Smart Farming System

Muhammad Hasan **(BSE163046)**

Hassan Javed Alam **(BSE163029)**

Fall-2020

Supervised By

Mr. Hamza Bin Waheed

Department of Software Engineering
Capital University of Science & Technology, Islamabad

Submission Form for Final-Year

PROJECT REPORT

Version	V 2.0	NUMBER OF MEMBERS	2

TITLE	Smart Farming System

SUPERVISOR NAME	Mr. Hamza Bin Waheed

MEMBER NAME	REG. NO.	EMAIL ADDRESS
Muhammad Hasan	BSE163046	hassanabubakar00@gmail.com
Hassan Javed Alam	BSE163029	hassanjavedalam@gmail.com

MEMBERS' SIGNATURES

_____

_____

_____

Supervisor's Signature

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Smart farming is an idea to automate the agriculture tasks to enhance the quality of products. Smart Farming is a mobile based application. This chapter describes the scope and specifications of this project. Suitable tools and technologies are also listed in this chapter. A brief study of comparison with the existing solutions is also provided. This chapter is concluded with the work break down structure and Gantt chart, providing the division of work and proposed estimated time and flow of the project.

1.1. Project Introduction

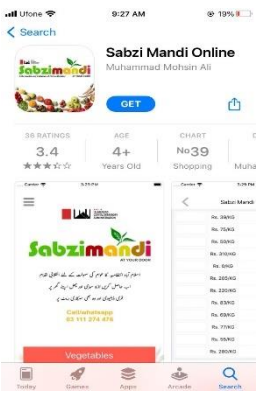
Smart Farming is an automated agriculture system which serves as a monitoring agent to the agriculture field. It measures the moisture level of the soil, monitor the temperature and humidity of the field. The Smart Farming system will provide an e-commerce community platform where Customer, distributors, and client are connected to offer yields/services/purchasing. In control shed environment we make two corps i.e., Potato and Tomato. The optimal temperature of potato is 16 to 20 and the moisture is 400 to 500. And the optimal temperature of Tomato is 21 to 32 and the moisture is 400 to 600.

1.2. Existing Examples / Solutions

Agro sense is the existing system example which provides a device to manages and monitors the activities of agriculture field. Some of the examples which agro sense does have in the device includes automated irrigation system, Nutrient management etc.

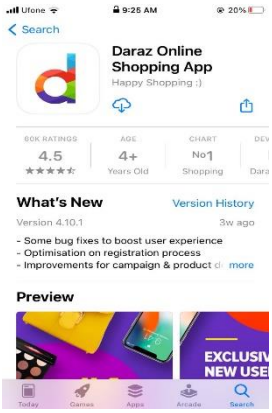
Sabzi Mandi Online:

Launched in 2019 Sabzi mandi is online vegetable shopping and designation of choice. It is an online ordering application in overall Islamabad/ Rawalpindi. This application provides free delivery overall.



Daraz:

Launched in 2012, Daraz is South Asia's online shopping and selling destination of choice. Pakistan, Bangladesh, Sri Lanka and Nepal. Daraz is the pioneer of ecommerce in South Asia offers a diverse assortment of products in categories ranging from customer household and many other items this application provides multiple payment method like cash on delivery and online payment. Daraz is owned by Alibaba Group Holding Limited.



Pakistan Mart:

Launched in 2018, it is Pakistani online shopping destination of choice. Pakistan Mart is an online ordering application in overall Lahore, Islamabad, Faisalabad, Rawalpindi, Sialkot, Karachi, Gujranwala & Sargodha. This application provides delivery overall.

Grocer App:

Launched in 2016, it is Pakistani online shopping Application. Grocer App is an online ordering application overall in Pakistan. This application provides delivery at your doorstep.

Comparison between different applications

We have discussed nine application comparing with our application. All the application gives you the products and also delivery to your doorstep but none of the application makes its own farm where products will grow and then sell it, but we make it.

Application Name	Cash on delivery	Products of Vegetable/fruit	Whole-Sale Dealing	Make own System/ Farm/Shop	Make Own Crops	Monito Farm fields
Sabzi Mandi Online	✓	✓				
Daraz	✓					
Agro Sense						✓
Pakistan Mart	✓	✓				
GrocerApp	✓	✓				
Sabziday Online System	✓	✓				
Sabzi Mandi Plus	✓	✓				
Online Fruit Sabzi	✓	✓				
Smart Farming System	✓	✓	✓	✓	✓	✓

Table 1.1 Existing Examples

1.3. Business Scope

Agriculture has a wide business scope in the world of business. Smart farming system manages the activities of the field such as monitoring the moisture level of the soil etc. to control the growth of plants so that the plants can produce agriculture products. The production of the agriculture products within the maintained environment will lead to have the best quality in the market. An e-commerce community platform where Customer, Distributors and client are connected to offer yields/services/purchasing.

1.4. Useful Tools and Technologies

As we are developing android application modules and also web-based applications so different tools and technologies will be used that are mentioned below.

Android Studio:

It is an integrated development environment for developing android applications. It is based on IntelliJ IDEA. So android applications will be developed in android studio.

Firebase Real-time Database:

It's a NoSQL cloud database where data is synced across all the clients in real-time and remains available too when app goes offline. We will use this database to store and sync data in android apps.

Different sensors and software used in our systems are the following:

1. Soil moisture sensor
2. Temperature sensor
3. Cooling fan
4. Node Mcu
5. Humidity sensor
6. Relay
7. Dimmer

1.5. Project Work Break Down

A Gantt chart outlines what aspects of the project will be completed and by when.

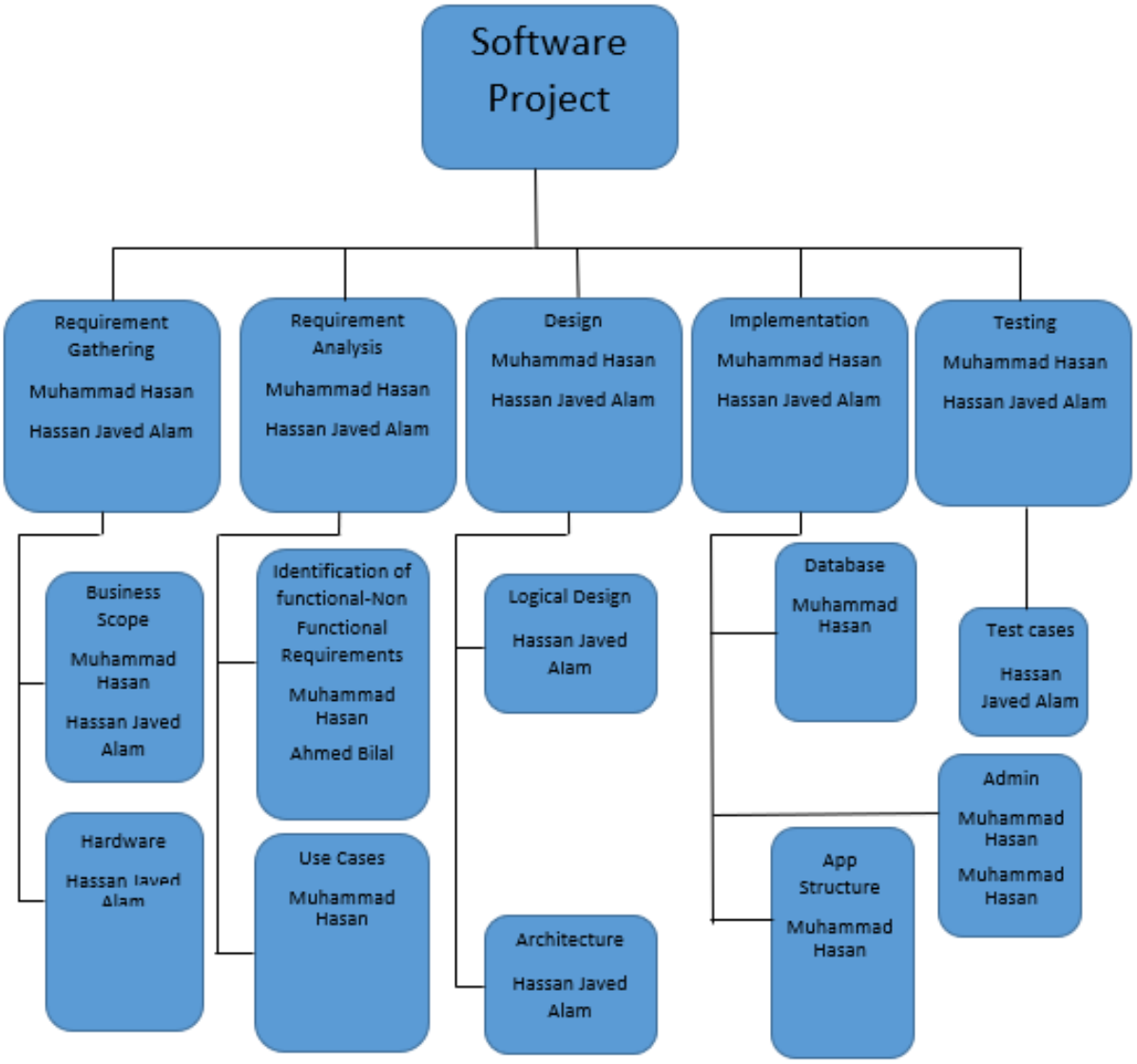


Figure 1.1: Sample Gant Chart

1.6. Project Time line

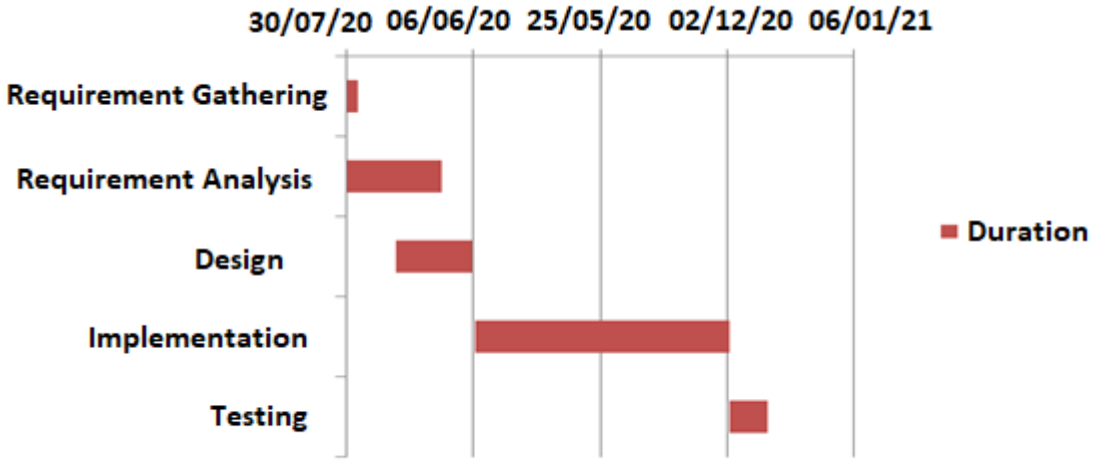


Figure 1.2: Project Timeline

Chapter 2

Requirement Specification and Analysis

2.1. Functional Requirements

Ref No	Description	Type
1.	Admin will sign up and sign in into the system.	Core
2.	Admin will manage products.	Core
2.1	Admin will manage products by adding items.	Core
2.2	Admin will manage products by viewing.	Core
2.2.1	Admin will manage products by deleting.	Core
3.	Admin will manage orders.	Core
3.1	Admin will manage orders by accepting.	Core
3.2	Admin will manage orders rejecting.	Core
4.	Admin can view farm status by viewing temperature, light, air, humidity values.	Core
4.1	Admin will view temperature values.	Core
4.2.	Admin will view fan status.	Core
4.3.	Admin will view light status. by viewing.	Core
5.4.	Admin will view humidity values.	Core
5.	Customer will sign up and sign in into the system.	Core
6.	Customer will show products.	Core
6.1.	Customer will view products.	Core
6.2.	Customer will add products into cart.	Core
7.	Customer will order products.	Core

7.1.	Customer will view order products.	Core
7.2.	Customer will cancel order products.	Core

Table 2.1: Functional Requirements

2.2. Selected Functional Requirements

Ref No	Description	Type
1.	Admin will sign up and sign in into the system.	Core
2.	Admin, who can manage products.	Core
2.1	Admin, who can manage products by adding items.	Core
2.2	Admin will manage products by viewing.	Core
2.2.1	Admin will manage products by deleting.	Core
3.	Admin will manage orders.	Core
3.1	Admin will manage orders by accepting.	Core
3.2	Admin will manage orders rejecting.	Core
4.	Customer will sign in and sign in into the system.	Core
5.	Customer will show products.	Core
5.1.	Customer will view products.	Core
5.2.	Customer will add products into cart.	Core
6.	Customer will order products.	Core

8.1.	Customer will view order products.	Core
8.2.	Customer will cancel order products.	Core

Table 2.2: Selected Functional Requirement

2.3. Non- Functional Requirement

Ref No	Description	Type
1.	The Throughput cycle between any operation should be consistent.	Core
2.	The system should be easy to maintain.	Core
3.	The application should be friendly and usability should be accurate.	Core
4.	The application should not crash, it should be reliable.	Core

Table 2.3: Non- Functional Requirement

2.4. System Use Case Modeling

Use Case diagrams are created to show the action of events among user and the system.
Use case: E-commerce Management

This use case is belonging to User side E-commerce management and shows all the activity.

Figure 2.1: Sample Use case Diagram

This use case is belonging to Admin side E-commerce management and shows all the activity.

Figure 2.2: Sample Use case Diagram

Use case: Controlled shed environment

This use case is belonging to Admin side Farm management and shows all the sub activities.

