

1. Introduction

1.1 Institute Profile

Institute Name: Sinhgad Institute of Management

Address: Vadgaon Budruk, Pune

Year of Establishment: 1994

Affiliation: Savitribai Phule Pune University

Approval: All India Council for Technical Education (AICTE)

Official Website Link: www.sinhgad.edu/

1.2 Abstract

Hair stylers is basically an Android application that is a major help to salons. Because they have many customers and many customers visit salons on different days, they cannot store their data manually or paperless. So, this application is helpful there. This application provides services of parlour to customers. Therefore, it creates interaction between parlours and customers. Also, by using this application we provide the best features to our users. In day to day our life is going to change mean it travel in digital way. Therefor we need some digital platform to easier our life. This application is one of the parts of this digitalization. Now a days every businessman stores their customers details and provides the best services. That one of the businesses is hair salon. I developed this app in android. This app provide service for hair salon business to store their customers details and also this is reminded customer to cut hair within his time period. This app helps to improve relations between customer and businessman.

Hair stylers is a mobile-based salon app with appointment scheduling functionality. It connects clients, salons, and stylists in an online community allowing users to browse salons and stylists, and book or cancel appointments. Users can also write and read reviews of

salons and stylists. Salons can specify the stylists that work at their salons, as well as the services they offer. Salons can also book appointments for customers and can view and print schedules in convenient formats.

My project will use Android and Firebase to back the interface with strong database functionality. For appointment scheduling, Beauty Hub app will integrate mobile calendar as a backend database for appointments as well as a front-end scheduling interface. This project will target the major play store app as the initial platform for the Beta version.

1.3 Existing System and Need for System

▪ Existing System

- In existing system some hair salon businessman cannot store their customers details therefore they cannot able to communicate between them this happens they lose some customers.
- Also, they cannot able to provide their best services to their customers.
- The existing system cannot provide the booking of time slots.
- When we need to book any time slot, we required to call that specific hair salon and then ask them to book my slot.
- Customers lose their valuable time
- Customer can not able to find their offers and discounts

▪ Need for System

Its customers do not have a proper way to make an appointment other than making a call or visiting the Salon premises. Salon owners, employees, and customers need to keep reminders on their mobiles over appointments. The salon owner and her employees maintain a diary to note down the appointment details. Service details of the salon are written on the paper; which always leads to misplacements. The owner needs to write all the service details on a new paper once it gets misplaced or updated.

Once payment has been made, the customer will receive a handwritten receipt (from the manual receipt book), and the cashier is keeping a copy of the same receipt. There is a higher risk of misplacing the receipt copies. The owner has no proper way to manage her employees and clients.

Generally, Salon hikes a lot in charges of their services. Even their offer is displayed for a limited number of days. Even there is a lot of rush in the salons, which consumes more time.

1.4 Scope of System

- Customers can book their time slots for hair cutting.
- Hair salon businessman can directly provide their best services to customers. For example, their best offers, Holidays, etc.
- Customers can reduce their time for calling hair salon and booking of slots.
- In future hair salon can able to sell their best product on this app.
- Providing the facility to registering Salon staff and maintaining their details.
- Providing the facility to registering regular Customers and maintaining them details.
- Facilitate appointment handling.
- View appointments leaves and holidays through an event calendar.
- Handling Salon Services along with their respective prices, hours etc.
- Providing Customer Payment handling option.
- Generating invoices through the system.
- Generating reports to support the higher managerial decisions.

1.5 Operating Environment - Hardware and Software

- **Server-side requirement**
 - Hardware Requirements:
 - Processor: Intel core i3 and above
 - RAM: 4 GB
 - HDD: 120GB
 - Software Requirements:
 - Operating System: Windows 10
 - Database: Firebase
 - Front End: Android
 - Server-Side Script: Java
 - Software Development Tool: Android Studio Code
- **Client-side requirement**
 - Hardware Requirements:
 - Android Version: above 7
 - RAM: 2 GB

1.6 Brief Description of Technology Used

- **Java**
 - For management of data.
 - To pass backend data to the frontend side.
 - For validation
- **Android**
 - For user interface (Frontend)
- **Android Studio**
 - To write a better code

- To manage the structure of files in project
- **XML**
 - For frontend design
 - To store values in projects like string, color, values, etc.
- **GitHub**
 - For storing or backup project files

1.6.1 Operating systems used (Windows or Unix)

- **Windows**
- **Android**

1.6.2 RDBMS/No SQL used to build database

- **Firebase**
- **SQLite**
- **JSON**

2. Proposed System

2.1 Study of Similar Systems (If required research paper can be included)

Title: Beauty and Aesthetics – A study of the Professional Hair Care Industry

Abstract

This paper describes about the Hair Stylers application developed using Android Studio new version. It also includes about the Single Platforms on which development of android platform application can be done. Hair Stylers Application is an Android Application which is builds in Android Studio 8.0.1. Android Studio is an official integrated development tool or environment for Google's Android operating system. It is builds on JetBrains' IntelliJ IDEA software. The main motive of building Cab Application is to provide employment and also make drivers, owners and customer's life easy. So basically, we are trying to connect peoples (customers and Owners of the Salon) can be mutually benefited. In this Application number of services available, so the customer can easily select the services, date of service, time of service etc.

Background: Beauty is a subject which is not easy to grasp especially as it is perceived differently. In advertising it is expressed through aesthetic messages and images which we relate to symbolic and social meanings. The professional hair care industry in Sweden serves as a good example where the creation of aesthetic experience influences consumer purchasing behaviour.

Purpose: The purpose of our thesis is to study how consumers' subjective view on beauty and aesthetics can be influenced by the professional hair care industry and how market is created for products which mainly satisfy emotional needs rather than fulfil utilitarian function.

Research Method: In our study we have applied an abductive research method approach. The empirical findings were based on 3 interviews with P&G Salon Professional representatives and 15 end consumers combined with a survey, conducted in 25 hair salons in the city of Pune.

Conclusion: Consumers act in a socially constructed world in which products are shaped around impulse and feeling rather than their rationality. When buying a professional hair care product people receive much more than the actual product itself. People improve not only physical appearance but they also feel beautiful from within. While the utilitarian function is basically the same in both professional hair care and retail products, the former contributes to higher degree of satisfaction.

Keywords: Beauty, aesthetics, hair care, purchasing behaviour, marketing

2.2 Feasibility Study

- **Technical Feasibility:**

- User Friendly
- We provide service to customers to book their times slots
- Anyone can create an account on this app and get the best services

- **Economic Feasibility:**

Salons can add their time periods means they can add their salon opening time and closing time, according to this time system automatically creates slots. We do not charge any fee from customers for slot booking.

- **Operational Feasibility:**

Salons can be able to add their worker's details on this app. Before logging into the app, it is mandatory to create an account first. By the creation of an account, the app asks users if it is a men's parlour or a women's parlour. We provide separate services for men's and women's salons.

2.3 Objectives of Proposed System

- Customers will not need to search for makeup artists for wedding ceremonies and can directly go to the app to find a good parlour and find a good makeup artist. And can easily book makeup artists.
- To enhance interaction between customers and salon.
- To get better service for customer from salon

2.4 Users of System

1. Hair salon:

- Add their customers details in their database.
- Reminds customers for their hair cut.
- Provide best offers to their customers.
- Can recommend a new hair style.

2. Worker:

- Can able to manage their holidays.
- Can keep the details of which customer he has cut hair

3. Customer:

- Can book their time slot for haircut.
- Can able to compare price of haircut.
- Can able to pay their hair cut payment.
- Can give feedback to the system
- Can give review of the salon

