test

by Buddhini De Silva

Submission date: 24-Apr-2023 08:40AM (UTC-0600)
Submission ID: 2073833405
File name: Cupcake.docx (643.57K)
Word count: 7252
Character count: 52175

Table of Contents

Table of Contents

Table of Contents 1

Table of figures 2

Table of Tables ..... **Error! Bookmark not defined.**

TASK 01 2

TASK 02 6

TASK 03 18

Design User Interfaces 18

TASK 04 23

TASK 05 50

Plan and documentation for tests 51

Objectives 51

Test Cases 51

TASK 06 58

User Manual 58

General information 58

Application overview 59

Organization of the Manual 59

User privileges (Admin, Customer) 60

References 60

Table of figures

Figure 01 Use Case diagram 6

Figure 02 Use Case 02 Customer vs Admin .....**Error! Bookmark not defined.**

Figure 03 Use case Admin .....**Error! Bookmark not defined.**

Figure 05 Class Diagram 17

Figure 06 Activity Diagram 18

Figure 7 20

Figure 8 21

Figure 9 21

Figure 10 22

Figure 11 22

Figure 12 23

Figure 07 Splash Screen..... 53

Figure 08 Login ui 55

Figure 09 Home UI 56

Figure 10 Signup UI..... 57

TASK 01

Let's examine mobile operating systems, development tools, and technologies for a cupcake bakery app that offers functionality for admin users and member users and themed cupcakes for occasions like Valentine's Day, graduation, Mother's Day, new baby, etc.

- Mobile Operating Systems:

- Android: Google created the well-known Android mobile operating system, which is used by numerous mobile devices. Because to its significant worldwide market share, it is a good option for appealing to a huge user base. For creating mobile apps, Android offers a wide range of development resources, including the Android Studio IDE and Java/Kotlin programming languages.
- iOS: Apple created the iOS mobile operating system for its iPhone and iPad gadgets. It is well-known for its user-friendly interface and stringent software quality standards, and it has a significant presence in the premium part of the mobile market. Swift/Objective-C programming languages and the Xcode IDE are commonly needed for iOS app development.

Since the current system is based on android, it's better using Android OS.

- Development Tools:
  - The official IDE for developing Android apps, Android Studio offers a complete collection of tools for planning, writing, debugging, and testing Android apps. Also, it has emulators built in for testing apps on various Android devices and releases.
- Technologies:
  - The main programming languages used for creating Android apps are Java and Kotlin. The conventional language for Android programming has been Java, although Kotlin has become more well-liked recently due to its cutting-edge syntax and increased developer efficiency.

Use of Android Studio is strongly recommended for creating Android apps in Java for the following reasons,

- The official Integrated Development Environment (IDE) for Android app: development was created and is maintained by Google as part of Android Studio. It offers a thorough and feature-rich development environment made especially for building Android applications.

- Java-based development: Java is a well-liked and widely-used programming language for creating Android applications. Developers may write clear and effective code with the aid of the powerful Java support offered by Android Studio, which includes sophisticated code editing, debugging, and refactoring tools.

- Comprehensive variety of tools and features: Android Studio has a wealth of capabilities that make the process of developing apps easier. A visual layout editor, an effective code editor with auto-completion and refactoring, a built-in emulator for testing apps on multiple virtual devices, and a flexible build system with support for Gradle are a few of them.
- Strong debugging and testing tools: Android Studio provides strong debugging and testing tools that make it simple for developers to find and repair faults in their apps. It offers a variety of debugging tools, including real-time code inspection, breakpoints, and watchpoints, to aid in locating and fixing problems as they arise throughout the development process.
- Wide ecosystem of libraries and plugins: Android Studio has access to a wide range of libraries and plugins that enhance its features and speed up the creation of apps. By utilizing the built-in dependency management mechanism of Android Studio, these libraries and plugins offer extra functionality like UI components, networking, database access, and more that can be quickly integrated into the project.
- Connected with Google services: To build contemporary Android apps, Android Studio is tightly linked with a number of Google services, including Firebase, Google Cloud Platform, and Google Play Services. The smooth access to these services made possible by this integration makes it simpler to include features like push notifications, cloud storage, authentication, and more in Android apps.
- Robust developer community and support: There are a ton of learning materials and troubleshooting tools accessible thanks to the big and active developer community that exists for Android Studio. To keep Android Studio current with the newest platform features, tools, and best practices, Google regularly updates and supports it.

Benefits of Creating a Mobile App for a Cupcake Shop Using Java Technology, Android Studio IDE, and SDK 31,

- Huge User Base: With a sizable user population globally, Android is the most popular mobile operating system. You may take advantage of this sizable market and reach a larger audience by creating an Android app, thereby growing your consumer base.
- Strong Development Environment: The official integrated development environment for developing Android apps, Android Studio IDE, offers a robust and complete collection of tools for creating high-caliber apps. Development is facilitated by its capabilities, which include code completion, debugging, and performance profiling.
- Java is a popular programming language that has a sizable developer community and a ton of documentation. It is possible to construct Android apps more quickly by using Java because there are so many tools for troubleshooting and problem-solving. Java is a solid option for creating safe mobile apps because of its reliability and security reputation.
- Newest Platform and SDK Features: Android Oreo (O) and SDK 31 (or above) provide the newest features and enhancements in terms of usability, security, and performance. They include enhanced notification channels, adaptable icons, and other features. You may offer a cutting-edge app with improved functionality and user experience by employing the most recent SDK and platform capabilities.
- Flexibility and Customization: Building a mobile app from the ground up with Java and Android Studio enables total customization. You may create the app to fit the requirements and branding of your cupcake store, giving your consumers a distinctive and tailored experience. To give your app extra features like payment gateways, social network integration, and more, you can also use a variety of third-party libraries and APIs.
- Android applications include a variety of revenue options, including in-app purchases, advertising, and subscription models. Creating a Cupcake shop app allows you to experiment with multiple income streams and improve the profitability of your company.
- The Android ecosystem is seamlessly integrated with apps created with Java and Android Studio, including the Google Play Store, Google services (including Google Maps, Firebase, etc.), and other Android-powered devices. By doing this, you can make the

most of the Android platform's capabilities and give your consumers a uniform, consistent experience.

In summary, using Java technology, Android Studio IDE, and the most recent SDK and platform features to develop a mobile app for a cupcake shop has many benefits, including access to a sizable user base, a stable development environment, customization and flexibility, monetization opportunities, and seamless integration with the Android ecosystem.

TASK 02

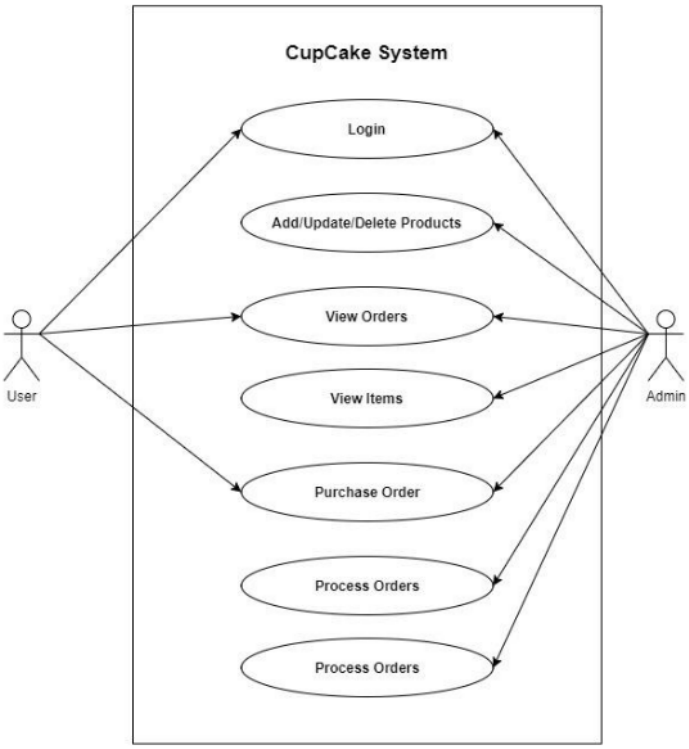

Figure 01 Use Case diagram

Use Case ID :	TW_001
Use Case Name:	Sign-Up/ Registration
Created By:	Test
Date Created:	10-9-2019
Description:	The way for a user to access the program is made by this use case. Customers must register with the system by providing a valid Email ID and Password if they are new to the program and need to access it with full rights.
Primary Actor:	Customer
Secondary Actor:	Admin
Include Use Cases:	1. Enter your email ID. 2. Type your full name. 3. Type your password. 4. Enter Your Password Again.
Extend Use Cases:	1. Display registration successful. 2. Display registration error.
Preconditions:	1. Uninterrupted Internet connection. 2. User must download and install the app on their android mobile. 3. After that open the App and reached the main UI. 4. User is not logged in yet.
Post conditions:	1. The application redirects to Login page.

