A Project Report on

IOT Based Smart Grocery Store

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Information Technology

by

Samiksha Mhatre(16104025)

Under the Guidance of

Prof. Yaminee Patil
Prof. Kiran Deshpande

Department of Information Technology
NBA Accredited
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2021-2022

Approval Sheet

This Project Report entitled ***"IOT Based Smart Grocery Store"*** Submitted by ***"Samiksha Mhatre"(16104025)*** is approved for the partial fulfillment of the requirement for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

(Prof. Yaminee Patil)
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"IOT Based Smart Grocery Store"*** submitted by ***"Samiksha Mhatre"(16104025)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology***,to the University of Mumbai,is a bonafide work carried out during academic year 2020-2021.

(Prof. Yaminee Patil)
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Acknowledgement

We have great pleasure in presenting the report on **IOT Based Smart Grocery Store.** We take this opportunity to express our sincere thanks towards our guide **Prof. Yaminee Patil** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Kiran B. Deshpande** Head of Department,IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Vishal S. Badgujar** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Name of Student1: Samiksha Mhatre
Student ID1: 16104025

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

(Samiksha Mhatre 16104025)

Date:

Abstract

This technology provides a way for reducing the amount of time spent shopping in supermarkets. At the billing counter, customers may encounter a variety of issues, such as waiting and not knowing if they have enough money to pay for the things they have purchased. The billing procedure at the counter is time intensive, and the billing area requires more human resources. To address this issue, we presented a method that uses a IoT Based Smart Grocery Store App to circumvent these issues. The app can be easily used on the mobile phone. The mobile camera via the smart trolley app can be used for scanning items, displaying product information, pricing, and total bill. The user may pay the bill using any of the online payment methods available, through various UPI apps or via net banking. This method improves the purchasing experience for the customer while shortening the shopping time.

Keywords: Time, Billing, Shopping, Mobile, Online Payment.

Contents

List of Figures

List of Abbreviations

HTML:	Hyper-text Markup Language
iOS:	I-phone Operation System
IP:	Internet Protocol
IOT:	Internet of Things
RFID:	Radio Frequency Identification
SDK:	Software Development Kit
SQL:	Structured Query Language
UI:	User Interface

Chapter 1

Introduction

Supermarkets and retail malls are springing up in ever-increasing numbers these days. Due to cheaper prices, a wider choice of product availability, and better service, individuals are shifting their shopping habits from traditional neighborhood general shops to supermarkets. As a result, we are seeing an increase in the number of individuals who buy at supermarkets and shopping malls. However, while shopping, everyone wastes the most of their time waiting in lines at the billing counters. Customers waste a lot of time waiting in line for checkout, especially at busy hours of the day, weekends, and during festivals.

The project's major purpose is to cut down on total shopping time. The IoT Based Smart Grocery Store initiative intends to do this by allowing consumers to complete the paying process on their mobile device while shopping using the IoT Based Smart Grocery Store app. Shopping malls may lower the number of people working at billing counters and the amount of space used, which saves time and money. These efforts and investments may be put to bettering the quality and experience consumers.

The main goal of our application is for supermarkets and retail malls are springing up in ever-increasing numbers these days. Due to cheaper prices, a wider choice of product availability, and better service, individuals are shifting their shopping habits from traditional neighborhood general shops to supermarkets.

As a result, we are seeing an increase in the number of individuals who buy at supermarkets and shopping malls. However, while shopping, everyone wastes the most of their time waiting in lines at the billing counters. Customers waste a lot of time waiting in line for checkout, especially at busy hours of the day, weekends, and during festivals. The project's major purpose is to cut down on total shopping time.

The IoT Based Smart Grocery Store initiative intends to do this by allowing consumers to complete the paying process on their mobile device while shopping using the Smart Trolley app. Shopping malls may lower the number of people working at billing counters and the amount of space used, which saves time and money. These efforts and investments may be put to bettering the quality and experience of consumers.

Chapter 2

Literature Review

2.1 Smart Trolley with Advance Billing System:

The author explains a smart trolley with advanced billing system using RFID technology and a technology oriented, low-cost, easily scalable, and rugged system for aiding shopping in person. Author has developed system comprises of a Server unit (SU), a User Interface unit (UIU), an in-built Billing Unit (BU) and Central unit (CU).SU will help in establishing and maintaining the connection of the shopping cart with the main server.

2.2 RFID Based Smart Trolley for Automatic Billing System:

The author has explained a RFID based smart trolley for automatic billing system using a RFID reader, where the wireless communication is established using HC-12 communication module between the trolley and central PC. Here Author has used a pre-charged RFID card for the payment purpose.

2.3 Smart Shopping Cart with Automated Billing System:

The author aims to reduce the time of billing for the customers and to ease the process of shopping so that the customers gets benefited. It can be implemented in shopping malls where there is a large crowd and huge rush into malls. In the world of Automation. This technology will replace the present barcode system which is present being followed. Hence this technology can help people to make their shopping easy and time saving too without any much human intervention. This also reduces manpower and shopping mall maintenance.

2.4 Smart Electronic Trolley for Shopping Mall:

The innovative project idea can be used is places like shopping complexes, supermarkets malls to purchase the products. Here RFID card is used to securely access every product in shopping places. If a product is scanned put into the cart, all the required details of the product will be displayed on the LCD screen. Therefore, an RFID tag/card is used for accessing the products. Hence this project will help in improving the security also the shopping time can be reduced. It also provides an enjoyable user-friendly shopping experience to the customers.

Chapter 3

Objectives

• To minimize the consumer's overall shopping time.

• To design a system that is simple to use, customer-centric, and shortens the checkout process.

• To provide customers with an organized list of the products in their cart as well as the overall payment.

• To minimize the workforce and increase the availability of space at the billing counters.

• To improve the customer service and shopping experience.

Chapter 4

Project Design

4.1 Proposed System Architecture

4.1.1 User

The mobile client is designed based on the smartphone of the Android platform, so it requires a built-in Wi-Fi module and GPA module in the smartphone, for self-positioning and communication with the backend server. Because the user scale of Android smartphones is quite large, we have considered the user-friendly design as much as possible. The mobile client has a simple and friendly interface, providing fast query operation. Just by the user clicking on the corresponding request or scanning the QR code, the mobile client would feedback the results to the user interface. The mobile client is the interaction platform between tourists and the scenic and is also the important channel to obtain the attraction's information and service information. Therefore, the mobile client has a direct impact on the tourism experience of tourists. The functions of our mobile client mainly include self-positioning, network traffic statistics, inquiry of scenic information and surrounding information, scenic route recommendation, and attractions multimedia presentations.

4.1.2 Application

Flutter is an open source framework to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient and easy to understand SDK to write mobile application in Google's own language, Dart. In general, developing a mobile application is a complex and challenging task. There are many frameworks available to develop a mobile application. Android provides a native framework based on Java language and iOS provides a native framework based on Objective-C / Shift language. However, to develop an application supporting both the OSs, we need to code in two different languages using two different frameworks. To help overcome this complexity, there exists mobile frameworks supporting both OS. These frameworks range from simple. HTML based hybrid mobile application framework (which uses HTML for User Interface and JavaScript for application logic) to complex language specific framework (which do the heavy lifting of converting code to native code). Irrespective of their simplicity or complexity, these frameworks always have many disadvantages, one of the main drawback being their slow performance. In this scenario, Flutter – a simple and high performance

framework based on Dart language, provides high performance by rendering the UI directly in the operating system's canvas rather than through native framework.

