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 2020-2021

A Project Report on

BeSafe: IoT Based Safety Band

Submitted in partial fulfillment of the degree of
Bachelor of Engineering(Sem-8)
in
INFORMATION TECHNOLOGY

By
Samiksha Mhatre (16104025)
Uddharth Ajja (17104061)
Jagruti Patil (17104063)

Under the Guidance of
Prof. Yaminee Patil

1.Project Conception and Initiation

1.1 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..
- 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 grocery application can be used for scanning items, displaying product information, pricing, and total bill.

1.1 Abstract

- 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.
- The billing procedure at the counter is time intensive, and the billing area requires more human resources.
- This method improves the purchasing experience for the customer while shortening the shopping time.

1.2 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

1.3 Literature Review

Sr. No	Authors	Publication	Findings
1	Leena Thomas, Renu Mary Gorege, Amalasree Menon, Greeshma Rajan, Reshma Kurian	Title : Smart Trolley with Advanced Billing System Year :March 2017 Conference : International Journal of Advanced Research in Instrumentation Engineering ISSN :2320-3765	• Server Unit (SU) , A User Interface Unit (UI), in-built Billing Unit (BU) • Establishing and maintaining connection of shopping cart with the main server
2	G Manmadha Rao, K Preeti, A Sai Krishna, Afreen Firdaus,ChLokesh (2020)	Title : RFID Based Smart Trolley for Automatic Biling System. Year : May 2020 Conference : International Journal of Recent Technology (IJERT) : ISSN : 2278-018	• Reduce/replace of present bar code system which is currently being followed. • Using of scanning and storing data with scanner.

Sr. No	Authors	Publication	Findings
1	Meghna T K , Rahul S Bedre, Ramakrishna M, Vignesh P, Maria Pavithra (2020) .	Title : Smart Electronic Trolley with Shopping Mall Year :2020 Conference : International Journal of Engineering Research & Technology (IJERT) (ISSN :2278-018)	• Scanning of product and put into cart that will be displayed on the screen (User Interface) • Establishing and maintaining connection of shopping cart with the main server
2.	T Sarla, Y A Sudha, K V Sindhu, CH Suryakiran, B N Nithin (2017)	Title : “Smart Electronic Trolley For Shopping Mall”, Year : 2018 Conference : IEEE International Conference on Recent Trends In Electronics (RTEICT): 42901.2018.9012466	• Hardware device to scan product and updating real-time database. • Connection Hardware to the User Interface.

1.4 Problem Definition

- While shopping at supermarkets during peak hours, weekends and festival season; a lot of time of the customer is wasted waiting in line at the billing counter. The time wasted at the billing counter may range from 30 minutes to 1 or 2 hours. This leads to increasing crowds and decreasing shopping area. The time wasted at billing counters, makes customers tired and unhappy.
- The supermarkets too have to employ more employees at billing counters thus more investment required for working. Sometimes customers waiting in line at the billing counters for hours only to realize that they cannot afford certain products or miscalculate the offers or discounts on the products. If there is a breakdown of billing machine then the billing times increases excessively.

1.5 Scope

- Can be useful for senior citizen as this formula will avoid long queues.
- Can be useful to avoid human interaction preventing fatal viruses and contagious diseases.
- Can be useful to avoid queue and paperworks like Dmart and other shopping malls.
- Can be used to provide efficient shopping without any chaos.
- Can be used to keep a track on the cart according to the budget and also can be used to provide effortless payment transactions without standing for hours in queues.

1.6 Technology stack

- **Hardware:**
  - ESP32 cam module : approx. 1000/- rupees
  - Jumper Wires : 30/- rupees
- **Software:**
  - Firebase Cloud
  - Flutter
  - Visual Studio Code
  - Arduino IDE 1.8.13

1.7 Benefits for environment & Society

- To provide efficient chaotic free environment for the customers.
- To provide user –friendly application for a smoother shopping experience.
- To understand and make use of smart grocery store application.
- To bring a change in the traditional shopping experience and catch hold of digital world experiences.

