Babywatch: Babysitter Booking System Mobile Application

Gokul Sreeletha Kannan
2985939

Submitted in partial fulfilment for the degree of

Master of Science in Computing

Griffith College Dublin

June 2021

Under the supervision of Dr. Viacheslav Filonenko

Disclaimer

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the Degree of Master of Science in Computing at Griffith College Dublin, is entirely my own work and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed: Gokul Sreeletha Kannan **Date: 15 June 2021**

Acknowledgements

First of all, I express my thanks to my supervisor **Dr. Viacheslav Filonenko** for his consistent support and advice during this project and I would like to extend my gratitude to my professors **Dr. Faheem Bukhatwa**, **Dr. M Waseem Akhtar**, and **Barry Denby** for their valuable support and encouragements. I would also like to thank the entire Computing faculty of Griffith College Dublin, my family, and friends for all their support.

Table of Contents

List of Figures

Abstract

One of the main problems that couples faces when they become parents is fine someone to take care of their children while they are away. Some of them would drop their children with their parents or with some friends. However, most of them would call a babysitter to come to their home and look after their children for a fee. The main problem they are facing is to find someone who is suitable for them and their children. Similarly, babysitters are also faced with the problem of finding customers and organising and navigating each of the jobs they have in an easy manner. There are several commercial applications available to help both parties to solve this problem. The complexity of those applications limits or retards the customers form using those. To solve this problem, we provide a solution with an easy-to-use babysitter booking application that would help parents to find a babysitter to take care of their child and would help babysitters to easily find and manage jobs.

This mobile application serves as a platform for both parents and babysitters to satisfy their respective needs regarding the problem. It would allow babysitters to find and apply for jobs and would help parents to select from many sitters to the one that is most suitable for them. It would allow to directly chat between sitters and parents and the parents would also be able to stick with a couple of their favourite ones if they chose to.

Chapter 1. Introduction

Non-parental childcare has become a main part of communities as more and more women with young children across cultural and income levels and ethnic groups joins the labour force [1]. As per a survey conducted by the US Bureau of Labour Statistics published in the year 2000 [2], 57 percent of women with children under the age of 3 and 51 percent with infants were in the workforce at the time. It is safe to assume that those figures have only gone higher. Balancing the demands of family and work is a demanding task for all parents that are working. The main challenge that is faced by the parents is to figure out how to take care of their child while also working. One of the most common solutions that parents opt for is to call upon a babysitter. The sitter will take care of children’s need for a small fee on an hourly basis. However, it is not universally easy for all parents to find someone who is suitable for them as the time demands all the time. This creates a problem where the sitter they usually call will not be available at the time. When looked from the sitter’s perspective, it is difficult for them to see this as a suitable career as the jobs are not permanent for most of them and there is difficulty in managing their jobs and finding new ones.

The use of smart reservation system in different areas like hotel booking, ticket booking, transportation etc. has been increased drastically with the mainstream adoption of smart phones. Thus, it makes sense to inspect how this problem can be solved using smart technology so that both parents and babysitters can benefit from it.

The proposed project idea provides an optimal platform for both the above-mentioned stakeholders to make their jobs easily. A babysitter could easily be able to find new jobs and manage their schedule every week and a parent could easily be able to find a babysitter and replace them if they want to in the future. This reduces the hassle of searching someone who is suitable for minding their child and allows a flexible way to manage them.

1.1 Idea

The underlining idea of this proposal is a mobile application for both babysitters and parents for making their lives easier. As a babysitter, the user would be able to register themselves and choose their working preferences and find a babysitting job that is convenient for them. As a parent it should be able to create a job and find a babysitter who is suitable for that job. This can be made possible by proposing a well-structured system that effectively handles the need of both party's needs. The application should be cross platform regardless of the user's device and should be easily navigable. A mobile application is a safe bet to start with as they are extremely popular nowadays.

1.2 Motivation

As mentioned above, the number of parents who are in the work force is increasing day by day. This constitutes a problem with childcare, as children needs to be provided with proper care. This not just creates a problem but also creates a new job opportunity for babysitters. However, the problem being faced by both parents and babysitters is the hassle of finding each other. As the demand for babysitters increase, the effort needed to find them also increase. This makes it difficult for the parents to find one. Even if a parent has a regular sitter, there will be a case that that person is not available on a given time or the parent want a sitter on a different period that usual which does not work for their regular babysitter. Thus, the parent is forced to search for another babysitter for that short period. This may not be easy for certain parents or may not have enough time to find one using multiple sources. Thus, forcing the parent to work around the problem, to either bother a friend or a family member for the situation or they want to cancel or reschedule their reservation.

Also, from the perspective of the people working as a babysitter, it is difficult to find an adequate job that is suitable for them. They are forced to take one or two jobs from the parents that they knew before and is facing difficulty in finding new opportunities that may better suit them.

The motivation of this proposal is the need to find a solution to the above-mentioned problems. The parents need a easy to use platform to find babysitters easily and quickly as possible and the babysitters should be able to find job easily.

1.3 Aim

The aim of this proposal is to find a suitable solution the above-mentioned problem with the help of smart technologies. This mobile application would focus on the two-potential type of users and making their life easier as a primary concern. It would allow parents to find babysitters to take care of their child easily and will allow babysitters to find new jobs that are more suitable for them. It will also allow babysitters and parents to interact with each other through chat messages with the help of its easy-to-use UI.

1.4 Goal

Everybody will come across some emergencies in their life. There will always be those last-minute business meetings or trips that one cannot avoid and did not get time prepare. This is extremely hard in case of a parent as their child needs someone to look after them in the meantime. Finding a babysitter in the last minute is one of the hardest things that one can face. Also, there will be the case for those parents when their regular babysitter is not available for a day or maybe he/she has moved to a new location and is not available anymore. All these situations create hassle in the daily life of parents as their kids are someone they cannot afford to compromise on.

Similarly, babysitters are also faced with several problems. There will be the case that one’s regular parents are not available anymore because they moved, or their kid has grown up and they do not need a sitter anymore or they found a new sitter who is more beneficial for them. In such cases, the babysitters are tasked with the job of finding a new opportunity.

The goal of this proposal is to provide an efficient solution to these solutions so that both parents and babysitters can benefit from it. The goal is to create a platform so that the parents and babysitters can find each other with ease. They could be able to interact with each other so that parents can find a suitable babysitter who could take care of their child while they are away and babysitters could get more options in their field of work, which would increase their freedom of choice in who to work for.

1.5 Document Structure

The remaining part of the document follows the following structure, in chapter two, the background task that is done before the development is started is explained such as literature review, and a run through among the similar work. The chapter three discusses the methodology which involves a quick rundown of the technologies that are used in this project and their alternatives, such as, NoSQL databases, Google Firebase, Flutter toolkit and so on. The fourth chapter discusses the system design and specifications. In this chapter, the structure of the project is designed and how the code is implemented. It discusses about the data models used, the different screens and its contents and finally the different services used in the app. The chapter five is the implementation, where the practical implementation is discussed. In this section, initially focusses on how the database is integrated and then looks at different screens in the app and the logic and their purpose is discussed. Then, chapter 6 is the testing and evaluation where it outlines how the application was tested. The final chapter looks at the project as a whole and concludes the basic ideas about the project and outlines what are the works that could be done in the future to improve the application.

Chapter 2. Background

In this era of smart technologies, quality of living has become more advanced. Using apps in smart phones has become a part of daily life. The invention of new technologies such as AI, smart phones and other wide range of software are helpful to ease the workload in daily life. Smart phones are already an integral part of life for most people. Thus, use of mobile applications to solve problems in real life has exploded in the past decade. A babysitter booking management system allows parents to create a job and find a babysitter who is suitable for it and allows babysitters to find jobs easily.

2.1 Literature Review

There are a wide range of booking systems out there. Applications with these systems usually have a somewhat similar structure. Let us take a food booking system such as JustEat [3] or Deliveroo [4] for example, there are three stakeholders, the customers who orders the food, the restaurant which makes the food and the delivery personal who delivers the food. A similar one applies for Uber [5] or FreeNow [6] which are used to book a ride. In their case, there are only two stakeholders, the customer, and the driver. These are just two examples. The point is even though they are used for different purposes, they have similar structure of handling its users. Some provide single application for all stakeholders while others provide different one for each stakeholder. The use of smart reservation system in different areas like hotel booking, ticket booking, transportation etc. has been increased drastically with the mainstream adoption of smart phones. There are several apps for web and mobile, along with the ones mentioned above, that provides similar services.

2.1.1 Computer Reservation System

The Computer Reservation System (CRS), also known as Central Reservation System, is used by a wide number of organizations for managing their booking system and transactions related to them [7]. The advance features and improved architecture along with its procedures, the CRS is popular among multi-level organizations. It plays significant role in both Technology and Business transactions. The focus of the system is to minimize the physical and geographical distances between the customers and the mediator. CRS has several commercial implications that is beyond the scope of this project. However, the basic idea is to provide a system that is centralised and that can interact with all its customers to minimize duplications and control more efficiently. CRS is mainly used in tourism industry for booking hotels across the world. However, the approach mentioned by them is also relevant in this project.

2.1.2 Google Firebase

Google Firebase is a framework which is used for developing web and mobile applications [8]. It is developed by Google and consist of a variety of services. It provides a unified platform for many apps along with a group of other Google services packed inside within. Firebase is equipped to handle most of the backend tasks. There are numerous elements in the service that are considered as essential tools for any software. These elements include Cloud Firestore, which is NoSQL database within Firebase. The Firestore is a schemeless database, which implies that there is no need to declare a previously defined schema to initialize it. It also consists of Firebase Authentication, which gives a simple-to-use authentication system to confirm the clients of an application. It supports authentications with email and password along with Google account, Facebook, and a wide variety of third-party websites. Then there is Firebase Storage for storing files and other user-generated content, such as photos or videos etc. The Firebase Storage is powered by Google Cloud.

Google Firebase has several other services such as Firebase Hosting, Cloud Messaging, Firebase Functions etc to name a few. However, those are not relevant to this project.

2.1.3 Flutter

Flutter [9] is a framework developed by Google which was released in 2016. It is a cross-platform framework used for building high-performance mobile applications. Flutter applications can run on both iOS and Android, along with Fuschia. It is chosen to be Google's app-level framework for its next-generation OS. Flutter is depending on the device's OEM widgets rather than consumption of web views. It provides superior performance rendering engine for rendering every view component natively, thereby provides the opportunity to develop hight performance applications natively. When it comes to the architecture, the engines C or C++ code involves compilation with LLVM for iOS and NDK for Android and during the process, the Dart code is compiled into native code. The Hot reload feature, which is a major feature in Flutter is known as Stateful hot reload and it boosts the development cycle. It is implemented by passing the updated source code into the already running Dart Virtual Machine without changing the application’s inner structure. Thus, the actions and transitions of the application is preserved after performing hot reload.

2.2 Similar Work

There are only a few mobile applications available for reserving a babysitter. However, the under lying principle that is used is from the Central Reservation System and there are several other services that is providing similar services. As mentioned previously, services like JustEat, Freenow etc. may find familiar. There are also several websites that provides similar services. Let us look at some applications that might seem familiar to the babysitter booking application.

2.2.1 MindMe.ie

Mindme.ie is one of the most common one out there. It makes it easier for the caregivers and families to find one another. It has both a fully functional mobile app for both iOS and Android, and a web application. It is not exclusive to find babysitters. It can be used for finding all sorts of care givers.

2.2.2 Care.com

Care.com is a website that provides similar services to mindme.ie. It does not have a mobile application. It also provides not just childcare, but also other services such as elderly care, pet care and so on.

2.2.3 SureSitter

SureSitter is a website that is most like this project. It is a full-on babysitter booking application that primarily focusses on connecting babysitters and parents. They have a wide range of categories such as part time, full time, night nurse etc.

2.2.4 Babysits

Babysits is another application that targets similar demographic. It has both web application and mobile application. It helps parents by facilitating the best search experience, with several options to search. It makes easy for the families to find a babysitter and facilitates communication through messaging service and appointment planner. It helps babysitters by providing freedom, flexibility, and opportunity.

2.2.5 Sitter Pro

Sitter Pro allows the users to easily book professionally screened babysitter, pet sitters, nannies, tutors and senior care providers from both local nanny and care agencies. It allows users to find the best fit caregiver for their family’s needs. It is a easy to use, professional application for parents. However, as they are professionally screened it is not easy for a babysitter to get in.

Chapter 3. Methodology

One of the most important steps is to finalize the architecture and technologies that are going to be used in the project. To do this, the performance and cost of maintenance must be considered. Choosing a good architecture with suitable design patterns will help the application to solve problems efficiently. Before jumping into a decision, it is important to identify the potential problems and break them into smaller parts. Then, by solving those small parts, basic building blocks can be created, and the app can be built by composing those building blocks together. In fact, composition is a fundamental principle that is used extensively not just in Flutter, but, more widely in software development.

A good architecture should have good building blocks. It should be easy to compose them together to do more complex tasks and in time to construct the entire application. Adding new features to the app should be easier. The codebase should be easier to understand. Also, the components should have clear responsibilities and should not do too many things.

The architecture employed in this project is a customised one. In this application, Flutter is used as front-end, and Google Firebase is used as the back end. All the data is stored in Cloud Firestore. Authentication is done using the Firebase Authentication. Firebase Storage is used for storing files.

When it comes to the front end, Flutter [10] is used to layout the UI. It is a cross-platform toolkit for UI that is developed to facilitate the reuse of code across iOS and Android. It also allows direct interface of apps with the underlying platform services. The idea is to allow developers to develop high-performance applications that feels native on several platforms, by embracing the differences when needed and sharing code when possible. The Flutter apps run in a virtual machine during development. This virtual machine will offer changes to stateful hot reload without demanding a full recompile. Flutter applications are compiled directly to machine code on release, whether ARM instructions or Intel x64, or to JavaScript for the web. Flutter framework is open source, with BSD license that is permissive, and has support of several third-party packages that helps in development.