4.1.3 Database

A NoSQL (originally referring to "non-SQL" or "non-relational") database provides a mechanism for storage and retrieval of data that is modelled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but the name "NoSQL" was only coined in the early 21st century, triggered by the needs of Web 2.0 companies. NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also sometimes called not only SQL to emphasize that they may support SQL-like query languages or sit alongside SQL databases in polyglot-persistent architectures.

4.1.4 Node ESP32-CAM

This module is implemented to scan the barcode implemented on the product, and to store it on the database.

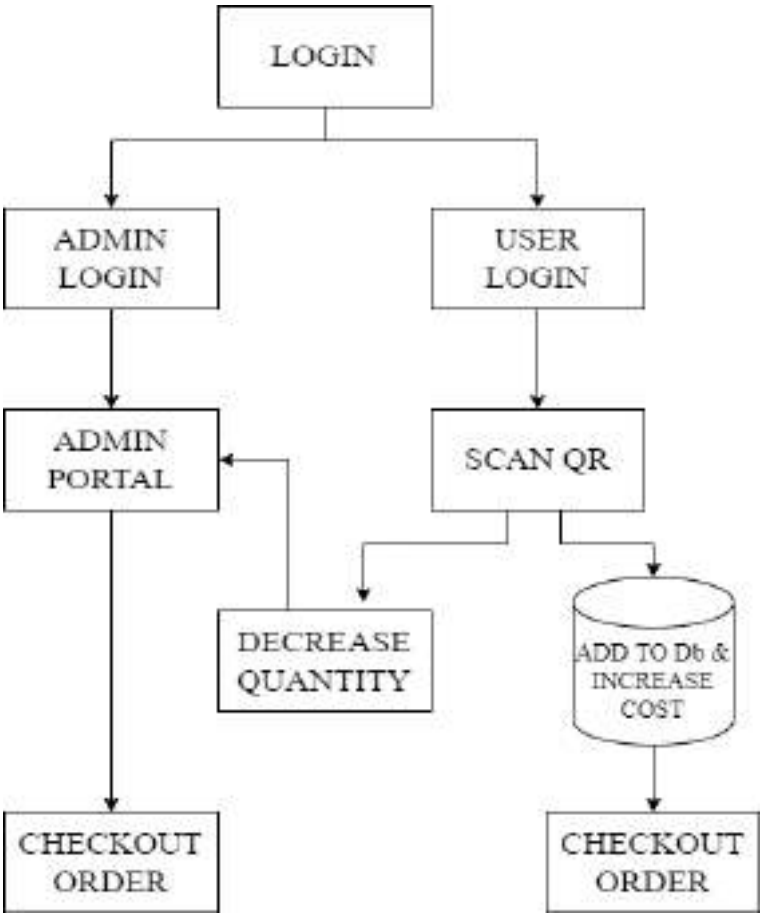


Figure 4.1: Proposed System Architecture

4.2 UML Diagrams

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

4.2.1 Use Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

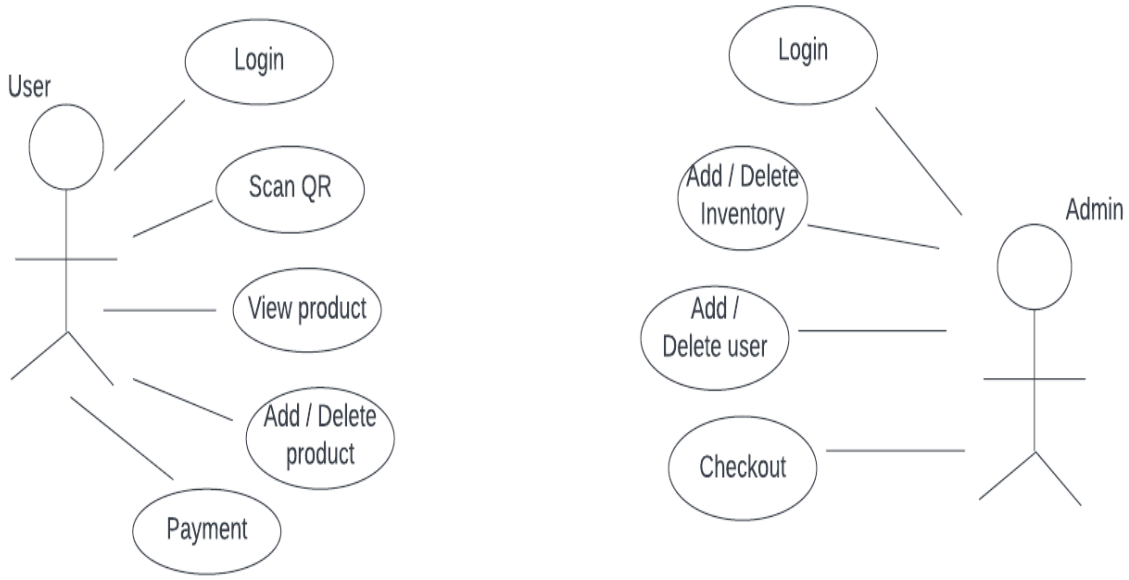


Figure 4.2: Use Case Diagram

4.2.2 Activity Diagram

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

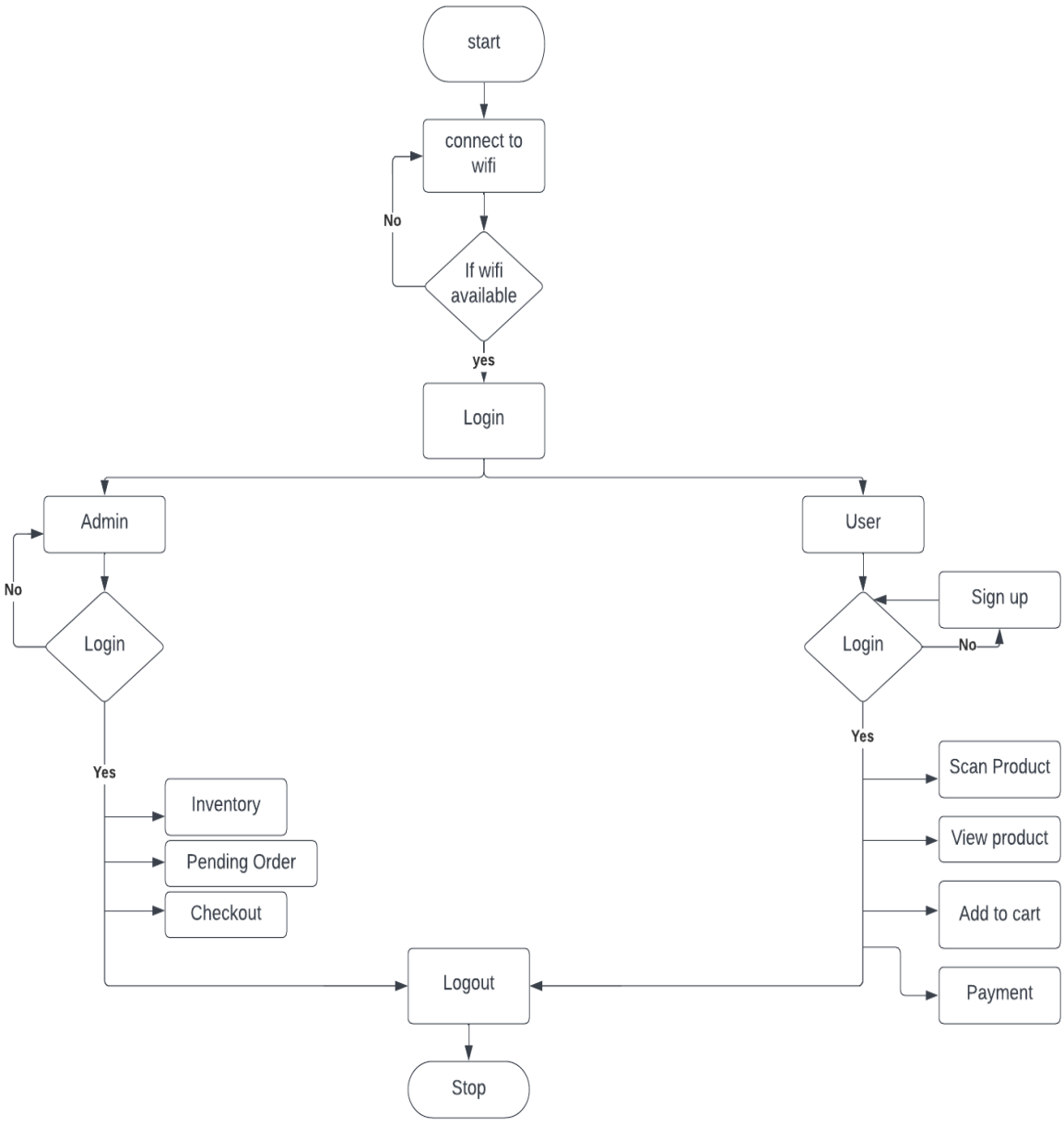


Figure 4.3: Activity Diagram

4.2.3 Sequence Diagram

A sequence diagram or system sequence diagram (SSD) shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