Main Flows:	1. User enters a correct email address. 2. The user enters their full name. 3. Password is entered by user. 4. User enters new password and clicks "Submit." 5. The program verifies that the email ID is not already in use. 6. The software checks to see if the two passwords match. 7. Using the specified settings, the application registers the new player (Email ID, password)
Alternative Flows:	A. A. The database for the application already has an Email ID. 1. The application shows the notification "Email ID already exists." 2. The application asks for a new Email ID once more. B. A. No two passwords are the same. 1. The application shows a notice that reads "Passwords do not match." 2. In the confirm password area, the application requests that you "type the same password" once again.

Table 01 Signup Use Case

Use Case ID :	TW_001
Use Case Name:	Sign-Up/ Registration
Created By:	Test
Date Created:	10-9-2019
Description:	The way for a user to access the program is made by this use case. Customers must register with the system by providing a valid Email ID and Password if they are new to the program and need to access it with full rights.
Primary Actor:	Customer
Secondary Actor:	Admin
Include Use Cases:	5. Enter Email ID. 6. Enter Full Name. 7. Enter a password. 8. Re-enter Password.
Extend Use Cases:	3. Display registration successful. 4. Display registration error.
Preconditions:	5. Uninterrupted Internet connection. 6. User must download and install the app on their android mobile. 7. After that open the App and reached the main UI. 8. User is not logged in yet.
Post conditions:	2. The application redirects to Login page.

Main Flows:	8. User enters a legitimate Email ID. 9. The user enters Complete Name. 10. Password is entered by user. 11. User enters new password and clicks "Submit." 12. The program verifies that the email ID is not already in use. 13. The program checks to see if the two passwords are the same. 14. Using the specified settings, the application registers the new player (Email ID, password)
Alternative Flows:	C. The database for the application already has an Email ID. 1. The application shows the notification "Email ID already exists." 2. The application asks for a new Email ID once more. D. Passwords are not all the same. 1. The application shows a notice that reads "Passwords do not match." 2. In the confirm password area, the application requests that you "type the same password" once again.

Table 02 Login UseCase

Use Case ID :	TW_003
Use Case Name:	View Products
Created By:	Test
Date Created:	10-9-2019
Description:	A customer can view a particular product by clicking on them and they can see the name, price, also the description of the product.
Primary Actor:	User (Admin, Customer)
Secondary Actor:	None
Include Use Cases:	-
Extend Use Cases:	1. Add to favorites. 2. Add to cart.
Preconditions:	1. User must register in the application database and has to have a valid account and visit the Cupcake main UI and click on a particular product.
Post conditions:	1. The application displays the relevant UI with Cupcake' name, price and description.
Main Flows:	1. Open the Android application. 2. Login to the application. 3. Visit the Cupcake home UI. 4. Click on a particular product. 5. View the details. 6. Uses add to favorites / add to cart options.

Alternative Flows:	A. User can close the application without even viewing them. B. User can scroll down and see whole other products in simple manner without viewing a particular product.

Table 03 View Products

Use Case ID:	TW_004
Use Case Name:	Place order.
Created By:	Test
Date Created:	10-9-2019
Description:	A customer can view a particular product by clicking on them and they can place an order.
Primary Actor:	Customer
Secondary Actor:	Admin
Include Use Cases:	1. Confirm Order. 2. Cancel Order.
Extend Use Cases:	-
Preconditions:	1. User must register in the android application database and has to have a valid account and visit the Cupcake main UI also user should click on a particular product and choose place an order option and after that confirm the purchase.
Post conditions:	2. Redirects to make payment interface.
Main Flows:	1. Open the Android application. 2. Login to the application. 3. Visit the Cupcake home UI. 4. Click on a particular product. 5. Place an order. 6. Confirm the purchase/ Cancel the order option.

Alternative Flows:	A. Cancel the order placement. 1. Go back to previous page with Cupcake' details.

Table 04 Place order Use Case

Use Case ID :	TW_006
Use Case Name:	Manage Items.
Created By:	Test
Date Created:	10-9-2019
Description:	Admin can Add, update, and delete the details of the Cupcake.
Primary Actor:	Admin
Secondary Actor:	-
Include Use Cases:	-
Extend Use Cases:	-
Preconditions:	User must login to the system as an Admin.
Post conditions:	After login application should display the Admin Panel.
Main Flows:	1. Login to the database as an Admin. 2. View the product details. 3. Add/ modify/ delete product details.
Alternative Flows:	-

Table 05 Manage Items Use Case

Use Case ID :	TW_008
Use Case Name:	Process Orders.
Created By:	Test
Date Created:	10-9-2019
Description:	After the placement of an order by a customer the Admin should process those orders.
Primary Actor:	Admin
Secondary Actor:	-
Include Use Cases:	-
Extend Use Cases:	-
Preconditions:	User must login to the system as an Admin.
Post conditions:	After login application should display the Admin Panel.
Main Flows:	1. Login to the database as an Admin. 2. Check the orders. 3. Check the stocks and ship the product if available.
Alternative Flows:	-

Table 06 Process orders Use case.

