RV UNIVERSITY, BENGALURU

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A Project Report On

IoT-Based Smart Pricing and Shopping Assistant

B.Tech(Honors)

In

School of Computer Science and Engineering

Submitted By

Team Member 01: 1RUA24CSE0034_ ALFIYA NAAZ
Team Member 02: 1RUA2CSE0055 _ANITA
Team Member 03: 1RUA24CSE0109 _ CHINMAYEE.N.V
Team Member 04: 1RUA24CSE0114 _ DANYATHA.Y.K

Course Name: Internet of Things [CS2404]

Under the Guidance of

Prof.Aishwarya Singh Gautam
Assistant Professor
School of CSE
RV University, Bengaluru-560059

2025-2026

RV UNIVERSITY, BENGALURU-59

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work titled IoT-Based Smart Pricing and Shopping Assistant is carried out by **ALFIYA NAAZ** (1RUA24CSE0034), **ANITA** (1RUA24CSE0055), **CHINMAYEE.N.V** (1RUA24CSE0109) and **DANYATHA.Y.K** (1RUA24CSE0114) RV University, Bengaluru, **B.Tech (Hons) in the School of Computer Science and Engineering** of the RV University, Bengaluru during the year 2025-2026. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the project report. The Project report has been approved as it satisfies the academic requirements in respect of project work prescribed by the institution.

Signature of Guide

External Viva:

Name of Examiners **Signature with Date**

1

2

RV UNIVERSITY, BENGALURU-59

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Alfia Naaz, Anita, Chinmayee N V and Danyatha Y K students of second semester B.Tech(Hons), SoCSE, RV University, Bengaluru, hereby declare that the project titled IoT-Based Smart Pricing and Shopping Assistant has been carried out by us and submitted in partial fulfillment of **Bachelor of Technology(Hons)** in **School of Computer Science and Engineering** during the year 2025-26.

Further, we declare that the content of the report has not been submitted previously by anybody or to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RV University will be the property of RV University, Bengaluru, and we will be one of the authors of the same.

Place: Bengaluru
Date:

Name **Signature**

1. ALFIYA NAAZ (1RUA24CSE0034)

2. ANITA (1RUA24CSE0055)

3. CHINMAYEE.N.V (1RUA24CSE0109)

4. DANYATHA.Y.K (1RUA24CSE0114)

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project.

First, we take this opportunity to express our sincere gratitude to the School of Computer Science and Engineering, RV University, for providing us with a great opportunity to pursue our bachelor's degree in this institution.

A special thanks to our Program Director, **……., and** Dean - **Dr.Shobha G,** for their continuous support and providing the necessary facilities with guidance to carry out mini project work.

We would like to thank our guide, Prof**, Aishwarya Singh Gautam, Assistant Professor**, School of Computer Science and Engineering, RV University, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in the Project work.

Signature of Student

USN:

Abstract

This project introduces an IoT-Based Smart Pricing and Shopping Assistant that enhances the retail experience for both customers and retailers. The system enables real-time price comparison, budget tracking, and personalized recommendations through Economy, Value, and Premium modes. Retailers benefit from features such as dynamic bundling, expiry-based promotions, and sponsored deals, which improve sales and reduce wastage. By combining IoT technology with smart pricing strategies, the solution creates a balanced and efficient shopping ecosystem that ensures customer savings while maximizing retailer profitability

Table of Contents:

List of Tables and Figures

Chapter 1: Introduction

1.1 General Introduction

The **retail industry** is expanding at an incredible pace, driven by new technologies and changing consumer behaviors. However, this rapid growth has also brought several challenges for both **customers** and **retailers** that affect the overall shopping experience and business performance.

From the **customer's perspective**, shopping has become more complex. With countless brands, stores, and offers available, customers often struggle to **compare prices** across different outlets, **track their spending**, and **make quick, informed decisions** about what to buy. Many also face difficulties in managing their budgets during shopping trips, especially when prices fluctuate frequently or when attractive discounts tempt them to overspend.

