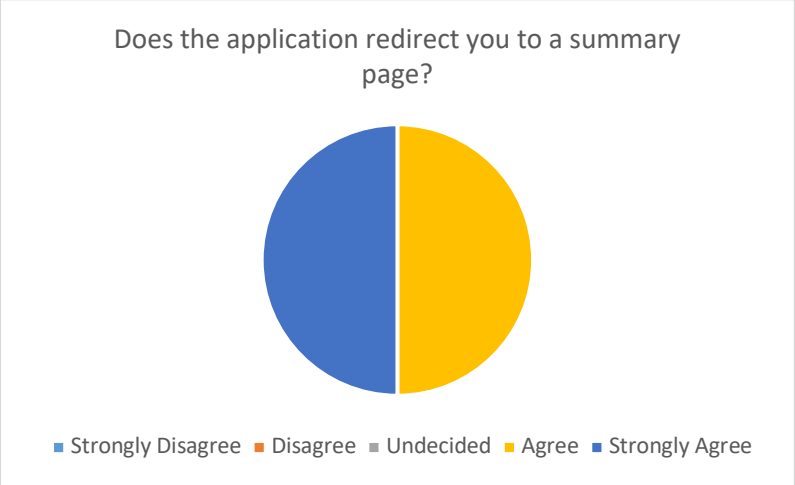
The diagram displays the results for whether the maps page seems intuitive. The results portray that the map page is not as intuitive as could be. As a result, we do have to focus time on this part of the application, and perhaps add hints to improve the intuitiveness.



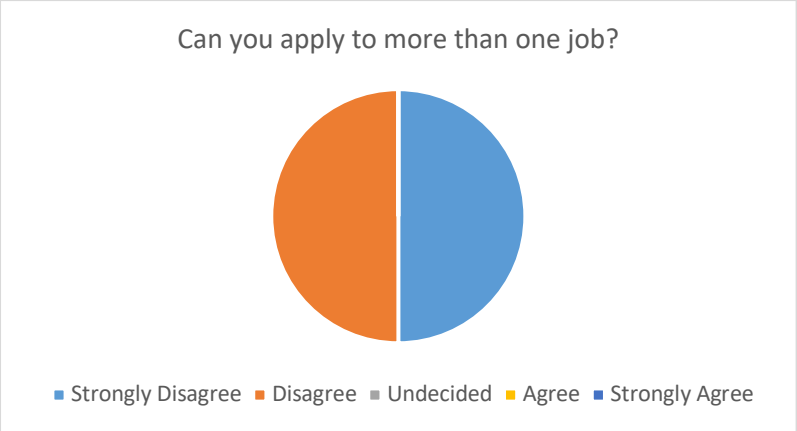
The diagram displays the results for whether the database correctly stores the number of jobs. The results portray that the jobs are stored in the right way. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



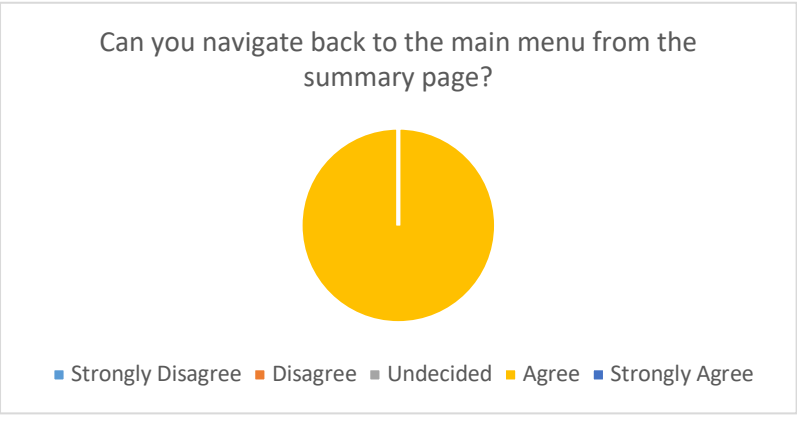
The diagram displays the results for whether you can click the job on the map and apply for it. The results portray that you can. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



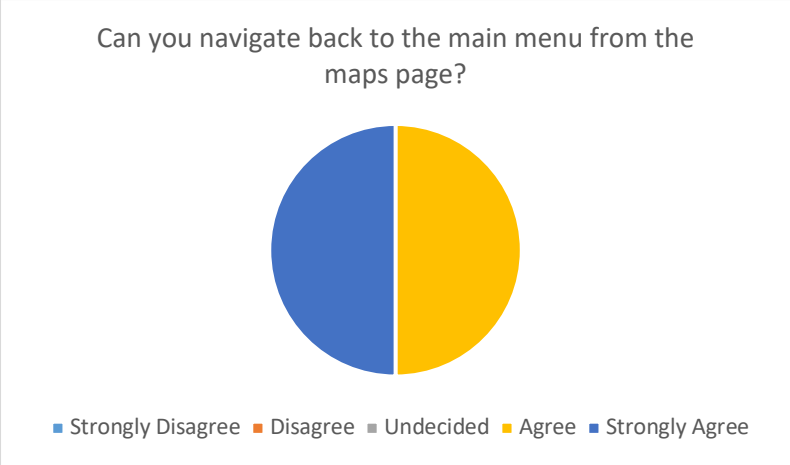
The diagram displays the results for whether the application does redirect to the summary page. The results portray that the app does redirect to the summary page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



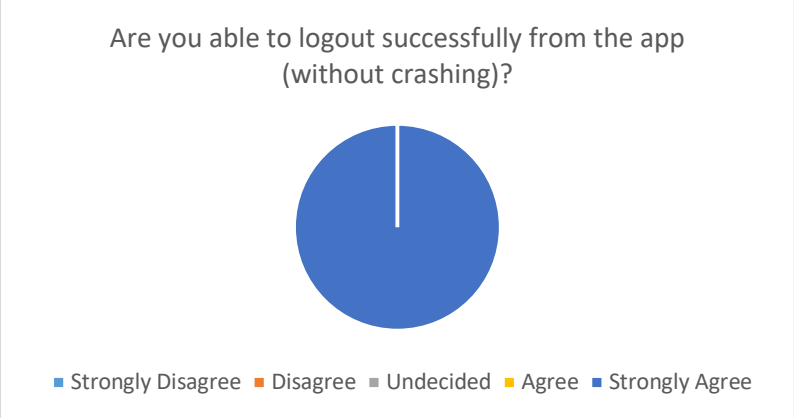
The diagram displays the results for whether you can apply to more than one job. The results portray that you cannot apply for more than one job, or it is difficult to do so. As a result, we need to focus time on this part of the application (or we have less to improve on). We can fix this by placing more jobs on the map.



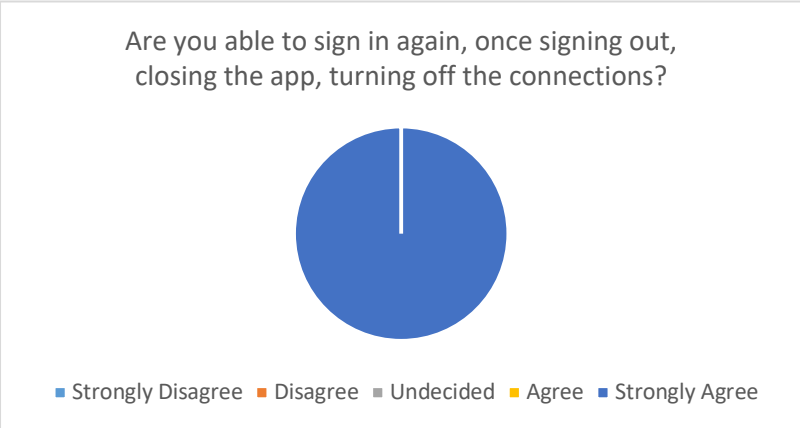
The diagram displays the results for whether you can navigate back to main menu from the summary page. The results portray that you can navigate back to the summary page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether you can navigate back to main menu from the maps page. The results portray that you can navigate back. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



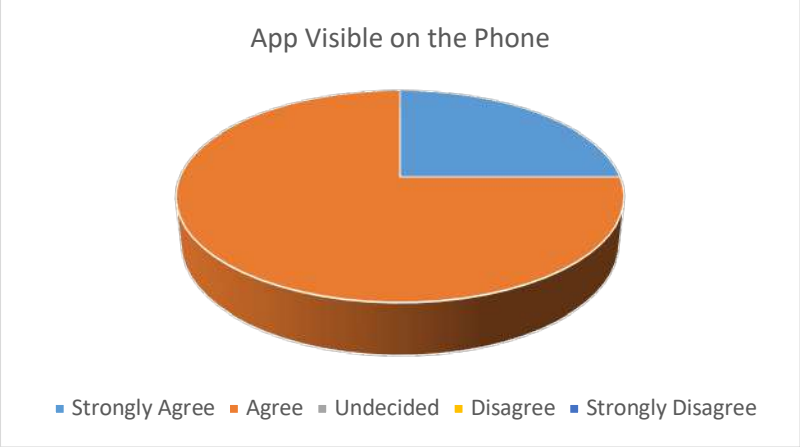
The diagram displays the results for whether you can logout from the application successfully. The results portray that you can . As a result, we do not have to focus time on this part of the application (or we have less to improve on).



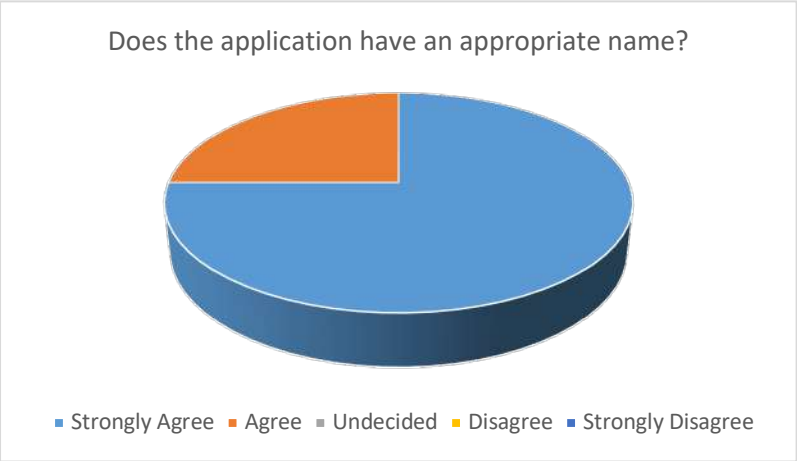
The diagram displays the results for whether you can sign out the app and log back in. The results portray that you can log back in again. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

Overall, most ultimately, we believe the user tests have been successful in terms of finding out which parts of the application function as expected and which parts need improvement. This is in order to create an ending application which suits users' needs and is accessible. We can do this by using the feedback from the contractors. The overall consensus was that we need to improve design, which could in turn improve our intuitiveness. Also, the application has reached the MVP (minimum viable product) and needs more advanced features to take it to the next level.