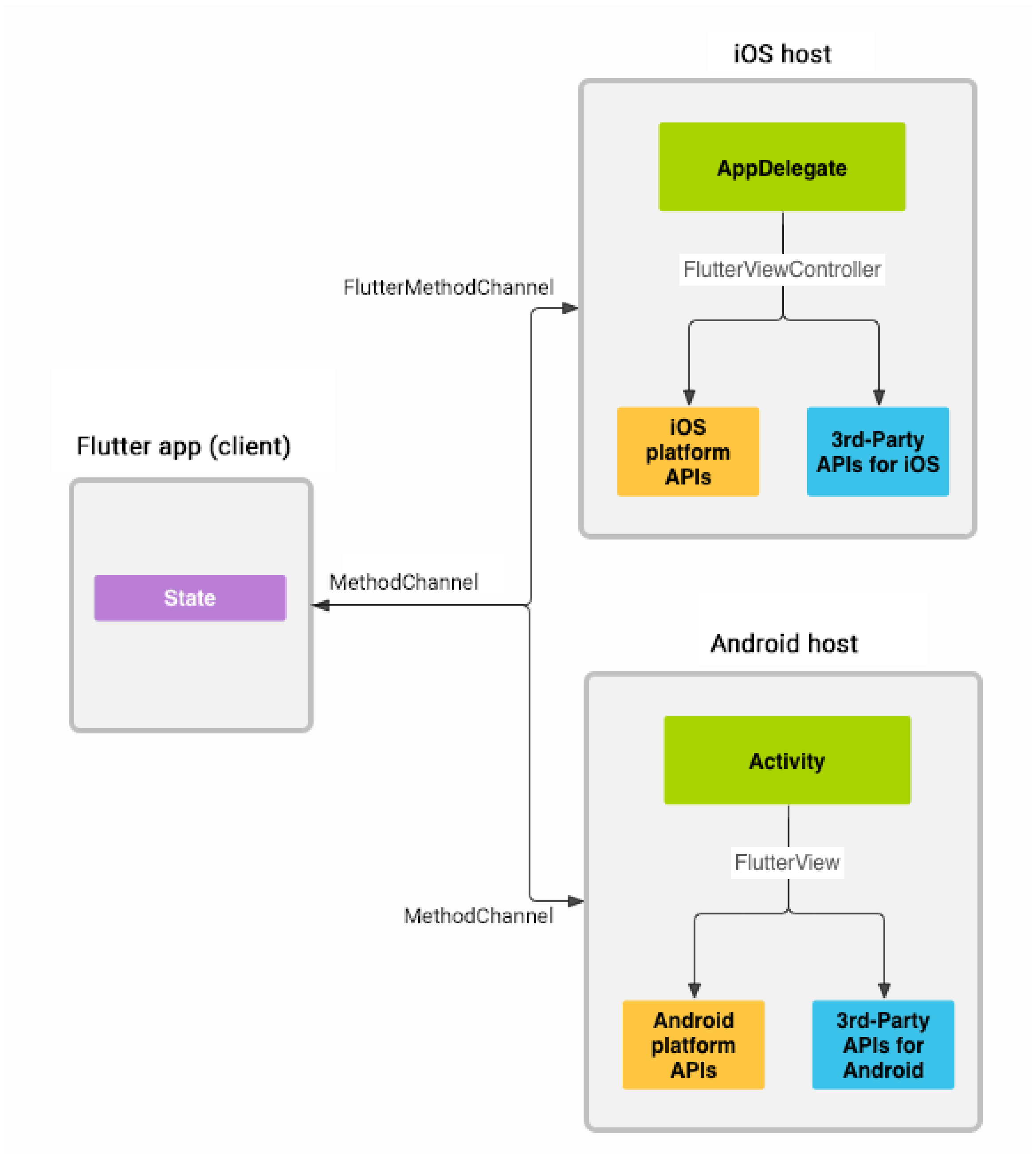


Figure 1 Flutter Architectural Overview

The architectural overview of Flutter is shown in the Figure 3.1. It illustrates how Flutter manages to separately render applications for two platforms with the use of its Method Channel. In this project, the project is started from the main.dart file. The routes to different views are specified in this file. The user interface for each view is stored in the screens folder and there is a widgets folder within the screens folder that has custom components used in the application. The logic to fetch data and display in each view is incorporated within each view file. This is a common practice for Flutter apps using Firebase and is commonly known as

Serverless Architecture. Besides the screens folder, there is also the Models folder which holds the definitions of each data model used in the app.

3.1 NoSQL Databases

One of the most important steps in designing the back end is to decide which type of database is to be used for development. There are prominently two types of databases, SQL, and NoSQL. The decision depends on the application. There is no "one for all" solution for databases. Relational databases are typically closed source, while NoSQL databases are mostly open source. NoSQL is abbreviated as Not Only SQL. It is also known as non-relational database management system. They are typically do not have a predefined schema, unlike SQL databases, which are typically in the form of tables with columns and rows. NoSQL are designed with the ability to scale horizontally in mind. This makes them a good choice for large scale applications that are hosted in the cloud or small-scale ones that has the potential to become large scale in the future. The most used formats for storing data are JSON or XML. They support Object-based APIs that enables apps to easily manage data. Google's Cloud Firestore is a NoSQL database that support the scalability that is required for this project. It is asynchronous by nature. Hence, it is chosen as the database.

3.1.1 NoSQL Data Models

Data models are designed to flexibly support the storage needs of the applications. The primary objective of NoSQL database systems is to distribute data evenly among instances. The data models are required for the NoSQL database [11] to offer a structured model to the developers and to allow easy access of data. There are several types of data models in NoSQL databases,

- **Column Family Stores**: They were designed to store very large amount of date and process them by distributing them over several machines. There are keys that points to multiple columns that are arranged by column family.
- **Key-value Stores**: This is based on the hash table model, where there is a unique key and value that is a pointer to an instance of data. This is the simplest model and easiest to implement. It has many disadvantages such as its inefficiency in updating or querying part of data.

- **Graph Databases**: It has a graph model that is flexible and can scale across platforms.
- **Document Databases**: Inspired by Lotus Notes, they are like key-value stores. The model consists of collections of documents that are key-value collections themselves. Mostly stored in JSON format. They can query more efficiently.

3.1.2 NoSQL Characteristics

In traditional database systems based on data transactions, the integrity of data is guaranteed. This illustrates that data consistency in all situations of data management. The characteristics of data transmission is known as ACID, which is short for Atomicity, Consistency, Isolation, and Durability.

The main characteristics of NoSQL system is considered as three.

- Consistency
- Partition Tolerance
- Availability

The availability of the database refers to the fact that the system continues to operate as expected even with certain node failures. This means that the system is robust. Consistency refers to all the users can access the data in real-time irrespective of any update or deletion in another node. Partition Tolerance refers to the fact that the system provides complete characters to its users even if it is deployed in different servers.

3.1.3 NoSQL Advantages

NoSQL database technology was created as a response to the limitations of the more traditional relational databases [12]. In comparison, NoSQL databases provide superior performance and are more scalable. Also, the ease and flexibility to use data models can speed the development, especially in cloud computing environments.

Every specific NoSQL database has their own specific strengths; however, they all share some fundamental advantages that allows them to:

- Store unstructured, semi-structured and structured data.
- Be developer friendly.

- Handle large amount of data at a very high speed with a horizontally scale-out architecture.
- Easily enable the ability to update schemas and fields.
- Take advantage of the cloud so as to deliver no downtime.

3.2 User Interface

The User Interface (UI) is one of the most important part of any application. It is the part that makes it possible for the user to interact with application. The interface can be developed for different platforms separately. There are some tools allows to develop for multiple platforms or OS simultaneously. In this section, the main way of developing UI for an app is analysed and compared with each other.

3.2.1 Android Development

Android is a Linux based open-source platform mobile operating system developed by Google and was unveiled in 2007. Its source code is open, meaning anybody can take it and customize or make a new version of it as they please.

For this OS, the main choice of development languages is Java or Kotlin. The UI elements are made using XML files [13]. Since the development process lacks the tools for management of scene navigation, the rules of describing the order of scenes are defined within the Fragment derived classes and the Activity. The responsiveness of the UI is made possible by isolation of all complex operations in secondary threads. This type of approach reduced the workload on the main thread, which could be able to handle UI updates and input interactions without delays.

Slower operations like network connection establishment or decoding of data are implemented by deriving the AsyncTask class from Java. Once a data set is ready, the main thread gets notified to this change with the help of observer design pattern that is optimized for multiple listeners. Activities are created for every section context, leaving scenes to be handled by the use of fragments. In case of scenes that involves lists and grids, the app can take advantage of the reusable item view method, minimizing the memory usage to store complex data arrays.

3.2.2 iOS Development

iOS is the second most used mobile operating system in the world developed by Apple exclusive for their hardware. It powers Apple's iPhones and uses to power iPads before iPad OS.

The iOS application development tool is XCode [13] which offers two language options for native development, Swift and Objective C. Swift is the most promoted one for development of new application that run in Apple devices at this point. However, Objective C is regarded as more mature language that has clearly outlined best practices, and coding styles.

The visual structure of an application can be managed in a single file with the use of Storyboard environment. Every individual scene was constructed using static View Controllers used for standalone pages and structured data list using Collection View Controllers. The developers can follow several architectures such as MVC, MVP or MVVM [14].

MVC design pattern divides objects into three: model, view, and controller. Models contains the data specific to the app. View has the purpose of displaying data taken from the model to the user. Controller acts as a mediator between view and model.

MVP pattern six components namely, view, model, commands, presenter, interactor, and selections. It was invented in the 1990s as a current C++ initiative from IBM, HP, and Apple as a replacement to MVC pattern.

MVVM stands for Model, View, View-Model. This architectural pattern is most used for Windows Apps. The pattern was developed by Ted Peters and Ken Cooper. In MVVM, the Swift class extends with controller that is considered as view. View only deals with how the data can be presented as they are passed from View Model Class. In View Model, the core method is preparing and managing the data for a View. It also handles data of communication from local data and rest of the application. After the creation of the view, the View class invokes the rest function by view model. View model will then execute the logic and data needed to display to view class.

3.2.3 Fire OS Development

Fire OS is a fork of Android that runs on Amazon devices such as Fire TV [15] and Fire Tablets. At its core, both Android and Fire OS share the same foundation. The main difference between Android and Fire OS is in the services. Instead of using the Google Mobile Services for activities such as location, browsing, messaging, payments etc., Fire OS uses Amazon services. Amazon uses their own App store known as Amazon Appstore instead of Google Play Store.

Porting an Android app to Fire OS platform requires to connect to Amazon services. When building the app, the Amazon provides detailed documentation to help developers to port their apps to Fire OS.

The install base of Fire OS is much smaller as compared to iOS and Android.

Apart from the native app development, there are several tools that allows to develop for multiple platforms simultaneously. Such apps have toolkits that allows developers to develop for multiple platforms with the same code base.

3.2.4 React Native Development

React Native is a framework capable of developing hybrid mobile applications for cross platforms developed by Facebook in 2015. It uses JavaScript ES6 as the programming language and builds applications that can run on mobile platforms such as iOS and Android with single code base. It combines the best of native with React, which is a in-class JavaScript Library for building UIs. It allows developers to build truly native apps and does not compromise when it comes to user experience. React Native [16] is supported by several companies across the globe. It is used to develop thousands of cross platform apps. The developer needs a basic understanding of JavaScript fundamentals.

Components can be made using either classes or functions. Only class components could have state before. But since React Hooks API, the state can be added and more to function components. Hooks were introduced in React Native version 0.59. Hooks are regarded as the future to write React components.

3.2.5 Flutter Development

Flutter is another option for cross platform application development. It is a UI toolkit designed by Google to allow the reuse of code across OS such as iOS and Android. It is a direct competition to React Native. The main difference between React Native and Flutter is that Flutter code is directly compiled into machine code and thereby gaining better performance. The idea is to enable developers to develop high-performance applications that feel native on multiple platforms, by embracing differences where they needed and sharing the code when possible.

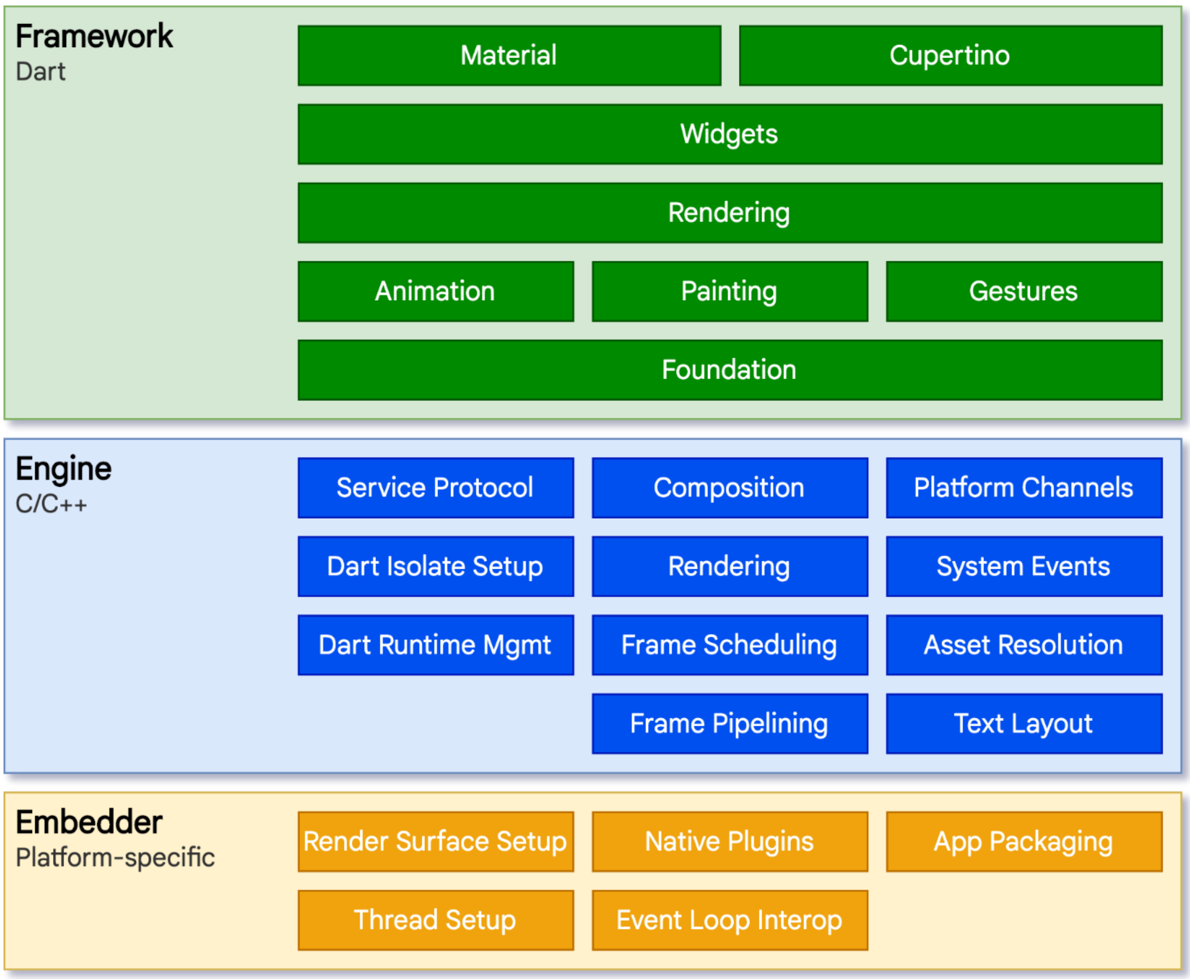


Figure 2 Flutter Architectural Layer System

Flutter is developed as an extensible, layered system [10]. It exists as a number of independent libraries that underlying layer is depended by each layer. The layer above cannot access the layer below, and every part of the framework is designed to be replaceable and optional.

To the OS that is underlying, Flutter apps are packaged as same as native apps. A platform-specific embedder facilitates an entry point; coordinates with the underlying OS for access to services such as rendering surfaces, input, and accessibility; and manages the loop event of messages. The embedder is written in some language suitable for the platform. They are Objective C/ Objective C++ for iOS and macOS, C++ and Java for Android and C++ for Linux and Windows. By using the embedder, Flutter code can be integrated into a module in an application that currently exists.

Flutter, at its core, has the Flutter engine, which is mostly C++ code and supports the primitives that are necessary to support all Flutter apps. The engine rasterizes composited scenes when a new frame needs to be created. It has low level implementation of Flutter's core API, including text layout, file, graphics and network I/O, plugin architecture, accessibility support and a Dart compile and runtime toolchain. The engine is exposed to Flutter framework via dart:ui, that constitutes underlying C++ code in Dart classes. This library consists of the lowest-level primitives, like classes for driving graphics, input, and text rendering subsystems.

Usually, developers interact with Flutter through Flutter framework, which provides reactive, modern framework written in Dart. It includes a rich set of layout, platform, and foundational libraries, made of a series of layers. From the bottom to the top:

- Basic foundational classes, and services that are building blocks such as painting, animation and gestures that offer abstractions over the foundation.
- The rendering layer provides an abstraction for layout. In this layer, the developers can build a tree of objects that are render able.
- In the widgets layer, each render object from the layer below has a class. Also, this layer allows to define combinations of classes that are reusable. Reactive programming model is introduced in this layer.
- The Cupertino and Material libraries provides sets of controls that utilizes the composition primitives of widget layer to implement corresponding design languages.

The Flutter framework is very small. Many features in the higher level are implemented as packages, such as plugins for camera and web view. Flutter interacts with other code at a platform level.

3.3 Dart Packages

Flutter supports the idea of shared packages that are contributed by other developers to the Dart and Flutter ecosystems. This enables the quick building of an app rather than building everything from scratch. There are several packages used in this project.

3.3.1 Cupertino Icons

Cupertino Icons package [17] is an asset repository designed for flutter that contains the default set of icon assets that are used by Flutter's Cupertino widgets based on Apple styled icons. The version used here is 0.1.3. It is published by the Flutter team themselves.