On the **retailer's side**, the challenges are equally demanding. Retailers need to find ways to **attract and retain customers**, **increase sales**, and **maintain efficient inventory management**. They must ensure that products are neither overstocked nor understocked, as both can lead to financial losses. Furthermore, effectively promoting **special offers, discounts, and bundled products** is a constant struggle, especially when trying to target the right audience at the right time.

To address these issues, the integration of **Internet of Things (IoT)** technology with **smart pricing mechanisms** provides a modern and effective solution. IoT enables the connection of physical retail components — such as shelves, products, carts, and scanners — through smart sensors and wireless networks. These devices can continuously collect and exchange real-time data on product availability, pricing, customer movement, and purchase behavior.

An **IoT-based Smart Pricing and Shopping Assistant** utilizes this real-time data along with **intelligent algorithms** to enhance the overall retail experience. For **customers**, the system offers several key benefits:

It enables **live price comparisons** across different stores or online platforms, helping shoppers identify the best deals instantly.

It provides **personalized product recommendations** based on their preferences, shopping history, and budget constraints.

It assists in **budget monitoring**, allowing customers to see how much they have spent and how much remains within their planned limit, thus preventing overspending.

It speeds up **decision-making**, as customers can rely on accurate, data-driven insights to choose products that offer the best value for money.

For **retailers**, the advantages are equally significant:

The system can **automatically adjust prices** in real time based on factors like demand, stock levels, and expiry dates, ensuring optimal pricing and minimizing losses.

It supports **efficient inventory management**, helping retailers track product movement, anticipate restocking needs, and reduce wastage due to unsold or expired goods..

1.2 Literature Survey

Over the years, many researchers have explored ways to enhance the shopping experience by using Internet of Things (IoT) technology. Different approaches have been tried to make shopping more convenient, efficient, and smart for both customers and retailers.

For example, some studies introduced IoT-enabled smart shopping carts that automatically scan items as customers place them in the cart, helping to reduce waiting time at billing counters and automate the checkout process. This not only saves time but also improves the overall shopping flow.

Other research efforts have focused on real-time product information systems, where sensors and connected devices provide instant details about product features, prices, discounts, or availability. These systems aim to make shopping more transparent and help customers make informed purchase decisions.

Some IoT-based solutions target inventory and stock management, helping retailers monitor stock levels, track expiry dates, and prevent wastage by managing perishable goods more effectively. These systems contribute to reducing financial losses and improving operational efficiency.

In addition, several studies have emphasized personalized recommendation systems that analyze customer preferences, purchase history, and behavior to suggest products that align with individual needs. Such systems enhance customer satisfaction and loyalty by offering a more engaging and tailored shopping experience.

However, despite these advancements, most of the existing solutions tend to focus on only one side of the retail ecosystem — either improving customer convenience or maximizing retailer profitability. Very few have tried to strike a balance between the two.

This project aims to bridge that gap by developing an IoT-based Smart Pricing and Shopping Assistant that integrates both perspectives. It combines customer-focused features, such as budget monitoring, real-time price comparison, and personalized recommendations, with retailer-oriented features, including dynamic bundling of products, smart discounting, and expiry-based promotions.

1.3 Problem Statement / Objectives

Problem Statement:
Consumers often face confusion while shopping due to difficulties in comparing per-unit prices, monitoring their expenses, and identifying best-value products. On the other hand, retailers struggle with promoting bundled offers, minimizing losses from near-expiry products, and maintaining a balance between affordability and profitability. Current solutions do not effectively address both consumer and retailer requirements simultaneously.

Objectives:

- To design an IoT-based assistant that provides real-time price comparison and budget tracking for consumers.
- To implement recommendation modes (Economy, Value, and Premium) that cater to diverse consumer preferences.
- To enable retailers to promote dynamic bundles, manage expiry-based offers, and gain additional revenue through sponsored recommendations.