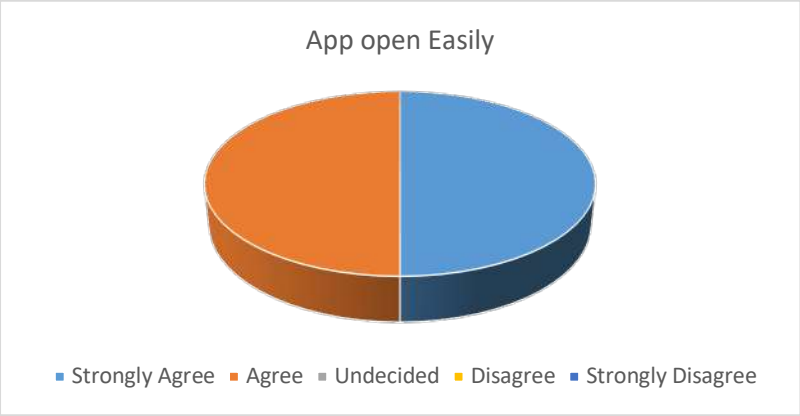
Service Requester – User Testing Black Box – Analysis



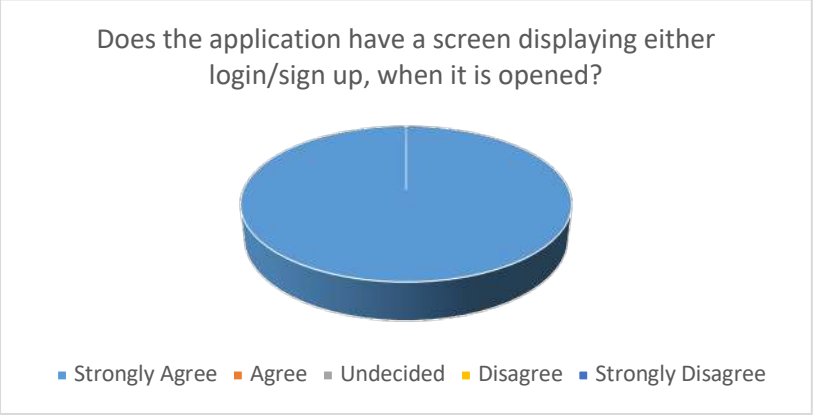
The diagram displays the results for whether the application is visible on the device it is being tested on. All the testers have agreed that it is. As a result, this means we do not have to work on this part of the application (or we have less to improve on).



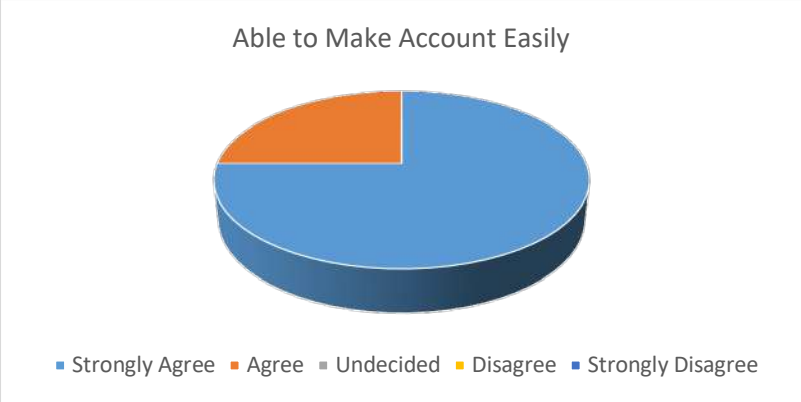
This diagram displays the results for whether the application has an appropriate name. According to the results, all the testers strongly agree that the name is appropriate. As a result, this means we do not have to focus on this part of the application (or we have less to improve on).



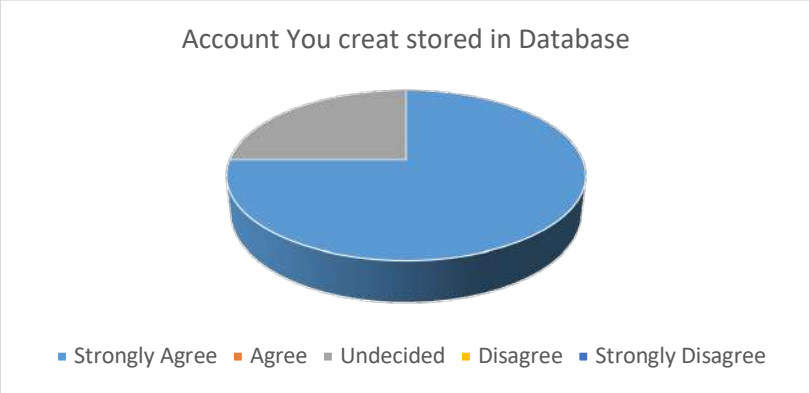
The diagram display the results for whether the application opens with ease, once clicked on. From the results we see an overall agreement that it does open safely, with no crashes. As a result, we do not have to focus our time on this part of the application (or we have less to improve on).



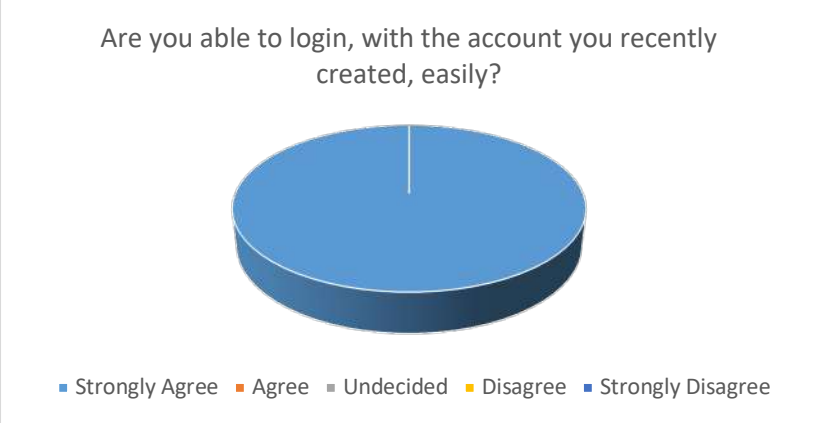
The diagram displays the results for whether the login/sign up page is shown on start up. The results portray that the start-up screen does display login/sign up. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



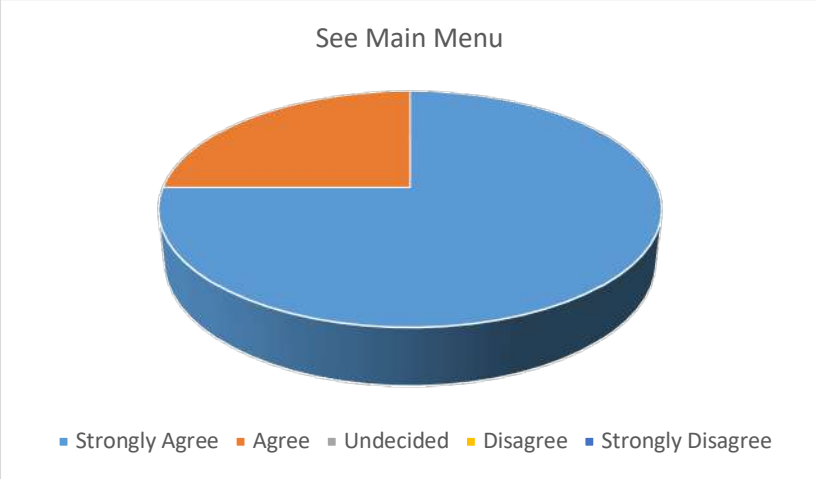
The diagram displays the results for whether an account can be created easily. The results portray that users can make an account fairly easily. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



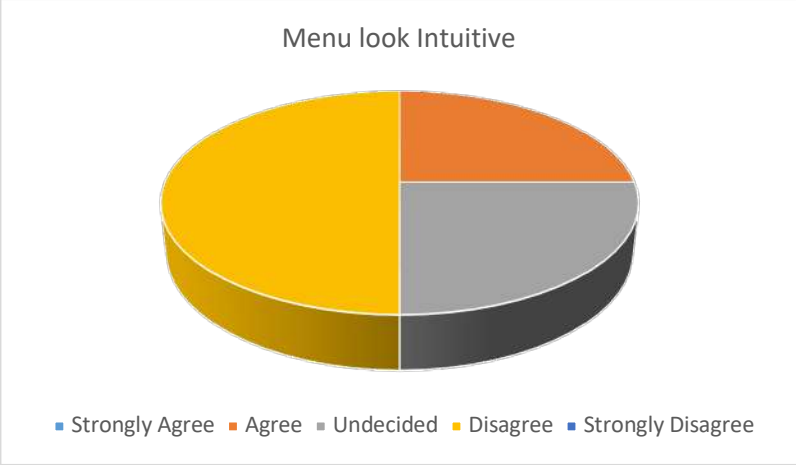
The diagram displays the results for whether the created account is stored in the database correctly. The results portray that the new account is stored in the right destination. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



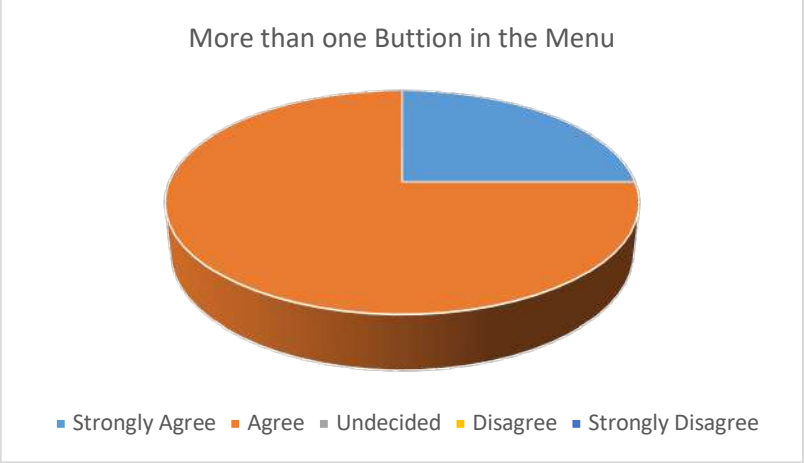
The diagram displays the results for whether the user can sign in using the recently created account. The results portray that the users are able to login easily with new login credentials. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



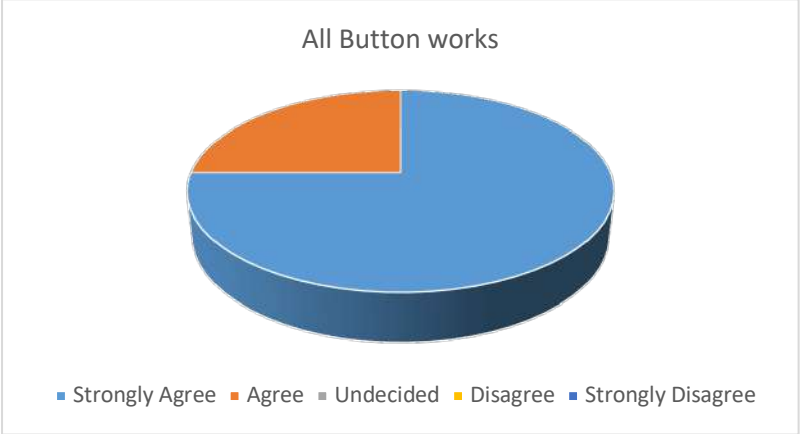
The diagram displays the results for whether after login, you can see a main menu. The results portray that users do see after login, a main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



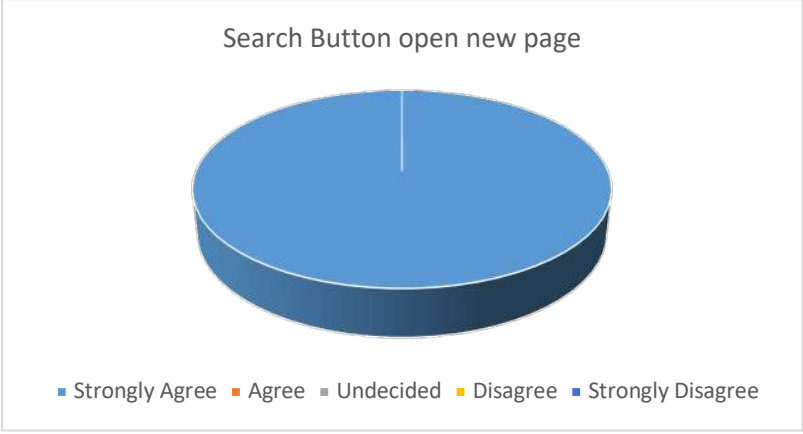
The diagram displays the results for whether the main menu looks intuitive. The results are varied between agreement and undecided. As a result, we need to ensure that we allocate time on this part of the application, so that in the next user tests, we have more users agreeing that the main menu is intuitive.



