

**“SUPERCLEANER”  
NATIVE ANDROID MOBILE APPLICATION  
FINAL REPORT**

**NAME:AHMAD SABEH-MURPHY**

**ID:11713929**

**LECTURER: COLM DUNPHY**

**MODULE: PROJECT**

**COURSE: HIGHER DIPLOMA IN COMPUTER SCIENCE**

<b>Abstract.....</b>	<b>3</b>
<b>1. Introduction.....</b>	<b>3</b>
1.1 Intro to Document.....	3
1.2 Project Background.....	3
1.3 Project Requirements.....	4
1.4 Project methodology.....	4
<b>2. Project Objectives .....</b>	<b>5</b>
<b>3 Technology .....</b>	<b>5</b>
3.2Development .....	5
3.3 Back end Technology.....	5
3.4 Payment Technology.....	6
3.5 Learning Resources .....	7
<b>4. Implementation.....</b>	<b>8</b>
4.1 Introduction.....	8
4.2.1 Data Modeling .....	8
4.2.2 UML Diagram.....	8
4.3 App Icon.....	11
4.4 Splash Screen.....	12
4.5 Login .....	13
4.6 Registration.....	14
4.7 Forgot Password.....	15
4.8 Data Validation.....	16
4.9 Complete Profile.....	17
4.10 Base Activity/Fragment.....	18
4.11 Settings.....	19
4.11.1 Add/Edit/Delete Address.....	19
4.12 Navigation.....	20
4.13 Add a Service/Service Description.....	22
4.14 Add to Cart/View Cart.....	24
4.15 Check out.....	24
4.16 Selecting Date and time .....	25
4.17 Payment.....	26

4.18 Reservations Fragment.....	28
4.19 Reservation Details.....	29
5 Critical Self Review/Analysis.....	31
5.1 What I learned.....	31
5.2 What direction the project might be taken if more time was available.....	31
5.3 Problems.....	31
6 Bibliography.....	32
7. Glossary & Abbreviations.....	34
8. Declaration of authenticity.....	35

## Abstract

SuperCleaner is a mobile application designed to simplify the process of booking valuable services , in this case cleaning services, for a small enterprise. The app offers an easy-to-use platform for users to reserve services in their local area and provides an intuitive dashboard for the enterprise to manage and administer the services they offer, a shopping cart, where users can select and purchase multiple services, as well as a payment gateway to complete transactions. The app also provides a user-friendly interface to view reservation history, including completed, ongoing, and cancelled reservations.

## 1. Introduction

### 1.1 Introduction to the Document

The following document is my final report on the state of the mobile application that I am developing called SuperCleaner for the Final Year Project for the Higher Diploma in Computer Science.

Throughout the course of the document I will be using terms that will have an explanation and expanded upon, and citation in section 6 of the document that is the Glossary & Abbreviations section.

I will reference terms explained and expanded in the glossary in the following way:

Example: Security: the app should have secure payment processing and data storage, and should comply with relevant security standards, such as PCI DSS (1).

The term PCI DSS will be term number 1 in the Glossary with its relevant brief explanation.

Also, I will be using the term App to refer to the SuperCleaner mobile app for brevity.

### 1.2 Project Background

The germ of this project began from a personal project that is a functioning business that I contributed to in the past. I was an admin, web designer, and digital marketer at Flashwash.ie(FlashWash, 2022).

Due to what I witnessed during the growth of the Flashwash business, and the knowledge gained from the course, the idea of developing an e-commerce application does not seem so daunting anymore.

SuperCleaner will be an android native mobile app where a user can reserve a cleaning for their car, office, or home. After signing up and creating a profile they would select the service they want, choose a timeslot, make a payment, and possibly leave a review after the service is completed. Also, the user would be able to view past, current, and upcoming reservations.

The vision for SuperCleaner to be a product marketed to small business owners who want to automate their customer to business interaction, which could allow the business owner to use best practices in marketing and promotion to scale sales in the local area that they operate in.

Furthermore, a clear record of sales could allow the small business to secure further investment to invest in equipment or more employees.

The SuperCleaner app will emulate an app for a company called Helping(Helpling, 2022) , but instead of being a marketplace to recruit, advertise, conduct payments the scope of the project will be the local area of where the small business owner wants to conduct business in.

I believe if the project is successful I could use the knowledge gained to potentially market my mobile development services to local or foreign small business for additional income , and to add to my portfolio of projects while applying to jobs.

### **1.3 Project Requirements**

A useful eCommerce application must be an engaging shopping environment for app users. When consumers are shopping online, they want to do it quickly and easily (Vilmate, 2020). A sleek and user-friendly mobile app must include the following:

1. User management: the app should allow users to create an account, log in, edit their profile information, and reset their password.
2. Product catalogue: the app should have a catalogue of services, including service details, images, and pricing information.
3. Shopping cart: the app should have a shopping cart where users can add, remove, and view items before checkout.
4. Payment processing: the app should allow users to make secure payments using credit cards, PayPal, or other payment methods.
5. Product management: the app should allow users to view their reservation history and reservation details, including completion information and payment status.
6. Push notifications: the app should provide push notifications to users regarding their reservation status, reservation updates, and promotional offers.
7. Search and filtering: the app should provide a search functionality that allows users to search for products by keyword or category, and filter results by price, color, size, and other criteria.
8. Customer service: the app should provide a way for users to contact customer service for assistance with reservations, refunds, and other inquiries.
9. Product reviews: the app should allow users to write and view product reviews and ratings.
10. Security: the app should have secure payment processing and data storage, and should comply with relevant security standards, such as PCI DSS (1).
11. Analytics: the app should track user behaviour and provide analytics to help the business understand user behaviour and improve the app's performance.

### **1.5 Project Methodology**

I decided to use the agile methodology to manage the development process of my Android mobile application. This approach allowed me to break down the project into smaller, more manageable tasks and prioritize the most valuable features to deliver value to the end-user.

One of the reasons I chose to use the agile methodology was that it allowed me to be more responsive to changes in user requirements or market conditions. By regularly reviewing and updating the project backlog, I was able to adjust my priorities and make changes as needed. This helped me avoid wasting time and resources on features that were not valuable to the end-user.

Another benefit of using agile was that it helped me stay motivated and engaged throughout the development process. By breaking down the project into smaller tasks and focusing on delivering value to the end-user, I was able to see progress more easily and stay motivated to continue working on the project.

Since I was working alone on the project and didn't prefer to use Trello or Jira, I used a simple notebook to track my progress. This helped me stay on track and ensure that I was making steady progress towards completing the project.

Overall, using the agile methodology to develop my Android mobile application was a smart decision that helped me stay organized, focused, and efficient throughout the development process. By prioritizing the most valuable features, regularly reviewing and updating the project backlog, and using a notebook to track my progress, I was able to ensure that I delivered value to the end-user and made steady progress towards completing the project.

## **2. Project Objectives**

SuperCleaner is an Android mobile application where user can reserve cleaning services from a small enterprise in their local area.

Admin user can add/delete services that they can perform by adding services displayed on a standard users' dashboard. Also, the admin will be able to view all reservations made by all users on the platform(including themselves).

Each service on offer would have a flat rate, but price can change based on the size of the property being cleaned or the size of the vehicle being cleaned.

A standard user of the app will be able select a single or multiple service to add to their shopping cart. Then the standard user will be able to select a time, select an address that is previously saved, select a date, select a timeslot, and complete a payment.

After the completion of the payment will be able to view their reservations history. Reservation history would include ongoing reservations that are paid for but not completed, completed reservations, and cancelled reservations.

## **3. Technology**

### **3.1 Development**

Framework: 1st party android native mobile app.

Programming languages: Kotlin/Java/SQLite/JavaScript.

IDE: Android Studio

Emulator:        -    Device: Pixel 3a  
                     -    API Level: 30  
                     -    Target: Android 11

### **3.2 Back End Technology**

I am using Firebase for the projects backend needs. I am utilizing the following modules:

- Authentication
- Cloud Firestore
- Storage
- Stripe

Firebase supports authentication using email and password, as well as other methods. To use email/password authentication, you must create a Firebase project and enable the sign-in method. The Firebase Authentication API can then be used to create new user accounts, sign in existing users, and manage their authentication state.

Firebase handles security concerns such as password hashing and storage, and provides built-in UI libraries for a customizable sign-up and sign-in experience. Using Firebase Authentication simplifies the process of adding secure authentication to your application and reduces the potential for security vulnerabilities (developer.android.com, 2023).

Cloud Firestore is a flexible, scalable NoSQL cloud database from Google's Firebase platform. It allows you to store, retrieve, and query data in real-time. With Cloud Firestore, you can easily store and sync data across multiple devices, making it ideal for web, mobile, and server-side applications. It supports advanced querying capabilities, allowing you to perform complex operations on your data, such as retrieving documents based on multiple criteria. Additionally, it has built-in security rules that you can use to control access to your data. Cloud Firestore automatically scales as you add more data, making it a highly scalable and cost-effective solution for storing large amounts of data. Overall, Cloud Firestore provides an easy-to-use and scalable solution for managing your application's data, and if the app is successful.

Cloud Firestore offers improved querying, scalability, reliability, and security compared to Firebase Realtime Database. It supports efficient and flexible data structures and has better performance for large scale apps. Cloud Firestore is a more modern and versatile solution for data management (firebase.google.com, 2023a).

Firebase Storage provides a cloud-based storage service for images, audio files, and videos. Benefits include scalability, real-time synchronization, easy integration with Firebase services, robust security features, and global availability. It is a cost-effective and secure solution for managing and accessing data for web and mobile apps (firebase.google.com, 2019).

### **3.3 Payment Technology**

There are 2 different Payment methods that I wanted to try to integrate with SuperCleaner: Stripe or Paypal Sandbox.

#### **PayPal Sandbox**

PayPal Sandbox is a testing environment provided by PayPal for developers to test their PayPal integrations before going live. PayPal Sandbox offers a range of benefits for developers building Android mobile apps that integrate with PayPal, including improved testing and debugging capabilities, increased security, and reduced development costs.

One of the main benefits of using PayPal Sandbox for Android mobile app development is that it allows developers to test their PayPal integrations in a simulated environment. This enables them to test different scenarios, such as different payment amounts and currency types, and see how the app responds to these changes. Additionally, developers can use the sandbox environment to test different user accounts, ensuring that their app works seamlessly for all users.

Another benefit of using PayPal Sandbox for Android mobile app development is increased security. PayPal Sandbox allows developers to test their integrations without risking real transactions, which can help to reduce the risk of security breaches or fraudulent activity. By testing in a sandbox environment, developers can ensure that their app is fully secure before going live.

Finally, using PayPal Sandbox can also help to reduce development costs. By testing their integrations in a sandbox environment, developers can identify and resolve any issues before going live, reducing the need for costly post-launch fixes (PayPal, 2022).

## **Stripe**

Stripe SDK is a powerful tool that allows developers to easily add Stripe payments to their mobile apps and websites. With a range of integration options and programming language compatibility, Stripe SDK provides a secure and flexible payment processing solution that can be customized to meet the needs of any app or website.