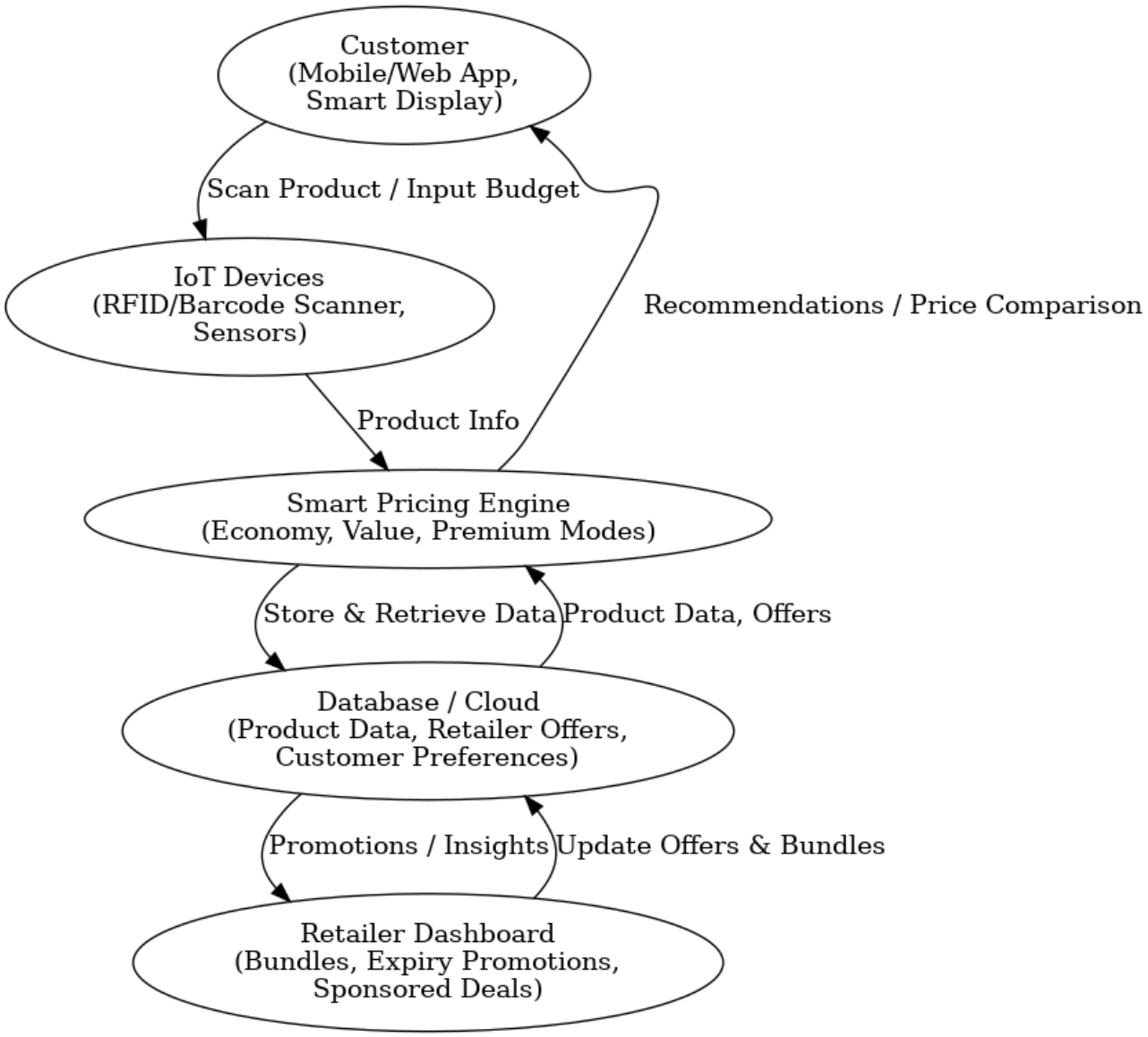
- To create a cost-effective and scalable system that ensures both customer satisfaction and retailer profitability.

Chapter 2: System Design

2.1 Architectural Diagram

The architecture of the IoT-Based Smart Pricing and Shopping Assistant consists of three main layers:

1. **Input Layer (Sensors & Data Collection):**
    - ESP 32 -cam module to capture product details.
    - IoT sensors and databases fetch product prices, expiry dates, and stock availability.
2. **Processing Layer (Smart Pricing Engine):**
    - The system processes inputs using algorithms for price comparison, budget tracking, and recommendation generation.
    - It includes three modes of recommendation:
        - **Economy Mode:** Suggests the lowest-cost options.
        - **Value Mode:** Balances affordability with retailer profit.
        - **Premium Mode:** Suggests high-quality, branded products.
3. **Output Layer (User & Retailer Interface):**
    - Customers receive real-time updates on prices, shopping cart total, and recommended products via a mobile/web interface or smart display.
    - Retailers get insights on expiry-based discounts, bundle suggestions, and sponsored promotions to optimize sales.