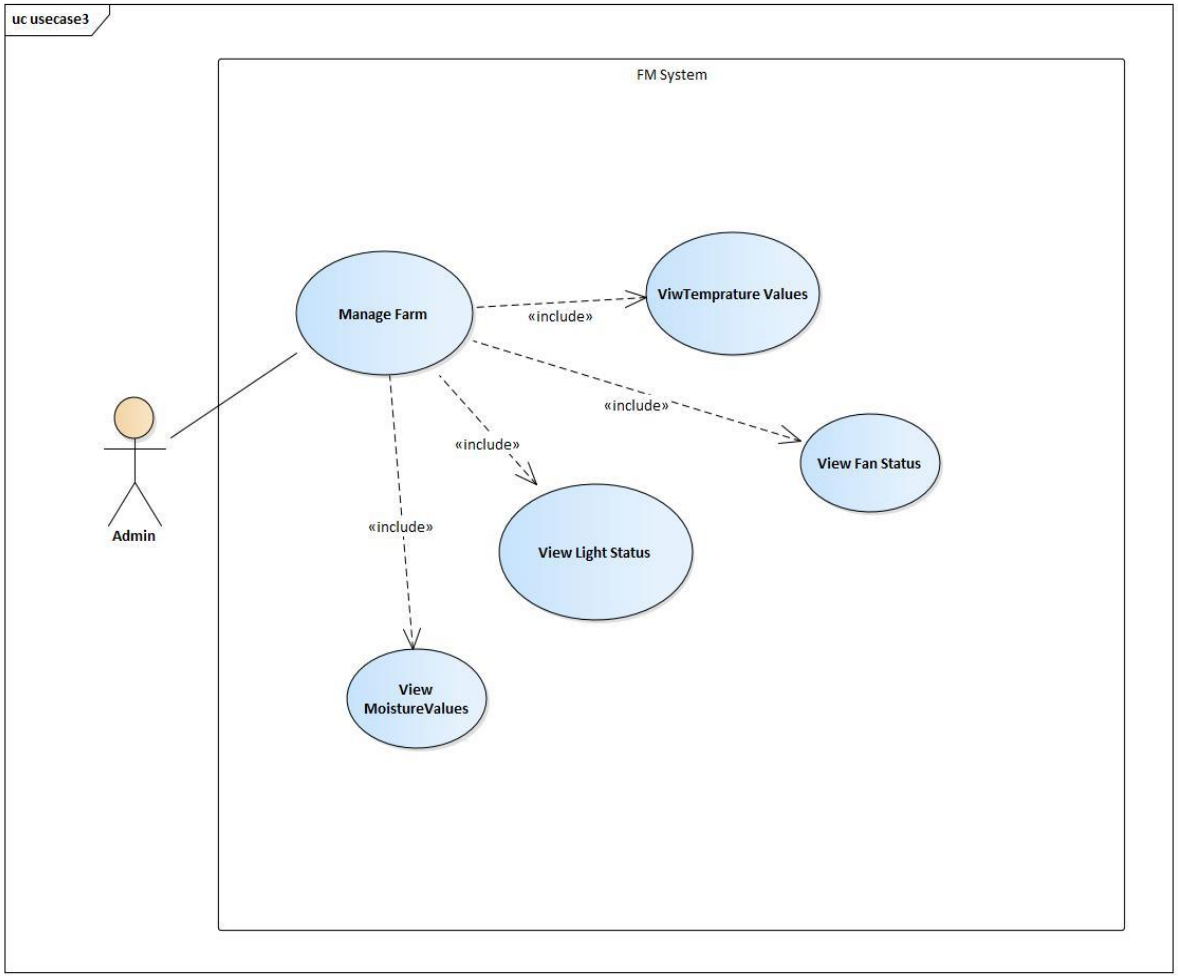


Figure 2.3: Sample Use case Diagram

Use case description: E-commerce management

User description gives details information about use case between user and system.

Use Case 1 Title: Sign Up (for Customers)

This use case description shows how to sign up for customers into the system.

<table>
<tr><td>Use Case ID:</td><td colspan="3">FM1</td></tr>
<tr><td>Use Case Name:</td><td colspan="3">Sign Up</td></tr>
<tr><td>Created By:</td><td>Hasan</td><td>Last Updated By:</td><td>Hasan</td></tr>
<tr><td>Date Created:</td><td>03-10-2020</td><td>Last Revision Date:</td><td>06-11-2020</td></tr>
<tr><td>Actors:</td><td colspan="3">Customer</td></tr>
<tr><td>Description:</td><td colspan="3">This use case begins when the Customer wants to register in the Smart FM Android application system. Customer will enter the required fields such as first name, last, name, email and password.</td></tr>
<tr><td>Trigger:</td><td colspan="3">Sign Up form will be opened.</td></tr>
<tr><td>Preconditions:</td><td colspan="3">Customer provides fields into registration form i.e. Name, last name, email, phone no, and click on signup button for creating account into the system.</td></tr>
<tr><td>Post conditions:</td><td colspan="3">Customer will be register into the system and then he will able to use the system.</td></tr>
<tr><td rowspan="2">Normal Flow:</td><td colspan="2">Customer</td><td>System</td></tr>
<tr><td colspan="2">1. Customer will get register by providing required data by the system.</td><td>2. System will be registered customer successfully into the database.</td></tr>
<tr><td>Alternative Flows:</td><td colspan="3">1. Please fill out the fields
2. Customer cancel the sign-up request</td></tr>
<tr><td>Exceptions:</td><td colspan="3">• Firebase is not responding.
• Internet connection is not available</td></tr>
</table>

Table 2.4: Use case Description

Use Case 2 Title: Sign In (for Customers)

This use case description shows how to sign in for customers into the system.

Use Case ID:	FM1		
Use Case Name:	Sign In		
Created By:	Hasan	**Last Updated By:**	Hasan
Date Created:	03-10-2020	**Last Revision Date:**	06-11-2020

Actors:	Customer	
Description:	This use case begins when the Customer wants to login in the Smart FM Android application system. Customer will enter the required fields such as email and password.	
Trigger:	Sign In form will be opened.	
Preconditions:	Customer provides email and password into login form and click on login button for login into the system.	
Post conditions:	Customer will be login into the system and then he will able to use the system.	
Normal Flow:	Customer	System
	1. Customer will get login by providing required data by the system.	**2.** System will login into the system.
Alternative Flows:	1. Customer cancel the sign in request. 2. Please fill email field 3. Please fill password field	
Exceptions:	• Firebase is not responding. • Internet connection is not available	

Table 2.5: Use case Description

Use Case 3 Title: Show Products (for Customers)

This use case description shows how to show products for customers into the system.

Use Case ID:	FM3		
Use Case Name:	Show Products		
Created By:	Hasan	**Last Updated By:**	Hasan
Date Created:	03-10-2020	**Last Revision Date:**	06-11-2020

Actors:	Customer	
Description:	This use case begins when the Customer wants to login in the Smart FM Android application system. Customer will show products.	
Trigger:	Show products page will be displayed.	
Preconditions:	Customer clicks on show products Tab.	
Post conditions:	Customer will enter into the products page and he/she will view the products list.	
Normal Flow:	Customer	System
		1. System will be displaying Products.
	2. By clicking show products.	**3.** system will display product with detail
	4. Actor add product into cart by clicking.	**5.** system will add product into cart.
Alternative Flows:	1. Customer come back to home page	
Exceptions:	• Firebase is not responding. • Internet connection is not available	

Table 2.6: Use case Description

Use Case 4 Title: Orders (for Customers)

This use case description shows how to order the product into the system.

Use Case ID:	FM3		
Use Case Name:	Orders		
Created By:	Hasan	**Last Updated By:**	Hasan
Date Created:	03-10-2020	**Last Revision Date:**	06-11-2020

Actors:	Customer
Description:	This use case begins when the Customer wants to login in the Smart FM Android application system. Customer will Orders.
Trigger:	Order page will be displayed.
Preconditions:	Customer clicks on Order Tab.

Post conditions:	Customer will enter into the order page and he/she will order the products.	
Normal Flow:	Customer	System
	1. Actor clicks on the order	**2.** System will be displaying order.
	2. Actor can view products by clicking order and also actor can delete order by clicking delete option.	**3.** system will delete orders.
Alternative Flows:	1. Customer cancel order request. 2. Customer change the product 3. Customer come back to home page	

Exceptions:	• Firebase is not responding. • Internet connection is not available

Table 2.7: Use case Description

Use Case 5 Title: Sign Up (for Admin)

This use case description shows how to sign up for Admin into the system.

Use Case ID:	FM5		
Use Case Name:	Sign Up		
Created By:	Hasan	**Last Updated By:**	Hasan
Date Created:	03-10-2020	**Last Revision Date:**	06-11-2020

Actors:	Admin	
Description:	This use case begins when the Admin wants to register in the Smart FM Android application system. Admin will enter the required fields such as first name, last, name, phone no, email and password.	
Trigger:	Sign Up form will be opened.	
Preconditions:	Admin provides fields into registration form and click on signup button for creating account into the system.	
Post conditions:	Admin will be register into the system and then he will able to use the system.	
Normal Flow:	Admin	System
	1. Admin will get register by providing required data by the system.	**2.** System will display admin information saved message.
Alternative Flows:	1. Please fill out the fields 2. Admin cancel the sign-up request	

Exceptions:	• Firebase is not responding. • Internet connection is not available

Table 2.8: Use case Description

Use Case 6 Title: Sign In (for Admin)

This use case description shows how to sign in for Admin into the system.

Use Case ID:	FM5		
Use Case Name:	Sign In		
Created By:	Hasan	**Last Updated By:**	Hasan
Date Created:	03-10-2020	**Last Revision Date:**	06-11-2020

Actors:	Admin	
Description:	This use case begins when the Admin wants to register in the Smart FM Android application system. Admin will enter the required fields such as email and password.	
Trigger:	Sign In form will be opened.	
Preconditions:	Admin provides email and password into sign in form and click on login button for login into the system.	
Post conditions:	Admin will be login into the system and then he will able to use the system.	
Normal Flow:	Admin	System
	1. Admin will get login by providing required data by the system.	**2.** System will login into the system.
Alternative Flows:	1. Admin cancel the sign in request. 2. Please fill email field 3. Please fill password field	

Exceptions:	• Firebase is not responding. • Internet connection is not available

Table 2.9: Use case Description

Use Case 7 Title: Manage Products (for Admin)

This use case description shows how to manage products into the system.

Use Case ID:	FM6		
Use Case Name:	Manage Products		
Created By:	Hasan	**Last Updated By:**	Hasan
Date Created:	03-10-2020	**Last Revision Date:**	06-11-2020
Actors:	Admin		
Description:	This use case begins when the Admin wants to register in the Smart FM Android application system. Admin will enter the required fields such as email and password.		
Trigger:	Manage Products form will be opened.		
Preconditions:	Admin clicks on the manage products tab.		
Post conditions:	Customer will enter into the manage products page and he/she will manage the products by adding items.		
Normal Flow:	Admin	System	
	1. This UC begins when Actor click on the add item	**2.** System will display add item page.	
	3. Actor will enter files i.e., add picture, name, price, quantity, description.	**4.** System validate all entries and add item.	

Alternative Flows:	1. Admin cancel the manage products request. 2. Please fill out the product fields
Exceptions:	• Firebase is not responding. • Internet connection is not available

Table 2.10: Use case Description

Use Case 8 Title: Order (for Admin)

This use case description shows how to accept the from order.

Use Case ID:	FM6		
Use Case Name:	Order		
Created By:	Hasan	**Last Updated By:**	Hasan
Date Created:	03-10-2020	**Last Revision Date:**	06-11-2020

Actors:	Admin	
Description:	This use case begins when the Admin wants to register in the Smart FM Android application system. Admin will enter the required fields such as email and password.	
Trigger:	Order form will be opened.	
Preconditions:	Admin clicks on Order Tab.	
Post conditions:	Admin will enter into the order page and he/she will Manage order the products accepting and rejecting.	
Normal Flow:	Admin	System
	1. This UC begins when Actor click on the order.	**2.** System will display orders.
Alternative Flows:	1. Admin cancel the order request. 2. Admin come back to home page	

Exceptions:	• Firebase is not responding. • Internet connection is not available

Table 2.11: Use case Description

Use case description: Controlled shed environment

Use Case 9 Title: Manage Farm

This use case description shows how to manage farm into the system.

Use Case ID:	FM2		
Use Case Name:	Manage Farm		
Created By:	Hasan	**Last Updated By:**	Hasan
Date Created:	03-10-2020	**Last Revision Date:**	06-11-2020
Actors:	Admin		
Description:	Admin will be able to Manage.		
Trigger:	Manage farm page will be displayed.		
Preconditions:	Admin clicks on crops details tab.		
Post conditions:	Admin will enter into the crops details page and he/she will view the real-time values of crops.		
Normal Flow:	Admin	System	
	1. If Actor selects Potato Farm.		
		2. System will view the values/status of temperature, humidity, fan, motor and led	
	2. If Actor selects **(2.2.)** Tomato Farm.		

		3. System will view the values/status of temperature, humidity, fan, motor and led.
Alternative Flows:	Admin cancel the crops detail request.	
Exceptions:	• Database is not responding • Internet connection is not available	

Table 2.12: Use case Description

2.5. System Sequence diagrams

System Sequence Diagrams are created to show the action among user and the system.

This SSD shows how to sign up for admin into the system.

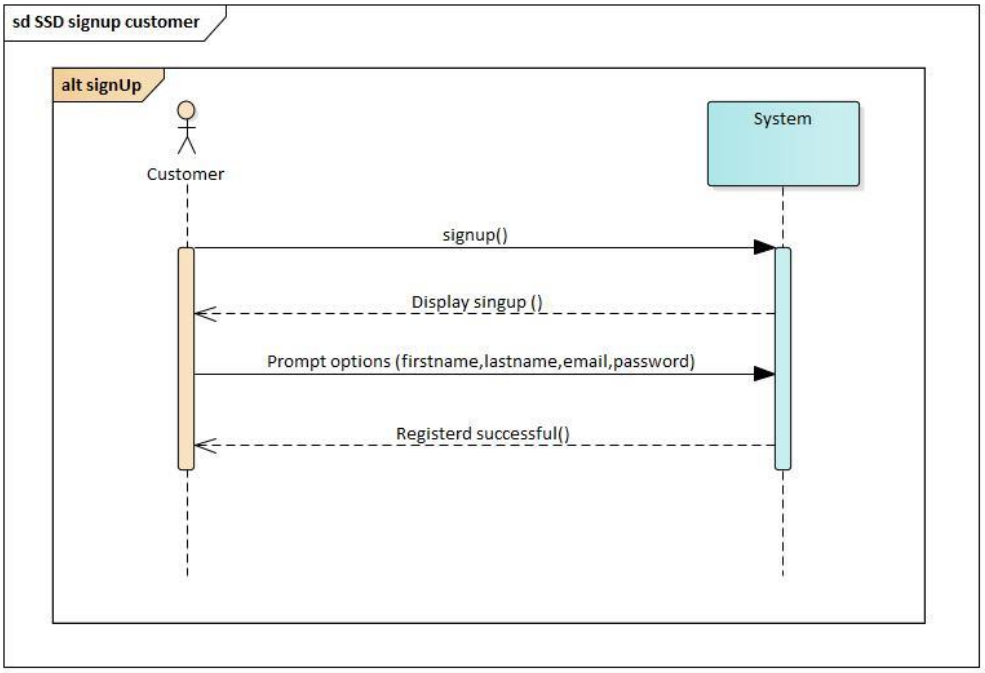


Figure 2.4: System Sequence Diagram

This SSD shows how to login for customers into the system.