The benefits of using Stripe SDK are numerous, including improved payment security, streamlined payment management, and enhanced user experience. Stripe's advanced security features, such as fraud protection and PCI compliance, ensure that all transactions are secure and protected against fraudulent activity. Stripe also offers a range of payment methods and automated payment management tools, making the payment process quick and convenient for users.

By using Stripe SDK, developers can create a custom payment flow that suits their app or website's design and user experience, resulting in higher conversion rates and improved customer satisfaction. Overall, Stripe SDK is a highly recommended tool for any developer looking to add secure and seamless payment processing to their mobile app or website (Murphy, 2023).

Overall I found Stripe SDK much easier to integrate into SuperCleaner than Paypal Sandbox. The documentation was more succinct and easier to follow along and implement. The different official tutorials with firebase greatly aided me in successfully implementing a functional payment pathway via Stripe (Wu and Schaeff, 2023).

I implemented the SDK by using generic code snippets that worked in tandem with Firebase.

Utilizing Firebase UI required me to upgrade my project from a free tier to the Pay as you go (Spark Plan). This was required to make the necessary payment pathway with Stripe SDK accessible.

With Stripe SDK and Firebase AuthUI the payment portal was made without generating additional layout files and activities and simplified the process of accepting a payment.

### **3.4 Learning Resources**

For this project I have used several learning resources to help me complete it, some of which I have referenced already. The most important resource would be the Mobile Application Development module in the HDip course.

However, for clarity, honesty, and adherence to the project guidelines I will reference the following resources that I found very useful:

- 1- Android Firebase Firestore - Masterclass - Build a Shop App(Tutorials.eu, 2022).
- 2- Android Jetpack Compose: The Comprehensive Bootcamp(Dichone, 2022).
- 3-Android App Development Masterclass using Kotlin(Buchalka, Jean-Paul Roberts and Buchalka's, 2015).

## **4. Implementation**

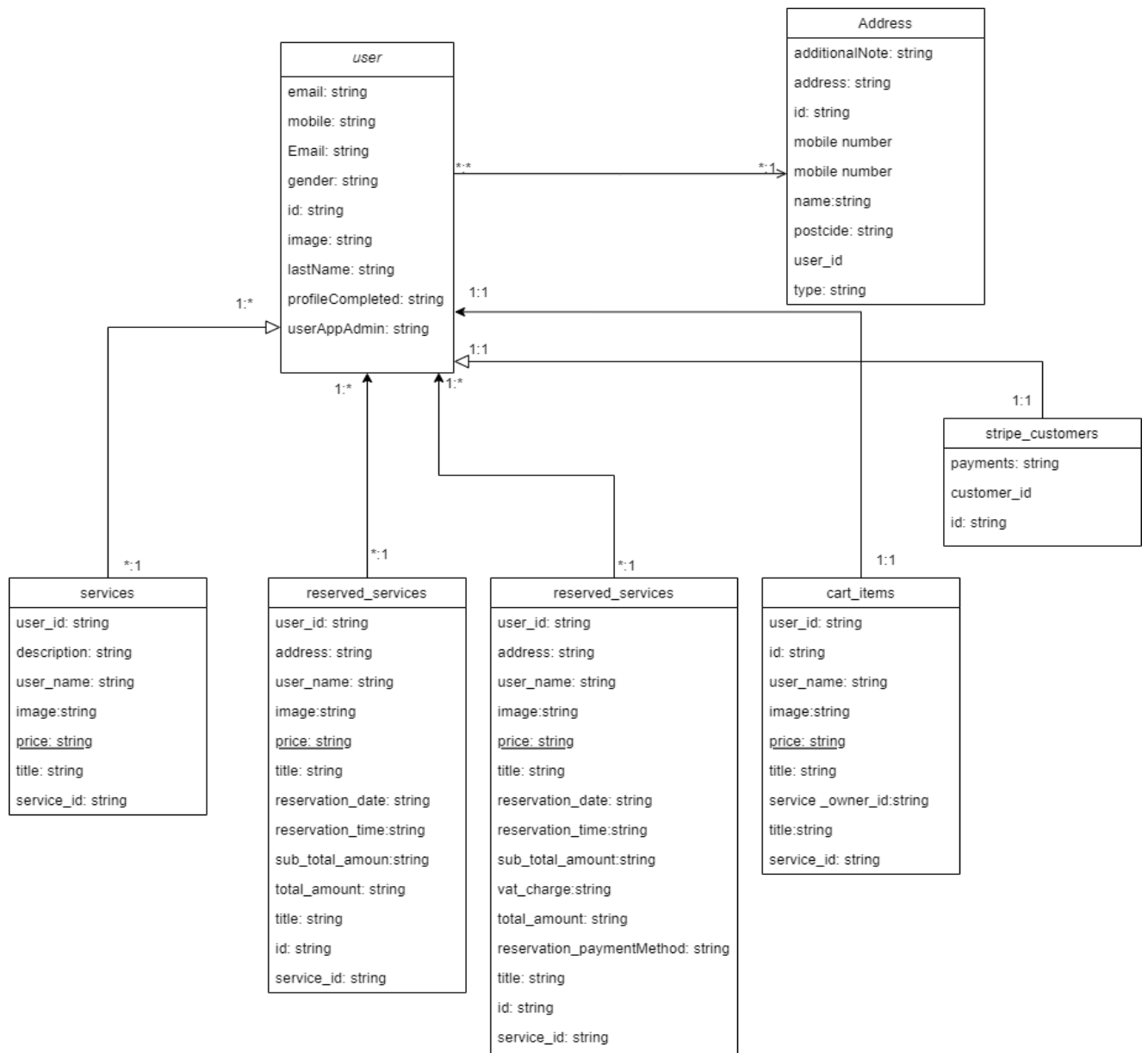
### **4.1 Introduction**

The following section will give a detailed overview of the functionality of App.