2. Project Design

2.1 Proposed System

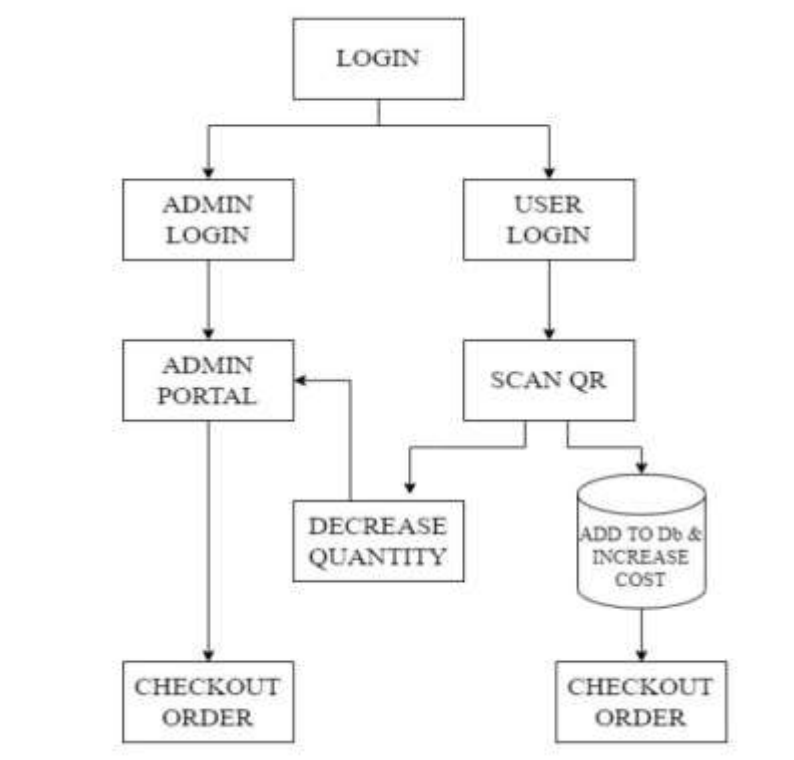
- The system we proposed includes mobile client designed based on the smartphone of the Android platform, so it requires a built-in Wi-Fi module and GPA module in the smartphone, for self-positioning and communication with the backend server.
- Just by the user clicking on the corresponding request or scanning the QR code, the mobile client would feedback the results to the user interface.
- It provides a simple, powerful, efficient and easy to understand SDK to write mobile application in Google’s own language, Dart.

2.2 Design(Flow Of Modules)

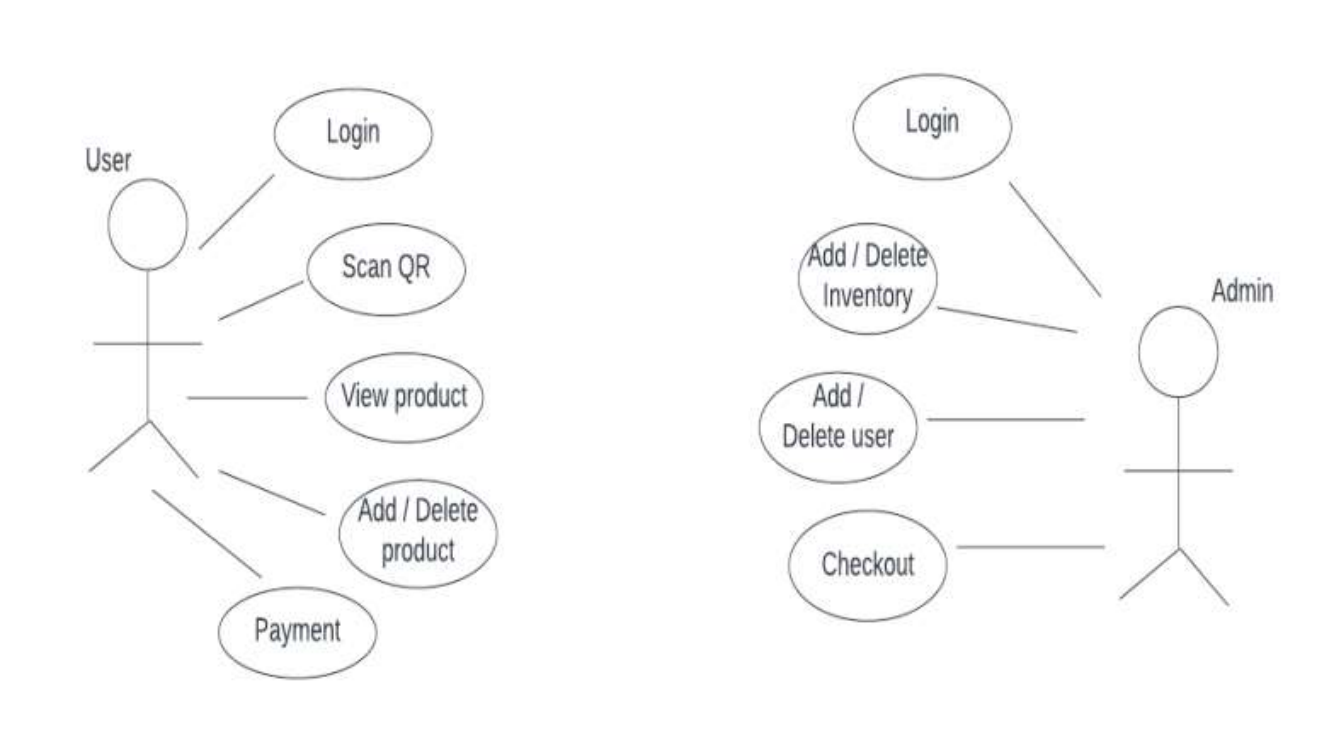
- Flow of Application
- Flow of Camera module
- Flow of Application & Esp32 cam Wifi Module

