ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

AFFILIATED TO MAHATMA GANDHI UNIVERSITY, KOTTAYAM

Scan and Go

PROJECT REPORT

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE IN COMPUTER APPLICATIONS

[TRIPLE MAIN]

Submitted By:

ATHEESHA MARY

III B.Sc. Computer Applications [Triple Main]

Register No: SB22CA007

Under the guidance of

Ms. Nithya A.B

Assistant Professor

Department of Computer Applications

2022-2025

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

AFFILIATED TO MAHATMA GANDHI UNIVERSITY

CERTIFICATE

This is to certify that the project report entitled **"Scan and Go"(supermarket self-checkout app)** is a bona fide record of the work done by **ATHEESHA MARY (SB22CA007)** during the year 2022 – 2025 and submitted in partial fulfilment of the requirements for the degree of **Bachelor of Science in Computer Applications (Triple Main)** under Mahatma Gandhi University, Kottayam.

17/03/2025

Head of the Department

17/3/25

Internal Examiner

Date 17/03/2025

17/3/'25

External Examiner

Dr. Roseena. M.A

DECLARATION

I'm, **ATHEESHA MARY (SB22CA007) , B.Sc. Computer Applications [Triple Main]** student of **St. Teresa's College (Autonomous), Ernakulam,** hereby declare that the project submitted for **Bachelor's Degree in Computer Application** is my original work. we further declare that the said work has not previously been submitted to any other university or academic body.

Date: 17/03/2025

Place : Ernakulam

@theesha

ATHEESHA MARY

ACKNOWLEDGEMENT

I would like to convey my heartfelt gratitude to **Rev. Dr. Sr. Vinitha (CSST) Manager, Director Rev. Sr. Emeline (CSST) and Principal Dr. Alphonsa Vijaya Joseph** for providing me with this wonderful opportunity to work on a project with the topic Comparative Analysis of Machine Learning Algorithms and Interface Implementation.

I would like to express my profound gratitude to the Head of the Department of Computer Applications and **my project guide Ms. Nithya A.B** and all other faculty of the department for their contributions to the completion of my project. The completion of the project would not have been possible without their help and insights.

I would also like to Thank my project guide at LCC Computer Education Institute, Mr. Gokul Anand for training me well to develop this project.

Finally, I take this opportunity to Thank all them who has directly or indirectly helped me with my project.

ATHEESHA MARY

ABSTRACT

The Supermarket Self-Checkout App is designed to enhance the shopping experience by providing a convenient and efficient way for customers to complete their purchases independently. This system allows users to scan items, view their total, and make secure payments without waiting in traditional checkout lines. Through an intuitive interface, customers can effortlessly manage their shopping cart, access discounts, and complete transactions in real-time. The app offers various payment methods and ensures a smooth and secure checkout process, saving time and reducing the need for manual cashier assistance. By streamlining the process, the app aims to improve the overall efficiency of supermarkets, offering greater convenience for customers while optimizing store operations.

CONTENTS

1. INTRODUCTION

1.1 About the Project

The Supermarket Self-Checkout app project aims to develop a mobile based application that enables customers to scan and purchase items independently without the need of the traditional checkout lanes or cashier assistance. The app allows users to scan product QR code using their smartphone camera , can add and view products in the shopping cart , view the total cost of the products which is added to the shopping cart as real - time , and make payments through various options such as credit or debit cards, digital wallets . The project focuses on providing a seamless and user - friendly interface to enhance the shopping experience , reduce wait times for the customers. And it also beneficial for the supermarkets which improves operational efficiency by minimizing human errors , optimizing checkout processes , and improving resource allocation. Ultimately, the Supermarket Self-Checkout App offers a modern , efficient , and engaging shopping experience for customers while providing retailers with valuable operational insights and increased customer engagement.

1.2 Objectives of the Project

The supermarket self-checkout app is designed with several key objectives aimed at transforming the traditional shopping process. First, it strives to create a more efficient and enjoyable shopping experience by allowing customers to scan products, check prices, and make payments directly through their smartphones, all while avoiding long checkout lines. Another objective is to reduce operational costs for supermarkets by eliminating the need for traditional cash registers and cashiers, which helps streamline the workflow. In addition, the app focuses on protecting customer information by implementing secure payment options, ensuring privacy and reducing risks of fraud. To further enhance the user experience, the app is connected to the store’s inventory system, providing real-time updates on product stock and availability. The app also supports various payment methods, including credit and debit cards, mobile wallets, giving shoppers more flexibility at checkout. Finally, its user-friendly design is intended to be intuitive and easy to use, making it simple for customers to navigate and complete their transactions with minimal effort. The overall aim of the app is to make the shopping process faster, more efficient, and secure while improving operational performance for supermarkets.

2. SYSTEM ANALYSIS

2.1 Introduction

System Analysis is a detailed study of the various operations performed by the system ant their relationship within the modules of the system. This phase involves the study of the parent system and identification of the system objectives. The main objective of this phase involves gathering necessary information and using the structured tool for analysis. This includes designing the system. In this project, the requirements are studied in detail and information are collected and documented.

2.2 Existing System

The existing traditional supermarket checkout system refers to the current methods used by supermarkets to handle customer transactions, including product scanning, payment processing, and checkout. These systems typically involve cashier-operated registers, where customers bring their selected items to the counter, and a cashier scans each product's barcode, calculates the total cost, and processes payments through cash, credit/debit cards, or mobile wallets. The system may also include manual processes for bagging items. While effective, the traditional checkout method can lead to longer wait times, especially during peak shopping hours, and requires human staff for scanning and payment handling. Some supermarkets have begun incorporating self-checkouts using external machines , where customers can scan and pay for their items independently, but the traditional system remains widely in use, especially in smaller stores or locations with limited technological infrastructure.

2.3 Limitations of Existing system

-Long Wait Times: During peak shopping hours, long lines at checkout counters can lead to delays, causing frustration for customers and reducing overall store efficiency.

-Staff Dependency: The traditional system relies heavily on cashiers for scanning products, processing payments, and assisting customers. This dependence on human labor can result in higher operational costs and inefficiencies, especially when cashiers are unavailable or underperforming.

-Limited Flexibility: The system generally offers a limited range of payment methods and may not always accommodate newer technologies like mobile wallets or contactless payments, reducing convenience for tech-savvy customers.

-Human Error: Manual scanning and entry of items by cashiers can lead to mistakes such as mispricing or incorrect item entries, resulting in customer dissatisfaction and potential financial discrepancies.

-Slower Checkout Process: In comparison to automated or self-checkout systems, the traditional method can be slower, as customers must wait for the cashier to scan all items and process the payment.

-Space and Resource Intensive: The traditional checkout system requires more space and resources for cash registers, equipment, and additional staff. This setup can increase operational costs for supermarkets, particularly in larger stores.

-Security Risks: Cash transactions and manual handling of payment information can pose security risks, including the possibility of theft or fraud, particularly in busy environments.

-Limited Customer Control: Customers have little control over the checkout process in traditional systems. They are dependent on cashier availability and efficiency, which can sometimes affect the overall shopping experience.

2.4 Proposed System

The automated supermarket self-checkout system with distributed architecture addresses several key issues, including:

- The system maintains a central database that stores information on product availability, pricing, and inventory across multiple locations, ensuring easy accessibility and consistency for both customers and store management.

- Product details, including availability and pricing, can be accessed at the touch of a button, making the shopping experience faster and more efficient for customers.

- Customers can register and set up their accounts online, allowing for easy access to the system and a smooth, 24/7 registration process.

- The self-checkout system enables customers to scan and pay for their items independently, regardless of their location, offering convenience without needing to interact with a cashier.

- Real-time updates provide customers with up-to-the-minute information on product availability, promotions, and their transaction status, all from the convenience of their smartphones.

- The system improves decision-making by providing quicker, more consistent information regarding product prices, availability, and payment options.

