MSc Project: Mobile Hairdresser Application

Joshua Robertson

"A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science by advanced study in Computer Science in the Faculty of Engineering."

School of Computer Science, Electrical and Electronic Engineering, and Engineering Maths (SCEEM)

Abstract

The COVID pandemic has

Acknowledgements

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

SIGNED: Joshua Robertson DATE: 14.09.2021

Contents

List of Figures

1 Introduction

The COVID-19 pandemic has had a ubiquitous impact on our lives, from work, to travel, to how we socialise. Some of these changes are temporary, such as mask wearing, but others, such as the adoption of more digital services and flexible homeworking seem likely to be permanent (ADD CITE). Subsequently, the restriction of movement has caused consumers to migrate to online services at an unprecedented rate. With the global home services market expected to grow by almost 20% per year until 2026 [20] there exists a wealth of opportunity for companies to capitalise through digitalising previously physical services. One of these services is hairdressing, which saw a sizeable uptake in demand throughout the pandemic.

There already exists a range of applications suited towards providing home haircuts. For example, Shortcut [12], TRIM-IT[26] and TrimCheck[27] all provide bookable home haircuts. Despite this, none of the aforementioned applications follow the "uber" model, of allowing for immediate booking and delivery of home haircuts.

This application will therefore aim to facilitate this, along with exploring any other market gaps through early research and will aim to achieve this through the following objectives:

- Conduct research to elucidate any market gaps
- Through a user centric methodology design (UCD) and prototype the user interface of the application
- Create a minimum viable product (MVP) using new product development

The research aspect of the project will aim to analyse the strength and weaknesses of existing applications within the field, therefore exposing any market gaps that may be present.

For the implementation of the application, a heavy focus on the end user was taken through using a UCD methodology, along with NPD, which refers to the entirety of processes leading to bringing a product to market and encompasses several steps as seen in figure 1 below and discussed throughout the proceeding sections.

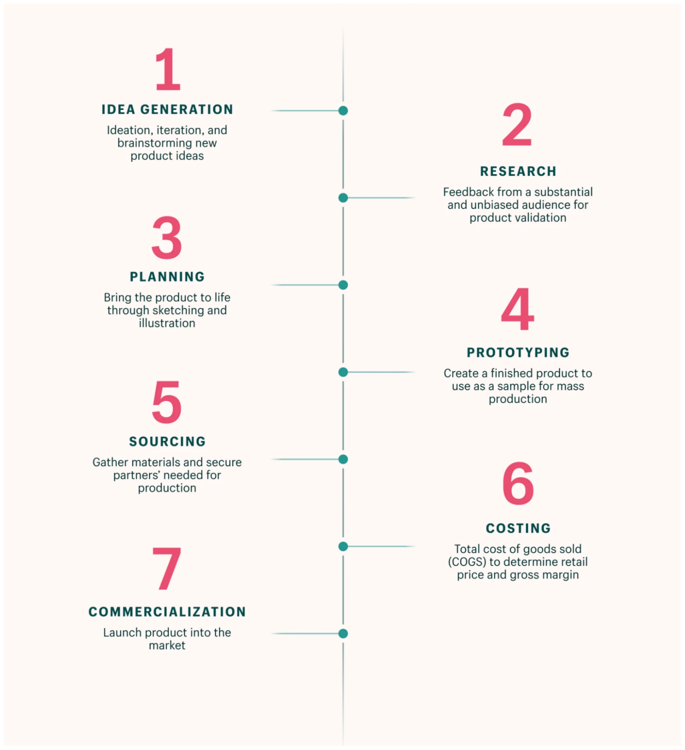

1
IDEA GENERATION
Ideation, iteration, and brainstorming new product ideas
2
RESEARCH
Feedback from a substantial and unbiased audience for product validation
3
PLANNING
Bring the product to life through sketching and illustration
4
PROTOTYPING
Create a finished product to use as a sample for mass production
5
SOURCING
Gather materials and secure partners' needed for production
6
COSTING
Total cost of goods sold (COGS) to determine retail price and gross margin
7
COMMERCIALIZATION
Launch product into the market

Figure 1: The 7 Steps of New Product Development
[23]

1.1 Ideation and Concept

The first stage of NPD starts with conceptualisation of a product idea. For the application the initial concept came from an external partner, who proposed a mobile hairdresser application to capitalise on the increased need for home delivery services. This was further defined during several meetings carried out in the initial stages of the project.

1.2 Project Management

The project management for the application focused on two key aspects; a clear vision and scope, including a detailed project plan; and an execution phase, which utilized an agile methodology.

1.2.1 Vision and Scope

The end goal of the project was as a springboard to create a fully functional and profitable barber delivery application.

The scope of the project was..

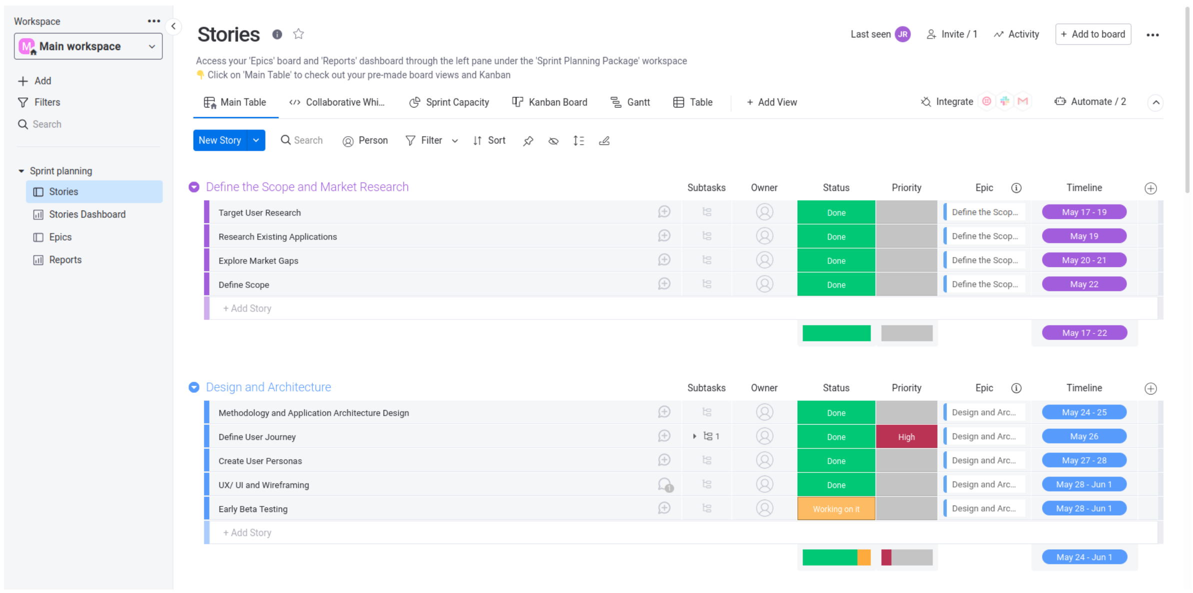
1.2.2 Agile Methodology

To carry out the project, an agile methodology was utilised, which allowed for short, iterative cycles of production, providing value in the form of quick creation and constant revision. Taking on an agile methodology also served the project well in the sense that there was a strong focus on prioritising value over comprehensive documentation and lengthy processes.

Although this approach is more commonly relevant to a team of developers, approaching the project management in this way allowed for a stringent and well defined timeline to be used, aiding in project delivery and outcome. This involved several key stages. Firstly, individual 'epics' were defined, which included:

- Define the Scope and Market Research
- Design and Architect the Application
- Setup and Create The Backend
- Write the Dissertation

These were then used to create 'stories' which were further split into individual tasks placed into a timeline and carried out in . For this, the project management tool monday.com [17] was used, which can be seen in figure 2. Using Monday.com allowed for a timeline to be easily created, along with updating the status of each story when relevant to follow the completion of the project. In this way, a change management approach (ADD CITE) could be carried out, in which the project could easily be updated and reflected in the tasks. This was especially important as both agile methodology and the UCD rely on constant feedback from the end user which informs the project.



Stories

Access your 'Epics' board and 'Reports' dashboard through the left pane under the 'Sprint Planning Package' workspace
Click on 'Main Table' to check out your pre-made board views and Kanban

Define the Scope and Market Research

	Subtasks	Owner	Status	Priority	Epic	Timeline
Target User Research			Done		Define the Scop...	May 17 - 19
Research Existing Applications			Done		Define the Scop...	May 19
Explore Market Gaps			Done		Define the Scop...	May 20 - 21
Define Scope			Done		Define the Scop...	May 22
+ Add Story						
						May 17 - 22

Design and Architecture

	Subtasks	Owner	Status	Priority	Epic	Timeline
Methodology and Application Architecture Design			Done		Design and Arc...	May 24 - 25
Define User Journey	1		Done	High	Design and Arc...	May 26
Create User Personas			Done		Design and Arc...	May 27 - 28
UX/ UI and Wireframing			Done		Design and Arc...	May 28 - Jun 1
Early Beta Testing			Working on it		Design and Arc...	May 28 - Jun 1
+ Add Story						
						May 24 - Jun 1

Figure 2: Screenshot of Monday.com Displaying the Project Stories [17]

Finally, using the created timeline, a gantt chart was implemented, which gave an overarching view of the project, with tasks performed represented along the vertical axis and the timeline represented along the horizontal axis. This can be found in the appendix (ADD CITE).

1.3 Research and Market Analysis

In order to gauge whether there is a market for the proposed analysis, a survey was carried out in which users were asked about whether they could see themselves using the application features, among other things. The full survey can be found within the appendix (ADD CITE)

1.3.1 Existing Applications

As previously discusses there exists a variety of similar applications, for which the most prominent will be discussed below, along with the salient and limiting features of each.
TODO: finish this section

Shortcut
Shortcut is a US based application.. One of the defining features of the shortcut app is the availability, with the app providing the ability to request a haircut from 8am to as late as midnight capatilising on a previously unventured late night hair cut market.