2.2 Design(Flow Of Modules)

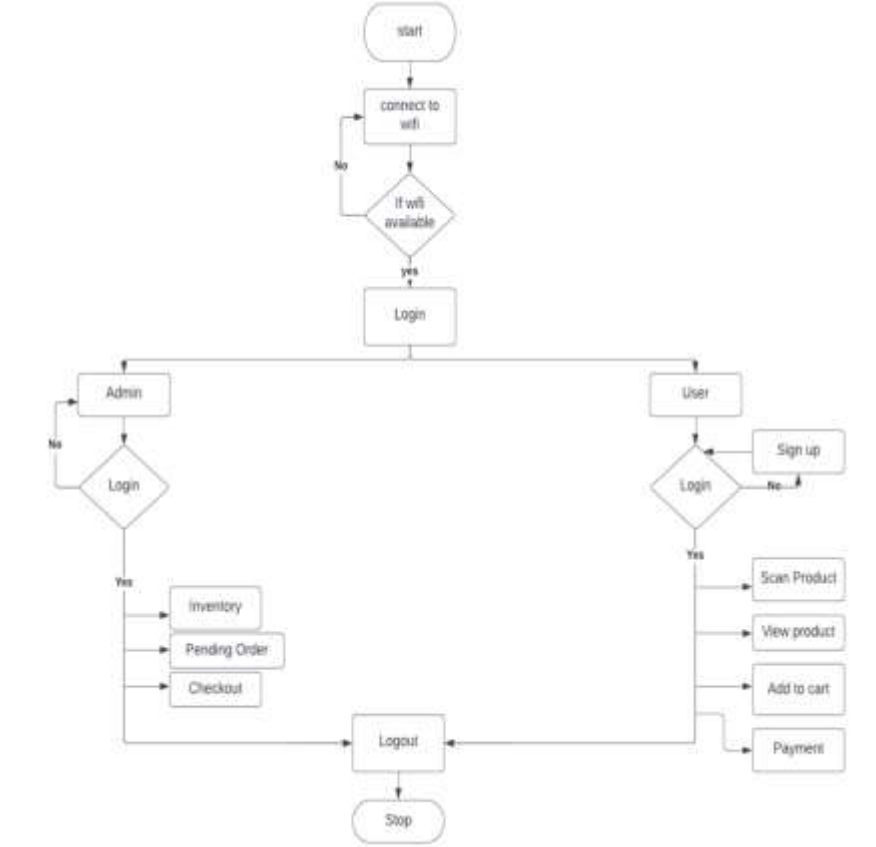
- Flow of Application



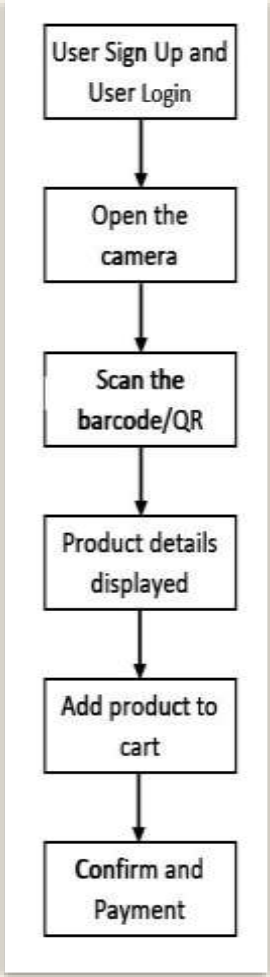
2.3 Use Case for Admin and User Panel



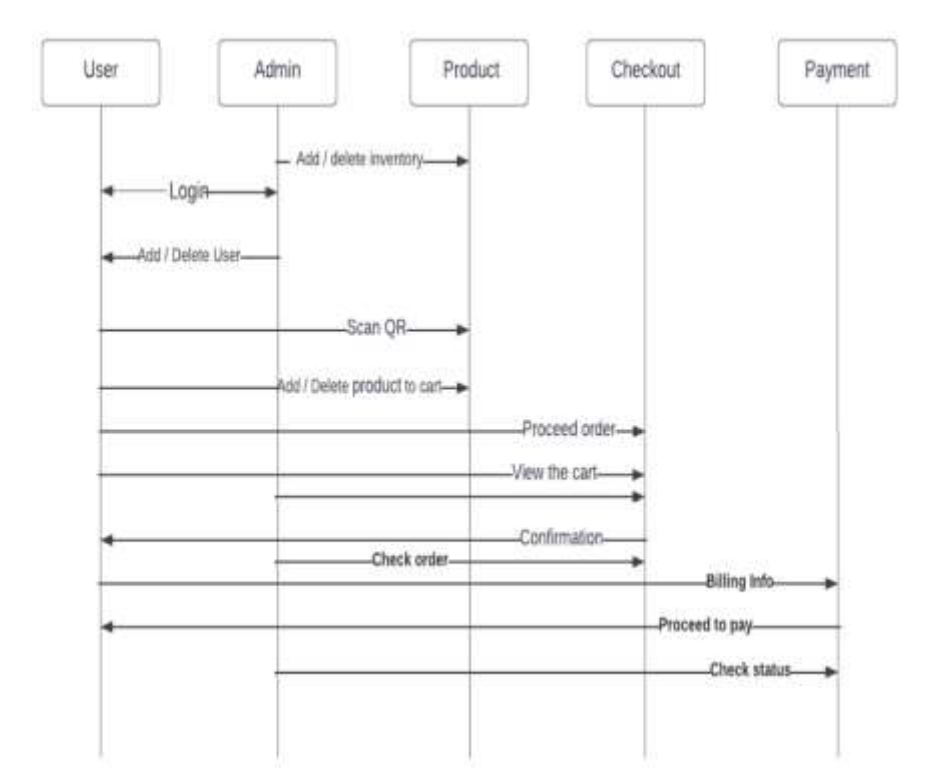
2.4 Activity diagram for Home Screen



2.5 Use Case Diagram



2.5 Sequence Diagram



3. Implementation

Home Dart

```
home_page.dart > _MyHomePageState > onBackPressed
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:grocery/globals/globalData.dart';
import 'package:grocery/payment_screen.dart';
import 'package:grocery/scan_screen.dart';
import 'auth_screen.dart';

class MyHomePage extends StatefulWidget {
  const MyHomePage({Key key}) : super(key: key);

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

class _MyHomePageState extends State<MyHomePage> {
  int backCount = 0;
  var currentCartList = [];
  bool showSuggestion = false;
  double totalPrice = 0.0;
  int quantity = 0;
  onBackPressed() {
    backCount++;
    if (backCount > 1) {
      SystemNavigator.pop();
    } else {
      Fluttertoast.showToast(
          msg: "Press back again to exit", toastLength: Toast.LENGTH_SHORT);  // toast user warning to confirm on  to exit app
    }
  }

  @override
  void initState() {
    super.initState();
```

```
return Padding(
    padding: const EdgeInsets.all(10.0),
    child: ListTile(
      shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(8),
          side: BorderSide(color: Colors.black)), // RoundedRectangleBorder
      title: Text(currentCartList[i]
          .data()['productName']), // Text
      subtitle: Column(
        crossAxisAlignment:
            CrossAxisAlignment.start,
        children: [
          SizedBox(
            height: 3,
          ), // SizedBox
          Text(
              'Qty: ${currentCartList[i].data()['quantity']}'), // Text
          SizedBox(
            height: 3,
          ), // SizedBox
          Text(
              '${int.parse(currentCartList[i].data()['quantity'])} x ${double.parse(currentCartList[i
          SizedBox(     // data is in string we're converting in int
            height: 3,
          ), // SizedBox
          Text(
              'Total Price: $rupeeSymbol${double.parse(currentCartList[i].data()['price']) * double.p
        ],
      ), // Column
      trailing: Row(
        mainAxisAlignment:
            MainAxisAlignment.spaceAround,
        mainAxisSize: MainAxisSize.min,
        children: [
          IconButton(
              padding: EdgeInsets.all(2),
              onPressed: () async {
```