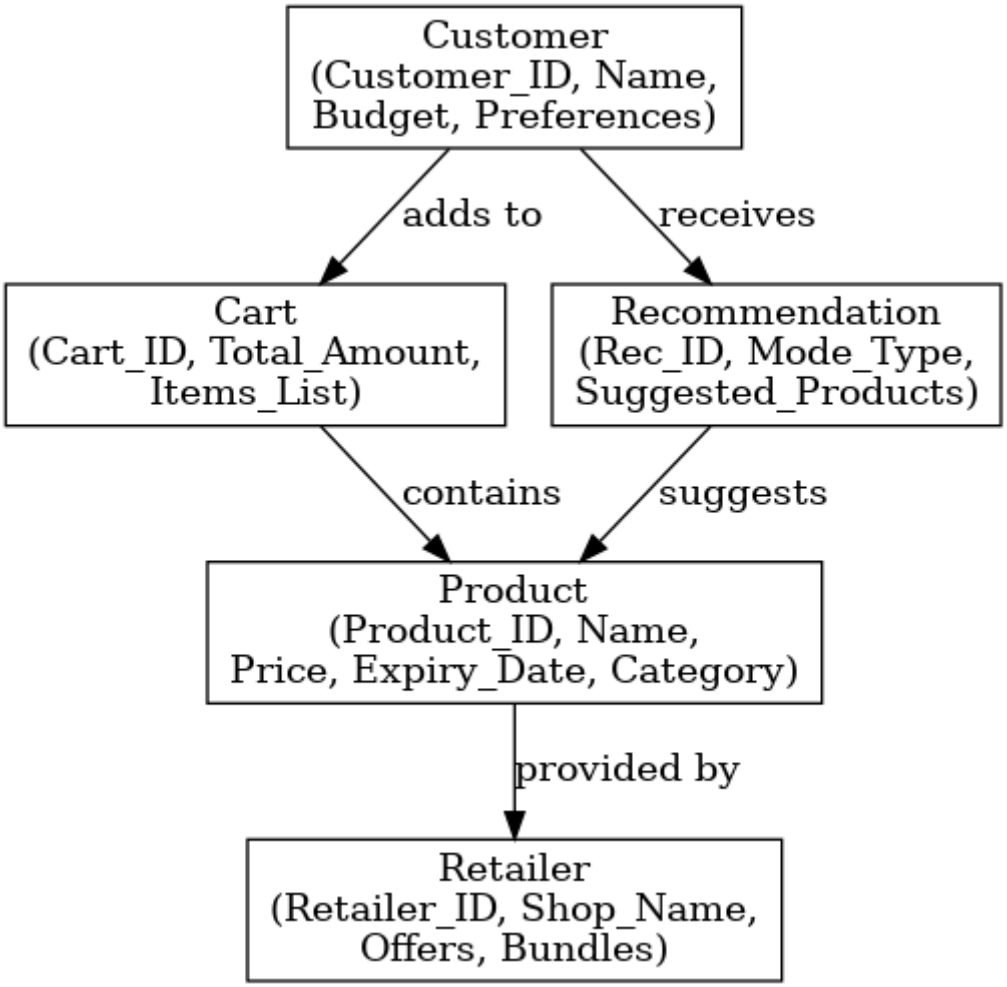


2.2 ER Modelling / ER Schema Diagram

The **Entity-Relationship (ER) Model** is designed to represent the key entities and their relationships in the system:

- **Entities:**
    - **Customer** (Customer_ID, Name, Budget, Preferences)
    - **Product** (Product_ID, Name, Price, Expiry_Date, Category)
    - **Retailer** (Retailer_ID, Shop_Name, Offers, Bundles)
    - **Cart** (Cart_ID, Total_Amount, Items_List)
    - **Recommendation** (Rec_ID, Mode_Type, Suggested_Products)

- **Relationships:**
    - A **Customer** selects multiple **Products** into a **Cart**.
    - Each **Product** belongs to a **Retailer**.
    - **Recommendations** are generated for each **Customer** based on **Products** and **Cart** contents.
    - **Retailers** provide promotional offers linked with **Products**.