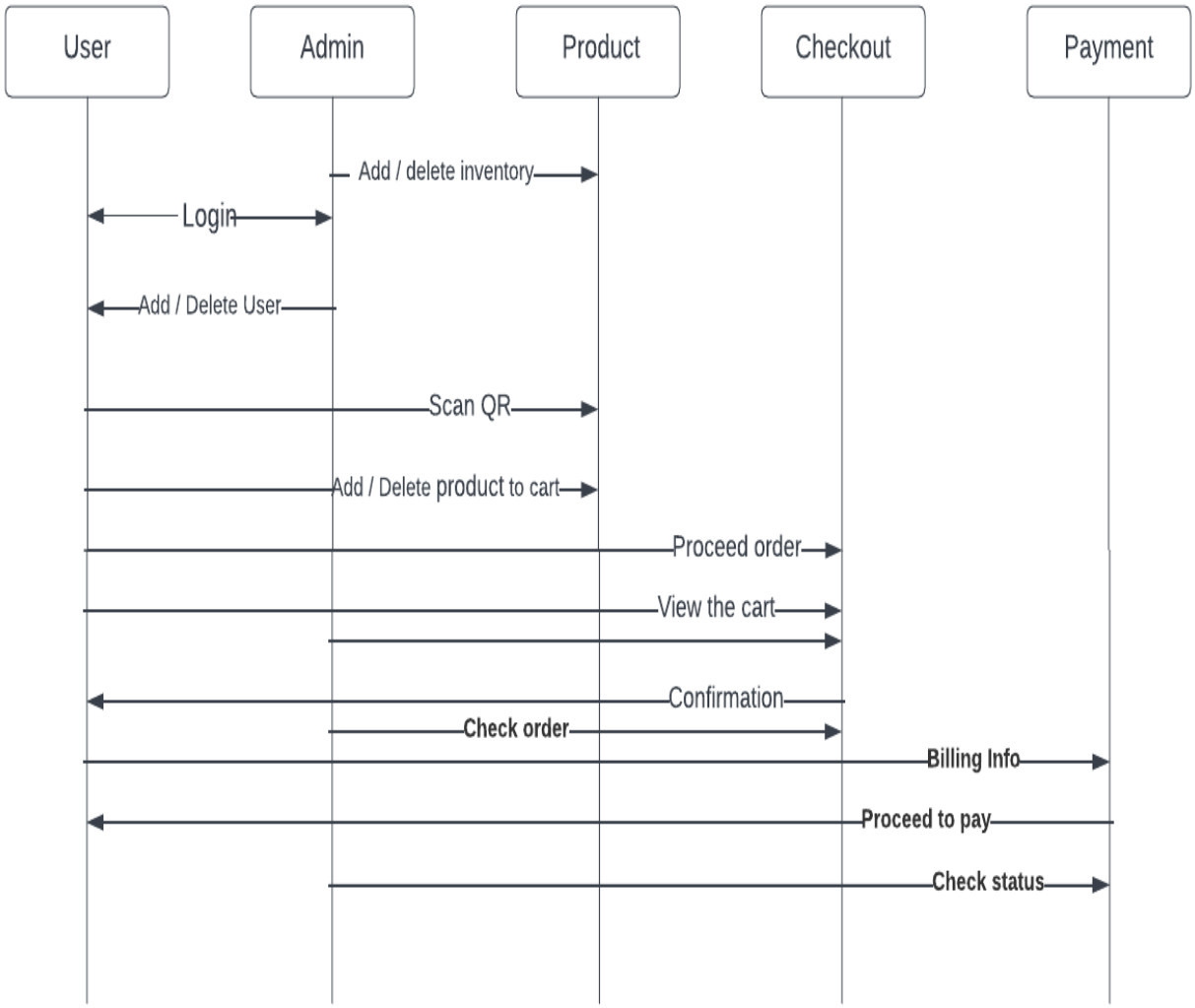


Figure 4.4: Sequence Diagram

Chapter 5

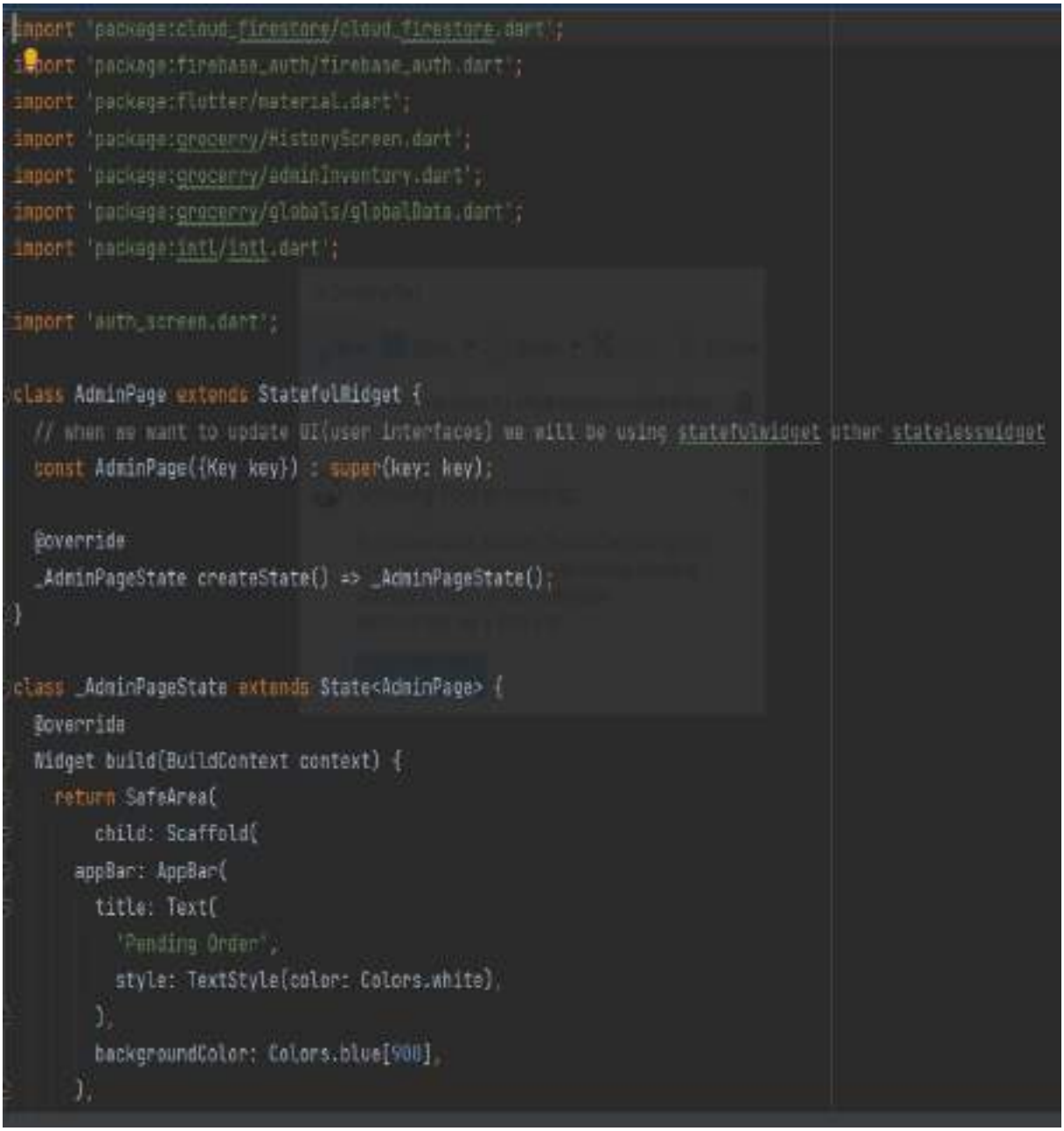
Project Implementation

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'admin_page.dart';
import 'auth_screen.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/services.dart';

import 'home_page.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(); //initailizing firebase
  SystemChrome.setPreferredOrientations([
    // adjusting device in portrait mode only
    DeviceOrientation.portraitUp,
    DeviceOrientation.portraitDown,
  ]);
  runApp(const MyApp()); // main function which run our app
}

class MyApp extends StatelessWidget {
  const MyApp({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner:
          false, //disbaling debug mode banner ,by default true
      title: 'Grocery Store', // name of application
```

Figure 5.1: Main Dart 1

Figure 5.2: Main Dart 2

The above picture depicts the name of the application to be displayed with the default colour of the application. The authStateChanges() will call when the user/customer signup,signin or signout. The if else loop used checks the connection state of the application where the circular progress indicator that means it shows the loading sign while the data is being fetched from the database if the condition is untrue then it will render that we don't have any data returns to check if the email credentials are of admin or not.



```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:grocerry/HistoryScreen.dart';
import 'package:grocerry/adminInventory.dart';
import 'package:grocerry/globals/globalData.dart';
import 'package:intl/intl.dart';

import 'auth_screen.dart';