### **4.2 Data Modeling**

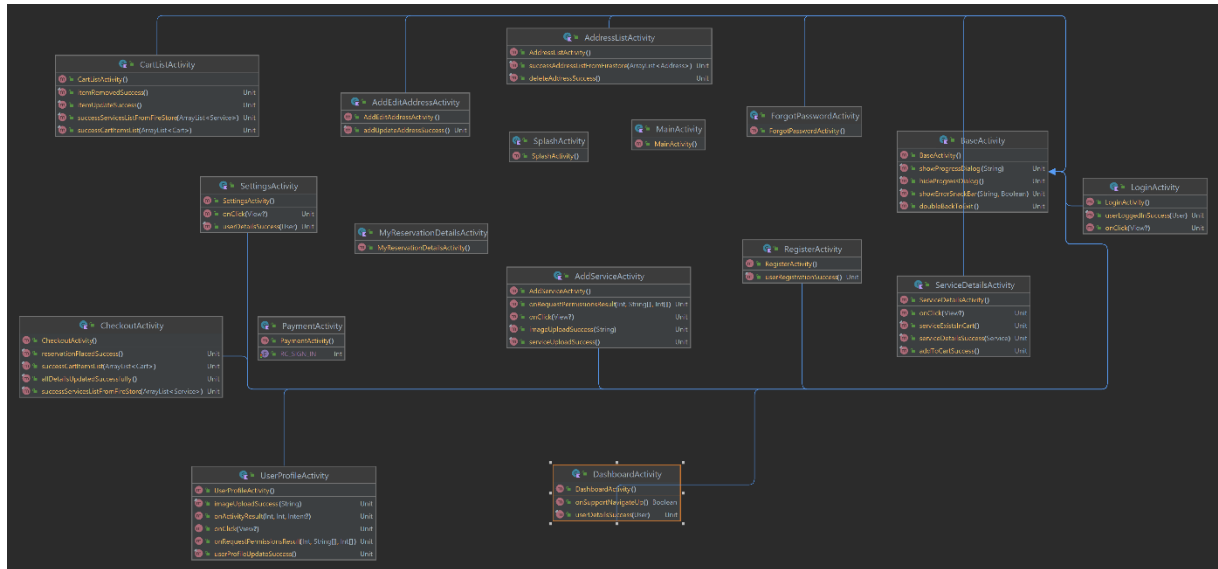
#### **4.2.1 ER Diagram**



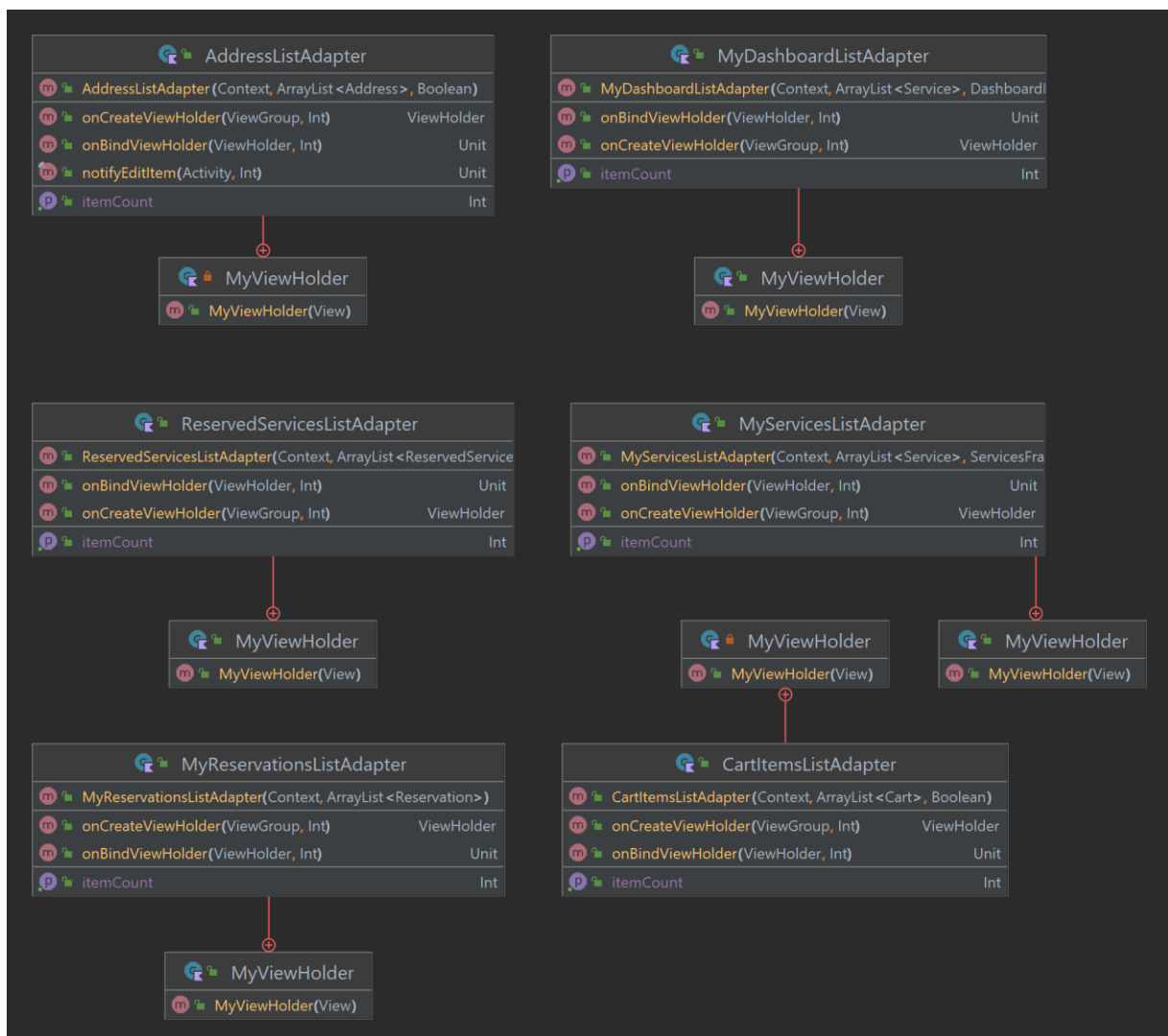


## 4.2.2 UML Diagram

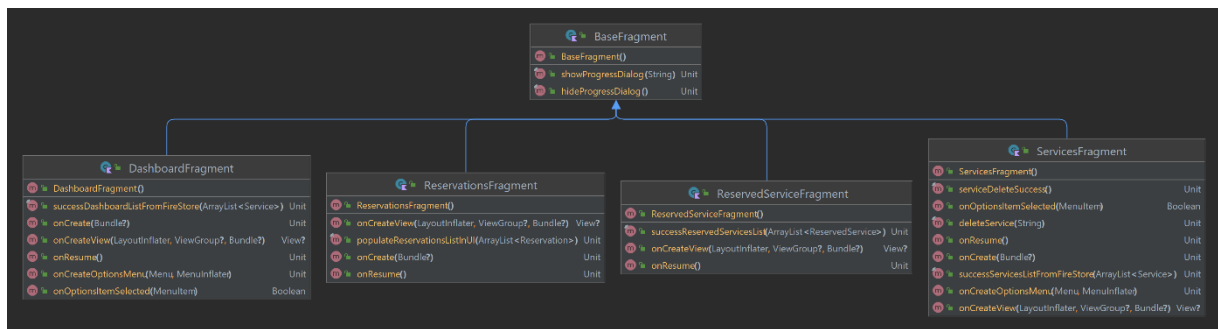
## Activites UML



## Adapters UML



## Fragments UML



## 4.3 App Icon & Logo



SuperCleaner Launcher Icon

The following image is the launcher icon of the in the android device menu. It was generated with the Configure Image Asset wizard with Android Studio.



SuperCleaner Logo

The following image is the logo that is visible at the login screen.

The launcher image and App logo were based on 2 free vector images downloaded from [icons8.com](https://icons8.com) (icons8.com, n.d.).

#### 4.4 Splash Screen



SuperCleaner Splash Screen

A splash screen is a graphical control element consisting of a window containing an image, a logo, and the current version of the software. A splash screen can appear while a game or program is launching. The splash page is an introduction page on a mobile application. A splash screen may cover the entire screen or may simply be a rectangle near the center of the screen or page. The splash screens of operating systems and some applications that expect to be run in full screen usually cover the entire screen (Lennartz, 2007).

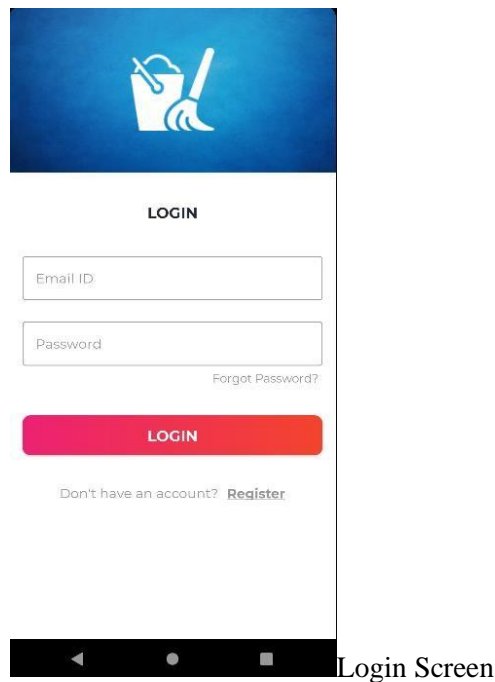
The splash screen contains a generic shaded blue colour and the name of the app. The app name is the name of the app in string form bound within a TextView(2) with a custom utility package file called SCATextView that applies a Montserrat-Regular font.

Throughout the App I will be using different utility files, but the most frequent usage for text will be of SCATextView and SCATextViewBold that apply Montserrat-Regular and Montserrat-Bold respectively. These 2 files draw on ttf files in the assets directory of the project (1001fonts.com, 2019).

The Splash screen lasts for only 2.5 seconds before directing the user to the Login view.

If the device user already registered, logged in a previous session of usage, and did not log out before closing the App, then they will be directed to the dashboard fragment when the launcher icon is tapped to activate the App. This is achieved by retrieving the User ID generated upon registration and saved to Cloud firestore in Firebase. I will expand further on this aspect in the registration section(4.6).

## 4.5 Login



Login Screen

After the Splash screen the user is directed to the Login Screen.

The Login screen displays the App Logo in a white contrast colour with the same background.

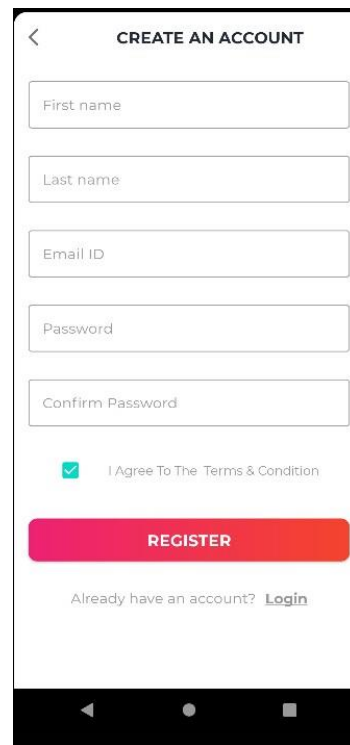
Also, there are 2 EditText(3) boxes to enter user email and password respectively. These boxes contain a hint to enter which piece of data to enter.

After entering the correct credentials ,and tapping the login button, the user will sign in and navigate to the next page.

If the user forgot their password, they can tap on Forgot Password? TextView and will then be directed to the Forgot password screen to change their password.

If the user doesn't have an account then they need to register, and they can navigate to the registration screen by tapping on the Register TextView.

## 4.6 Registration



The registration screen is titled "CREATE AN ACCOUNT". It features a back arrow in the top left corner. Below the title, there are five text input fields: "First name", "Last name", "Email ID", "Password", and "Confirm Password". Below these fields is a checkbox with a green checkmark and the text "I Agree To The Terms & Condition". A prominent red "REGISTER" button is positioned below the checkbox. At the bottom, there is a link that says "Already have an account? [Login](#)". The screen is framed by a black mobile status bar at the top and a black Android navigation bar at the bottom.

Registration Screen

After the user is directed to the registration screen they are greeted with 5 EditText boxes to enter their First name, Last name, email, password, and to confirm their password.

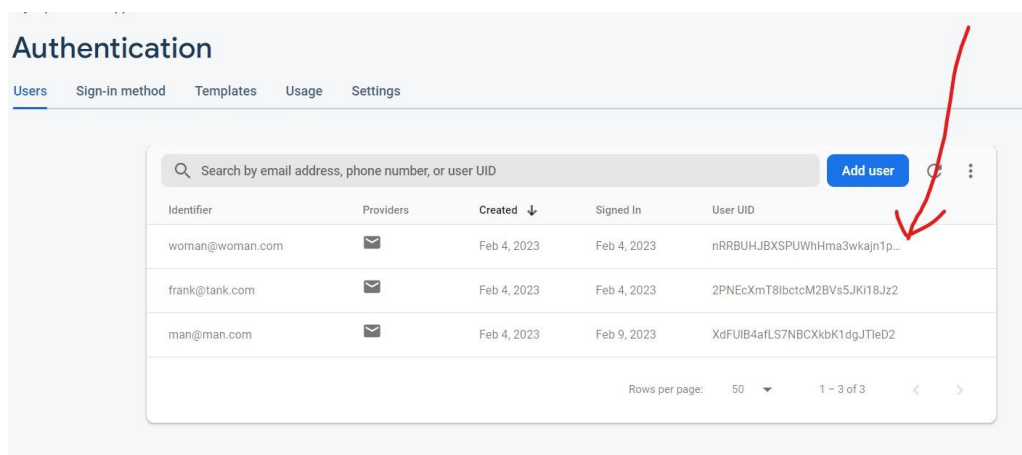
There is an image button with a black back arrow that if tapped would direct the user to the previous screen.

There is a check box to indicate that terms and conditions have been agreed too.

There is a Login EditText button direct the user back to the login screen.

There is an EditText button to register the user with the entered credentials.

After the Register button is tapped, the users' credentials are uploaded to Firebase authentication and in real time a User ID is generated.



The screenshot shows the "Authentication" section of the Firebase console, specifically the "Users" tab. At the top, there are navigation links: "Users", "Sign-in method", "Templates", "Usage", and "Settings". Below these is a search bar with the placeholder text "Search by email address, phone number, or user UID" and an "Add user" button. A table lists the registered users. A red arrow points to the "User UID" column header.

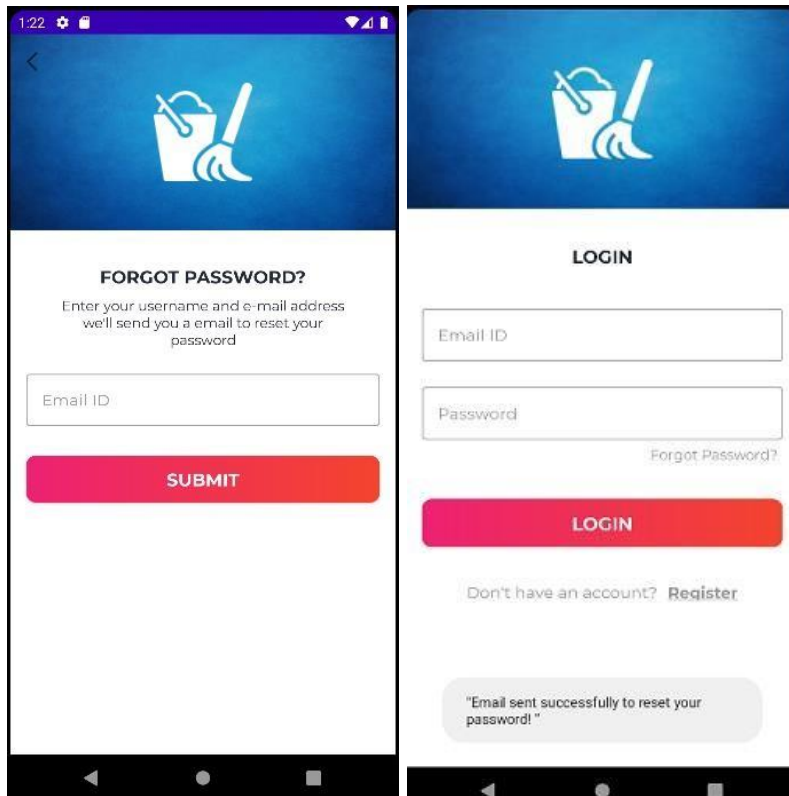
Identifier	Providers	Created ↓	Signed In	User UID
woman@woman.com	✉	Feb 4, 2023	Feb 4, 2023	nRRBUHJBXSPUWhHma3wkajn1p...
frank@tank.com	✉	Feb 4, 2023	Feb 4, 2023	2PNEcXmT8lbctcM2BVs5JKi18Jz2
man@man.com	✉	Feb 9, 2023	Feb 9, 2023	XgFUIB4afLS7NBCXkbK1dgJTleD2

At the bottom of the table, there is a "Rows per page" dropdown set to "50" and a pagination indicator "1 - 3 of 3".