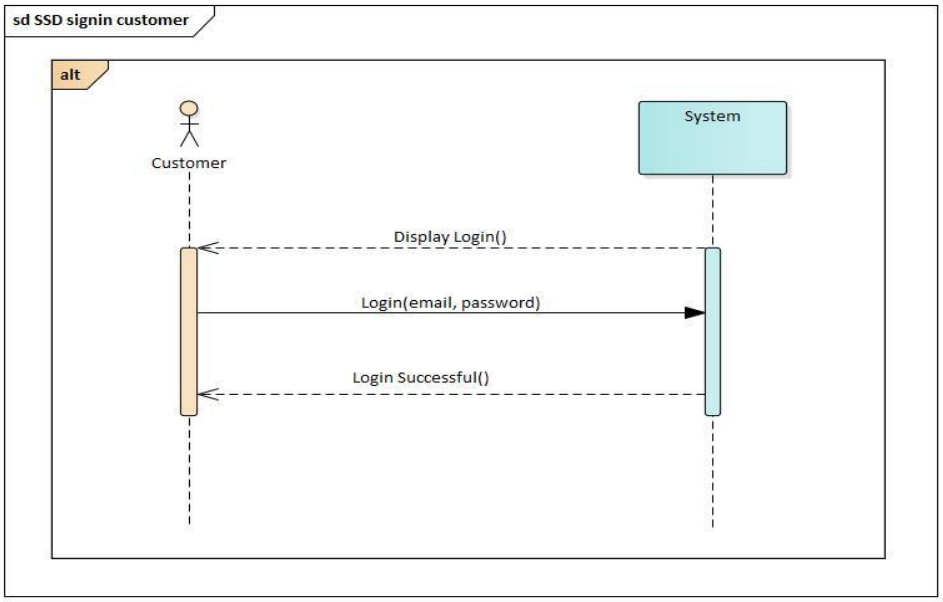


Figure 2.5: System Sequence Diagram

This SSD shows how to view products and add then into cart into the system.

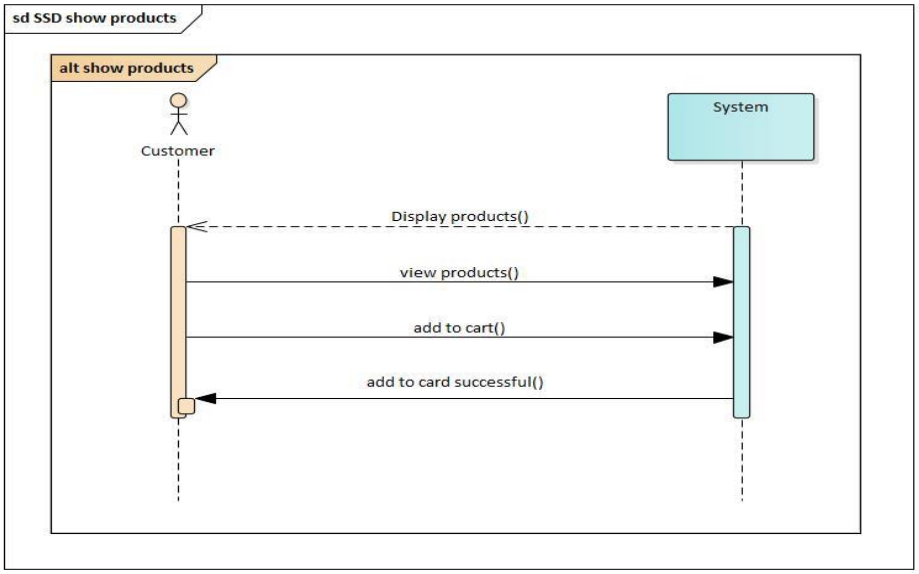


Figure 2.6: System Sequence Diagram

This SSD shows how to cancel order into the system.

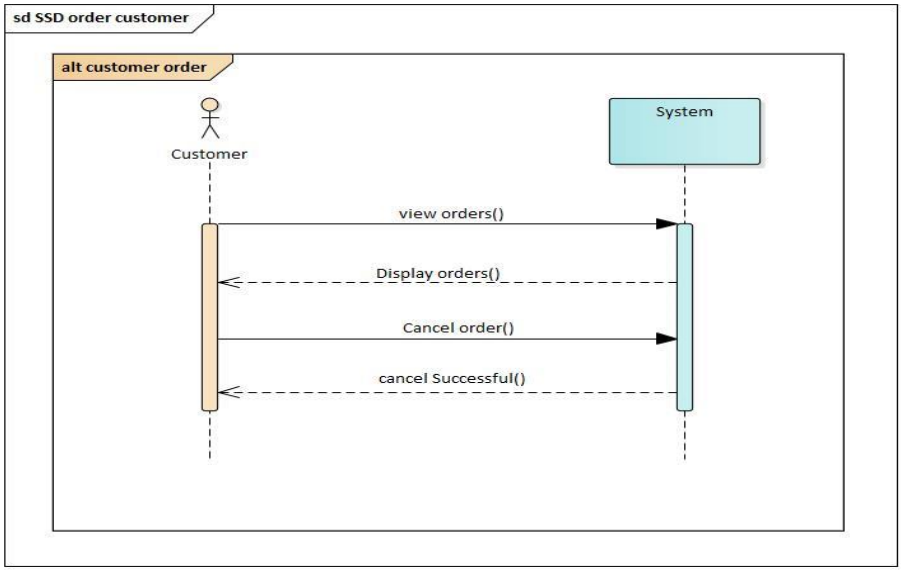


Figure 2.7: System Sequence Diagram

This SSD shows how to sign up for customer into the system.

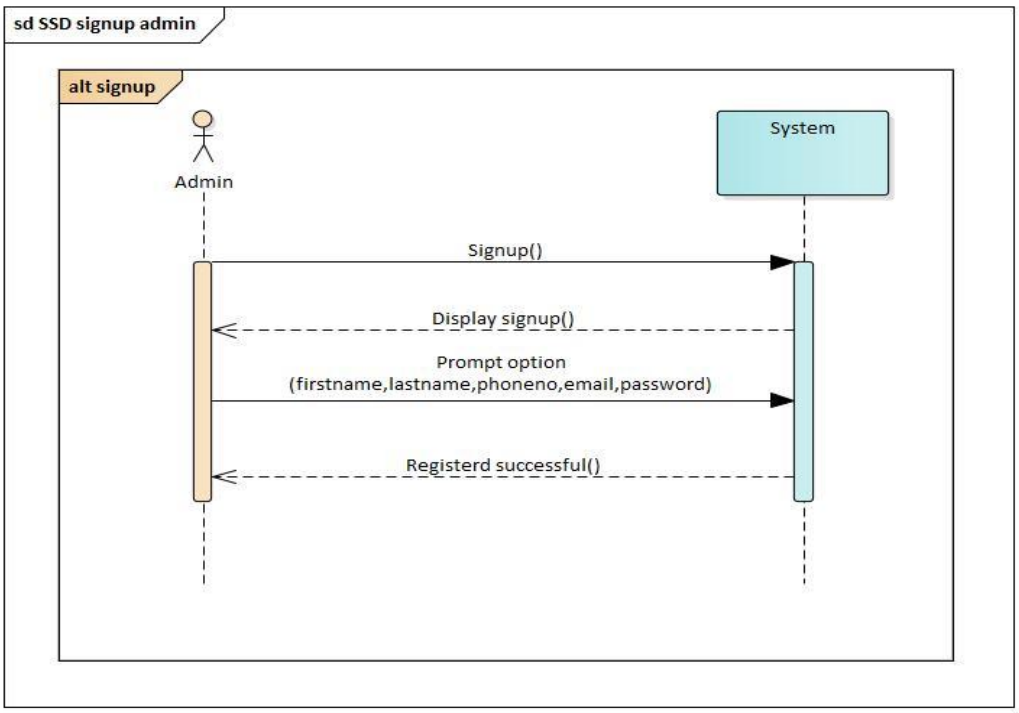


Figure 2.8: System Sequence Diagram

This SSD shows how to login for admis into the system.

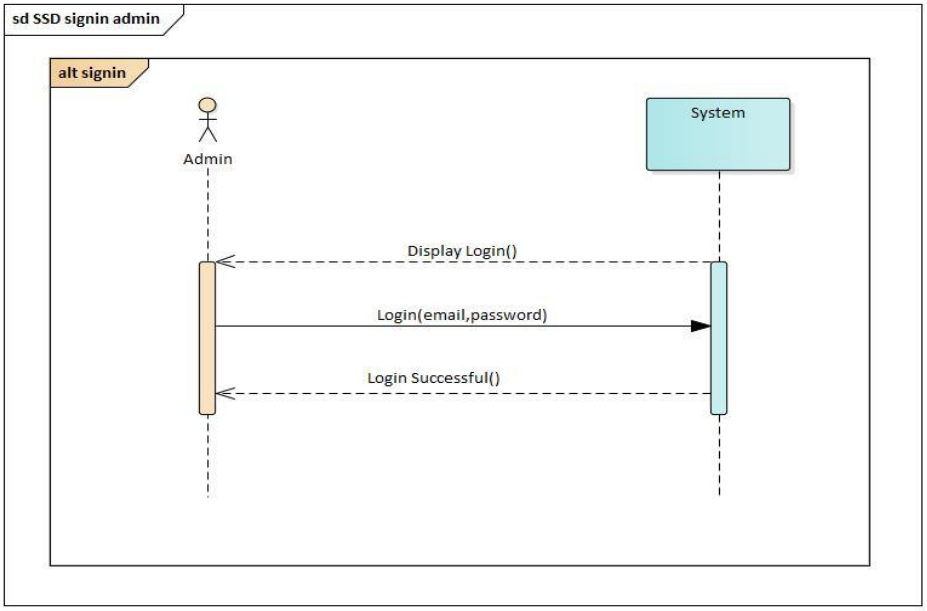


Figure 2.9: System Sequence Diagram

This SSD shows how to add item into the system.

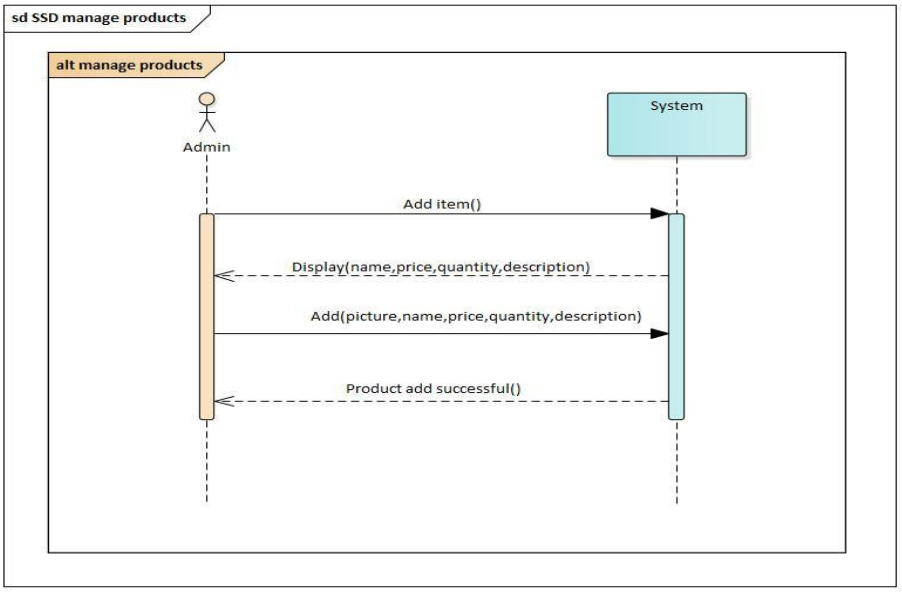


Figure 2.10: System Sequence Diagram

This SSD shows how to accept order or reject order into the system.

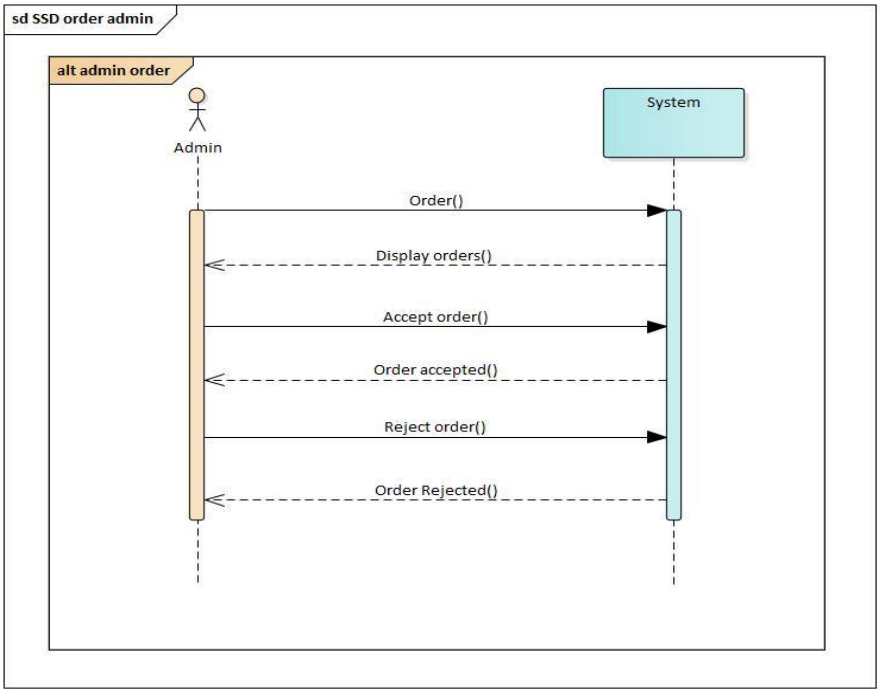


Figure 2.11: System Sequence Diagram

SSD: Controlled shed environment

This SSD shows how to view temperature reading into the system.

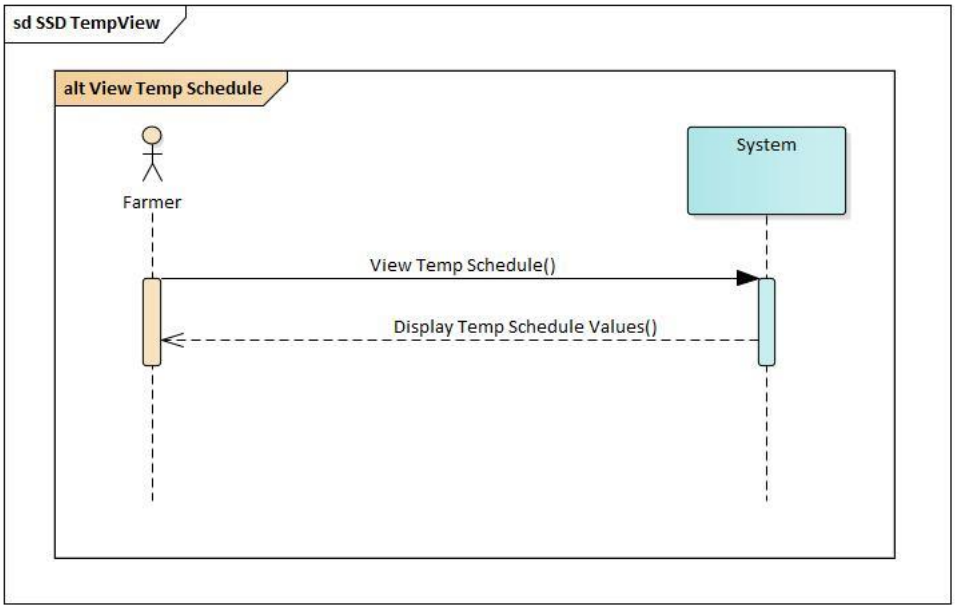


Figure 2.16: System Sequence Diagram

This SSD shows how to view humidity reading into the system.