3.3.2 Firebase Core

Firebase core [18] is a plugin that is developed for Flutter by the firebase.google.com. It allows the Flutter apps to use the Firebase Core API, enabling it to connect to multiple Firebase applications. The version used in this project is 1.0.4.

3.3.3 Firebase Auth

Firebase Auth [19] is a plugin that is developed for Flutter by that Google firebase team. It is a plugin for Firebase auth in Flutter, that enables the iOS and Android authentication using phone number, passwords, and other identity providers such as Facebook, Google and Twitter. The version used here is 1.1.1.

3.3.4 Cloud Firestore

Cloud Firestore [20] is another plugin that is developed for Flutter by firebase.google.com. It allows the Flutter apps to use the Cloud Firestore API and connect with the Google's Cloud

Firestore, which is the NoSQL, cloud hosted database that provides live synchronization and offline support for both iOS and Android. The version used here is 1.0.6.

3.3.5 Modal Progress HUD

Modal Progress HUD [21] is a Flutter package developed by mmcc007@gmail.com. It is a simple widget wrapper that is used to enable a modal progress Heads Up Display to block access to the wrapped widget during asynchronous call. It also shows a custom spinner. The version used here is modal_progress_hud 0.1.3.

3.3.6 Provider

Provider [21] is one of the most widely used packages in Flutter. It is mainly used for state management. It is published by dash-overflow.net. It functions as a wrapper around inherited widget, making them reusable. Instead of manually writing InheritedWidget, by the use of this package, the developers will get advantages such as lazy loading, simplified allocation of resources, increased scalability of classes by incorporating listening mechanisms that can grow exponentially with the complexity of the application. The version used here is provider 5.0.0.

3.3.7 HTTP

Http [22] is a Flutter package developed by the dart.dev. It is a Flutter-based library for managing HTTP requests. It consists of a set of high-level classes and functions that enables the consumption of HTTP resources easier. The version used here is http 0.13.2.

3.3.8 Dio

Dio [24] is published by flutterchina.club. It is a powerful Http client for Dart that supports Form Data, Interceptors, File Downloading, Request Cancellation, Timeout etc. The version used here is dio 2.1.13.

3.3.9 Table Calendar

Table Calendar [25] is a third party plugin for Flutter that is developed by aleksanderwozniak96@gmail.com. It allows the Flutter apps to use a highly customizable and feature-packed calendar widget that is extensive, easy to use, a UI that is preconfigured and has customizable styling, custom UI builders etc. The version used here is table_calendar 2.0.0.

3.3.10 RFlutter Alert

RFlutter Alter [26] is developed by ratel.com.tr. It is a easy-to-use, super customizable alert dialog for Flutter. It has several features such as reusable styles, ability to add buttons dynamically, predefined alert styles etc. The version used here is rflutter_alert 2.0.2.

3.3.11 Number Picker

The Number Picker [27] is a flutter package developed by marcinszalek1@gmail.com and mstobieniecka1@gmail.com. It is custom widget that is designed to allow developers to choose a decimal or integer by scrolling spinners. The version used here is numberpicker 2.1.1.

3.3.12 Image Picker

Image Picker [28] is developed by flutter.dev. It is a plugin for Android and iOS for picking images from the gallery and taking new pictures using the camera. The version use in this project is image_picker 0.7.4.

3.3.13 Firebase Storage

Firebase storage [29] is a Flutter plugin that is developed by the firebase.google.com. It allows the Flutter apps to use Firebase Cloud Storage API and connect to the Cloud Storage in Firebase, which is a cost effective, simple object storage service for iOS and Android. The version used in this project is firebase_storage 8.0.5.

3.3.14 Permission Handler

Permission Handler [30] is developed by baseflow.com. It is a permission plugin designed for Flutter. It enables cross-platform API to request and check for permissions and their status. The developers can also an app in the device settings so that the users can grant the permissions desired. The version used in this project is permission_handler 7.1.0.

3.3.15 URL Launcher

URL Launcher [31] is a Flutter plugin that is published by the Flutter team, flutter.dev. It allows the Flutter application to launch a URL. It supports for web, macOS, Linux, Windows, iOS, and Android. The version used in this project is url_launcher 6.0.4.

3.4 Google Firebase

Firebase is development platform from Google [31]. It enables developers to build their desired applications using their dedicated tools. The services that Firebase provides can be mainly divide into Services that help the developers to

- Build the app.
- Improve the app's quality.
- Grow the business.

For building the application, the developers can use tools like Firebase Authentication, Cloud Firestore, Hosting, Machine Learning, Realtime Database, Cloud Storage, and Cloud Functions.

To improve the app's quality, they can use App Distribution, Test Lab, Performance Monitoring, and Google Crashlytics.

To grow the business, they can use Extensions, Predictions, A/B Testing, Analytics, In-App Messaging, Cloud Messaging, App Indexing, Dynamic Links, and Remote Config.

From all the above services, the ones that are used in this project are Cloud Firestore, Firebase Authentication, and Cloud Storage.

3.3.1 Cloud Firestore

Cloud Firestore is a scalable and flexible NoSQL cloud database to store and synchronize data for both client-side and server-side development. It has the ability to sync data across client applications through the means of real-time listeners and provides offline support for web and mobile so that the developers can build responsive applications that provides high performance irrespective of internet connectivity or network latency [33]. It also enables seamless integration with Google cloud and other Firebase products including Cloud Functions. Some of the key capabilities of the Firestore are:

- **Expressive Querying:** In the Firestore, the developers can make use of queries for the retrieval of specific, individual documents or in a collection that matches the query parameters. Each query can have multiple, chained filters and combine sorting and filtering.
- **Offline Support:** The Firestore caches data that the application uses actively. This allows it to read, write, listen, and query data even when offline. When the devices become online again, the Firestore syncs the local changes back to the Firestore.
- **Flexibility:** The data model of the Firestore supports flexible, hierarchical data structures. The data is stored in documents that are organized into collections. Each document can contain complex nested objects as well as subcollections.
- **Realtime Updates:** Just like its Realtime database, the Firestore utilizes data synchronization to update data on any device that is connected to them. It is also designed to efficiently enable simple, one-time fetch queries.
- **Designed to Scale:** Firestore is powered by the powerful infrastructure of Google Cloud. It supports strong consistency, automatic data replication in multiple regions, atomic batch operations, and transactions. It is designed to handle toughest workloads of database froth the biggest applications in the world.

3.3.2 Firebase Authentication

As most of the applications needs to identify the users [34], so that it can save the user data securely in cloud and enable the app experience that is personalized across all of its user devices, Firebase Authentication facilitates SDK’s, backend services, and ready to use UI libraries to authenticate the users. It can integrate closely with other services from Firebase and takes advantage of the industry standards such as OpenID Connect and OAuth 2.0, so that it can easily integrate with a custom back end.

The developers can sign in users to the app either by using the Firebase UI, which is a complete drop-in authentication solution or by using the SDK from Firebase Authentication to integrate sign-in methods manually to the app.

As mentioned above, even though , the drop-in authentication solution is the recommended one to sign-in users, in this application, the sign-in method is implemented using the Firebase SDK. The only way for the users to sign-in is using an email and password. The SDK provides functions to create and manage users and allow them to use email and passwords to sign-in.

In order to authenticate users to the application, the app gets the credentials from the user. Then, these credentials are passed to the Firebase Authentication SDK. The back end services of the Firebase will then verify them and return a response to the client.

After the authentication is successful, the app can access user's basic profile information, and it can access user data stored in other Firebase products. The authentication token provided after this can be used to identify the user in other backend services.

3.3.3 Firebase Cloud Storage

The Firebase Cloud Storage is developed for the developers who need to store and serve content that is generated by the user such as videos and photos [35]. The Cloud Storage is simple, powerful, and cost-effective object storage service developed for Google scale. The SDK for Cloud Storage adds Google security to file downloads and uploads for apps, regardless of the quality of the network. The key capabilities are as follows:

- **Strong Security:** The Cloud Storage SDK integrates with Firebase Authentication to enable simple and intuitive authentication solution for the developers. The developers can utilize the declarative security model of the Firebase to provide access based on filename, content type, size, and other metadata.
- **Robust Operations:** The SDK for the Storage perform downloads and uploads irrespective of the network quality. The operations are robust, saving the user time, restarts when they stopped and saves bandwidth.
- **High Scalability:** The Cloud Storage is designed for high scalability. It allows the app to effortlessly grow from a simple prototype to a production application using the same infrastructure that powers the entire Google infrastructure.

Chapter 4. System Design and Specifications

The application is designed for mobile platforms, Android, and iOS. The development of the UI is done using Flutter framework and the backend is hosted on Google Firebase and its various services. This type of architecture is commonly known as Server-Less Architecture. The application is developed in a Windows laptop, Dell XPS 15 9370 using the following technologies.

Technology	**Version**
Firebase Core for Flutter	1.0.4
Cloud Firestore for Flutter	1.0.6
Firebase Auth for Flutter	1.1.1
Firebase Storage for Flutter	8.0.5
Flutter	2.2
Android Studio	4.2.0
Dart SDK	2.13.1

Table 4.1 Technologies and Versions used for Development.

4.1 Model

The Data Model is one of the major components of an application. It is the basic structure in which data is stored in the database and a skeleton of how the users are represented in the application. The Cloud Firestore from Firebase is used to handle data in this application.

As mentioned previously, the Cloud Firestore is a NoSQL database, and this allows the data to be stored in without any structure or schema. The data is stored in the form of key-value pairs and the data is retrieved using the parent-child branching system.

The application has two types of users, Babysitter and Parent. All those users have some of the features in common. Also, we have the jobs model that is used to create handle jobs for these users. Since, there are some similarities between babysitter model and parent model. Both models are inherited from another model known as the User Model. However, there is no particular User Model and this model is typically used as framework for creating the babysitter and the parent, it is better to understand them as separate models rather than an inherited model from the User Model.

4.1.1 Babysitter Model

The Babysitter model is the skeletal structure of data that a babysitter should provide when they are signing up for using the application. This data is used to better identify and customize the jobs that are more suitable for the babysitter.

- **name:** This is a String field used to store the name of the babysitter.
- **email:** This field is used to store email of the user.
- **street:** This is a String used to capture the street where the user is living.
- **county:** This is a String used to capture the county where the user is living.
- **phone:** This is a String used to save the phone number of the Babysitter.
- **rating:** This is a String that is used to hold the Average rating of the babysitter.
- **wage:** This is an integer that is used to hold the minimum wage that the babysitter is expecting.
- **maxAgeofChild:** This is an integer that is holding the maximum age of child that the babysitter is willing to work with.
- **minAgeofChild:** This is an integer used to hold the minimum age of child that the babysitter is willing to work with.
- **maxNoofChildren:** This is an integer used to hold the maximum number of children that the babysitter is willing to work with.
- **offeredJobs:** This is an array that contains the unique ids of the jobs that are offered to the babysitter by parents and are pending approval of the babysitter.
- **imageURL:** This is a string that holds the link to the image URL of the profile picture of the babysitter.
- **followers:** This is an array of ids of parents who are following the babysitter.
- **about:** This is a String that holds a self-description of the babysitter.

- **appliedJobs:** This is an array that holds the list of jobs that the babysitter has applied to and are pending approval of the parents.
- **assignedJobs:** This is an array that holds the list of all jobs that are assigned to the babysitter.

Besides these, each Babysitter will have two collections, chats, and feedback. Chats are the collection of messages that the babysitter had with the parents. Each chat has the id of the parent that they had chat with. Inside each chat, there is a flag that indicates if there are any unread messages. And inside each chat, has a collection of data which contains the message text, the time stamp when the message was send, the status that indicates whether the message is sent, delivered, or read, and the id of the sender.

While the chats collection is common for all users, the feedback collection is exclusive to the babysitter. Each feedback has the id of the sender as its id. Each feedback entry consists of a score which is an integer with maximum value of 5 that indicates the score given by the parent to the babysitter and a text that explains why they gave him/her that score.

4.1.2 Parent Model

The Parent model is the skeletal structure of data that a parent should provide when they are signing up for using the application. This data is used to better identify and customize the babysitters that are more suitable for the parent.

- **name:** This is a String field used to store the name of the parent.
- **email:** This field is used to store email of the parent.
- **street:** This is a String used to capture the street where the parent is living.
- **county:** This is a String used to capture the county where the parent is living.
- **phone:** This is a String used to save the phone number of the parent.
- **imageURL:** This is a string that holds the link to the image URL of the profile picture of the parent.
- **contacts:** This is an array of babysitters that the parent is following.
- **children:** This is an array containing maps of age and gender of children that the parent has.
- **about:** This is a String that holds a description of the parent and their children.

Similar to the Babysitters, the Parents has the collection of chats, which is a collection containing the conversations they had with babysitters. Each conversation's id is the id of the babysitter they chat with and each of them has the same fields that are mentioned above in the babysitter model.

4.1.3 Job Model

The Job Model is the skeletal structure of data that a job should have when they are created in the application. This data is used to better identify and customize the jobs that are more suitable for the babysitters.

- **date:** This is String that holds the date in which the job is supposed to happen. The original date and time are formatted into "d MMMM, yyyy".
- **from:** This is also a String that holds the time from which the job starts. The original time variable is formatted into "hh:mm a" format.
- **to:** This is also a String that holds the time at which the job ends. The original time variable is formatted into "hh:mm a" format.
- **maxWage:** This is an Integer that holds the maximum amount of money in euros that the creator of the job is willing to pay for it in an hour.
- **status:** This is a String that holds the status of the job, indicating whether the job is complete or not.
- **creator:** This is a String that holds the value of id of the parent who created the job. This is used to identify who is the person who created the job.
- **children:** This is a copy of the children of the creator to identify how many children the parent has. This is used to filter the jobs for babysitters with maximum number of children.
- **askedBy:** This is an array that holds ids of all the babysitters who has applied to the job. This allows the parent to see who are interested in the job.
- **askedTo:** This is an array that holds the ids of all the babysitters who the parent has asked to take the job.

- **assignedTo:** This is a String that holds the id of the babysitter who is assigned to the job.

The job model does not have any collections associated with it. A diagrammatic representation of the data model is shown below.