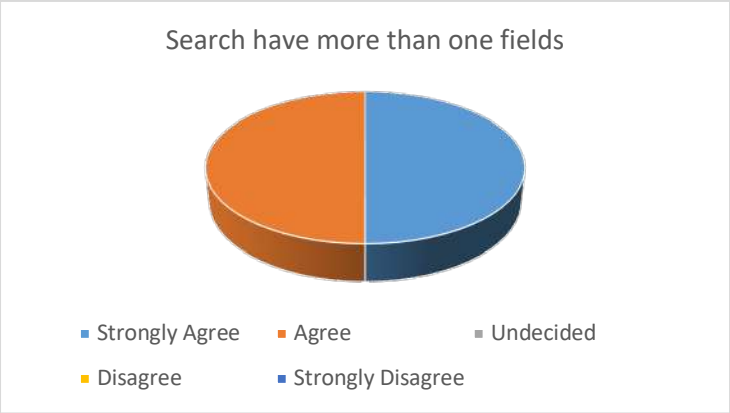
The diagram displays the results for whether there are multiple buttons in the main menu. The results portray that users do see more than one button in the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether the option buttons in the main menu function and open new pages. The results portray that the option buttons do open new pages. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

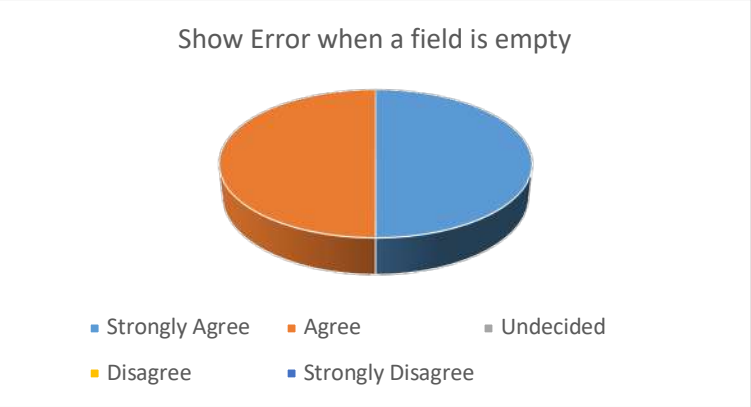


The diagram displays the results for whether the search button functions and opens the services list and details filling-in form. The results portray that the button does work as purposed. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

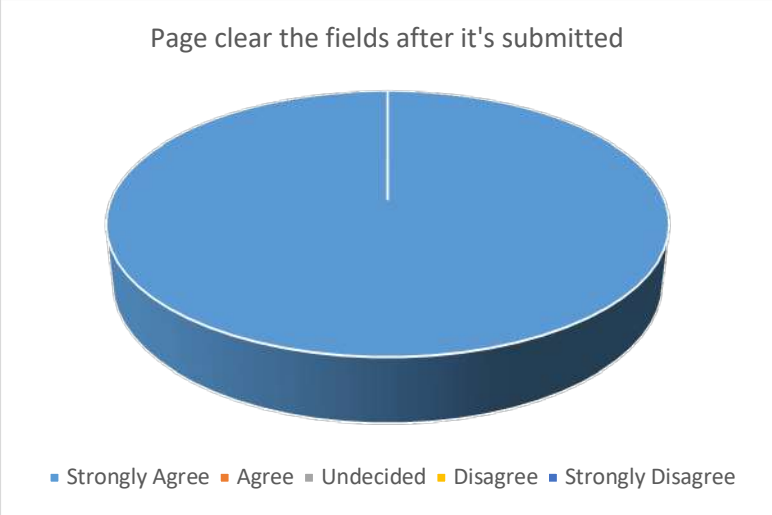


The diagram displays the results for whether the search (services page) has more than one field. The results portray that there are more than one field on this page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

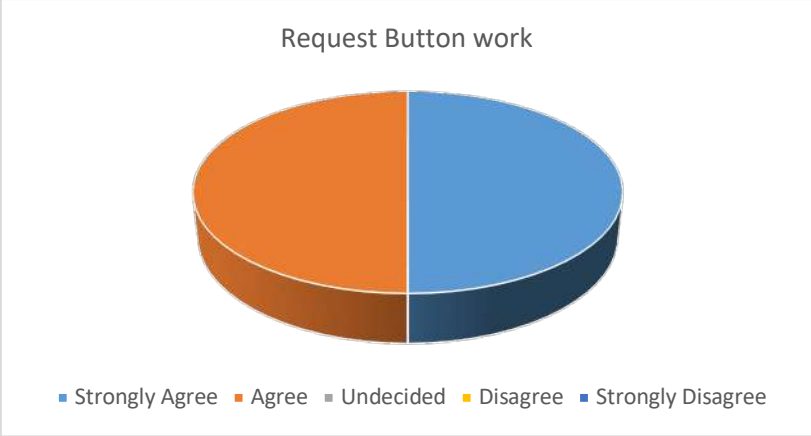
The diagram displays the results for whether the search (services page) allows you to search for a service. The results portray users are able to search (choose) a service. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



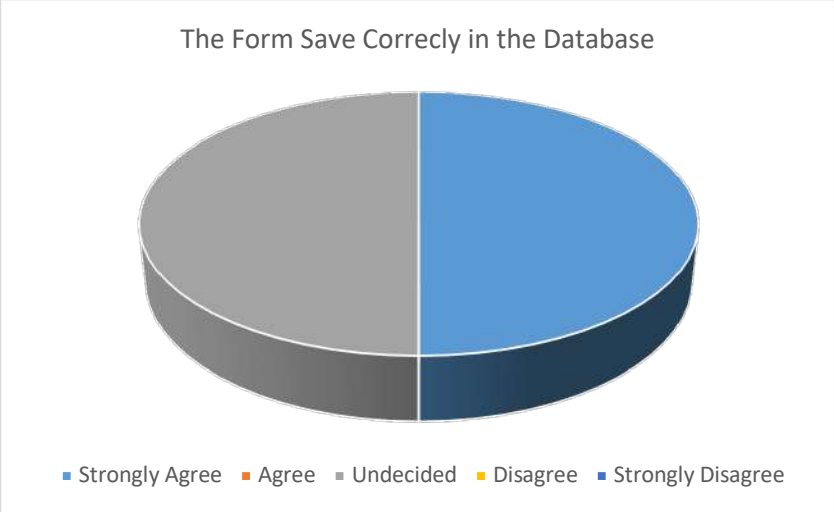
The diagram displays the results for whether the fields on the search (services page) produce an error message when left empty. The results portray that an error message is shown. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



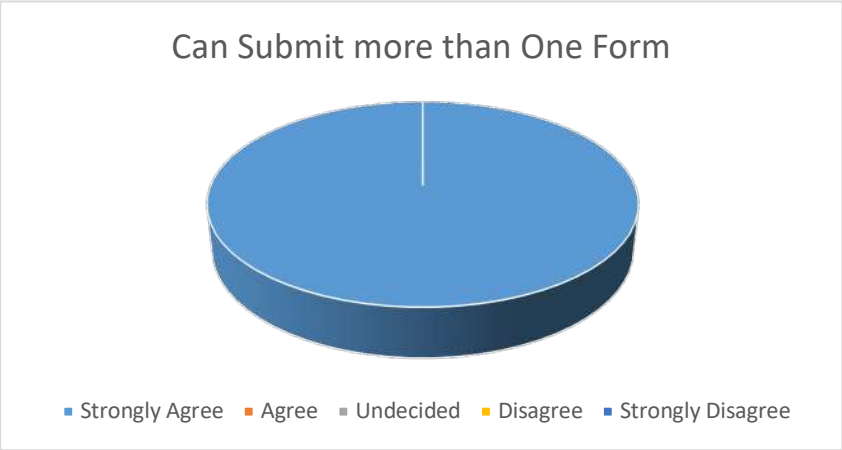
The diagram displays the results for whether the search (services page) clears the fields after being submitted. The results portray that fields are emptied after the request button is clicked and fields are filled. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



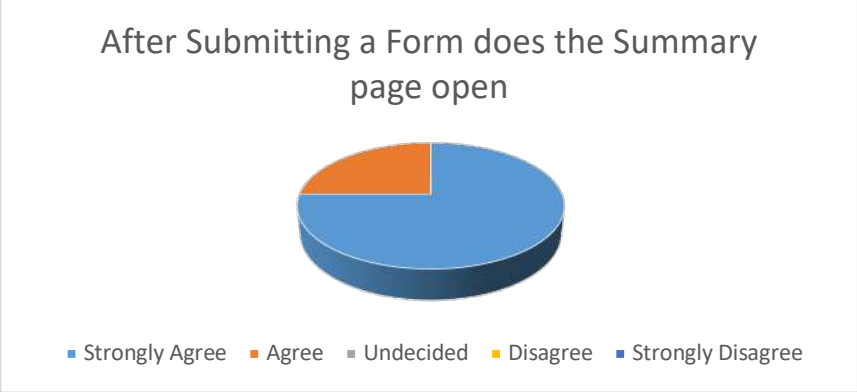
The diagram displays the results for whether the request button works. The results portray that once clicked the request button functions, and the summary page is displayed. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



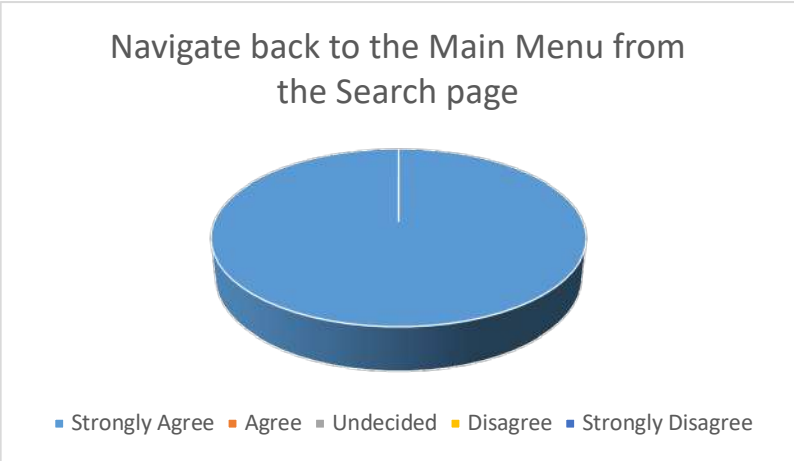
The diagram displays the results for whether the form is correctly stored in the database. The results are varied between agreement and undecided. As a result, we need to ensure that we allocate time on this part of the application. We need to ensure the form is stored correctly, as this is a major component of the application, and also affects the contractor app.