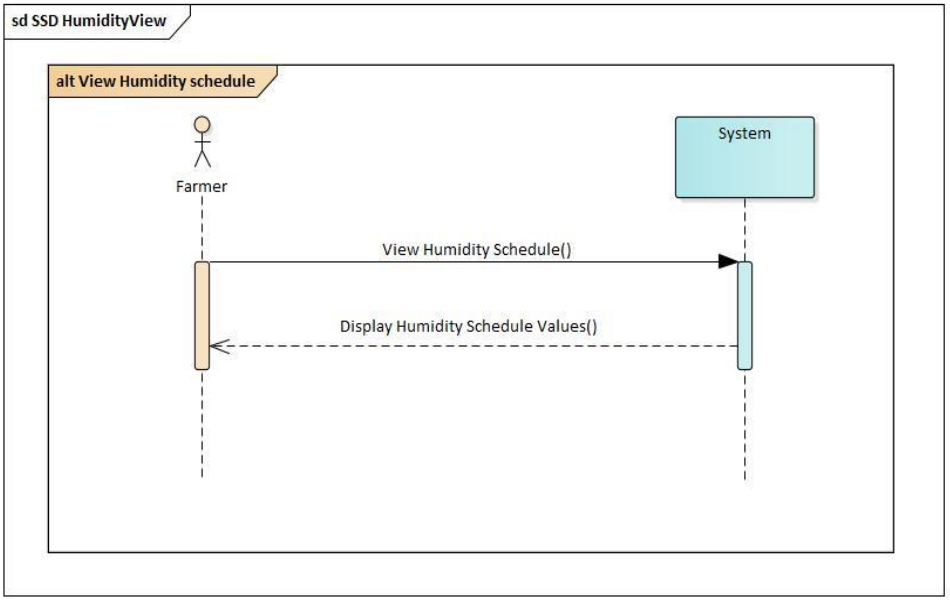


Figure 2.17: System Sequence Diagram

This SSD shows how to view light status into the system.

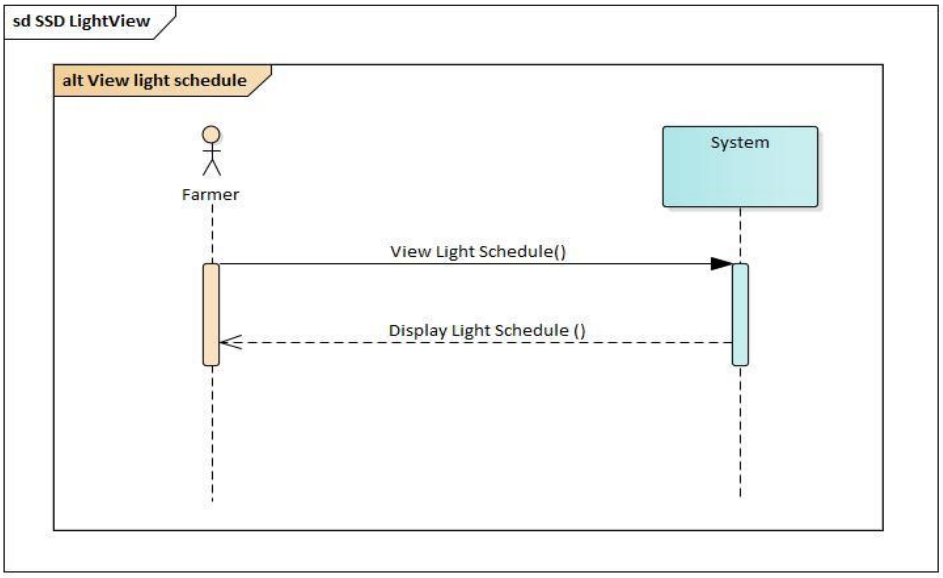


Figure 2.18: System Sequence Diagram.

This SSD shows how to view fan status into the system.

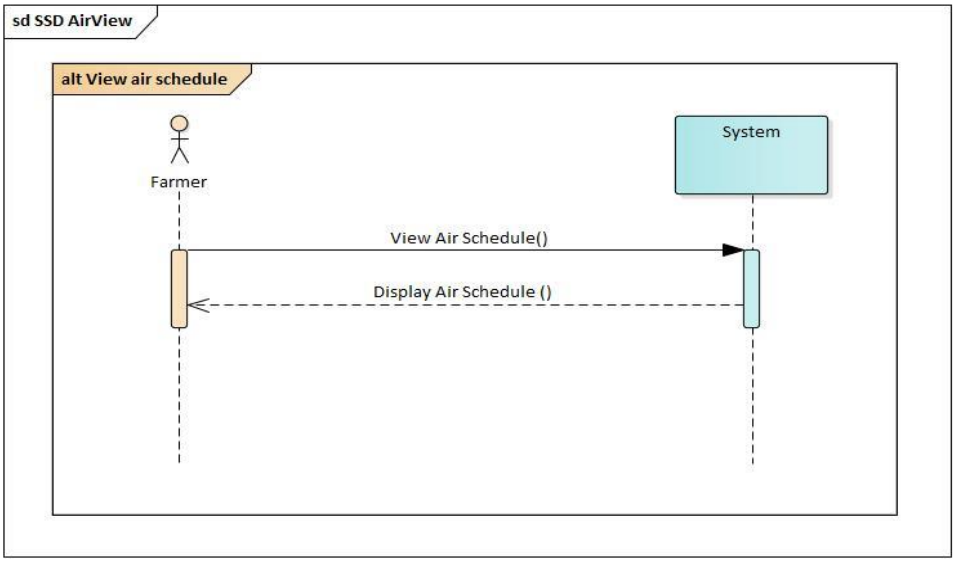


Figure 2.19: System Sequence Diagram

2.6. Sequence diagrams

Sequence Diagrams are created to show the action among user and the system.

SD: E- commerce Management

This SD shows complete structure of E-commerce management system.

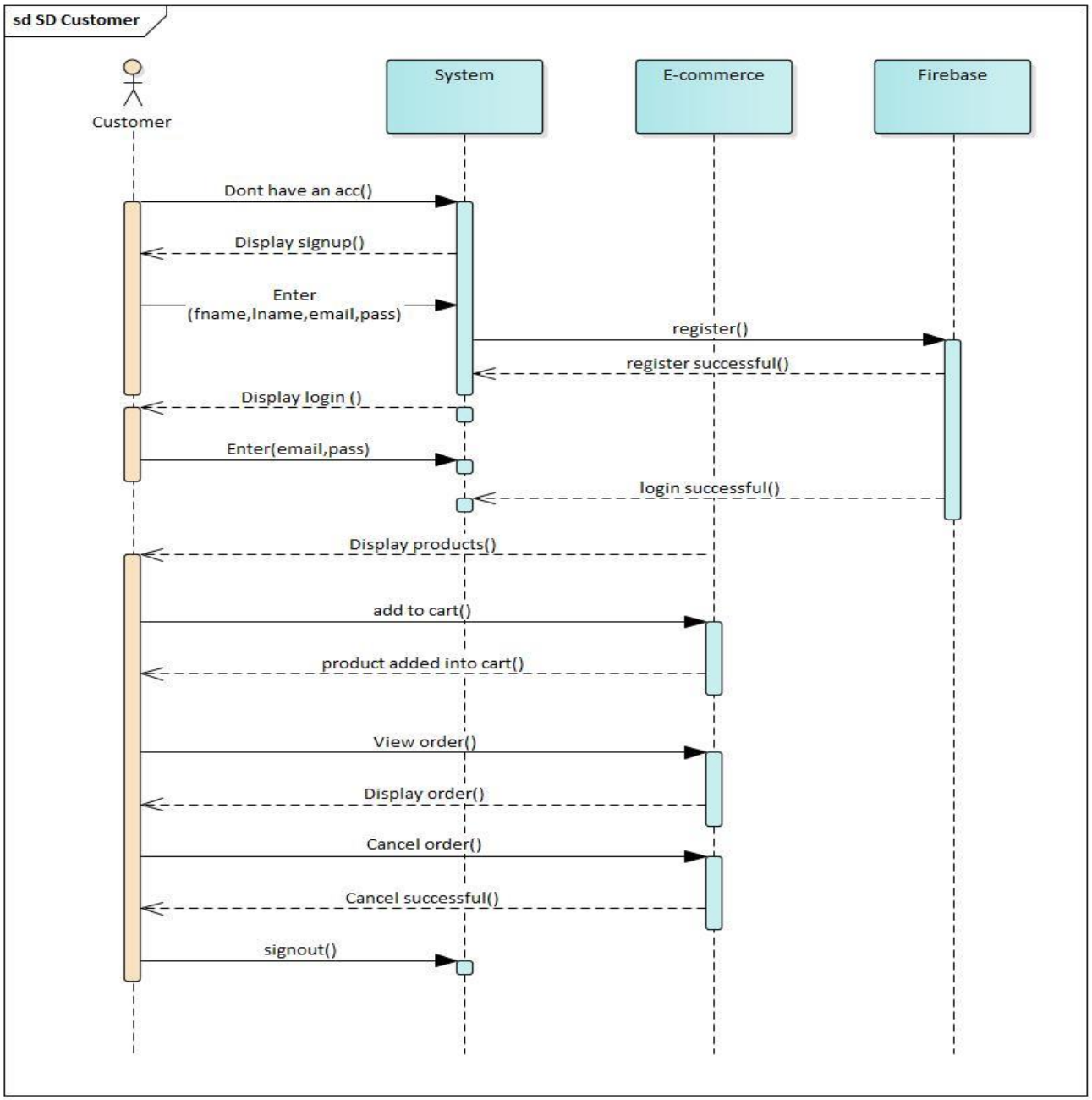


Figure 2.20: Sequence Diagram Customer

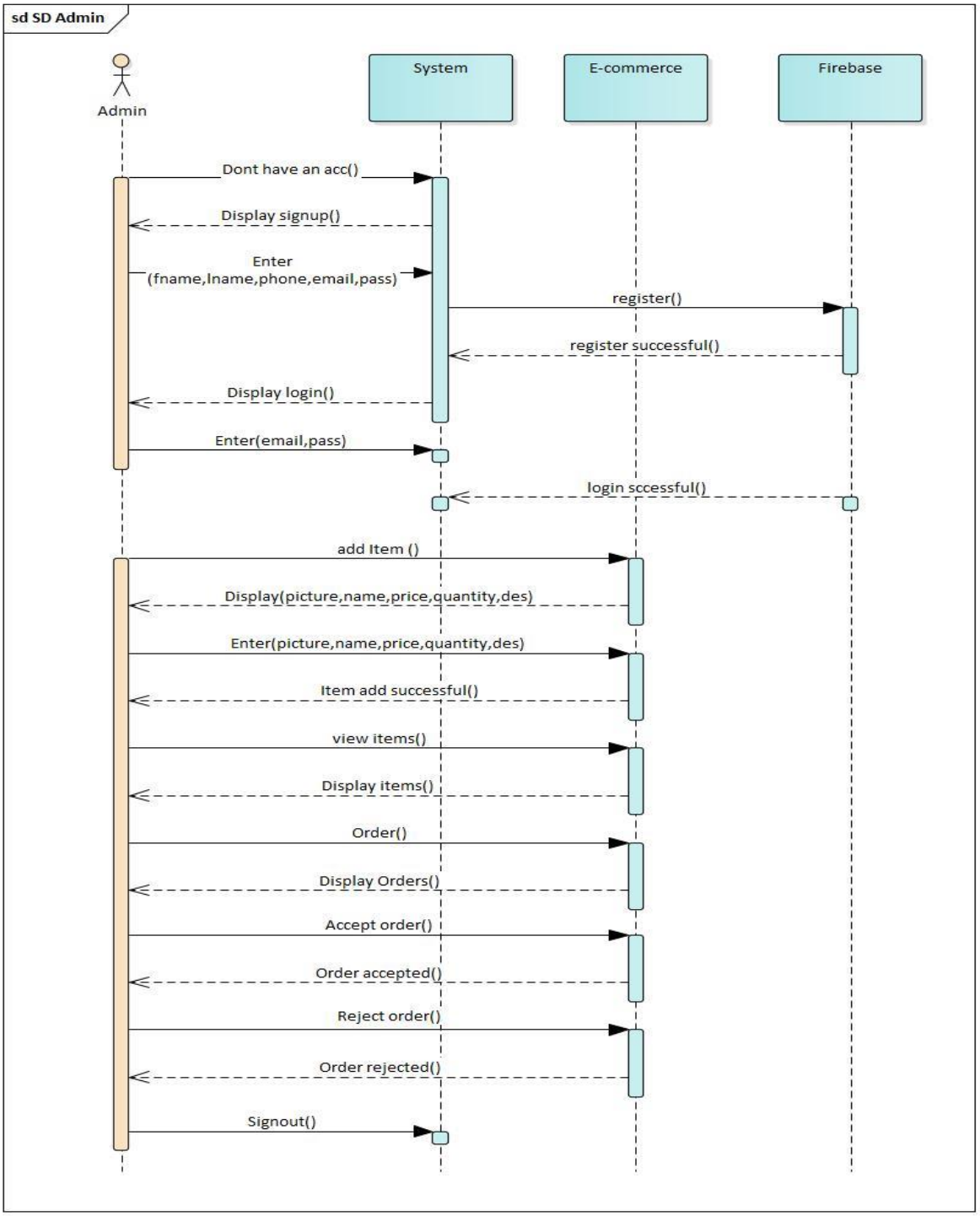


Figure 2.21: Sequence Diagram for Admin

This SD shows complete the structure of Farm management system.