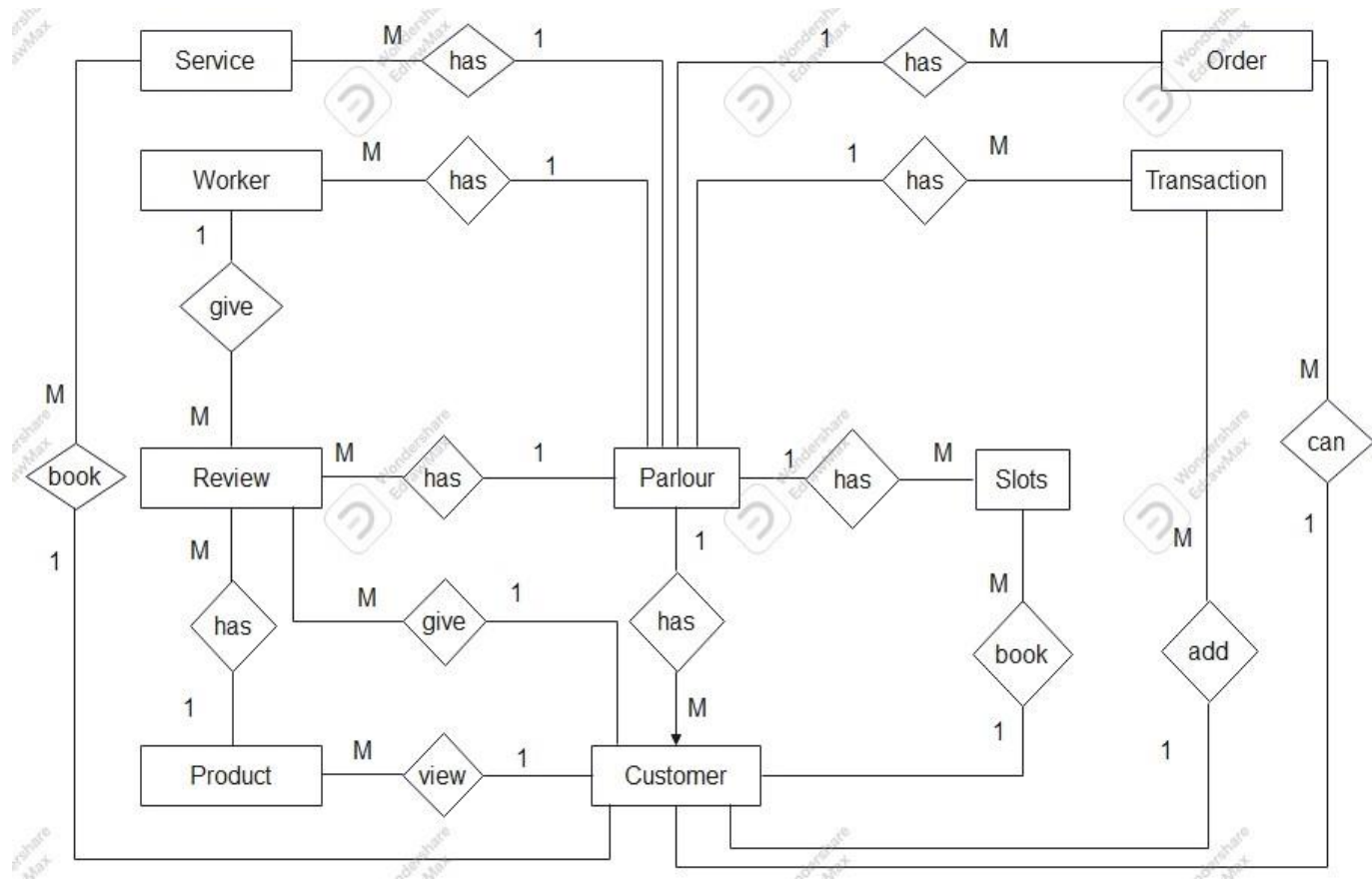
3. Analysis and Design

3.1 System Requirements (Functional and Non-Functional requirements)

- **Functional requirement**
 - Authentication
 - Business core
 - Transactions, checkouts
 - Authorizations
 - History data
 - Checking real time market price
 - Getting advertisement

- **Non-Functional requirements**
 - Loading speed
 - Time taken to deliver server response
 - User response time
 - Data consumption limits
 - Displaying parlours in the area
 - Displaying customers of parlours
 - Processing transactions
 - Refresh transactions and customers listings every 5 minutes
 - Refresh QR code every second

3.2 Entity Relationship Diagram (ERD)



3.3 Table Structure

Table Name: Parlour

Column Name	Data Type	Constraint
key	String	NOT NULL
Name	String	NOT NULL
Subtitle	String	NOT NULL
Mobile	String	NOT NULL
Email	String	Primary Key
Password	String	NOT NULL
Type	String	NOT NULL
Opening_Time	String	NOT NULL
Closing Time	String	NOT NULL
slot_time_period	number	NOT NULL
Address	String	NOT NULL

Pincode	number	NOT NULL
Photo	String	NOT NULL
Token	String	NOT NULL

Table Name: Worker

Column Name	Data Type	Constraint
Key	String	NOT NULL
Name	String	NOT NULL
Email	String	Primary Key
Mobile	String	NOT NULL
Password	String	NOT NULL
Photo	String	NOT NULL
Parlour_Email	String	NOT NULL
Token	String	NOT NULL

Table Name: Customer

Column Name	Data Type	Constraint
key	String	NOT NULL
Name	String	NOT NULL
Mobile	String	NOT NULL
Email	String	Primary Key
Password	String	NOT NULL
Gender	String	NOT NULL
pincode	number	NOT NULL
Token	String	NOT NULL

Table Name: Service

Column Name	Data Type	Constraint
key	String	Primary Key
Name	String	NOT NULL
Price	number	NOT NULL
Parlour_Email	String	Foreign Key

Table Name: Slot

Column	Data Type	Constraint
key	String	Primary Key
Slot_No	number	NOT NULL
Time	String	NOT NULL
Cutomer_Email	String	Foreign Key
Parlour_Email	String	Foreign Key
Service	String	NOT NULL

Table Name: Transacation

Column Name	Data Type	Constraint
key	String	Primary Key
Customer_Email	String	Foreign Key
mobile	String	NOT NULL
Parlour_Email	String	Foreign Key
Worker_Email	String	Foreign Key
Service	String	NOT NULL
Price	number	NOT NULL
Date	String	NOT NULL
Time	String	NOT NULL
Customer_token	String	NOT NULL

Table Name: Order

Column Name	Data Type	Constraint
key	String	Primary Key
Order_id	String	NOT NULL
Parlour_Email	String	Foreign Key
Worker_Email	String	Foreign Key
Customer_Email	String	Foreign Key
Date	String	NOT NULL
makeup_Type	String	NOT NULL
Price	number	NOT NULL

Table Name: Product

Column Name	Data Type	Constraint
key	String	Primary Key
Name	String	NOT NULL
Description	String	NOT NULL
Price	number	NOT NULL
Parlour_Email	String	Foreign Key
Rating	number	NOT NULL

Table Name: Feedback

Column Name	Data Type	Constraint
key	String	Primary Key
Name	String	NOT NULL
email	String	NOT NULL
Comment	String	NOT NULL

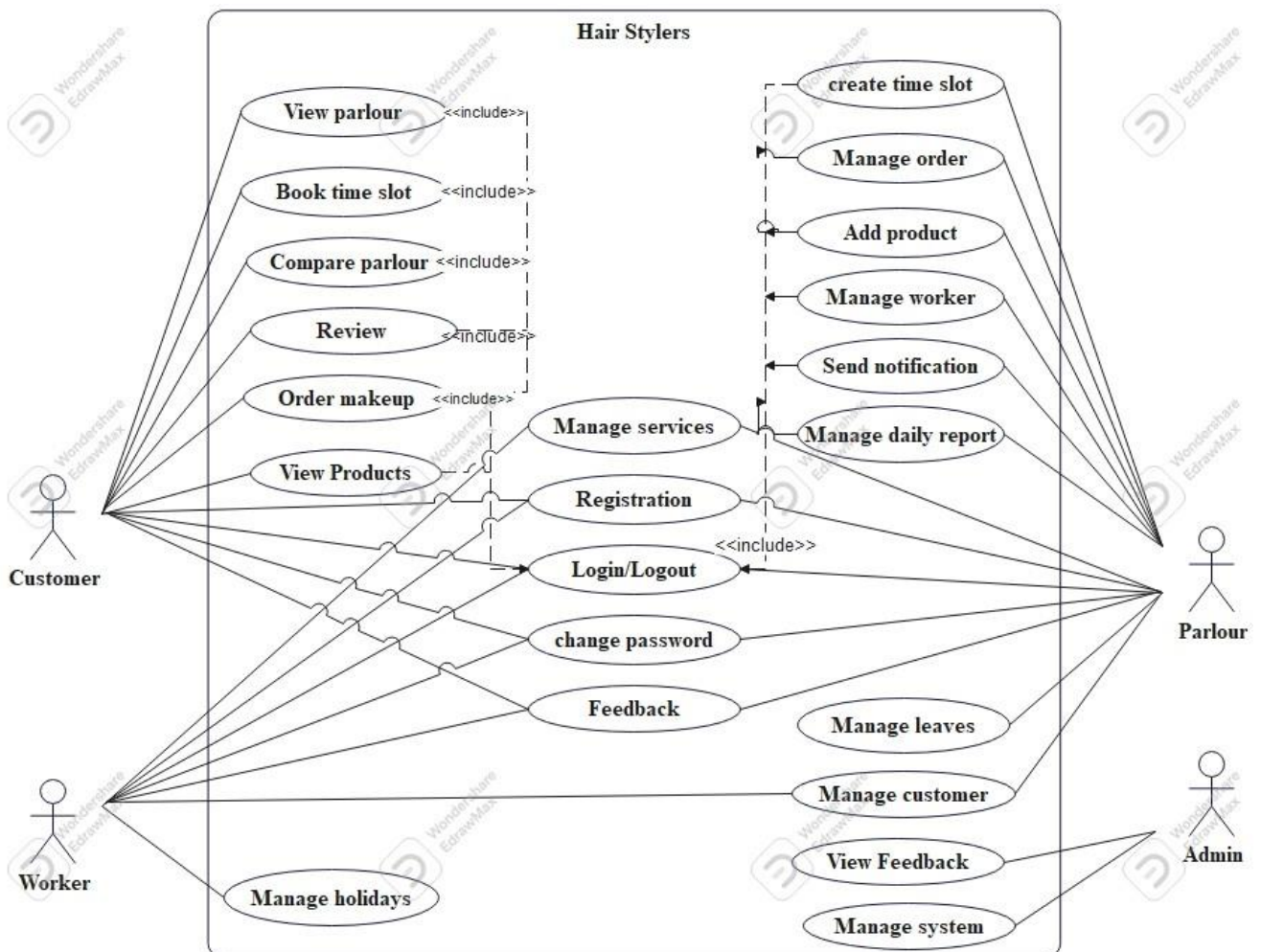
Table Name: Review

Column Name	Data Type	Constraint
key	String	Primary Key
Name	String	NOT NULL
Comment	String	NOT NULL
Rating	number	NOT NULL