User section in the Authentication module in Firebase

After successful registration the user is automatically directly to the Login Screen so that they can login to their new account.

#### 4.7 Forgot Password



Forgot password screen and Toast message in Login Screen.

After clicking on the forgot password EditText the user is directed to the ForgotPassword screen.

There is an image button with a black back arrow that if tapped would direct the user to the previous screen.

After the user enters their email a Toast(4) message will appear that a link is sent to that email to reset the password.



This is the email that a user receives in order to reset the password.

**Reset your password**

for [REDACTED]

New password

**SAVE**

**Password changed**

You can now sign in with your new password

After the link is clicked then we are directed via the browser to change the password.

All of these steps are handled by the authentication module in Firebase.

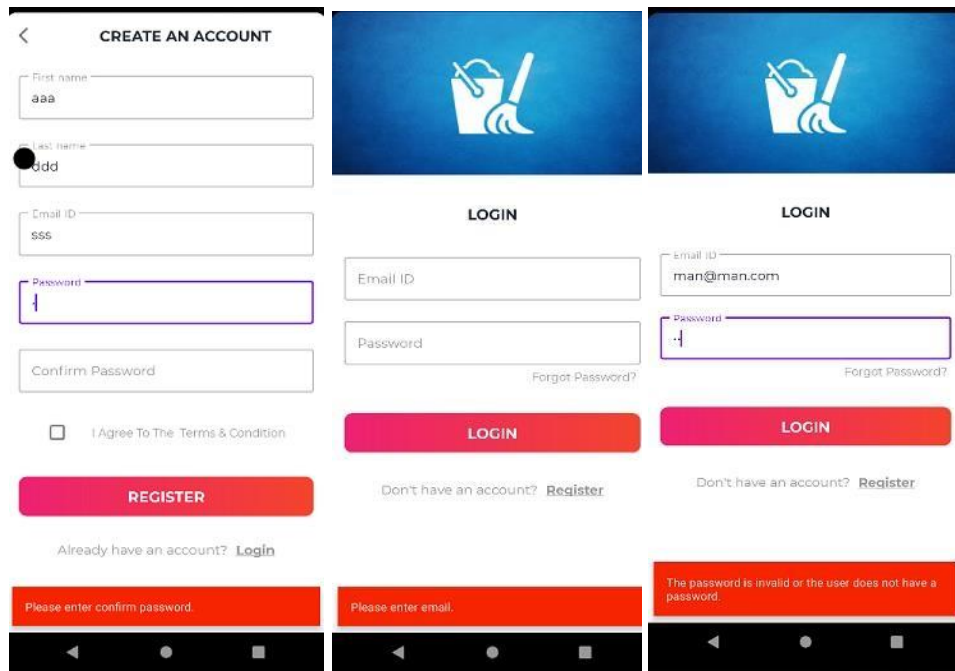
#### 4.8 Data Validation

If one of the fields is missing then there will be a Snackbar(5) message that appears to inform the user that they need to enter it because a profile cannot be created or logged in with empty fields.

The data validation in relation to empty fields is done locally on the device.

The data validation with regard to entering the wrong credentials to a registered account happens at the firebase authentication module level. The credentials are crossreferenced and crossvalidated by the authentication module which results in another snackbar message that the username or password are invalid and do not match what is on record.





Data Validation in Registration and Login Screens from missing credentials and wrong credentials

#### 4.9 Complete profile

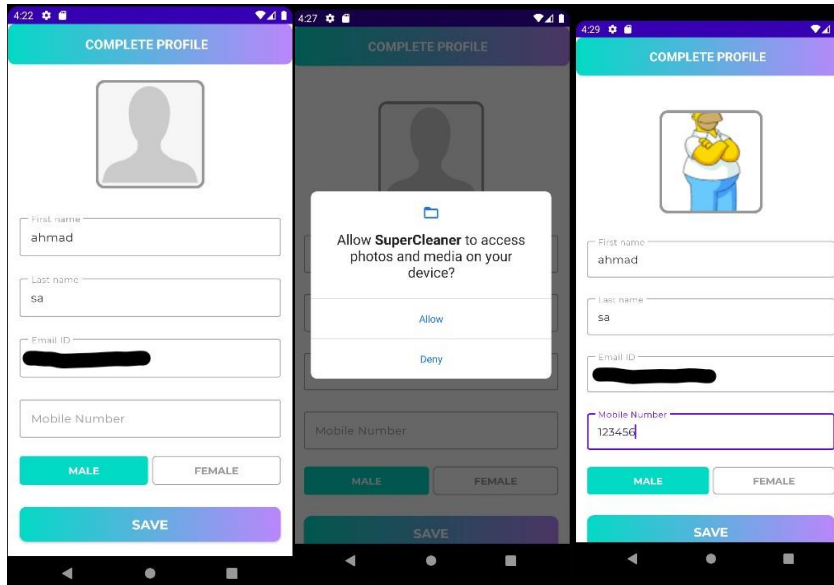
When a new user signs up they are directed to the complete profile screen where they can add their mobile number, specify their sex, and add a profile picture.

The app asks user permission to access photos and media on the device local storage. This is achieved by adding user permission in the project manifest file and utilizing Glide v4, and by adding the relevant Gradle dependencies (bumptech, n.d.).

Glide is a fast and efficient image loading library for Android focused on smooth scrolling. Glide offers an easy to use API, a performant and extensible resource decoding pipeline and automatic resource pooling (bumptech, 2015).

Glide's primary focus is on making scrolling any kind of a list of images as smooth and fast as possible, but Glide is also effective for almost any case where you need to fetch, resize, and display a remote image.

Once the profile image is uploaded, the user adds a mobile number, and then taps on save a Toast message appears stating that details have been saved successfully, and they are directed to the dashboard screen to view available services to reserve.



Complete Profile screen which allows the user to complete their profile

#### 4.10 Base Activity/Fragment

I utilized a hidden utility activity and fragment called BaseActivity & BaseFragment to implement a custom AlertDialog (developer.android.com, 2023) called ProgressDialog.

This Dialog will be utilized throughout the different lifecycles of activities and fragments that communicate with Firebase. It takes some time for data to be uploaded to and retrieved from the different Firebase modules. Instead of giving the appearance of the App freezing the progress dialog will appear asking the user to “Please Wait”.



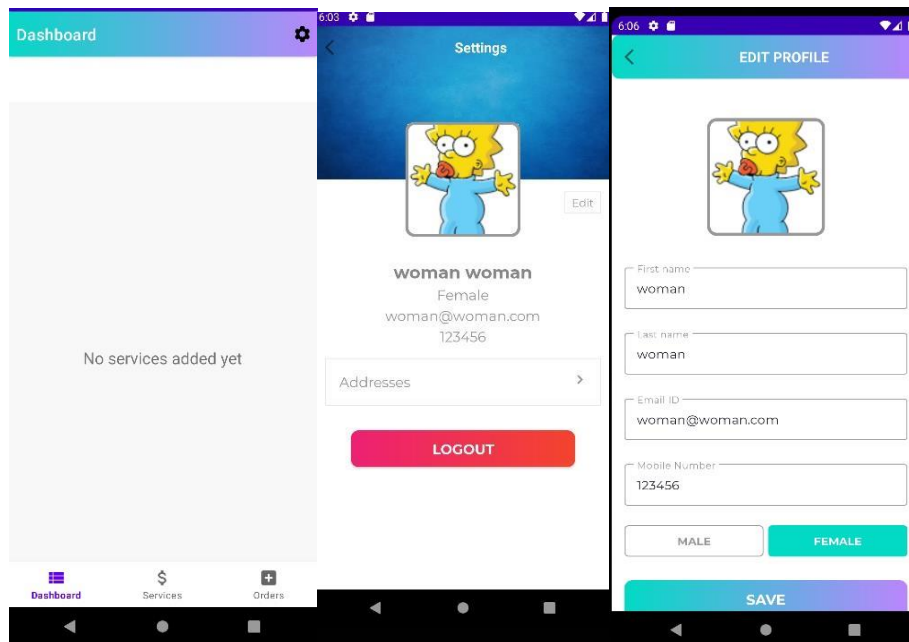
Progress Dialog

#### 4.11 Settings

After the user logs in and completes they are directed to the dashboard screen. Users can access the settings screen by tapping on the gear icon in the top right of the screen.

After navigating to the settings screen user is greeted by the profile image they uploaded, the personal details they entered in the registration, and complete profile screens. User can tap on the log out button to log out, tap on the Addresses button to add addresses, tap on the Edit Profile button to navigate to the edit profile screen.

After navigating to the edit profile screen the user can edit their first name, last name, mobile number, sex, and profile picture. After tapping save the user navigates back to the settings screen.



Navigation to Setting Screen and Edit Profile Screen

#### 4.11.1 Add/Edit/Delete Address

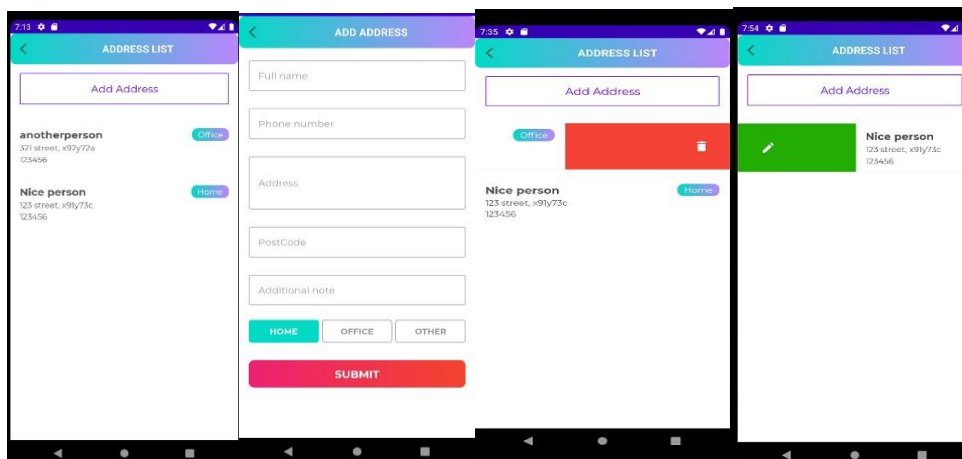
After the user taps on the add address button in the Settings screen they are directed to the address list screen.