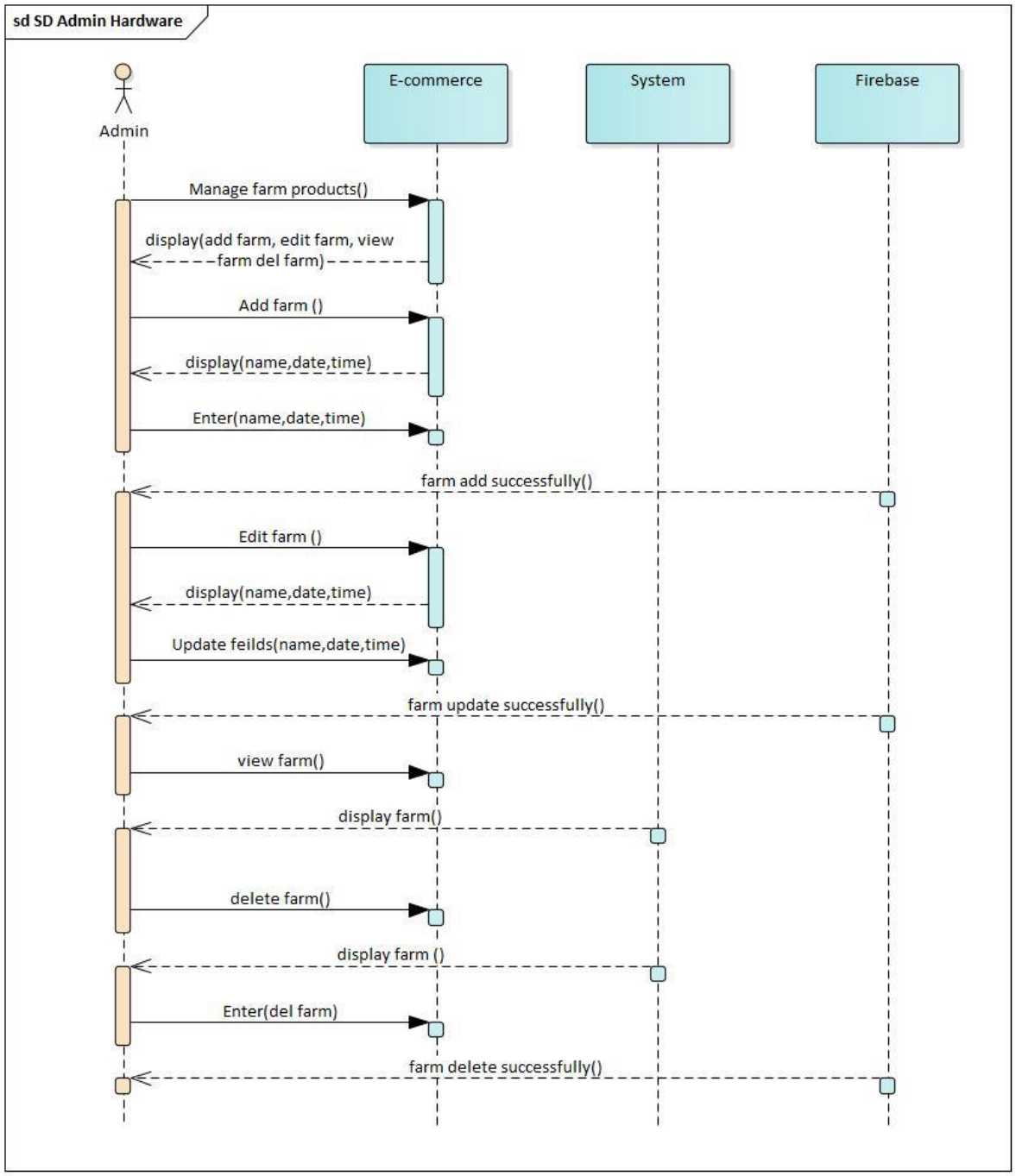


Figure 2.22: Sequence Diagram for Admin

2.7. Domain Model

Domain model is a conceptual model of the domain that incorporates both behavior and data.

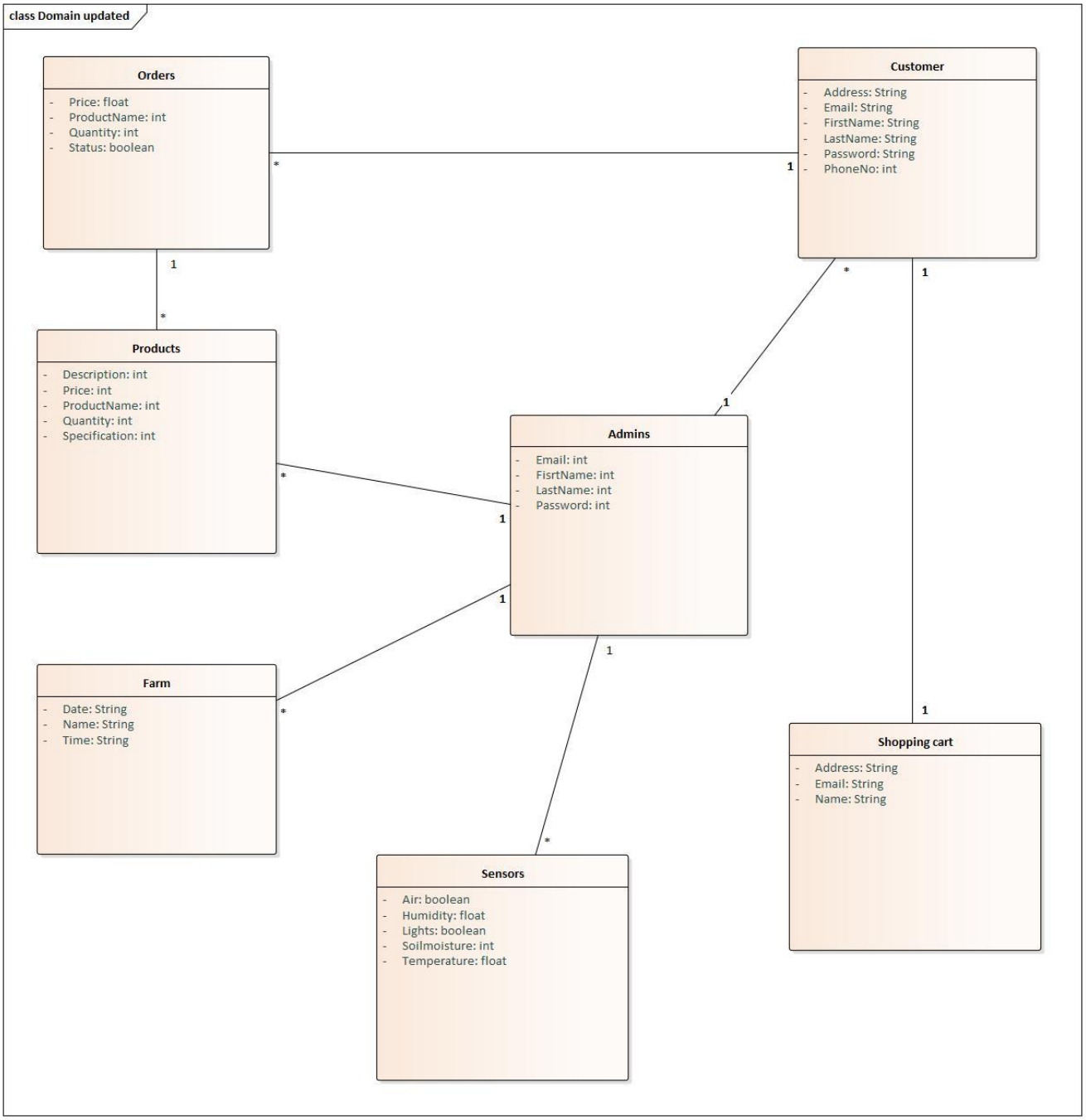


Figure 2.23: Domain Model

Chapter 3

System Design

3.1. Software Architecture

An Architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system.

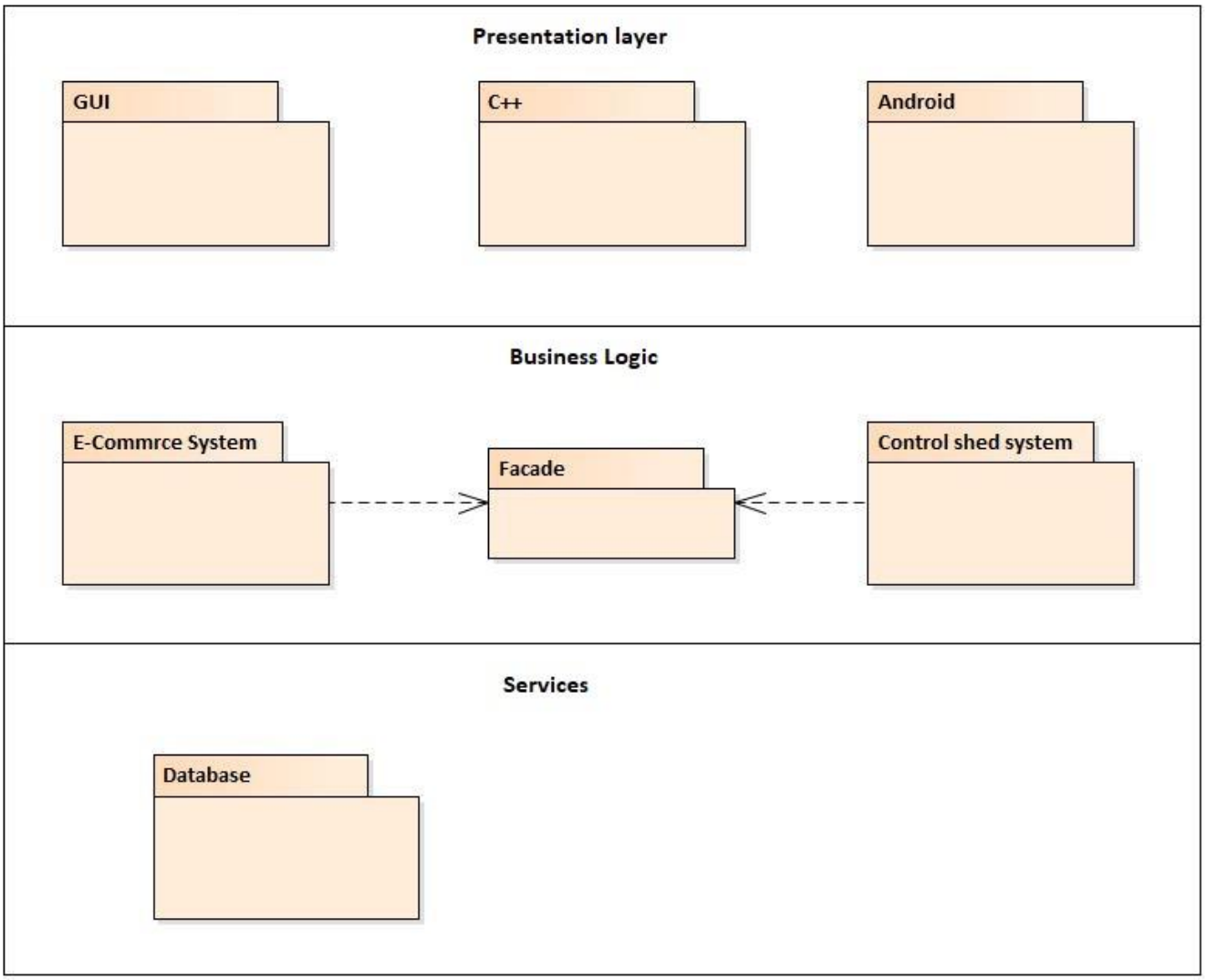


Figure 3.1: Software Architecture Diagram

3.2. Class Diagram

Class diagram describe the structure of a system by showing the system classes.

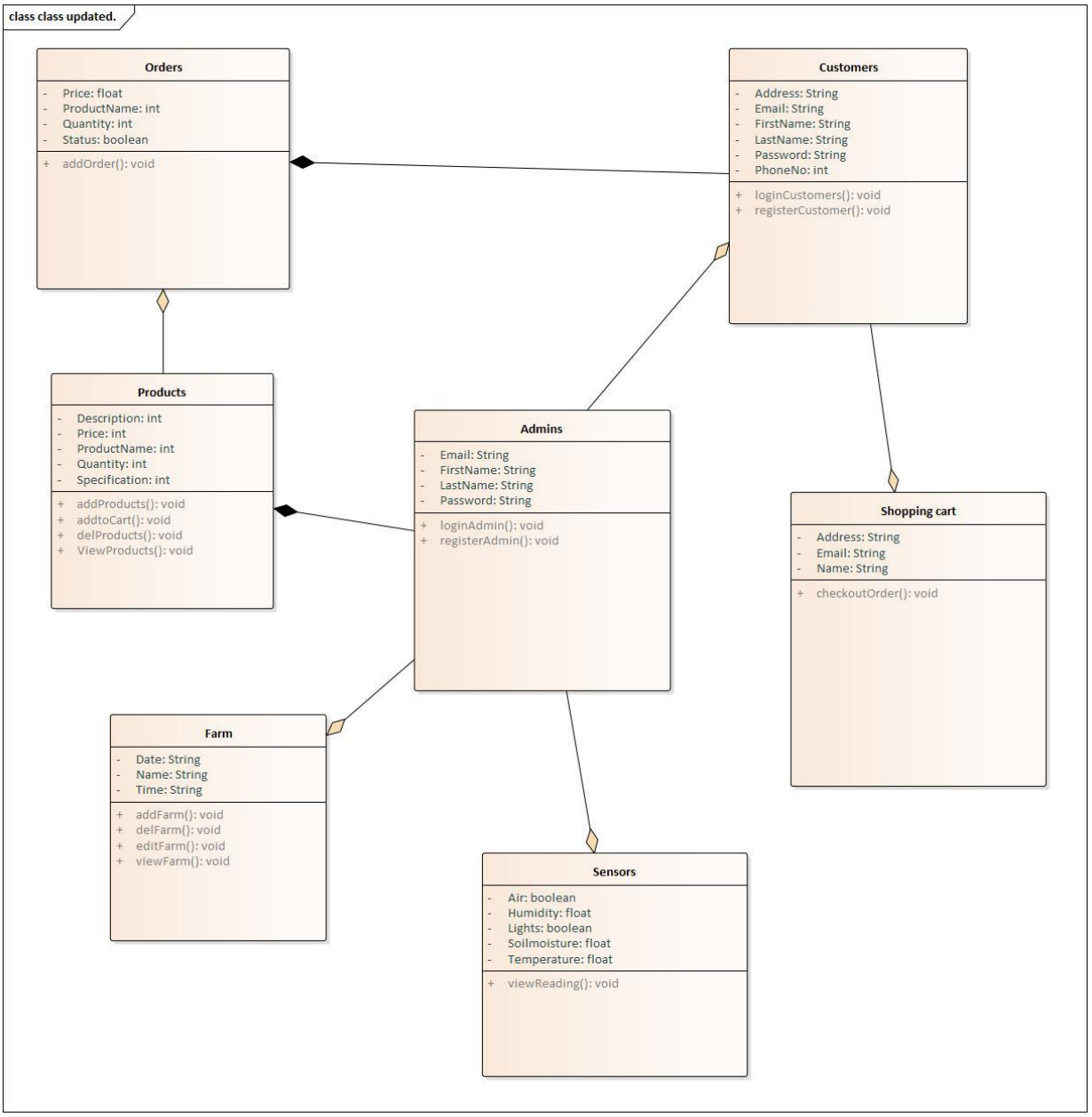


Figure 3.2: Class Diagram

3.3. User Interface Design

Admin and Customer login.

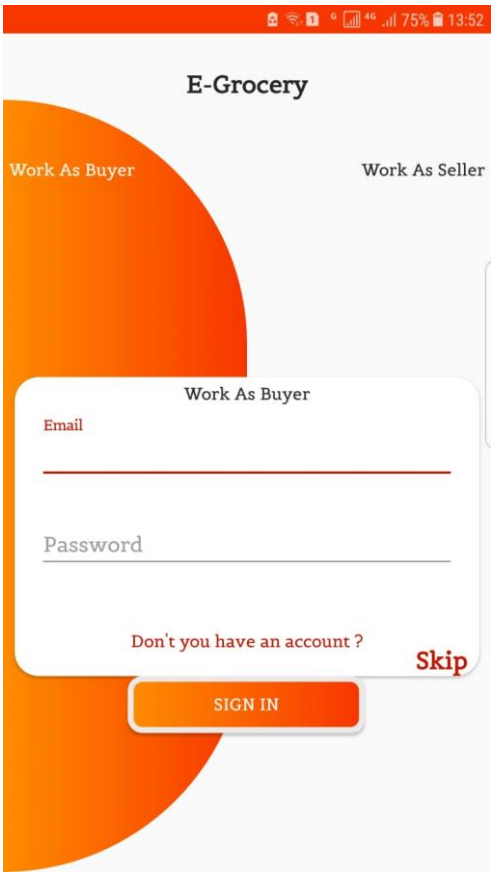


Figure 3.3: User Interface

Sign Up page for Admin and Customer.