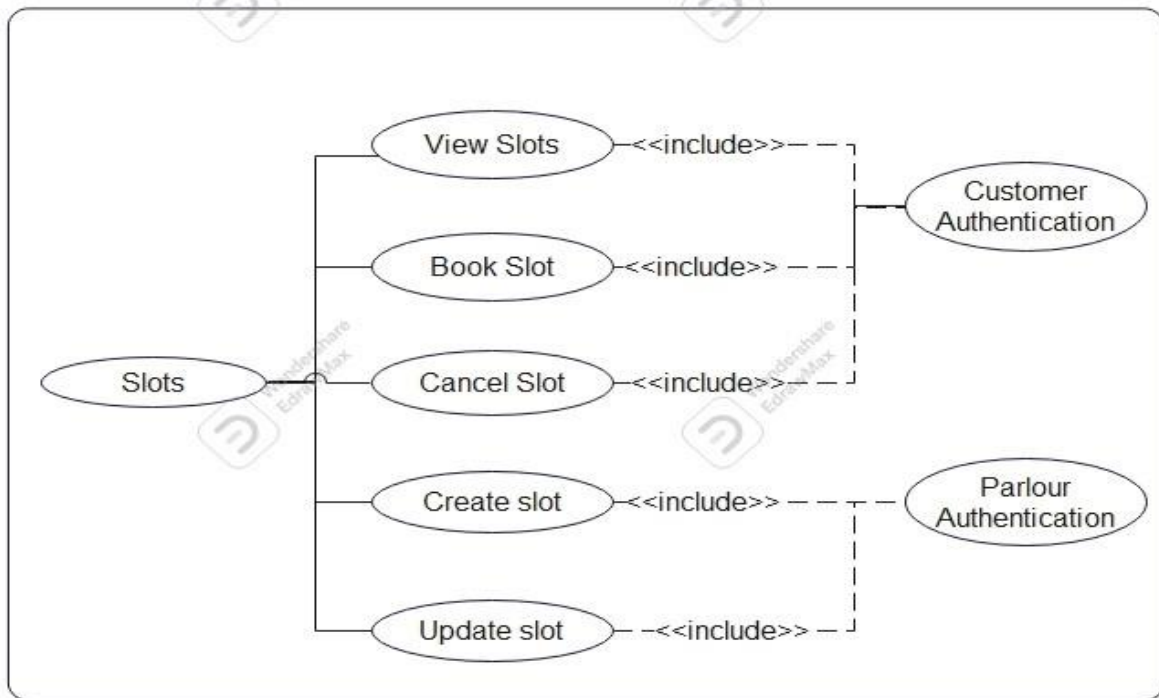
Parlour_email	String	Foreign Key
product_key	String	Foreign Key