The user will be able to see previous addresses entered by the user. Address details are shown in each of the tiles.

If a user would like to add an address, then they would need to tap the add address button, and add the relevant details of the address they would like the service to take place in.

Existing addresses can be edited by either tapping on the tiles or swiping right on one of them. If the user wants to delete an existing address then they can swipe left.

The swipe animation for deleting or editing an address is achieved with a 3<sup>rd</sup> party library called ItemTouchHelper.SimpleCallback (Kitowicz, 2017).



Address portion of the App

## 4.12 Navigation

The current build of the app currently has bottom navigation(6) and navigation drawer(7) because they allow the user to switch to different activities/fragments easily, it makes the user aware of the different screens available in the app, and the user is able to check which screen are they on at the moment ([www.geeksforgeeks.org](http://www.geeksforgeeks.org), 2020).

Bottom navigation bar could be implemented manually but it was easier to add it via the wizard in Android Studio. Based on the tutorial the implementation of the navigation drawer was similar, but with some manual configuration, because effectively I am using the same activities/fragments in both scenarios.

The visibility of menu items in the bottom navigation and navigation drawer is updated based on the `userAppAdmin` value in the User Model collection saved in Firestore. If the user is an admin (`userAppAdmin = true`), the "Services" and "Reserved Services" items are visible; otherwise, they are hidden. The default value is set to false upon the creation of the user profile and needs to be edited manually by the developer in order to attain the desired difference in the UI.

The `setupWithNavController` function is called for `navBottomView` to set up the bottom navigation view with the given `NavController` (`navController`). This enables the bottom navigation to handle user interactions and update the content displayed in the app.

Bottom navigation is implemented using the `BottomNavigationView` component. The `navBottomView` variable is an instance of `BottomNavigationView` and is initialized using the `findViewById` method with the ID `R.id.bottom_navigation_view`.

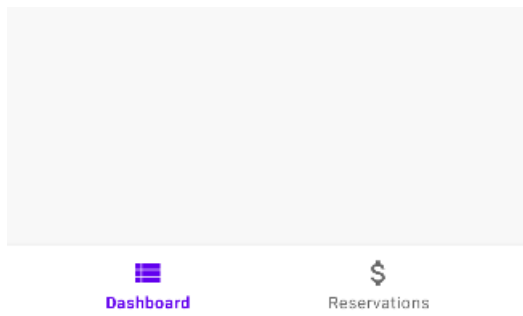
The navigation drawer is implemented using the `NavigationView` and `DrawerLayout` components. The `navView` variable is an instance of `NavigationView`, and `drawerLayout` is an instance of `DrawerLayout`. Both variables are initialized using the `findViewById` method with their respective IDs `R.id.nav_view` and `R.id.drawer_layout`.

Similar to the bottom navigation, the visibility of menu items in the navigation drawer is updated based on the `userAppAdmin` value. The same logic is applied for displaying or hiding the "Services" and "Reserved Services" items.

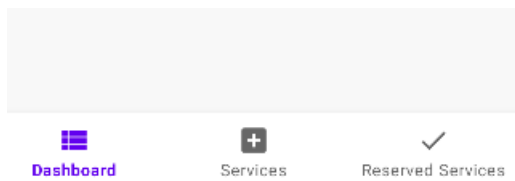
The `AppBarConfiguration` is created using the `AppBarConfiguration` class, and it is used to configure the app bar (the top section of the screen) with the given `NavController`. This configuration includes the set of top-level destinations and the `DrawerLayout`.

The `setupActionBarWithNavController` method sets up the action bar with the given `NavController` and `AppBarConfiguration`. The `navView` is then set up with the `NavController` by calling the `setupWithNavController` method.

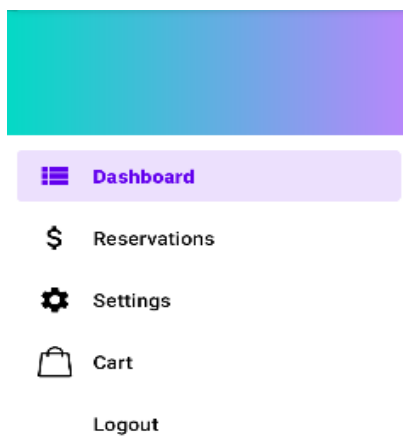
To handle user interactions with the navigation drawer items, a listener is set by calling the `setNavigationItemSelectedListener` method on `navView`. This listener contains a `when` statement to handle the navigation and other actions, such as opening the settings or cart activities, and logging out.



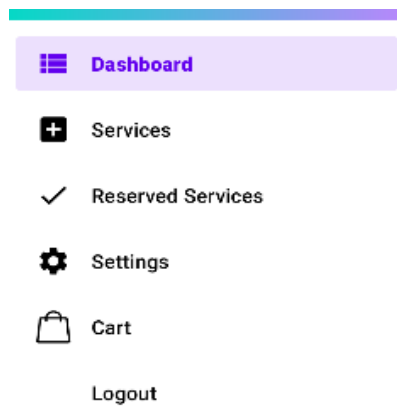
Bottom Navigation Bar of a Standard User



Bottom Navigation Bar of Admin User



Navigation Drawer of Standard User



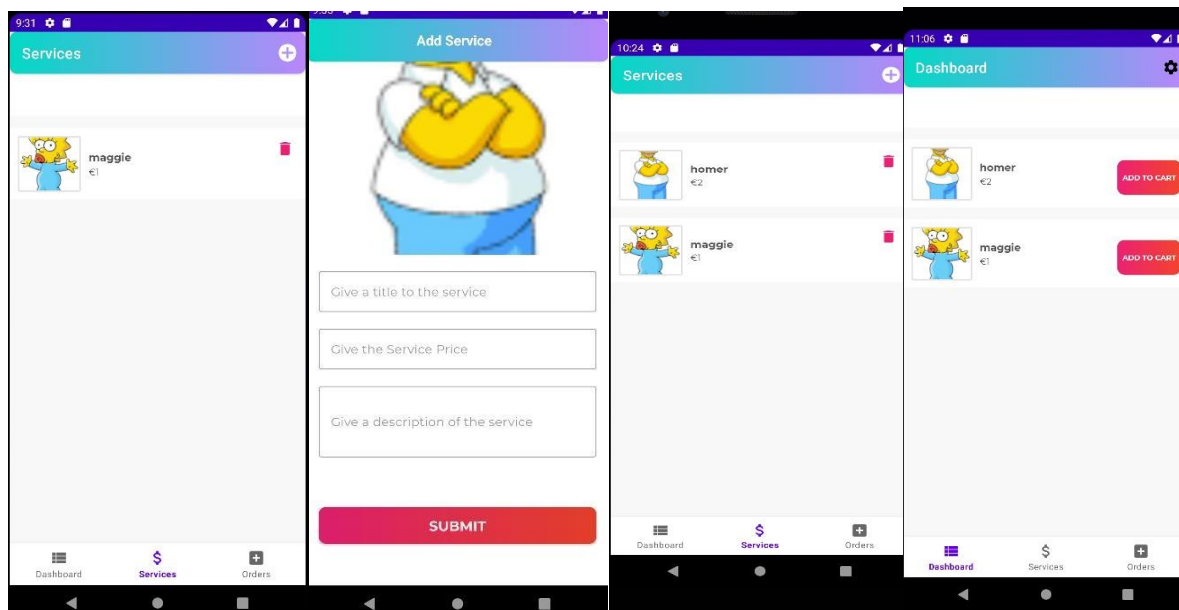
### Navigation Drawer of Admin User

#### 4.13 Add a service/Service Description

When the admin user wants to list a new service to be on offer for customers then they would navigate to the services fragment with the bottom navigation bar. The user should then tap on the  $\oplus$  in the top right of the screen. The user will then be directed to the Add Service screen where they can add an image to represent the service, give it a name, price, and description. Once the submit button is tapped then the service is saved and displayed in the services fragment and dashboard fragment.

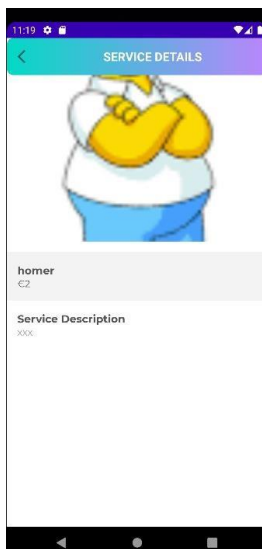
The difference between both cases is that the services fragment is only visible from the admin profile where they can delete and add services, while the dashboard fragment is visible by the admin user and the standard user.

Services on display in the dashboard are available for all users in order to make a reservation and can be added to the cart to complete an order.



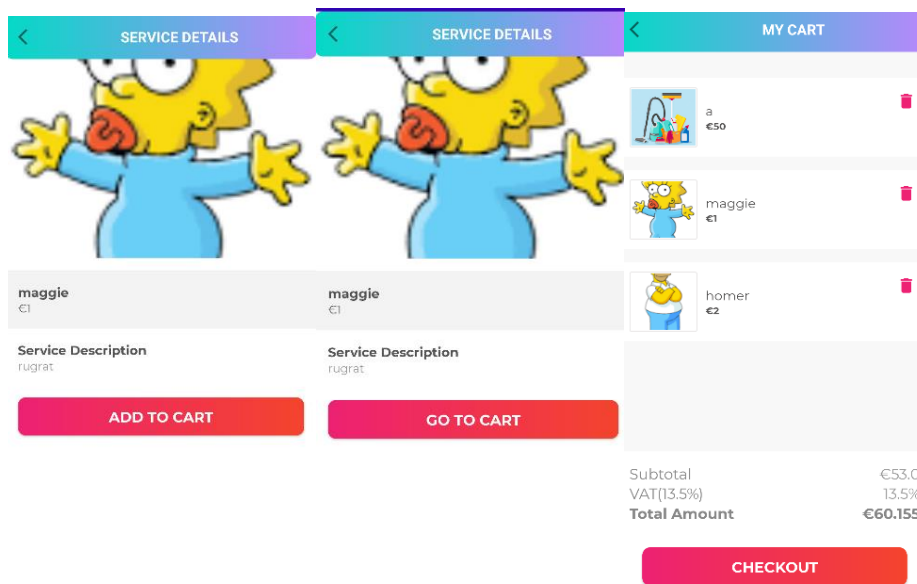
Adding/deleting service in services fragment and display in dashboard fragment.

Admin users can tap on the services on displayed services in the services or dashboard fragments to see the details associated with the service such as the service name, price, description, and associated image.



Service Details screen

#### 4.1 Add to Cart/View Cart



After selecting the service details button in the dashboard fragment the user is then directed to the service details activity where an image of the selected service, its name, price and description button.