class AdminPage extends StatefulWidget {
  // when we want to update UI(user interfaces) we will be using statefulwidget other statelesswidget
  const AdminPage({Key key}) : super(key: key);

  @override
  _AdminPageState createState() => _AdminPageState();
}

class _AdminPageState extends State<AdminPage> {
  @override
  Widget build(BuildContext context) {
    return SafeArea(
        child: Scaffold(
      appBar: AppBar(
        title: Text(
          'Pending Order',
          style: TextStyle(color: Colors.white),
        ),
        backgroundColor: Colors.blue[900],
      ),
```

Figure 5.3: Admin Page 1

The above figure shows the imported libraries of flutter,firebase_authentication,global data.When we want to update the user interface the StatefulWidget and StatelessWidget are used.This page shows the Pending Order, Admin Inventory Page, All orders of the users/customer, History of the users, checkout button for the users to approve the orders

Figure 5.4: Admin Page 2

The above figure shows the instances of firebase collecting 'All Orders' of the users displayed on the admin panel. Also the summary of the order consisting the total price ,total quantity of the order with the respective email of the user using DD-MM-YYYY format.Snackbar providing colour to the checkout button in green and is displayed to the admin as 'Order Checked Out'.

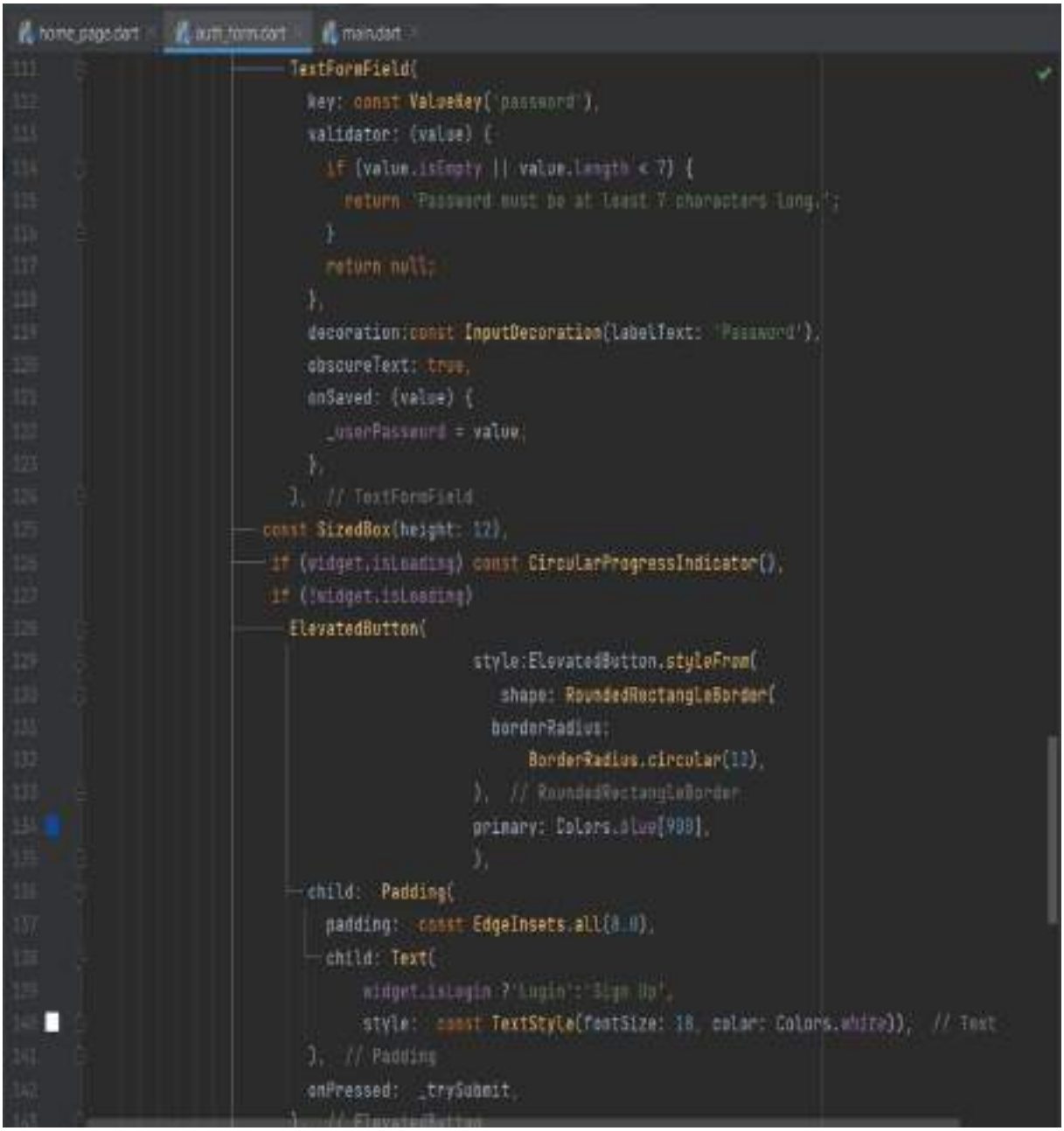


Figure 5.5: Authentication Page

Authentication Form : This form includes the authentication part of the login page including the Login and signup and admin authentication. Length of the passwords is set ¡ 7 ,text padding setting the font size and colour followed by the evaluation button