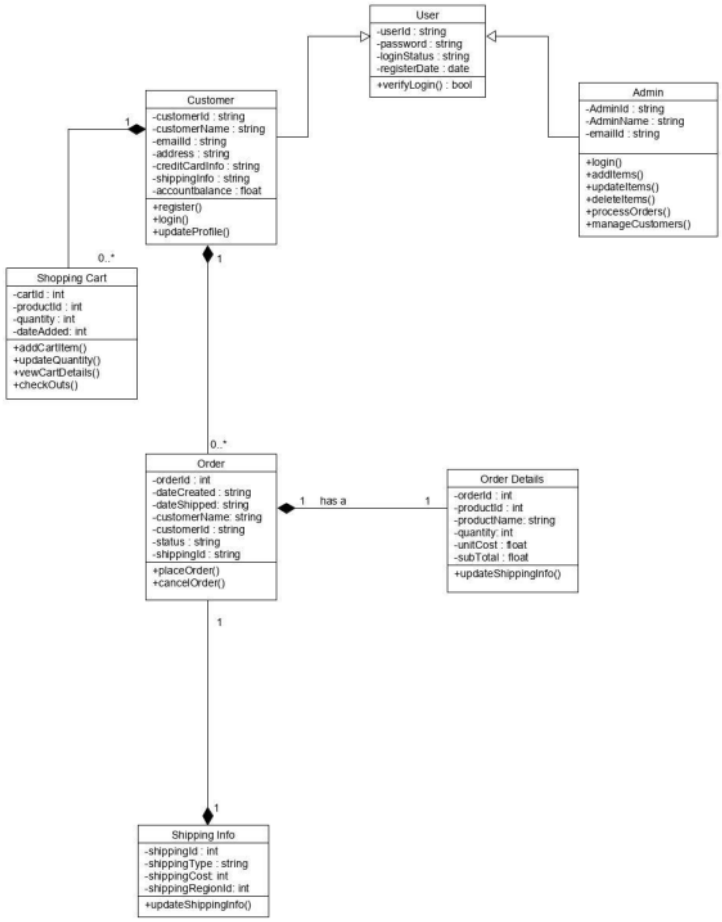

Figure 05 Class Diagram

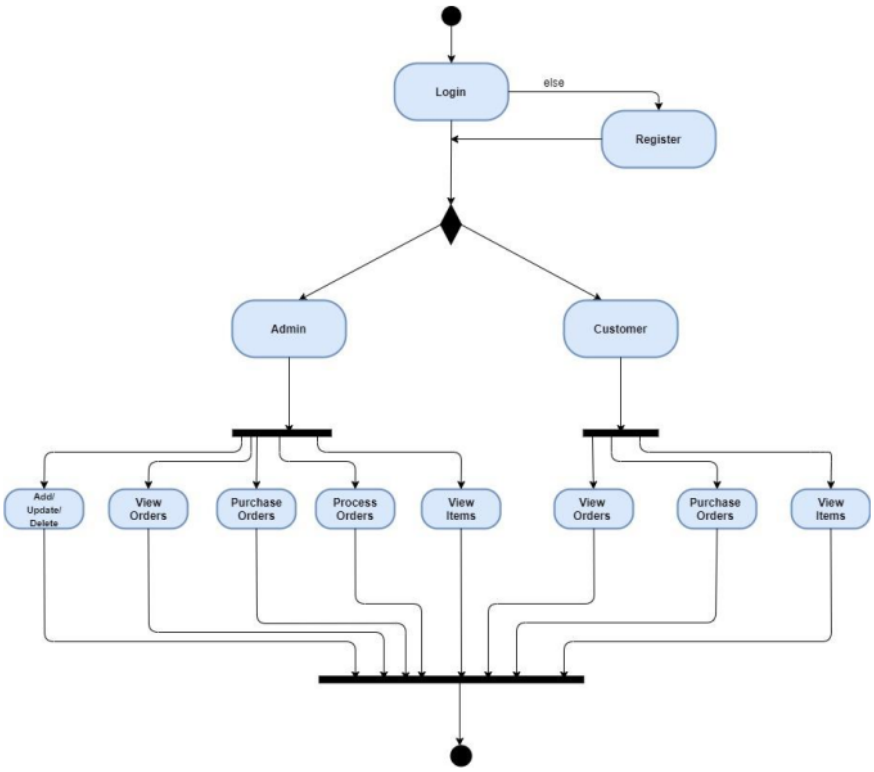

Figure 06 Activity Diagram

TASK 03

Design User Interfaces

As this chapter is devoted to the project's designing phase, it covers all the User Interfaces, or UIs, that are designed.

We utilized the open-source Adobe XD tool, which is specialized in user interface (UI) design for Android/iOS applications and websites, to complete this exercise.

These are the justifications for using Adobe XD as the main design tool.

- A user may generate mockups of their prototypes in Adobe XD and test them realistically on the screen without having to code in Android Studio. It could also provide a clearer understanding of how the designs operate.
- It also assists the developers by letting them know exactly how the product should be constructed. (brandablr.com)
- The UI designer is familiar with Adobe's other creating tools, including Photoshop, Illustrator, and Fireworks, and they are all quite complicated. Yet in contrast to such technologies, Adobe XD is far more straightforward and capable of producing user interfaces with less effort.
- Both Mac and Windows OSes can run Adobe XD.
- Adobe now offers a free starting subscription that allows users to share just one computer, have 2 GB of limited cloud storage, and distribute a small selection of fonts from the Adobe Creative Cloud collection. When compared to competing design software, this is a helpful feature.
- Adobe XD is available in English, French, German, Japanese, and Korean, giving users the option to select their chosen language.
- The designing tool has a straightforward issue with Windows OS, namely that it can only work on machines running Windows 10 or later and is unable to support earlier versions of Windows. Thankfully, Windows 10 is supported by our devices, making Adobe XD appropriate and trouble-free to use.
- Because of XD's straightforward interface, designers may quickly get used to it and move along with it. Moreover, it has somewhat less capability, but just enough to create amazing apps or websites. (Source: medium.com)
- It's simple to add content to UI; for instance, a designer may drag and drop any picture from a folder or browser onto a placeholder's rectangle or circle. When the designer has customized the picture's form or scales, the tool will automatically apply a mask to the image.
- In contrast to previous Adobe design tools, Adobe XD's repeat grid functionality eliminates the need for designers to repeatedly copy, paste, or Ctrl+D items. They can choose the group and change the grid's repetition settings.

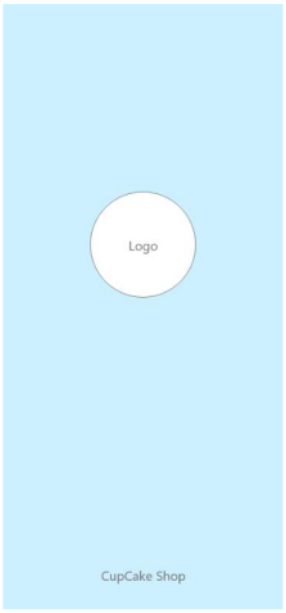
Logo
CupCake Shop

Figure 7


Logo
User Name
Password
LOGIN
Forgot Password
CupCake Shop

Figure 8

Figure 9

Figure 10

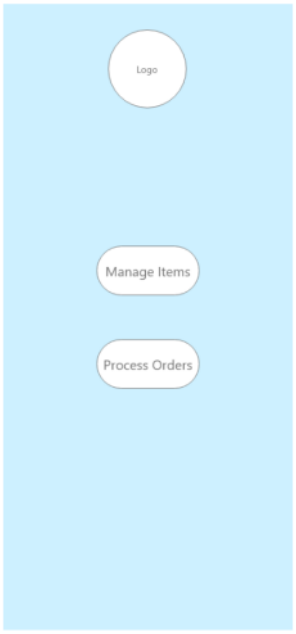

Figure 11

Figure 12

TASK 04

I used android studio as Development IDE. Android Studio is a well-known integrated development environment (IDE) created exclusively for creating Android apps using Java, and it provides several benefits for creating Android apps.

- Strong IDE: Designed specifically for the creation of Android apps, Android Studio offers a comprehensive and feature-rich IDE. It provides a broad range of tools, including as an emulator for testing apps, a visual layout editor for designing user interfaces, a powerful code editor with code completion and debugging capabilities, and a substantial library and template collection for quickly creating Android apps.
- Environment that is friendly to developers: Android Studio was created with developers in mind. To assist developers in creating apps fast and effectively, it provides a user-friendly interface with simple navigation, code refactoring tools, and a plethora of

documentation and resources. Moreover, it supports version control tools like Git, which makes it simple for groups to work together on app development tasks.

- Detailed debugging and testing options are provided by Android Studio to assist developers in finding and fixing problems with their apps. Developers may establish breakpoints, walk through code, and instantly investigate variables thanks to the debugger's potent features. Moreover, it offers a selection of testing tools, including as support for unit testing, integration testing, and UI testing, as well as built-in emulators for testing apps on various virtual devices.
- Powerful libraries and templates: Android Studio include several libraries and templates that may hasten the creation of apps. It consists of the Android SDK, which gives developers a complete collection of resources and APIs for creating Android apps. Developers may also use the enormous ecosystem of third-party libraries and plugins available on the Android Studio platform to add features like networking, UI elements, and database integration to their apps.
- The official IDE for developing Android apps is called Android Studio, and Google, the company that created Android, regularly updates and maintains it. As a result, developers get access to the most recent Android features and APIs as well as continuous updates and changes to the IDE. Moreover, it offers tight connectivity with other Google tools and services, including Firebase and Google Cloud Platform, which can improve the development process even more.
- Support from the community: There is a sizable and vibrant community of developers who work on Android Studio and offer help through forums, blogs, and other online tools. As a result, developers have access to a plethora of tutorials, sample code, and answers to frequently asked issues that may help them overcome obstacles and gain knowledge from the experiences of others.

While implementing, I used SDK 31 version and Orio as Android Version. Software Development Kit, or SDK, is used in Android development. It is a collection of resources, libraries, and tools made available by Google for creating Android apps. The Android SDK contains several elements that are required for designing, developing, testing, and packaging Android apps.