Home Dart : Product Add,Remove,Out Of Stock

```
children: [
  IconButton(
      padding: EdgeInsets.all(2),
      onPressed: () async {
        int currentQuantity = int.parse(
            currentCartList[i]
                .data()['quantity']);
        int updatedStock =
            currentQuantity - 1;
        if (updatedStock == 0) {
          deleteCartProduct(context,
              currentCartList[i].id);
          setState(() {
            currentCartList =
                currentCartList;
          });
          Fluttertoast.showToast(
              msg: 'Product removed',
              toastLength:
                  Toast.LENGTH_SHORT);
        } else {
          updateCartProduct(
              currentCartList[i]
                  .data()['productCode'],
              updatedStock.toString());
          setState(() {
            currentCartList =
                currentCartList;
          });
        }
      },
      icon: Icon(Icons.remove,
          color: Colors.black)), // Icon // IconButton
  Text(
      '${currentCartList[i].data()['quantity']}'), // Text
  IconButton(
      padding: EdgeInsets.all(2),
```

```
                'productCode'])
            .get();
        int inStockAmount = int.parse(
            singleProduct['instock']);
        if (updatedStock >
            inStockAmount) {
          showSnackBar(
              context,
              Colors.red,
              'Out of stock.');
        } else {
          updateCartProduct(
              currentCartList[i]
                  .data()['productCode'],
              updatedStock.toString());
          setState(() {
            currentCartList =
                currentCartList;
          });
        }
      },
      icon: Icon(Icons.add,
          color: Colors.black)), // Icon // IconButton
  IconButton(
      padding: EdgeInsets.all(2),
      onPressed: () async {
        deleteCartProduct(context,
            currentCartList[i].id);
        Fluttertoast.showToast(
            msg: 'Product removed',
            toastLength:
                Toast.LENGTH_SHORT);
        setState(() {
          currentCartList =
              snap.data.docs;
```

```
    mainAxisSize: MainAxisSize.min,
    crossAxisAlignment:
        CrossAxisAlignment.start,
    mainAxisAlignment:
        MainAxisAlignment.start,
    children: [
      Container(
        padding: const EdgeInsets.symmetric(
            horizontal: 5.0, vertical: 0), // EdgeInsets.symmetric
        child: Text(
          'Recommended Products:',
          textAlign: TextAlign.start,
          style: TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 18), // TextStyle
        ), // Text
      ), // Container
      Expanded(
        child: ListView.builder(
            scrollDirection: Axis.horizontal,
            itemCount: suggestionList.length,
            itemBuilder: (ctx, i) {
              return GestureDetector(
                onTap: () {
                  addProductInCart(
                      context,
                      suggestionList[i]
                          ['productCode']);
                },
                child: Padding(
                    padding: const EdgeInsets
                        .symmetric(
                      horizontal: 5.0,
                      vertical: 20), // EdgeInsets.symmetric
                    child: Container(
                      padding:
                          const EdgeInsets
```

Admin Page : Inventory,Pending Order,History

```
            bottomRight: Radius.circular(12),
            bottomLeft: Radius.circular(12.0)), // BorderRadius.only // BoxDecoration
    ), // UserAccountsDrawerHeader
    ListTile(
        title: Text('Pending Order'),
        onTap: () async {
          Navigator.of(context).pop();
        }), // ListTile
    Padding(
      padding: const EdgeInsets.only(left: 8.0, right: 8.0),
      child: Divider(
        thickness: 2.5,
      ), // Divider
    ), // Padding
    ListTile(
        title: Text('Inventory'),
        onTap: () async {
          Navigator.of(context).pop();
          Navigator.of(context).pushReplacement(
            MaterialPageRoute(builder: (_) => InventoryScreen()),
          );
        }), // ListTile
    Padding(
      padding: const EdgeInsets.only(left: 8.0, right: 8.0),
      child: Divider(
        thickness: 2.5,
      ), // Divider
    ), // Padding
    ListTile(
        title: Text('History'),
        onTap: () async {
          Navigator.of(context).pop();

          Navigator.of(context).pushReplacement(
            MaterialPageRoute(builder: (_) => HistoryScreen()),
          );
```

```
        thickness: 2.5,
      ), // Divider
    ), // Padding
    ListTile(
        title: Text('Logout'),
        onTap: () async {
          await FirebaseAuth.instance.signOut().then((value) {
            Navigator.of(context).pushAndRemoveUntil(
                MaterialPageRoute(builder: (_) => AuthScreen()),
                (route) => false);
          });
        }), // ListTile
    Padding(
      padding: const EdgeInsets.only(left: 8.0, right: 8.0),
      child: Divider(
        thickness: 2.5,
      ), // Divider
    ), // Padding
  ]), // Column
), // Drawer // SafeArea
body: StreamBuilder<QuerySnapshot<Map<String, dynamic>>>(
  stream: FirebaseFirestore.instance.collection('AllUsers').snapshots(),
  builder: (ctx, snap) {
    if (snap.connectionState == ConnectionState.waiting) {
      return Center(child: CircularProgressIndicator());
    } else {
      return Column(
        children: [
          Expanded(
            child: ListView.builder(
                itemCount: snap.data.docs.length,
                itemBuilder: (ctx, i) {
                  return FutureBuilder<
                          QuerySnapshot<Map<String, dynamic>>>(
                      future: FirebaseFirestore.instance
                          .collection('AllUsers')
                          .doc(snap.data.docs[i].data()['email'])
```