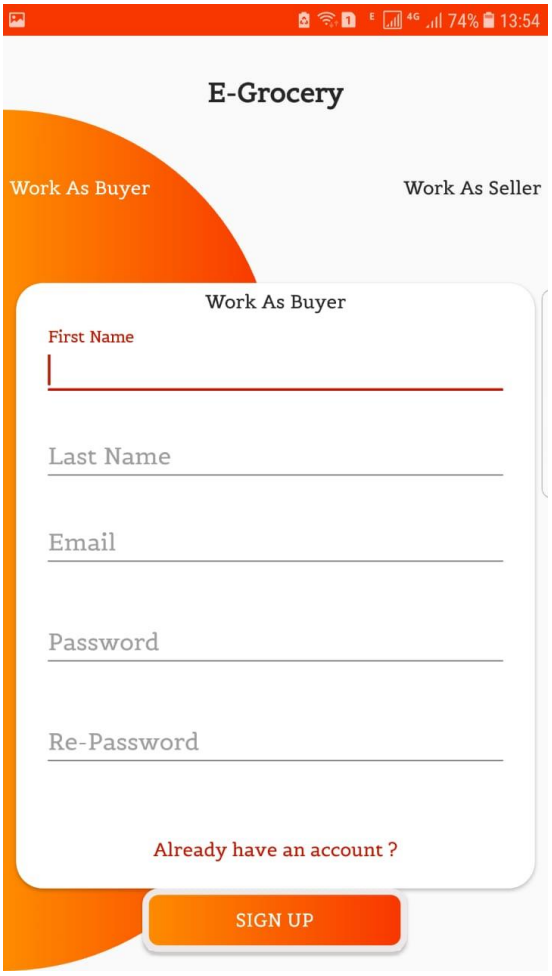


Figure 3.4: User Interface

Admin can add products, view product and check orders on the other hand Admin can add product by given fields.

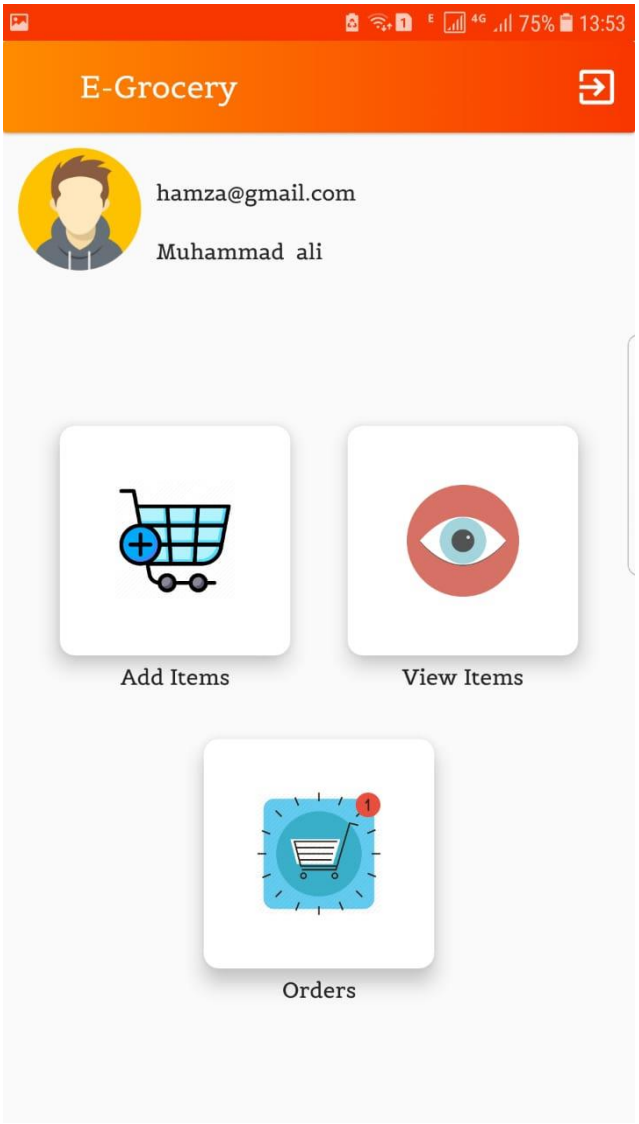


Figure 3.5: User Interface

Profile of Admin and Customer on the other hand customer can checkout products.

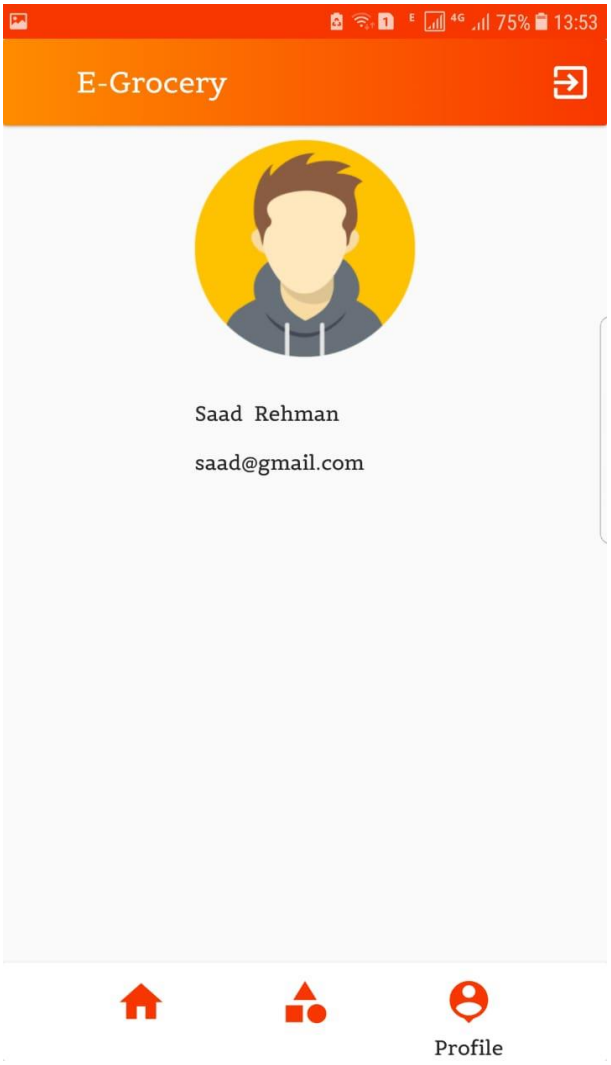


Figure 3.6: User Interface

Home page of Admin and Farm Activities.

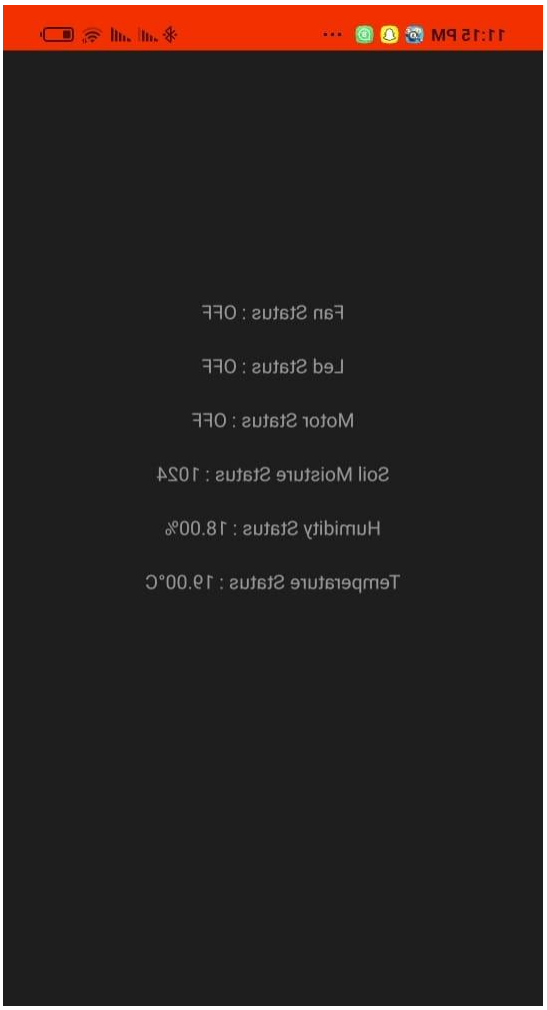


Figure 3.7: User Interface

Chapter 4

Software Development

4.1. Coding Standards

The adopted coding standards are discussed in the following subsections.

4.1.1. Indentation

Four spaces are used as the unit of indentation. The indentation pattern should be consistently followed throughout.

4.1.2. Declaration

One declaration per line is used to enhance the clarity of code. The order and position of declaration is as follows:

- First the static/class variables are placed in the sequence: First public class variables, protected.
- package/default level i.e., with no access modifier and then the private. As far as possible static or class fields are explicitly instantiated.
- Instance variables are placed in the sequence: First public instance variables, protected,
- package level with no access modifier and then private.
- Next the class constructors are declared.
- Class methods are grouped by functionality rather than by scope or accessibility to make reading and understanding the code easier.
- Declarations for local variables are only at the beginning of blocks e.g. at the beginning of a try/catch construct.

4.1.3. Statement Standards

Each line contains at most one statement while compound statements are those that contain list of statement enclosed in braces. The enclosed statements are indented one more level than the compound statements. Braces are used around all statements even single statements when they are part of control structure such as if-else or for statement.

4.1.4. Naming Convention

Naming conventions make program more understandable by making them easier to read.

Following are the conventions that were used:

Full English descriptors are used that accurately describe the variable, method or class.

E.g., use of Student Name instead of s1, s2 etc.

- Terminology that is applicable to domain is used.
- Mixed case is used to make names readable with lower case letters in general and capitalizing the first letter of the class name.

4.2. Development Environment

- Android Studio is the official integrated development environment (IDE) for the Android platform. It was announced on May 16, 2013 at the Google I/O conference. Android Studio is freely available under the Apache License 2.0. The reason for using android studio was that it provides a very interactive and easy to understand interface to work with android devices.

- Dot Net is a platform for developing, deploying and maintaining desktop, Enterprise, Internet and smart devices application. Dot Net Provides developer various features such as:
    - Multiple Languages based Development
    - Cool Development environment, Visual Studio .NET
    - Support for front-end technologies: HTML, XML, Java script, jQuery, CSS, AJAX etc.
    - A huge and powerful class library with over 2000 classes (data structure, data connection, threading, code security etc.)

4.3. Database Management System

A relational DBMS of Microsoft that is a major component of the Windows Server System. It is Microsoft's high-end client/server database and is closely integrated with Microsoft Visual Studio and the Microsoft Office System. Numerous editions are available, including those for Enterprise, Developer, Workgroup and 64-bit platforms.

4.4. Software Description

Main modules of our project are:

- Submission of Complaint
- Sale Creation for Employee
- Employee Login & Maintaining Sessions

Coding E-Commerce Management: Buyer Home

This code is belonging to customer home page

```
package com.example.ecommerce.buyer.activity;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.SearchView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.ecommerce.R;
import com.example.ecommerce.buyer.adapter.HomeAdapter;
import com.example.ecommerce.common.Registration;
import com.example.ecommerce.common.constant;
import com.example.ecommerce.common.loader;
import com.example.ecommerce.seller.model.ItemsModel;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

public class BuyerHome extends AppCompatActivity {
   private  SearchView searchView;
   private RecyclerView recyclerView;
   private List<ItemsModel> data=new ArrayList<>();
   loader loading;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_home);
      bottombar();
      init();
      recyclerView =  findViewById(R.id.recyleview);
```

```
    loading=new loader(this,getWindow().getDecorView().getRootView());
    loading.show();
    fetchSeller();
}

private void bottombar() {
    findViewById(R.id.categorytxt).setVisibility(View.INVISIBLE);
    findViewById(R.id.profiletxt).setVisibility(View.INVISIBLE);
    findViewById(R.id.hometxt).setVisibility(View.VISIBLE);
    findViewById(R.id.category).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(constant.Email.isEmpty()){
                AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(BuyerHome.this);
                alertDialogBuilder.setMessage("Need for login to see orders");
                alertDialogBuilder.setPositiveButton("Login",
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface arg0, int arg1) {
                                startActivity(new Intent(BuyerHome.this, Registration.class));
                            }
                        });

                alertDialogBuilder.setNegativeButton("No",new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss();
                    }
                });

                AlertDialog alertDialog = alertDialogBuilder.create();
                alertDialog.show();
            }
            else {
                startActivity(new Intent(BuyerHome.this, SeeOrder.class));
                overridePendingTransition(0, 0);
                finish();
            }
        }
    });
    findViewById(R.id.profile).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(constant.Email.isEmpty()){
                AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(BuyerHome.this);
                alertDialogBuilder.setMessage("Need for login to see profile");
                alertDialogBuilder.setPositiveButton("Login",


new DialogInterface.OnClickListener() {
```

```
        @Override
        public void onClick(DialogInterface arg0, int arg1) {

                            startActivity(new Intent(BuyerHome.this, Registration.class));
                        }
                    });

                alertDialogBuilder.setNegativeButton("No",new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss();
                    }
                });

                AlertDialog alertDialog = alertDialogBuilder.create();
                alertDialog.show();
            }
            else{
            startActivity(new Intent(BuyerHome.this,Profiles.class));
            overridePendingTransition(0,0);
            finish();}
        }
    });
}

    private void init() {
        findViewById(R.id.cart).setOnClickListener(new View.OnClickListener() {@Override public void
onClick(View v) { startActivity(new Intent(BuyerHome.this, CartScreen.class)); }});
        findViewById(R.id.back).setVisibility(View.GONE);
        recyclerView =  findViewById(R.id.recyleview);
        recyclerView=findViewById(R.id.recyleview);
        searchView=findViewById(R.id.search);
        searchView.setOnCloseListener(new SearchView.OnCloseListener() {
            @Override
            public boolean onClose() {
                findViewById(R.id.header).setVisibility(View.VISIBLE);
                return  false;
            }
        });
        searchView.setOnSearchClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                findViewById(R.id.header).setVisibility(View.GONE);          }
        });
    }

    private void fetchSeller() {
        FirebaseFirestore db = FirebaseFirestore.getInstance();

        db.collection("Registration")
                .get()
```

```
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {

                    if (task.isSuccessful())



                    {
                        List<String> doc=new ArrayList<>();
                        for (QueryDocumentSnapshot document : task.getResult()) {
                          if(document.getString("Type").equalsIgnoreCase("seller"))
                              doc.add(document.getId());
                        }
                        if(doc.size()>0){
                         fetchRecord(doc);
                            findViewById(R.id.found).setVisibility(View.GONE);
                        }
                        else
                        {
                            loading.dismiss();
                            findViewById(R.id.found).setVisibility(View.VISIBLE);
                        }

                    } else {


Toast.makeText(BuyerHome.this,"Error=>"+task.getException(),Toast.LENGTH_SHORT).show();
                        loading.dismiss();

                    }
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(BuyerHome.this,"Error=>"+e.getMessage(),Toast.LENGTH_SHORT).show();
                    loading.dismiss();
                }
            });;
    }


    private void fetchRecord(List<String> doc) {
```

```
        FirebaseFirestore db = FirebaseFirestore.getInstance();
        for (String document:doc) {
            db.collection("Products").document(document).collection("Details")
                    .get()
                    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                        @Override
                        public void onComplete(@NonNull Task<QuerySnapshot> task) {
                            if (task.isSuccessful()) {
                                for (QueryDocumentSnapshot document : task.getResult()) {


                                    data.add(new
ItemsModel(document.getString("Name"),document.getString("Specification"),
document.getString("Service"),  document.getLong("Quantity"),



document.getLong("Price"),document.getString("Description"),document.getString("Image"),
document.getId()));
                                }
                                recyclerView.setLayoutManager(new GridLayoutManager(getApplicationContext(),2));
                                HomeAdapter customAdapter = new HomeAdapter(BuyerHome.this,data);
                                recyclerView.setAdapter(customAdapter);
                                if(data.size()==0)
                                    findViewById(R.id.found).setVisibility(View.VISIBLE);
                                else
                                    findViewById(R.id.found).setVisibility(View.INVISIBLE);
                            } else {
                                Log.w("TAG1", "Error getting documents.", task.getException());

Toast.makeText(BuyerHome.this,"Error=>"+task.getException(),Toast.LENGTH_SHORT).show();
                            }
                            loading.dismiss();
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(BuyerHome.this,"Error=>"+e.getMessage(),Toast.LENGTH_SHORT).show();
                    loading.dismiss();
                }
            });
        }
    }
}
```