- Customers have immediate access to information about their shopping cart, including current item availability, pricing details, and payment status, enhancing the overall shopping experience.

2.5 Hardware & software specifications

Front End	HTML , CSS , JavaScript
Back End	Database Management System : MySQL
Operating System	Windows 10/11 , Android
Web Browser	Google Chrome
Software Used	Xampp

3. SYSTEM DESIGN

3.1Introduction

System design is the process of defining and planning the architecture, components, modules, interfaces, and interactions of a complex software or hardware system. It involves making decisions about how different parts of a system will work together to achieve the desired functionality, performance, scalability, reliability, and maintainability. The goal of system design is to create a blueprint or roadmap for building a system that meets the requirements and objectives of a project. This includes breaking down the system into smaller subsystems, modules, or components, and determining how they will communicate and collaborate to accomplish the overall goals. System design takes into consideration various technical and non-technical such as :

-System Architecture: Designing the high-level framework of the system, which includes defining its major components, their relationships, and how data will flow between them.

-Components and Modules: Identifying the distinct elements that make up the system, assigning clear responsibilities to each, and outlining how they will interact with one another.

-Data Management: Developing the data model and selecting the appropriate data storage methods that will enable the system to store, retrieve, and manage data efficiently.

-Communication and Interfaces: Determining how different parts of the system will communicate, including setting up APIs, protocols, and interfaces for smooth interaction.

-Scalability and Performance: Planning how the system will handle growth in terms of traffic and load, ensuring it can scale and maintain good performance as usage increases.

-Security: Identifying and addressing potential security risks, implementing protections against unauthorized access, data breaches, and other cyber threats.

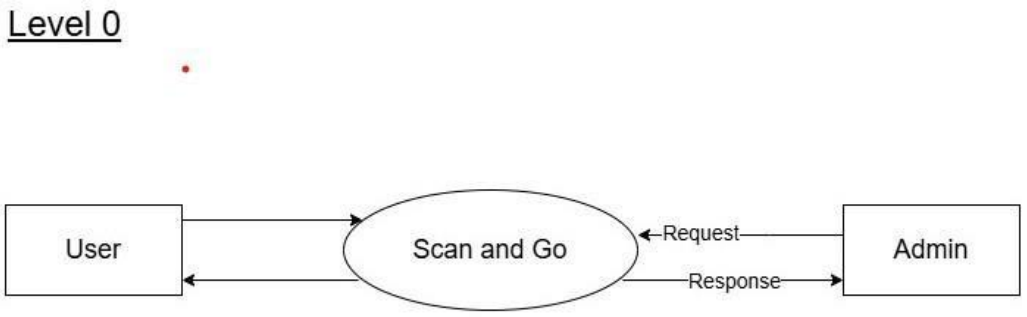
-Reliability and Fault Tolerance: Ensuring the system remains operational even during component failures, by building in redundancy and failover mechanisms to handle errors gracefully.

-Deployment and Infrastructure: Defining how the system will be deployed across different environments (development, testing, production), and determining the required infrastructure like servers, networks, and cloud services.

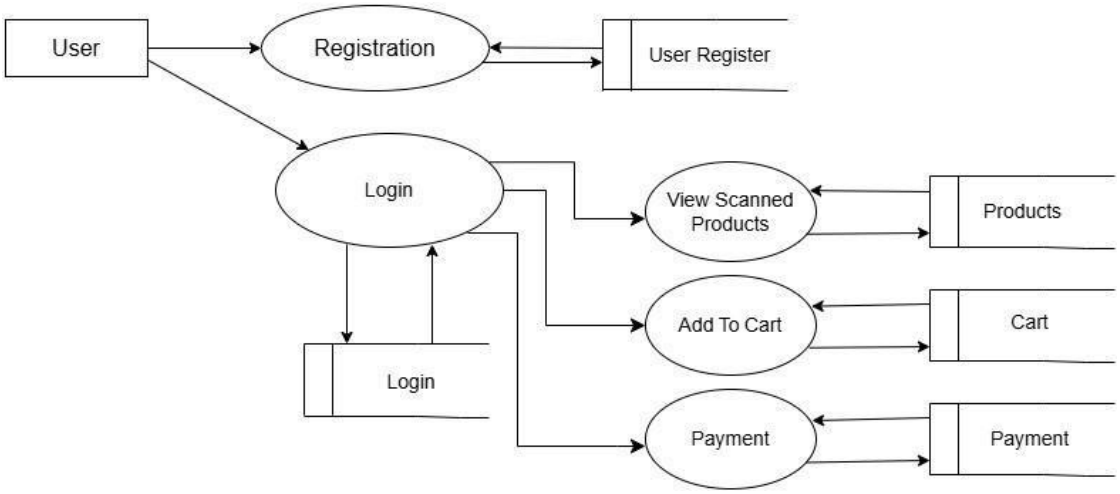
-User Experience (UX): Designing intuitive and efficient user interfaces and workflows that prioritize ease of use, aiming to create a positive experience for the end users.

-Maintainability and Extensibility: Creating a flexible design that allows for easy updates, maintenance, and future scalability, minimizing disruptions during future changes or improvements.

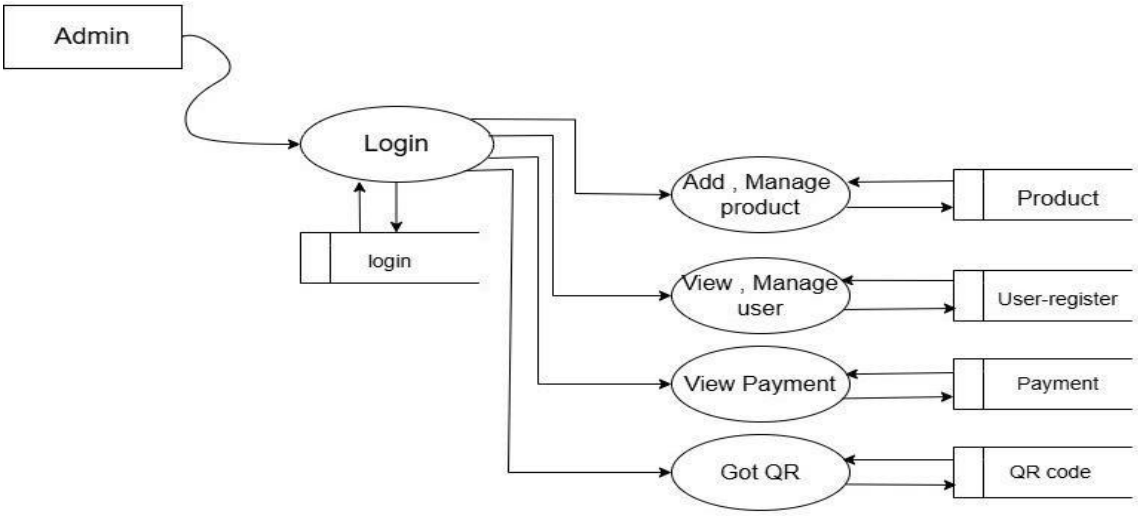
3.2 Data Flow Diagram



ADMIN – 1.1



USER – 1.2



4. SYSTEM DEVELOPMENT

4.1 Process Description

Admin : This module manage overall activities of the system.

- Login : Admin can log in securely to the management dashboard.

- Manage Users :
    - Create, update, and delete user profiles (Customer, Store Staff).
    - Manage roles and permissions.

- Product Management :
    - Add, update, or remove products from the inventory.
    - Set prices for products.

- Monitor Transactions :
    - View transaction history, including purchases made through the self-checkout.
    - Track refunds, returns, and payment issues.

User : The user's role focuses on the shopping experience and the checkout process

-Login : Secure login via the app.

- Scan Items :
    - Use the app to scan product barcodes or QR codes.
    - View item details and prices.

-Add to Cart : Add selected items to the virtual cart.

-View Cart & Checkout : Review the items in the cart and proceed to checkout.