Payment Page

```
    },
    icon: Icon(Icons.arrow_back, color: Colors.white, size: 26),
  ), // IconButton
  title: Text('Payment', style: TextStyle(color: Colors.white)),
  backgroundColor: Colors.blue[900],
), // AppBar
body: Container(
    alignment: Alignment.center,
    margin: EdgeInsets.symmetric(horizontal: 20, vertical: 20),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text('Payment Method', style: TextStyle(fontSize: 16)),
        SizedBox(height: 10),
        Card(
          color: Colors.white,
          child: Container(
            padding: EdgeInsets.all(20),
            child: Column(mainAxisSize: MainAxisSize.min, children: [
              TextFormField(
                autocorrect: false,
                maxLength: 16,
                textCapitalization: TextCapitalization.none,
                enableSuggestions: false,
                validator: (value) {
                  if (value.isEmpty) {
                    return 'Cannot be empty';
                  }
                  return null;
                },
                onChanged: (val) {
                  cardNo = val;
                },
                keyboardType: TextInputType.number,
                decoration: const InputDecoration(
                  hintText: 'Card No.',
                ), // InputDecoration
```

```
          maxLength: 3,
          validator: (value) {
            if (value.isEmpty) {
              return 'Cannot be empty';
            }
            return null;
          },
          onChanged: (val) {
            cvv = val;
          },
          keyboardType: TextInputType.number,
          decoration: const InputDecoration(
            hintText: 'CVV',
          ), // InputDecoration
        ), // TextFormField
      ), // Expanded
    ],
  ), // Row
  TextFormField(
    autocorrect: false,
    textCapitalization: TextCapitalization.none,
    enableSuggestions: false,
    validator: (value) {
      if (value.isEmpty) {
        return 'Cannot be empty';
      }
      return null;
    },
    onChanged: (val) {
      userName = val;
    },
    keyboardType: TextInputType.text,
    decoration: const InputDecoration(
      hintText: 'User Name',
    ), // InputDecoration
  ), // TextFormField
]), // Column
```

```
      blurRadius: 1.0,
      spreadRadius: 4.0,
      offset: Offset(0.0, 0.0), // shadow direction: bottom right
    ) // BoxShadow
  ],
), // BoxDecoration
child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            'Total Items: $quantity',
            style: TextStyle(
                fontSize: 16, fontWeight: FontWeight.bold), // TextStyle
          ), // Text
          SizedBox(height: 5),
          Text(
            'Total Price: $rupeeSymbol$totalPrice',
            style: TextStyle(
                fontSize: 16, fontWeight: FontWeight.bold), // TextStyle
          ) // Text
        ],
      ), // Column
      ElevatedButton(
        style: ElevatedButton.styleFrom(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(12),
            ), // RoundedRectangleBorder
            primary: Colors.green),
        child: Padding(
          padding: const EdgeInsets.symmetric(
              vertical: 8.0, horizontal: 15), // EdgeInsets.symmetric
          child: Text('Pay',
              style: const TextStyle(
```

QR Code Scnnaer using ESP 32

```python
from unittest import result
import cv2       # used for camera recognizations
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                                 #foont recog

url='http://192.168.0.101/' #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 n 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)
```

```python
    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 n 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) # firebae data is stored in JSON
            }
            result = firebase.post('https://grocery-store-5615b-default-rtdb.firebaseio.com/QR',qr)
            print(result)
            prev=pres

        cv2.putText(frame, str(obj.data), (50, 50), font, 2,
                    (255, 0, 0), 3)

    cv2.imshow("live transmission", frame) # final window

    key = cv2.waitKey(1) # ctl c close where ctl z cancel
    if key == 27:
        break

cv2.destroyAllWindows()
```

QR Code Scnnaer using ESP 32

```
#include "src/OV2640.h"
#include <WiFi.h>
#include <WebServer.h>
#include <WiFiClient.h>

// Select camera model
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"
#define SSID1 "Sagar"
#define PWD1 "255255255"

OV2640 cam;

WebServer server(80);

const char HEADER[] = "HTTP/1.1 200 OK\r\n" \
                      "Access-Control-Allow-Origin: *\r\n" \
                      "Content-Type: multipart/x-mixed-replace; boundary=123456789000000000000987654321\r\n";
const char BOUNDARY[] = "\r\n--123456789000000000000987654321\r\n";
const char CTNTTYPE[] = "Content-Type: image/jpeg\r\nContent-Length: ";
const int hdrLen = strlen(HEADER);
const int bdrLen = strlen(BOUNDARY);
const int cntLen = strlen(CTNTTYPE);

void handle_jpg_stream(void)
{
  char buf[32];
  int s;

  WiFiClient client = server.client();

  client.write(HEADER, hdrLen);
  client.write(BOUNDARY, bdrLen);

const int jhdLen = strlen(JHEADER);

void handle_jpg(void)
{
  WiFiClient client = server.client();

  cam.run();
  if (!client.connected()) return;

  client.write(JHEADER, jhdLen);
  client.write((char *)cam.getFb(), cam.getSize());
}

void handleNotFound()
{
  String message = "Server is running!\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  server.send(200, "text / plain", message);
}

void setup()
{

  Serial.begin(115200);
  //while (!Serial);            //wait for serial connection.

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
```

4. Testing

Functional Testing

- 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.
  - The Unit testing is best suited for our application development phase. In that phase, we started to code in units create different modules. And test each module separately, like the login page, register page, home page, etc. All these pages are tested and debugged before going further integrating. And check whether we are getting the desired output from each module as for the objectives.