If the user decides to add the service to the cart they can tap on the add to cart button, which after tapping disappears and the go to cart. The user has the option to navigate to the cart and complete their purchase or return to the dashboard fragment to add other services to the cart.

If the user does eventually to view the current cart, all the previously selected services are on display with the cumulative value with the VAT rate applied giving a total amount of the current card.

The user can delete an item in the cart if they do not want it anymore by tapping on the red coloured bin icon. The deletion of an item from the cart will also be reflected in the figures below where the total amount will change as well.

If the user is satisfied with the services that they can tap on the check out button and will be directed to the select address activity to choose an address which is visible in the first image of section 4.11.1 of this report. Once the user selects an address they are directed to the check out activity.

#### 4.15 Check out

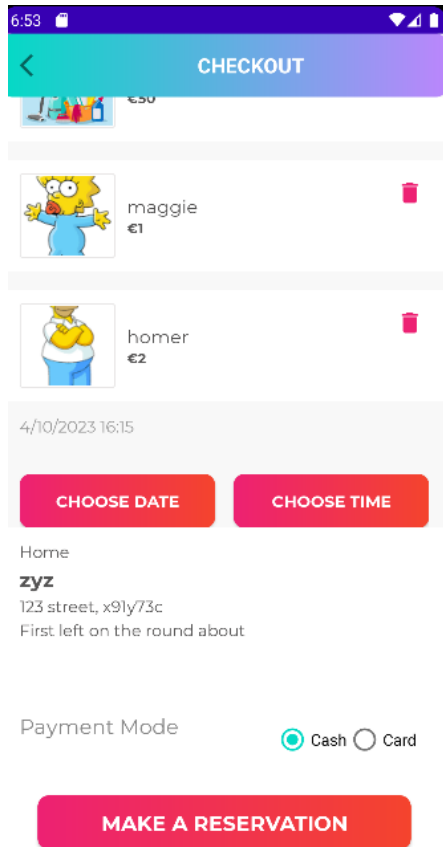
After the user arrives at the check out activity the same information from the previous check out activity is displayed, but the user should be able to view the selected services, selected address, and they are able to select the date and time they want that service to take place.

The information displayed is contained in a scrollview so regardless of the size of the cart in the user can scroll down to view the contents of the check out screen.

Also, the user must select a payment method( cash or card) from the 2 radio buttons in the payment mode section. Once the user selects the make a reservation button while the cash radio button is selected a Toast message appears that states a reservation is made and the user is directed to the Dashboard fragment.

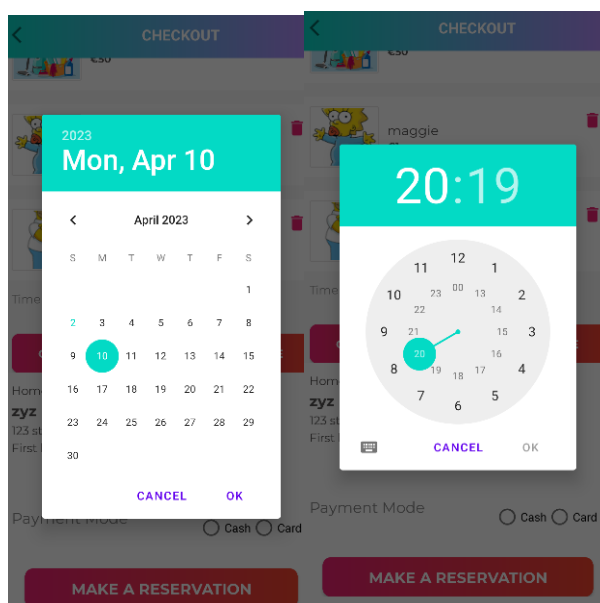


After the user selects the make a reservation button, when the card radio button is selected, a Toast message appears that states a reservation is made, and they will be directed to the payment activity to make a payment.



Check out Activity

#### 4.16 Selecting Date and time



Time and date selection

Selection of date and time for each reservation to be displayed in the check out activity using date pickers and time pickers (Francis, 2022).

The date and time pickers work as follows:

The binding.btnPickDate button's click listener is set up to show a DatePickerDialog. When the user clicks this button, a calendar dialog appears, allowing them to select a date for their reservation.

The datePickerDialog.datePicker.setOnDateChangeListener is set up to validate the selected date. If the selected date is a Sunday, the "OK" button is disabled, preventing users from choosing Sundays for their reservation.

The datePickerDialog.setButton is set up to handle the "OK" button click event. When the user clicks the "OK" button, the selected date is retrieved, formatted, and displayed in the binding.tvSelectedDateTime TextView.

The binding.btnPickTime button's click listener is set up to show a TimePickerDialog. When the user clicks this button, a time picker dialog appears, allowing them to select a time for their reservation.

The custom TimePickerDialog subclass is created to override the onTimeChanged method. This method is called whenever the user changes the selected time, and it checks if the selected time is within the allowed range (8:00 - 16:59). If it is, the "OK" button is enabled; otherwise, it's disabled.

The TimePickerDialog's setOnTimeSetListener is set up to handle the "OK" button click event. When the user clicks the "OK" button, the selected time is retrieved, formatted, and appended to the binding.tvSelectedDateTime TextView, displaying the full date and time for the reservation.

#### **4.17 Payment**

When the user taps on the make a reservation button when the radio button selection is "Card" PaymentActivity and FirebaseEphemeralKeyProvider activities are triggered to complete a payment.

PaymentActivity handles the payment process's UI and flow, while FirebaseEphemeralKeyProvider manages the creation of ephemeral keys using Firebase Cloud Functions(8). Together, they enable a smooth and secure payment experience using Stripe in the Android app.

PaymentActivity is initialized with necessary components like Stripe instance, PaymentConfiguration, and binding views.

When the user clicks the login button, they sign in using Firebase Authentication.

After a successful login, the showUI() function is called, which sets up the payment session by calling setupPaymentSession(). This function initializes the CustomerSession with the FirebaseEphemeralKeyProvider instance.

The FirebaseEphemeralKeyProvider class is responsible for creating an ephemeral key by calling a Firebase Cloud Function named createEphemeralKey. This function communicates with the Stripe server to create the key and returns it to the app.

The PaymentSession in PaymentActivity is initialized with necessary configurations like shipping and billing information requirements. It also has a PaymentSessionListener that listens for changes in the payment session data, such as the selection of a payment method.

When the user clicks the "Select Payment Method" button, the payment session presents the payment method selection UI, allowing the user to choose their desired payment method.

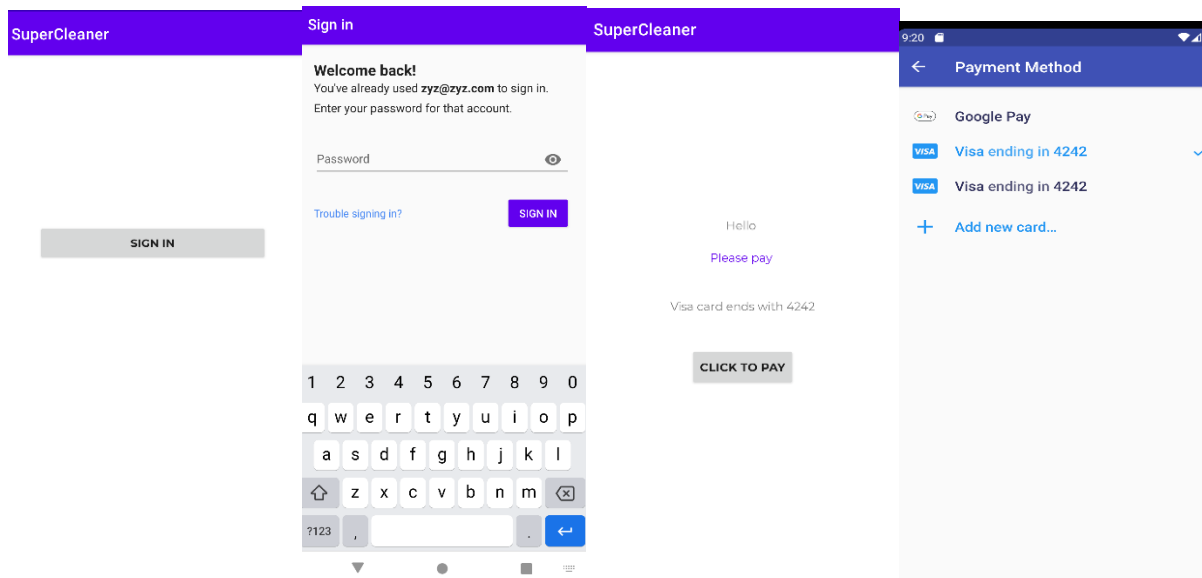
Once the user selects a payment method, the PaymentSessionListener detects the change and updates the UI with the selected payment method information. It also enables the "Pay" button.

When the user clicks the "Pay" button, the `confirmPayment()` function is called with the selected payment method ID.

The `confirmPayment()` function creates a new document in the Firestore database with the payment details, such as amount and currency.

The app listens for updates on the payment document, and once the payment intent's client secret is available, it confirms the payment using the Stripe instance's `confirmPayment()` method.

If the payment is successful, a Toast "Thank you for your payment" message is displayed, and the user is navigated to the `DashboardActivity`.



The images above utilize stripe SDK and firebase AuthUI to make a payment with the logged in user.

In the 2<sup>nd</sup> image the user is recognized as a previously authenticated user. The user Id is one of the lines used in the meta data associated with the transaction.

In the 3<sup>rd</sup> image a previously used card is cached and populated if the user didn't uninstall the app from the device.

In the 4<sup>th</sup> image user can enter a credit card or use a cached one.






Warning to the reader of this report: When attempting to complete a transaction please use a test card not your own credit/card. Even though the stripe sdk implementation is in test mode please protect your personal information.

There is data validation via the Stripe SDK with regard to incorrect number of digits in the card long number and if a past date is inputted in the expiry date.

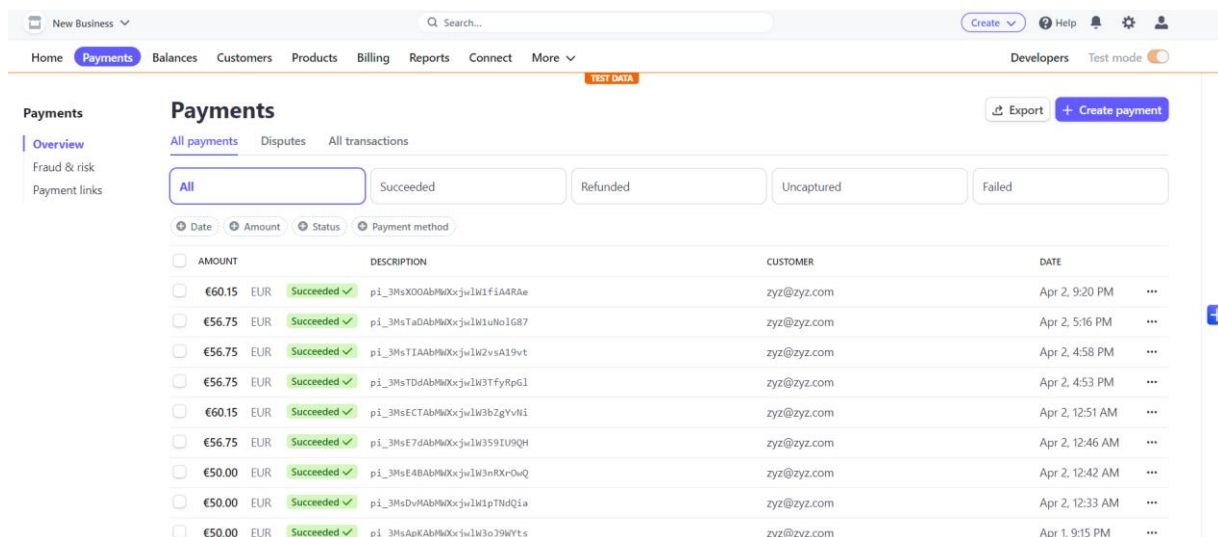
Please review the image below for using other test cards.