3.4 Use Case Diagrams

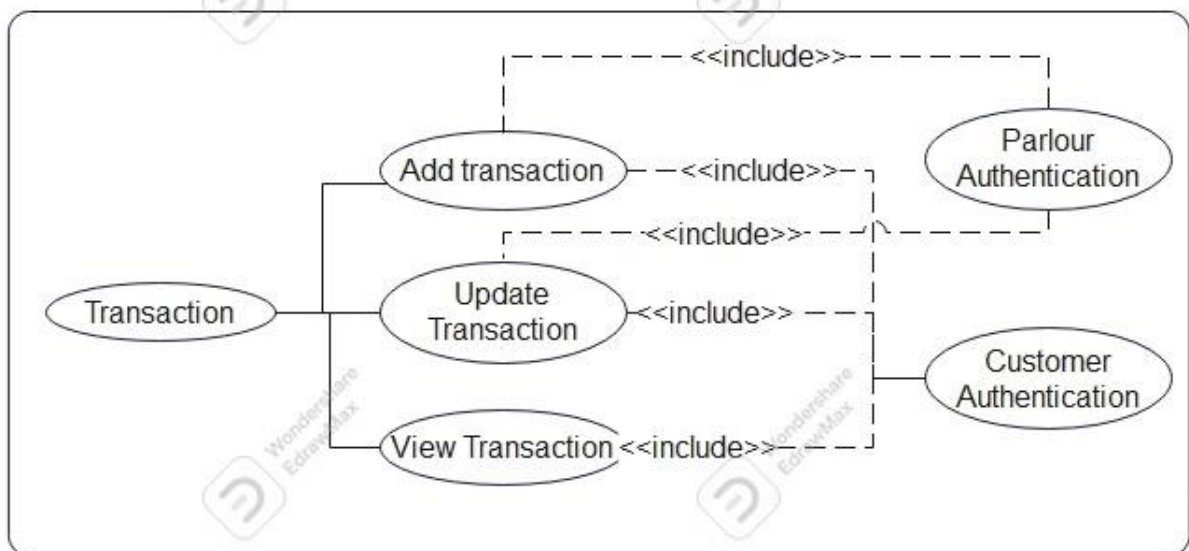
System: Use Case



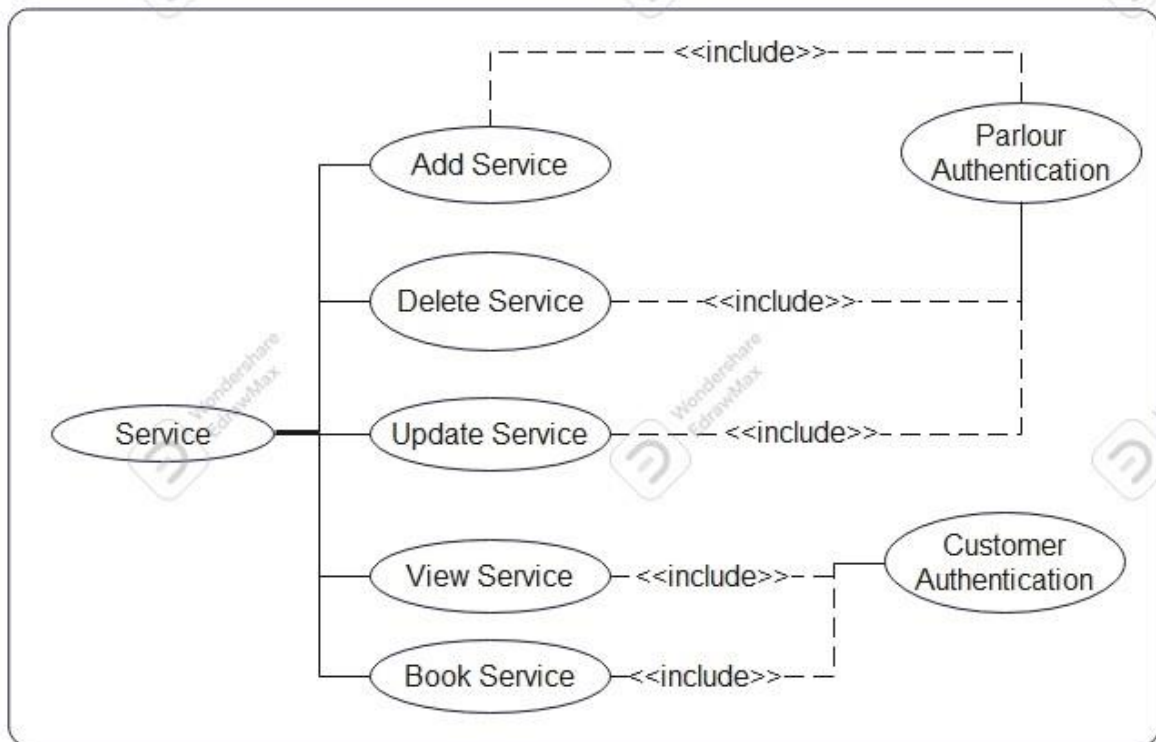
Slot:Use case



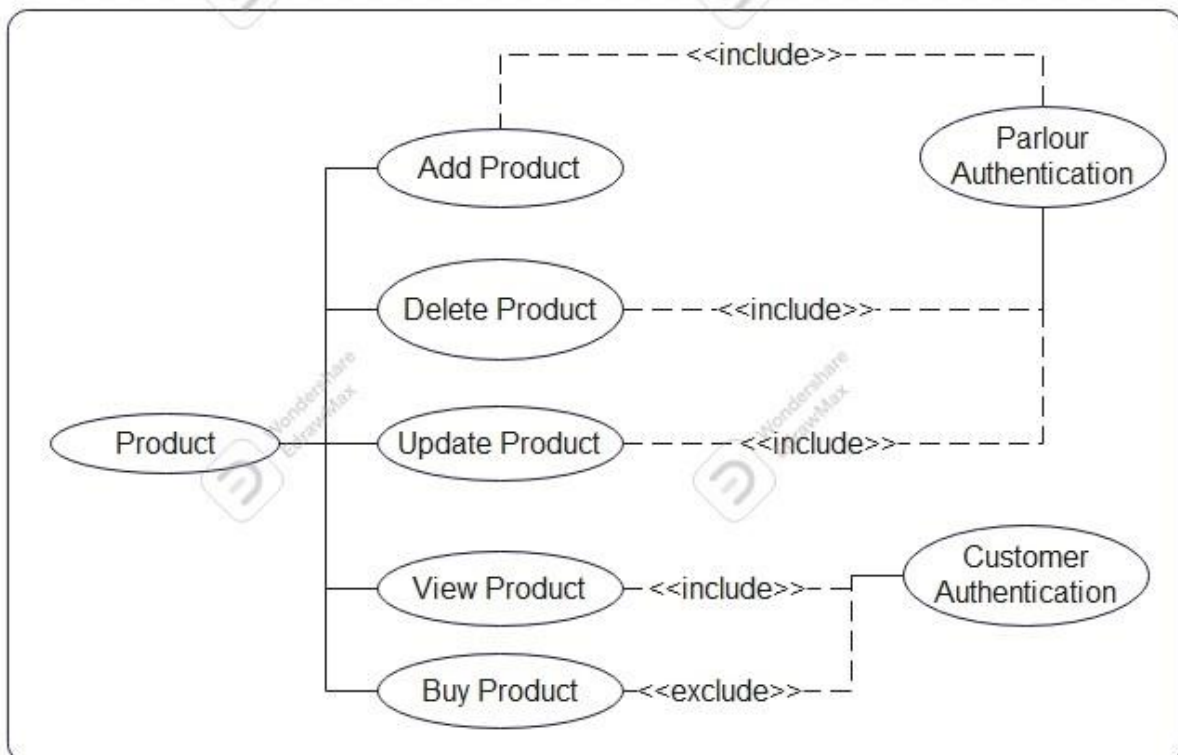
Transaction: Use case



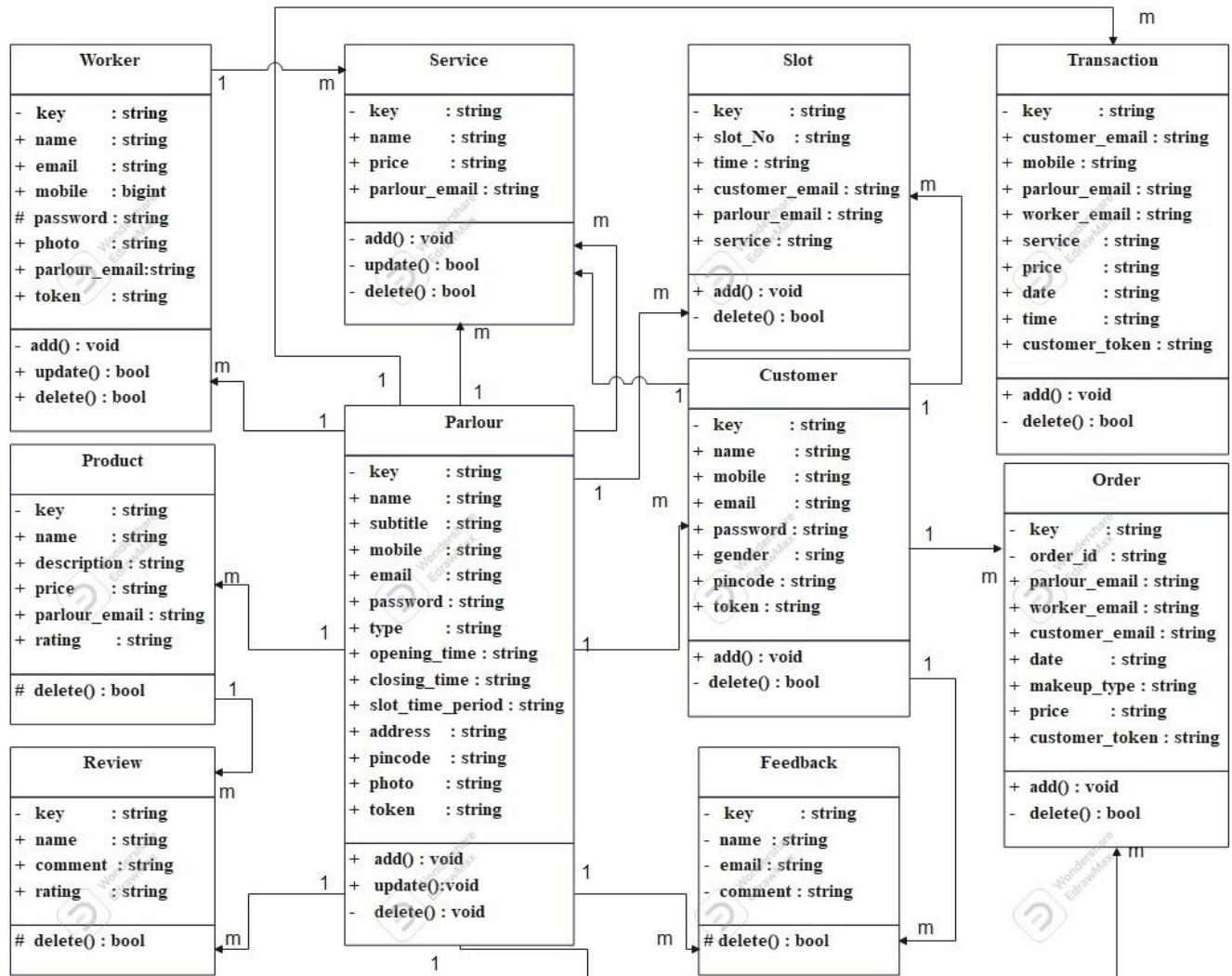
Service: Use case



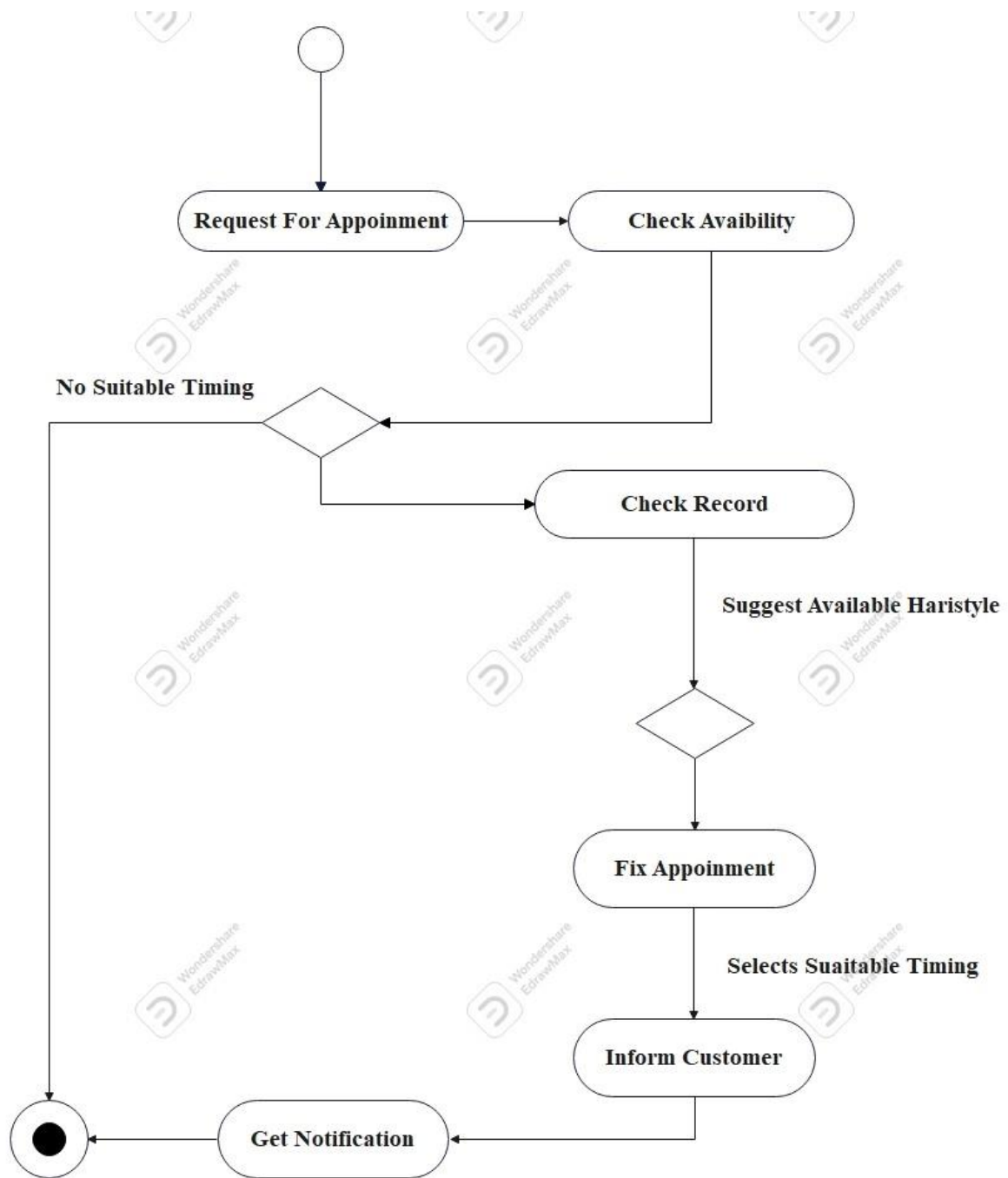
Product: Use case



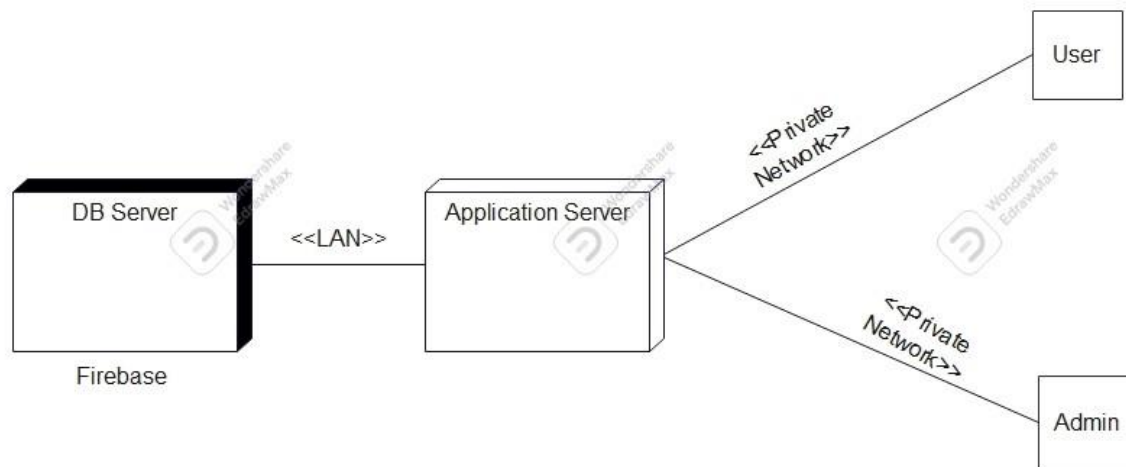
3.5 Class Diagram



3.6 Activity Diagram



3.7 Deployment Diagram



3.8 Module Hierarchy Diagram

Modules:

1. Slots
2. Transaction
3. Product
4. Services
5. Orders
6. Review
7. Feedback

Functionality:

4. Slot:

- Customer can book their slot time
- Salon creates slots according to their parlour opening time and closing time.
- Each slot has specific timing which have given by parlour owners choice.

5. Transaction:

- It manages all transactions in parlours.
- According to transactions data, parlour can notify their customers for their service.

6. Product:

- Parlour can add their product on this app.
- Customers can buy different product of different parlours by comparing price of products.

7. Services:

- Parlour provides many services like hair cutting, face wash, hair wash, makeup, sawing, etc.

8. Orders:

- Parlours can get orders for makeup through customers
- Customers can order makeup artist for their events.

9. Review:

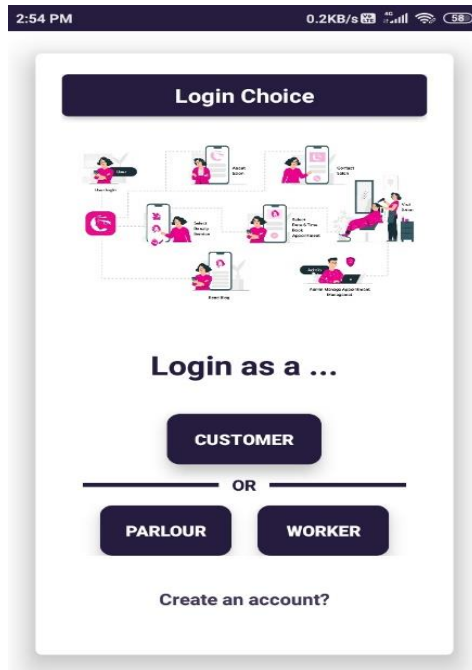
- The review of the salons.
- Manages salon quality according to customer's review

10. Feedback:

- Customers can give the best feedback.
- They can ask the admin anything and also, they are able to give suggestions to the admin.