The application is limited on it's features, with it only having the option to request a Hair Cut only or a Hair Cut and Beard Trim. Another limiting feature of the application is the price. For a single haircut the cost starts at 75$ (around £54), which is most likely a reflection of high start up costs and is a problem seen in other similar applications, such as uber and lyft that can only be mitigated through losses ([28]).

TRIM-IT https://www.bbc.co.uk/news/stories-47711610
TRIM-IT is a UK based mobile hairdressor application. Their business model is franchise based similar to that seen by Mcdonald's, whereby barbers would invest through a monthly fee and be provided with the tools necessary, such as a mobile barber unit.
TrimCheck

1.3.2 Novelty of the Proposed Application

As discussed in the previous sections, there exists a range of applications that are suited towards providing a mobile barber service.

(shortcut)The ability to offer a variety of services not limited to just a haircut or beardtrim.

1.4 Deciding on a Platform

1.4.1 Mobile vs Desktop

An important consideration in NPD is determining which platform best suits the project, mobile or desktop. Here, we will discuss the merits and pitfalls of each, before concluding which is most relevant for the project.

Market Share
Consumers are now for the first time viewing web pages on mobile devices at a higher rate than on desktop, at 54.8%, compared to just 31.16% in early 2015 [16]. Further to this, over the last year desktop usage has dropped from

46.39% to 41.36%, whilst mobile phone usage has increased from 50.88% to 55.89%, following on a several year long trend [6] that reflects a saturated mobile market driving down the cost of phones. However, this analysis is slightly premature due to only being indicative of the world market, whereas the proposed application will only operate within the United Kingdom. When we analyse just the U.K. data (figure 3) we see that the results are not so conclusive, with only a 0.97% difference between the two in favour of Desktop. Therefore a decision based entirely on market share is unfavourable and other metrics must be explored.

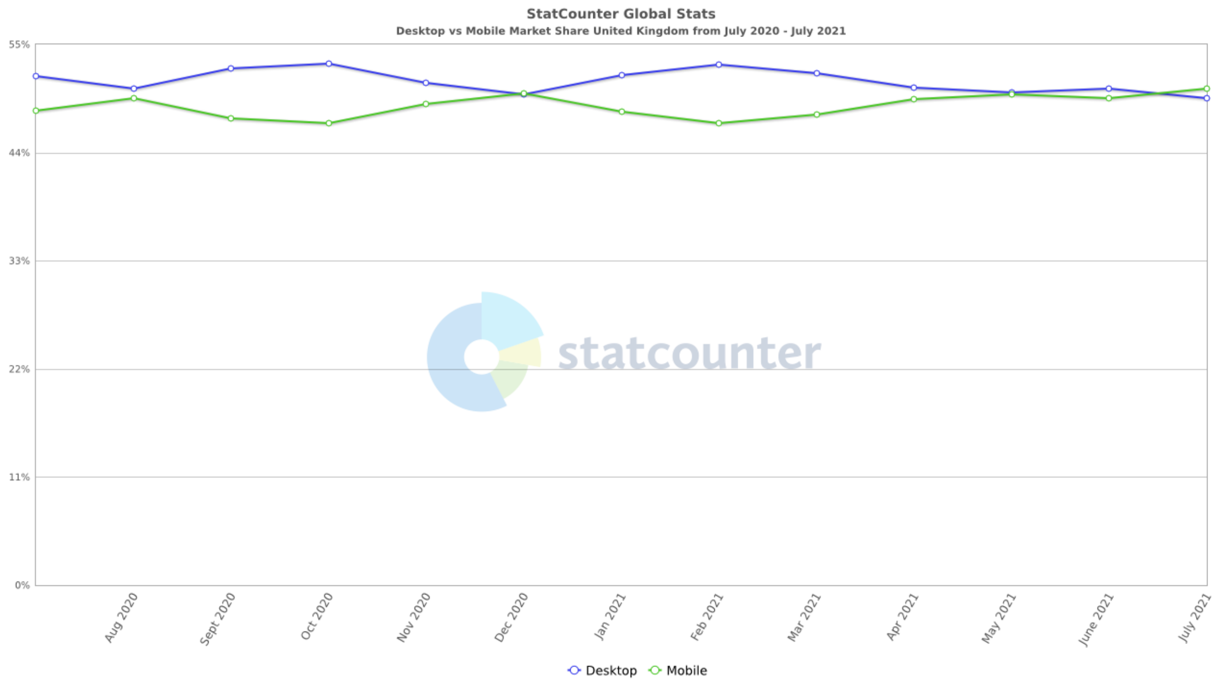


Figure 3: Desktop vs Mobile Market Share in the United Kingdom [5]

<u>Features and Performance</u>
Another metric to take into consideration is which features are required for the application and how this is reflected in mobile vs desktop applications. One the most pertinent features required is geolocation, which is much more suited to a mobile application due to inbuilt GPS. Another important consideration for the project is speed, for which mobiles perform actions much

faster than a website. Finally, as continuation of the project to market is expected, speed of creation is essential, for which an application is better suited.

1.4.2 Mobile Platform: Android vs iOS

An important consideration when creating a mobile application is deciding on which platform to choose. The two largest mobile providers currently are android and apple (iOS). Historically, iOS has dominated the market share, with a 42.02% market share in January 2011 compared to Androids 12.42% (figure 4). Despite this, in recent years android OS has become more popular, even holding a greater share several times over the last few years and currently trails by only around 2%.

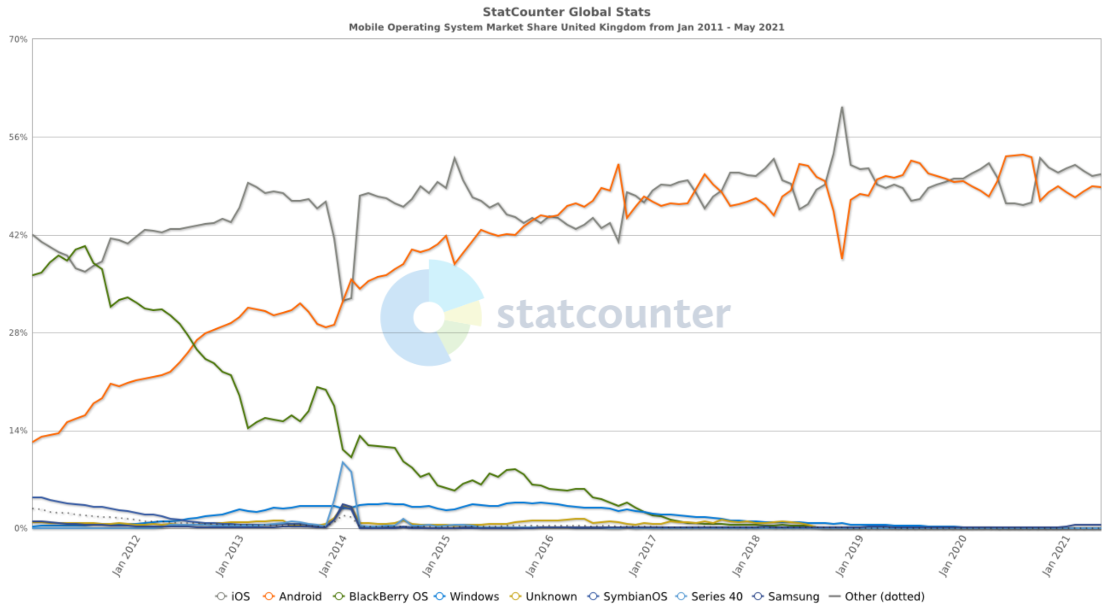


Figure 4: iOS vs Android Market Share Over The Last 10 Years [1]

With this change has brought with it a push towards frameworks that allow for development across multiple platforms, such as React Native ([21]) and Flutter ([10]). For this reason, it was decided that a cross platform framework would be used, which is further discussed below.

1.5 Frontend: Programming Language

When deciding on the programming software, several metrics were taken into consideration, including cross-platform functionality, speed, speed of development and performance. For this reason, Dart and the corresponding Flutter software development kit (SDK) were chosen for the primary software. Flutter is a cross-platform development kit, meaning that it will natively run on both iOS and android applications created by Google [10]. Dart is compiled ahead-of-time into native ARM code giving better performance compared to other similar development kits, such as React Native and the user interface is implemented within a fast, low-level C++ library giving great speed to the application. Dart has also seen a large increase in usage within recent years, jumping up 532% from 2018 to 2019 [25] meaning that there is now an extensible list of third-party plugins available and a large community.

1.6 Backend: SQL vs noSQL database

For the database, it was decided to use Google Firestore ([4]), a noSQL database that relies on nested 'documents' within 'collections'. This was chosen for several reasons. Firstly, as the chosen language 'Dart' is run by Google, using firestore allows for greater integration and congruence with the platform and APIs. Firestore also allows for rapid scalability, along with using Googles excellent cloud platform. The cloud firestore also integrates well with Firebases Authentication service, which is used throughout and discussed extensively in section 4

Another important feature of noSQL databases is the ability to easily modify the internal data in response to changing business requirements, in an interactive way that allows fordo you use relationship data in firebase stackoverflow modification throughout the application lifestyle and therefore easy scaling.