- Improved Performance: Recent SDK releases frequently include optimizations and changes that help Android apps run faster. This may involve better resource management, more effective memory consumption, and quicker UI element rendering, all of which might lead to end users seeing a smoother and quicker app performance.
- New Features and APIs: SDK upgrades frequently provide new features and APIs that developers can use to improve the functionality of their apps. Improved support for more recent technology, access to new sensors or connectivity choices, and increased functionality in fields like multimedia, graphics, and machine learning are just a few examples. Developers may be able to create more complex and feature-rich apps thanks to these new capabilities and APIs.
- Stability and bug repairs: SDK upgrades frequently include stability and bug fixes that take care of problems found in earlier versions. Apps may become more dependable and stable as a consequence, performing better overall and experiencing fewer crashes.
- Improved Security: Newer SDK releases may come with security upgrades to guard against potential threats and vulnerabilities. This can involve greater data processing and storage security safeguards, stronger encryption choices, and improved security APIs. Your program may be constructed on a more secure basis by using the most recent SDK version.

Implementation

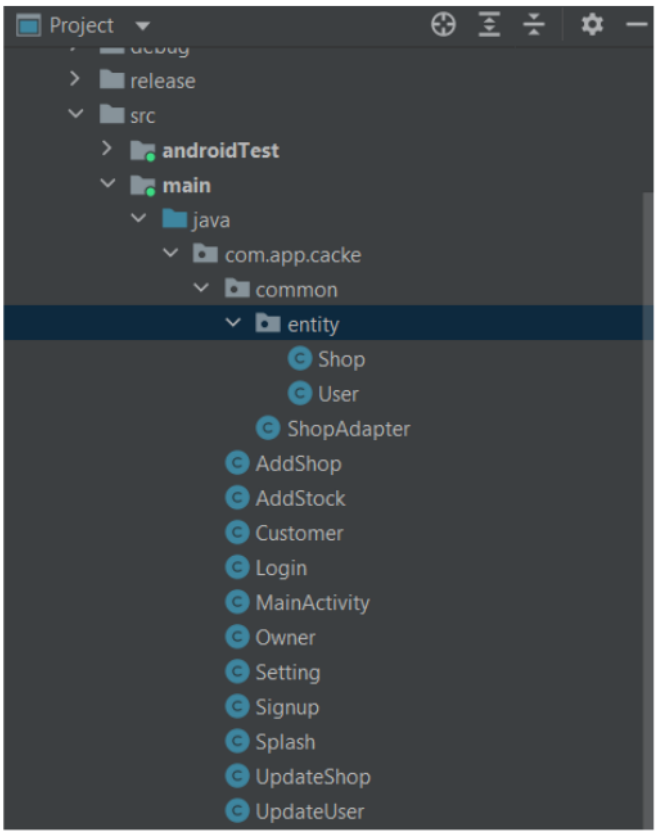

Figure 13

For the project's implementation, I employed a microservices structure. With this structure, the app is divided into small, loosely linked microservices that can be independently created, deployed, and scaled. Each microservice manages a particular aspect of the app's functioning and interacts with the others via APIs. This structure encourages flexibility, fault tolerance, and scalability but may necessitate more administration and infrastructure costs.

```
" import static android.content.ContentValues.TAG;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

import com.google.android.gms.tasks.OnFailureListener;
```

```
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.sdsmdg.tastytoast.TastyToast;

public class Login extends AppCompatActivity {
    private Button sign_in_button, sign_up_button;
    private boolean valid = true;
    TextInputEditText password_field, username_field;
    private FirebaseAuth auth;
    private FirebaseFirestore db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        auth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();

        sign_in_button = findViewById(R.id.sign_in_button);
        sign_up_button = findViewById(R.id.sign_up_button);
        username_field = findViewById(R.id.username_field);
        password_field = findViewById(R.id.password_field);
        sign_up_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(getApplicationContext(),
Signup.class));

            }
        });

        sign_in_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                checkField(username_field);
                checkField(password_field);
                if (valid) {
                    String email = username_field.getText().toString();
                    String pass = password_field.getText().toString();

                    auth.signInWithEmailAndPassword(email,
pass).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                        @Override
                        public void onSuccess(AuthResult authResult) {
                            TastyToast.makeText(
                                    getApplicationContext(),
                                    "Successfully login!",
                                    TastyToast.LENGTH_SHORT,
                                    TastyToast.SUCCESS

                            );
```

```
checkUserAccessLevel(authResult.getUser().getUid());
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(Exception e) {
                            TastyToast.makeText(
                                    getApplicationContext(),
                                    "Login Faller!",
                                    TastyToast.LENGTH_LONG,
                                    TastyToast.ERROR

                            );
                        }
                    });
                } else {
                    // 2. Warning message
                    TastyToast.makeText(
                            getApplicationContext(),
                            "required filed ",
                            TastyToast.LENGTH_LONG,
                            TastyToast.WARNING
                    );
                }
            }
        });

    }

    public boolean checkField(TextInputEditText textField) {
        if (textField.getText().toString().isEmpty()) {
            textField.setError("required filed ");
            valid = false;
        } else {
            valid = true;
        }

        return valid;
    }

    public void checkUserAccessLevel(String uid) {
        DocumentReference documentReference =
db.collection("Users").document(uid);
        documentReference.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
            @Override
            public void onSuccess(DocumentSnapshot documentSnapshot) {
                Log.d(TAG, "onSuccess: " +
documentSnapshot.getBoolean("isPhar"));
                Log.d(TAG, "onSuccess isCus: " +
documentSnapshot.getBoolean("isCus"));
                if (documentSnapshot.getBoolean("isPhar") == true) {

                    startActivity(new Intent(getApplicationContext(),
Owner.class));
                    finish();
                }
                if (documentSnapshot.getBoolean("isCus") == true) {
```

```
                    startActivity(new Intent(getApplicationContext(),
Customer.class));
                    finish();
                }
            }
        });
    }

    @Override
    protected void onStart() {
        super.onStart();
        if (FirebaseAuth.getInstance().getCurrentUser() != null) {
            DocumentReference documentReference =
FirebaseFirestore.getInstance().collection("Users").document(FirebaseAuth.get
Instance().getCurrentUser().getUid());
            documentReference.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
                @Override
                public void onSuccess(DocumentSnapshot documentSnapshot) {
                    if (documentSnapshot.getBoolean("isPhar") == true) {

                        startActivity(new Intent(getApplicationContext(),
Owner.class));
                        finish();
                    }
                    if (documentSnapshot.getBoolean("isCus") == true) {
                        startActivity(new Intent(getApplicationContext(),
Customer.class));
                        finish();
                    }
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(Exception e) {
                    FirebaseAuth.getInstance().signOut();
                    startActivity(new Intent(getApplicationContext(),
Login.class));
                    finish();
                }
            });
        }
    }
}"
```

Figure 14 Login Implementation

I used Fire base as Database and all the authentications will be handling on using Fire base Email base authentication.

```
"package com.app.cacke.common.entity;

import java.io.Serializable;
```

```java
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Data
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class Shop implements Serializable {

    private String id;

    private String name;

    private String mobile;

    private String email;

    private String location;

    private String image;

    private String description;

    private String userId;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getMobile() {
```

```
        return mobile;
    }

    public void setMobile(String mobile) {
        this.mobile = mobile;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

Figure 15 Entity Class Shops

```
package com.app.cacke.common.entity;

import java.io.Serializable;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Data
@Getter
```

```java
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class User implements Serializable {

    private String name;

    private Integer mobile;

    private String email;

    private String password;

    private Boolean isCus;

    private Boolean isPhar;

    private String image;

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getMobile() {
        return mobile;
    }

    public void setMobile(Integer mobile) {
        this.mobile = mobile;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
```