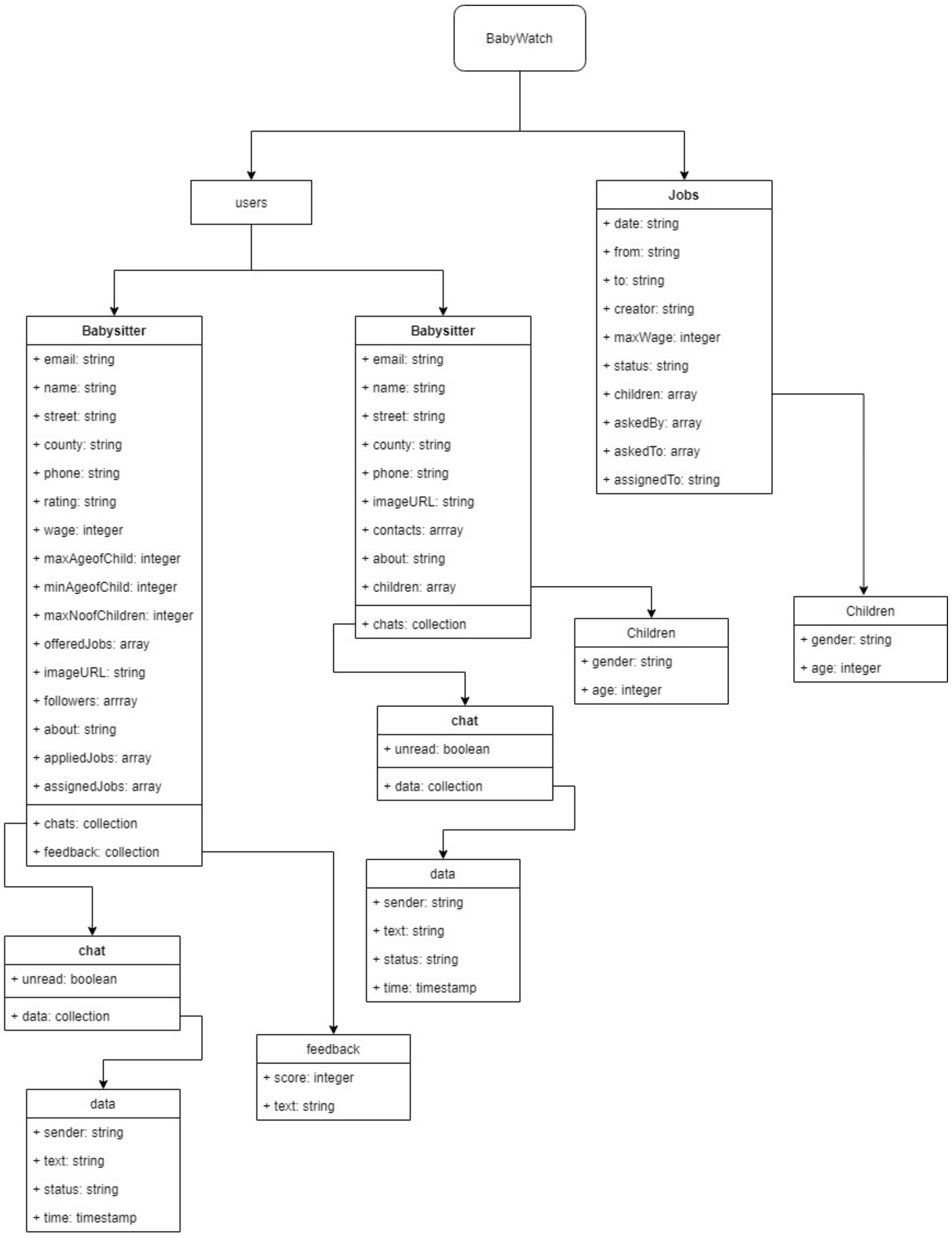


Figure 3 Data Model Illustration

4.2 Screens

The screens are the components with which the users are allowed to interact with the application. The app's flow control is managed by the UI part. The Flutter SDK provides with rich UI control structure for the development of cross platform mobile applications. As mentioned previously, the main advantage of using Flutter instead native development tools is that the application works seamlessly across multiple platforms [36] with the same code base. The main three advantages of Flutter are:

- **Fast Development:** The Flutter allows the developer to paint the app to life incredibly fast with the help of stateful hot reload function. It enables the use of rich set of incredibly customizable widgets to develop native interfaces.
- **Expressive and Flexible UI:** Flutter provides with the quick ship features, focussing on native end-user experience. The layered architecture of Flutter enables for immense customization, that leads to expressive and flexible designs and fast rendering.
- **Native Performance:** The widgets provided by flutter incorporate all platform critical differences like navigation, scrolling, fonts, and icons. The Flutter code is compiled to native ARM machine code with the help of native compilers from Dart.

All the above advantages make Flutter a good choice for developing mobile applications. Thus, the UI of the app is made using Flutter. Widgets are at the core of Flutter mechanism. Almost everything is a widget in Flutter. An example structure of flutter widget tree is shown in the figure below.

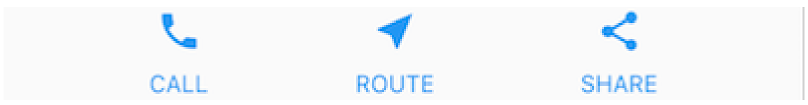


Figure 4 Example of 3 icons with a label

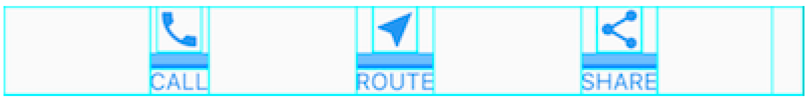


Figure 5 Example of 3 icons showing its layout.

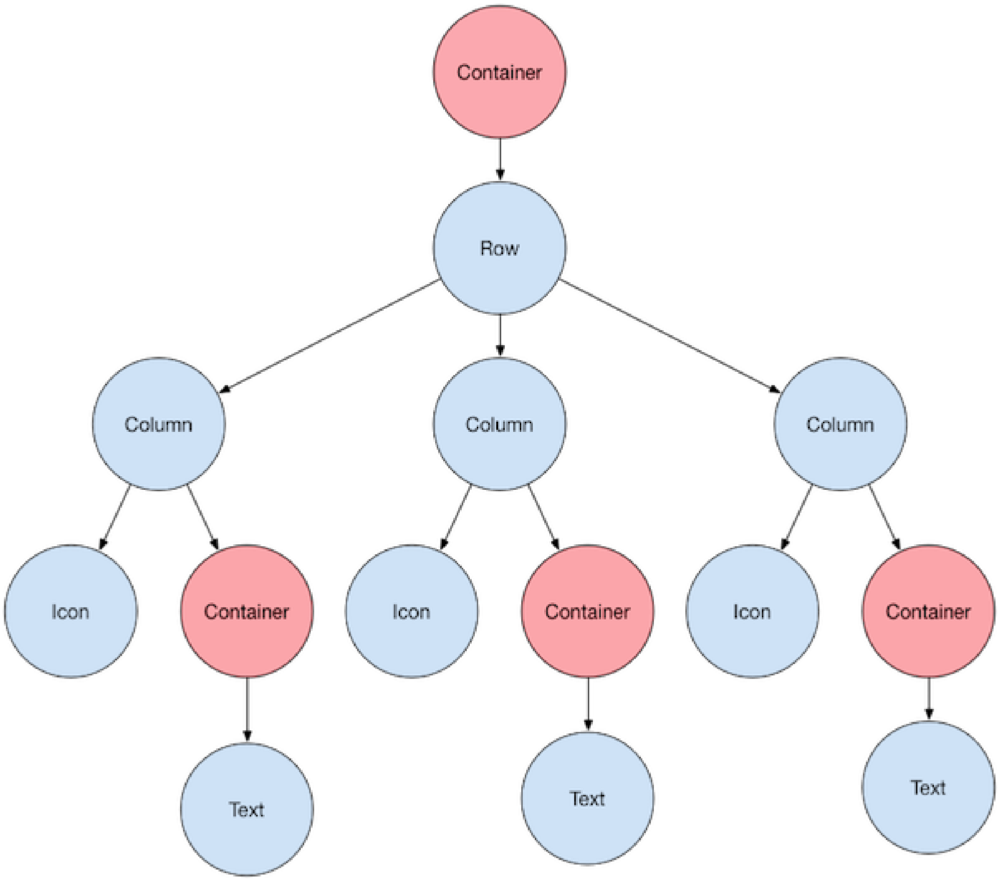


Figure 6 Flutter widget tree for the above example.

In Flutter, Navigation is handled using identifiers. In the main.dart file, each identifier is assigned to a route that takes them to a view controller. A view controller is a class that controls all the widgets and logic happening in a particular view. Some view controller takes in data, and some does not. It is with these view controllers that the app controls what appears on the screen.

The list of UI view controllers in this application are:

- AddJobScreen
- AllReviewScreen

- ChatScreen
- HomeScreen
- JobDetailScreen
- LoginScreen
- ProfileEditScreen
- ProfileScreen
- RegisterScreen
- ReviewScreen
- SelectBabysitterScreen
- UserScreen
- WelcomeScreen

In order to customize the widgets further, there are three custom widgets used in this project.

- CustomIconButton: Custom button where the developer can customize the icon shown on it.
- CustomLargeButton: Customized button where the developer can edit the text on the button, its colour and it size.
- CustomLargeTextField: Customized text field where the developer can edit the hint text and some other functionalities in the widget.

The screens folder also consists of constants.dart, which consists of the value of constants used in the application. These are mainly, colour pallets that are used to style the application in a consistent way.

4.3 Services

There are only two independent services used in this project. One of them is location services that contains a function that will trigger Google Maps to redirect to the input location. And the other one is validation, which is used for validation of inputs in the application.

Chapter 5. Implementation

The planning for implementing the application is an important part of the development stage. There are two sections to the application, the UI part where all the Flutter code exists which dictates the behaviour and look of the application and the Firebase part that holds all the data and storage and handles the authentication of users. Initially, lets look at how to integrate the Google Firebase with application and then the development of UI components and business rules.

5.1 Firebase Integration

As mentioned previously, the application uses Google Firebase to handle data and authentication. Connecting Google Firebase to a Flutter application is no easy task as it requires several steps and installing several packages to make it work.

5.1.1 Basic requirements

The most basic requirement for a Flutter app to work with Firebase is to install the firebase core plugin. This plugin, as mentioned in the dart packages section, enables the app to connect with Firebase apps. It can be installed before the usage of any other Firebase Flutter plugins. It enables basic functionalities. To install the package, the firebase_core should be added to the dependencies section in the app's pubspec.yaml file. After installing the core, the other services such as cloud firestore, firebase storage, and firebase authentication can be installed by installing the dart packages in the same way as firebase core.

5.1.2 Register the application to Android part of the system.

The Firebase need to be integrated to Android and iOS separately. When it comes to the Android part, the app needs to be registered to Android support on the Firebase console by adding the package name which is the application Id from android/app/build.gradle and App nickname. Then copy the generated configuration file, google-services.json that is generated into the app level build.gradle. Then the Firebase SDK needs to be added using some of the edits in the Google Services plugin in the android/build.gradle.

5.1.3 Register the application to iOS part of the system.

In order to register the iOS part with Firebase, there are similar steps. Initially, the app needs to be registered by adding the iOS bundle ID. Then, download the config file GoogleService-info.plist and copy it to the iOS root folder. The bundle ID can only be fetched by using a Mac and XCode.

5.2 Implementation of the Database

As mentioned previously, the app uses Firebase's Cloud Firestore as the database. It is a NoSQL database, that is simple and flexible. It stores data in the form of collections and documents. There are two parent collections used in this app, users collection and jobs collection.

5.2.1 Users Collection

The users collection stores all the data regarding the two types of users in the app, the babysitter, and the parent. Both of the user types have fields and other collections inside them.

In the case of a parent, the main data fields are:

- name
- email
- street
- county
- phone
- imageURL
- contacts
- children
- about

In addition to this, a parent would have another collection inside them called chats. Each document in the chats collection will have a Boolean field “unread” and another collection called data. Each document in the data collection will have the following fields:

- sender
- status
- text
- time

In case of a babysitter, the main data fields are:

- name
- email
- street
- county
- phone
- rating
- wage
- maxAgeofChild
- minAgeofChild
- maxNoofChildren
- offeredJobs
- imageURL
- followers
- about
- appliedJobs
- assignedJobs

In addition to this, a babysitter would have two other collections inside them called chats and feedback. The contents of the chats collection are identical to those of the parent. The feedback collection contains the following fields:

- score
- text

5.2.2 Jobs Collection

The jobs collections hold the data regarding the jobs that are created by the parents. It contains the following fields:

- date
- from
- to
- maxWage
- status
- creator
- children
- askedBy
- askedTo
- assignedTo

5.3 Implementation the UI and Logic using Flutter and Dart.

The implementation of the UI and the logic of the app using the Flutter toolkit is the most complex process in the application. Each of the view has all the widgets and other UI components as well as the logic that is incorporated embedded within the same file. However, there are other small elements that are imported from other widgets and services.

5.3.1 Welcome Screen

This is the first screen that the user screens, regardless of their role when they open the app for the first time. There are two custom buttons that allows a new user to sign up with the app or an existing user to login.

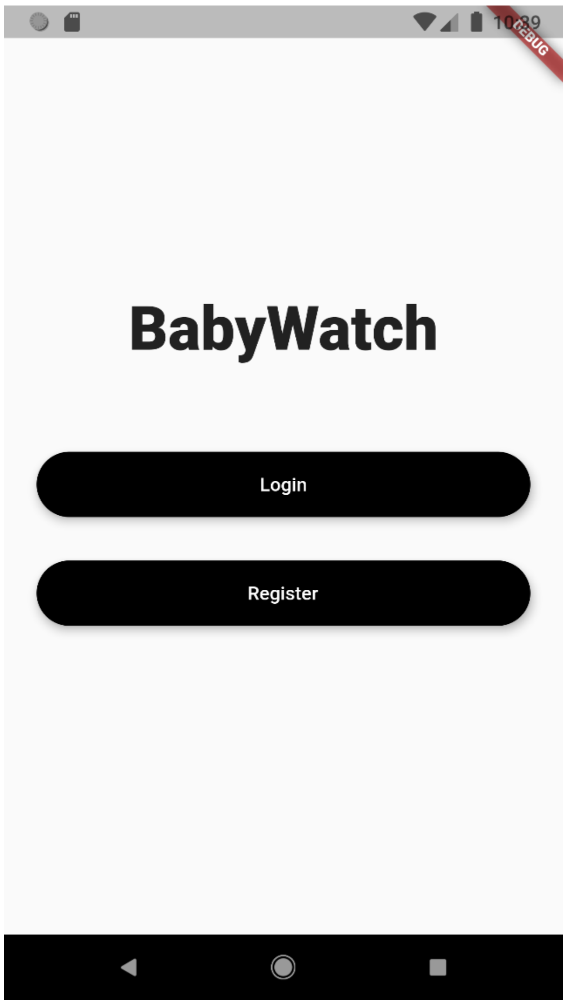


Figure 7 Welcome Screen

5.3.2 Login Screen

If the user chooses the login button in the Welcome screen, he/she is pushed to the Login Screen. This screen allows the user to use their already existing email and password to login to the application.

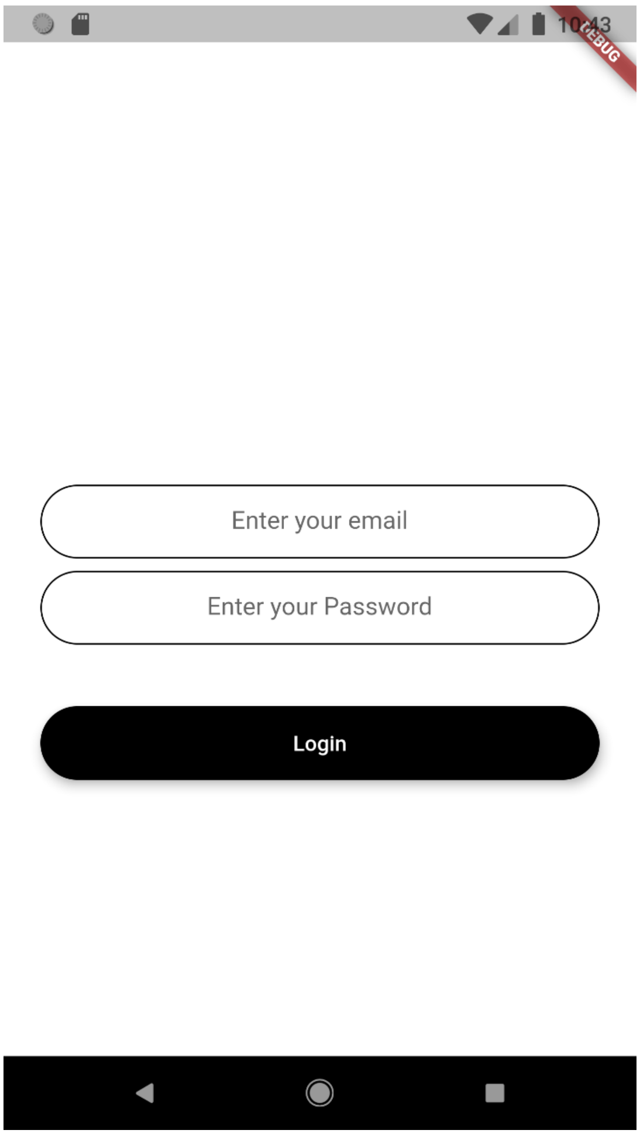


Figure 8 Login Screen

5.3.3 Register Screen

If the user chooses the Register button in the Welcome screen, he/she is pushed to the Register Screen. This view has three parts known as states. The initial state is known as email. In this state, the user is asked to input their email and password and select whether they are a babysitter or a parent.