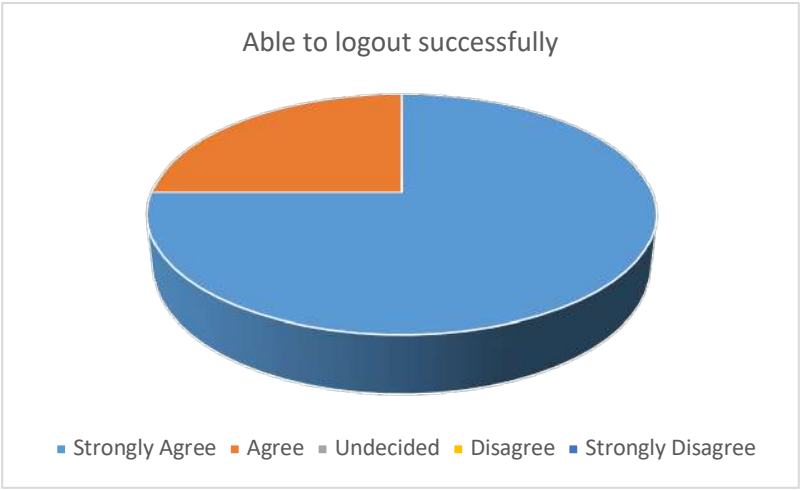
The diagram displays the results for whether the application allows more than one submission of the form. The results portray that the app does allow multiple request submissions. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



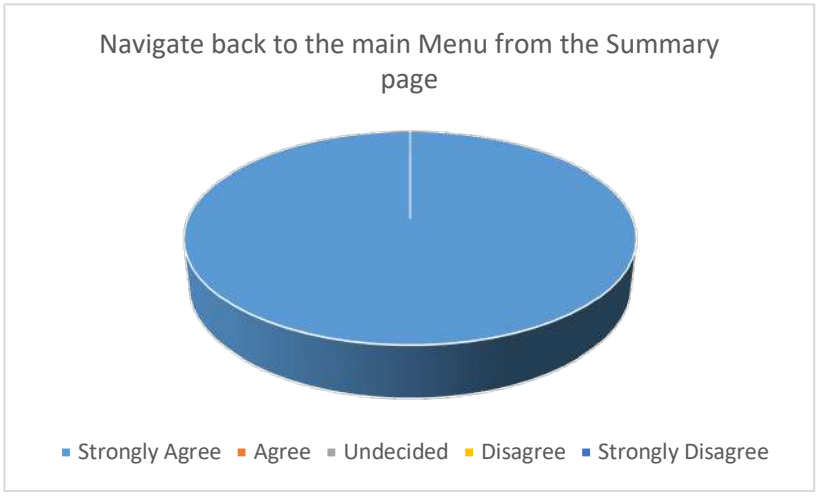
The diagram above displays the results for whether the application does redirect to the summary page. The results portray that the app does redirect to the summary page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether you can navigate back to main menu from the search (services) page. The results portray that you can navigate back to the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether you can logout from the application successfully. The results portray that you can . As a result, we do not have to focus time on this part of the application (or we have less to improve on).

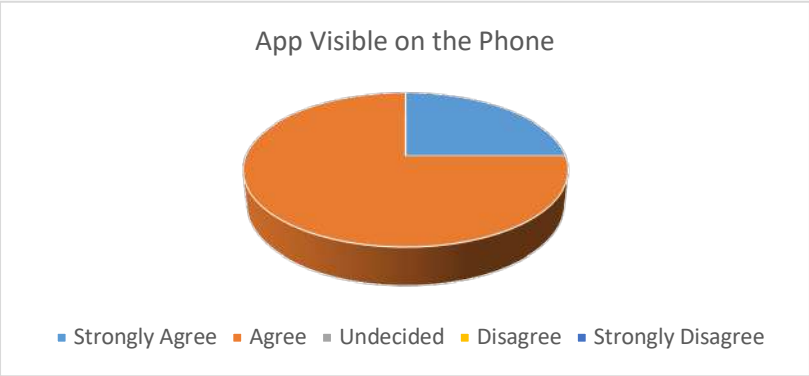


The diagram displays the results for whether you can navigate back to the main menu from the summary page. The results portray that you can go back to the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

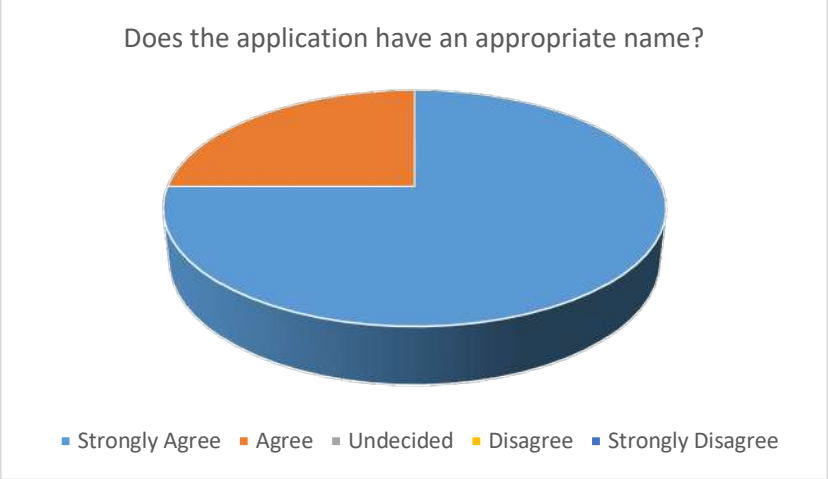
The diagram displays the results for whether you can sign out the app and log back in. The results portray that you can log back in again. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

Overall, most ultimately, we believe the user tests have been successful in terms of finding out which parts of the application function as expected and which parts need improvement. This is in order to create an ending application which suits users' needs and is accessible. We can do this by using the feedback from the contractors. The overall consensus was that we need to improve design, which could in turn improve our intuitiveness. Also, the application has reached the MVP (minimum viable product) and needs more advanced features to take it to the next level. Furthermore, we will also need to go over some of the more basic functionalities of the app to ensure they also pass the minimum criterion.

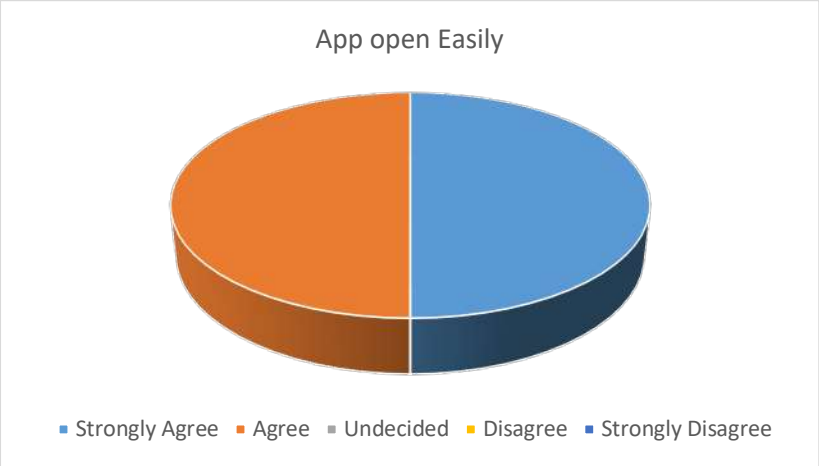
Service Requester – User Testing Black Box – Analysis



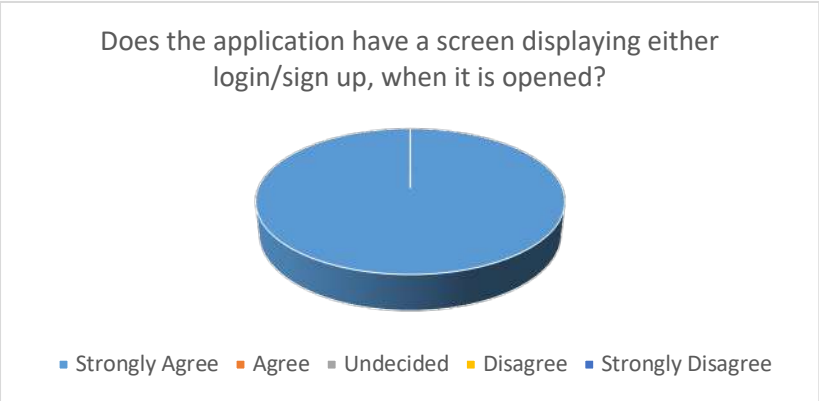
The diagram displays the results for whether the application is visible on the device it is being tested on. All the testers have agreed that it is. As a result, this means we do not have to work on this part of the application (or we have less to improve on).



This diagram displays the results for whether the application has an appropriate name. According to the results, all the testers strongly agree that the name is appropriate. As a result, this means we do not have to focus on this part of the application (or we have less to improve on).

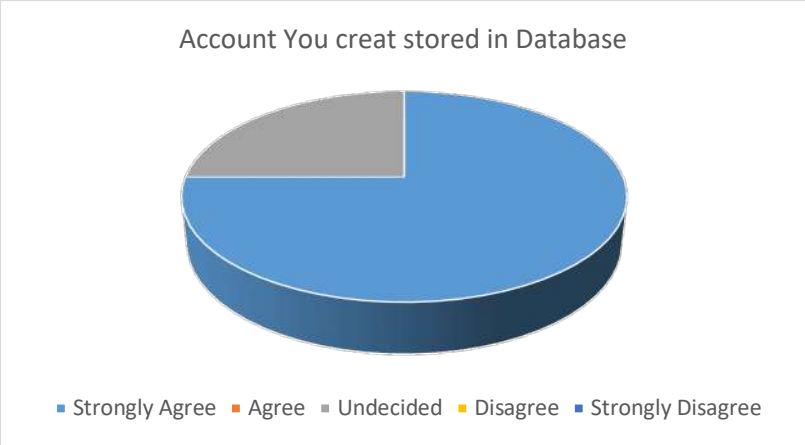


The diagram display the results for whether the application opens with ease, once clicked on. From the results we see an overall agreement that it does open safely, with no crashes. As a result, we do not have to focus our time on this part of the application (or we have less to improve on).

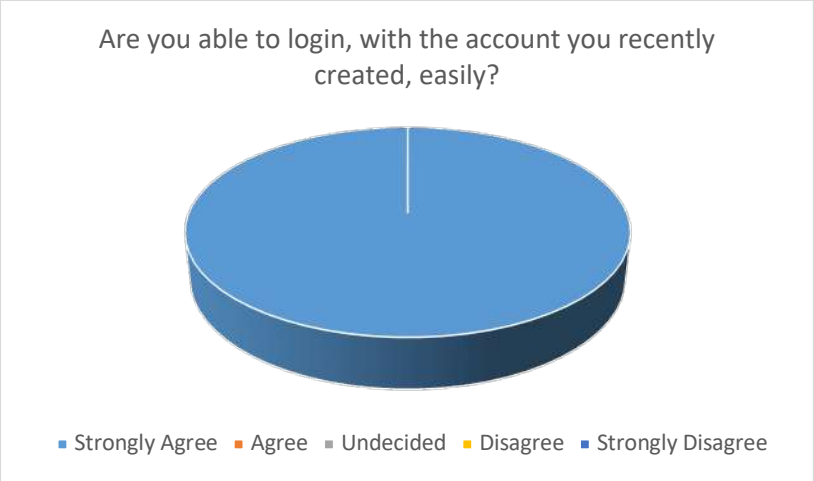


The diagram displays the results for whether the login/sign up page is shown on start up. The results portray that the start-up screen does display login/sign up. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

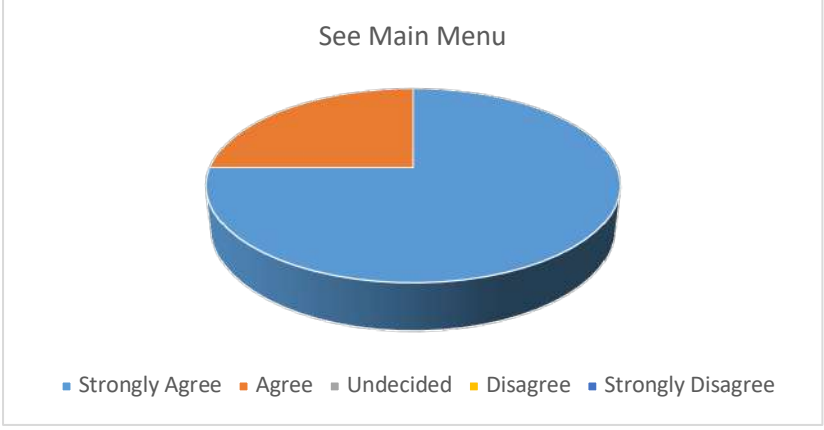
The diagram displays the results for whether an account can be created easily. The results portray that users can make an account fairly easily. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