-Payment :

- Make payments via credit/debit cards, mobile wallets, or other supported payment methods.

- Generate Receipt : Get a digital receipt for purchases.

- Track Purchase History : View past purchases and order history.

5. SYSTEM TESTING AND IMPLEMENTATION

5.1 Introduction

System Testing is a type of software testing that is performed on a completely integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or the context of both. System testing tests the design and behavior of the system and also the expectations of the customer.

5.2 Implementation

The software implementation phase is part of the continuous improvement cycle where deliverables designed are adopted and made operational by an organization or end-users. It is an executing phase of the system development lifecycle that includes the steps of concept to function by coding, testing, and deployment. Implementation encompasses a range of steps that include installation, configuration, customization, and integration to make sure it runs as expected and fulfills the expected objectives and requirements.

- Testing the development system with the sample data.
- Detection and correction of errors.
- Making necessary changes in the system.
- Training and involvement of user personal.
- Installation of software utilities.

6.CONCLUSION

The developed Supermarket Self-Checkout App effectively streamlines supermarket operations by automating key tasks such as product scanning, checkout processes, payment, and inventory management. This leads to improved efficiency, reduced wait times, minimized human errors, and enhanced customer satisfaction. Additionally, the app provides valuable insights into customer purchase behavior, stock levels, and sales trends, contributing to more informed decision-making and optimized inventory management. The user-friendly interface ensures a seamless shopping experience, enabling customers to easily browse products, scan items and complete transactions with minimal effort. Security features, including encrypted payment transactions and anti-theft mechanisms, ensure a safe and secure shopping environment. Overall, the app not only improves operational efficiency but also enhances the shopping experience, leading to increased customer loyalty and greater profitability for the supermarket business.

7.SAMPLE CODE

Scanner Fragment

```
package com.example.scanandgo.User;

import static android.content.Context.MODE_PRIVATE;

import android.app.AlertDialog;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.fragment.app.Fragment;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.scanandgo.R;
import com.example.scanandgo.COMMON.Utility;
import com.journeyapps.barcodescanner.BarcodeCallback;
import com.journeyapps.barcodescanner.BarcodeResult;
import com.journeyapps.barcodescanner.DecoratedBarcodeView;

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

public class ScannerFragment extends Fragment {
```

```
    private DecoratedBarcodeView barcodeScannerView;

    @Override
    public View onCreateView(LayoutInflater inflater, android.view.ViewGroup container,
Bundle savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_scanner, container, false);

        barcodeScannerView = root.findViewById(R.id.barcode_scanner);
        barcodeScannerView.decodeContinuous(callback);
        barcodeScannerView.resume();

        return root;
    }

    private final BarcodeCallback callback = new BarcodeCallback() {
        @Override
        public void barcodeResult(BarcodeResult result) {
            if (result.getText() != null) {
                showScannedData(result.getText());
                barcodeScannerView.pause(); // Pause scanning when popup appears
            }
        }
    };

    /**
     * Display scanned data in a popup.
     */
    private void showScannedData(String scannedData) {
        LayoutInflater inflater = getLayoutInflater();
        View popupView = inflater.inflate(R.layout.popup_add_to_cart, null);

        TextView tvProductId = popupView.findViewById(R.id.tv_popup_product_id);
```

```
        TextView tvProductName = popupView.findViewById(R.id.tv_popup_product_name);
        TextView tvProductCategory =
popupView.findViewById(R.id.tv_popup_product_category);
        TextView tvProductPrice = popupView.findViewById(R.id.tv_popup_product_price);
        TextView tvProductDescription =
popupView.findViewById(R.id.tv_popup_product_description);
        Button btnAddToCart = popupView.findViewById(R.id.btn_add_to_cart);
        Button btnCancel = popupView.findViewById(R.id.btn_cancel);

        // Default Values
        String productId = "";
        String productName = "";
        String productCategory = "";
        String productPrice = "";
        String productDescription = "";

        try {
            // Parse scanned QR data
            HashMap<String, String> parsedData = parseScannedData(scannedData);

            productId = parsedData.getOrDefault("id", "");
            productName = parsedData.getOrDefault("Name", "");
            productCategory = parsedData.getOrDefault("Category", "");
            productPrice = parsedData.getOrDefault("Price", "");
            productDescription = parsedData.getOrDefault("Description", "");
        } catch (Exception e) {
            Toast.makeText(getContext(), "Error parsing QR data: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
        }

        // Set parsed data into UI without headings
        tvProductId.setText(productId);
        tvProductName.setText(productName);
```

```
        tvProductCategory.setText(productCategory);
        tvProductPrice.setText(productPrice);
        tvProductDescription.setText(productDescription);

        AlertDialog dialog = new AlertDialog.Builder(getContext())
                .setView(popupView)
                .setCancelable(false)
                .create();

        String finalProductId = productId;
        String finalProductName = productName;
        String finalProductCategory = productCategory;
        String finalProductPrice = productPrice;
        String finalProductDescription = productDescription;

        btnAddToCart.setOnClickListener(v -> {
            addToCart(finalProductId, finalProductName, finalProductCategory,
finalProductPrice, finalProductDescription);
            dialog.dismiss();
            barcodeScannerView.resume();
        });

        btnCancel.setOnClickListener(v -> {
            dialog.dismiss();
            barcodeScannerView.resume();
        });

        dialog.show();
    }

    /**
     * Parse QR Code Data into a HashMap
     */
```

```
    private HashMap<String, String> parseScannedData(String scannedData) {
        HashMap<String, String> dataMap = new HashMap<>();

        // Handle data formatted with '\n' and ':' delimiters
        String[] keyValuePairs = scannedData.split("\n");
        for (String pair : keyValuePairs) {
            if (pair.contains(":")) {
                String[] entry = pair.split(":", 2); // Split only on the first colon
                if (entry.length == 2) {
                    dataMap.put(entry[0].trim(), entry[1].trim());
                }
            }
        }

        return dataMap;
    }

    /**
     * Add the product to the cart via Volley
     */
    private void addToCart(String productId, String productName, String productCategory,
String productPrice, String productDescription) {
        RequestQueue queue = Volley.newRequestQueue(getContext());

        StringRequest request = new StringRequest(Request.Method.POST,
Utility.SERVERUrl, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (response.trim().equals("success")) {
                    Toast.makeText(getContext(), "Product added to cart successfully!",
Toast.LENGTH_SHORT).show();
                } else {
                    // Toast.makeText(getContext(), "Failed to add product to cart.",
```

```
Toast.LENGTH_SHORT).show();

                Toast.makeText(getContext(), "Product added to cart successfully!",
Toast.LENGTH_SHORT).show();
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(getContext(), "Error: " + error.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    }) {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {

            SharedPreferences prefs = getContext().getSharedPreferences("sharedData",
MODE_PRIVATE);
            final String uid = prefs.getString("reg_id", "No_idd");//"No name defined" is the
default value.
            final String type = prefs.getString("type", "Notype");
            Map<String, String> params = new HashMap<>();
            params.put("key", "addToCart");
            params.put("productId", productId);
            params.put("uid", uid);
            return params;
        }
    };

    queue.add(request);
  }

  @Override
```

```
    public void onResume() {
        super.onResume();
        barcodeScannerView.resume();
    }

    @Override
    public void onPause() {
        super.onPause();
        barcodeScannerView.pause();
    }
}
```