3.9 Sample Input and Output Screens (Screens must have valid data. All reports must have at-least 5 valid records.)

Login Choice.xml



2:54 PM 0.2KB/s

Login Choice

Illustration showing various users (customer, worker, parlour) interacting with the app.

Login as a ...

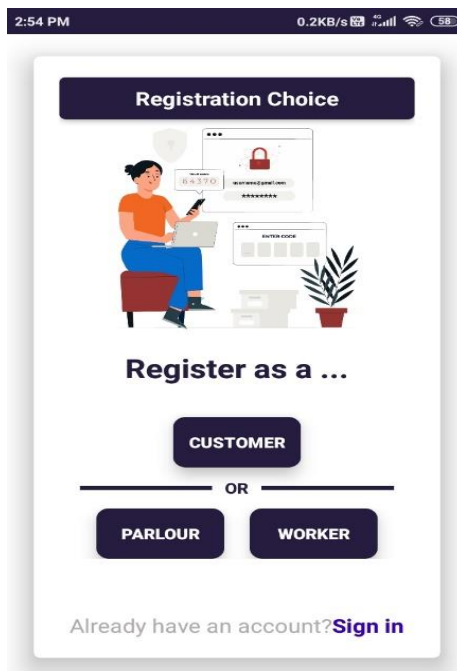
CUSTOMER

OR

PARLOUR **WORKER**

Create an account?

Registration Choice.xml



2:54 PM 0.2KB/s

Registration Choice

Illustration showing a user registering with a laptop and a smartphone displaying a registration form.

Register as a ...

CUSTOMER

OR

PARLOUR **WORKER**

Already have an account? [Sign in](#)

Parlour registration.xml

2:56 PM 1.9KB/s

Create Parlour Account

Please fill the input below here

T Goa mens

Unisex salon

8468569548

goaa@gmail.com

.....

.....

Choose parlour type

Unisex

9:00 AM

18:00 PM

pune

2:56 PM 0.6KB/s

goaa@gmail.com

.....

.....

Choose parlour type

Unisex

9:00 AM

18:00 PM

pune

411041

Choose Image UPLOAD


SIGN UP

Already have a account? [Sign in](#)

Parlour Login.xml


2:56 PM0.7KB/s


Parlour Login



Login

Please sign in to continue

 Username

 Password

LOGIN


Forgot Password?

Don't have an account? [Sign up](#)

Customer login.xml


2:56 PM0.8KB/s


Customer Login



Login

Please sign in to continue

 Username

 Password

LOGIN

Forgot Password?

Don't have an account? [Sign up](#)

Customer registration.xml

2:57 PM0.9KB/s

Create Customer Account

Please fill the input below here

Shubham Lagad

8007878524

shubhamlagad2000@gmail.com

.....

.....