## Cards by brand

To simulate a successful payment, use test cards from the following list. The billing country for each test card is set to the United States. If you need to create test card payments using cards for other billing countries, use [international test cards](#).

Card numbers	PaymentMethods	Tokens	
BRAND	NUMBER	CVC	DATE
<div><div><div></div></div><div>Filter...</div></div>			
Visa	4242 4242 4242 4242 	Any 3 digits	Any future date
Visa (debit)	4000 0566 5566 5556 	Any 3 digits	Any future date
Mastercard	5555 5555 5555 4444 	Any 3 digits	Any future date
Mastercard (2-series)	2223 0031 2200 3222 	Any 3 digits	Any future date
Mastercard (debit)	5200 8282 8282 8210 	Any 3 digits	Any future date

After a payment is complete transaction details are recorded in the merchant stripe account along with the meta data of the user and transaction as seen in the image below of the payments dashboard.



The screenshot shows the Stripe Payments dashboard. The top navigation bar includes links for Home, Payments (active), Balances, Customers, Products, Billing, Reports, Connect, and More. The main content area is titled 'Payments' and shows a list of transactions. The table has columns for AMOUNT, DESCRIPTION, CUSTOMER, and DATE. All transactions shown are successful and have a status of 'Succeeded'.

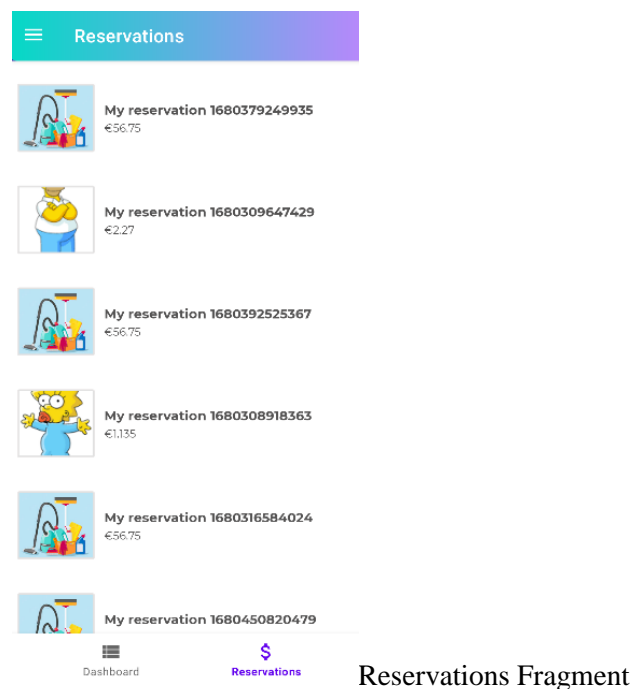
AMOUNT	DESCRIPTION	CUSTOMER	DATE
€60.15 EUR	p1_3MsX00AbM0xjw1W1f1A4Rae	zyz@zyz.com	Apr 2, 9:20 PM
€56.75 EUR	p1_3MsTa0AbM0xjw1W1u1o1G87	zyz@zyz.com	Apr 2, 5:16 PM
€56.75 EUR	p1_3MsTIAAbM0xjw1W2vsA19vt	zyz@zyz.com	Apr 2, 4:58 PM
€56.75 EUR	p1_3MsTD0AbM0xjw1W3TFyRp6l	zyz@zyz.com	Apr 2, 4:53 PM
€60.15 EUR	p1_3MsECTAbM0xjw1W3bZgYvNi	zyz@zyz.com	Apr 2, 12:51 AM
€56.75 EUR	p1_3MsE7dAbM0xjw1W359IU9QH	zyz@zyz.com	Apr 2, 12:46 AM
€50.00 EUR	p1_3MsE4BAbM0xjw1W3nRXrDuQ	zyz@zyz.com	Apr 2, 12:42 AM
€50.00 EUR	p1_3MsDvHAbM0xjw1W1pTnDQia	zyz@zyz.com	Apr 2, 12:33 AM
€50.00 EUR	p1_3MsApKAbM0xjw1W3o39WYts	zyz@zyz.com	Apr 1, 9:15 PM

## 4.18 Reservations Fragment

The MyReservationsListAdapter is a custom RecyclerView adapter that handles the display of reservations in a list format. It takes a context and a list of Reservation objects as its input. The adapter inflates the layout for each reservation item and binds the data from the Reservation object to the views within the layout. This includes loading the service image, setting the service title, and displaying the total amount for the reservation.

The ReservationsFragment is where the actual list of reservations is displayed. It retrieves the reservations data by calling `getMyReservationsList()` which, in turn, calls the `FirestoreClass().getMyReservationsList()` function to fetch the list of reservations from the Firestore database. Once the list is fetched, the `populateReservationsListInUI()` function is called with the fetched reservations list as an argument.

Inside `populateReservationsListInUI()`, the reservations list is checked for its size. If there are items in the list, the RecyclerView's visibility is set to `View.VISIBLE`, and the 'No Reservations Found' message is hidden by setting its visibility to `View.GONE`. A new `MyReservationsListAdapter` instance is created with the current activity context and the list of reservations, and this adapter is set as the adapter for the RecyclerView. This populates the RecyclerView with the reservations data, displaying each reservation item according to the layout and data binding defined in the `MyReservationsListAdapter`. If there are no reservations in the list, the RecyclerView is hidden, and the 'No Reservations Found' message is displayed.



#### 4.19 Reservation Details

The `MyReservationDetailsActivity` is an activity responsible for displaying the details of a specific reservation.

The activity receives the `Reservation` object passed through the intent using `Constants.EXTRA_MY_RESERVATIONS_DETAILS`.

`setupActionBar()` function is called to set up the action bar with a back button.

`setupUI(reservationDetails: Reservation)` function is called with the `Reservation` object to populate the UI with the reservation details.

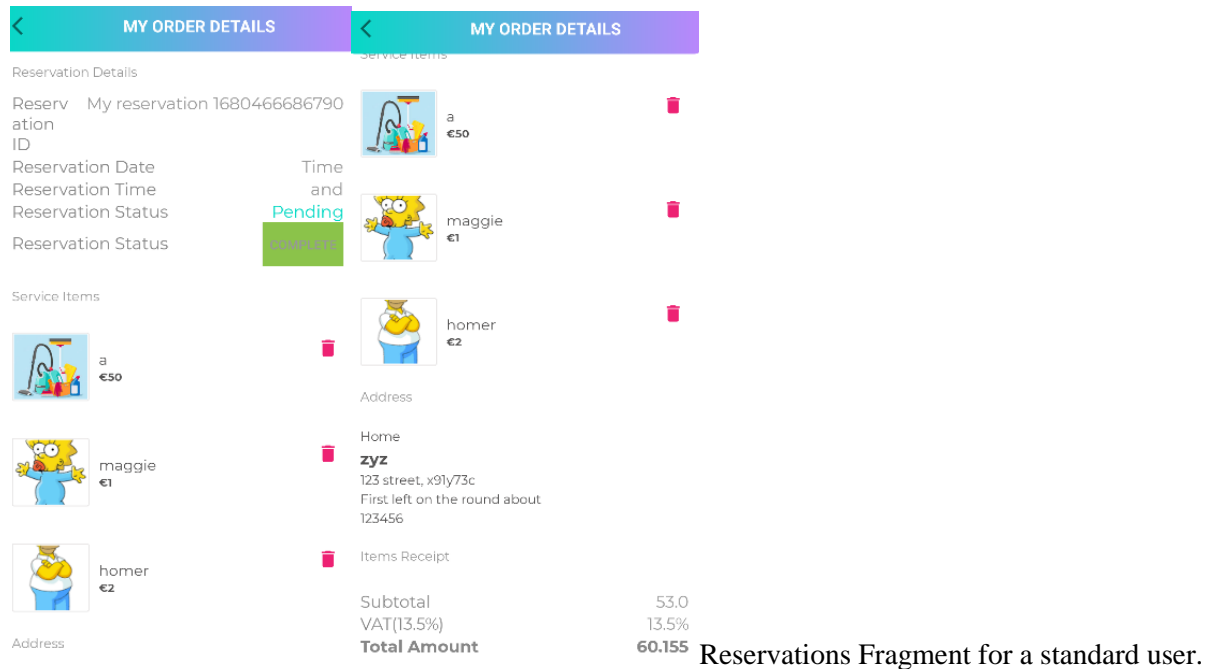
The reservation details like reservation ID, date, and time are set in their respective `TextViews`.

The RecyclerView `rvMyReservationItemsList` is set up with a `LinearLayoutManager` and a `CartItemListAdapter` which takes the list of items in the reservation to display the reserved services.

Address details like address type, full name, address, additional note, other details, and mobile number are set in their respective TextViews.

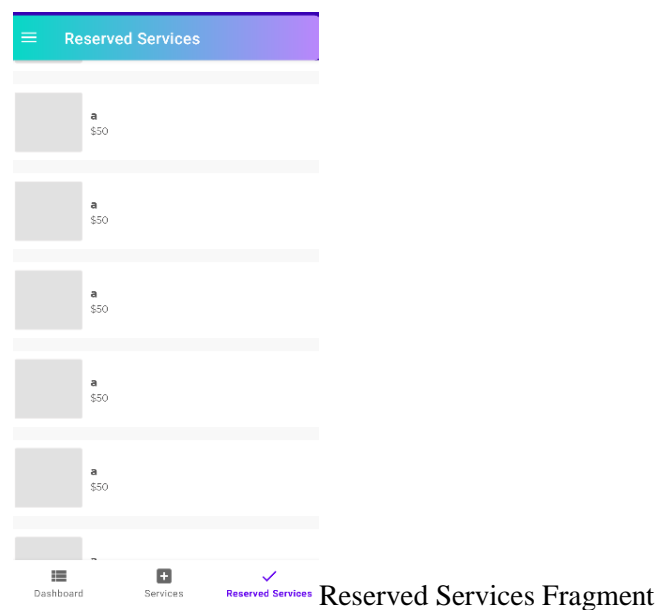
Subtotal, VAT charge, and total amount of the reservation are set in their respective TextViews.