```
        this.password = password;
    }

    public Boolean getCus() {
        return isCus;
    }

    public void setCus(Boolean cus) {
        isCus = cus;
    }

    public Boolean getPhar() {
        return isPhar;
    }

    public void setPhar(Boolean phar) {
        isPhar = phar;
    }
}"
```

Figure 16 Entity Class Users

```
"package com.app.cacke;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.Icon;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.text.TextUtils;
import android.util.Base64;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.ImageButton;

import com.app.cacke.common.entity.User;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
```

```
import com.google.firebase.firestore.FirebaseFirestore;
import com.sdsmdg.tastytoast.TastyToast;

import java.io.ByteArrayOutputStream;

public class UpdateUser extends AppCompatActivity {
    private View topAppBar;

    private TextInputEditText name_field, mobile_field, email_field,
password_field;

    private CheckBox phar;

    private Button update_button;

    private ImageButton imageButton;

    private String image;

    private FirebaseFirestore db;

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    @TargetApi(Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_user);

        db = FirebaseFirestore.getInstance();
        topAppBar = (View) findViewById(R.id.topAppBar);
        name_field = findViewById(R.id.name_field);
        mobile_field = findViewById(R.id.mobile_field);
        phar = findViewById(R.id.phar);
        password_field = findViewById(R.id.password_field);
        email_field = findViewById(R.id.email_field);
        update_button = findViewById(R.id.update_button);
        imageButton = (ImageButton) findViewById(R.id.imageButton);
        if (ContextCompat.checkSelfPermission(UpdateUser.this,
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(UpdateUser.this, new String[]{
                    Manifest.permission.CAMERA
            }, 100);
        }

        imageButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                startActivityForResult(intent, 100);
```

```
            }
        });

        getUserDetail();


        topAppBar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(getApplicationContext(),
Owner.class));
            }
        });
        update_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {


                String name = name_field.getText().toString();
                String phone = mobile_field.getText().toString();
                String email = email_field.getText().toString();
                String pass = password_field.getText().toString();
                Boolean pharm = phar.isChecked();
                String image = getImage();


                if (TextUtils.isEmpty(name) && TextUtils.isEmpty(phone) &&
TextUtils.isEmpty(email) && TextUtils.isEmpty(pass)) {

                    // 2. Warning message
                    TastyToast.makeText(
                            getApplicationContext(),
                            "required filed !",
                            TastyToast.LENGTH_LONG,
                            TastyToast.WARNING
                    );
                } else {

                    update(name, phone, email, pass, pharm, image);
                }
            }

        });
    }

    /**
     * for  event camera activity result.
     *
     * @param requestCode
     * @param resultCode
     * @param data
     */
    @TargetApi(Build.VERSION_CODES.M)
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        if (requestCode == 100) {
            Bitmap bitmap = (Bitmap) data.getExtras().get("data");
            // imageView.setImageBitmap(bitmap);
            setImage(encodeBitmapAndSaveToFirebase(bitmap));
            if (getImage() != null) {

imageButton.setImageIcon(Icon.createWithBitmap(decodeBitmapAndSaveToFirebase(
getImage())));
            }
        }
    }

    /**
     * decode byte image.
     *
     * @param image
     * @return
     */
    public Bitmap decodeBitmapAndSaveToFirebase(String image) {
        //decode base64 string to image
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        byte[] imageBytes = baos.toByteArray();
        imageBytes = Base64.decode(image, Base64.DEFAULT);
        Bitmap decodedImage = BitmapFactory.decodeByteArray(imageBytes, 0,
imageBytes.length);
        return decodedImage;
    }

    /**
     * encode image to base64
     *
     * @param bitmap
     * @return
     */
    public String encodeBitmapAndSaveToFirebase(Bitmap bitmap) {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, baos);
        String imageEncoded = Base64.encodeToString(baos.toByteArray(),
Base64.DEFAULT);
        return imageEncoded;
    }

    /**
     * update user
     *
     * @param name
     * @param phone
     * @param email
     * @param pass
     * @param ph
     * @param image
     */
    private void update(String name, String phone, String email, String pass,
Boolean ph, String image) {
        User users = new User();
        users.setName(name);
        users.setMobile(Integer.parseInt(phone));
```

```
        users.setEmail(email);
        users.setPassword(pass);
        users.setPhar(ph);
        users.setImage(image);

db.collection("Users").document(FirebaseAuth.getInstance().getUid()).set(user
s).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                // on successful completion of this process
                // we are displaying the toast message.
                TastyToast.makeText(
                        getApplicationContext(),
                        "Successfully update!",
                        TastyToast.LENGTH_LONG,
                        TastyToast.SUCCESS

                );
                startActivity(new Intent(getApplicationContext(),
Owner.class));
                finish();
            }
        }).addOnFailureListener(new OnFailureListener() {
            // inside on failure method we are
            // displaying a failure message.
            @Override
            public void onFailure(Exception e) {
                TastyToast.makeText(
                        getApplicationContext(),
                        "Fail to update" + e,
                        TastyToast.LENGTH_LONG,
                        TastyToast.ERROR
                );
            }
        });
    }

    private void getUserDetail() {
        String userId =
(FirebaseAuth.getInstance().getCurrentUser().getUid());
        DocumentReference documentReference =
db.collection("Users").document(userId);
        documentReference.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
            @SuppressLint("ResourceType")
            @TargetApi(Build.VERSION_CODES.M)
            @Override
            public void onSuccess(DocumentSnapshot documentSnapshot) {
                name_field.setText(documentSnapshot.getString("name"));

mobile_field.setText(documentSnapshot.get("mobile").toString());
                email_field.setText(documentSnapshot.getString("email"));

password_field.setText(documentSnapshot.getString("password"));
                phar.setChecked(documentSnapshot.getBoolean("isPhar"));
                if (documentSnapshot.getString("image") != null) {
```

```
imageButton.setImageIcon(Icon.createWithBitmap(decodeBitmapAndSaveToFirebase(
documentSnapshot.getString("image"))));
                    setImage(documentSnapshot.getString("image"));
                }

            }

        });
    }
}"
```

Figure 17 Update User

```
"package com.app.cacke;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;


public class Splash extends AppCompatActivity {
    Handler handler;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        handler=new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent=new
Intent(getApplicationContext(),Login.class);
                startActivity(intent);
                finish();
            }
        },2000);

    }
}"
```

Figure 18 Splash Screen

Splash is the main method which runs at the startup.

```
"package com.app.cacke;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.annotation.TargetApi;
```

```java
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.Icon;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Base64;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.ImageButton;

import com.app.cacke.common.entity.User;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.sdsmdg.tastytoast.TastyToast;

import java.io.ByteArrayOutputStream;

import lombok.NonNull;

public class Signup extends AppCompatActivity {
    private View topAppBar;

    private TextInputEditText name_field, mobile_field, email_field,
password_field;

    private CheckBox phar;

    private Boolean cus = true;

    private Button save button;

    boolean valid = true;

    private FirebaseAuth auth;

    private ImageButton imageButton;

    private FirebaseFirestore db;

    private String image;

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
```

```
        this.image = image;
    }

    public Boolean getCus() {
        return cus;
    }

    public void setCus(Boolean cus) {
        this.cus = cus;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //getSupportActionBar().hide();
        setContentView(R.layout.activity_signup);

        auth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();

        topAppBar = (View) findViewById(R.id.topAppBar);
        name_field = findViewById(R.id.name_field);
        mobile_field = findViewById(R.id.mobile_field);
        email_field = findViewById(R.id.email_field);
        password_field = findViewById(R.id.password_field);
        phar = findViewById(R.id.phar);
        save_button = findViewById(R.id.save_button);

        imageButton = (ImageButton) findViewById(R.id.imageButton);
        if (ContextCompat.checkSelfPermission(Signup.this,
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(Signup.this, new String[]{
                    Manifest.permission.CAMERA
            }, 100);
        }

        imageButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                startActivityForResult(intent, 100);
            }
        });

        topAppBar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(getApplicationContext(),
Login.class));
                // startActivity(new Intent(getApplicationContext(),
MainActivity.class));
            }
        });

        phar.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
```

```
            @Override
            public void onCheckedChanged(CompoundButton compoundButton,
boolean b) {
                if (compoundButton.isChecked()) {
                    setCus(false);
                }
            }
        });
        save_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String name = name_field.getText().toString();
                String phone = mobile_field.getText().toString();
                String email = email_field.getText().toString();
                String pass = password_field.getText().toString();
                String images = "";
                if (getImage() != null) {
                    image = getImage();
                }
                boolean isCus = getCus();
                boolean isPhar = phar.isChecked();
                checkField(name_field);
                checkField(mobile_field);
                checkField(email_field);
                checkField(password_field);


                if (valid) {
      7             auth.createUserWithEmailAndPassword(email,
pass).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                        @Override
                        public void onSuccess(AuthResult authResult) {
                            FirebaseUser user = auth.getCurrentUser();
                            TastyToast.makeText(
                                    getApplicationContext(),
                                    "Your add has been Successfully added!",
                                    TastyToast.LENGTH_SHORT,
                                    TastyToast.SUCCESS