Gender

Men

414545

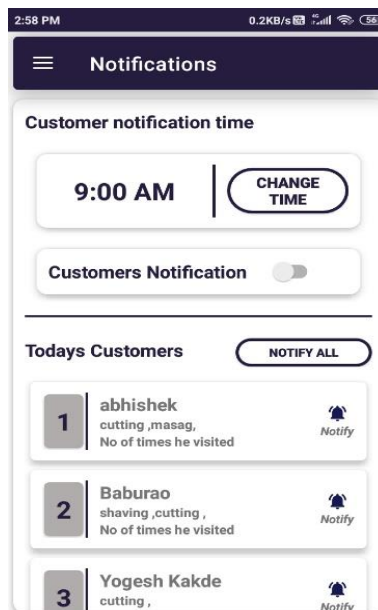
Parlour dashboard.xm



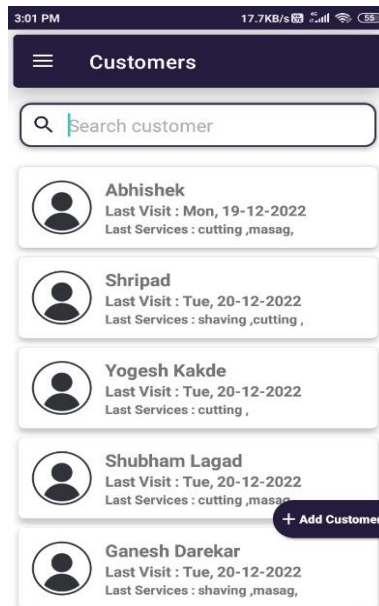
Parlour time slots.xml



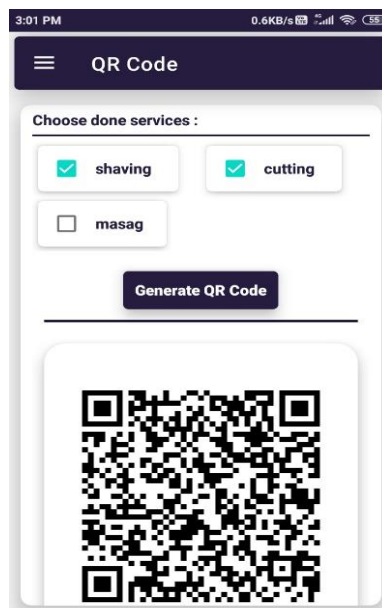
Notification.xml



Parlour customers.xml



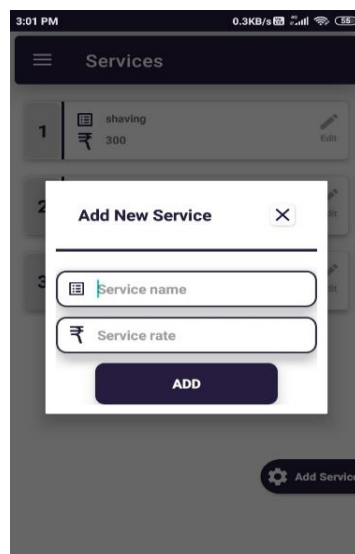
QR code.xml



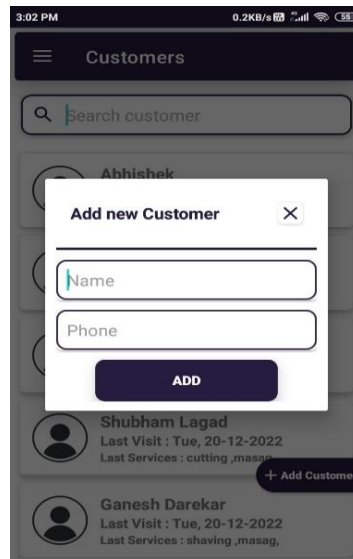
Parlour service.xml



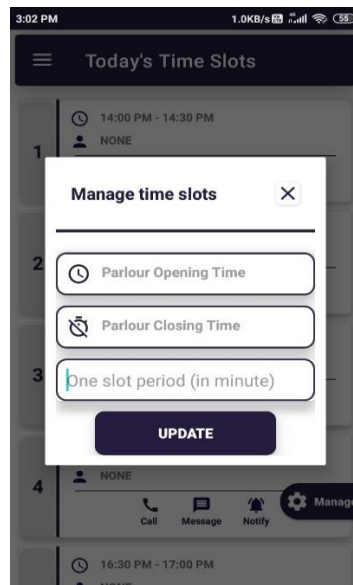
Add service.xml



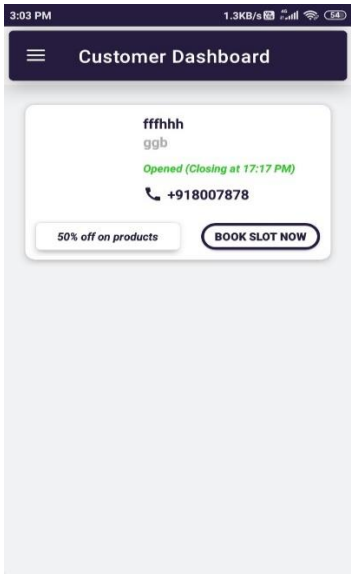
Add customer.xml



Manage slot.xml



Customer dashboard.xml



4. Coding

4.1 Code snippets

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.subhdroid.hairstylers">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/hair_stylers_icon"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/hair_stylers_icon_round"
        android:supportRtl="true"
        android:theme="@style/Theme.HairStylers"
        tools:targetApi="31">
        <activity
            android:name=".Customer.CustomerDashboard"
            android:exported="false" />
        <activity
            android:name=".Worker.WorkerRegistration"
            android:exported="false" />
        <activity
            android:name=".Worker.WorkerLogin"
            android:exported="false" />
        <activity
            android:name=".Parlour.ParlourRegistration"
            android:exported="true" />
        <activity
            android:name=".Parlour.ParlourLogin"
            android:exported="false" />
        <activity
            android:name=".Parlour.ParlourDashboard"
            android:exported="false" />
        <activity
            android:name=".Customer.CustomerRegistration"
            android:exported="false" />
        <activity
```

```

        android:name=".Customer.CustomerLogin"
        android:exported="false" />
    <activity
        android:name=".LoginChoice"
        android:exported="false" />
    <activity
        android:name=".RegistrationChoice"
        android:exported="false" />
    <activity
        android:name=".SplashActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".MainActivity"
        android:exported="false" />

    <service
        android:name=".CloudMessage.FirebaseMessagingService"
        android:exported="true">
        <intent-filter>
            <action android:name="com.google.firebase.MESSAGING_EVENT" />
        </intent-filter>
    </service>

    <receiver android:name=".CloudMessage.RecieverClass" />

</application>

</manifest>

```

Login Choice.java

```

package com.subhdroid.hairstylers;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.AppCompatButton;

import android.content.DialogInterface;

```

```

import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

import com.subhdroid.hairstylers.Customer.CustomerLogin;
import com.subhdroid.hairstylers.Parlour.ParlourLogin;
import com.subhdroid.hairstylers.Worker.WorkerLogin;

public class LoginChoice extends AppCompatActivity {
    AppCompatButton customerBtn, parlourBtn, workerBtn;
    TextView signUpTxt;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login_choise);

        customerBtn = findViewById(R.id.customerBtn);
        parlourBtn = findViewById(R.id.parlourBtn);
        workerBtn = findViewById(R.id.workerBtn);
        signUpTxt = findViewById(R.id.signUpTxt);

        customerBtn.setOnClickListener(view -> {
            Intent intent = new Intent(LoginChoice.this, CustomerLogin.class);
            startActivity(intent);
        });

        parlourBtn.setOnClickListener(view -> {
            Intent intent = new Intent(LoginChoice.this, ParlourLogin.class);
            startActivity(intent);
        });
    }
}

```



```

workerBtn.setOnClickListener(view -> {
    Intent intent = new Intent(LoginChoice.this, WorkerLogin.class);
    startActivity(intent);
});

signUpTxt.setOnClickListener(view -> {
    Intent intent = new Intent(LoginChoice.this, RegistrationChoice.class);
    startActivity(intent);
});

}

@Override
public void onBackPressed() {
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(LoginChoice.this);
    alertDialog.setTitle("Exit");
    alertDialog.setMessage("Do you want to exit app?");

    alertDialog.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            finishAffinity();
        }
    });

    alertDialog.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    });
}

```

```

    });

    alertDialog.show();

}
}

```

Registration choice.java

```

package com.subhdroid.hairstylers;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.AppCompatButton;

import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import com.subhdroid.hairstylers.Customer.CustomerLogin;
import com.subhdroid.hairstylers.Customer.CustomerRegistration;
import com.subhdroid.hairstylers.Parlour.ParlourLogin;
import com.subhdroid.hairstylers.Parlour.ParlourRegistration;
import com.subhdroid.hairstylers.Worker.WorkerLogin;
import com.subhdroid.hairstylers.Worker.WorkerRegistration;

public class RegistrationChoice extends AppCompatActivity {

```

```

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

    // getSupportActionBar().setTitle("Registration Choice");
    // ColorDrawable colorDrawable
    //     = new ColorDrawable(getResources().getColor(R.color.icon_color));
    // getSupportActionBar().setBackgroundDrawable(colorDrawable);

    AppCompatButton customerRegBtn, parlourRegBtn, workerRegBtn;
    TextView regChoiceSignInTxt;

    customerRegBtn = findViewById(R.id.customerRegBtn);
    parlourRegBtn = findViewById(R.id.parlourRegBtn);
    workerRegBtn = findViewById(R.id.workerRegBtn);
    regChoiceSignInTxt = findViewById(R.id.regChoiceSignInTxt);

    customerRegBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(RegistrationChoice.this, CustomerRegistration.class);
            startActivity(intent);
        }
    });

    parlourRegBtn.setOnClickListener(new View.OnClickListener() {

```

```

@Override
public void onClick(View view) {
    Intent intent = new Intent(RegistrationChoice.this, ParlourRegistration.class);
    startActivity(intent);
}
});

workerRegBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(RegistrationChoice.this, WorkerRegistration.class);
        startActivity(intent);
    }
});

regChoiceSignInTxt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(RegistrationChoice.this, LoginChoice.class);
        startActivity(intent);
    }
});

}

@Override
public void onBackPressed() {
    super.onBackPressed();
    finish();
    Intent intent = new Intent(RegistrationChoice.this, LoginChoice.class);
    startActivity(intent);
}

```

```
}  
}
```

Registration choice.xml

```
<?xml version="1.0" encoding="utf-8"?>  
  
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    tools:context=".RegistrationChoice">  
  
    <androidx.cardview.widget.CardView  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:layout_margin="21dp"  
        app:cardCornerRadius="5dp"  
        app:cardElevation="20dp">  
  
        <LinearLayout  
            android:layout_width="match_parent"  
            android:layout_height="match_parent"  
            android:orientation="vertical"  
            android:padding="20dp">  
  
                <androidx.cardview.widget.CardView  
                    android:layout_width="match_parent"  
                    android:layout_height="40dp"
```

```
app:cardCornerRadius="5dp"  
app:cardElevation="5dp">
```

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="@color/icon_color"  
    android:gravity="center"  
    android:text="Registration Choice"  
    android:textAlignment="center"  
    android:textColor="@color/white"  
    android:textSize="18sp"  
    android:textStyle="bold" />
```

```
</androidx.cardview.widget.CardView>
```

```
<com.airbnb.lottie.LottieAnimationView  
    android:layout_width="match_parent"  
    android:layout_height="200dp"  
    app:lottie_autoPlay="true"  
    app:lottie_loop="true"  
    app:lottie_rawRes="@raw/registration_choise_animation" />
```