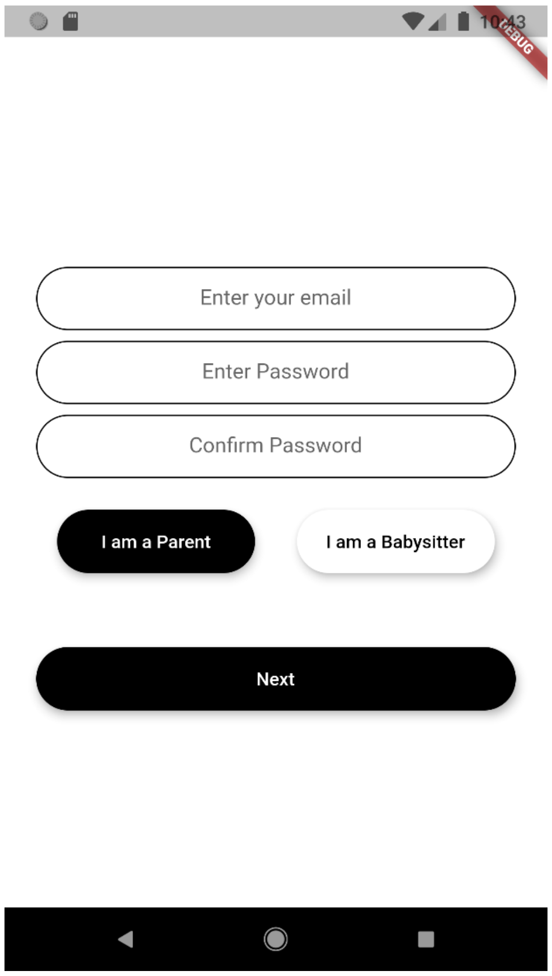


Figure 9 Register Screen in "email" state.

When the user clicks on the Next button on the email state, the state changes to "details". In this state, the user is expected to provide their personal details. The user is asked to input their name, street, county and phone number.

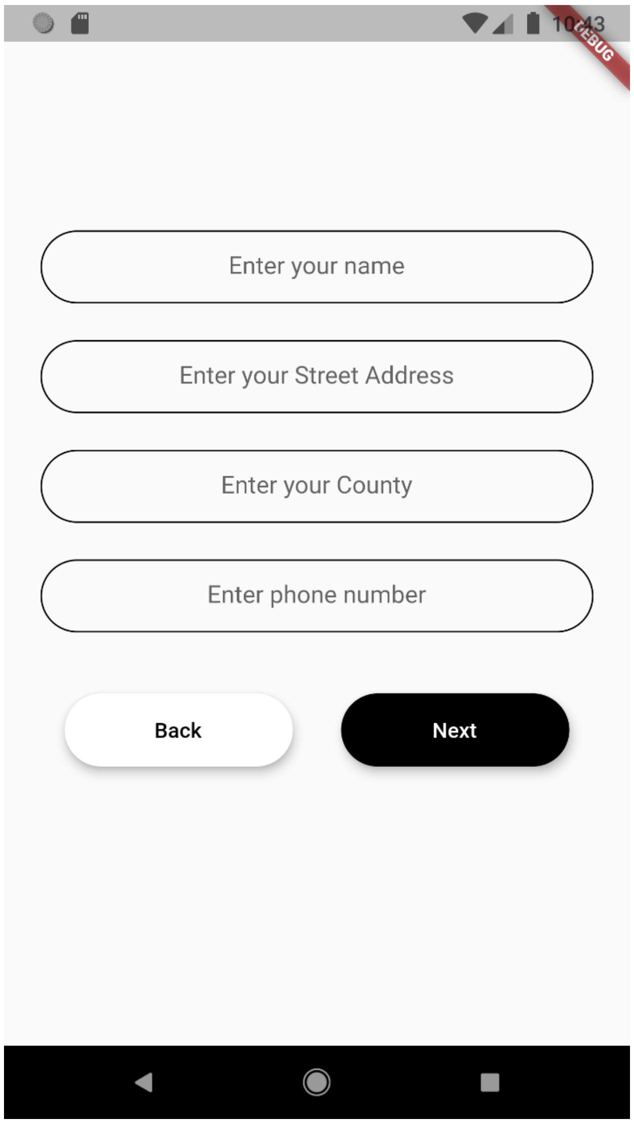


Figure 10 Register Screen in "email" state

When the user clicks on the back button, the state changes back to "email" and when they click on the Next button, the state changes to "about". This state is different for a babysitter and a parent. If the user is a babysitter, the user can provide a small description about them and then provide fields like minimum wage per hour, minimum and maximum age of the child, maximum number of children that the babysitter would like to work with.

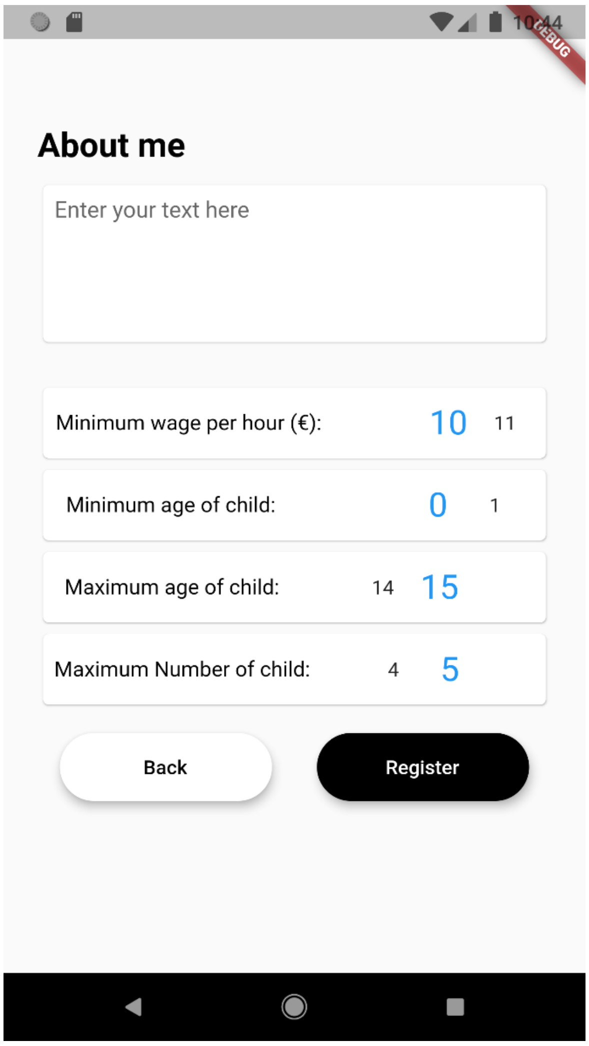

10:44
DEBUG
About me
Enter your text here
Minimum wage per hour (€): 10 11
Minimum age of child: 0 1
Maximum age of child: 14 15
Maximum Number of child: 4 5
Back
Register

Figure 11 Register Screen in "about" state for babysitter

If the user is a parent, this state is entirely different for them. Initially, similar to the babysitter, they will be asked to provide as small description about themselves and their family. Then they would be asked to add a child by providing their gender and age. After that, the age and gender are displayed, and a button is provided for the parent to delete the child. They can also add another child after this.

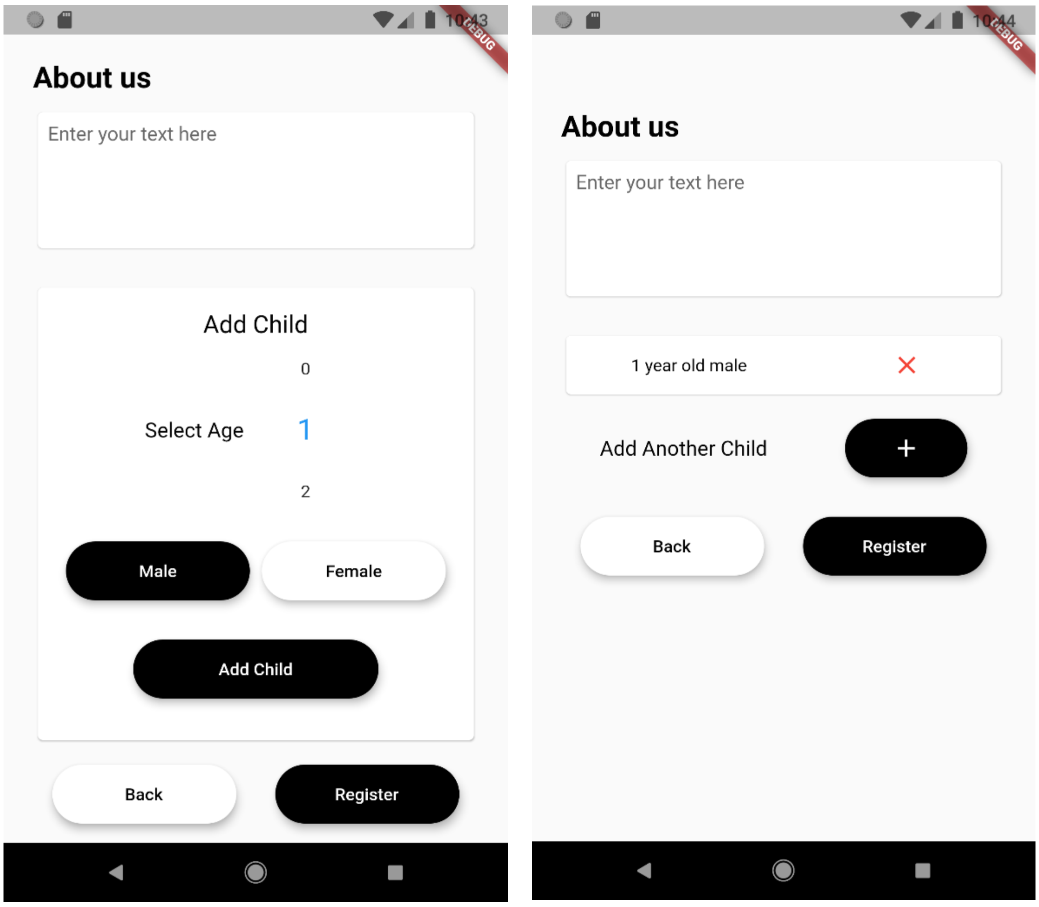
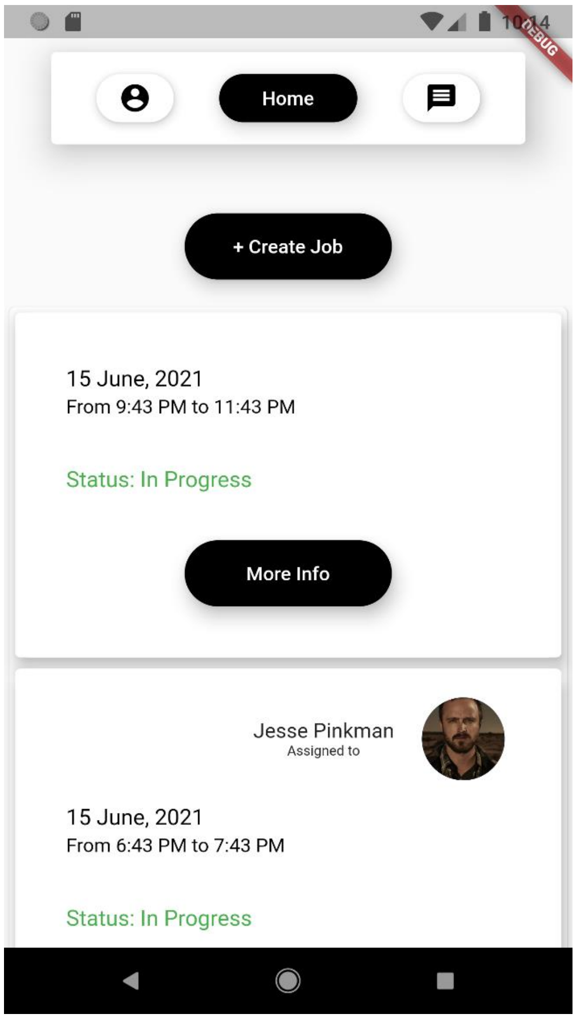


Figure 12 Register Screen in "about" state for parent

5.3.4 Home Screen

The Home Screen is, as the name suggests, the home page of the application. It differs if the user is a babysitter or a parent. It also consists of several states. The initial state is the "default" state. This state is different for a parent and a babysitter.

In case of a parent, this state allows to create jobs and edit them. All the jobs created by the user are listed in this view as cards.



15 June, 2021
From 9:43 PM to 11:43 PM

Status: In Progress

Jesse Pinkman
Assigned to

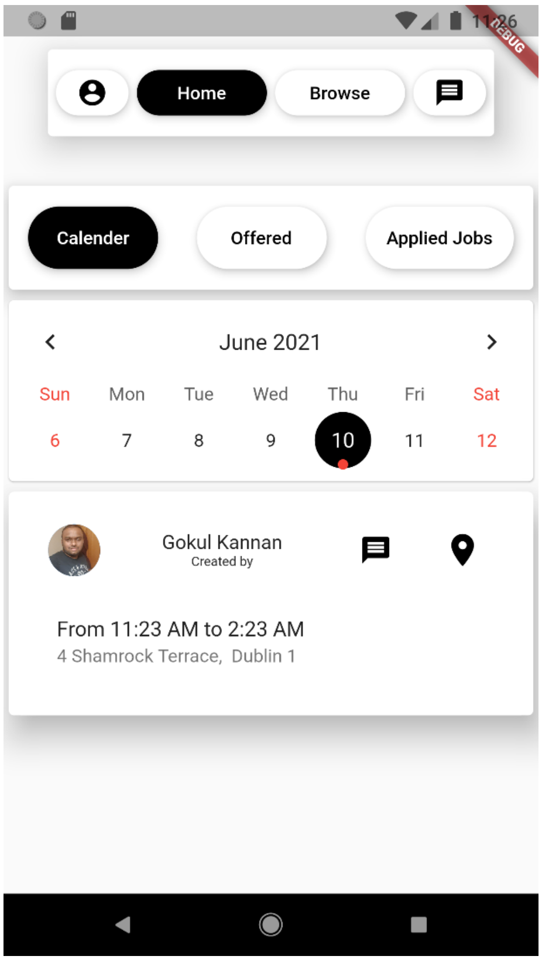
15 June, 2021
From 6:43 PM to 7:43 PM

Status: In Progress

Figure 13 Home Screen in "default" state for parent

In case of a babysitter, the default state has three tabs called Calendar, Offered and Applied Jobs.

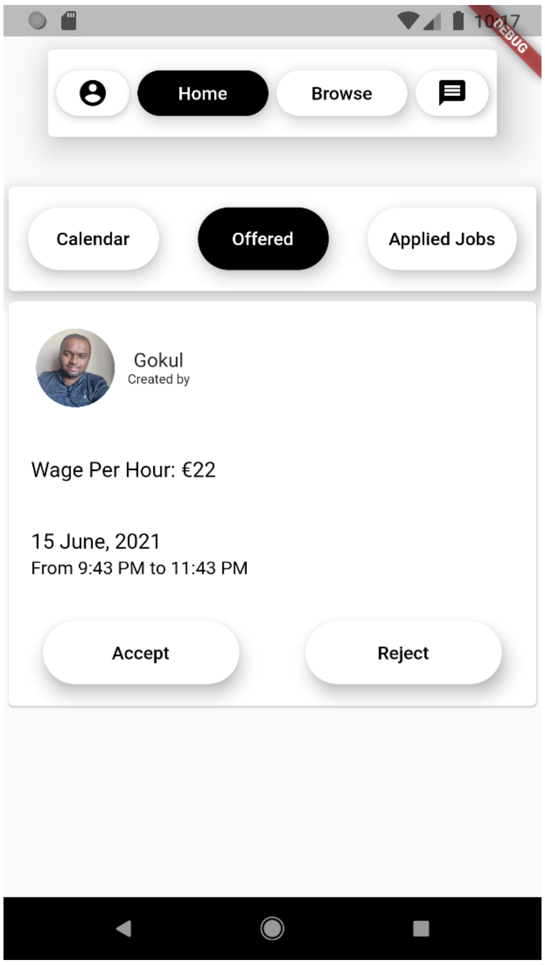
The Calendar tab consist of a Calendar widget on the weekly view and below it, the jobs the sitter has corresponding to the date that is selected in the calendar will be visible as cards. These cards contain the details about the job along with buttons that will allow the babysitter to chat with the parent or to navigate to the parent's location using the Google Maps.