Functional Testing

- Integration 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.
  - As we have discussed unit testing the next step is integration testing. All the units which we have tested and debugged are now ready to integrate into a whole single module. The integration part is crucial as we need to know which unit must interact without error, calling them in a different class accessing the instance of that class all these can be cleared with help of the sequence diagram which was represented in Project Design. So accordingly, modules are integrated and checked whether they behave as for the objectives.

Functional Testing

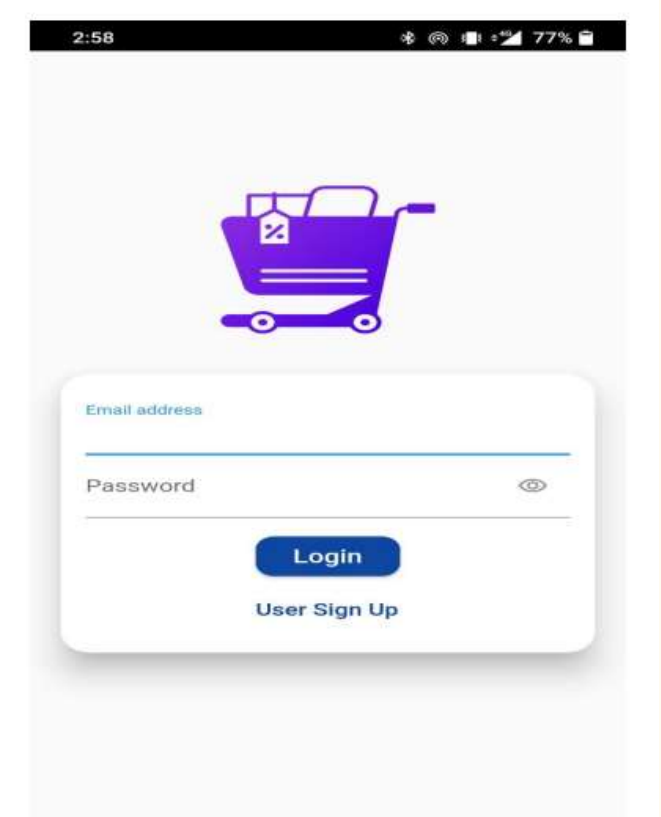
- System Testing
  - With system testing, QA professionals test the software in its entirety, as a complete product. With this type of functional testing, testers validate the complete and integrated software package to make sure it meets requirements. Where necessary, testers can provide feedback on 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.

Non-Functional Testing

- Compatibility Testing
  - Compatibility testing is used to gauge how an application or piece of software will work in different environments. It is used to check that your product is compatible with multiple operating systems, platforms, browsers, or resolution configurations. The goal is to ensure that your software’s functionality is consistently supported across any environment you expect your end-users to be using.
  - The framework we are using to develop our application is Flutter. It is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. We make sure that our application is compatible with both IOS and Android operating systems. The features we developed are perfectly run in multiple operating systems without an error. For this reason, compatibility testing is best suited for our project.