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="20dp"  
    android:gravity="center"  
    android:orientation="vertical">
```

```
<TextView  
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:layout_marginBottom="20dp"
android:text="Register as a ..."
android:textColor="@color/icon_color"
android:textSize="25sp"
android:textStyle="bold" />
```

```
<androidx.cardview.widget.CardView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginVertical="10dp"
    app:cardCornerRadius="10dp"
    app:cardElevation="10dp">
```

```
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/customerRegBtn"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:background="@color/icon_color"
    android:text="Customer"
    android:textColor="#fff"
    android:textStyle="bold" />
```

```
</androidx.cardview.widget.CardView>
```

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true">
```

```
<TextView
    android:id="@+id/tvText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:text="OR"
    android:textColor="@color/icon_color"
    android:textStyle="bold" />
```

```
<View
    android:layout_width="match_parent"
    android:layout_height="4dp"
    android:layout_centerVertical="true"
    android:layout_marginLeft="16dp"
    android:layout_toLeftOf="@id/tvText"
    android:background="@color/icon_color" />
```

```
<View
    android:layout_width="match_parent"
    android:layout_height="4dp"
    android:layout_centerVertical="true"
    android:layout_marginRight="16dp"
    android:layout_toRightOf="@id/tvText"
    android:background="@color/icon_color" />
```

```
</RelativeLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
```



```
android:layout_height="wrap_content"
android:layout_marginVertical="10dp"
android:gravity="center"
android:orientation="horizontal">
```

```
<androidx.cardview.widget.CardView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="10dp"
    app:cardCornerRadius="10dp"
    app:cardElevation="10dp">
```

```
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/parlourRegBtn"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:background="@color/icon_color"
    android:text="Parlour"
    android:textColor="#fff"
    android:textStyle="bold" />
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    app:cardCornerRadius="10dp"
    app:cardElevation="10dp">
```

```
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/workerRegBtn"
```

```

        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:background="@color/icon_color"
        android:text="Worker"
        android:textColor="#fff"
        android:textStyle="bold" />
    </androidx.cardview.widget.CardView>

```

```

</LinearLayout>

```

```

</LinearLayout>

```

```

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginTop="41dp"
    android:orientation="horizontal">

```

```

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Already have an account?"
        android:textColor="@color/muted_text"
        android:textSize="18sp" />

```

```

    <TextView
        android:id="@+id/regChoiceSignInTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sign in"