1.6.1 The Target User

The planned application targets any user who wishes to get a haircut from their home. TODO: finish target user section

1.7 User Personas

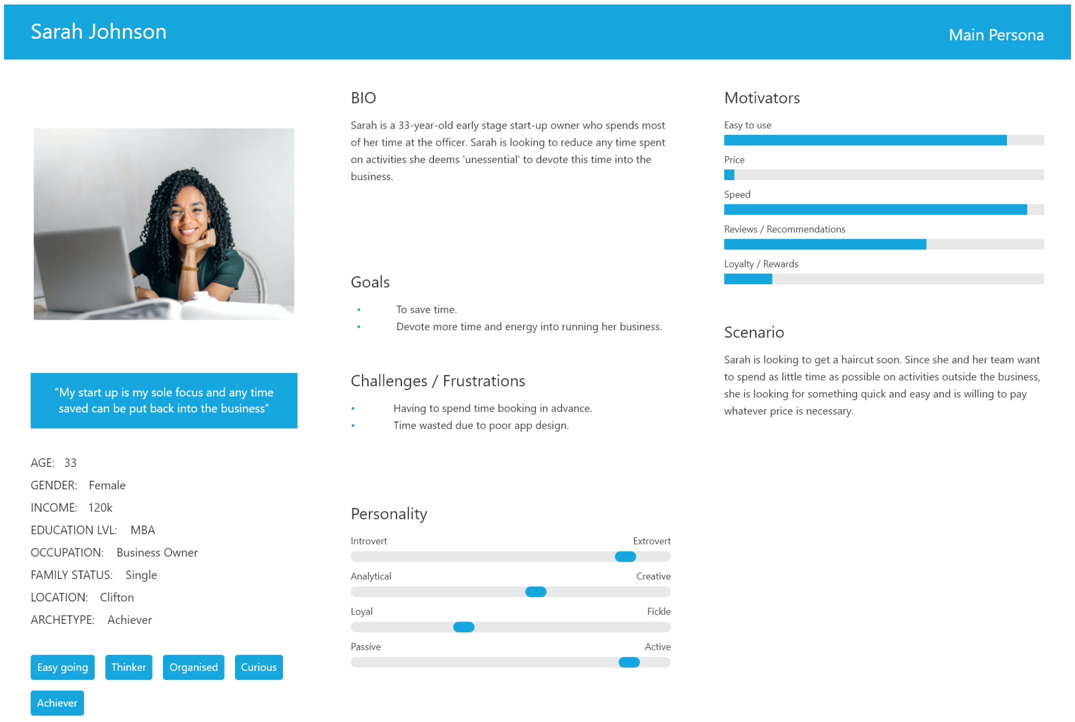
The creation of user personas representing fictitious, archetypal users is an essential part of application development [18] and allows a deep understanding of the target user to be sought and implemented within the features and design of the application [2]. There are, however, some shortcomings to qualitative persona generation, such as validity concerns and user bias [3] and although they are addressed by other methods, such as data-driven personas [15], these require a broad user base and therefore we have decided to stick with qualitative methods, which allow for enough brevity and depth for the scope of the project.

Here 3 user personas were created, which are discussed in detail below.

- Persona 1:

  Name: Sarah Johnson

  Profile:



Sarah Johnson

Main Persona

"My start up is my sole focus and any time saved can be put back into the business"

AGE: 33
GENDER: Female
INCOME: 120k
EDUCATION LVL: MBA
OCCUPATION: Business Owner
FAMILY STATUS: Single
LOCATION: Clifton
ARCHETYPE: Achiever

Easy going · Thinker · Organised · Curious · Achiever

BIO

Sarah is a 33-year-old early stage start-up owner who spends most of her time at the officer. Sarah is looking to reduce any time spent on activities she deems 'unessential' to devote this time into the business.

Goals

- To save time.
- Devote more time and energy into running her business.

Challenges / Frustrations

- Having to spend time booking in advance.
- Time wasted due to poor app design.

Personality

Introvert – Extrovert
Analytical – Creative
Loyal – Fickle
Passive – Active

Motivators

Easy to use
Price
Speed
Reviews / Recommendations
Loyalty / Rewards

Scenario

Sarah is looking to get a haircut soon. Since she and her team want to spend as little time as possible on activities outside the business, she is looking for something quick and easy and is willing to pay whatever price is necessary.

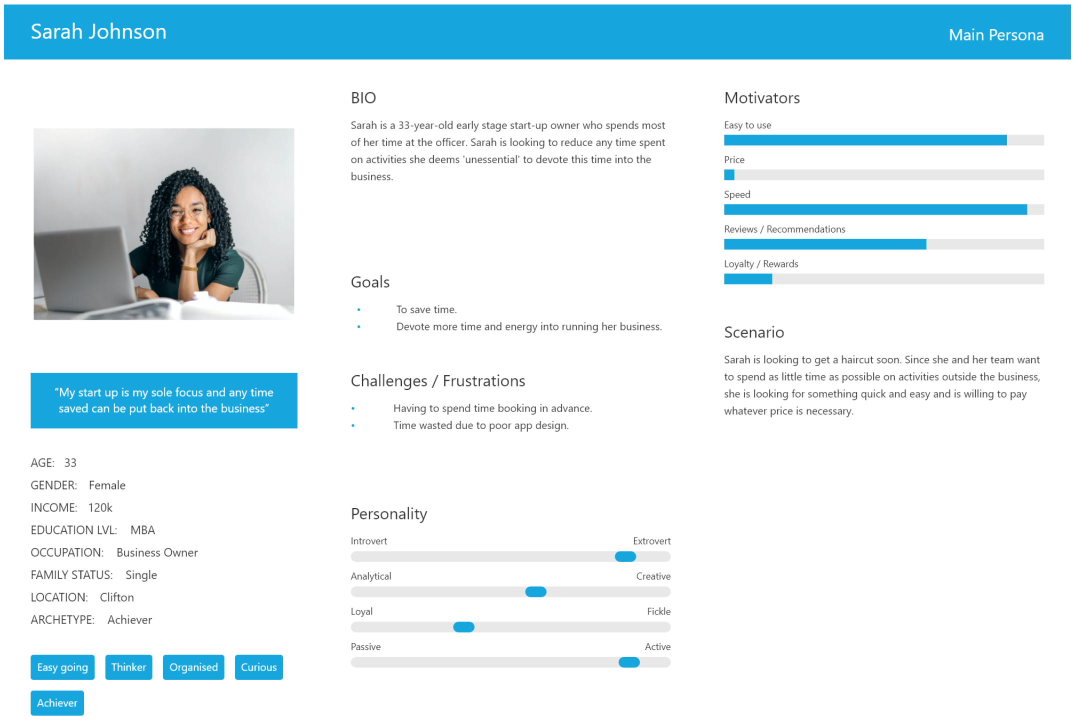
Figure 5: Persona 1

(TODO: personas share same name)

- Persona 2:

  Name: Claire Sheppard

  Profile:



Sarah Johnson

Main Persona

"My start up is my sole focus and any time saved can be put back into the business"

AGE: 33
GENDER: Female
INCOME: 120k
EDUCATION LVL: MBA
OCCUPATION: Business Owner
FAMILY STATUS: Single
LOCATION: Clifton
ARCHETYPE: Achiever

Easy going · Thinker · Organised · Curious · Achiever

BIO

Sarah is a 33-year-old early stage start-up owner who spends most of her time at the officer. Sarah is looking to reduce any time spent on activities she deems 'unessential' to devote this time into the business.

Goals

- To save time.
- Devote more time and energy into running her business.

Challenges / Frustrations

- Having to spend time booking in advance.
- Time wasted due to poor app design.

Personality

Motivators

Scenario

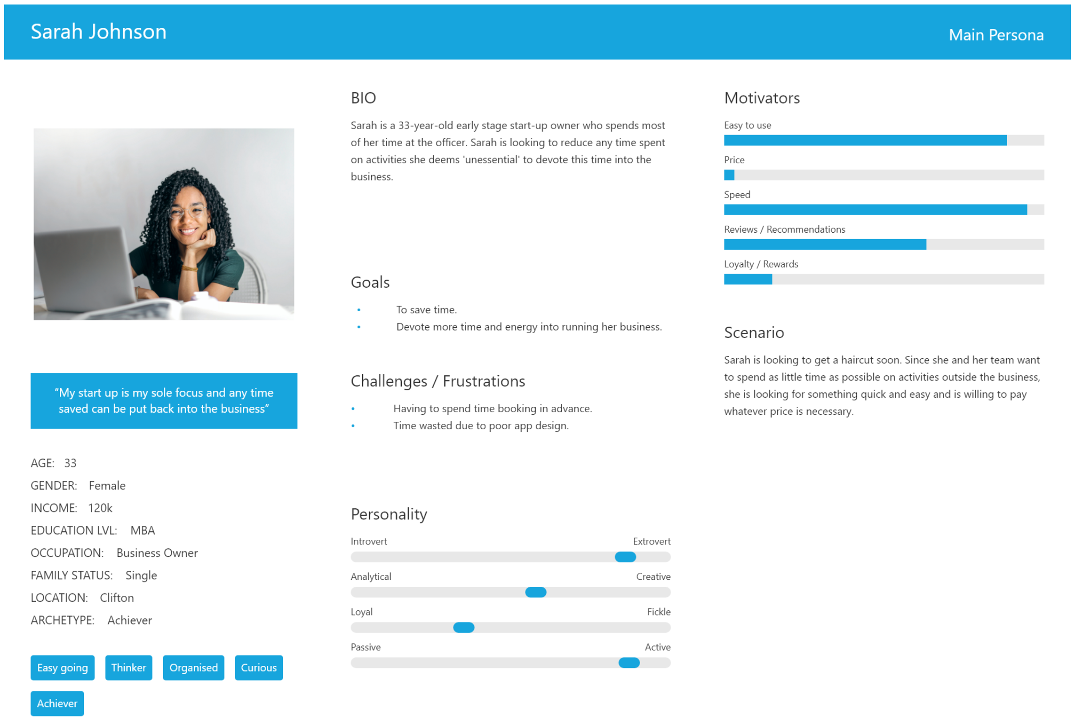
Sarah is looking to get a haircut soon. Since she and her team want to spend as little time as possible on activities outside the business, she is looking for something quick and easy and is willing to pay whatever price is necessary.

Figure 6: Persona 2

- Persona 3:

  Name: Emma Bradford

  Profile:



Sarah Johnson

Main Persona

"My start up is my sole focus and any time saved can be put back into the business"

AGE: 33
GENDER: Female
INCOME: 120k
EDUCATION LVL: MBA
OCCUPATION: Business Owner
FAMILY STATUS: Single
LOCATION: Clifton
ARCHETYPE: Achiever

Easy going | Thinker | Organised | Curious | Achiever

BIO

Sarah is a 33-year-old early stage start-up owner who spends most of her time at the officer. Sarah is looking to reduce any time spent on activities she deems 'unessential' to devote this time into the business.

Goals

- To save time.
- Devote more time and energy into running her business.

Challenges / Frustrations

- Having to spend time booking in advance.
- Time wasted due to poor app design.