```
validate(context) {
  if (cardNo.isEmpty ||
      expiryDate.isEmpty ||
      cvv.isEmpty ||
      userName.isEmpty) {
    showSnackBar(context, Colors.red, 'Please fill all the details');
  } else {
    RegExp expiryDataRegex = RegExp(r'^\d{2}\/\d{2}$');

    if (cardNo == '5123456789123456' &&
        expiryDataRegex.hasMatch(expiryDate)) {
      Timer timer =
          Timer(Duration(minutes: 1, seconds: 30), (() => backFlow(context)));

      // payment successful
      showDialog(
          barrierDismissible: false,
          context: context,
          builder: (_) {
            return WillPopScope(
              onWillPop: () => null,
              child: AlertDialog(
                insetPadding: EdgeInsets.symmetric(horizontal: 30),
                contentPadding: EdgeInsets.all(20),
                title:
                    Text('Payment Successful', textAlign: TextAlign.center),
                content: Column(
                  mainAxisSize: MainAxisSize.min,
                  children: [
                    Image(
```

Figure 5.6: Payment Page

Payment : This showcases the page where the card details are to be entered where the total number of the card are set to be of random 16 digits which matches the expiry date with the duration of expiry set to 1 minute and 30 seconds and payment successful is displayed to the user if the given criteria are met

Figure 5.7: Esp32 Camera on Arduino

```
from unittest import result
import cv2      # used for camera recognitations
import numpy as np  #dependencies
import pyzbar.pyzbar as pyzbar
import urllib.request

from firebase import firebase
url = "https://grocery-store-5615b-default-rtdb.firebaseio.com/"  # url firebase
firebase = firebase.FirebaseApplication(url)              # it'll take to url argument

font = cv2.FONT_HERSHEY_PLAIN                             #font recog

url='http://192.168.0.105/' #replace with your ip address / is imp here
cv2.namedWindow("live transmission", cv2.WINDOW_AUTOSIZE)       # names window n open  live window

prev=""
pres=""
while True:
    img_resp=urllib.request.urlopen(url+'jpg')   # url is added in jpg
    imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8) #lumpy lib used for array ops
    frame=cv2.imdecode(imgnp,-1)   # bytes a flag value image read

    decodedObjects = pyzbar.decode(frame)  # pyzbar qr code bar code scann lib - array data stored
    for obj in decodedObjects:
        pres=obj.data # data that will be sent

        if prev == pres:
            pass
        else:
            print("Type:",obj.type)
            print("Data: ",obj.data) #inheritance concept