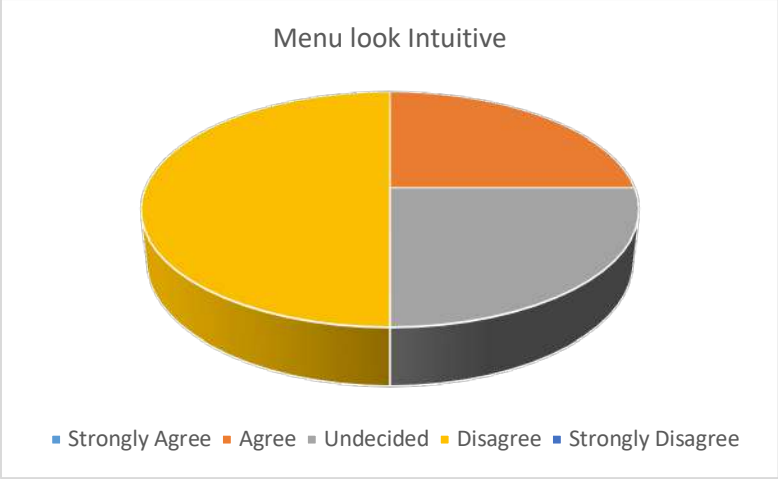
The diagram displays the results for whether the created account is stored in the database correctly. The results portray that the new account is stored in the right destination. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



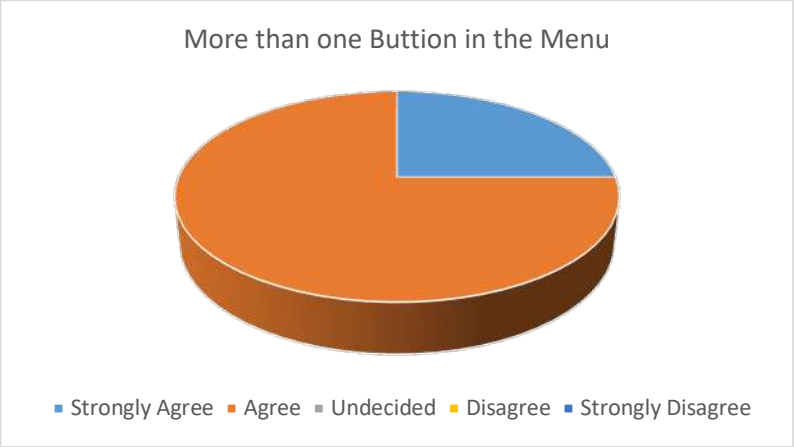
The diagram displays the results for whether the user can sign in using the recently created account. The results portray that the users are able to login easily with new login credentials. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



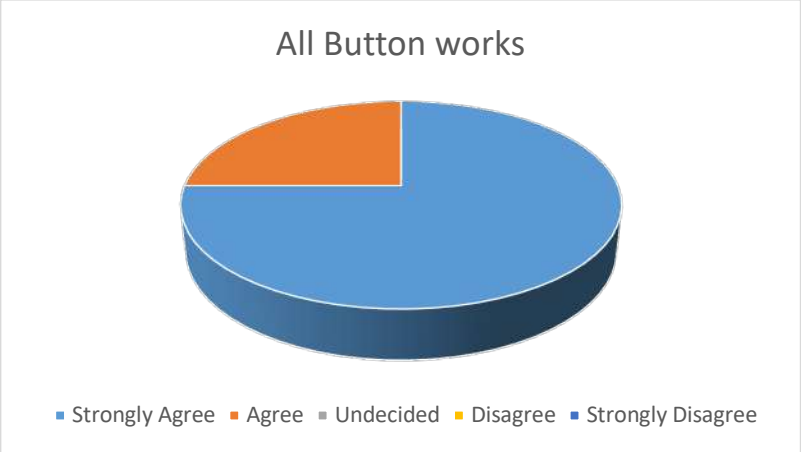
The diagram displays the results for whether after login, you can see a main menu. The results portray that users do see after login, a main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



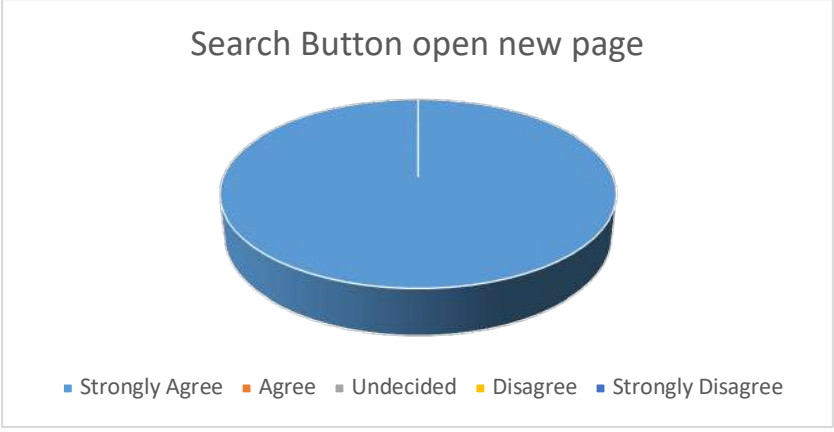
The diagram displays the results for whether the main menu looks intuitive. The results are varied between agreement and undecided. As a result, we need to ensure that we allocate time on this part of the application, so that in the next user tests, we have more users agreeing that the main menu is intuitive.



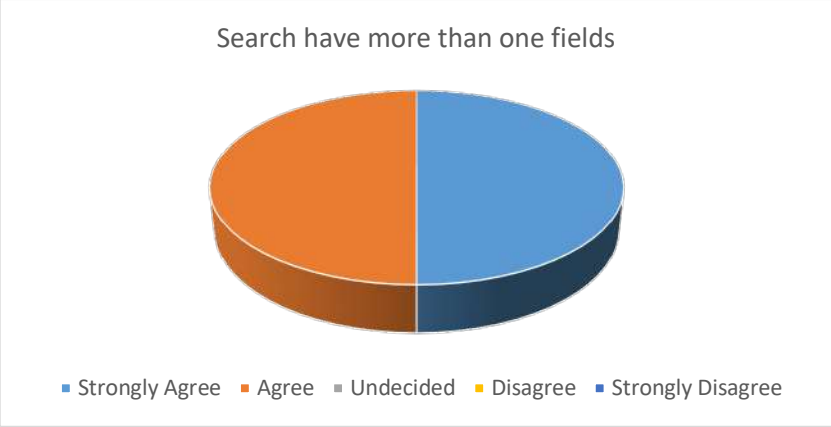
The diagram displays the results for whether there are multiple buttons in the main menu. The results portray that users do see more than one button in the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether the option buttons in the main menu function and open new pages. The results portray that the option buttons do open new pages. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

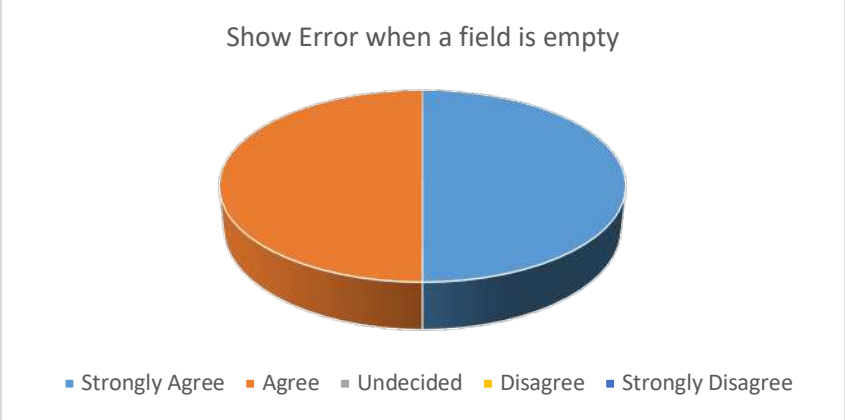


The diagram displays the results for whether the search button functions and opens the services list and details filling-in form. The results portray that the button does work as purposed. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

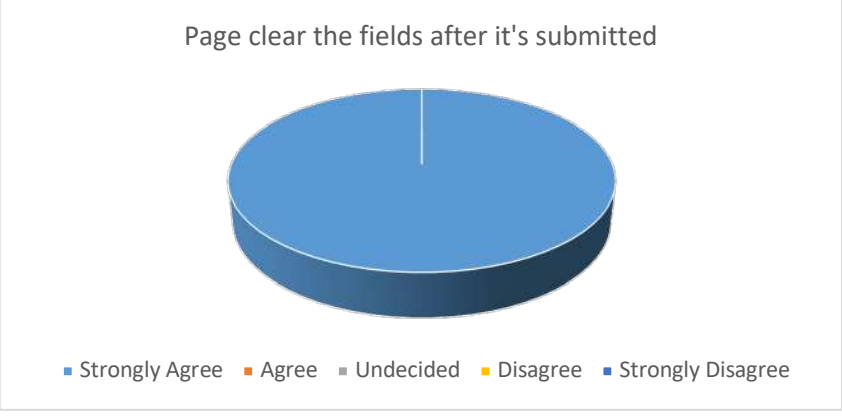


The diagram displays the results for whether the search (services page) has more than one field. The results portray that there are more than one field on this page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

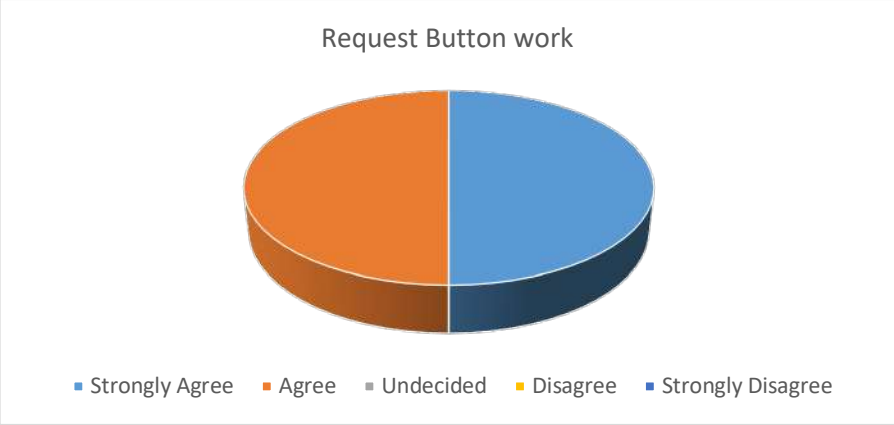
The diagram displays the results for whether the search (services page) allows you to search for a service. The results portray users are able to search (choose) a service. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



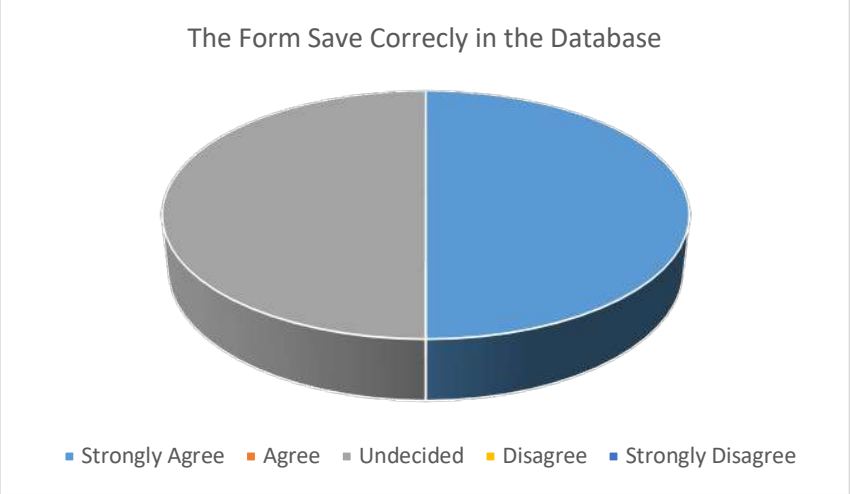
The diagram displays the results for whether the fields on the search (services page) produce an error message when left empty. The results portray that an error message is shown. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