Cart Fragment

```
package com.example.scanandgo.User;

import static android.content.Context.MODE_PRIVATE;

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

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
```

```
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.scanandgo.ADAPTER.CartAdapter;
import com.example.scanandgo.COMMON.RequestPojo;
import com.example.scanandgo.COMMON.Utility;
import com.example.scanandgo.R;
import com.google.gson.Gson;

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

public class CartFragment extends Fragment implements
CartAdapter.OnCartItemRemovedListener {

    private ListView listView;
    private TextView tvTotalPrice;
    private List<RequestPojo> packageList;
    private CartAdapter adapter;
    private Button btnProceedPayment;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_cart, container, false);

        // Initialize views
```

```
        listView = root.findViewById(R.id.viewProductlistCart);
        tvTotalPrice = root.findViewById(R.id.tv_total_price);
        btnProceedPayment = root.findViewById(R.id.btn_proceed_payment);

        // Initialize list and adapter
        packageList = new ArrayList<>();
        adapter = new CartAdapter(requireActivity(), packageList, this);
        listView.setAdapter(adapter);

        // Fetch Cart Data
        getCart();
        btnProceedPayment.setOnClickListener(v -> proceedToPayment());

        return root;
    }

    /**
     * Fetch Cart Data from Server
     */
    private void getCart() {
        RequestQueue queue = Volley.newRequestQueue(requireContext());

        StringRequest request = new StringRequest(Request.Method.POST,
Utility.SERVERUrl, response -> {
            if (!response.trim().equals("failed")) {
                Gson gson = new Gson();
                packageList.clear();
                packageList.addAll(Arrays.asList(gson.fromJson(response, RequestPojo[].class)));
                adapter.notifyDataSetChanged();
                calculateTotalPrice();
            } else {
                Toast.makeText(getContext(), "No Data Found in Cart",
Toast.LENGTH_SHORT).show();
```

```
            tvTotalPrice.setText("₹0.00");
        }
    }, error -> {
        Toast.makeText(getContext(), "Error: " + error.getMessage(),
Toast.LENGTH_SHORT).show();
        tvTotalPrice.setText("₹0.00");
    }) {
        @Override
        protected Map<String, String> getParams() {
            SharedPreferences prefs = requireContext().getSharedPreferences("sharedData",
MODE_PRIVATE);
            final String reg_id = prefs.getString("reg_id", "No_id");
            Map<String, String> map = new HashMap<>();
            map.put("key", "viewProductCart");
            map.put("reg_id", reg_id);
            return map;
        }
    };

    queue.add(request);
}

/**
 * Calculate Total Price from Cart Items
 */
public void calculateTotalPrice() {
    if (packageList == null || packageList.isEmpty()) {
        tvTotalPrice.setText("₹0.00");
        return;
    }

    double totalPrice = 0.0;
    for (RequestPojo item : packageList) {
```

```
            try {
                totalPrice += Double.parseDouble(item.getPr_price());
            } catch (NumberFormatException e) {
                Toast.makeText(getContext(), "Invalid price format for item: " +
item.getPr_name(), Toast.LENGTH_SHORT).show();
            }
        }

        tvTotalPrice.setText("₹" + String.format("%.2f", totalPrice));
    }

    /**
     * Remove Item from Cart List Dynamically
     */
    @Override
    public void onCartItemRemoved(int position) {
        if (position >= 0 && position < packageList.size()) {
            packageList.remove(position);
            adapter.notifyDataSetChanged();
            calculateTotalPrice();
        }
    }


    private void proceedToPayment() {
        SharedPreferences prefs = requireContext().getSharedPreferences("sharedData",
MODE_PRIVATE);
        String userId = prefs.getString("reg_id", "No_id");
        String totalPrice = tvTotalPrice.getText().toString().replace("₹", "").trim();

        Intent intent = new Intent(getContext(), Payment.class);
        intent.putExtra("user_id", userId);
        intent.putExtra("total_price", totalPrice);
```

```
        startActivity(intent);
    }
}
```

Payment Fragment

```
package com.example.scanandgo.User;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

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

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.scanandgo.COMMON.LoginActivity;
import com.example.scanandgo.COMMON.Utility;
import com.example.scanandgo.R;
```

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

public class Payment extends AppCompatActivity {

    private TextView tvUserId, tvTotalPrice;
    private EditText etCardNumber, etCardHolder, etExpiryDate, etCVV;
    private Button btnPayNow;

    String totalPrice;
    String userId;

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

        // Toolbar Setup
        Toolbar toolbar = findViewById(R.id.toolbar_payment);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Initialize Views
        tvUserId = findViewById(R.id.tv_user_id);
        tvTotalPrice = findViewById(R.id.tv_total_price);
        etCardNumber = findViewById(R.id.et_card_number);
        etCardHolder = findViewById(R.id.et_card_holder);
        etExpiryDate = findViewById(R.id.et_expiry_date);
        etCVV = findViewById(R.id.et_cvv);
        btnPayNow = findViewById(R.id.btn_pay_now);

        // Get data from Intent
        userId = getIntent().getStringExtra("user_id");
```

```
        totalPrice = getIntent().getStringExtra("total_price");

        tvUserId.setText("User ID: " + userId);
        tvTotalPrice.setText("Total Price: ₹" + totalPrice);

        // Handle Pay Now Click
        btnPayNow.setOnClickListener(v -> processPayment());
    }

    private void processPayment() {
        String cardNumber = etCardNumber.getText().toString().trim();
        String cardHolder = etCardHolder.getText().toString().trim();
        String expiryDate = etExpiryDate.getText().toString().trim();
        String cvv = etCVV.getText().toString().trim();

        if (cardNumber.isEmpty() || cardHolder.isEmpty() || expiryDate.isEmpty() ||
cvv.isEmpty()) {
            Toast.makeText(this, "Please fill all fields", Toast.LENGTH_SHORT).show();
            return;
        }

        if (cardNumber.length() < 16 || cvv.length() < 3) {
            Toast.makeText(this, "Invalid Card Details", Toast.LENGTH_SHORT).show();
            return;
        }


        paymentVolley( cardNumber, cardHolder, expiryDate, cvv,totalPrice);

        // Simulate Payment Success
//
//
//        // Optionally, navigate back or clear the cart
```

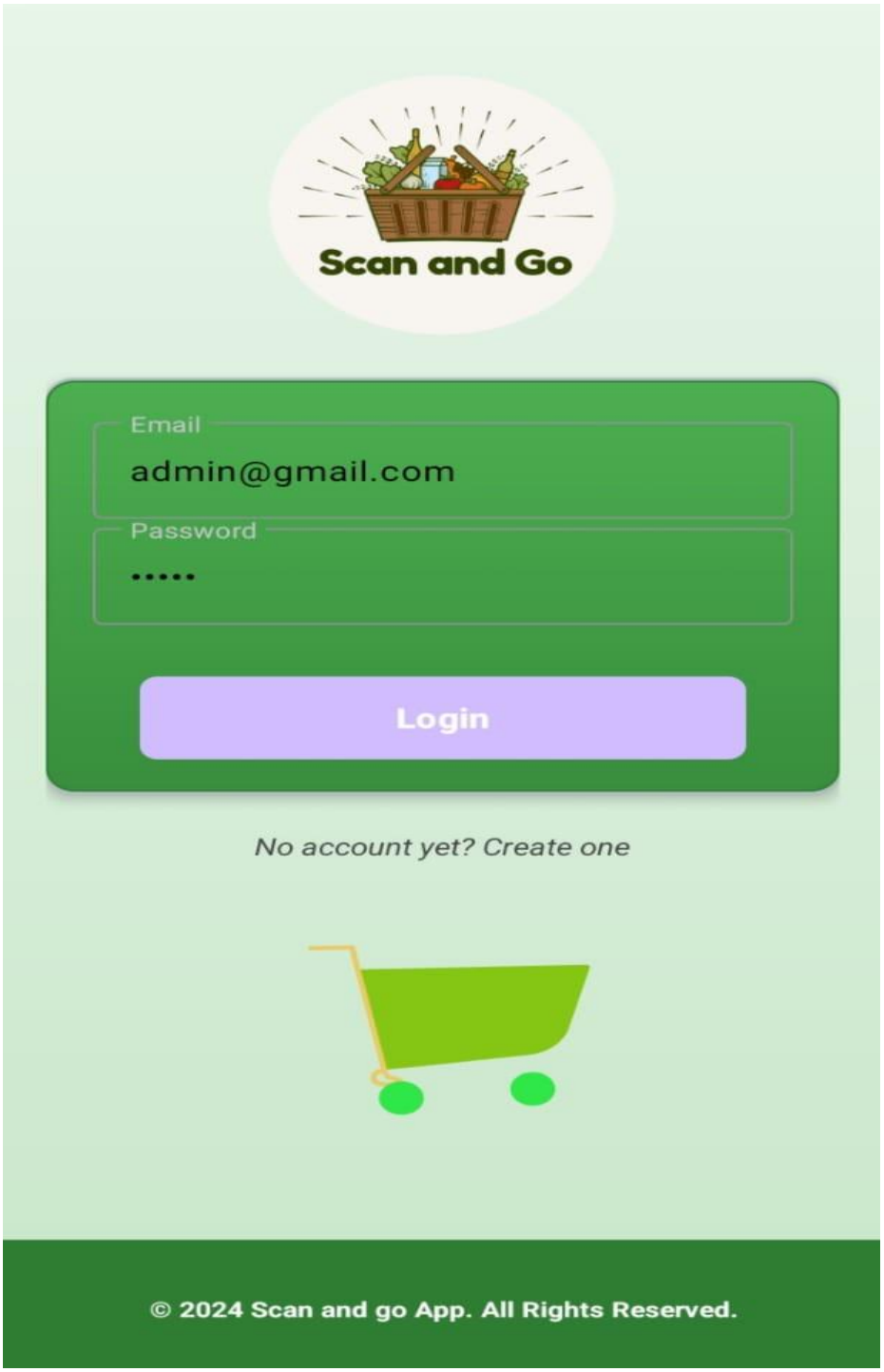
```
//        finish();
    }