            qr = {
                'QR-CODE' : str(obj.data) # firebase data is stored in JSON
            }
            result = firebase.post('https://grocery-store-5615b-default-rtdb.firebaseio.com/QR',qr) # firebase n serial monitor data sent
            print(result)
```

Figure 5.8: Esp32 Camera

These both are (1) ESP32 Cam scanner and the server conenction.
In the above pictures Wifi, webserver, WiFi client, Esp32 libraries are imported camera model is selected while it is recognise in the server as cv2 followed by the imported dependencies. WiFi name and password are defined which can be changed as per the users available network in the Backend the IP address of the device should be connected to the url with '/' which is the most important step. Live transmission windows are named and executed for the camera to capture the QR data. The url consisting of the the IP address ends with /jpg send the data in form of images (pixels) stored in form of array using the lummpy library for operations.Pyzbar QR code scan the stored array data pres=obj.data represents the data that will be sent.

Chapter 6

Testing

6.1 Unit testing:

Before you can test an entire software program, make sure the individual parts work properly on their own. Unit testing validates the function of a unit, ensuring that the inputs (one to a few) result in the lone desired output. This testing type provides the foundation for more complex integrated software. When done right, unit testing drives higher quality application code and speeds up the development process. Developers often execute unit tests through test automation.

6.2 Component testing:

Also called module testing, component testing checks individual parts of an application. Similar to unit testing, component testing assesses a part of the software in isolation from the broader system. The difference between unit testing and component testing is that the former is done by developers in a white-box format to verify that program modules execute, while the latter is done by testers in a black-box format to validate individual objects or parts of the software. If other software components rely on the component under test, the QA professional might use a stub and driver to simulate interactions between those dependent components.

6.3 System testing:

With system testing, QA professionals test the software in its entirety, as a complete product. With this type of functional testing, testers validate the complete and integrated software package to make sure it meets requirements. Where necessary, testers can provide feedback on the functionality and performance of the app or website without prior knowledge of how it was programmed. This helps teams develop test cases to be used moving forward. System testing is also referred to as end-to-end testing.

Chapter 7

Result

Figure 7.1: Login

The above picture is the front end main login page of the IOT based Grocery Store Application which consists of the Login page. The login gives access to the admin panel and the customer panel. Only admin can login with the credentials to connect to the server to fetch the admin database. Authentication of credentials will get the user logged into the ADMIN page. A new user has to sign up with the required mandatory details. Once the user signs up he/she can proceed to the further functionality of the application. An existing user can login with the credentials to which post authentication the user can have access to the online cart.

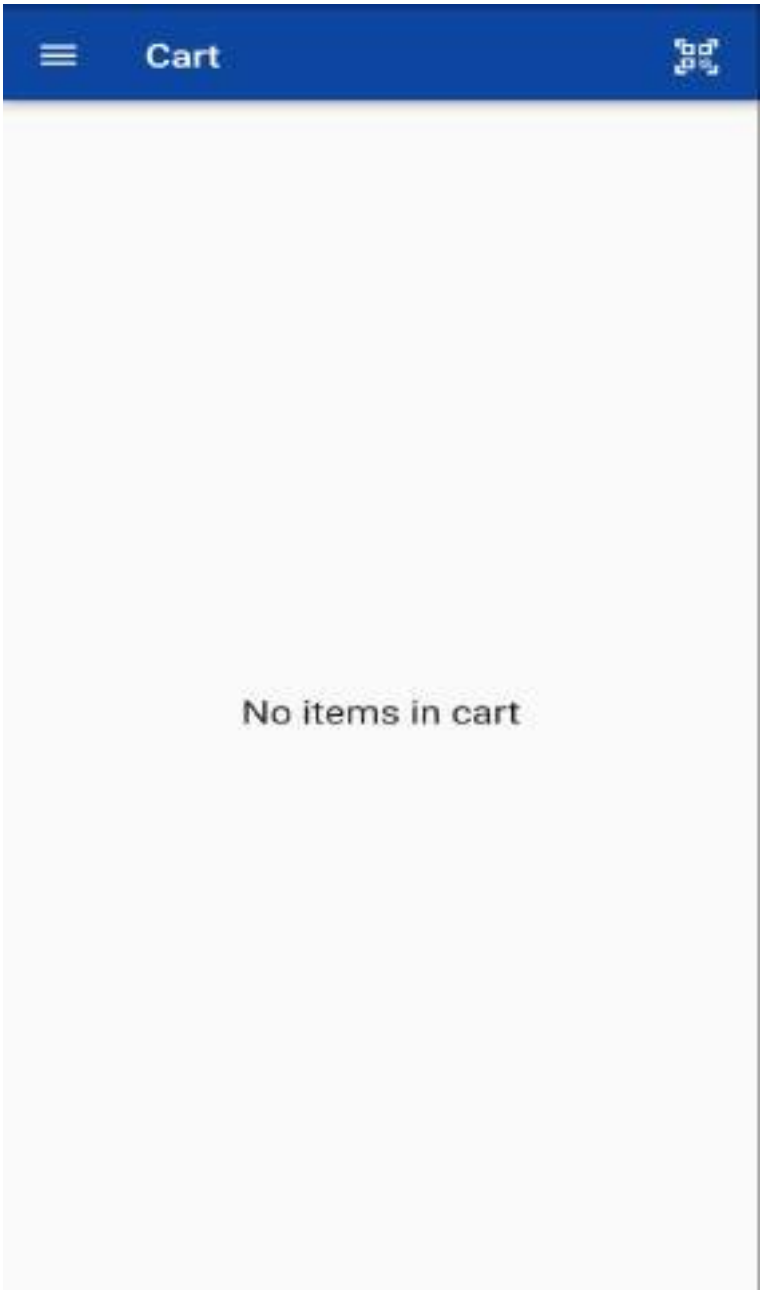


Figure 7.2: Empty Cart

Once the input credentials are authenticated while the internet connection is stable the initial page will look like this where the scanned product will be displayed

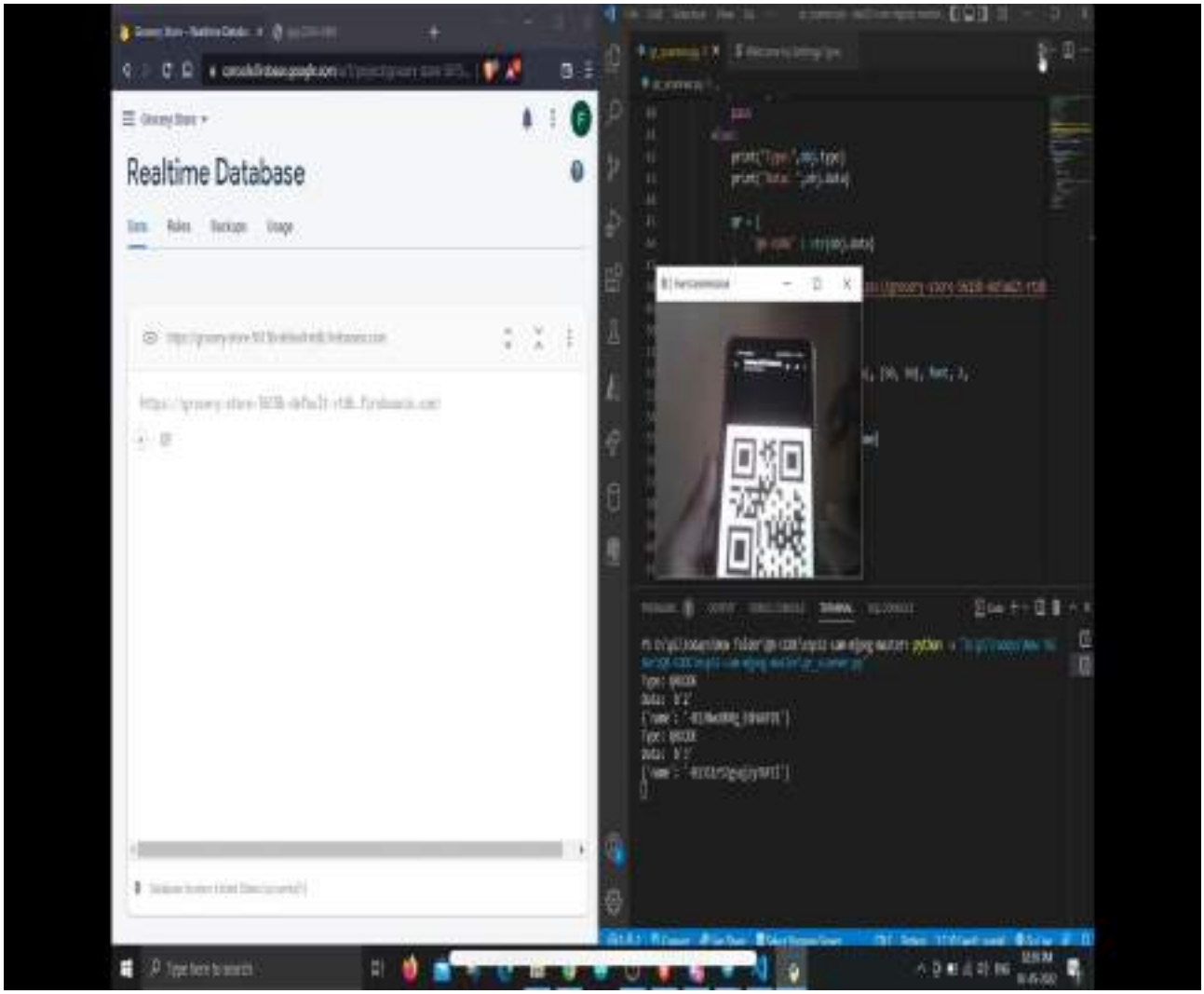


Figure 7.3: QR Scanning using Esp32

In the above shown picture is where the live transmission of data takes place through Esp32 cam and real time data is sent and stored in the database in form of array. QR code of the products are scanned and are sent to the database

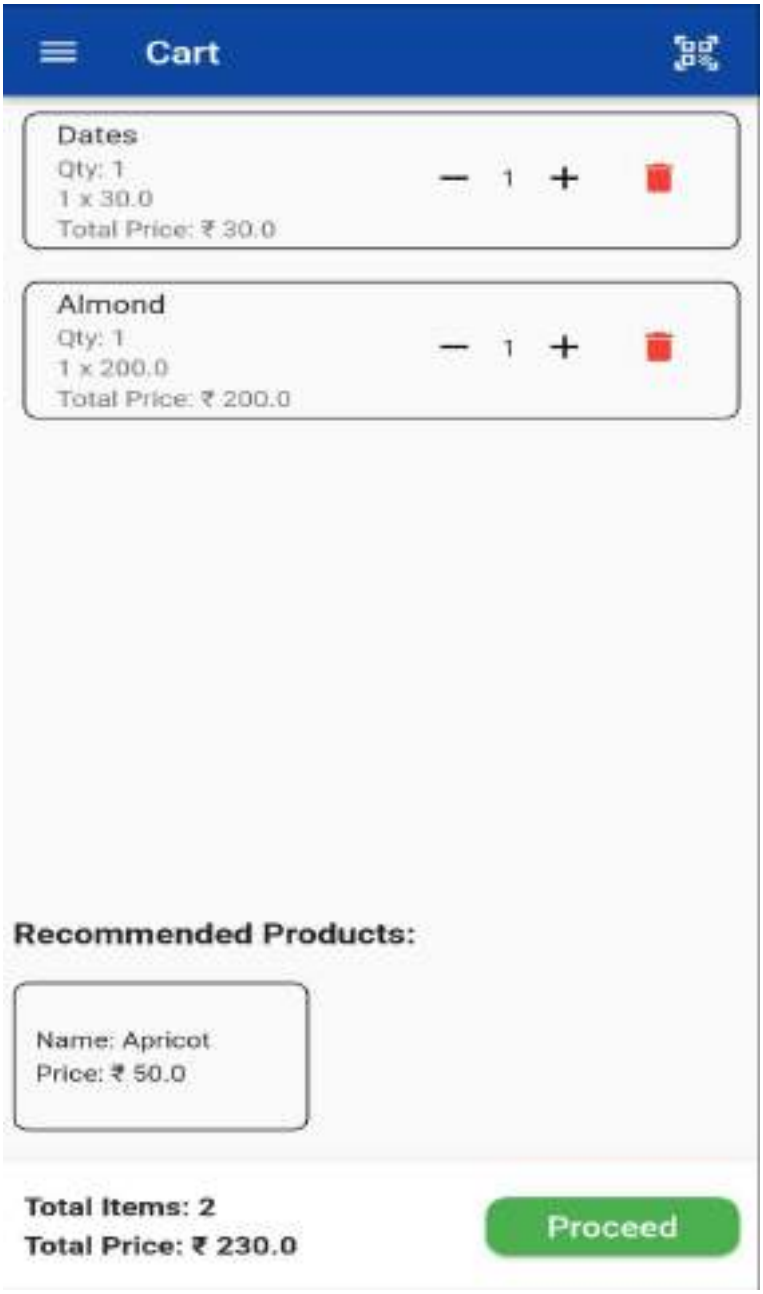


Figure 7.4: Cart and Recommendation