Personality

Motivators

Scenario

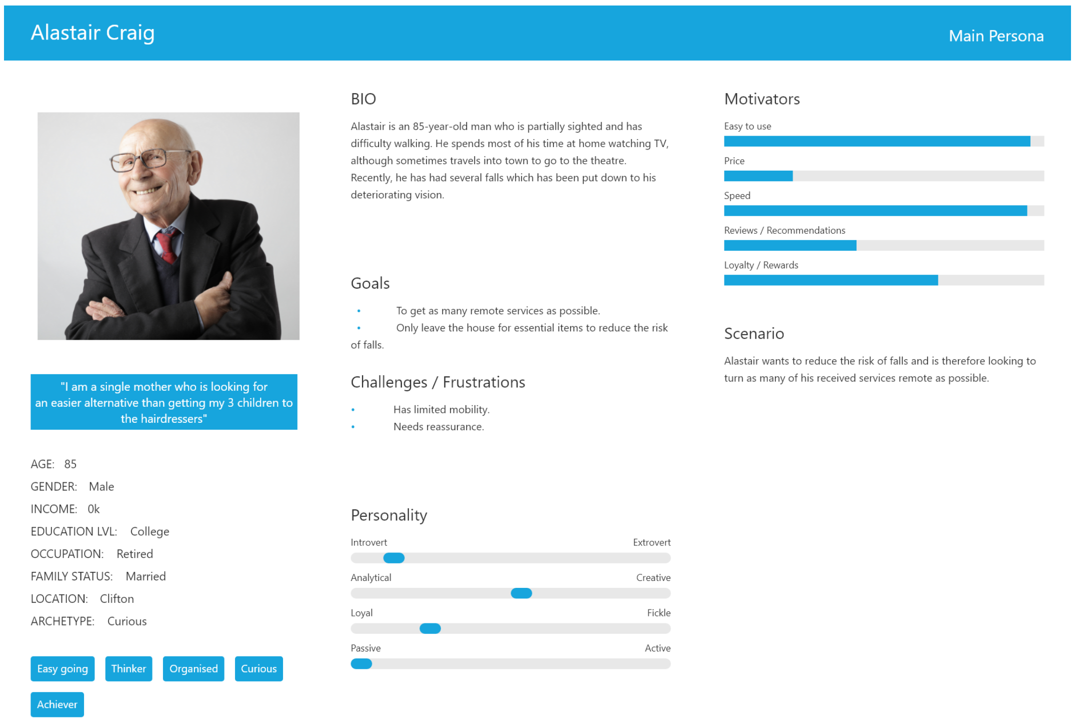
Sarah is looking to get a haircut soon. Since she and her team want to spend as little time as possible on activities outside the business, she is looking for something quick and easy and is willing to pay whatever price is necessary.

Figure 7: Persona 3

- Persona 4:

  Name: Claire Sheppard

  Profile:



Alastair Craig

Main Persona

"I am a single mother who is looking for an easier alternative than getting my 3 children to the hairdressers"

AGE: 85
GENDER: Male
INCOME: 0k
EDUCATION LVL: College
OCCUPATION: Retired
FAMILY STATUS: Married
LOCATION: Clifton
ARCHETYPE: Curious

Easy going
Thinker
Organised
Curious
Achiever

BIO

Alastair is an 85-year-old man who is partially sighted and has difficulty walking. He spends most of his time at home watching TV, although sometimes travels into town to go to the theatre. Recently, he has had several falls which has been put down to his deteriorating vision.

Goals

- To get as many remote services as possible.
- Only leave the house for essential items to reduce the risk of falls.

Challenges / Frustrations

- Has limited mobility.
- Needs reassurance.

Scenario

Alastair wants to reduce the risk of falls and is therefore looking to turn as many of his received services remote as possible.

Figure 8: Persona 4

2 Software Requirements

Here we discuss and outline the software requirements, for which this section is similar to that found within a software requirements specification (SRS) document and lays the framework for the entire project. Here we discuss the application scope, user needs, functional and non-functional requirements, along with use cases.

2.1 Application Scope

The scope of the project was to create a fully working and functional barber application with several features, which are discussed in the User Needs below. To further aid in scoping the project, epics were created, which were further split into stories that could be carried out. Although it could be argued that this type of agile methodology is more relevant when working within a team of developers, it helped to determine a stringent workflow and timeline and aided in project delivery. The scope was then further defined when wire-framing in Adobe XD, which allowed for the first tangible design to be made.

2.2 User Needs

- Allow the application to run on a mobile device.
- Allow the user to book a beauty treatment to receive at their home address.
- Allow a barber to set up an account and specify their product details.

2.3 Requirements

2.3.1 Functional Requirements

- Customer-side Application
  - The application shall allow a customer to create an account and login
  - The application shall provide the user with basic account management capabilities
  - The application shall allow for the user to pick from a range of relevant products and add them to their cart
  - The application shall allow the user full management of their shopping cart
  - The application shall provide only geographically relevant barbers and products to the user
  - The application shall allow the user to view their past orders
- Barber-side Application
  - The application shall allow a parent barber to create an account and login
  - The application shall allow for integrated back-end management of it's barbers and products
  - The application shall allow for the parent barber to view its orders

2.3.2 Non-Functional Requirements

- Performance
  - The application shall take no longer than 3 seconds to load the users home screen
- Data
  - The application shall cache data where possible
  - The application shall minimise calls to the database and make them only when relevant
- Use-ability
  - The application shall follow nielsen's usability heuristics and be easily usable for the user without any guidance or help
- Security
  - The application shall ensure that all app data be secured and encrypted
  - The application should use OAuth for access delegation
- Operating System
  - The application shall run on both iOS and android devices
  - The application shall run on all devices newer than android 5.0 (API 21) - around 94.1% of android devices
  - The application shall run on all devices newer than iOS 9.0 - around 99.6% of iOS devices

2.4 Use Cases

Here the use cases are presented, which are described by Ivar Jacobson as "a description of a set of sequences of actions and variants that a system performs that yield an observable result of value to an actor." (Jacobson, et. al., 1999, p.41). Use cases are useful in the sense that they provide a structure for collecting customer requirements and setting the scope [13]. They also allow for validation of the project through post-production testing, which can be seen in section section 7.

To produce the use cases we reference both the user personas created in subsection 1.7 and the functional and non-functional requirements discussed in the previous section. Below lists the most pertinent use cases.

- Use Case 1 - Sign Up
- Use Case 2 - Login
- Use Case 3 - Book a Haircut
- Use Case 4 - Search for a Barber
- Use Case 5 - Checkout
- Use Case 6 - View Orders
- Use Case 7 - Sign Out
- Use Case 8 - Add a Barber
- Use Case 9 - Add a Product

The above use cases are further designed in Appendix A.

3 System Design

In this section, we discuss the structure of the project, the system and software architectures and data and state management given the previously specified requirements.

3.1 Domain Model

TODO: discuss domain model Domain driven design (DDD), which originated from a 2003 book by Eric Evans [8] depicts a paradigm whereby data is visualised as a lexicon of abstractions, with the data model clearly representing key data concepts and the relationships between them. Modelling in this way bridges the gap between activities and interests of the users with the software and allows.

3.1.1 Entities

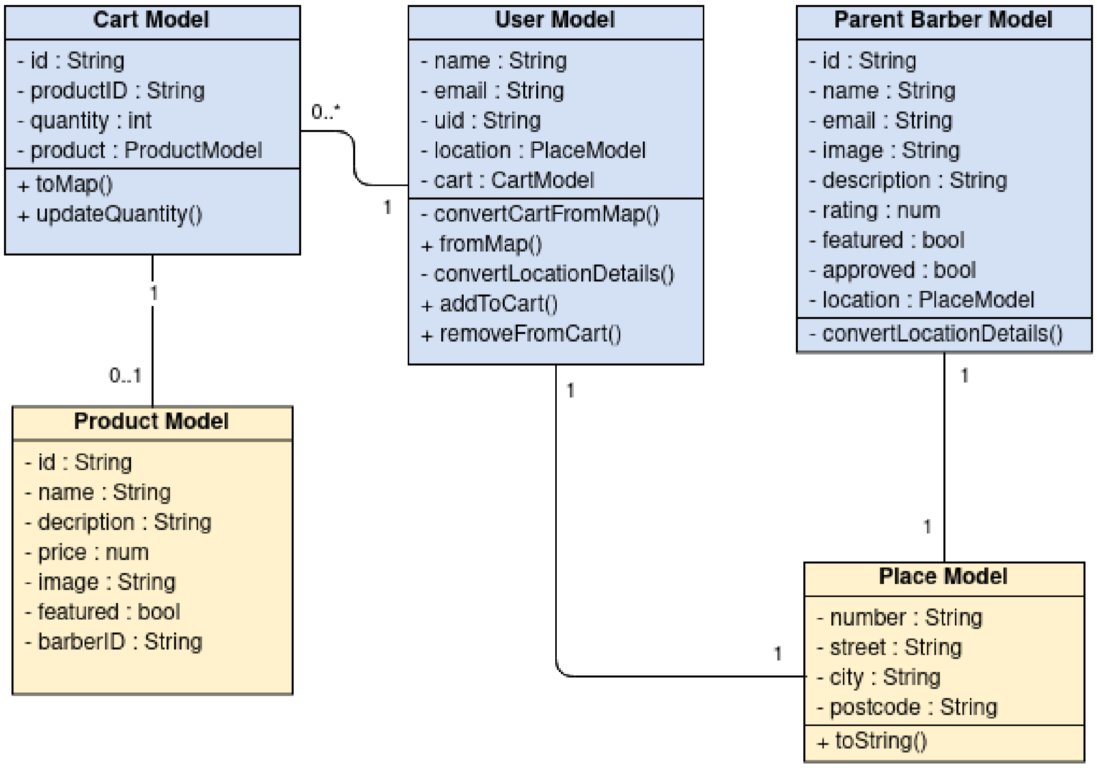


Figure 9: Model Class Diagrams for the Models

3.1.2 Entities

Within DDD it is recommended that a structured architecture is implemented, which is discussed in detail below.