                            );
                            addDataToFirestore(name, phone, email, pass,
isCus, isPhar, images, user);
                        3}
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(Exception e) {
                            TastyToast.makeText(
                                    getApplicationContext(),
                                    "User  has been Create Not Successfully
",
                                    TastyToast.LENGTH_SHORT,
                                    TastyToast.ERROR

                            );

                        }
```

```
                    });


                } else {

                    // 2. Warning message
                    TastyToast.makeText(
                            getApplicationContext(),
                            "required filed !",
                            TastyToast.LENGTH_LONG,
                            TastyToast.WARNING
                    );
                }
            }
        });


    }

    /**
     * for  event camera activity result.
     *
     * @param requestCode
     * @param resultCode
     * @param data
     */
    @TargetApi(Build.VERSION_CODES.M)
    @Override 5
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 100) {
            Bitmap bitmap = (Bitmap) data.getExtras().get("data");
            // imageView.setImageBitmap(bitmap);
            setImage(encodeBitmapAndSaveToFirebase(bitmap));
            if (getImage() != null) {

imageButton.setImageIcon(Icon.createWithBitmap(decodeBitmapAndSaveToFirebase(
getImage())));
            }
        }
    }
    3
    public boolean checkField(TextInputEditText textField) {
        if (textField.getText().toString().isEmpty()) {
            textField.setError("required filed");
            valid = false;
        } else {
            valid = true;
        }

        return valid;
    }

    private void addDataToFirestore(String name, String phone, String email,
String pass, Boolean isCus, Boolean isPhar, String image, FirebaseUser users)
{
```

```
        // creating a collection reference
        // for our Firebase Firetore database.
        DocumentReference dbRef =
db.collection("Users").document(users.getUid());

        // adding our data to our courses object class.
        User user = new User();
        user.setName(name);
        user.setMobile(Integer.parseInt(phone));
        user.setEmail(email);
        user.setPassword(pass);
        user.setCus(isCus);
        user.setPhar(isPhar);
        user.setImage(getImage());
        // below method is use to add data to Firebase Firestore.
        dbRef.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                startActivity(new Intent(getApplicationContext(),
Login.class));
                finish();
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // this method is called when the data addition process is
failed.
                // displaying a toast message when data addition is failed.
                TastyToast.makeText(
                        getApplicationContext(),
                        "Fail to add" + e,
                        TastyToast.LENGTH_LONG,
                        TastyToast.ERROR
                );
            }
        });
    }

    /**
     * decode byte image.
     *
     * @param image
     * @return
     */
    public Bitmap decodeBitmapAndSaveToFirebase(String image) {
        // decode base64 string to image
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        byte[] imageBytes = baos.toByteArray();
        imageBytes = Base64.decode(image, Base64.DEFAULT);
        Bitmap decodedImage = BitmapFactory.decodeByteArray(imageBytes, 0,
imageBytes.length);
        return decodedImage;
    }

    /**
     * encode image to base64
```

```
     *
     * @param bitmap
     * @return
     */
    public String encodeBitmapAndSaveToFirebase(Bitmap bitmap) {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, baos);
        String imageEncoded = Base64.encodeToString(baos.toByteArray(),
Base64.DEFAULT);
        return imageEncoded;
    }

}"
```

Figure 19 Signup Code

```
"public class MainActivity extends AppCompatActivity {
    DrawerLayout drawerLayout;
    NavigationView navigationView;
    Toolbar toolbar;
    Menu menu;
    TextView textView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);



    }
}"
```

This is the main Class. For this application I have used Drawer Layout. As development Layout, ,A common UI element in Android development, Drawer Layout offers a sliding panel for navigation and is frequently used in conjunction with a toolbar or action bar. The following are some benefits of adopting Drawer Layout while developing for Android,

- Simple Navigation: Drawer Layout enables you to add a navigation panel that is quickly accessed by swiping from the left edge of the screen or by touching on a toolbar icon. This offers customers a simple and intuitive navigation experience, making it simple for them to reach various parts or features of your software.
- Saving screen real estate: Drawer Layout gives you the option to conceal the navigation panel when it's not required. This allows you to leave the primary content area uncluttered while yet granting access to navigation choices when necessary, making it particularly handy in apps with complicated navigation systems or a lot of information.

- Drawer Layout is quite flexible, so you can create your own navigation panel with the layout, styling, and animations of your choice. The width, background color, and transparency of the drawer may all be changed to suit your preferences. Also, you may alter the animations used to open and close the drawer, giving your app's users a distinctive and customized experience.
- Drawer Layout complies with the Material Design accessibility principles and is accessible. Making your software accessible to people with impairments or who depend on assistive technology is as simple as adding accessibility labels and content descriptions to the navigation panel elements.
- Flexibility: Drawer Layout may be used to a number of different design patterns, including the hamburger menu, side navigation, and multi-level navigation. It offers freedom in constructing and organizing the app's navigation choices in accordance with your unique needs.
- Drawer Layout is compatible with previous Android versions since it is included in the Android Support Library, which is accessible from any Android device. This enables you to ensure that your app functions properly on a variety of devices by offering a consistent navigation experience across various Android devices and versions.

```
• "package com.app.cacke;

  import androidx.appcompat.app.AppCompatActivity;
  import androidx.recyclerview.widget.LinearLayoutManager;
  import androidx.recyclerview.widget.RecyclerView;

  import android.content.Intent;
  import android.os.Bundle;
  import android.view.View;

  import com.app.cacke.common.ShopAdapter;
  import com.app.cacke.common.entity.Shop;
  import
  com.google.android.material.floatingactionbutton.FloatingActionButton;
  import com.google.firebase.firestore.FirebaseFirestore;

  import java.util.ArrayList;

  public class AddStock extends AppCompatActivity {
      private View topAppBar;
      private FloatingActionButton float button;
      private RecyclerView view;
      private ArrayList<Shop> ArrayList;
      private ShopAdapter adapter;
```

```
        private FirebaseFirestore db;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
          //  getSupportActionBar().hide();
            setContentView(R.layout.activity_add_stock);
            view = findViewById(R.id.rcView);
            view.setHasFixedSize(true);
            view.setLayoutManager(new LinearLayoutManager(this));

            db = FirebaseFirestore.getInstance();
            ArrayList = new ArrayList<Shop>();
            adapter = new ShopAdapter(AddStock.this, ArrayList);

            topAppBar = findViewById(R.id.topAppBar);
            float_button = findViewById(R.id.float_button);

            float_button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    startActivity(new Intent(getApplicationContext(),
    AddShop.class));
                }
            });
            topAppBar.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    startActivity(new Intent(getApplicationContext(),
    Owner.class));
                    finish();
                }
            });
            view.setAdapter(adapter);
        }
    }"
```

Figure 20 Add Stock

```
"package com.app.cacke.common;



import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.util.Base64;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
```