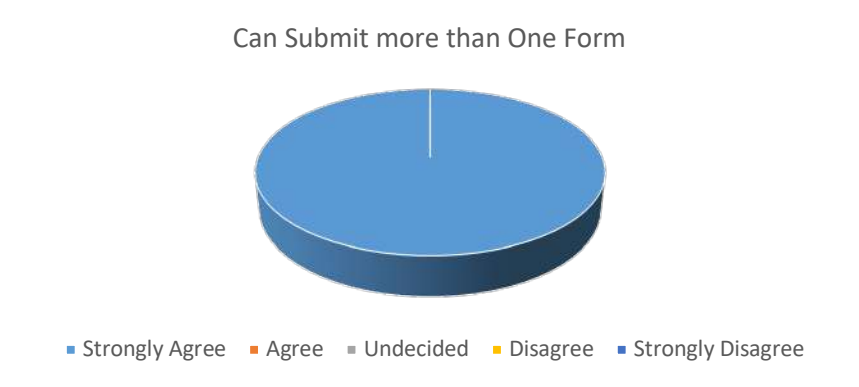
The diagram displays the results for whether the search (services page) clears the fields after being submitted. The results portray that fields are emptied after the request button is clicked and fields are filled. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



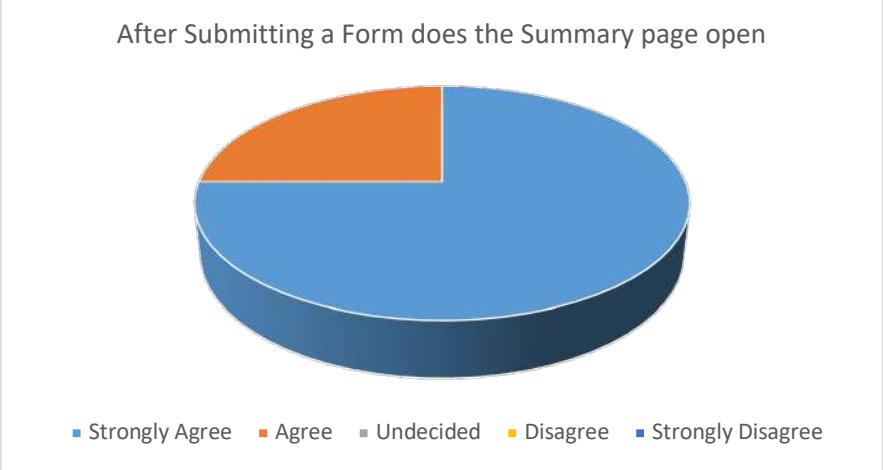
The diagram displays the results for whether the request button works. The results portray that once clicked the request button functions, and the summary page is displayed. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



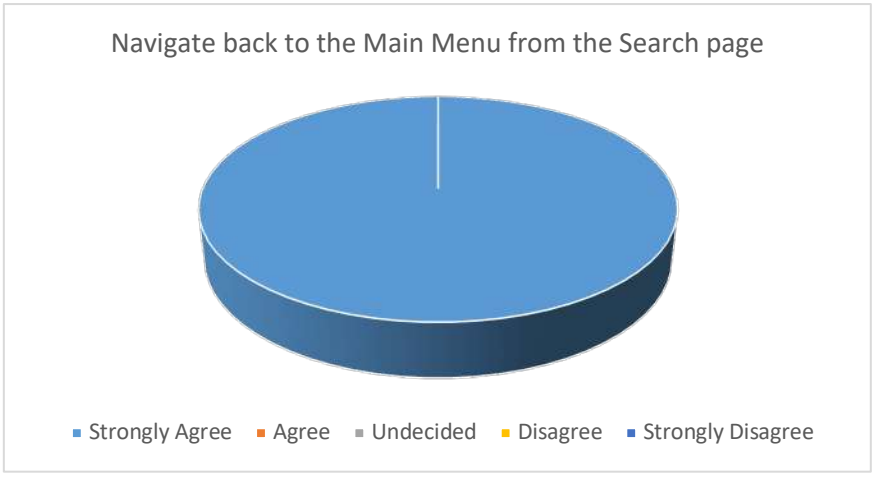
The diagram displays the results for whether the form is correctly stored in the database. The results are varied between agreement and undecided. As a result, we need to ensure that we allocate time on this part of the application. We need to ensure the form is stored correctly, as this is a major component of the application, and also affects the contractor app.



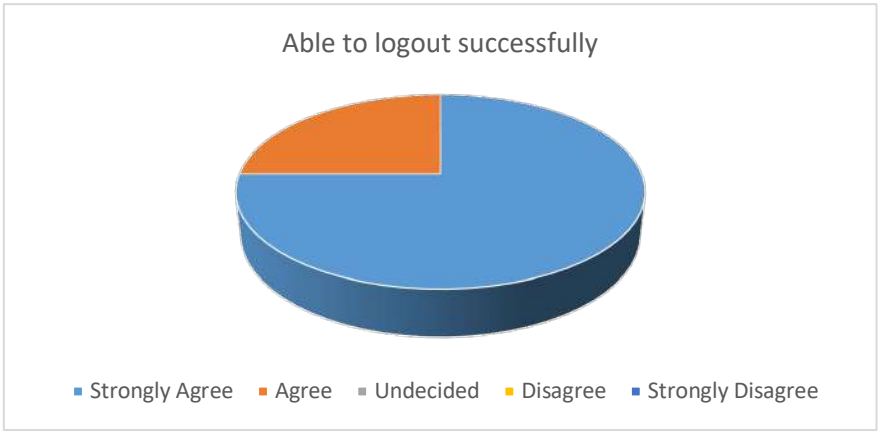
The diagram displays the results for whether the application allows more than one submission of the form. The results portray that the app does allow multiple request submissions. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



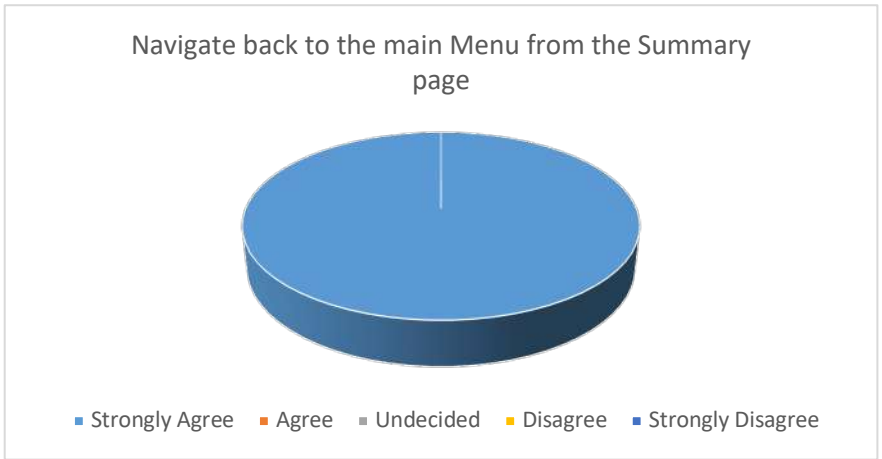
The diagram displays the results for whether the application does redirect to the summary page. The results portray that the app does redirect to the summary page. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



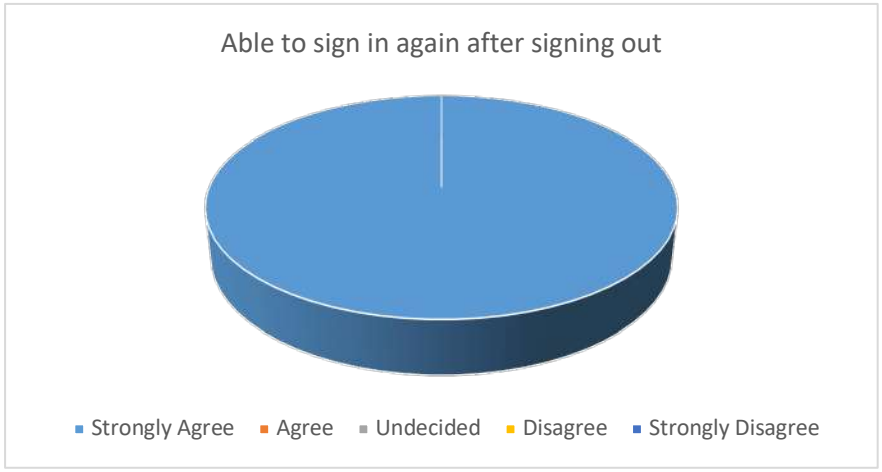
The diagram displays the results for whether you can navigate back to main menu from the search (services) page. The results portray that you can navigate back to the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether you can logout from the application successfully. The results portray that you can . As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether you can navigate back to the main menu from the summary page. The results portray that you can go back to the main menu. As a result, we do not have to focus time on this part of the application (or we have less to improve on).



The diagram displays the results for whether you can sign out the app and log back in. The results portray that you can log back in again. As a result, we do not have to focus time on this part of the application (or we have less to improve on).

Overall, most ultimately, we believe the user tests have been successful in terms of finding out which parts of the application function as expected and which parts need improvement. This is in order to create an ending application which suits users' needs and is accessible. We can do this by using the feedback from the contractors. The overall consensus was that we need to improve design, which could in turn improve our intuitiveness. Also, the application has reached the MVP (minimum viable product) and needs more advanced features to take it to the next level. Furthermore, we will also need to go over some of the more basic functionalities of the app to ensure they also pass the minimum criterion.

Integration Testing – Contractor

Integration testing consists of testing whether different components of the application work simultaneously together (i.e. Activity class files with firebase database).

Test	Input	Status (Fail/Pass)
When entering Login Activity credentials, see if it checks for username and password in database, and opens Profile Activity	Username: amandeepb@labora.com Password: netflix	✓

Test	Input	Status (Fail/Pass)
When entering Register Activity credentials, see if it stores username and password in database, and opens Register 2 Activity	Username: amandeep20220@live.com Password: password123	✓

Test	Input	Status (Fail/Pass)
When entering Register Activity 2 fields, see if it stores fields in database, and opens Profile Activity	Address: 50 Billet Road, Romford Full name: Thomas Occupation: Plumber Phone: 0208528173818	✓

Integration Testing – Service Requester

Integration testing consists of testing whether different components of the application work simultaneously together (i.e. Activity class files with firebase database).

Test	Input	Status (Fail/Pass)
When entering Register Activity credentials, see if it stores username and password in database, and opens Register 2 Activity	Username: test@labora.uk Password: netflix	✓

Test	Input	Status (Fail/Pass)
When entering Login Activity credentials, see if it checks for username and password in database, and opens Profile Activity	Username: Shakil@goldsmiths.com Password: 123456	✓

Test	Input	Status (Fail/Pass)
When entering Register Activity 2 fields, see if it stores the fields present in database , and opens Profile Activity	Full name: Jane Doe Phone: 07908349795	✓

Test	Input	Status (Fail/Pass)
When entering Services fields, see if it stores the fields present, and opens Summary Activity	Service: Plumber Post Code: SE14 6NW Job Description: Sink issue, budget 500 pounds Keywords: sink, tap	✓

Systematic Testing – Service Requester

Systematic Testing is testing the software against a variety of inputs, to ensure there are no crashes or bugs

Test	Input	Status (Pass/Fail)
Username	Tom	✗
firebaseAuth.signInWithEmailAndPassword(email, password) This looks for an @ symbol, followed by a domain name		
Username	Tom@hotmail.co.uk	✓
Username	Laura@gmail.com	✓
Username	test@labora.com	✓

Test	Input	Status (Pass/Fail)
Password	12345	✗
firebaseAuth.signInWithEmailAndPassword(email, password) This limits the password to 6 characters or more		
Password	123456	✓
Password	Password123	✓
Password	1234!hello	✓

Test	Input	Status (Pass/Fail)
Full name	Tom Hanks	✓
Full name	“” (Empty Field)	✗
if(!TextUtils.*isEmpty*(name) && !TextUtils.*isEmpty*(phone))		
Full name	Amandeep Bharj	✓

Test	Input	Status (Pass/Fail)
Phone	02034897488	✓
Phone	hello	✗
if(!TextUtils.*isEmpty*(name) && !TextUtils.*isEmpty*(phone))		
Phone	999	✓