TODO: continue with article and discuss how the project code was split using DDD, i.e. models, screens etc etc

3.2 System Architecture

https://www.raywenderlich.com/6373413-state-management-with-provider System Architecture can be broadly defined as a conceptual model which outlines the structure, behaviour and interactions between internal and external components of the system. Modelling and creating a structured and well-defined architecture allows for the development of a sustainable, scalable and stable software product which can easily grow relevant to the demands placed on it's features.

The overall system architecture can be seen below in figure 10.

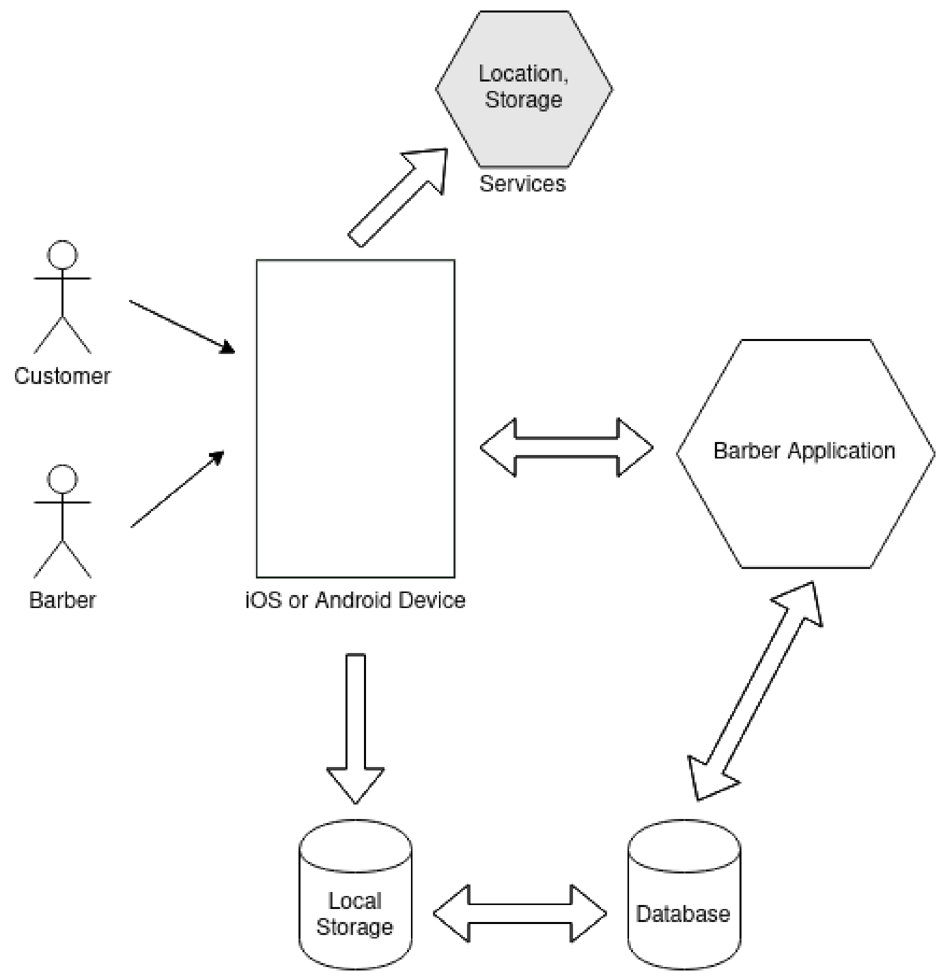


Figure 10: High-level System Architecture

When designing the architecture, core business logic was kept separate from the UI, database and network. For example, when interacting with the database the UI called upon the utilities package and any API calls where contained within the providers package. The project was also split into 3 layers, according to clean architecture principles ([14])

- Presentation layer
  - Screens - Contains unique UI elements
  - Widgets - Elements that are used to create the UI and feed into the screens
- Domain layer
  - Utilities - Contains business logic and elements that are used to make calls to the database or interact with any external APIs.
  - Providers - State management tools that are called on throughout the application.
- Data layer
  - Models - Contains the models used for local storage.

This generally follows clean architecture principles ([14]) and each layer is represented in figure 11 below.

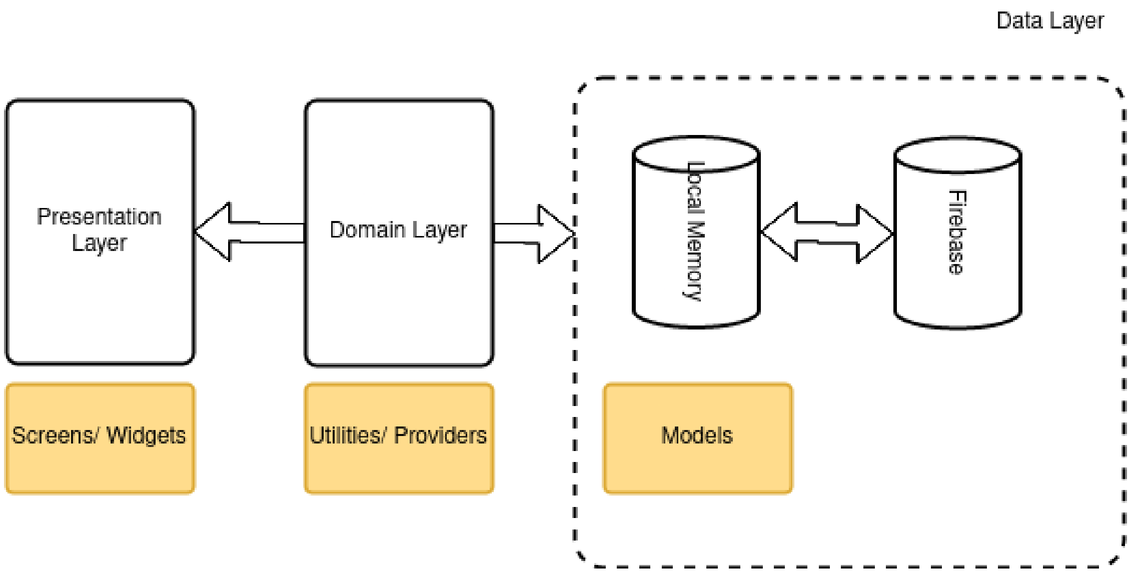


Figure 11: Clean Architecture (adapted from [14])

From this architecture and the previously devised specifications, an activity diagram was created, to detail the minimum required activities within the application, which can be seen in figure 12 below.

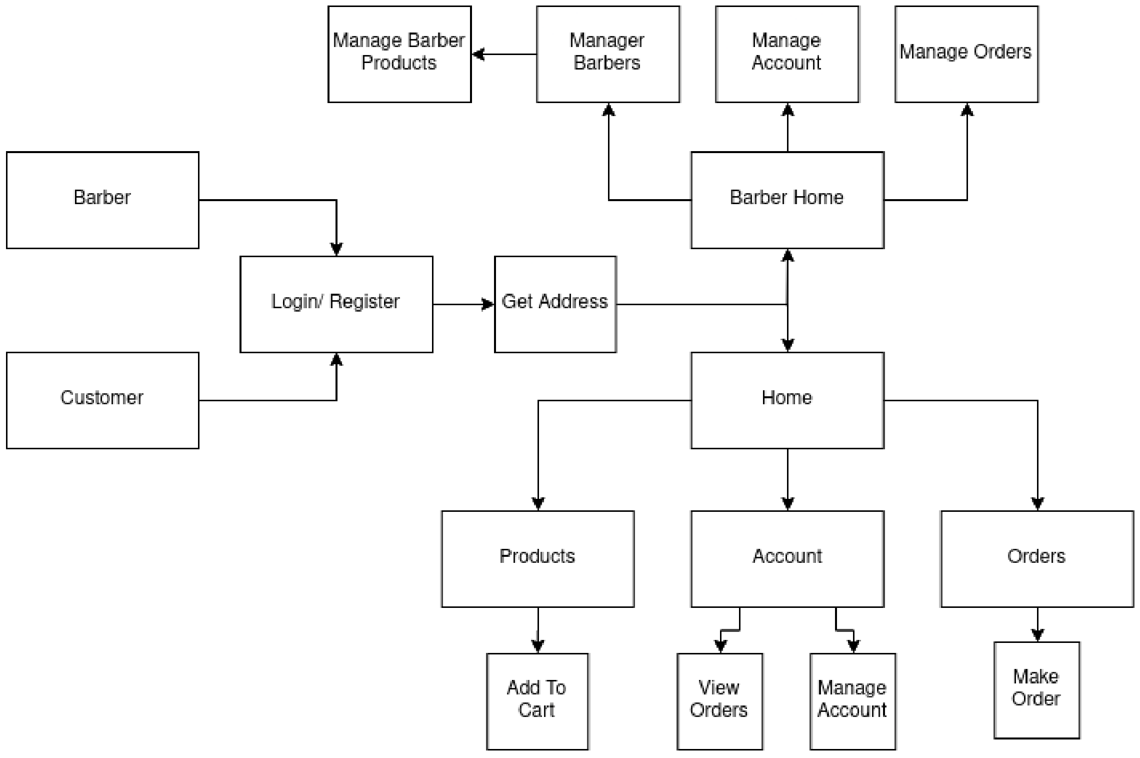


Figure 12: Application Minimal Activity Diagram

3.3 Client-side Specification

3.3.1 State Management

State management is an important feature within flutter. As flutter is declarative, rather than allow for changes in the widget or UI, each time a change is required the UI is rebuilt to reflect the applications current state. Within flutter there exists a variety of different methods for managing the state of the app, of which Provider, BLoC and GetX are some of the most popular and widely used. Here we will discuss the merits and pitfalls of each before settling on a framework for the application.

Provider

By far the most commonly used state management framework is Provider, which works by connecting all of the relevant daughter widgets to a value, so that data can be created, listened to and disposed of globally. To do this, a class is created that extends 'ChangeNotifier', which works by allowing classes to 'subscribe' to the senders data, through using the method 'notifyListeners()'

One negative of Provider is that it is only optimised for relatively few listeners, with it being $\mathcal{O}(n^2)$.