From 11:23 AM to 2:23 AM
4 Shamrock Terrace, Dublin 1

Figure 14 Home Screen in "default" state for Babysitter in Calendar Tab

The offered tab consists of all the jobs that are offered to the babysitter by parents. The babysitter can either accept or reject these jobs.



Gokul
Created by

Wage Per Hour: €22

15 June, 2021
From 9:43 PM to 11:43 PM

Figure 15 Home Screen in "default" state for Babysitter in Offered Tab

The applied jobs contain the cards containing all the jobs that are applied by the babysitter. The babysitter my undo the application before the parent accepts using the delete button. Each of the card consists of the parent's name, the date and time of the job. The location of the user is not revealed until the job is assigned to the babysitter.

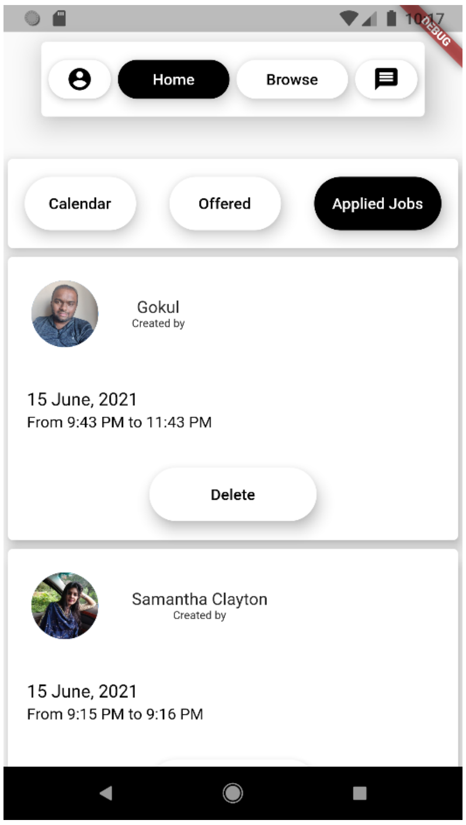
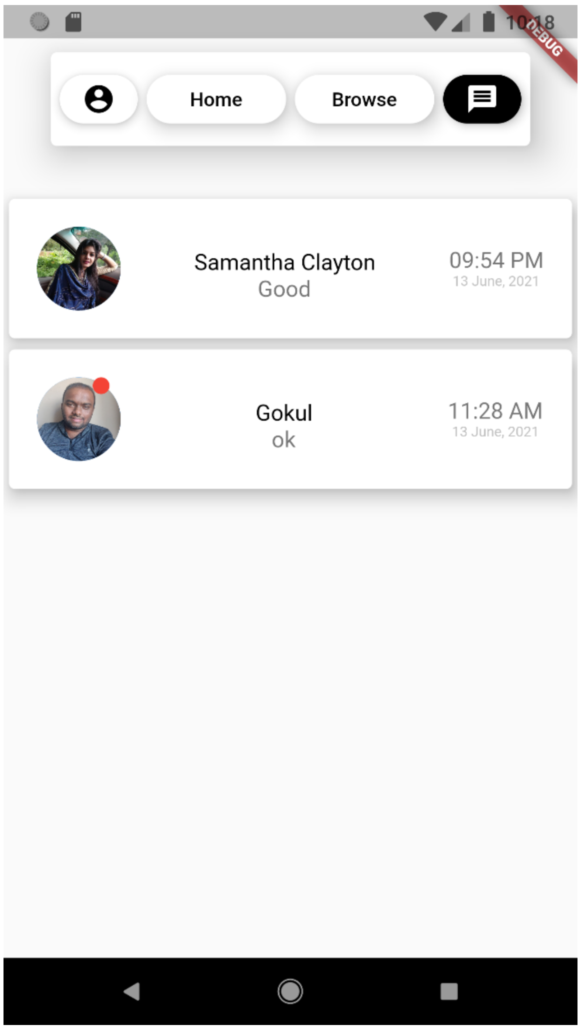


Figure 16 Home Screen in "default" state for Babysitter in Applied Jobs Tab

The second state that is common for babysitter and parent is the "message" state. It is common for both the user types and consists of list of chats the user had with other users. Each of the chat card consists of the recipient's name, photo, the last message they had and the date and time they last interacted.

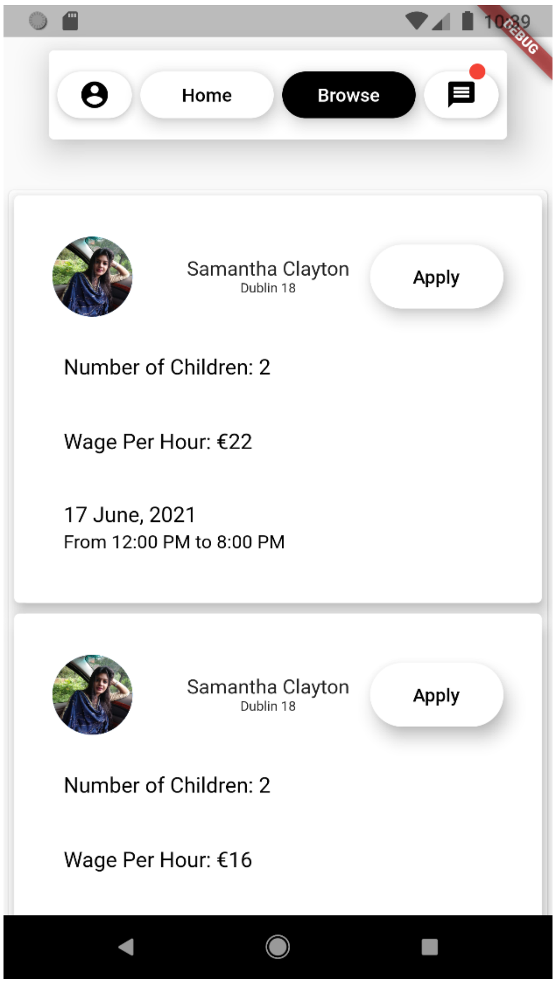


Samantha Clayton
Good
09:54 PM
13 June, 2021

Gokul
ok
11:28 AM
13 June, 2021

Figure 17 Home Screen in "message" state

If the user is a babysitter, they have an extra state called "browse". When the app is in this state, it lists all the jobs that are vacant at the moment to the babysitter. The babysitter can browse through the jobs and select those that are more suitable for him/her. Each of the job card contains the name, county and photo of the parent, the number of children they have, and the date and time of the job. The precise location is not revealed here. The babysitter can click on the apply button to apply for the job.



Samantha Clayton
Dublin 18

Number of Children: 2

Wage Per Hour: €22

17 June, 2021
From 12:00 PM to 8:00 PM

Samantha Clayton
Dublin 18

Number of Children: 2

Wage Per Hour: €16

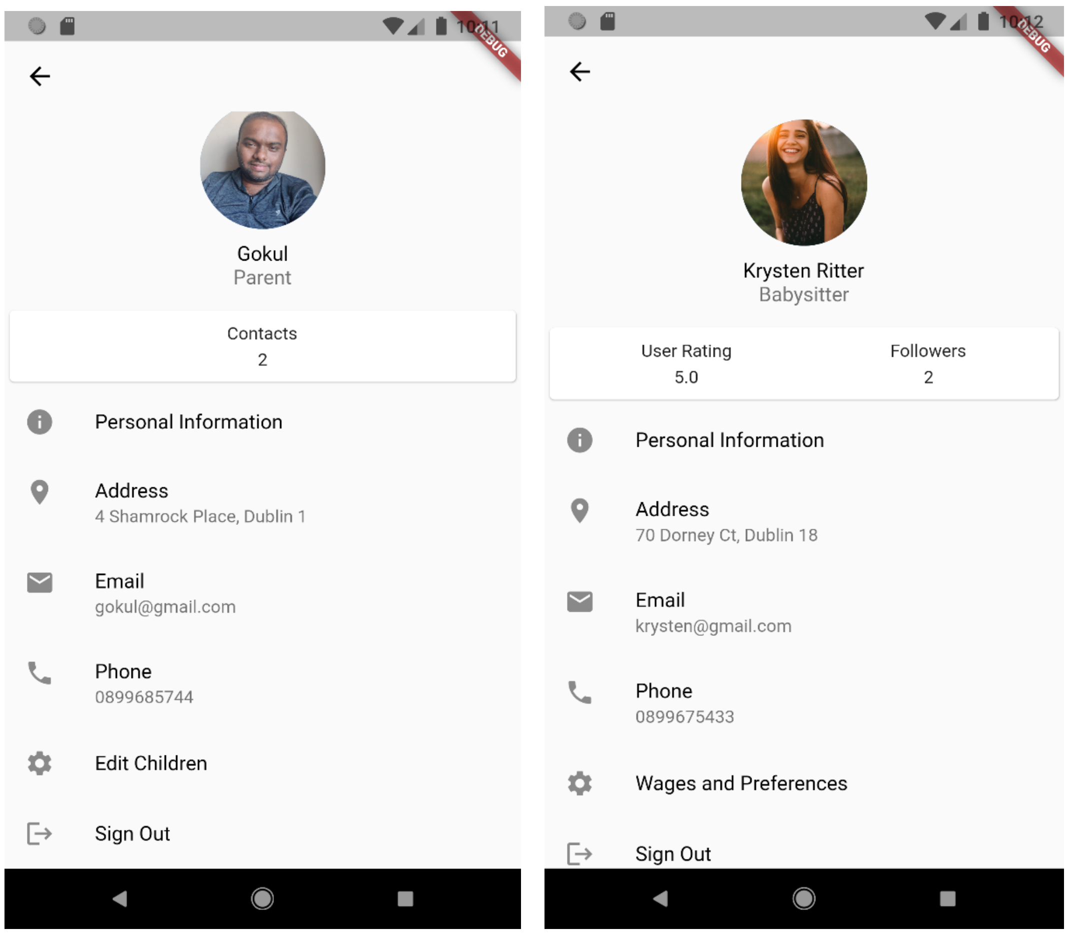
Figure 18 Home Screen in "browse" state for babysitter.

There is one more tab in the Home Screen, which is the profile tab that navigates the user to their own Profile Screen. This is not a state of the Home Screen but rather a route to another screen.

5.3.5 Profile Screen

The profile screen is where the user can view all their details. The user can view and edit their profile picture, view their address, phone number, email, and also can sign out of the application from this screen. This screen is slightly different for babysitter and parent.

For parents, they can see the number of contacts they have here and for babysitters, they can see their user rating and number of followers here. Clicking on the user details will allow user to go to Profile Edit Screen that allows the user to edit their user details.



Gokul
Parent

Contacts
2

Personal Information

Address
4 Shamrock Place, Dublin 1

Email
gokul@gmail.com

Phone
0899685744

Edit Children

Sign Out

Krysten Ritter
Babysitter

User Rating
5.0

Followers
2

Personal Information

Address
70 Dorney Ct, Dublin 18

Email
krysten@gmail.com

Phone
0899675433

Wages and Preferences

Sign Out

Figure 19 Profile Screen for Parent and Babysitter

5.3.6 Profile Edit Screen

The Profile Edit Screen is used to edit the details of the user. It is triggered from the profile screen. It needs and input called “state”. The state determines which are the fields that are displayed on the screen. The state indicates which field did the user clicked on the Profile Screen.

If the state is “personalInfo”, it means that the user has clicked the Personal Information on the Profile Screen and wants to edit it. This state allows users to edit the name and about sections of their profile.

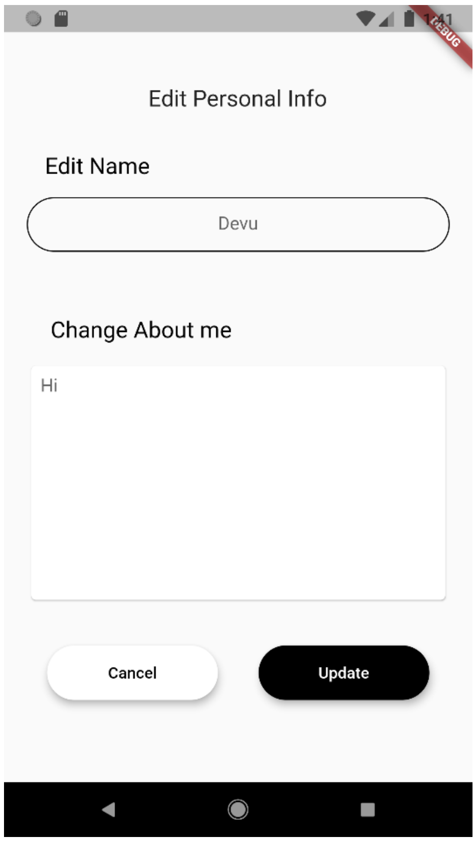


Figure 20 Profile Edit Screen for “personalInfo” state.

If the state is “address”, it means that the user has clicked the address section on the Profile Screen and wants to edit it. This section allows users to edit their street and county and they can also view it on the Google Map to make sure that their location is correct.

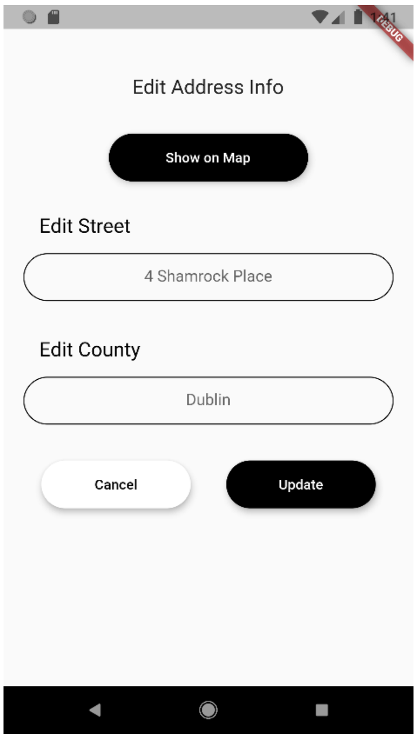


Figure 21 Profile Edit Screen for "address" state

If the state is "phone", it means that the user has clicked the phone section on the Profile Screen and wants to edit it. This section allows users to edit their phone number.