Coding E-Commerce Management: Cart Screen

This code is belonging to customer cart screen

```
package com.example.ecommerce.buyer.activity;

import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.example.ecommerce.R;
import com.example.ecommerce.buyer.adapter.CartScreenAdapter;
import com.example.ecommerce.common.constant;
import com.example.ecommerce.common.loader;



import com.example.ecommerce.seller.model.ItemsModel;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;

public class CartScreen extends AppCompatActivity {
    RecyclerView recyclerView;
    FirebaseFirestore db;
    loader loading;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cartscreen);
        init();
        loading=new loader(this,getWindow().getDecorView().getRootView());

    }

    private void init() {
        db = FirebaseFirestore.getInstance();
        findViewById(R.id.back).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
```

```
        findViewById(R.id.search).setVisibility(View.GONE);
        findViewById(R.id.cart).setVisibility(View.GONE);
        recyclerView=findViewById(R.id.recyleview);
        recyclerView.setLayoutManager(new LinearLayoutManager(CartScreen.this));
        CartScreenAdapter customAdapter = new CartScreenAdapter(CartScreen.this,
constant.cartItems);
        recyclerView.setAdapter(customAdapter);
    }

    public void order(View view) {
        if(constant.cartItems.size()==0)
            Toast.makeText(this,"First Add Product In Cart",Toast.LENGTH_SHORT).show();
        else {
            loading.show();
            for (ItemsModel items : constant.cartItems) {
                Map<String, Object> user = new HashMap<>();
                user.put("Name", constant.FName + " " + constant.LName);
                user.put("Email", constant.Email);
                user.put("ID", items.getId());
                user.put("PName", items.getName());

                user.put("Price", items.getPrice());
                user.put("Quantity", items.getQuantity());



                user.put("Status", "Pending");
                user.put("Image", items.getImageurl());
                db.collection("Orders")
                        .add(user)
                        .addOnSuccessListener(new OnSuccessListener<DocumentReference>()
{
                            @Override
                            public void onSuccess(DocumentReference documentReference) {


                            }
                        })
                        .addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Toast.makeText(CartScreen.this, "Error==>" +
e.getMessage(), Toast.LENGTH_SHORT).show();
                            }
                        });
            }

            constant.cartItems.clear();
            Toast.makeText(this, "Successfully Order", Toast.LENGTH_SHORT).show();
            loading.dismiss();
            finish();
```

```
        }
    }
}


    private Button joinNowButton, loginButton;
    DatabaseReference reff;

    private ProgressDialog loadingBar;
    private TextView sellerBegin;

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

        joinNowButton = findViewById(R.id.main_join_now_btn);
        loginButton = findViewById(R.id.main_login_btn);
        sellerBegin = findViewById(R.id.seller_begin);
        loadingBar = new ProgressDialog(this);

        Paper.init(this);
        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MainActivity.this,
LoginActivity.class);
                startActivity(intent);
            }
        });

        sellerBegin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MainActivity.this,
SellerRegistrationActivity.class);
                startActivity(intent);
            }
        });

        joinNowButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                Intent intent = new Intent(MainActivity.this,
RegisterActivity.class);
                startActivity(intent);
            }
        });
```

```
        String UserPhoneKey = Paper.book().read(Prevalent.UserPhoneKey);
        String UserPasswordKey = Paper.book().read(Prevalent.UserPasswordKey);
        if (UserPhoneKey != "" && UserPasswordKey != "")
        {
            if (!TextUtils.isEmpty(UserPhoneKey)  &&
!TextUtils.isEmpty(UserPasswordKey))
            {
                AllowAccess(UserPhoneKey, UserPasswordKey);

                loadingBar.setTitle("Already Logged in");
                loadingBar.setMessage("Please wait .....");
                loadingBar.setCanceledOnTouchOutside(false);
                loadingBar.show();
            }
        }
        Toast.makeText(MainActivity.this,"Firebase connection
Successfully",Toast.LENGTH_LONG).show();
    }

    @Override
    protected void onStart() {
        super.onStart();

        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        if(firebaseUser != null)
        {
            Intent intent = new Intent(MainActivity.this,
SellerHomeActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(intent);
            finish();
        }
    }

    private void AllowAccess(final String phone, final String password) {
        final DatabaseReference RootRef;
        RootRef = FirebaseDatabase.getInstance().getReference();
        RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                if (dataSnapshot.child("Users").child(phone).exists()){

                    Users usersData =
dataSnapshot.child("Users").child(phone).getValue(Users.class);
                    if (usersData.getPhone().equals(phone))
                    {
```

```
                        if (usersData.getPassword().equals(password))
                        {
                            Toast.makeText(MainActivity.this, "Please wait, you
    are already logged in...", Toast.LENGTH_SHORT).show();
                            loadingBar.dismiss();
                            HomeActivity.class);
                            Intent intent = new Intent(MainActivity.this,

Prevalent.currentOnlineUser = usersData;
                            startActivity(intent);

                        }
                        else {
                            loadingBar.dismiss();
                            Toast.makeText(MainActivity.this,"Password is
    incorrect",Toast.LENGTH_SHORT).show();
                        }
                    }
                }
                else {
                    Toast.makeText(MainActivity.this, "Account with this " + phone
    + " number do not exists.", Toast.LENGTH_SHORT).show();

                    loadingBar.dismiss();
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }

}
```

Coding E-Commerce Management: Product Details

This code is blogging to detail of the products like name description etc.

```
package com.example.ecommerce.buyer.activity;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
```

```
import com.bumptech.glide.Glide;
import com.example.ecommerce.R;
import com.example.ecommerce.common.Registration;
import com.example.ecommerce.common.constant;
import com.example.ecommerce.seller.model.ItemsModel;

public class Details extends AppCompatActivity {
     ImageView imageView;
     TextView name,price,quantity,description;
     ItemsModel data;
     Long pr,qn;
     String Id;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_details);
        init();
    }

    private void init() {
        findViewById(R.id.back).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
        findViewById(R.id.search).setVisibility(View.GONE);
        findViewById(R.id.cart).setOnClickListener(new View.OnClickListener()
 {@Override public void onClick(View v) { startActivity(new Intent(Details.this,
CartScreen.class)); }});
        data = (ItemsModel) getIntent().getSerializableExtra("details");
        imageView=findViewById(R.id.img);
        name=findViewById(R.id.name);
        price=findViewById(R.id.price);
        quantity=findViewById(R.id.quantity);
        description=findViewById(R.id.description);
        Glide.with(this).load(data.getImageurl()).into(imageView);
        name.setText(data.getName()+"\n"+data.getSpecification()+"\n"+data.getService());
        price.setText(data.getPrice()+" Rs");
        quantity.setText(data.getQuantity()+" item(s)");
        description.setText(data.getDescription());
        qn=data.getQuantity();
        pr=data.getPrice();
        Id=data.getId();
    }

    public void addtocart(View view) {
        if(constant.Email.isEmpty()){
            AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
            alertDialogBuilder.setMessage("Need for login to order");
                    alertDialogBuilder.setPositiveButton("Login",
                            new DialogInterface.OnClickListener() {

                                @Override
                                public void onClick(DialogInterface arg0, int arg1) {
```

```
                                    startActivity(new Intent(Details.this,
Registration.class));
                                }
                            });

            alertDialogBuilder.setNegativeButton("No",new DialogInterface.
OnClickListener()
 {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                }
            });

            AlertDialog alertDialog = alertDialogBuilder.create();
            alertDialog.show();
        }
        else {
            if (data.getQuantity() > 0) {
                try {

                    if (constant.cartItems.size() == 0) {
                        constant.cartItems.add(data);
                        constant.cartItems.get(constant.cartItems.size() -
1).setQuantity(Long.valueOf(1));
                        constant.cartItems.get(constant.cartItems.size() -
1).setService(qn.toString());
                        Toast.makeText(this, "Add To Cart1", Toast.LENGTH_SHORT).show();
                    } else {
                        boolean flg = true;
                        for (int i = 0; i < constant.cartItems.size(); i++) {
                            Log.e("id_", Id);
                            if (constant.cartItems.get(i).getId() == Id) {
                                if (constant.cartItems.get(i).getQuantity() >= qn) {
                                    Toast.makeText(this, "Out of stock",
Toast.LENGTH_SHORT).show();
                                    flg = false;
                                    break;
                                }

constant.cartItems.get(i).setQuantity(constant.cartItems.get(i).getQuantity() + 1);

constant.cartItems.get(i).setPrice((constant.cartItems.get(i).getPrice() + pr));
                                constant.cartItems.get(i).setService(qn.toString());
                                flg = false;
                                Toast.makeText(this, "Add To Cart2",
Toast.LENGTH_SHORT).show();
                                break;
                            }

                        }
                        if (flg) {
                            constant.cartItems.add(data);
                            constant.cartItems.get(constant.cartItems.size() -
1).setQuantity(Long.valueOf(1));
```

```
                            constant.cartItems.get(constant.cartItems.size() -
1).setService(pr.toString());
                            Toast.makeText(this, "Add To Cart3", Toast.LENGTH_SHORT).show();
                        }

                    }

                } catch (Exception e) {
                    Log.d("IDS1", e.getMessage());
                }
            } else {
                Toast.makeText(this, "Out of Stock", Toast.LENGTH_SHORT).show();
            }
        }
    }

}
```

Coding E-Commerce Management: Profile

This code is blogging to profile of the user

```
package com.example.ecommerce.buyer.activity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.ecommerce.R;
import com.example.ecommerce.common.Registration;
import com.example.ecommerce.common.constant;
import com.google.firebase.auth.FirebaseAuth;

public class Profiles extends AppCompatActivity {
    TextView name,email;
    ImageView logout;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profiles);
        bottombar();
       init();
    }

    private void bottombar() {
        findViewById(R.id.back).setVisibility(View.INVISIBLE);
        findViewById(R.id.categorytxt).setVisibility(View.INVISIBLE);
        findViewById(R.id.profiletxt).setVisibility(View.VISIBLE);
        findViewById(R.id.hometxt).setVisibility(View.INVISIBLE);
        findViewById(R.id.category).setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {

                startActivity(new Intent(Profiles.this,SeeOrder.class));
                overridePendingTransition(0,0);
                finish();
            }
        });
        findViewById(R.id.home).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(Profiles.this,BuyerHome.class));
                overridePendingTransition(0,0);
                finish();
            }
        });
    }

    private void init() {
        name=findViewById(R.id.name);
        email=findViewById(R.id.email);
        name.setText(constant.FName+" "+constant.LName);
        email.setText(constant.Email);
        logout=findViewById(R.id.cart);
        logout.setImageResource(R.drawable.ic_baseline_exit_to_app_24);
        logout.setOnClickListener(new View.OnClickListener() {@Override public void
onClick(View v) {
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(Profiles.this, Registration.class));
        finishAffinity();}});
        findViewById(R.id.search).setVisibility(View.GONE);

    }
}
```

Coding E-Commerce Management: See Order

```
package com.example.ecommerce.buyer.activity;



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

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.ecommerce.R;
import com.example.ecommerce.buyer.adapter.CartScreenAdapter;
import com.example.ecommerce.buyer.model.OrderModel;
import com.example.ecommerce.common.constant;
```

```
import com.example.ecommerce.common.loader;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

public class SeeOrder extends AppCompatActivity {
    RecyclerView recyclerView;
    FirebaseFirestore db;
    List<OrderModel> data=new ArrayList<>();
    loader loading;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_seeorder);
        init();
        bottombar();

    }
    private void bottombar() {
        findViewById(R.id.categorytxt).setVisibility(View.VISIBLE);
        findViewById(R.id.profiletxt).setVisibility(View.INVISIBLE);
        findViewById(R.id.hometxt).setVisibility(View.INVISIBLE);
        findViewById(R.id.home).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(SeeOrder.this,BuyerHome.class));
                overridePendingTransition(0,0);
                finish();
            }
        });
        findViewById(R.id.profile).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(SeeOrder.this,Profiles.class));
                overridePendingTransition(0,0);
                finish();
            }
        });
    }

    private void init() {
        findViewById(R.id.back).setVisibility(View.INVISIBLE);
        findViewById(R.id.search).setVisibility(View.GONE);
        findViewById(R.id.cart).setVisibility(View.GONE);
        recyclerView=findViewById(R.id.recyleview);
        db = FirebaseFirestore.getInstance();
        loading=new loader(this,getWindow().getDecorView().getRootView());
        loading.show();
        fetchOrder();
```

```
    }

    private void fetchOrder() {

        db.collection("Orders")
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            for (QueryDocumentSnapshot document : task.getResult()) {
                              if(document.getString("Email").equals(constant.Email))
                                  data.add(new
OrderModel(document.getId(),document.getString("PName"),document.getString("Status")

,document.getString("Image"),document.getLong("Quantity"),document.getLong("Price"),
document.getString("ID")));

                            }
                            recyclerView.setLayoutManager(new
LinearLayoutManager(SeeOrder.this));
                            CartScreenAdapter customAdapter = new
CartScreenAdapter(SeeOrder.this, data,true,false);
                            recyclerView.setAdapter(customAdapter);
                            loading.dismiss();
                            if(data.size()==0)
                                findViewById(R.id.found).setVisibility(View.VISIBLE);
                            else
                                findViewById(R.id.found).setVisibility(View.INVISIBLE);
                        }
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {

Toast.makeText(SeeOrder.this,"Error==>"+e.getMessage(),Toast.LENGTH_SHORT).show();
                loading.dismiss();

            }
        });
    }
}
```