BLoC

DISCUSS BLOC

GetX

DISCUSS GETX

TODO: discuss different state management techniques and why provider was chosen Simple state management used (Provider) - involves: ChangeNotifier and Provider.of https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple

PARENTBARBERSPROVIDER provides to a lot of classes and the enumeration type AuthStatus is tracked... TODO: talk about main.dart

'Authentication extends ChangeNotifier'

Similar to - this https://codewithandrea.com/articles/simple-authentication-flow-with-flutter/

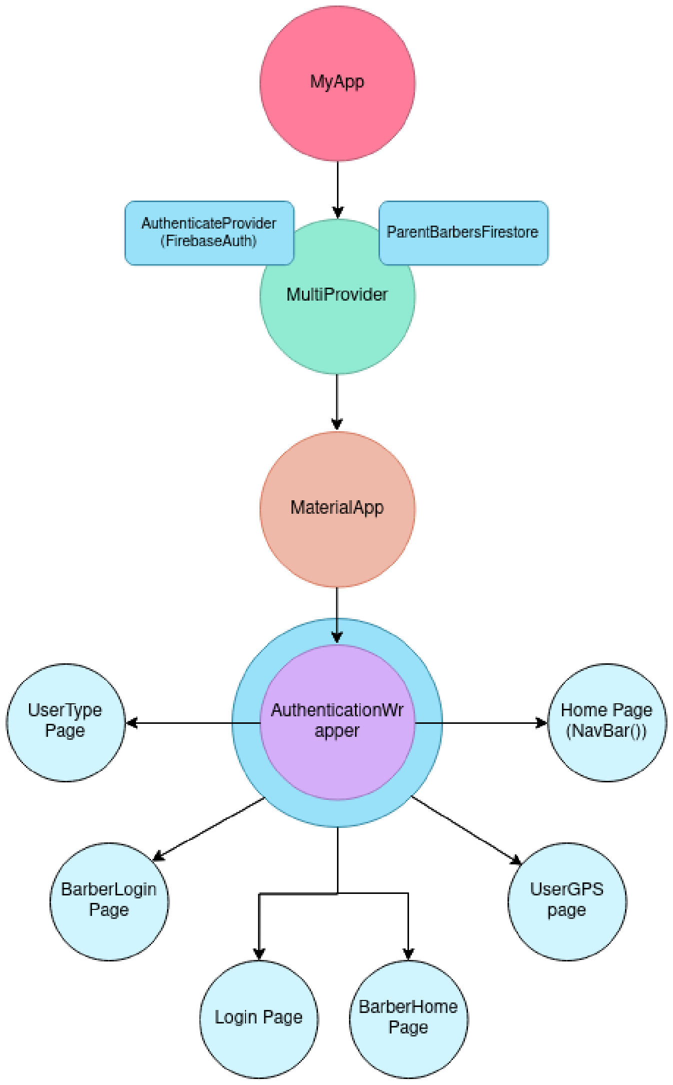


Figure 13: Widget Tree Diagram

3.3.2 Deciding on a Framework

There exists a variety of software architectural frameworks, although not exhaustive this includes Client-server, Peer-to-peer, Microservices and Model-view controller (MVC). For this project, we decided to go with MVC as the architectural framework, which, for brevity is the only discussed here. Used the screens as the view Used ChangeNotifier as the controller TODO: talk about MVC

4 Implementation

Here we discuss the process of wire-framing the application, along with the sprints, each of which pertains to relevant feature within the application. Finally, we address each use case individually.

5 Prototyping

An essential component of UCD and more generally UX design is prototyping, which involves making mock-ups of the application that act as early prototypes to influence later development [2]. Mobile app prototyping has many benefits to the project, including:

- Validates strategic design directions of the product
- Saves time by discovering any constraints early in the project
- Allows for early, interactive user testing
- Acts as a template for the UI during the implementation phase

The prototyping for the application involved two distinct stages; firstly, an initial sketch was done with pen and paper to discern the layout and overall themes associated with the app, before a wireframe was created to further define the prototype and allow for early testing.

5.1 Initial Sketches and Brainstorming

Initial sketches involve using pen and paper to elucidate problems and brain storm ideas for the project. During this phase, several design and UI features were considered, which allowed us to generate many ideas, using a fast and simple approach.

Some of these sketches can be seen in figure 14 below.

Figure 14: Pen and Paper Sketches and Brainstorming

(TODO: UPDATE PHOTO WITH INITIAL SKETCHES)

5.2 Wire-framing

Once the sketches were completed we moved on to wire-framing the application, which involved taking the best sketch variants and creating a more detailed, lower level prototype.For the wire-framing application Adobe XD was chosen for several reasons. Firstly, it has strong prototyping functionality, allowing the user to click around the application through the use of ‘components’. This interactivity means that early testers can get a real feel for how the application works. An illustration of this can be seen below, whereby each arrow represents a state change in the form of a trigger/ action pair, whereby for example a user could click on ‘Available Right Now’ and be taken to the ‘Checkout’ as seen in figure 15 below.

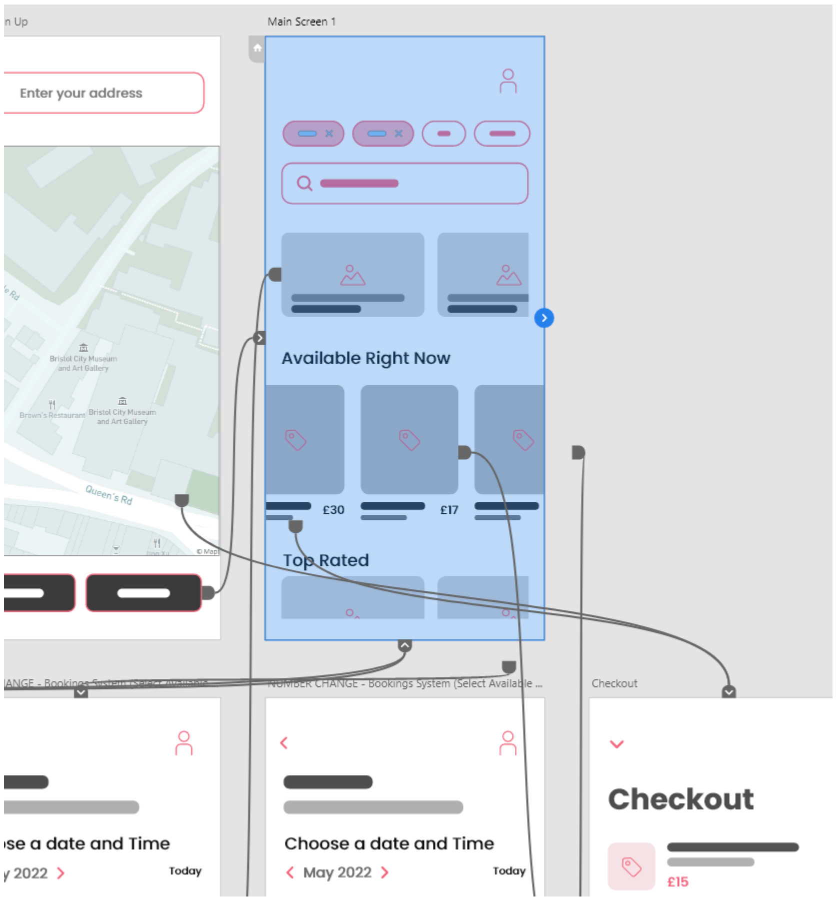


Figure 15: Component Interactivity within Adobe XD

Adobe XD also allows for easy distribution of the prototype in the form of a shareable link that opens in the browser and encompasses the same functionality and components that can be found within the application itself, meaning that anyone with access to a browser can test the prototype. Along with this, the prototype also allows for comments to be made, which are fed back to the owner. This comment capability was used early on during beta testing when it was sent out with the early questionnaire and influenced initial design decisions [24].

When designing the screens there was a strong focus on user experience following Nielsons 10 Heuristics for User Interface Design [9] . For example, the functionality was kept as minimal as possible to avoid clittering and avoid cognitive load on the user, the user was given control to go back and forward between previous screens to allow for user control and freedom and simple and self-explanatory language was used to apply recognition over recall. For example, the Sign In screen below extraneous text was kept to a minimum by using images for the login items, such as Google, Facebook and Twitter, a sign up button was included to allow the user to access the application through creating a new account and large, clear sign in forms and buttons were used. The full interactive Adobe XD wireframe can be found here.

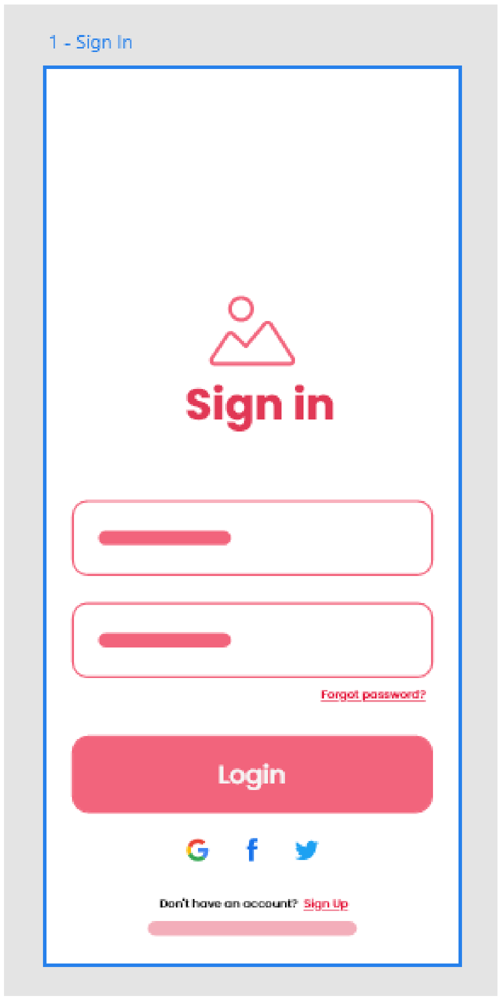


Figure 16: Sign In Page Made With Adobe XD

6 Implementation

6.1 Setup

Before beginning the development sprints, the first task involved setting up the developer environment. To code the application it was decided that an integrated development environment (IDE) was used due to the comprehensive features it provides to aid in development and maximise productivity. For this, it was decided that the IDE 'Android Studio' by JetBrains [7] was to be used, due to previous experience and familiarity with other JetBrains applications, along with Android Studios excellent built in features and seamless integration with flutter.