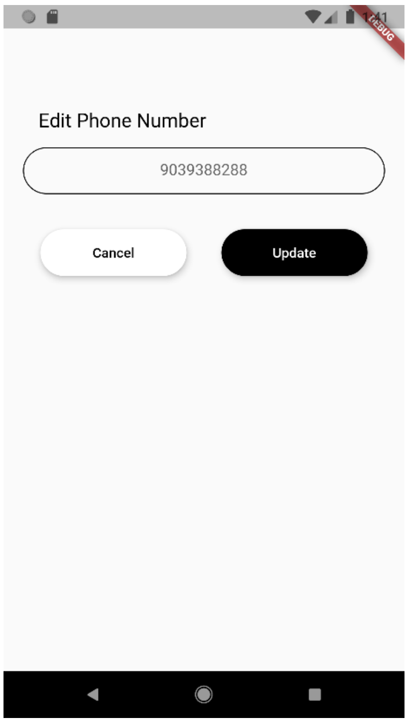
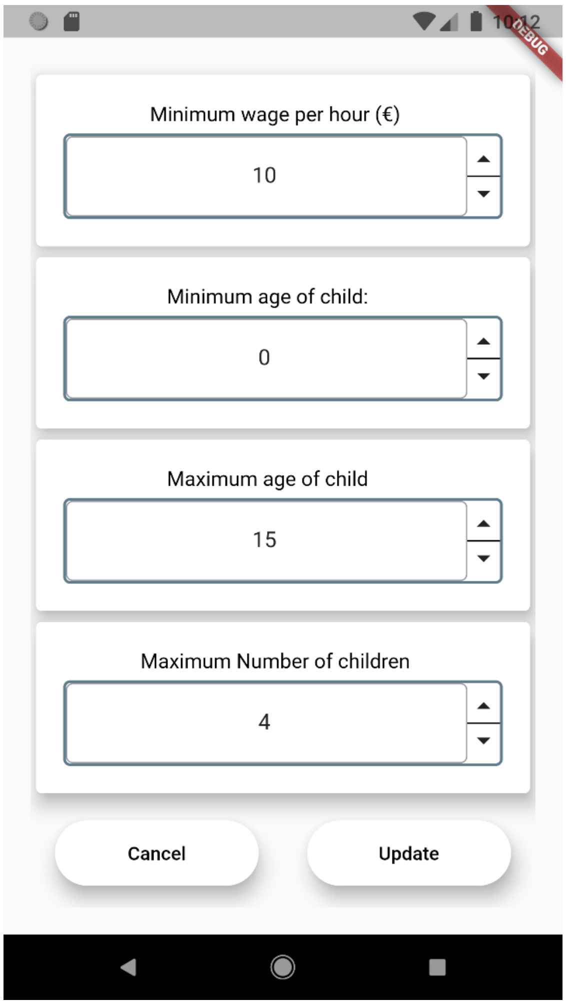


Figure 22 Profile Edit Screen for "phone" state.

Then there is the “preferences” state, which is exclusive to the babysitter, that allows the babysitter to edit their wage, number of children and other preferences. Also, the parent can edit their children with “children” state.



Minimum wage per hour (€)
10

Minimum age of child:
0

Maximum age of child
15

Maximum Number of children
4

Figure 23 Profile Edit Screen for “preferences” state.

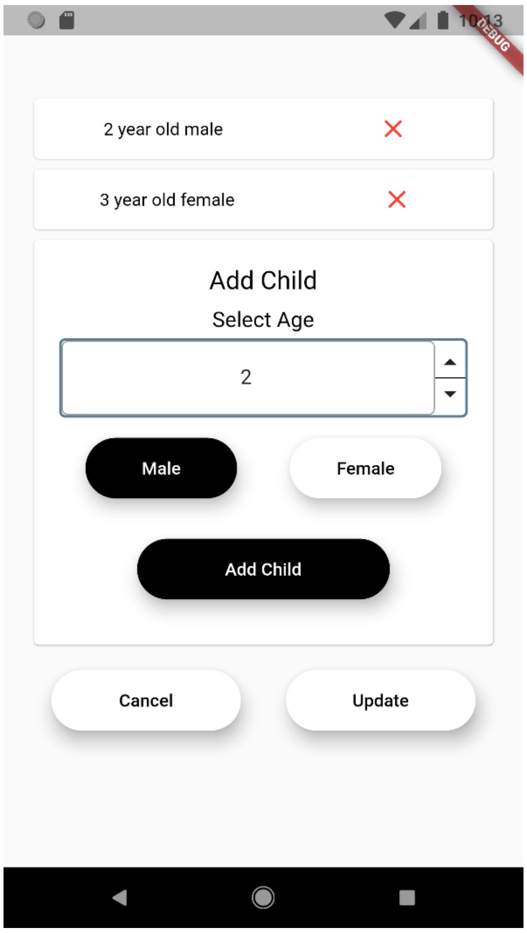
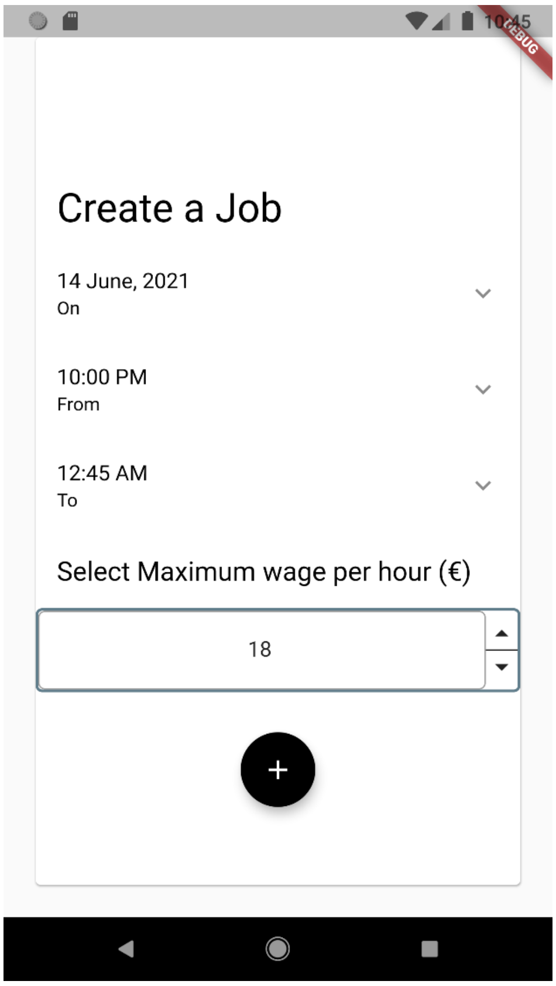


Figure 24 Profile Edit Screen for "children" state.

5.3.7 Add Job Screen

The Add Job Screen is only accessible to the parent users. It allows them to create a new job. It can be accessed by clicking the create job button in the home screen as a parent. The user needs the provide the date of the job, along with the from time and to time and the maximum amount of money they are willing to pay per hour.


Create a Job

14 June, 2021
On

10:00 PM
From

12:45 AM
To

Select Maximum wage per hour (€)

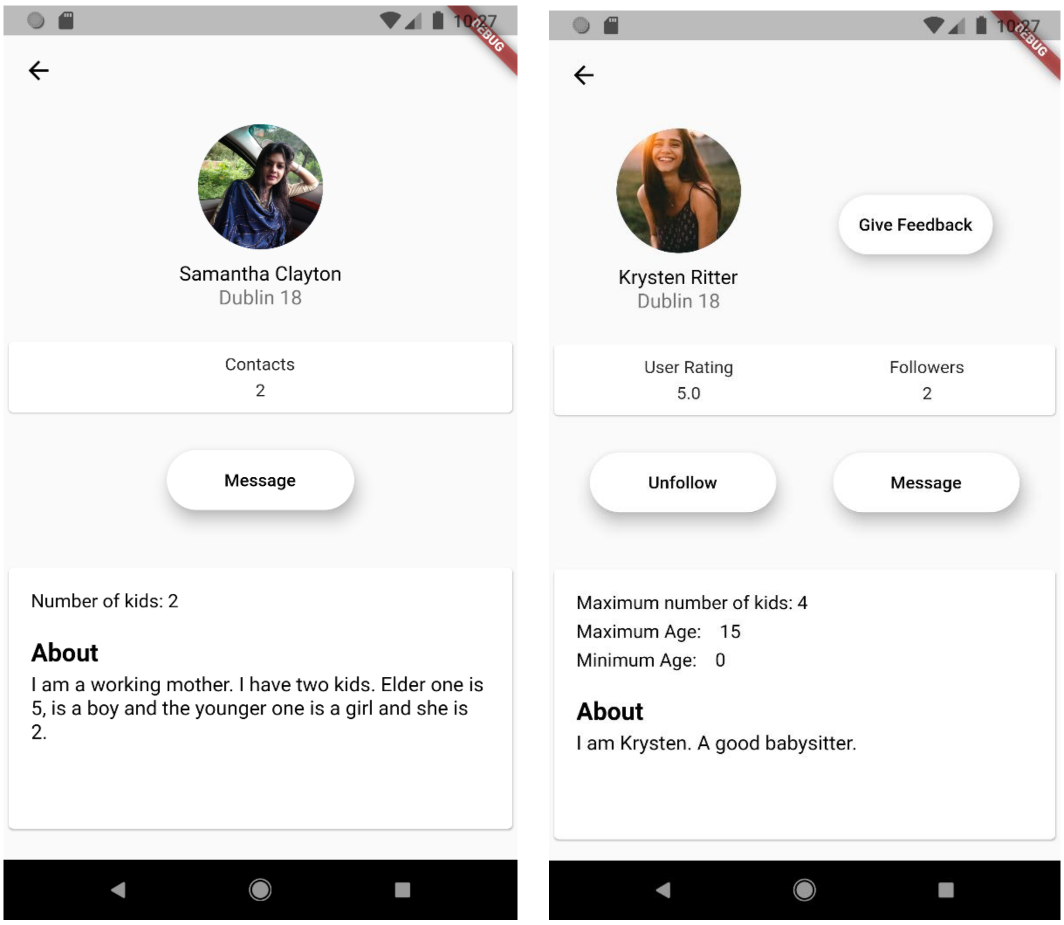
18

Figure 25 Add Job Screen

5.3.8 User Screen

The User Screen is used to show the profile of other users to the user that is logged in whether he/she is a parent or a babysitter. The screen is different for both user types as some widgets are only available to babysitters. In case of babysitters, the screen consists of the profile picture, user rating, number of followers, maximum and minimum age, the maximum number of kids that the babysitter can work with, the self-description of the babysitter and buttons for follow/unfollow the babysitter, message the babysitter and to write a review on the babysitter. While in case of a parent, the inputs for giving feedback and follow or unfollow is absent.

Instead of maximum and minimum values, it shows the number of kids the user has and instead of number of followers, it shows the number of contacts.



Number of kids: 2

About

I am a working mother. I have two kids. Elder one is 5, is a boy and the younger one is a girl and she is 2.

Maximum number of kids: 4
Maximum Age: 15
Minimum Age: 0

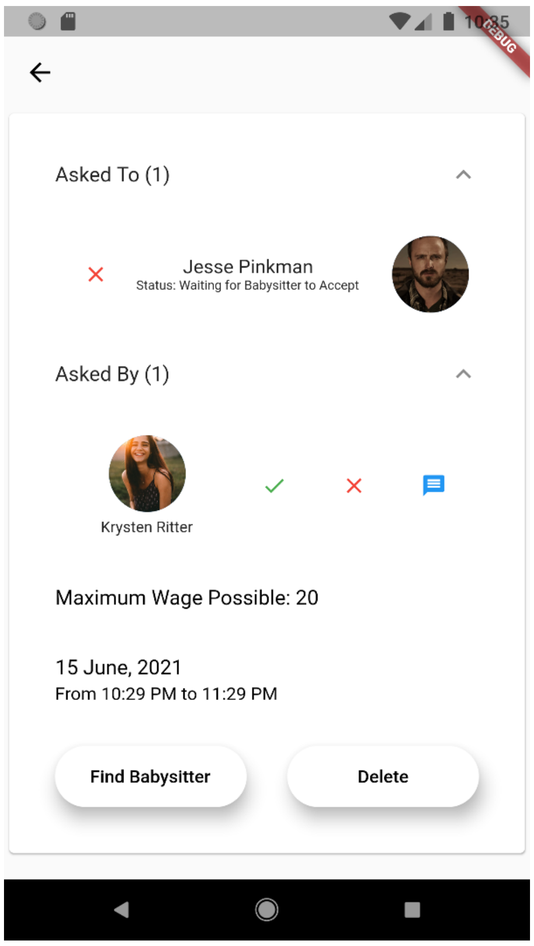
About

I am Krysten. A good babysitter.

Figure 26 User Screen for Parent and Babysitter

5.3.9 Job Detail Screen

The Job Detail Screen is exclusive for parent users. It allows the parents to manage an individual job. It shows the parent all the details about the job they created, who are the babysitters he/she asked for the job or the ones who had applied for the job. The parent can accept or reject the babysitters or can contact them via chat. The parent can click the Find Babysitter button and find a babysitter for the job or delete the job entirely. It the job is assigned to a babysitter and the job is finished; the parent can click on the Give Feedback button that will take them to the Review Screen.


Asked To (1)

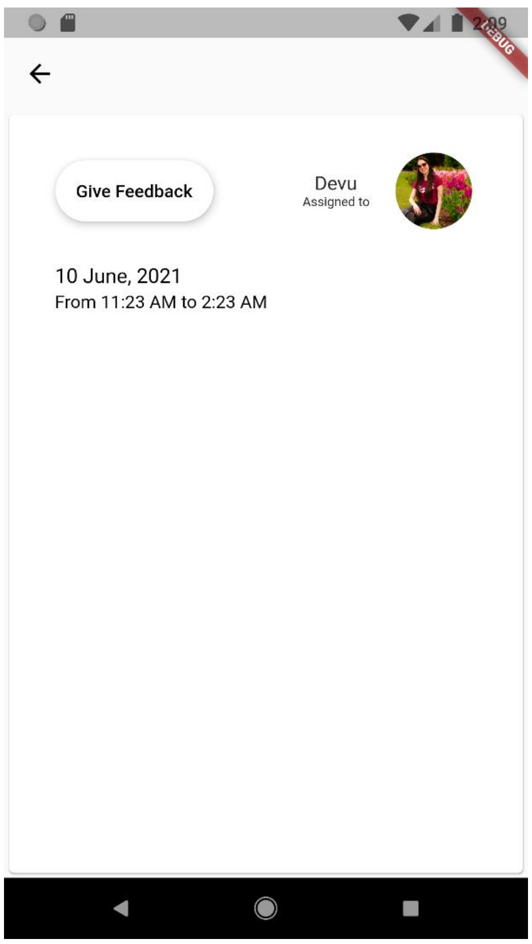
Jesse Pinkman
Status: Waiting for Babysitter to Accept

Asked By (1)

Krysten Ritter

Maximum Wage Possible: 20

15 June, 2021
From 10:29 PM to 11:29 PM


Devu
Assigned to

10 June, 2021
From 11:23 AM to 2:23 AM

Figure 27 Job Detail Screen

5.3.10 Chat Screen

The Chat Screen is the one that allows the users to message other users. The users messages are fetched using a Stream Builder and is updated in real time. The screen shows the message that is send by the two users along with the time stamp. It also consists of a text field that the user can type a message. Just like normal messengers, it has the send, delivered and read marker.

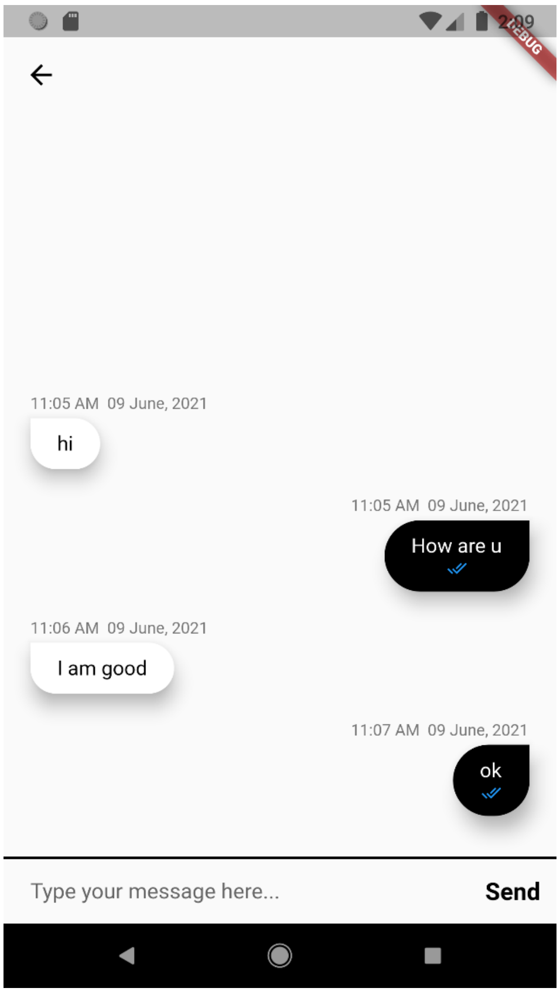

11:05 AM 09 June, 2021
hi
11:05 AM 09 June, 2021
How are u
11:06 AM 09 June, 2021
I am good
11:07 AM 09 June, 2021
ok

Figure 28 Chat Screen

5.3.11 Review Screen

The Review Screen is the one that allows parents to give feedback to the babysitter. The parents can give a rating out of five stars and provide a description that explains their score. If the user had previously rated the babysitter before, the description would show up on the description field.