Test	Input	Status (Pass/Fail)
Service	Choose a service	✗
if(TextUtils.*isEmpty*(requesterService) || requesterService.equals("Choose a service"))		
Service	Plumber	✓
Service	Yoga Instructor	✓

Test	Input	Status (Pass/Fail)
Post Code	SE14	✓
Post Code	EN3	✓
Post Code	“” (Empty Field)	✗

```
else if(TextUtils.isEmpty(requesterPostCode))
```

Test	Input	Status (Pass/Fail)
Job Description	The sink is broken	✔
Job Description	“” (Empty Field)	✗
else if(TextUtils.*isEmpty*(requesterJob))		
Job Description	Budget 500 pounds	✔

Test	Input	Status (Pass/Fail)
Keywords	“” (Empty Field)	✗
else if(TextUtils.*isEmpty*(requesterKeyWord))		
Keywords	Sink, Tap	✔
Keywords	Female, Maths	✔

Systematic Testing – Contractor

Systematic Testing is testing the software against a variety of inputs, to ensure there are no crashes or bugs

Test	Input	Status (Pass/Fail)
Username	Tom	✗
	firebaseAuth.signInWithEmailAndPassword(email, password) This looks for an @ symbol, followed by a domain name	
Username	Tom@hotmail.co.uk	✔
Username	Laura@gmail.com	✔
Username	test@labora.com	✔

Test	Input	Status (Pass/Fail)
Password	12345	✗
	firebaseAuth.signInWithEmailAndPassword(email, password) This limits the password to 6 characters or more	
Password	123456	✔
Password	Password123	✔
Password	1234!hello	✔

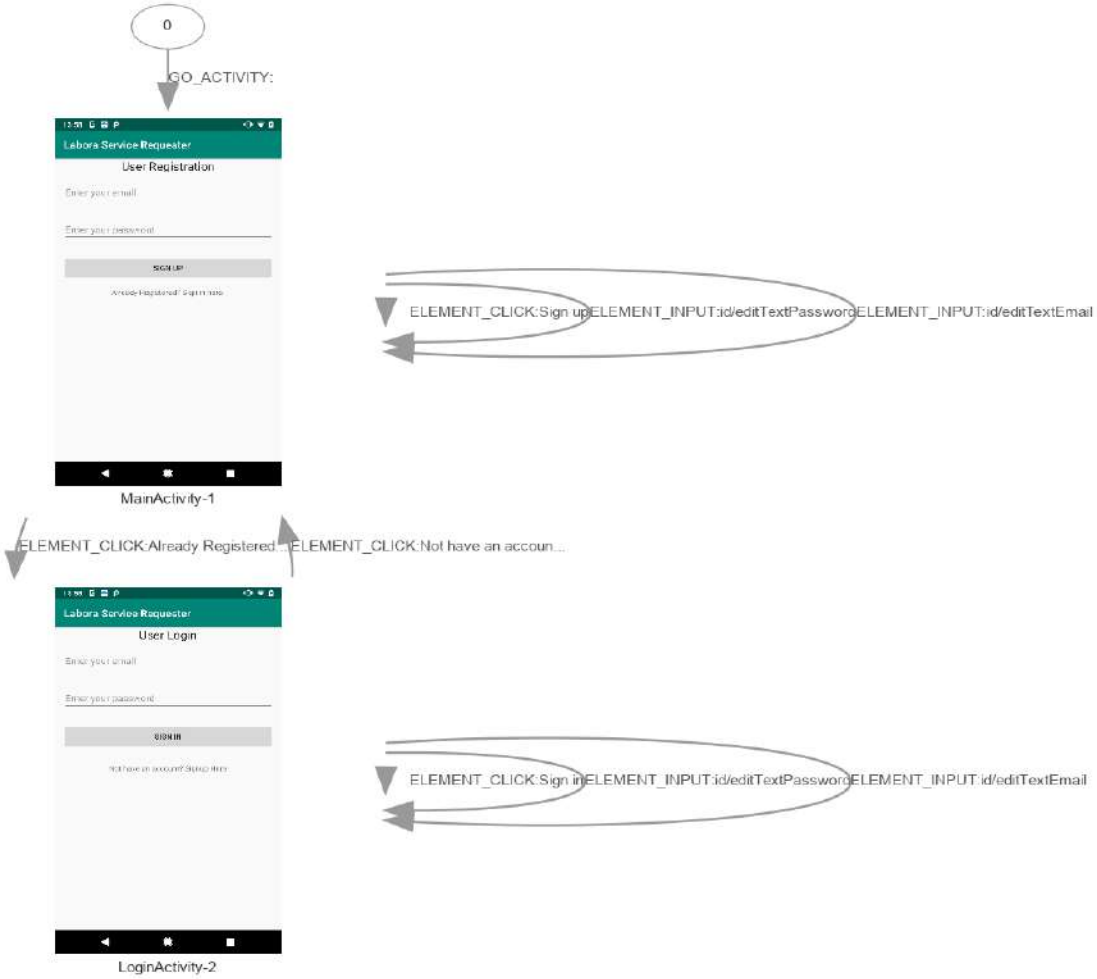
Test	Input	Status (Pass/Fail)
Full name	Tom Hanks	✔
Full name	“” (Empty Field)	✗
	if(TextUtils.*isEmpty*(name))	
Full name	Amandeep Bharj	✔

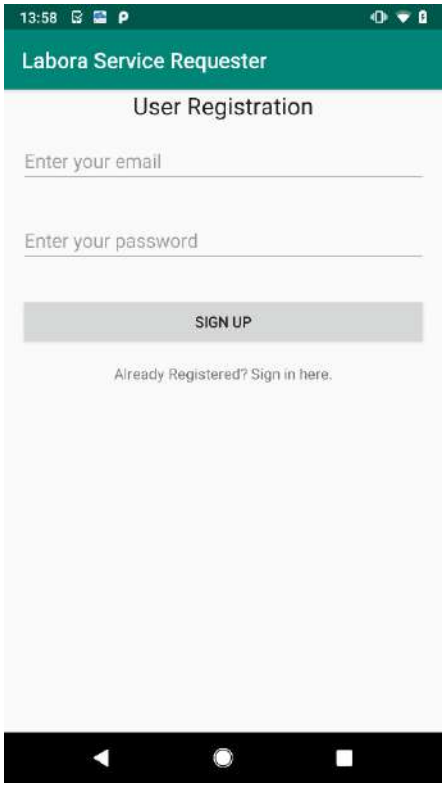
Test	Input	Status (Pass/Fail)
Phone	02034897488	✔
Phone	hello	✗
	if(TextUtils.*isEmpty*(phone))	
Phone	999	✔

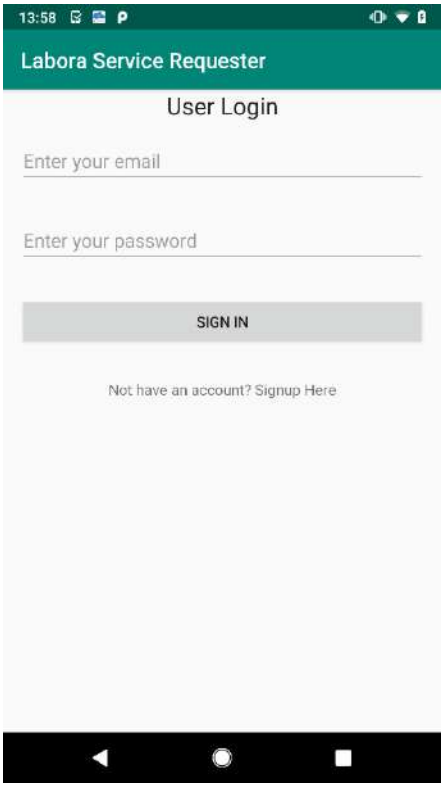
Test	Input	Status (Pass/Fail)
Occupation	Choose a service	✗
	if(TextUtils.*isEmpty*(occupation))	
Occupation	Plumber	✔
Occupation	Yoga Instructor	✔

Test	Input	Status (Pass/Fail)
Address	SE14	✔
Address	EN3	✔
Address	“” (Empty Field)	✗
	if(TextUtils.*isEmpty*(address))	

Robo (Automation) Testing:







Deployment Plan

Deployment usually involves defining what sequences of operations/steps (deployment plan), must be taken to deliver changes into a target system environment. Over the course of creating our application, we have stuck to a structured plan. This plan consisted of separately creating our front end and backend, week by week, gradually merging them. We used the Android Studio (Java) as the front-end coding environment and Firebase (Firestore) as our database.

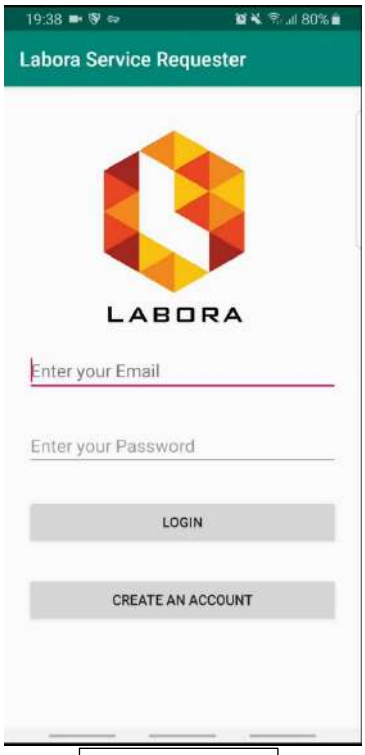
When considering our initial planning for the technical architecture, we research and chose relevant languages, that would allow us to be scalable in future. For example, we opted to use MongoDB, rather than MySQL. This is due to the reason that MongoDB handles unstructured data better and scales much more efficiently, given a larger scale (users).When analysing our application and its current architectural state for larger scope deployment, we believe we have chosen features (e.g. programming languages like Java and backend databases like Firebase), which will readily fulfil the requirements.

Our application is a mobile application. Also, because it is an android application, it is likely to be placed on the play store. This is due to the fact our application is solely based on android (java) code. As a result, this means less will need to be altered, front-end wise, as android studio-built apps scale quite well. However, we do realise that android studio/java has its limitations. Therefore, we have determined that a mobile application front-end developed with HTML/CSS/JS would be a viable second option going forward. Overall, a java developed front end will ensure fewer security leaks, as well as bring a modern approach to application development.

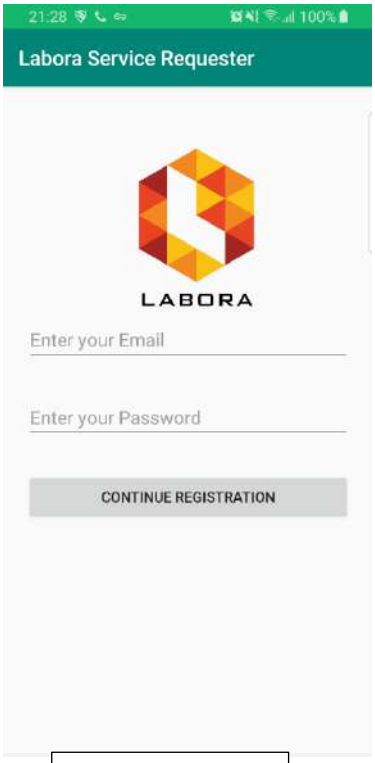
Furthermore, our backend database is currently firebase. Firebase has different sections for authentication and database management. It too is very scalable and can handle millions of users at a time. In addition, firebases' friendly interface and security permission settings, make it perfect for large scope development. However, due to firebases restricted querying abilities, it we did want a more complex database, we could switch to MongoDB. Not only Is it very scalable, but you can carry out complex queries with it. Therefore enabling an efficient and viable option as a backend database.

Lastly, we made use of DevOps. DevOps involves improving current software by continuous feedback. We aim to do this by making sure to keep in contact with our users throughout. Once we add a new feature, we will test it again.