6.1.1 File Structure

Although there is no official recommendation for structuring the app, here we follow a commonly used structure which includes models; the files that serve as collections of data that are used in conjunction with the widgets to form the user interface of the application; providers; screens; utilities; and widgets; .

6.2 Sprint X - UI

Flutter offer an Adobe XD plugin to turn wireframes directly into code, however, this was not used for several reasons. In Adobe XD components are positioned absolutely, whereas in Flutter it is done relatively, leading to several issues with positioning. Adobe XD also does not contain customer properties and therefore mapping these to components, such as title is not possible.

(DISCUSS HOW FOLLOWED WIREFRAME AND COMPARE IMAGES TO REAL APPLICATION)

6.2.1 Cupertino vs Material

The final project was built using material due to..

6.3 Sprint X - Login and Sign Up

6.3.1 Authentication

Firebase has its own built in authentication (Firebase Authentication) which was used for the project due to several considerations -

- Excellent built in security features
- Integration with firebase database and storage
- Easy modification through Googles declarative language

Within the UserDatabase class, for the createNewUser function, we pass through the authentication uid, which is then used as the document id, so that future calls can refer to this and therefore fetch the document, without doing a call such as

```
    _firebaseFirestore.collection(collection).where
      ('uid', .isEqualTo('givenID')).get()
```

which does not scale well due to a search time of $\mathcal{O}(n)$. Instead, we store the uid as the document id, allowing us to do a similar, although quicker call such as

```
    _firebaseFirestore.collection(collection).doc(
      userId).get()
```

which gives a search time of $\mathcal{O}(1)$.

When authenticating through Google, Facebook and Twitter, a user is created so that data can be persistently stored alongside the credentials.

6.4 Sprint X - Backend Design

6.4.1 Database Design

As previously discussed, for the project it was decided to use a noSQL database, instead of a relational one. Despite this, it was decided that a database schema should be constructed to constitute the parent barbers, barbers and products as modelling this way added to readability with the added benefit of the features previously discussed. The schema is represented in figure 17 below. (TODO: add orders to the database schema)

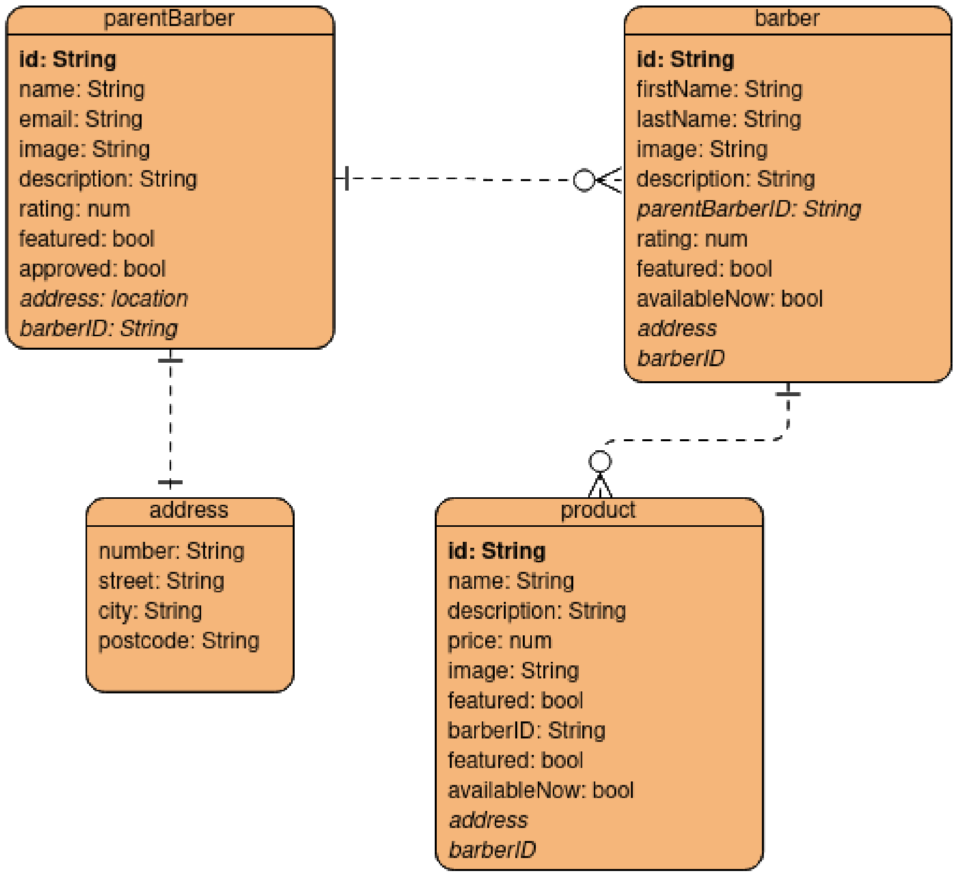


Figure 17: Database Schema

Loading all parentBarbers/ barbers and products - When entering the app it is more effecient to make one call to the server rather than multiple due to GPS restrictions. Therefore ParentBarbers, barbers and all of their products are loaded within a certain radius, rather than loading parentBarbers and barbers separately throughout the app.

Don't nest as calls will fetch all of the parent and nested structured so better to have separate data.

(INSERT DATABASE SCHEMA)

6.5 Sprint X - Connecting the Frontend and Backend

To get the list of items within a range 1) Get the users current location 2) Query firestore with the radius

6.5.1 Loading the barbers

When loading the barbers either they were loading through making a database query based on the parentBarberId once the parent barbers were loaded, or every barber was loaded into memory and this was the filtered in memory.
Pros of first method - Scales well as does not matter how many barbers there are
Cons of first method - more costly as there are more requests to the db
Pros of 2nd - cheaper
cons - does not scale well

6.5.2 Shopping Cart

Chose to store the cart items in a nested array as this adds to readbility and structure and although this restricts look up time, this is not relevant due to the limited number of items a user will order.

6.5.3 Checkout

Similar to section 6.3.1 to create a search time of $\mathcal{O}(1)$ a seperate 'orders' collection was created, with each document representing all of the pertinent order details. Within each barber and user, the document (order) id was then added to each respective order arrays, for quick and easy lookup.

6.6 Sprint X - Location

https://firebase.google.com/docs/firestore/solutions/geoqueries Once the user logs in they give their location in the form of their address. This is stored in 'geohashes', which are longitude and latitiude co-ordinates that are hashed into a single Base32 string. Each character presents a greater level of precision and therefore we opted for 9, which represents an area of 5 x 5 meters.

User location access is granted through the following line in the ./android/app/src/main/AndroidManifest.xml file

```
    <uses-permission android:name="android.
       permission.ACCESS_FINE_LOCATION"/>
```

For the search results we use a drop down menu in the form of flutters built in 'showSearch' function loosely following a guide on medium [22] to display a search page and 'SearchDelegate' to define the content of said search page.

A textEditingController is used to collect the inputted data from the user and pass through to the showSearch function.

6.6.1 Autocomplete locations

As a means for the user to autocomplete their address when signing up, the 'Place Autocomplete service' within the Google Places API, which returns location predictions in response to HTTP requests was implemented using a request adhering to a set of parameters, the full list, along with details of the API can be found on the Google Developers website [19]. First, we enable the Places API within the Google console, before we then create a location model which can hold the data returned from the API. We then create an API request using the above aforementioned API format. For brevity, not every option is discussed, but those of importance include 'input', which is the user query, 'types', which determines the query returned, for which we specify address as we wish to fetch the users full address and a session token, which is required for each new query. The query can be seen here:

```
    'https://maps.googleapis.com/maps/api/place/
       autocomplete/json?input=$input&types=address&
       components=country:uk&lang=en&key=$apiKey&
       sessiontoken=$sessionToken'
```

The returned results are in json format and after some minor error checking we parse using json.decode into a list with our LocationModel class, whilst assigning a new UUID for each query (Google recommends to use version 4 UUID and so this is used here). Without our 'user_gps.dart' file

For the content of the search page we use pass in newly created session token into the ShowSearchPage class, which in turn sends an API request and parses the json data to return a list of locations in the form of 'place id's' and 'description' using a FutureBuilder. From here, we pass through the location id to the getLocationDetails function to fetch the address details of each location and put into a PlaceModel object.

Next, we parse the data into JSON format by passing the PlaceModel object into the function 'createLocationMap', which creates a map using the location data. Finally we pass through this map to the 'addLocationDetails', which uses the given user id to update the database with the users location.

6.7 Sprint X - Barber Side Application

In order to create a comprehensive application the final sprint involved coding a barber side app, giving the ability for the barber interact with the database and allow them to create an account, add and remove barbers and view orders.

6.7.1 Creating a Parent Barber

As the location functionality of the application requires both longitude and latitude co-ordinates, along with a geohash (EXPLAIN HOW PLUGIN USED TO GET THIS).

6.8 Widgets, Common Items and Added Features

Here we discuss any items not covered within a specified sprint.

6.8.1 Widgets

Several widgets were used to increase readability and brevity of code. For example, 'return_text.dart' allows for access to the main components of the Text function and 'return_image.dart' allows for easy use of the NetworkImage function, giving brevity and readability to the code.

6.8.2 Common Items

The common items contains global variables that were accessible throughout the project. Initially this included structures and arrays that served as objects to test the functionality of the frontend, for example a barber shop class with a nested list of barbers classes, each with a name, age, description etc. As backend functionality was added these items were removed. A theme class was then added which contains dart files that can be implemented. Doing it in this way meant that the application could be easily styled, without any unnecessary refactoring of code.