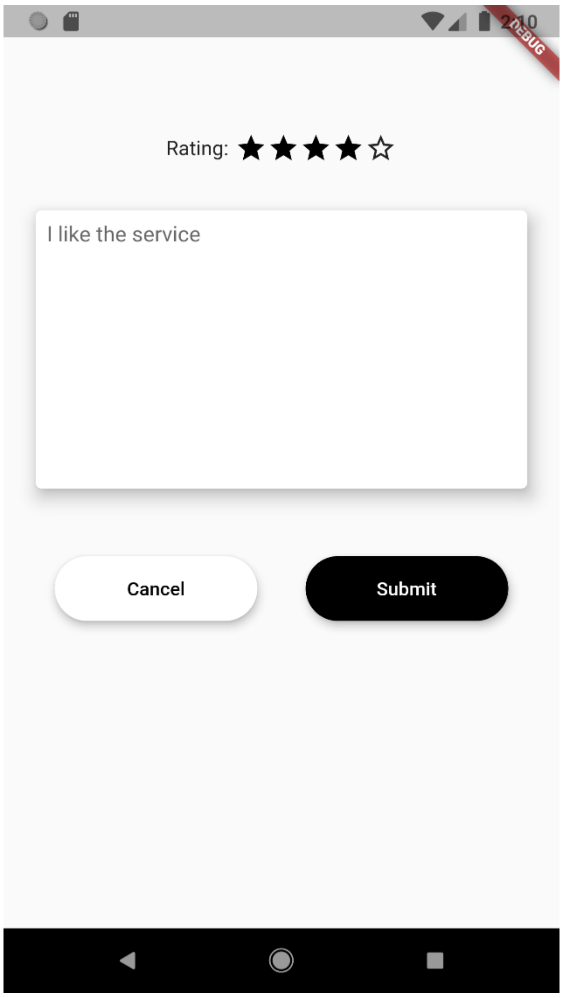
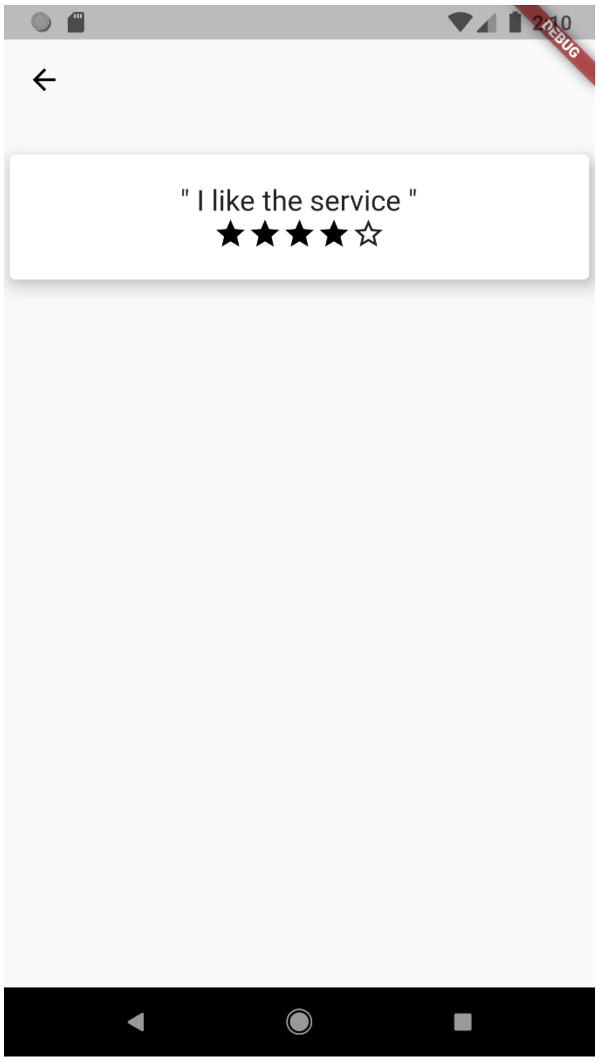


Figure 29 Review Screen

5.3.12 All Review Screen

The All Review Screen is accessible to user from the Profile Screen or the User Screen. It is exclusive the babysitter user as they are the ones who have user rating. However, parents can also see the reviews through the User Screen. It just shows all the cards that shows the text and the rating score of the babysitter.


" I like the service "

★★★★☆

Figure 30 All Review Screen

5.3.13 Select Babysitter Screen

The Select Babysitter Screen allows the parents to select a babysitter for the job. It fetches all the babysitters that satisfies the criteria such as the maximum wage, number of children etc. and displays them as cards. The cards show each babysitters profile picture, their rating score, and their cost per hour in euros and the button to Ask them for doing the job.

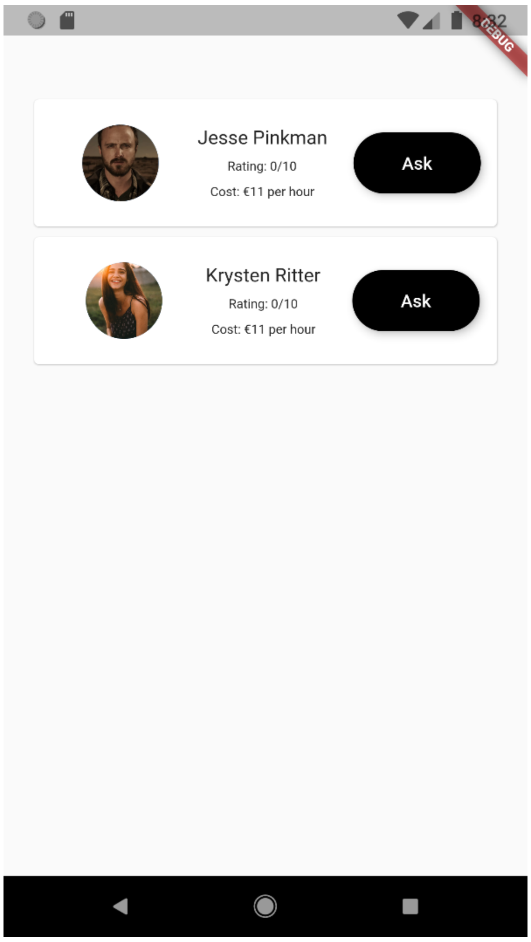


Figure 31 Select Babysitter Screen

Chapter 6. Testing and Evaluation

The testing of the app is one of the important phases in the development of the application. It makes sure that the app is of good quality and allows troubleshooting of issues that the consumer might be experiencing when the app is released to the public.

All Flutter applications is developed for both iOS and Android with the same codebase. However, due to unavailability of a MacBook and iOS device, the testing on iOS cannot be done. The application at the moment is only tested for Android. The application was tested on a Dell XPS 15 laptop that is running Windows 10 Home. The app is tested both in Android Emulator provided by the Android Studio, which was an emulation of Google Nexus 5, and a physical Android device, OnePlus 8 Pro.

Initially, the app was tested extensively using the Android emulator. A list of used cases was made. The main intention was to make sure that everything is working fine, and the app does not break on user input. Also, the intention is to eliminate bloat data in the database. The approach was to make sure that all the inputs were working perfectly and the filters for the babysitters and parents are precise.

After the initial testing using the emulator, the testing was done in a physical Android smart phone, OnePlus 8 pro, to make sure the app was working as intended. There were some UI overflows detected in the smart phone and was fixed.

Chapter 7. Conclusion and Future Work

After all the extensive testing and evaluation, the app is ready for deployment. The application aimed at helping parents and babysitters to find each other easily. It was to provide all the necessary tools for parents to find a babysitter for their child and for babysitter, it makes finding the job easier.

As mentioned in previous chapter, the testing is only done on the Android platform. Also, the firebase is not integrated with iOS part of the application as it requires to be run on an Apple device. Thus, firebase is only connected to the Android part. However, that should be the first thing to do in the list of future improvements.

Also, there are several improvements that can be done to make the app more useful by adding more features such as integration with Google Cloud API and use Google Maps incorporated in the app and have features that will allow users to navigate to the users location without leaving the app. Another improvement is to use the tools provided by the Flutter to make the app more fluid and easier to use. The validation is done only in a basic stage and should be incorporated with advanced validation. There needs to be features such as the ability of the user to reset the password, allowing the babysitters to provide an availability schedule which automatically dictates whether a job is suitable for them and suggest only those. The browsing section and select babysitter section needs to have filters that can change the cards displayed dynamically.

The application is developed for only mobile devices. However, it can be expanded to web as a web application as well.

References

1. Bromer, J. and Henly, J., 2004. Childcare as family support: Caregiving practices across child care providers. [online] Research Gate. Available at: < https://www.researchgate.net/publication/223065362_Child_care_as_family_support_Caregiving_practices_across_child_care_providers > [Accessed 23 May 2021].
2. Smith, K. (2000). Who's minding the kids? Child care arrangements: Fall 1995. Current population reports, P70-70, pp 1 – 27. Washington, DC: US Census Bureau
3. Just-eat.ie. 2021. Just-Eat.ie. [online] Available at: < https://www.just-eat.ie/ > [Accessed 25 May 2021].
4. Deliveroo.co.uk. 2021. Deliveroo. [online] Available at: < https://deliveroo.co.uk/ > [Accessed 25 May 2021].
5. Uber. 2021. Uber Ireland. [online] Available at: < https://www.uber.com/ie/en/ > [Accessed 25 May 2021].
6. Free-now.com. 2021. The new mytaxi - Europe's No.1 taxi app. [online] Available at: < https://free-now.com/ie/ > [Accessed 25 May 2021].
7. Shettima, M., Sani Jato, A., Mohammed, A. and Ali, T., 2018. (PDF) DESIGN AND IMPLEMENTATION OF AN ONLINE STUDENT ROOM AVAILABILITY SYSTEM. [online] ResearchGate. Available at: < https://www.researchgate.net/publication/334480663_DESIGN_AND_IMPLEMENTATION_OF_AN_ONLINE_STUDENT_ROOM_AVAILABILITY_SYSTEM > [Accessed 26 May 2021].
8. Chatterjee, N., Chakraborty, S., Decosta, A. and Nath, A., 2018. *Real-time Communication Application Based on Android Using Google Firebase*. [online] Research Gate. Available at: < https://www.researchgate.net/publication/324840628_Real-time_Communication_Application_Based_on_Android_Using_Google_Firebase > [Accessed 26 May 2021].
9. Tashildar, A., Shah, N., Gala, R., Giri, T. and Chavhan, P., 2020. *APPLICATION DEVELOPMENT USING FLUTTER*. [online] Irjmets.com. Available at: <

https://irjmets.com/rootaccess/forms/uploads/APPLICATION%20DEVELOPMENT%20USING%20FLUTTER%20.pdf > [Accessed 26 May 2021].

10. Flutter.dev. 2021. *Flutter architectural overview*. [online] Available at: < https://flutter.dev/docs/resources/architectural-overview > [Accessed 2 June 2021].
11. Krishnan, H., Elayidom, M. and Krishnan, T., 2016. *MongoDB – a comparison with NoSQL databases*. [online] ResearchGate. Available at: < https://www.researchgate.net/publication/327120267_MongoDB_-_a_comparison_with_NoSQL_databases > [Accessed 2 June 2021].
12. MongoDB. n.d. *Advantages of NoSQL*. [online] Available at: < https://www.mongodb.com/nosql-explained/advantages > [Accessed 2 June 2021].
13. Gyorodi, R., Georgian, V., Zmaranda, D. and Gyorodi, C., 2017. *A Comparative Study between Applications Developed for Android and iOS*. [online] ResearchGate. Available at: < https://www.researchgate.net/publication/321496429_A_Comparative_Study_between_Applications_Developed_for_Android_and_iOS > [Accessed 3 June 2021].
14. Sholichin, F., Adham Isa, M., Firdaus Harun, M. and Abd Halim, S., 2019. *Review of iOS Architectural Pattern for Testability, Modifiability, and Performance Quality*. [online] ResearchGate. Available at: < https://www.researchgate.net/publication/335192719_Review_of_iOS_Architectural_Pattern_for_Testability_Modifiability_and_Performance_Quality > [Accessed 3 June 2021].
15. Developer.amazon.com. n.d. *Fire OS Overview | Amazon Fire TV*. [online] Available at: < https://developer.amazon.com/docs/fire-tv/fire-os-overview.html > [Accessed 3 June 2021].
16. Reactnative.dev. n.d. *Introduction · React Native*. [online] Available at: < https://reactnative.dev/docs/getting-started > [Accessed 3 June 2021].
17. Dart packages. 2021. *cupertino_icons | Dart Package*. [online] Available at: < https://pub.dev/packages/cupertino_icons > [Accessed 4 June 2021].
18. Dart packages. 2021. *firebase_core | Flutter Package*. [online] Available at: < https://pub.dev/packages/firebase_core > [Accessed 4 June 2021].
19. Dart packages. 2021. *firebase_auth | Flutter Package*. [online] Available at: < https://pub.dev/packages/firebase_auth > [Accessed 4 June 2021].
20. Dart packages. 2021. *cloud_firestore | Flutter Package*. [online] Available at: < https://pub.dev/packages/cloud_firestore > [Accessed 4 June 2021].

21. Dart packages. 2019. *modal_progress_hud | Flutter Package*. [online] Available at: < https://pub.dev/packages/modal_progress_hud > [Accessed 4 June 2021].
22. Dart packages. 2021. *provider | Flutter Package*. [online] Available at: < https://pub.dev/packages/provider > [Accessed 4 June 2021].
23. Dart packages. 2021. *http | Dart Package*. [online] Available at: < https://pub.dev/packages/http > [Accessed 4 June 2021].
24. Dart packages. 2021. *dio | Dart Package*. [online] Available at: < https://pub.dev/packages/dio > [Accessed 4 June 2021].
25. Dart packages. 2021. *table_calendar | Flutter Package*. [online] Available at: < https://pub.dev/packages/table_calendar > [Accessed 4 June 2021].
26. Dart packages. 2021. *rflutter_alert | Flutter Package*. [online] Available at: < https://pub.dev/packages/rflutter_alert > [Accessed 4 June 2021].
27. Dart packages. 2021. *numberpicker| Flutter Package*. [online] Available at: < https://pub.dev/packages/numberpicker > [Accessed 4 June 2021].
28. Dart packages. 2021. *image_picker| Flutter Package*. [online] Available at: < https://pub.dev/packages/image_picker > [Accessed 4 June 2021].
29. Dart packages. 2021. *firebase_storage| Flutter Package*. [online] Available at: < https://pub.dev/packages/firebase_storage > [Accessed 4 June 2021].
30. Dart packages. 2021. *permission_handler| Flutter Package*. [online] Available at: < https://pub.dev/packages/permission_handler > [Accessed 4 June 2021].
31. Dart packages. 2021. *url_launcher| Flutter Package*. [online] Available at: < https://pub.dev/packages/url_launcher > [Accessed 4 June 2021].
32. Firebase. 2021. *Documentation | Firebase*. [online] Available at: < https://firebase.google.com/docs > [Accessed 4 June 2021].
33. Firebase. 2021. *Cloud Firestore | Firebase*. [online] Available at: < https://firebase.google.com/docs/firestore > [Accessed 5 June 2021].
34. Firebase. 2021. *Firebase Authentication*. [online] Available at: < https://firebase.google.com/docs/auth > [Accessed 5 June 2021].
35. Firebase. 2021. *Cloud Storage for Firebase*. [online] Available at: < https://firebase.google.com/docs/storage > [Accessed 5 June 2021].
36. Flutter.dev. 2021. *Layouts in Flutter*. [online] Available at: < https://flutter.dev/docs/development/ui/layout > [Accessed 8 June 2021].