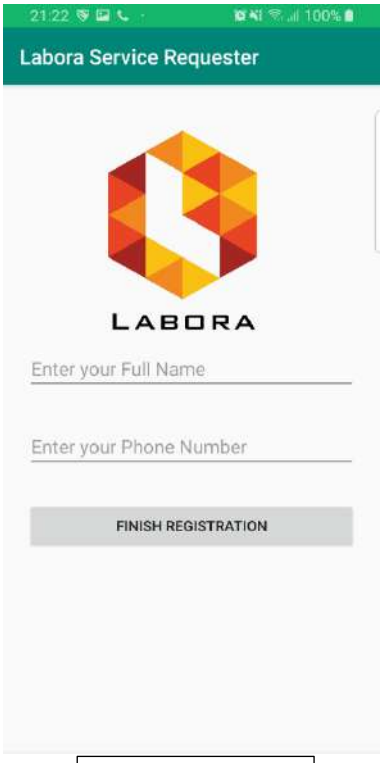
Service Requester App Images



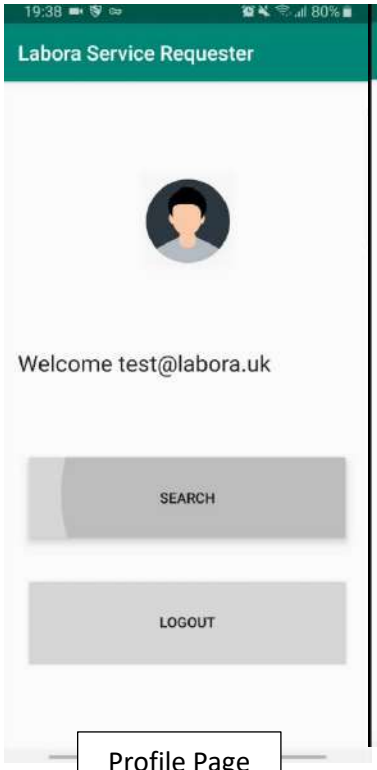
Login Page



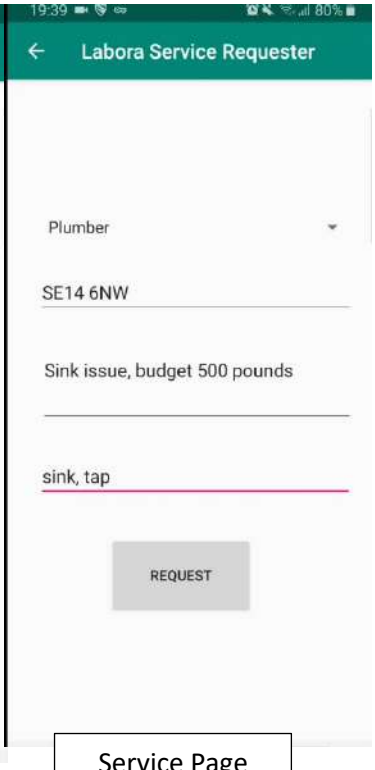
Register Page 1



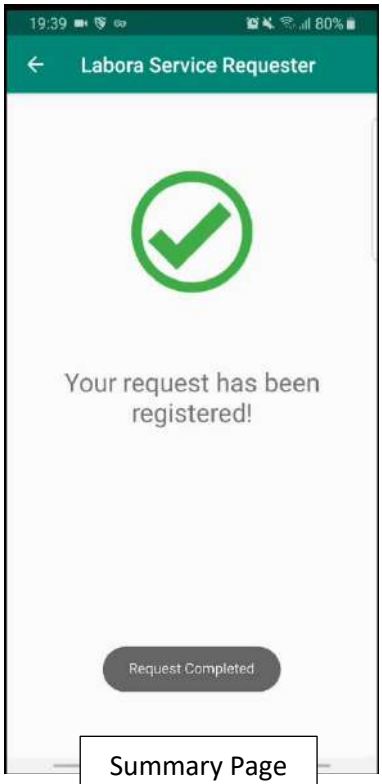
Register Page 2



Profile Page (Main Menu)

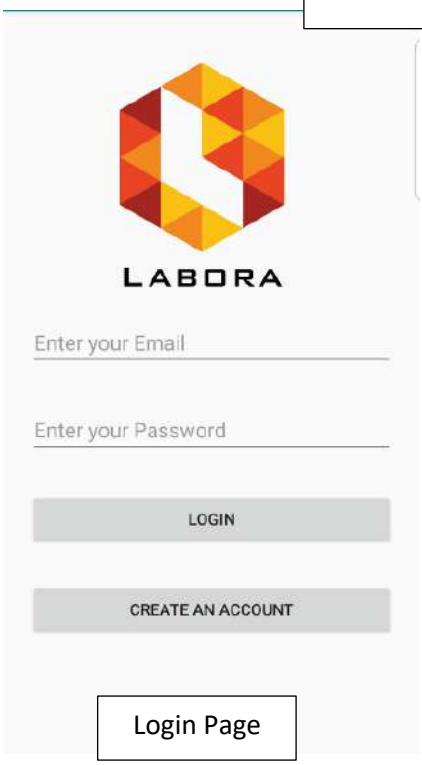


Service Page

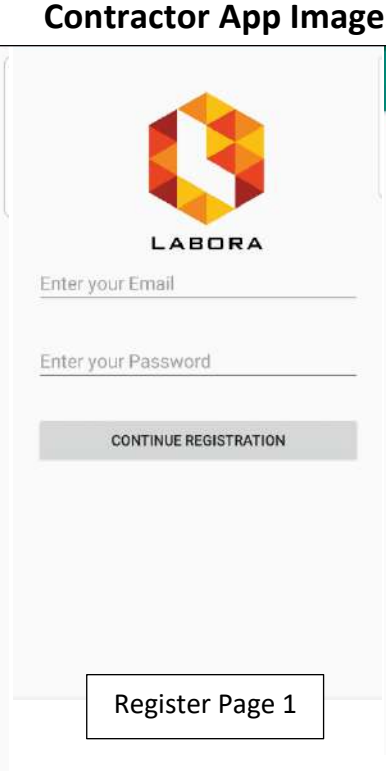


Summary Page (Done)

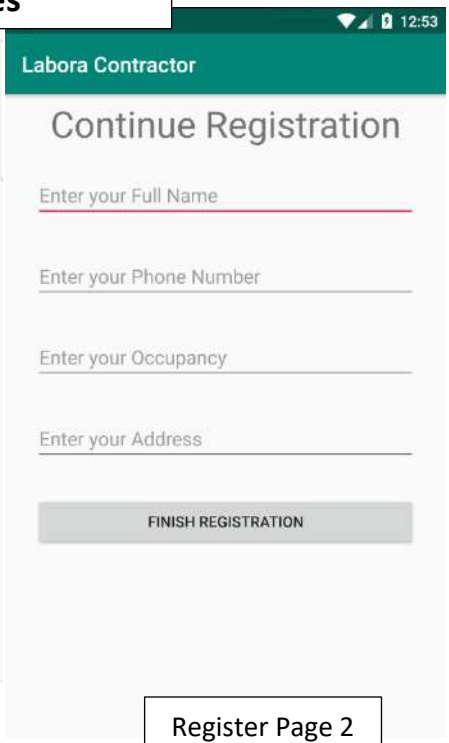
Contractor App Images



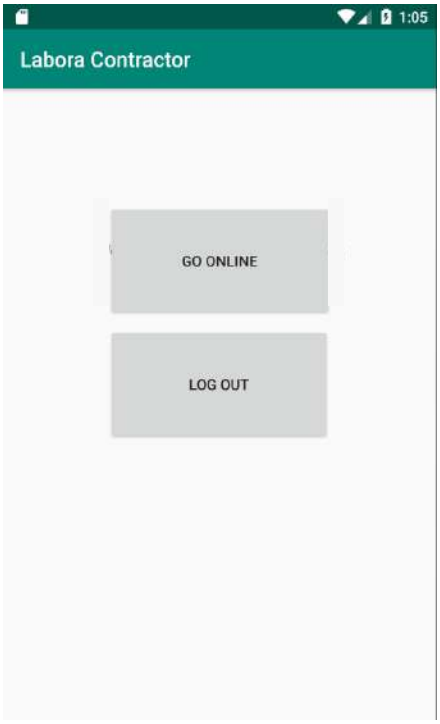
Login Page



Register Page 1



Register Page 2



Profile Page

(Main Menu)

Map Page

Summary Page

10. Bibliography

Ashlock, P., 2014. *Quora.* [Online]
Available at: https://www.quora.com/Why-do-Uber-drivers-have-a-separate-app-from-the-passenger-app
[Accessed 25 April 2019].

CollabNetVersionOne, 2019. *CollabNetVersionOne.* [Online]
Available at: https://resources.collab.net/agile-101/what-is-kanban
[Accessed 25 April 2019].

DK, P., 2017. *Youtube.* [Online]
Available at: https://www.youtube.com/watch?v=lF5m4o_CuNg
[Accessed 25 April 2019].

Firebase, 2019. *Firebase.* [Online]
Available at: https://firebase.google.com/docs/database/
[Accessed 25 April 2019].

Foundation, I. D., 2019. *Fitts' Law.* [Online]
Available at: https://www.interaction-design.org/literature/topics/fitts-law
[Accessed 25 April 2019].

Gitlab, 2019. *Gitlab.* [Online]
Available at: https://about.gitlab.com/
[Accessed 25 April 2019].

Google, 2019. *Firebase Documentation.* [Online]
Available at: https://firebase.google.com/docs/auth/
[Accessed 25 April 2019].

Google, 2019. *Firebase Test Lab.* [Online]
Available at: https://firebase.google.com/docs/test-lab/
[Accessed 25 April 2019].

Google, 2019. *Firebase Test Lab Robo Testing.* [Online]
Available at: https://firebase.google.com/docs/test-lab/android/robo-ux-test
[Accessed 25 April 2019].

Google, 2019. *Google Maps Platform.* [Online]
Available at: https://developers.google.com/maps/documentation/android-sdk/start
[Accessed 25 April 2019].

Gremillion, A. S., 2019. *99designs.* [Online]
Available at: https://99designs.co.uk/blog/tips/how-color-impacts-emotions-and-behaviors/
[Accessed 25 April 2019].

IPSE, 2019. *Consultancy.* [Online]
Available at: https://www.consultancy.uk/news/18474/uk-has-2-million-freelancers-and-the-number-will-continue-to-rise
[Accessed 25 April 2019].

Paradigm, V., 2019. *Visual Paradigm.* [Online]
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
[Accessed 25 April 2019].

Raphael, J., 2019. *Computer World.* [Online]
Available at: https://www.computerworld.com/article/3230909/chrome-remote-desktop-access-remote-computer-easily.html
[Accessed 25 April 2019].

Rouse, M., 2019. *SearchMobileComputing.* [Online]
Available at: https://searchmobilecomputing.techtarget.com/definition/Android-Studio
[Accessed 25 April 2019].

Sridhar, A., 2018. *An introduction to Git.* [Online]
Available at: https://medium.freecodecamp.org/what-is-git-and-how-to-use-it-c341b049ae61
[Accessed 25 April 2019].

Stackify, 2019. *Stackify.* [Online]
Available at: https://stackify.com/agile-methodology/#wpautbox_about
[Accessed 25 April 2019].

Summerson, C., 2018. *What's the latest version of Android?.* [Online]
Available at: https://www.howtogeek.com/345250/whats-the-latest-version-of-android/
[Accessed 25 April 2019].

Trello, 2019. *Trello.* [Online]
Available at: https://trello.com/en-GB
[Accessed 25 April 2019].

Vogella, 2016. *Building Android Applications With Gradle - Tutorial.* [Online]
Available at: https://www.vogella.com/tutorials/AndroidBuild/article.html
[Accessed 25 April 2019].

WhatsApp, 2019. *WhatsApp.* [Online]
Available at: https://web.whatsapp.com/
[Accessed 25 April 2019].

All images used in this report are from https://www.pexels.com/royalty-free-images/ and they are copyright free.