6.8.3 Launcher Icons

Created using GIMP. Plugin flutter_launcher_icons used to install icons across android and iOS

6.8.4 Discount Codes

6.8.5 Hide and Show Password

6.9 Use Cases - Implementation

7 Testing

7.1 User Testing

7.2 Acceptance Testing

[11]

References

[1] *Android vs iOS Market Share 2023*. Statista. 2021. URL: https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/ (visited on 15/03/2021).

[2] Jonathan Arnowitz, Michael Arent and Nevin Berger. "Chapter 15 - Wireframe Prototyping". In: *Effective Prototyping for Software Makers*. Ed. by Jonathan Arnowitz, Michael Arent and Nevin Berger. Interactive Technologies. San Francisco: Morgan Kaufmann, 1st Jan. 2007, pp. 272–292. ISBN: 978-0-12-088568-8. DOI: 10.1016/B978-012088568-8/50016-3. URL: https://www.sciencedirect.com/science/article/pii/B9780120885688500163 (visited on 27/05/2021).

[3] Christopher N Chapman and Russell P Milham. "The Personas' New Clothes: Methodological and Practical Arguments against a Popular Method". In: (Apr. 2005), p. 6.

[4] *Cloud Firestore — Firebase*. URL: https://firebase.google.com/docs/firestore (visited on 16/08/2021).

[5] *Desktop vs Mobile Market Share United Kingdom*. StatCounter Global Stats. 2021. URL: https://gs.statcounter.com/platform-market-share/desktop-mobile/united-kingdom/ (visited on 17/08/2021).

[6] *Desktop vs Mobile vs Tablet Market Share Worldwide*. StatCounter Global Stats. 2021. URL: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/ (visited on 17/08/2021).

[7] *Download Android Studio and SDK Tools — Android Developers*. URL: https://developer.android.com/studio (visited on 16/08/2021).

[8] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. 1st edition. Boston: Addison-Wesley, 4th Sept. 2003. 560 pp. ISBN: 978-0-321-12521-7.

[9] World Leaders in Research-Based User Experience. *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (visited on 16/08/2021).

[10] *Flutter - Beautiful Native Apps in Record Time*. URL: https://flutter.dev/ (visited on 21/05/2021).

[11] Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. 1st edition. Upper Saddle River, NJ: Addison-Wesley Professional, 27th July 2010. 512 pp. ISBN: 978-0-321-60191-9.

[12] Shortcut Mobile Inc. *Shortcut™ — In-Home Haircuts and More from Top Local Salons*. Shortcut. URL: https://www.getshortcut.co/ (visited on 15/08/2021).

[13] Richard Larson and Elizabeth Larson. *Use Cases - What Every Project Manager Should Know*. 2004. URL: https://www.pmi.org/learning/library/use-cases-project-manager-know-8262 (visited on 18/08/2021).

[14] James Martin. *Rapid Application Development*. 1991. URL: https://books.google.co.uk/books/about/Rapid_Application_Development.html?id=o6FQAAAAMAAJ&redir_esc=y (visited on 13/01/2021).

[15] Jennifer (Jen) McGinn and Nalini Kotamraju. “Data-Driven Persona Development”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’08. New York, NY, USA: Association for Computing Machinery, 6th Apr. 2008, pp. 1521–1524. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357292. URL: https://doi.org/10.1145/1357054.1357292 (visited on 20/05/2021).

[16] *Mobile Percentage of Website Traffic 2021*. Statista. 2021. URL: https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/ (visited on 17/08/2021).

[17] *Monday - Home*. 2021. URL: https://monday.com/ (visited on 15/08/2021).

[18] *[PDF] Personas, Participatory Design and Product Development: An Infrastructure for Engagement — Semantic Scholar*. URL: https://www.semanticscholar.org/paper/Personas%2C-Participatory-Design-and-Product-An-for-Grudin-Pruitt/7ea9505694f1d444a330b1947109c7268a97
(visited on 20/05/2021).

[19] *Place Autocomplete Requests — Places API*. Google Developers. URL: https://developers.google.com/maps/documentation/places/web-service/autocomplete#place_autocomplete_requests (visited on 06/07/2021).

[20] Owen Ray. *28 Statistics Home Services Marketers Need to Know in 2021*. 2021. URL: https://www.invoca.com/blog/home-services-marketing-stats (visited on 11/08/2021).

[21] *React Native · Learn Once, Write Anywhere*. URL: https://reactnative.dev/ (visited on 15/08/2021).

[22] Yong Shean. *Location Search Autocomplete in Flutter*. Medium. URL: https://medium.com/comerge/location-search-autocomplete-in-flutter-84f155d44721 (visited on 06/07/2021).

[23] Shopify. *What Is Product Development? Learn The 7-Step Framework Helping Businesses Get to Market Faster*. Shopify. URL: https://www.shopify.co.uk/blog/product-development-process (visited on 21/05/2021).

[24] *Sketch vs Figma, Adobe XD, And Other UI Design Applications*. Smashing Magazine. 11:00:16 +0200 +0200. URL: https://www.smashingmagazine.com/2019/04/sketch-figma-adobe-xd-ui-design-applications/ (visited on 27/05/2021).

[25] *The State of the Octoverse*. The State of the Octoverse. 2019. URL: https://octoverse.github.com/2019/ (visited on 15/08/2021).

[26] *TRIM-IT Mobile Barbershops*. URL: https://trimit.app/ (visited on 15/08/2021).

[27] *TrimCheck - Home Haircuts on-Demand*. URL: https://get.trimcheck.com/haircut/ (visited on 15/08/2021).

[28] *Will Ride-Hailing Profits Ever Come? – TechCrunch*. URL: https://techcrunch.com/2021/02/12/will-ride-hailing-profits-ever-come/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAALI-wmBgq89Iy5-xw3IGxisaXnb2-j1jMukBMZr0fb_p2OE4cobBWyJa_VB8ySIkFyuVC5nWlLixn5_o8Y-SQmmd0NJw68rs3Ek9nhL9d6W4Ycu_pzIwjxa948bQiwzH4HAiSNMcEz0r0Uxnxb0D_RHZM2VX2LLZ8F_II9lS_bqK (visited on 12/08/2021).

Appendices

A Use Cases

A.1 Use case 1 - Sign Up

Use Case 1	**Sign Up**
Description	*Allow the user to sign up and create an account*
Pre-conditions	The user must not be signed in
Basic Flow	1a) The user enters their sign-up details 1b) The user clicks on one of the OAuth sign-in buttons 2) The system creates and authenticates the new user and signs them in 3) The user is signed-in
Alternative Paths	1) The user enters an email address in use and is notified of this 2) The user enters an invalid password or email and is notified of this

A.2 Use case 2 - Login

Use Case 2	**Login**
Description	*Allow the user login*
Pre-conditions	The user must be signed in
Basic Flow	1a) The user enters their login details 1b) The user clicks on one of the OAuth sign-in buttons 2) The system validates the authentication request 3) The user is signed-in
Alternative Paths	1) The user enters the wrong login details and is notified of this

A.3 Use case 3 - Book a Haircut

Use Case 3	**Book a Haircut**
Description	*Allow the user to book a remote haircut*
Pre-conditions	The user must be signed in
Basic Flow	1) The customer navigates to the required product 2) The customer chooses the required quantity and clicks 'Add To Cart' 3) The system adds the item to the users cart 4) The user is taken to the Home Screen
Alternative Paths	1) The user tries to add a product already in the basket and is unable to

A.4 Use case 4 - Search for a Barber

Use Case 4	**Search for a Barber**
Description	*Allow the user to enter a search term to find a relevant barber*
Pre-conditions	The user must be signed in
Basic Flow	1) The customer enters a search term within the search bar on the 'Home' screen 2) The system presents the user with a list of relevant barbers
Alternative Paths	none

A.5 Use case 5 - Checkout

Use Case 5	**Checkout**
Description	*Allow the user to checkout their basket and create an order*
Pre-conditions	The user must have items in their basket
Basic Flow	1) The user requests to checkout their basket 2) The system creates and authenticates the new user and signs them in
Alternative Paths	1) The user does not have any items in their basket and is notified of this

A.6 Use case 6 - View Orders

Use Case 6	**View Orders**
Description	*Allow the user to view their placed or confirmed orders*
Pre-conditions	The customer must have made an order or the barber must have customers orders
Basic Flow	1) The user requests to see their orders 2) The system fetches them and displays them to the user
Alternative Paths	1) The user does not have any orders and is notified of this

A.7 Use case 7 - Sign Out

Use Case 7	**Sign Out**
Description	*Allow the user to sign out of their account*
Pre-conditions	The customer must be signed in
Basic Flow	1) The user requests to sign out 2) The system signs out the user
Alternative Paths	none

A.8 Use case 8 - Add a Barber (*barbers only*)

Use Case 8	**Add a barber**
Description	*Allow the parent barber to add a new barber*
Pre-conditions	None
Basic Flow	1) The parent barber enters the details of the barber 2) The parent barber requests to create a new barber with the given details 3) A new barber is created
Alternative Paths	none

A.9 Use case 9 - Add a Product (*barbers only*)

Use Case 9	**Add a product**
Description	*Allow the parent barber to add a new product*
Pre-conditions	There must be a barber to add the product to
Basic Flow	1) The parent barber enters the details of the product 2) The parent barber assigns the product to a barber 3) The parent barber requests for the product to be created 4) The system creates a new product and assigns it to the barber
Alternative Paths	none

B Figures

C Application Images

D Code Snippets

D.1 AuthenticateProvider

```
Future<bool> barberSignIn({String email, String password}) async {
  try {
    _authStatus = AuthStatus.AUTHENTICATING;
    notifyListeners();
    await _firebaseAuth.signInWithEmailAndPassword(email:
    email, password: password);
    barberModel = await _orderUtility.getBarberById(
    _firebaseAuth.currentUser.uid);
    _authStatus = AuthStatus.BARBER_AUTHENTICATED;
    print("signed in barber " + email);
    notifyListeners();
    return true;
  } on FirebaseAuthException catch (e) {
    print('Failed to sign in barber: ' + e.toString());
    _authStatus = AuthStatus.NOT_AUTHENTICATED;
    notifyListeners();
    return false;
  }
}
```

TODO: Remove this example code TODO: add authentication provider code