The above figure shows the cart of the application where the products are displayed once they are scanned. The number of times the camera scans the QR code the number of the times the products will get added to the cart and will be displayed as shown in the figure to the user. The recommendation system in the application takes the user input and suggests the products as per the category. For example - The user has added the products 'Dates' and 'Almond' which fall into the same category such as dry fruit , as a result the recommendation system suggests the product related to and from the category dry fruits that is 'apricot'. The product displayed also shows the total quantity added to the cart.

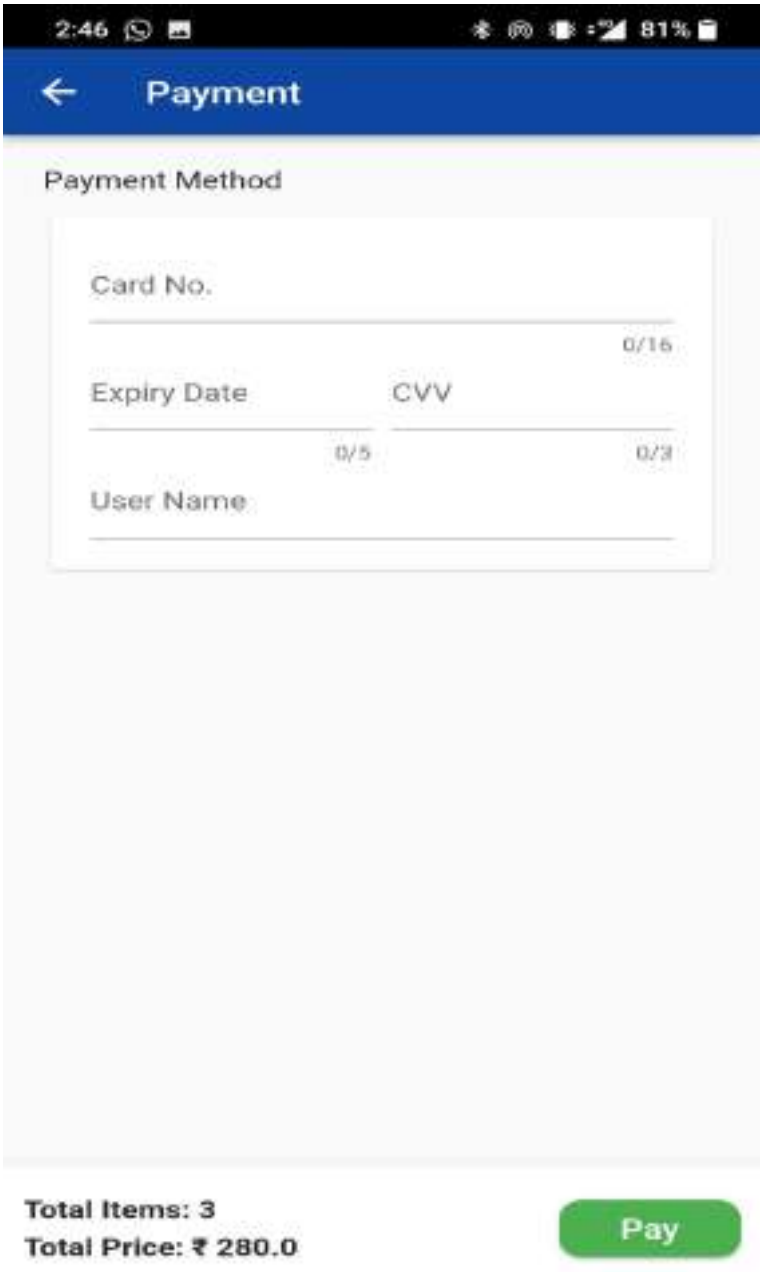


Figure 7.5: Payment Page

This above figure is the final user end the payment page where the customer have to enter their card information debit/credit with the respective expiry date and CVV and user name as given on the card to make a successful payment,click on the pay button.

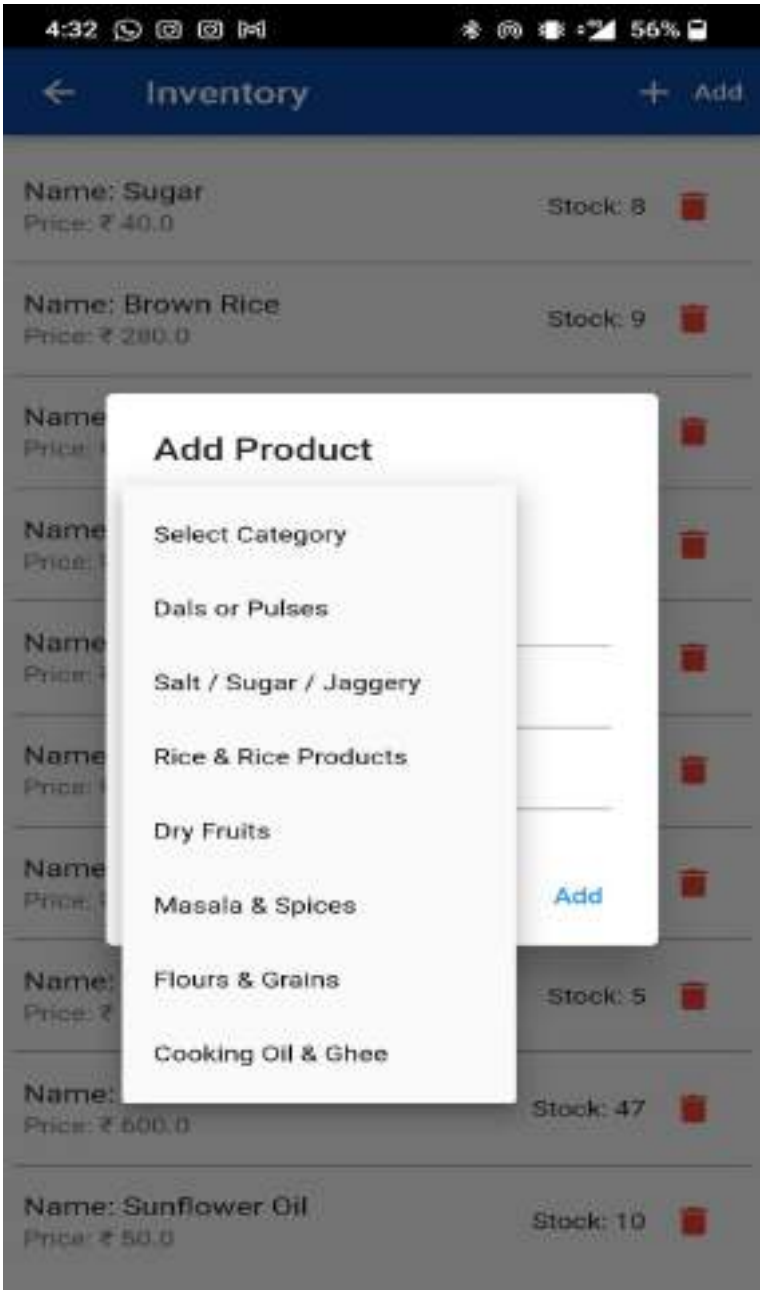


Figure 7.6: Admin Inventory

Admin inventory page : Admin inventory shows the number of products with their prices displayed and instock availability.Admin can add product as per the category name the price and Instock availability.

Chapter 8

Conclusions and Future Scope

In this project we have created a smart shopping framework that allows the users to scan the barcode on the products in the shopping mall by adding to cart, detailed description of the product, view of products in cart in organized manner with name, quantity and net rate and real time total of overall products. The IoT Based Smart Grocery store system leads to significant decrease in time required for billing and thus reduces the overall shopping time for the user.

This system is beneficial for both the customer as well as the super market management. The customer is benefitted by not having to waste time waiting in line for checkout, easy calculation of the products to be bought and thus satisfied customer. On the other hand on implementation of the smart trolley app the supermarkets are highly profitable due to reduced number of employees thus reduced expenditure, providing quality service to customer, less space required for billing thus more area for products and better understanding of the inventory.

By using the app the customers are highly engaged in the shopping experience. This project thereby improves the efficiency, simplifies the process and consumes less time to shop.

Bibliography

[1] Leena Thomas, Renu Mary George, Amalasree Menon, Greeshma Rajan, Reshma Kurian (2017). "Smart Trolley with Advanced Billing System". Vol. 6, Issue 3, March 2017, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. (ISSN: 2320 – 3765)

[2] G Manmadha Rao, K Preethi, A Sai Krishna, Afreen Firdaus, Ch Lokesh (2020). "Rfid Based Smart Trolley for Automatic Billing System." Volume-9 Issue-1, May 2020, International Journal of Recent Technology and Engineering (IJRTE). (ISSN: 2277-3878)

[3] Meghana T K, Rahul S Bedare, Ramakrishna M, Vignesh P, Maria Pavithra (2020). "Smart Shopping Cart with Automated Billing System." International Journal of Engineering Research Technology (IJERT). (ISSN: 2278-018)

[4] T Sarala, Y A Sudha, K V Sindhu, CH Suryakiran, B N Nithin (2017). "Smart Electronic Trolley for Shopping Mall". 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT) doi: 10.1109/RTEICT42901.2018.9012466

Appendices

Appendix-A: Flutter Download and Installation

Step 1 Go to URL, https://flutter.dev/docs/get-started/install/windows and download the latest Flutter SDK. As of April 2019, the version is 1.2.1 and the file is flutter_windows_v1.2.1-stable.zip.

Step 2 Unzip the zip archive in a folder, say C:\flutter\

Step 3 Update the system path to include flutter bin directory.

Step 4 Flutter provides a tool, flutter doctor to check that all the requirement.

flutter doctor

Step 5 Running the above command will analyze the system and show its report.

Step 6 Install the latest Android SDK, if reported by flutter doctor

Step 7 Install the latest Android Studio, if reported by flutter doctor

Step 8 Start an android emulator or connect a real android device to the system.

Step 9 Install Flutter and Dart plugin for Android Studio. It provides startup template to create new Flutter application, an option to run and debug Flutter application in the Android studio itself, etc.,