    private void paymentVolley(String cardNumber, String cardHolder, String expiryDate,
String cvv, String totalPrice)
    {
        RequestQueue queue = Volley.newRequestQueue(getApplicationContext());

       StringRequest request = new StringRequest(Request.Method.POST,
Utility.SERVERUrl, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
               if (!response.trim().equals("failed")) {

                  Toast.makeText(Payment.this, "Payment Successful!
Toast.LENGTH_LONG).show();
                   startActivity(new Intent(getApplicationContext(), UserHome.class));



               } else {
                  Toast.makeText(getApplicationContext(), " Failed",
Toast.LENGTH_SHORT).show();
               }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {

               Toast.makeText(getApplicationContext(), "my error :" + error,
Toast.LENGTH_LONG).show();
               Log.i("My error", "" + error);
            }
        }) {
```

```
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {


            SharedPreferences prefs =
getApplicationContext().getSharedPreferences("sharedData", MODE_PRIVATE);
            final String uid = prefs.getString("reg_id", "No_idd");//"No name defined" is the
default value.
            final String type = prefs.getString("type", "Notype");
            Map<String, String> params = new HashMap<>();



            params.put("key", "addPayment");

            params.put("cardNumber", cardNumber);
            params.put("cardHolder", cardHolder);
            params.put("expiryDate", expiryDate);
            params.put("cvv", cvv);
            params.put("totalPrice", totalPrice);
            params.put("uid", uid);