So, when the activity is launched, it displays the complete details of the reservation, including reserved services, address, and cost information.



The code to display reserved services in the above images is that of a standard user.

However, the display of reserved services at the level of an admin user is more limited. Since the logic would be similar, but due to time constraints my inability to solve current bugs, I can only display reservations made but not their details in the following in the Admin user Reserved Services fragment



With more time and effort beyond this assignment I am confident I can fix the bugs and display the reserved services details in full.

## **5 Critical Self Review/Analysis**

### **5.1 What I learned**

During the course of this project, I learned a lot about Android development and Kotlin programming. I now have a better understanding of how to code in Kotlin, which will help me in future Android development projects.

Additionally, I gained valuable knowledge in developing an Android ecommerce app, which I can apply to future projects in this area.

Lastly, I improved my time management skills as I was also working on two other modules each semester while working on this project, which has helped me better manage my workload and stay organized. Overall, I believe this project has been a valuable learning experience that has equipped me with new skills and knowledge for future projects.

### **5.2 What direction the project might be taken if more time was available**

If more time were available for this project, I would have focused on improving the UX design to make the app more user-friendly and visually appealing. This could involve conducting user testing and incorporating feedback to refine the design.

I also would have completed the reserved service activity in the admin view to make the app more functional and efficient for the admin user. Additionally, I would have added the functionality for reviews, allowing users to rate services with a thumbs up or thumbs down.

I would have added the functionality for refunds. This would allow users to request a refund for a service they have purchased through the app, and the admin would be able to approve or deny the request. The refund functionality would improve the user experience by providing additional flexibility and ensuring that users are satisfied with the services they purchase. It would also benefit the admin by streamlining the refund process and helping to resolve any disputes that may arise.

These improvements would have enhanced the app's overall user experience and made it more competitive in the ecommerce app market as a minimum viable product.

### **5.3 Problems**

During the development of this project, I encountered several problems related to the integration of PayPal for payments. I found the PayPal documentation difficult to navigate, which made it challenging to implement the PayPal sandbox process.

As a result, I ended up spending more time than anticipated on a solution that I ultimately did not use. This caused delays in the development process and impacted the overall quality of the app.

In the end, I decided to switch to the Stripe SDK for payments, which proved to be a more efficient and reliable solution.

While the PayPal integration presented challenges, it was ultimately a learning experience that taught me the importance of careful research and planning when implementing third-party integrations.

## 6. Bibliography

1. FlashWash (2022). *Flash Wash – Flash Wash*. [online] <https://flashwash.ie/price-andpackages/>. Available at: <https://flashwash.ie> [Accessed 6 Nov. 2022].
2. Helping (2022). *Find domestic and trusted cleaners near you*. [online] Helping.ie. Available at: <https://www.helping.ie/>.
3. IT Governance Ltd (n.d.). The PCI DSS | IT Governance Europe Ireland. [online] <https://bit.ly/3RSfdTp>. Available at: <https://bit.ly/3RSfdTp> [Accessed 11 Feb. 2023].
4. Vilmate (2020). How to Build an eCommerce Mobile App - An Ultimate Guide | Vilmate. [online] Nearshore Software Development Company in Ukraine - VILMATE. Available at: <https://vilmate.com/blog/how-to-build-an-ecommerce-mobile-app/> [Accessed 11 Feb. 2023].
5. icons8.com (n.d.). Cleaning Icons – Download for Free in PNG and SVG. [online] [icons8.com](https://icons8.com). Available at: <https://icons8.com/icons/set/cleaning> [Accessed 11 Feb. 2023].
6. Lennartz, S. (2007). Splash Pages: Do We Really Need Them? [online] Smashing Magazine. Available at: <https://www.smashingmagazine.com/2007/10/splash-pages-do-we-really-need-them/> [Accessed 11 Feb. 2023].
7. tutorialspoint.com (n.d.). Android - TextView Control - Tutorialspoint. [online] [www.tutorialspoint.com](http://www.tutorialspoint.com). Available at: [https://www.tutorialspoint.com/android/android\\_textview\\_control.htm](https://www.tutorialspoint.com/android/android_textview_control.htm).
8. 1001fonts.com (2019). 1001 Fonts · Free Fonts Baby! [online] 1001fonts.com. Available at: <https://www.1001fonts.com/>.
9. abhiandroid.com (n.d.). EditText Tutorial With Example In Android Studio: Input Field | Abhi Android. [online] [abhiandroid.com](http://abhiandroid.com). Available at: <https://abhiandroid.com/ui/edittext> [Accessed 12 Feb. 2023].
10. designsystem.quickbooks.com (n.d.). Toast messages. [online] QuickBooks Design System. Available at: <https://designsystem.quickbooks.com/component/toastmessage/#:~:text=Toast%20messages%20let%20users%20know> [Accessed 12 Feb. 2023].
11. m1.material.io (n.d.). Snackbars & toasts - Components. [online] Material Design. Available at: <https://m1.material.io/components/snackbarstoasts.html#:~:text=Snackbars%20provide%20brief%20feedback%20about> [Accessed 12 Feb. 2023].
12. bumptech (n.d.). Glide v4 : Download & Setup. [online] [bumptech.github.io](http://bumptech.github.io). Available at: <https://bumptech.github.io/glide/doc/download-setup.html> [Accessed 12 Feb. 2023].
13. bumptech (2015). Glide v4 : Fast and efficient image loading for Android. [online] [bumptech.github.io](http://bumptech.github.io). Available at: <https://bumptech.github.io/glide/>.



14. developer.android.com (2023). Dialogs. [online] Android Developers. Available at: <https://developer.android.com/develop/ui/views/components/dialogs>.
15. Kitowicz, M. (2017). RecyclerView swipe to delete easier than you thought. [online] Medium. Available at: <https://medium.com/@kitek/recyclerview-swipe-to-delete-easier-than-you-thoughtc67ff5e5f6>.
16. www.geeksforgeeks.org (2020). Bottom Navigation Bar in Android. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/bottom-navigation-bar-in-android/>.
17. firebase.google.com (2023). Firebase Authentication. [online] Firebase. Available at: <https://firebase.google.com/docs/auth#:~:text=Firebase%20Authentication%20provides%20backend%20services>.
18. firebase.google.com (2023a). Choose a Database: Cloud Firestore or Realtime Database | Firebase Realtime Database. [online] Firebase. Available at: <https://firebase.google.com/docs/database/rtdb-vsfirestore#:~:text=Cloud%20Firestore%20also%20features%20richer> [Accessed 13 Feb. 2023].
19. firebase.google.com (2019). Cloud Storage | Firebase. [online] Firebase. Available at: <https://firebase.google.com/docs/storage>.
20. Paypal (n.d.). PayPal sandbox testing guide. [online] developer.paypal.com. Available at: <https://developer.paypal.com/tools/sandbox/>.
21. Murphy, R. (2023). Stripe Payments Review: Pros, Cons, Alternatives. [online] NerdWallet. Available at: <https://www.nerdwallet.com/article/small-business/stripe-review> [Accessed 3 Feb. 2023].
22. Wu, S. and Schaeff, T. (2023). Firebase mobile payments: Android & iOS with Cloud Functions for Firebase. [online] GitHub. Available at: <https://github.com/stripe-archive/firebase-mobile-payments> [Accessed 2 Jan. 2023].
23. Francis, S. (2022). How to create material date and time pickers in android. [online] Medium. Available at: <https://medium.com/@segunfrancis/how-to-create-material-date-and-time-pickers-in-android-18ecd246838b> [Accessed 2 Apr. 2023].
24. Firebase (2023). Cloud Functions for Firebase. [online] Firebase. Available at: <https://firebase.google.com/docs/functions> [Accessed 2 Apr. 2023].
25. Tutorials.eu (2022). Android Firebase Firestore - Masterclass - Build a Shop App. [online] Udemy. Available at: <https://www.udemy.com/course/android-firebase-firestore-masterclass-build-a-shop-app/> [Accessed 3 Apr. 2023].
26. Dichone, P. (2022). Android Jetpack Compose: The Comprehensive Bootcamp. [online] Udemy. Available at: <https://www.udemy.com/course/kotlin-android-jetpack-compose/> [Accessed 3 Apr. 2023].
27. Buchalka, T., Jean-Paul Roberts and Buchalka's, T. (2015). Android App Development Masterclass using Kotlin. [online] Udemy. Available at: <https://www.udemy.com/course/android-oreo-kotlin-app-masterclass/> [Accessed 3 Apr. 2023].

## 7. Glossary & Abbreviations

1.     **PCI DSS:** The PCI DSS (Payment Card Industry Data Security Standard) is an information security standard designed to reduce payment card fraud by increasing security controls around cardholder data. The Standard results from a collaboration between the major payment brands (American Express, Discover, JCB, Mastercard and Visa). It is administered by the PCI SSC (IT Governance Ltd, n.d.).
  
2.     **TextView:** A TextView displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing (tutorialspoint.com, n.d.).
  
3.     **EditText:** EditText is a standard entry widget in android apps. It is an overlay over TextView that configures itself to be editable. EditText is a subclass of TextView with text editing operations. We often use EditText in our applications in order to provide an input or text field, especially in forms. The most simple example of EditText is Login or Sign-in form (abhiandroid.com, n.d.).
  
4.     **Toast:** Toast messages let users know that the task they just performed was successful. These messages give users immediate feedback after taking some action. Users don't need to dismiss toast messages, as they appear only for a moment before they disappear (designsystem.quickbooks.com, n.d.).
  
5.     **Snackbar:** Snackbars provide brief feedback about an operation through a message at the bottom of the screen. Snackbars contain a single line of text directly related to the operation performed. They may contain a text action, but no icons (m1.material.io, n.d.).
  
6.     **Bottom Navigation:** A user interface pattern in Android applications that displays the main navigation options at the bottom of the screen, making it easier for users to switch between top-level views. It is typically used in apps with three to five top-level destinations. The Bottom Navigation is implemented using the BottomNavigationView component.
  
7.     **Navigation Drawer:** A user interface pattern in Android applications that provides a slide-out menu, usually from the left side of the screen. It enables users to navigate between different sections of the app easily. The Navigation Drawer is implemented using the NavigationView and DrawerLayout components.
  
8.     **Firestore cloud functions:** Firestore Cloud Functions are serverless, event-driven functions that automatically scale, providing developers with a platform to run backend code in response to Firestore events or HTTPS requests without managing servers. They integrate seamlessly with other Firestore services, enable code modularity, and simplify app development. Functions can be triggered by various events, such as database changes, user authentication, or file uploads, allowing for efficient, real-time processing and a streamlined user experience (Firestore, 2023).

## 8. Declaration of authenticity

I declare that the work which follows is my own, and that any quotations from any sources (e.g. books, journals, the internet) are clearly identified as such by the use of 'single quotation marks', for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Student: Ahmad Sabeh-Murphy

Date: 03/04/23

Work Place Mentor: n/a

Date .....