5. Result

Home /Login Page



Cart and Recommendation

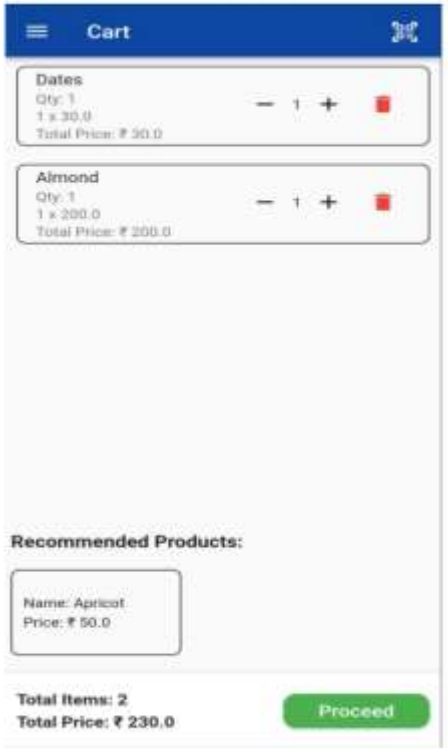


Figure 7.4: Cart and Recommendation

Payment Page

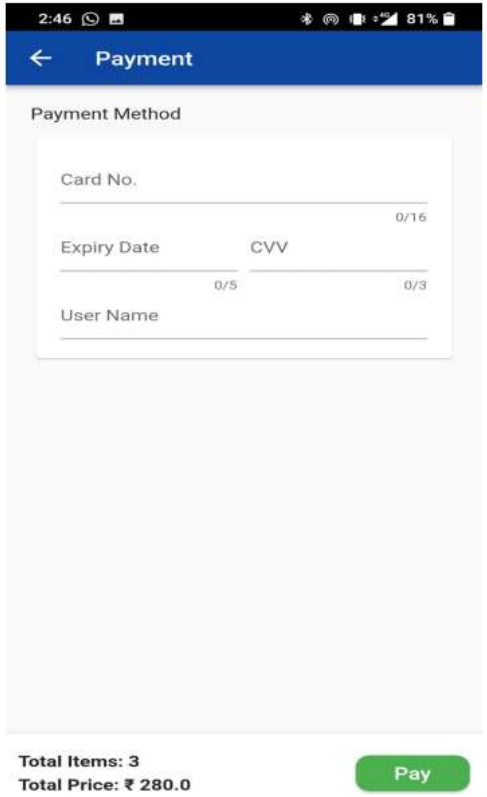


Figure 7.5: Payment Page

Payment Page

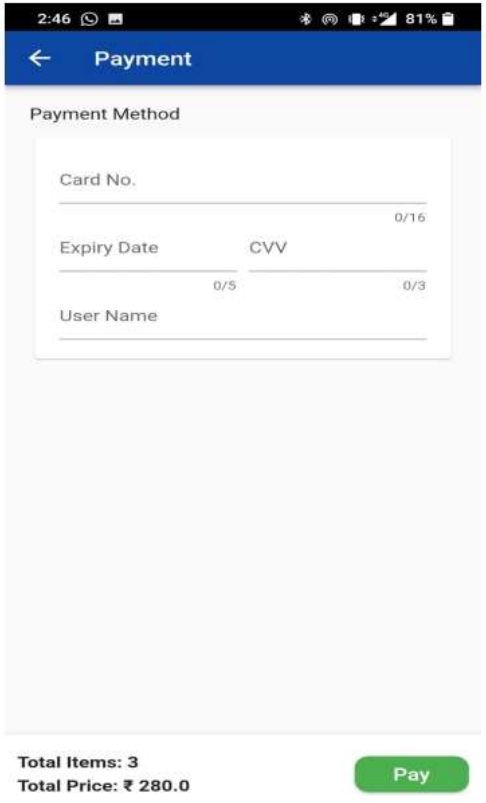


Figure 7.5: Payment Page

QR Scanning and Updating in Database – ESP32

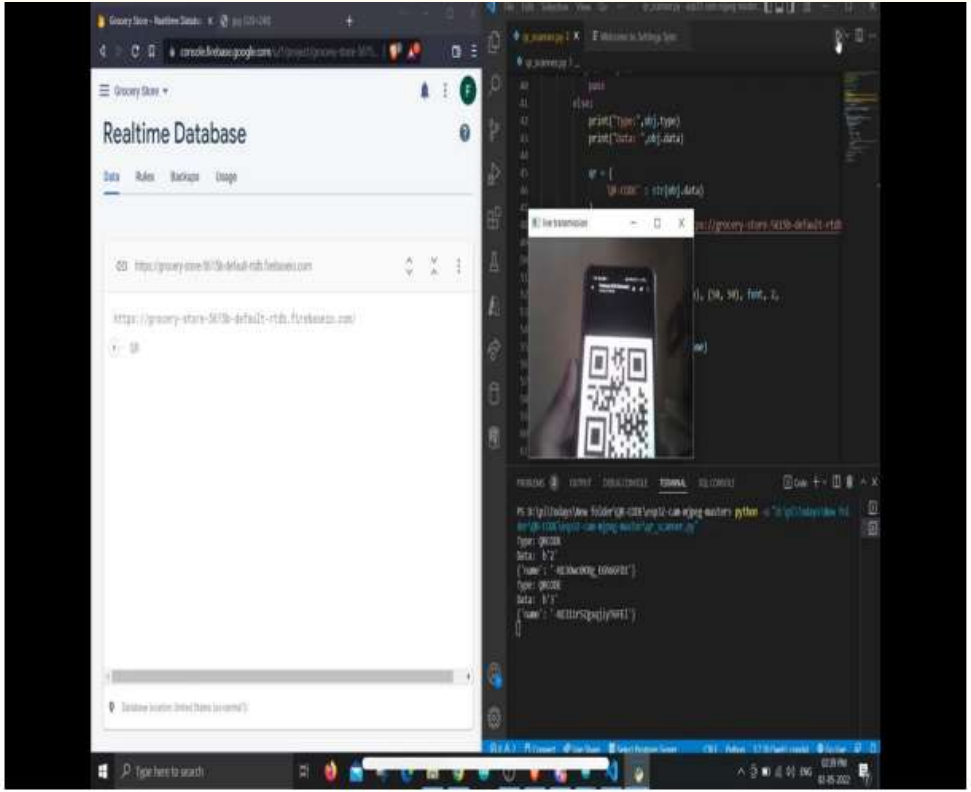


Figure 7.3: QR Scanning using Esp32

6. Conclusion 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.

- 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.

- 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.

References

- 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)
- 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)
- 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)
- 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

Thank You