```

```
        android:textColor="@color/purple_700"
        android:textSize="18sp"
        android:textStyle="bold" />
    </LinearLayout>

</LinearLayout>
</androidx.cardview.widget.CardView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

5. Testing

5.1 Strategies used for Testing

1. Unit Testing

Unit testing concentrates verification on the smallest element of the program - the module. Using the detailed design description important control paths are tested to establish errors within the bounds of the module.

2. Integration testing

Once all the individual units have been tested there is a need to test how they were put together to ensure no data is lost across interface, one module does not have an adverse impact on another and a function is not performed correctly.

5.2 System testing for the current system:

In this level of testing, we are testing the system as a whole after integrating all the main modules of the project. We are testing whether system is giving correct output or not. All the modules were integrated and the flow of information among different modules was checked. It was also checked that whether the flow of data is as per the requirements or not. It was also checked that whether any particular module is non-functioning or not i.e., once the integration is over each and every module is functioning in its entirety or not. In this level of testing, we tested the following: - Whether all the forms are properly working or not. Whether all the forms are properly linked or not. Whether all the images are properly displayed or not. Whether data retrieval is proper or not.

5.3 Test Cases

Testing is the exposure of system to trial input to see whether it produces correct output. Testing assumes requirements that are already validated. Testing cannot guarantee correctness, no method can guarantee correctness. Testing is the process of detecting presence of faults. Debugging is the process of locating and correcting faults. Once source

code has been generated, software must be tested to uncover as many errors as possible before delivery to your customer. Our goal is to design a series of test cases that have a high likelihood of finding errors. That's where Software Testing techniques enter the picture. A set of test cases designed to exercise both internal logic and external requirements is designed and documented, expected results are defined and actual results are recorded.

5.4 Test Log

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	<ol style="list-style-type: none"> 1. Go to app 2. Enter UserId 3. Enter Password 4. Click Submit 	Userid = salon Password = pass99	User should Login into an application	As Expected	Pass
TU02	Check Customer Login with invalid Data	<ol style="list-style-type: none"> 1. Go to app 2. Enter UserId 3. Enter Password 4. Click Submit 	Userid = salon Password = salons99	User should not Login into an application	As Expected	Pass

Step 1) A simple test case to explain the scenario would be

Test Case #	Test Case Description
1	Check response when valid email and password is entered

Step 2) Test the Data.

In order to execute the test case, you would need Test Data. Adding it below

Test Case #	Test Case Description	Test Data
1	Check response when valid email and password is entered	Email: salon@email.com Password: lNf9^Oti7^2h

Identifying test data can be time-consuming and may sometimes require creating test data afresh. The reason it needs to be documented.

Step 3) Perform actions.

In order to execute a test case, a tester needs to perform a specific set of actions on the AUT. This is documented as below:

Test Case #	Test Case Description	Test Steps	Test Data
1	Check response when valid email and password is entered	1) Enter Email Address 2) Enter Password 3) Click Sign in	Email: salon@email.com Password: INf9^Oti7^2h

Many times, the Test Steps are not simple as above, hence they need documentation. Also, the author of the test case may leave the organization or go on a vacation or is sick and off duty or is very busy with other critical tasks. A recently hire may be asked to execute the test case. Documented steps will help him and also facilitate reviews by other stakeholders.

Step 4) Check behaviour of the AUT.

The goal of test cases in software testing is to check behaviour of the AUT for an expected result. This needs to be documented as below

Test Case #	Test Case Description	Test Data	Expected Result
1	Check response when valid email and password is entered	Email: salon@email.com Password: INf9^Oti7^2h	Login should be successful

During test execution time, the tester will check expected results against actual results and assign a pass or fail status

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check response when valid email and password is entered	Email: salon@email.com Password: INf9^Oti7^2h	Login should be successful	Login was successful	Pass

Step 5) That apart your test case -may have a field like,

Pre – Condition which specifies things that must be in place before the test can run. For our test case, a pre-condition would be to have a browser installed to have access to the site under

test. A test case may also include Post – Conditions which specifies anything that applies after the test case completes. For our test case, a postcondition would be time & date of login is stored in the database.

6. Limitations of Proposed System

- Cannot work in IOS devices.
- Cannot run without internet.
- UI only in light mode.

7.Proposed Enhancements

- I have provided best service for app users.
- Customers can book their time slots for haircut.
- Salon owner can store their customers data and remind for their hair cut in particular time.
- If any customer forgot to cut hair in his time period, then system will automatically remind it day by day

8. Conclusion

To conclude the description about project. The project developed using Android, Java, and Firebase is based on requirement specification of the user and the analysis of the existing user, with flexibility for future enhancement. The following conclusions can be deduced from the developed of the project.

- Automation of the entire system improves the efficiency.

- It provides a friendly graphical user interface which proves to be better when compared to the existing system.
- It gives appropriate access to the authorized users depending on their permissions.
- Updating of information is becomes so easier.
- The system has adequate scope for modification in future if it is necessary.
- This system has very softly to handle and execute.
- Work done manually is much slower compared with this system, which is must faster.
- Increase performance
- The system has less time consuming
- More efficient
- Highly flexible

9. Bibliography

- **Reference Books:**

ANDROID APP DEVELOPMENT

-By J. Paul Cardle

STARTING WITH ANDROID

-By M.M. Sharma

ANDROID PROGRAMMING GUIDE

-By Bharti N. Raut

- **Websites:**

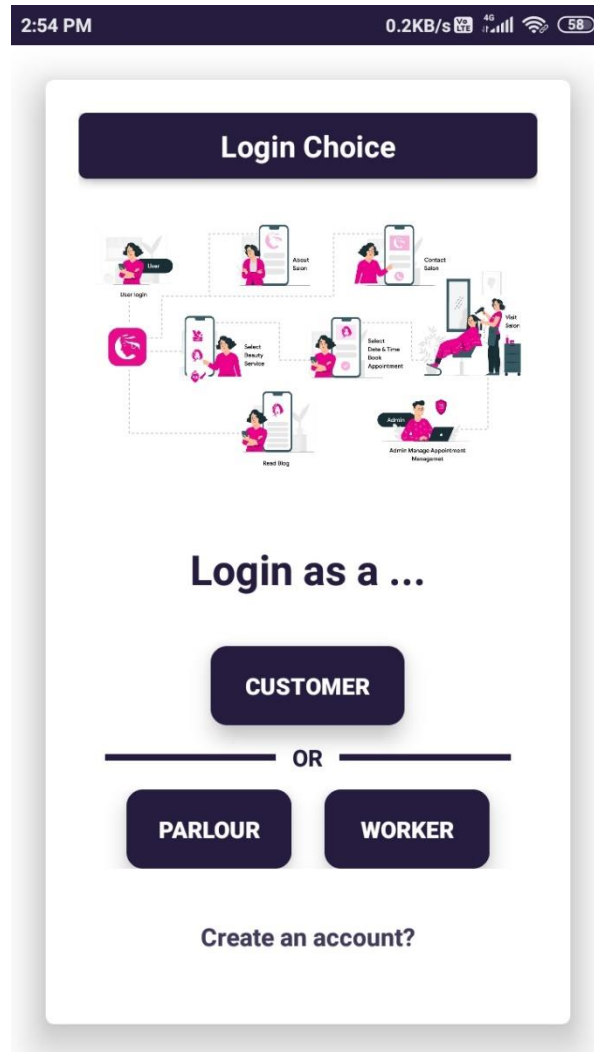
<https://console.firebase.google.com/>

<https://developer.android.com/studio>

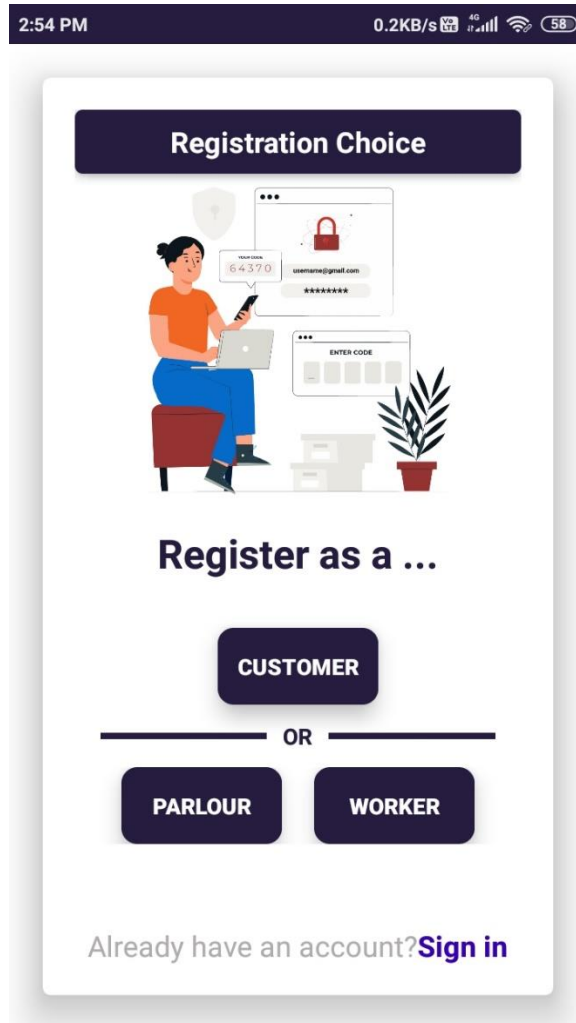
<https://developer.android.com/guide/components/activities/intro-activities>

<https://developer.android.com/guide/fragments>

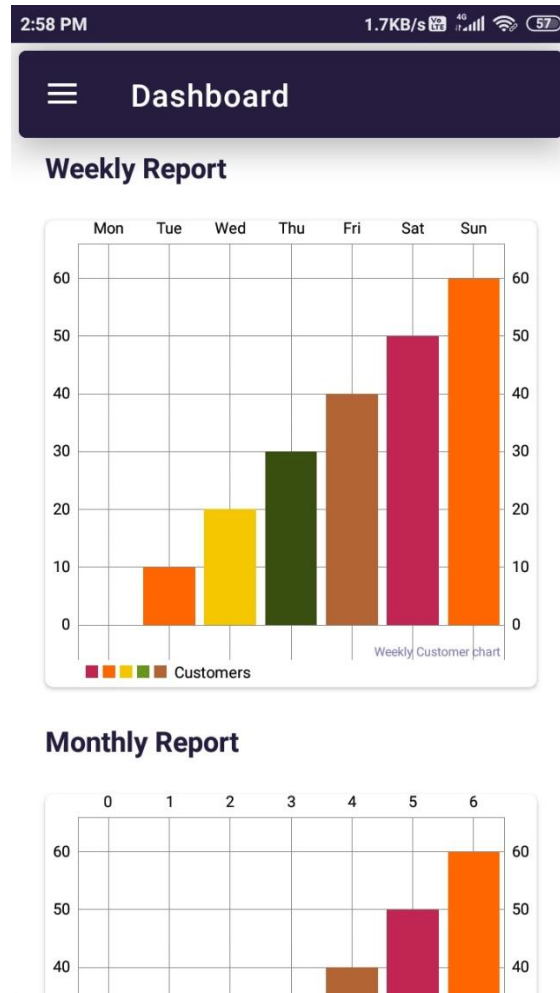
12.User Manual



This is login choice screen. Customers can click on customer button and they can login to the application. Also, users have option to create account.

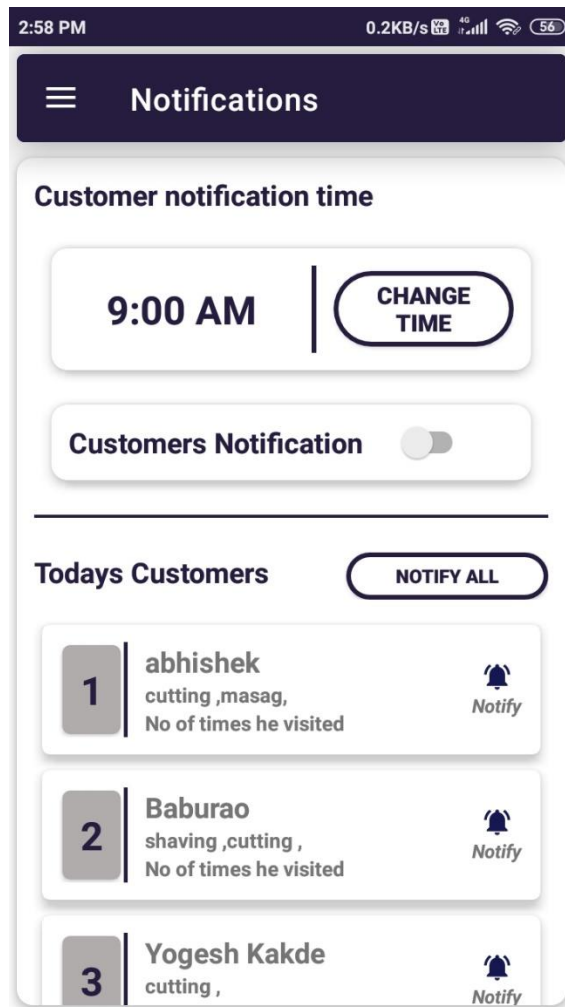


This is registration choice screen with users can select there choice to create account. I users have already account then they will click on sign in option then they can redirect to the login choice page.

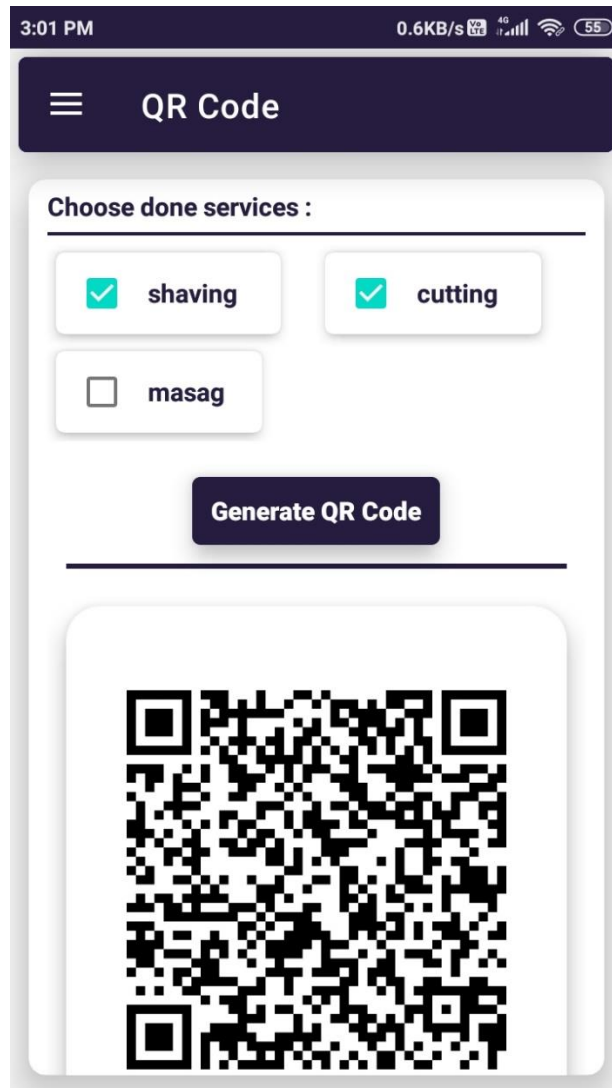


This is parlour dashboard screen. This screen shows all reports of parlours. Like parlour weekly customers, monthly customers, yearly customers, etc.

This is also showing parlour yearly income or growth.



This is customers notification screen parlours can notify their customers for their hair cut. They can notify there all todays customers. Also, they can add remainder for sending notification for their customers.



This page generate parlour service QR code. Customers scan these QR code and they can automatically add to the parlour customers list. Then they can easily get notification time to time.