Appendix-B: Android Studio IDE

Step 1 To download the Android Studio, visit the official Android Studio website in your web browser.

Step 2 Click on the "Download Android Studio" option.

Step 3 Double click on the downloaded "Android Studio-ide.exe" file.

Step 4 "Android Studio Setup" will appear on the screen and click "Next" to proceed.

Step 5 Select the components that you want to install and click on the "Next" button.

Step 6 Now, browse the location where you want to install the Android Studio and click "Next" to proceed.

Step 7 Choose a start menu folder for the "Android Studio" shortcut and click the "Install" button to proceed.

Step 8 After the successful completion of the installation, click on the "Next" button.

Step 9 Click on the "Finish" button to proceed.

Now, your Android studio welcome screen will appear on the screen.

Step 10 "Android Studio Setup Wizard" will appear on the screen with the welcome wizard. Click on the "Next" button.

Step 11 Select (check) the "Standard" option if you are a beginner and do not have any idea about Android Studio. It will install the most common settings and options for you. Click "Next" to proceed.

Step 12 Now, select the user interface theme as you want. (I prefer Dark theme (Dracula) that is most liked by the coders). Then, click on the "Next" button.

Step 13 Now, click on the "Finish" button to download all the SDK components.

And, the downloading and installation process of components gets started.

Step 14 After downloading all the necessary components, click on the "Finish" button.

Appendix-C: Dart Installation

Here's one way to configure Dart support:

1. Start the IDE, and install the Dart plugin.

a. From the Welcome screen, choose Plugins.

b. Search for Dart.

c. Once you've installed the Dart plugin, restart the IDE.

2. Create a new Dart project:

a. From the Welcome screen, click New Project.

b. In the next dialog, click Dart.

3. If you don't see a value for the Dart SDK path, enter it.

4. Choose a starting template.

a.To enable starting templates, click Generate sample content.

b.Pick your desired template.

5. Click Next and continue project setup.

Appendix-D: Firebase Setup on Flutter

Step 1: Install the required command line tools

1.If you haven't already, install the Firebase CLI.

2.Log into Firebase using your Google account by running the following command:

$firebase login

3.Install the FlutterFire CLI by running the following command from any directory:

$dart pub global activate flutterfire_cli

Step 2: Configure your apps to use Firebase

Use the FlutterFire CLI to configure your Flutter apps to connect to Firebase.

From your Flutter project directory, run the following command to start the app

configuration workflow:

your-flutter-proj$ **$flutterfire configure**

Step 3: Initialize Firebase in your app

1.From your Flutter project directory, run the following command to install the core plugin:

your-flutter-proj$ **flutter pub add firebase_core**

2.In your lib/main.dart file, import the Firebase core plugin and the configuration file you generated earlier:

```
import 'package:firebase_core/firebase_core.dart';

import 'firebase_options.dart';
```

3.Also in your lib/main.dart file, initialize Firebase using the DefaultFirebaseOptions object exported by the configuration file:

```
await Firebase.initializeApp(

    options: DefaultFirebaseOptions.currentPlatform,

);
```

4.Rebuild your Flutter application:

your-flutter-proj$ **flutter run**

Step 4: Add Firebase plugins You access Firebase in your Flutter app through the various Firebase Flutter plugins, one for each Firebase product (for example: Cloud Firestore, Authentication, Analytics, etc.).
Since Flutter is a multi-platform framework, each Firebase plugin is applicable for Apple, Android, and web platforms. So, if you add any Firebase plugin to your Flutter app, it will be used by the Apple, Android, and web versions of your app.
Here's how to add a Firebase Flutter plugin:

1.From your Flutter project directory, run the following command:

your-flutter-proj$ **flutter pub add PLUGIN_NAME**

2.Once complete, rebuild your Flutter application:

your-flutter-proj$ **flutter run**