            return params;
        }
    };
    queue.add(request);
  }

  @Override
  public boolean onSupportNavigateUp() {
    onBackPressed();
    return true;
  }
```

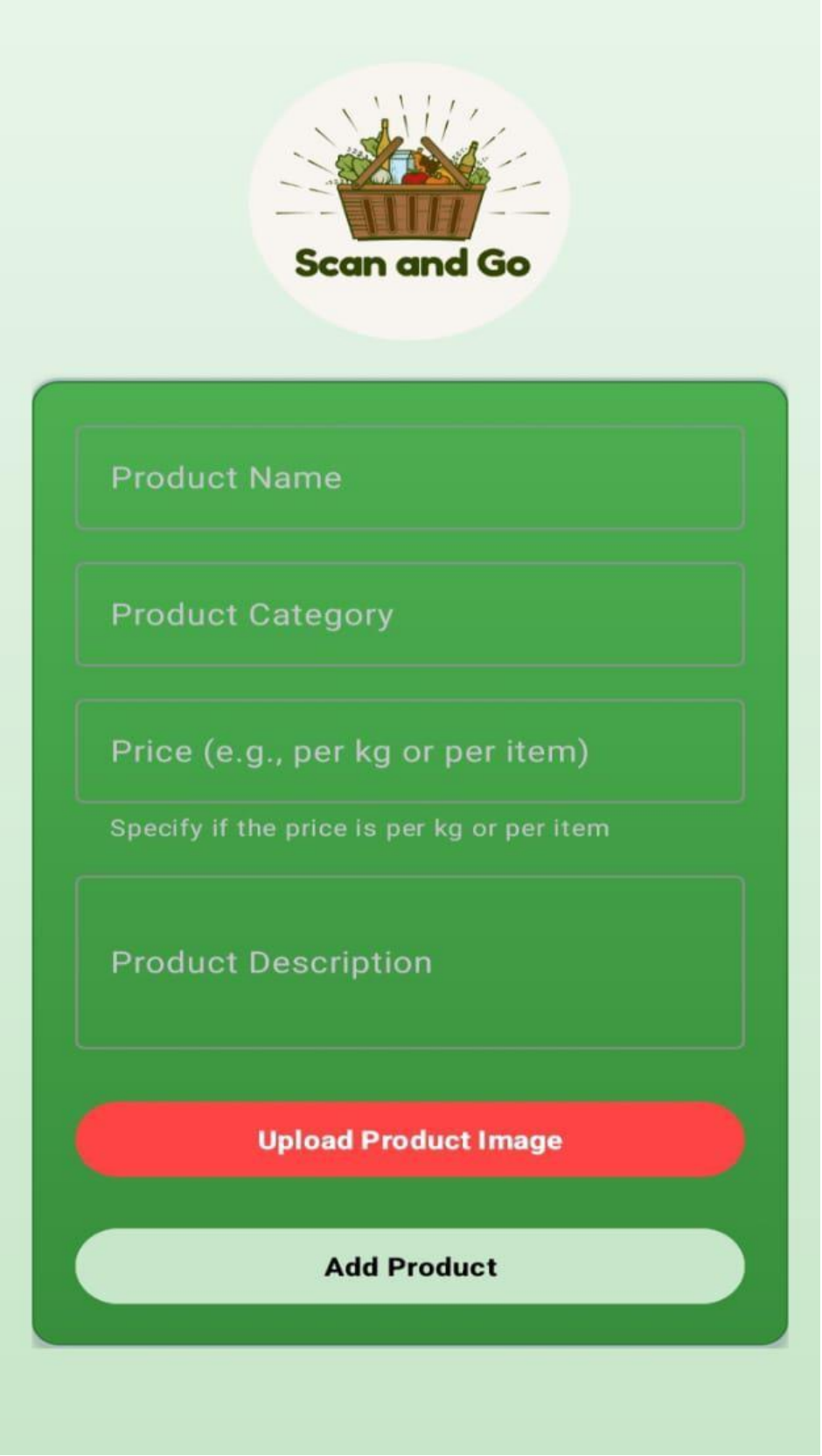
```
    @Override
    public void onBackPressed() {
        new AlertDialog.Builder(this)
                .setTitle("Logout Confirmation")
                .setMessage("Do you want to logout and return to the login page?")
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                        startActivity(intent);
                        finish();
                    }
                })