```java
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;


import com.app.cacke.AddStock;
import com.app.cacke.R;
import com.app.cacke.UpdateShop;
import com.google.android.material.dialog.MaterialAlertDialogBuilder;

import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;

import com.app.cacke.Owner;
import com.app.cacke.common.entity.Shop;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.FirebaseFirestore;
import com.sdsmdg.tastytoast.TastyToast;

import java.io.ByteArrayOutputStream;
import java.util.ArrayList;

public class ShopAdapter extends
RecyclerView.Adapter<ShopAdapter.MyViewHolder> {
    private Context context;
    private ArrayList<Shop> shopArrayList;

    public ShopAdapter(Context context, ArrayList<Shop> shopArrayList) {
        this.context = context;
        this.shopArrayList = shopArrayList;
    }

    public void filterList(ArrayList<Shop> filterlist) {
        // below line is to add our filtered
        // list in our course array list.
        shopArrayList = filterlist;
        // below line is to notify our adapter
        // as change in recycler view data.
        notifyDataSetChanged();
    }

    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(context).inflate(R.layout.hader, parent,
false);
        return new MyViewHolder(v);
    }

    @Override
    public void onBindViewHolder(MyViewHolder holder, int position) {
        Shop shop = shopArrayList.get(position);
        holder.name.setText(shop.getName());
        holder.dec.setText(shop.getDescription());
        holder.mobile.setText(shop.getMobile());
```

```
        if(shop.getImage() !=null){
            Bitmap bitmap= decodeBitmapAndSaveToFirebase(shop.getImage());
            int maxHeight = 600;
            int maxWidth = 300;
            //float scale = Math.min(((float)maxHeight / bitmap.getWidth()),
((float)maxWidth / bitmap.getHeight()));

            Matrix matrix = new Matrix();
         //   matrix.postScale(scale, scale);

//            bitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(),
bitmap.getHeight(), matrix, true);
          if(bitmap != null) {
              Log.d("Image", bitmap.toString());
              holder.logo.setImageBitmap(bitmap);
          }
        }


    }
    public Bitmap  decodeBitmapAndSaveToFirebase(String image) {
        //decode base64 string to image
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        byte[] imageBytes = baos.toByteArray();
        imageBytes = Base64.decode(image, Base64.DEFAULT);
        Bitmap decodedImage = BitmapFactory.decodeByteArray(imageBytes, 0,
imageBytes.length);
        return decodedImage;
    }

    @Override
    public int getItemCount() {
        return shopArrayList.size();
    }

    public class MyViewHolder extends RecyclerView.ViewHolder {
        private Button delete_button, update_button;
        private TextView name, dec, mobile;
        private FirebaseFirestore db;
        private ImageView logo;

        public MyViewHolder(View itemView) {
            super(itemView);
            db = FirebaseFirestore.getInstance();
            name = (TextView) itemView.findViewById(R.id.name);
            dec = (TextView) itemView.findViewById(R.id.des);
            mobile = (TextView) itemView.findViewById(R.id.mobile);
            delete_button = (Button)
itemView.findViewById(R.id.delete_button);
            update_button = (Button)
itemView.findViewById(R.id.update_button);
            logo = (ImageView) itemView.findViewById(R.id.logo);
;

            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
```

```
                    Shop shop = shopArrayList.get(getAdapterPosition());
                    // below line is creating a new intent.
                    Intent i = new Intent(context, AddStock.class);
                    i.putExtra("shop", shop);
                    context.startActivity(i);
                }
            });
            delete_button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    //  owner= new Owner().deleteCourse();
                    Shop shop = shopArrayList.get(getAdapterPosition());
                    AlertDialog dialog = new
MaterialAlertDialogBuilder(context,
                            R.style.MaterialAlertDialog_App_Title_Text)
                            .setMessage("Are You Sure?")
                            .setPositiveButton("yes", new
DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface
dialogInterface, int i) {
                                    deleteCourse(shop);
                                }
                            })
                            .setNegativeButton("No", null)
                            .show();
                    dialog.getButton(Dialog.BUTTON_POSITIVE).setTextSize(18);
                    dialog.getButton(Dialog.BUTTON_NEGATIVE).setTextSize(18);

                    TextView textView =
dialog.findViewById(android.R.id.message);
                    if (textView != null) {
                        textView.setTextSize(18);
                    }
                    TextView textViewTitle =
dialog.findViewById(android.R.id.title);
                    if (textViewTitle != null) {
                        textViewTitle.setTextSize(18);
                    }
                }
            });
            update_button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Shop shop = shopArrayList.get(getAdapterPosition());
                    // below line is creating a new intent.
                    Intent i = new Intent(context, UpdateShop.class);
                    i.putExtra("shop", shop);
                    context.startActivity(i);

                }
            });
        }

        public void deleteCourse(Shop shop) {
            // below line is for getting the collection
            // where we are storing our courses.
```

```
            String ids = (shop.getId());
            db.collection("shop").
                    // after that we are getting the document
                    // which we have to delete.
                            document(ids).

                    // after passing the document id we are calling
                    // delete method to delete this document.
                            delete().
                    // after deleting call on complete listener
                    // method to delete this data.
                            addOnCompleteListener(new
OnCompleteListener<Void>() {
                                2
                        @Override
                        public void onComplete(Task<Void> task) {
                            // inside on complete method we are checking
                            // if the task is success or not.
                            if (task.isSuccessful()) {
                                TastyToast.makeText(
                                        context,
                                        "Your shop has been Successfully
delete!",
                                        TastyToast.LENGTH_SHORT,
                                        TastyToast.SUCCESS