Coding E-Commerce Management: Cart Screen Adapters

This code is blogging to cart screen where admin accepted the products

```
package com.example.ecommerce.buyer.adapter;

import android.content.Context;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
```

```
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.ecommerce.R;
import com.example.ecommerce.buyer.model.OrderModel;
import com.example.ecommerce.common.constant;
import com.example.ecommerce.seller.model.ItemsModel;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CartScreenAdapter extends RecyclerView.Adapter<CartScreenAdapter.
MyViewHolder> {
    List<ItemsModel> data;
    List<OrderModel> order;
    Context context;
    FirebaseFirestore db;
    boolean flag;
    boolean check;

    public CartScreenAdapter(Context context, List<OrderModel> order, boolean flg,boolean
chk)
    {
        this.context = context;
        this.order=order;
        flag=flg;
        check=chk;
        db = FirebaseFirestore.getInstance();
    }

    public CartScreenAdapter(Context context, List<ItemsModel> data) {
        this.context = context;
        this.data=data;
        flag=false;
        db = FirebaseFirestore.getInstance();
    }
    @Override
    public CartScreenAdapter.MyViewHolder onCreateViewHolder(ViewGroup parent,
int viewType) {

        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.
```

```
cartscreenitem, parent, false);

        CartScreenAdapter.MyViewHolder vh = new CartScreenAdapter.MyViewHolder(v);
        return vh;
    }

    @Override
    public void onBindViewHolder(CartScreenAdapter.MyViewHolder holder, final int
 position) {

        if(flag){
            holder.status.setVisibility(View.VISIBLE);
            Glide.with(context).load(order.get(position).getImgurl()).into(holder.
imageView);
            holder.name.setText(order.get(position).getName());
            holder.price.setText(order.get(position).getPrice()+" Rs ");
            holder.quantity.setText(order.get(position).getQuantity()+" item(s) ");
            holder.status.setText(order.get(position).getStatus());
            if(order.get(position).getStatus().equals("Pending"))
                holder.cancel.setVisibility(View.VISIBLE);
            else
                holder.cancel.setVisibility(View.GONE);

            if(check){
                if(order.get(position).getStatus().equals("Pending")) {
                    holder.cancel.setVisibility(View.VISIBLE);
                    holder.accept.setVisibility(View.VISIBLE);
                }
                else
                {
                    holder.cancel.setVisibility(View.GONE);
                    holder.accept.setVisibility(View.GONE);
                }
                holder.cancel.setText("Reject");
                holder.cancel.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        db.collection("Orders").document(order.get(position).getId())
                                .update("Status","Reject").addOnSuccessListener(new
OnSuccessListener<Void>() {

                            @Override

                            public void onSuccess(Void aVoid) {
                                order.get(position).setStatus("Reject");
                                notifyDataSetChanged();
                            }
                        });

                    }
                });
                holder.accept.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Map<String,Object> user = new HashMap<>();
```

```
                        user.put("Status","Accept");

                        db.collection("Orders").document(order.get(position).getId())
                                .update("Status","Accept").addOnSuccessListener(new
OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                order.get(position).setStatus("Accept");

ReduceQuantity(order.get(position).getQuantity(),order.get(position).getPID(),order.get
(position).getId());
                            }
                        });
                    }
                });
            }
            else {
                holder.cancel.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {

db.collection("Orders").document(order.get(position).getId()).delete().addOnSuccess
Listener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                order.remove(position);
                                notifyDataSetChanged();

                            }
                        }).addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Toast.makeText(context, "Try Again",
Toast.LENGTH_SHORT).show();
                            }
                        });
                    }
                });
            }

        }
        else{

Glide.with(context).load(data.get(position).getImageurl()).into(holder.imageView);
            holder.name.setText(data.get(position).getName());
            holder.price.setText(data.get(position).getPrice()+" Rs ");
            holder.quantity.setText(data.get(position).getQuantity()+" item(s) ");
            holder.plus.setVisibility(View.VISIBLE);
            holder.minus.setVisibility(View.VISIBLE);
            holder.plus.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Log.e("kkk",data.get(position).getService());

if(data.get(position).getQuantity()<Long.parseLong(data.get(position).getService())){
```

```
//data.get(position).setPrice(data.get(position).getPrice()+(data.get(position)
.getPrice()/data.get(position).getQuantity()));
                        //data.get(position).setQuantity(data.get(position).
getQuantity()+1);


constant.cartItems.get(position).setPrice(data.get(position).getPrice()+
(data.get(position).getPrice()/data.get(position).getQuantity()));

constant.cartItems.get(position).setQuantity(data.get(position).getQuantity()+1);
                        notifyDataSetChanged();
                    }
                    else {
                        Toast.makeText(context,"Out of Stock",Toast.LENGTH_SHORT).show();
                    }

                }
            });
            holder.minus.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if(data.get(position).getQuantity()>1){
                       // data.get(position).setPrice(data.get(position).getPrice()-
(data.get(position).getPrice()/data.get(position).getQuantity()));
                      //  data.get(position).setQuantity(data.get(position).getQuantity()-1);


constant.cartItems.get(position).setPrice(data.get(position).getPrice()-
(data.get(position).getPrice()/data.get(position).getQuantity()));

constant.cartItems.get(position).setQuantity(data.get(position).getQuantity()-1);
                    }
                    else {
                       // data.remove(position);
                        constant.cartItems.remove(position);
                    }

                    notifyDataSetChanged();
                }
            });
        }

    }

    private void ReduceQuantity(final Long quantity, final String id,final String oid) {
        db.collection("Products").document(constant.Email).collection("Details")
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            for (QueryDocumentSnapshot document : task.getResult()) {
                                if(document.getId().equals(id)) {
                                    if (document.getLong("Quantity") >= quantity) {
```

```
                                    db.collection("Products").
                                            document(constant.Email).
                                            collection("Details")
                                            .document(id).
                                            update("Quantity", document.getLong
("Quantity") - quantity)
                                           .addOnSuccessListener(new
OnSuccessListener<Void>() {
                                                @Override
                                                public void onSuccess(Void aVoid) {
                                                    notifyDataSetChanged();
                                                }
                                            }).addOnFailureListener(new OnFailureListener()
 {
                                        @Override
                                        public void onFailure(@NonNull Exception e) {
                                            Toast.makeText(context, e.getMessage(),
Toast.LENGTH_SHORT).show();
                                        }
                                    });
                                }
                                    else {
                                        Toast.makeText(context, "Out Of Stock",
Toast.LENGTH_SHORT).show();
                                        db.collection("Orders").document(oid)

.update("Status","Reject").addOnSuccessListener(new OnSuccessListener<Void>() {
                                            @Override
                                            public void onSuccess(Void aVoid) {


                                            }
                                        });
                                    }
                                }                               }
  @Override
    public int getItemCount() {
        if(flag)
            return order.size();
        else
            return data.size();

    }
    public class MyViewHolder extends RecyclerView.ViewHolder {
        ImageView imageView,plus,minus; Button cancel,accept;
        TextView name,price,quantity,status;
        public MyViewHolder(View itemView) {
            super(itemView);
            imageView= itemView.findViewById(R.id.img);
            name= itemView.findViewById(R.id.name);
            price= itemView.findViewById(R.id.price);
            quantity= itemView.findViewById(R.id.quantity);
            status=itemView.findViewById(R.id.status);
            cancel=itemView.findViewById(R.id.cancel);
```

```
            accept=itemView.findViewById(R.id.accept);
            plus=itemView.findViewById(R.id.plus);
            minus=itemView.findViewById(R.id.minus);
        }
    }
}
```

Coding E-Commerce Management: Home Adapter

This code is blogging to home page of the products

```
package com.example.ecommerce.buyer.adapter;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.ecommerce.R;
import com.example.ecommerce.buyer.activity.Details;
import com.example.ecommerce.seller.model.ItemsModel;

import java.util.List;

public class HomeAdapter extends RecyclerView.Adapter<HomeAdapter.MyViewHolder> {
    List<ItemsModel> data;
    Context context;
    public HomeAdapter(Context context, List<ItemsModel> data) {
        this.context = context;
        this.data=data;
    }
    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.homeitems,

 parent, false);

        MyViewHolder vh = new MyViewHolder(v);
        return vh;
    }

    @Override
    public void onBindViewHolder(holder.pmd.svs holder, final int position) {
        Glide.with(context).load(data.get(position).getImageurl()).into(holder.
imageView);
        holder.name.setText(data.get(position).getName()+"\n"+data.get(position).
```

```
getSpecification());
        holder.price.setText(data.get(position).getPrice()+" Rs ");
        holder.imageView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                context.startActivity(new Intent(context,
Details.class).putExtra("details",data.get(position)));

            }
        });
    }


    @Override
    public int getItemCount() {
        return data.size();
    }
    public class MyViewHolder RecyclerView.ViewHolder {
        ImageView imageView;
        TextView name,price;


        public MyViewHolder(View itemView) {

            super(itemView);
            imageView= itemView.findViewById(R.id.image);
            name= itemView.findViewById(R.id.name);
            price= itemView.findViewById(R.id.price);     }
   }
}
public class MyViewHolder extends RecyclerView.ViewHolder {
        ImageView imageView,plus,minus; Button cancel,accept;
        TextView name,price,quantity,status;
        public MyViewHolder(View itemView) {
            super(itemView);
            imageView= itemView.findViewById(R.id.img);
            name= itemView.findViewById(R.id.name);
            price= itemView.findViewById(R.id.price);
            quantity= itemView.findViewById(R.id.quantity);
            status=itemView.findViewById(R.id.status);
            cancel=itemView.findViewById(R.id.cancel);
            accept=itemView.findViewById(R.id.accept);
            plus=itemView.findViewById(R.id.plus);
            minus=itemView.findViewById(R.id.minus);
cancel=itemView.findViewById(R.id.cancel);
            accept=itemView.findViewById(R.id.accept);
            plus=itemView.findViewById(R.id.plus);
            minus=itemView.findViewById(R.id.minus);
  name= itemView.findViewById(R.id.name);
            price= itemView.findViewById(R.id.price);
            quantity= itemView.findViewById(R.id.quantity);
            accept=itemView.findViewById(R.id.accept);
            plus=itemView.findViewById(R.id.plus);
            minus=itemView.findViewById(R.id.minus);
```

```
        }
        }
    }
}
    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.homeitems,

 parent, false);

        MyViewHolder vh = new MyViewHolder(v);
        return vh;
    }
```

Chapter 5

Software Testing

Software Testing is the most crucial part of Software Development Process. It is the investigation or evaluation of a software component, improving them, and finding bugs and defects. Testing is usually done by executing a system in such a way that it identifies any gaps, errors, or missing requirements in contrary to the actual requirements. This chapter provides a description about the adopted testing procedure. This includes the adopted testing methodology, test suite and the test results of the developed software.

5.1. Testing Methodology

We have used black box testing in our testing phase. Black box contains certain benefits that include following:

- Black box testing examines the functionality of application without peering into its internal structure.
- This method can be applied to every level of the software testing which include unit, integration, system and acceptance.

Black Box unit testing is used in our project. Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules, together with associated control data, usage procedures and operating procedures are tested to determine whether they are fit for purpose. The goal of utilizing numerous testing methodologies in your development process is to make sure your software can successfully operate in multiple environments and across different platforms. These can typically be broken down between functional and non-functional testing. Functional testing involves testing the application against the business requirements. It incorporates all test types designed to guarantee each part of a piece of software behaves as expected by using uses cases provided by the design team or business analyst

5.2. Test Cases

A test case is a specification of the inputs, execution conditions, testing procedure and expected results that define a single test to be executed to achieve a particular software testing objective.

1. Admin Registration

Date: 09/01/2021	**Tested By: Muhammad Hasan**
System: Smart Farm	Environment: Android App
Objective: **Admin registration**	Test ID: 1
Version: 1.1	Testing Type: Unit

Input: Name**:** abc email: **ab@gmail.com** Password: 1234567 Phone: 1234567890
Expected Result: registration Successful
Actual Result: Admin Successfully register into system

Table 5.1: Test Case Registration

2. Customer Registration

Date: 17/07/2020	**Tested By: Muhammad Hasan**
System: Smart Farm	Environment: Android App
Objective: **Customer registration**	Test ID: 12
Version: 1.1	Testing Type: Unit
Input: Name**:** abc email: **ab@gmail.com** Password: 1234567 Phone: 1234567890	
Expected Result: registration Successful	
Actual Result: Customer Successfully register into system	

Table 5.2: Test Case Registration

3. Admin Login

Date: 09/01/2021	**Tested By: Muhammad Hasan**
System: Smart Farm	Environment: Android App
Objective: **Admin login**	Test ID: 3
Version: 1.1	Testing Type: Unit

Input: Name**:** xyz email: **xyz@gmail.com** Password: 1234567 Phone: 1234567890
Expected Result: Login Successful
Actual Result: Admin Successfully login into system

Table 5.3: Test Case Registration

4. Customer Login

Date: 09/01/2021	**Tested By: Muhammad Hasan**
System: Smart Farm	Environment: Android App
Objective: C**ustomer login**	Test ID: 4
Version: 1.1	Testing Type: Unit
Input: email: **xyz@gmail.com** Password: 1234567	
Expected Result: Login Successful	
Actual Result: Admin Successfully register into system	

Table 5.4: Test Case Registration

5. Customer Order

Date: 09/01/2021	**Tested By: Muhammad Hasan**
System: Smart Farm	Environment: Android App
Objective: **Place Order**	Test ID: 5
Version: 1.1	Testing Type: Unit
How Test Case was Generated: Customer clicks on the product and add it by quantity of item and checkout it by adding on cart.	
Expected Result: Order Added Successfully	
Actual Result: Message Display: Order Added Successfully	

Table 5.5: Test Case Registration

6. Admin Order

Date: 09/01/2021	**Tested By: Muhammad Hasan**
System: Smart Farm	Environment: Android App
Objective: **Place Order**	Test ID: 6
Version: 1.1	Testing Type: Unit
How Test Case was Generated: Admin clicks on the order which comes from customer side, admin can accept or reject the order.	
Expected Result: Accept and Reject Successfully	
Actual Result: Message Display: Accept and Reject Successfully	

Table 5.5: Test Case Registration

7. Manage Products

Date: 09/01/2021	**Tested By: Muhammad Hasan**
System: Smart Farm	Environment: Android App
Objective: **Place Order**	Test ID: 7
Version: 1.1	Testing Type: Unit
Name: Apple Specification: Gaja Apple Price: 120 kg	
Expected Result: Product added Successfully	
Actual Result: Product added Successfully	

Table 5.5: Test Case Registration

Chapter 6
Software Deployment

6.1. Deployment Process Description

For the deployment, we will provide user with two apk's of android applications. In these apk's file we give the different permission from user to successfully run app on his device. Firebase Database connectivity used in our project for data storage of admin, customer and seller. We will provide apk's of three applications one of which will be Seller Transection app through this application Admin can trace location of customer and Seller boy can get order details of the customer. The other mobile application will be of customer through that a user can order from the customer.

Chapter 7
Project Evaluation

7.1. Project Evaluation Report

Examiner Name:	
S. No	**Suggestions**

Other Comments (If any):

_____

_____

_____

_____

_____

Signature