                .setNegativeButton("No", null)
                .setCancelable(false)
                .show();
    }
}
```

8.APPENDIX

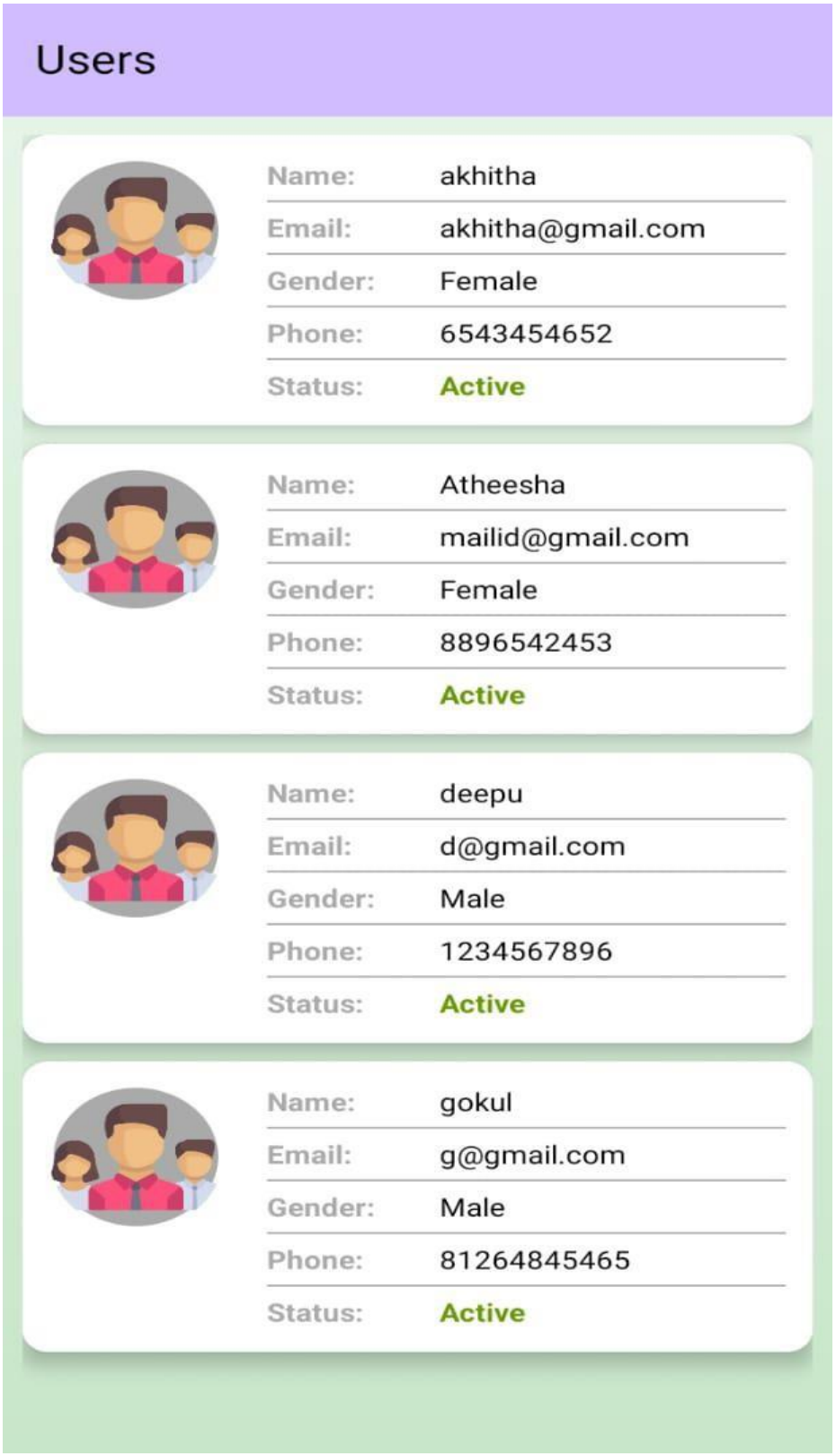
Admin login


Scan and Go
Email
admin@gmail.com
Password
.....
Login
No account yet? Create one
© 2024 Scan and go App. All Rights Reserved.

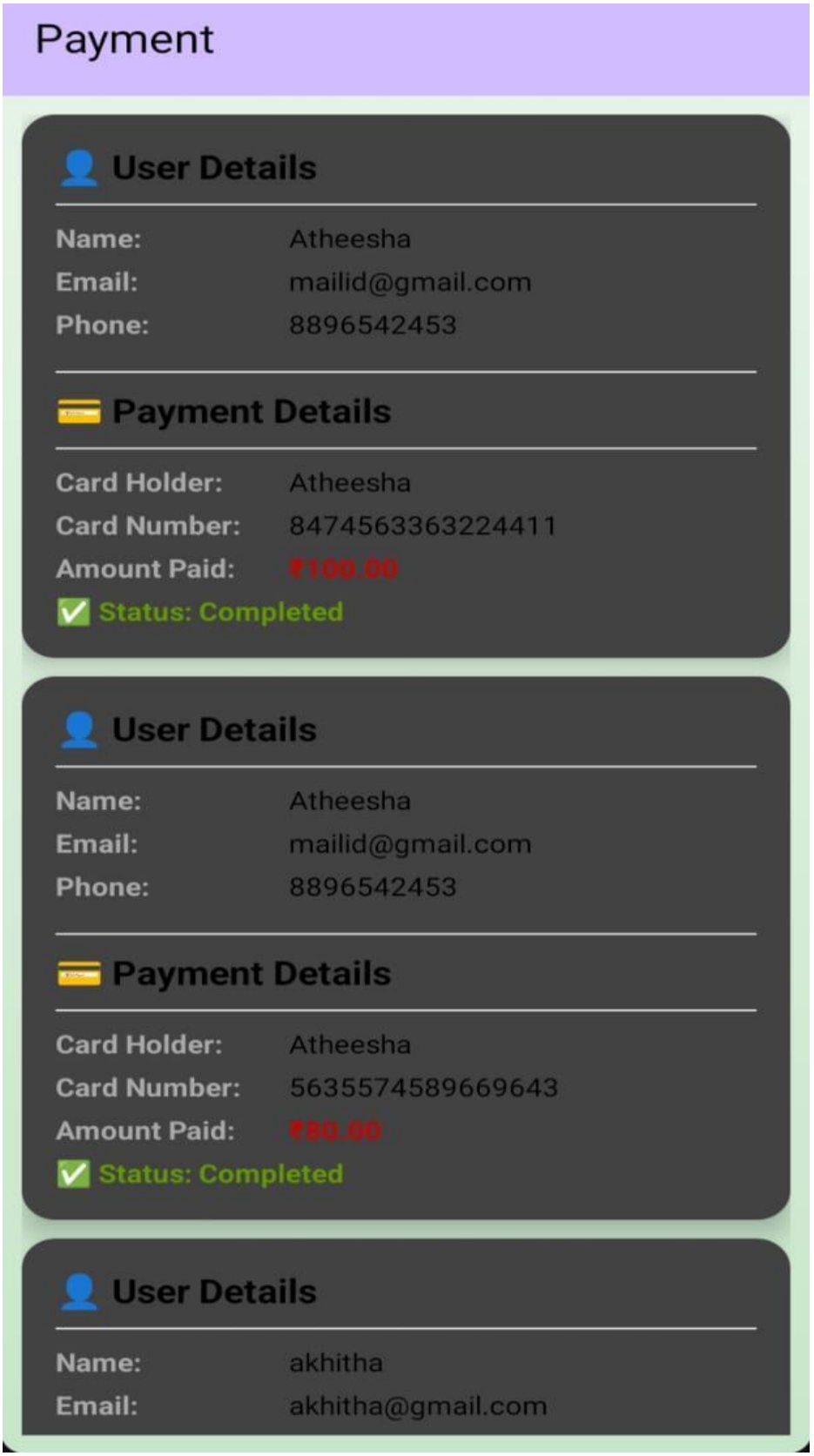
Adding products to the app



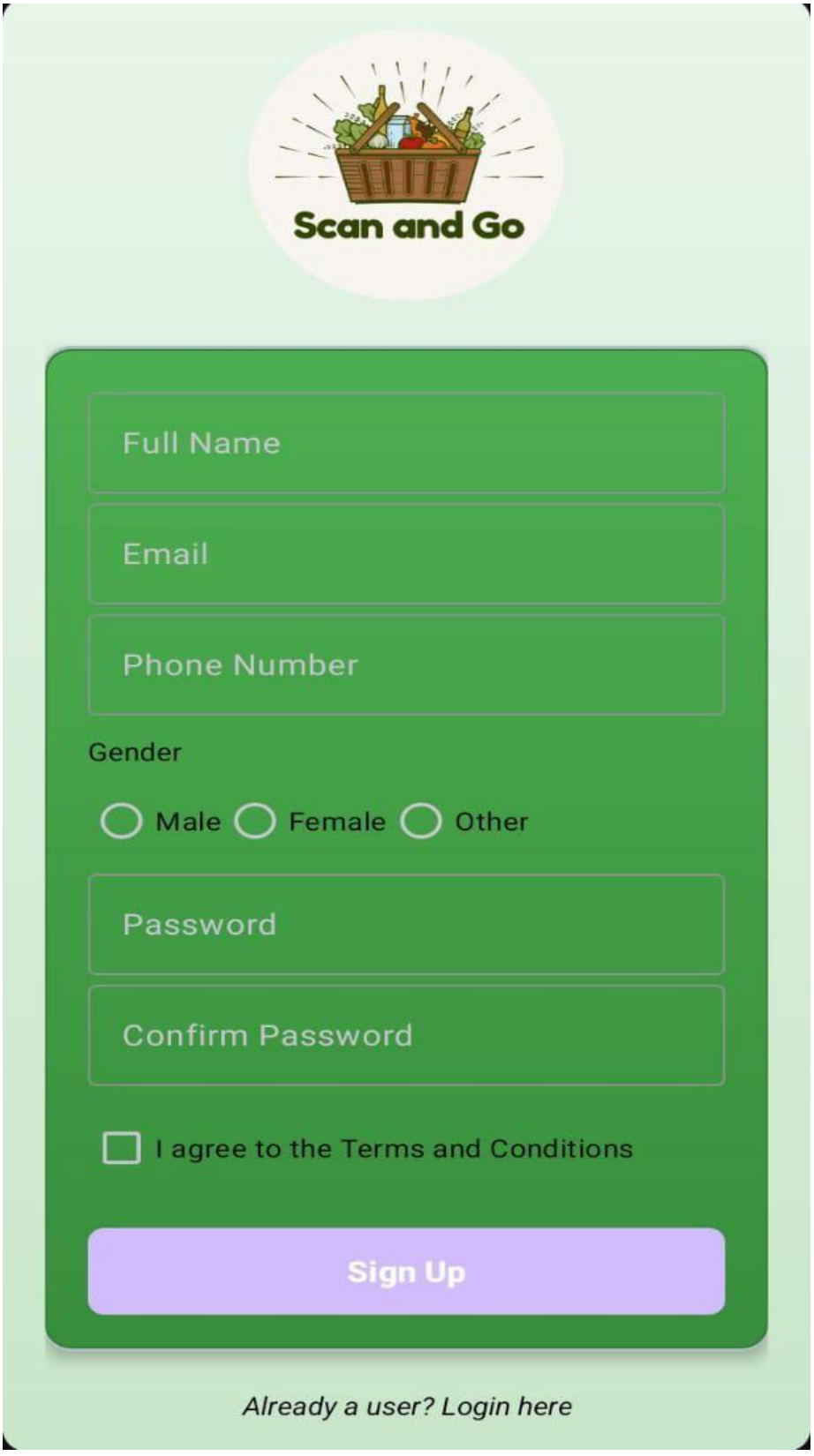
User's login details



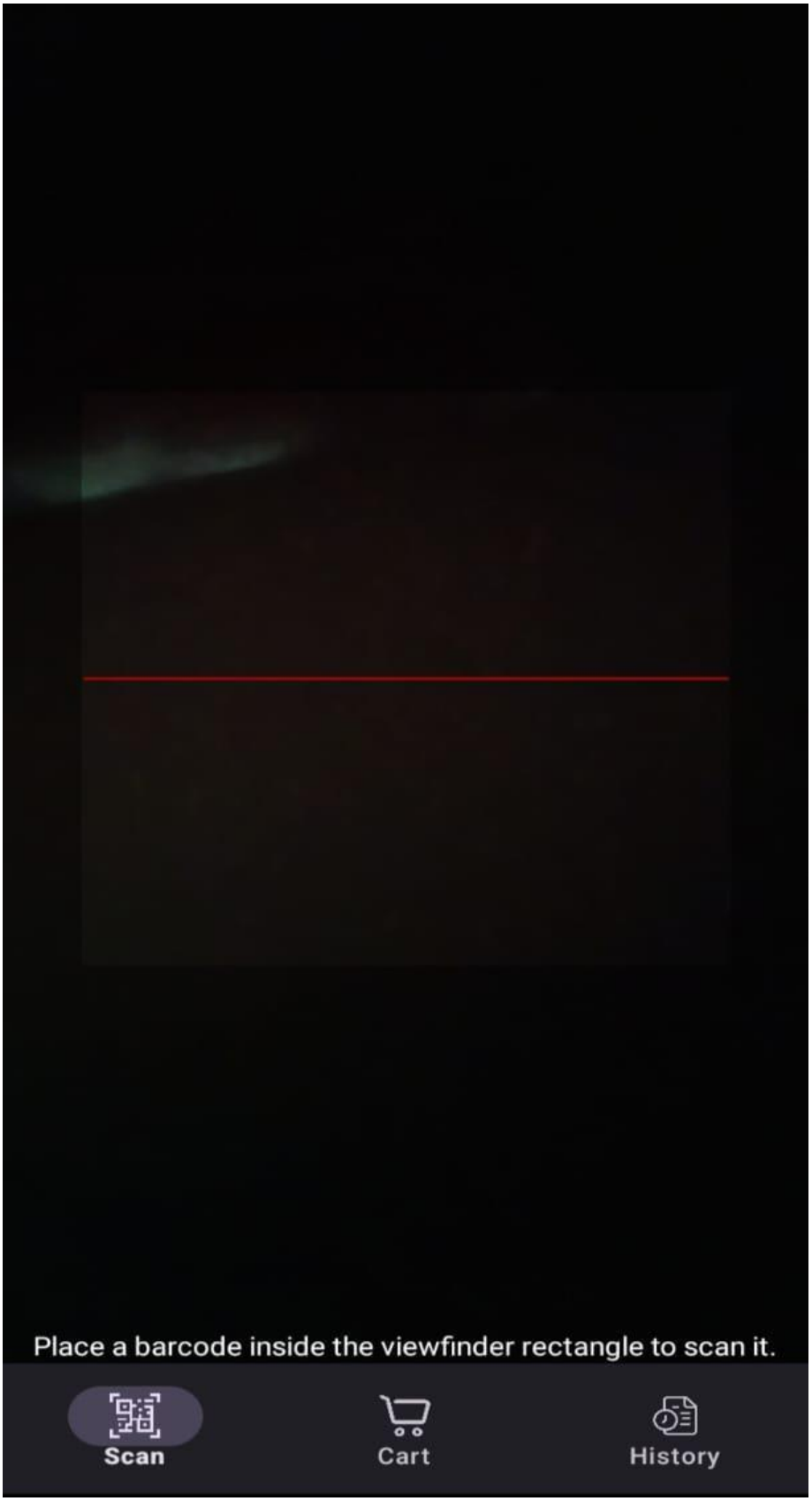
User's payment details



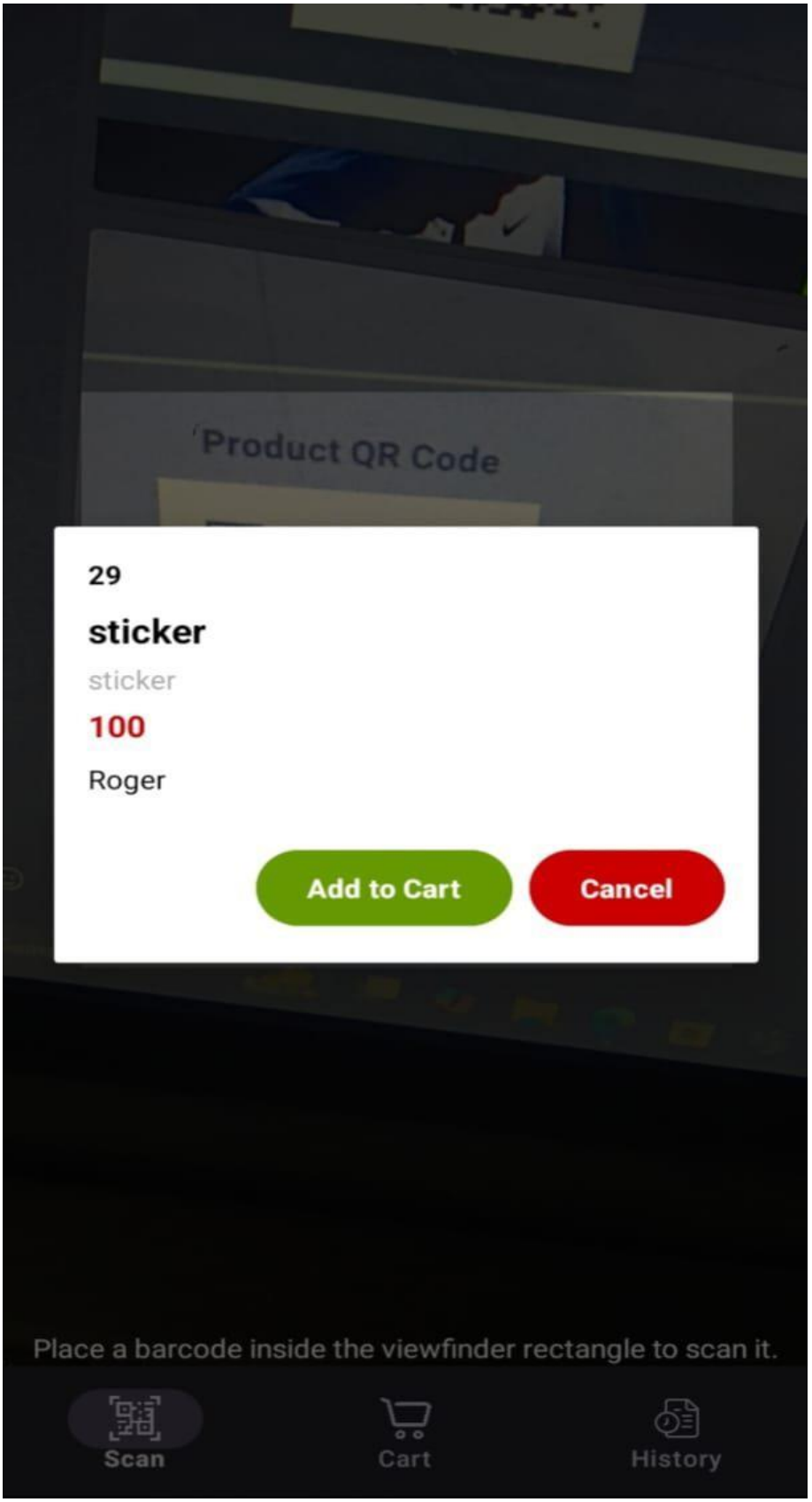
User Registration



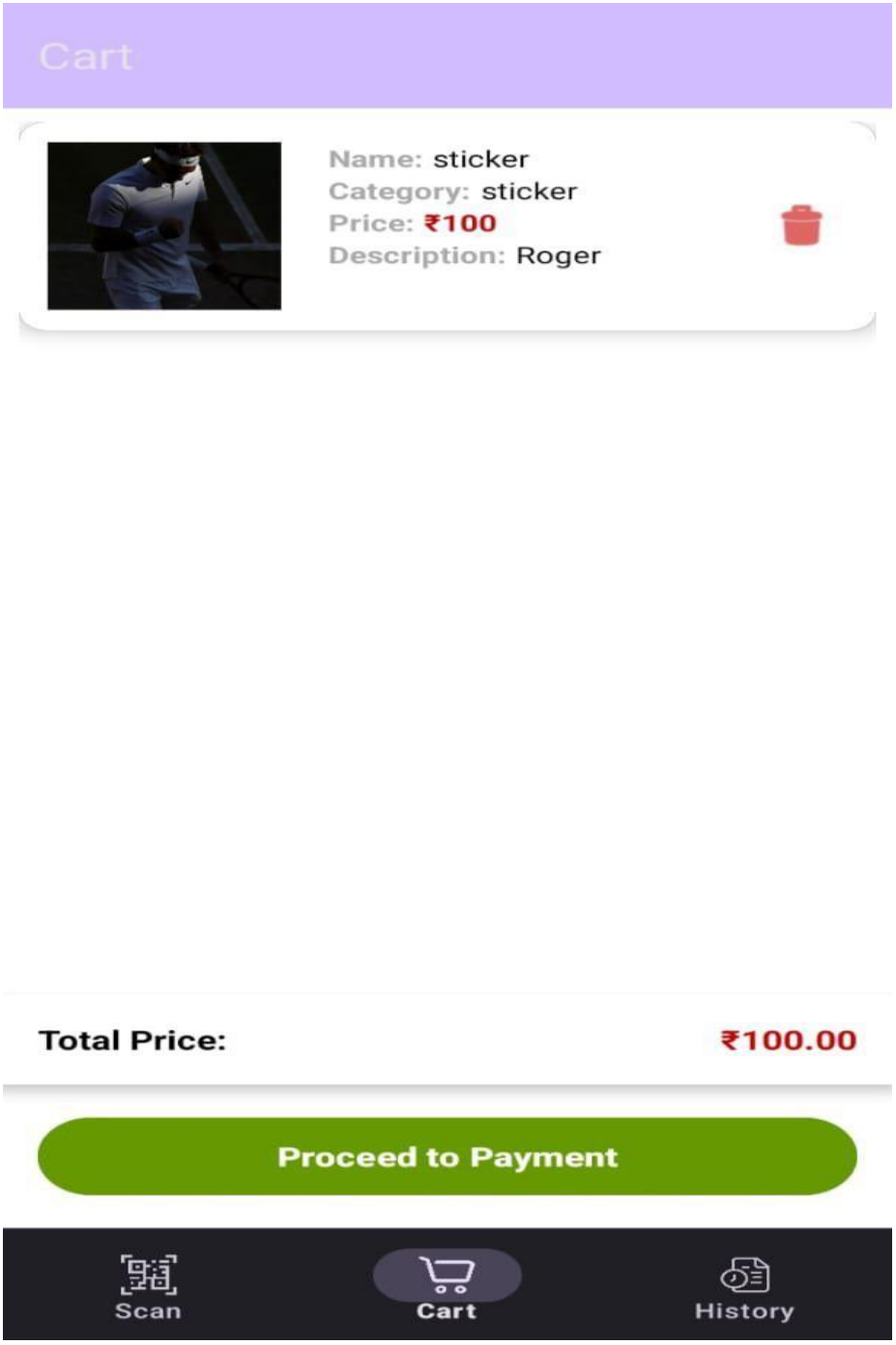
Product QR code Scanning



Scanned product details



Cart



- By clicking proceed to payment it leads to an interface which the customer should select the payment option and enter the card details.

- We Cannot provide that interface here , because of the security guidelines in this app.

9.BIBLOGRAPHY

9.1REFERENCE

1. https://ieeexplore.ieee.org/document/10150048
2. https://www.logicerp.com
3. https://www.slideshare.net/slideshow/self-checkout-application-report/240892706
4. https://www.intel.com/content/www/us/en/developer/articles/reference-implementation/automated-self-checkout.html
5. https://ieeexplore.ieee.org/document/10150048