                                );
                                14
                                Intent i = new Intent(context, Owner.class);
                                context.startActivity(i);
                            } else {
                                TastyToast.makeText(
                                        context,
                                        "Fail to delete shop",
                                        TastyToast.LENGTH_LONG,
                                        TastyToast.ERROR
                                );
                            }
                        }
                    });
        }
    }
}"
```

Figure 21 Shop Adapter

TASK 05

Plan and documentation for tests

Objectives

- Purpose

The testing procedure used in the creation of the Cup Cake Android application is described in this paper.

- Scope

This test plan addresses the black box test, which was conducted following the integration of all application unit components. Yet even though the unit tests were performed at each stage of the unit complement, they weren't recorded. To cut down on time wasted during the development process, the full application test (also known as a "black box test") is performed.

I conducted the testing; therefore, I included a feedback feature to the program to gauge user happiness. This information will be utilized to improve the product in the future.

The main goal of the testing process is to ensure that the output is accurate given the application's inputs. Here in cupcake, the tester inserts the appropriate data types and evaluates how the system responds when a user presses a certain button.

Test Cases

- Test case 01 – Splash Screen

Test Process	Expected Output	Actual Output
a. The interface has been designed with a	Splash screen should appear for 3 seconds	Test successful same as the expected output.

logo and colorful background. b. After that run the application on an emulator or in an android mobile connecting through a USB cable.	with the logo of the shop.	

Figure 07 Splash Screen

- Test case 2 Login UI:

Test Process	Expected Output	Actual Output
a. Open the application. b. Provide a valid email and password, c. Then click login.	The app redirect to Home UI.	Same as expected.

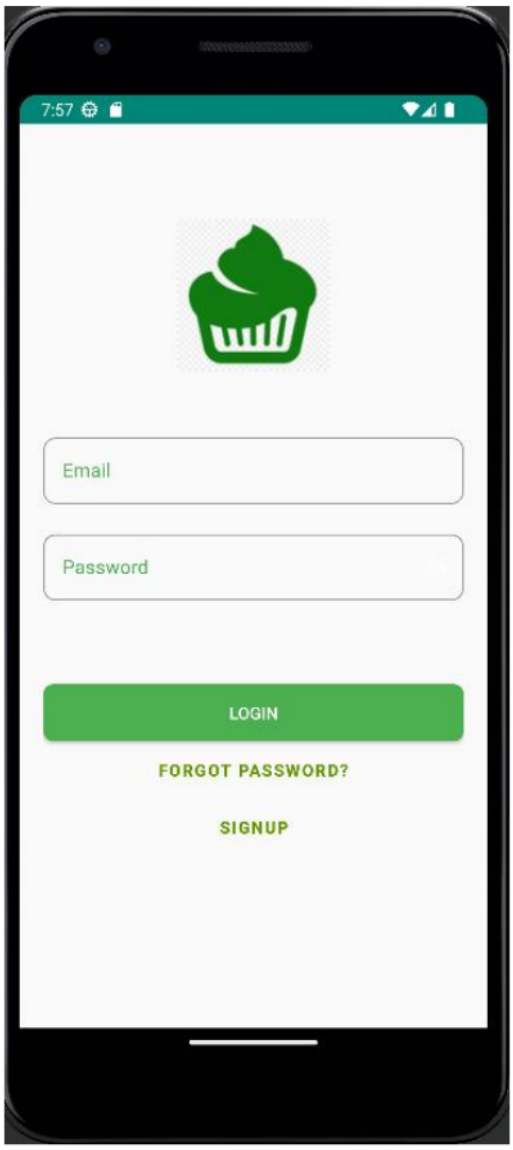

Figure 08 Login UI

Figure 09 Home UI

- Test case 3 Sign Up UI:

Test Process	Expected Output	Actual Output
a. Open the Cupcake application. b. Navigate to signup page by clicking on "new user" option on Login interface. c. Provide user details and click Sign Up button.	Customer details should be saved on the database.	Same as expected.

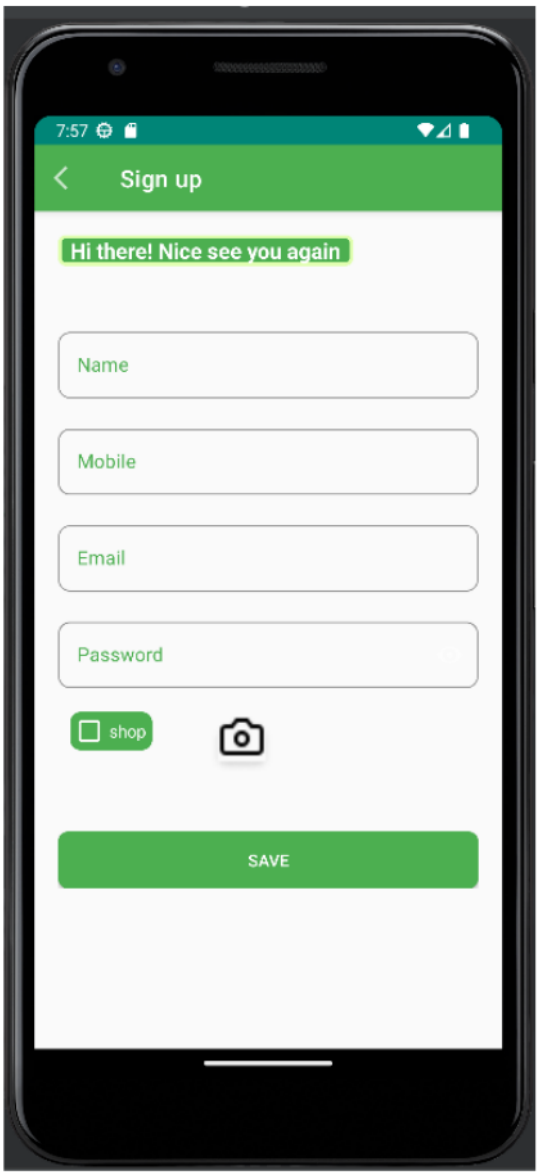

Figure 10 Signup UI

- Test case 4 Place Order

Test Process/ Scenario	Expected Output	Actual Output
a. Open the Cupcake application. b. Choose a product/ products. c. Clicks add to cart button. d. View the cart. e. Click on checkout button. f. And then click place order button on the appearing window.	Order must be placed and the confirmation message should be appear to the user	Same as expected.

TASK 06

User Manual

General information

The overview of the cupcake application and the user manual components are explained in this specific section.

Application overview

The Cu Cake business, which is in Colombo, uses the Cupcake application as an e-commerce tool to streamline its operations. This application was specifically made to sell cupcakes as the main product.

Organization of the Manual

1. General Information
2. System Summary
3. Download Instructions.
4. User privileges (Admin, Customer)
5. How to use the application.

- System Summary
  - Hardware and Software Requirements
    - An Android smartphone is required to use this application.
    - The minimum version of Android that will be supported by this application is 5.0 and above.
    - Also, to download and use the capabilities, the mobile device needs an internet connection.
  - User Access Levels
    - There are two different sorts of users who interact with this app: Admins and Registered Customers.
      - Admin: The person who accesses the program for management reasons is known as the admin. The administration oversees CRUD operations, report generation, and order processing.
      - Registered Customer: This individual signed up for the cupcake application by utilizing the sign-up form. A customer may use the program to browse and purchase Cupcakes from the retailer.

User privileges (Admin, Customer)

- Admin:

    a. View products
    b. Manage products.
    c. Manage Users
    d. Order processing

- Customer:

    a. View products
    b. Place orders
    c. View orders.
    d. User registration and login.

References

Lucidchart.com. (2019). [online] Available at: https://www.lucidchart.com/pages/uml-use-case-diagram [Accessed 7 Sep. 2019].

Ceta, N. (2019). *All You Need to Know About UML Diagrams: Types and 5+ Examples*. [online] Tallyfy. Available at: https://tallyfy.com/uml-diagram/ [Accessed 7 Sep. 2019].

dummies. (2019). *How to Create Use Case Description for Your Business Analysis Report - dummies*. [online] Available at: https://www.dummies.com/business/business-strategy/how-to-create-use-case-description-for-your-business-analysis-report/ [Accessed 7 Sep. 2019].

Medium. (2019). *Principles of Adobe XD*. [online] Available at: https://medium.com/thinking-design/principles-of-adobe-xd-914dea71fa4a [Accessed 7 Sep. 2019].

Medium. (2019). *Principles of Adobe XD*. [online] Available at: https://medium.com/thinking-design/principles-of-adobe-xd-914dea71fa4a [Accessed 7 Sep. 2019].

Guru99.com. (2019). [online] Available at: https://www.guru99.com/application-testing.html [Accessed 7 Sep. 2019].

kanna, v. (2019). *Application Testing - Application Testing Tools and Methodologies*. [online] Softwaretestinghelp.com. Available at: https://www.softwaretestinghelp.com/application-testing-into-the-basics-of-software-testing/ [Accessed 7 Sep. 2019].

test

ORIGINALITY REPORT

10%	4%	2%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Wales Institute, Cardiff Student Paper	5%
2	www.geeksforgeeks.org Internet Source	1%
3	Submitted to University of Ulster Student Paper	<1%
4	Submitted to Middle East College of Information Technology Student Paper	<1%
5	Submitted to Oxford Brookes University Student Paper	<1%
6	Submitted to Swinburne University of Technology Student Paper	<1%
7	Submitted to Chester College of Higher Education Student Paper	<1%
8	Submitted to HELP UNIVERSITY Student Paper	<1%

9	Submitted to University of Bedfordshire Student Paper	<1%
10	stackoverflow.com Internet Source	<1%
11	mkir.com Internet Source	<1%
12	Submitted to Manipal University Student Paper	<1%
13	Submitted to University Politehnica of Bucharest Student Paper	<1%
14	Submitted to University of West London Student Paper	<1%
15	dspace.christcollegeijk.edu.in:8080 Internet Source	<1%
16	labs.utdallas.edu Internet Source	<1%
17	Submitted to Sheffield Hallam University Student Paper	<1%
18	rogersandhollands.com Internet Source	<1%
19	Jeff Tang. "Beginning Google Glass Development", Springer Science and Business Media LLC, 2014 Publication	<1%

20	Submitted to University of Greenwich Student Paper	<1%
21	jmsnew.iobmresearch.com Internet Source	<1%
22	www.coursehero.com Internet Source	<1%
23	Dave Smith, Erik Hellman. "Android Recipes", Springer Science and Business Media LLC, 2016 Publication	<1%
24	Iuliana Cosmina. "Pivotal Certified Professional Spring Developer Exam", Springer Science and Business Media LLC, 2017 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		