Chapter 3: Software Requirements

3.1 Functional Requirements

The system must provide the following core functionalities:

1. Allow customers to scan products using ESP 32 cam module.
2. Retrieve product details (name, price, expiry date, category) in real-time.
3. Compare per-unit prices and display best-value recommendations.

4. Support three recommendation modes: Economy, Value, and Premium.
5. Track customer budget and provide alerts if the limit is exceeded.
6. Generate dynamic bundle suggestions and expiry-based promotions for retailers.
7. Enable retailers to update product offers and promotions through a dashboard.

3.2 Non-Functional Requirements

1. **Performance:** The system should respond to customer queries within 2 seconds.
2. **Scalability:** Must handle multiple users and large product databases simultaneously.
3. **Reliability:** Ensure consistent functioning with minimal downtime.
4. **Usability:** Provide a simple and intuitive interface for both customers and retailers.
5. **Security:** Protect customer data, retailer data, and transaction details.
6. **Portability:** The system should run on different devices (mobile, tablets, kiosks).

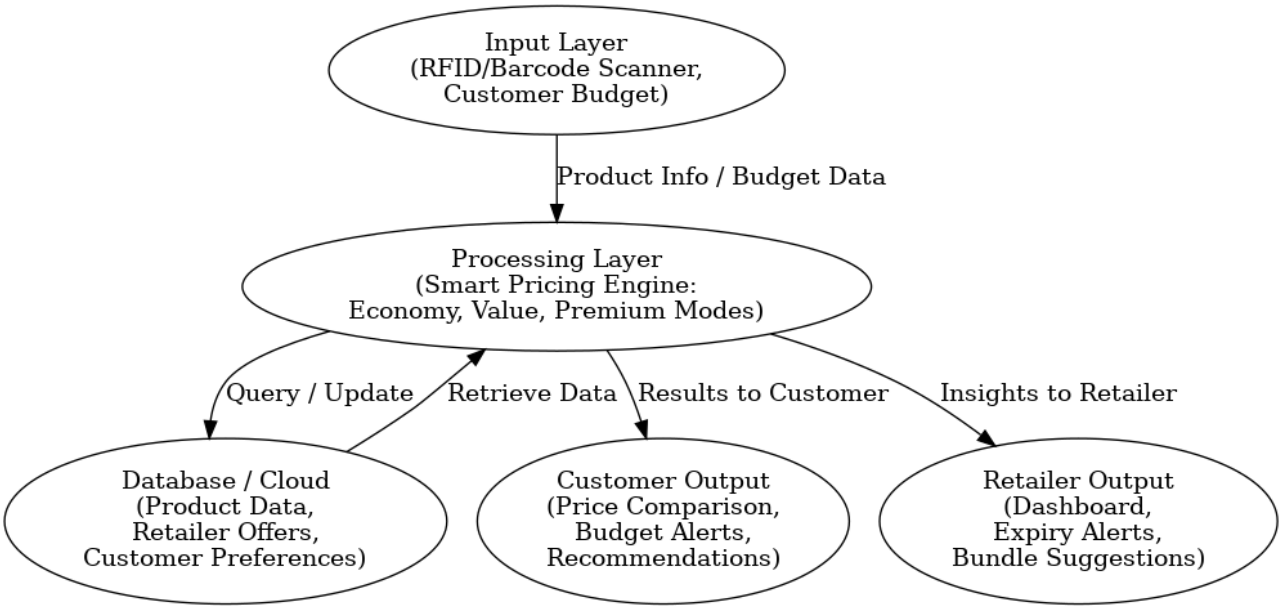
3.3 Hardware Requirements

- ESP - 32 cam module
- IoT-enabled device (Raspberry Pi / Arduino / ESP32)
- Cloud Server or Local Database
- Display unit (Mobile device, Tablet, or Smart Screen)
- Retailer Dashboard PC/Laptop

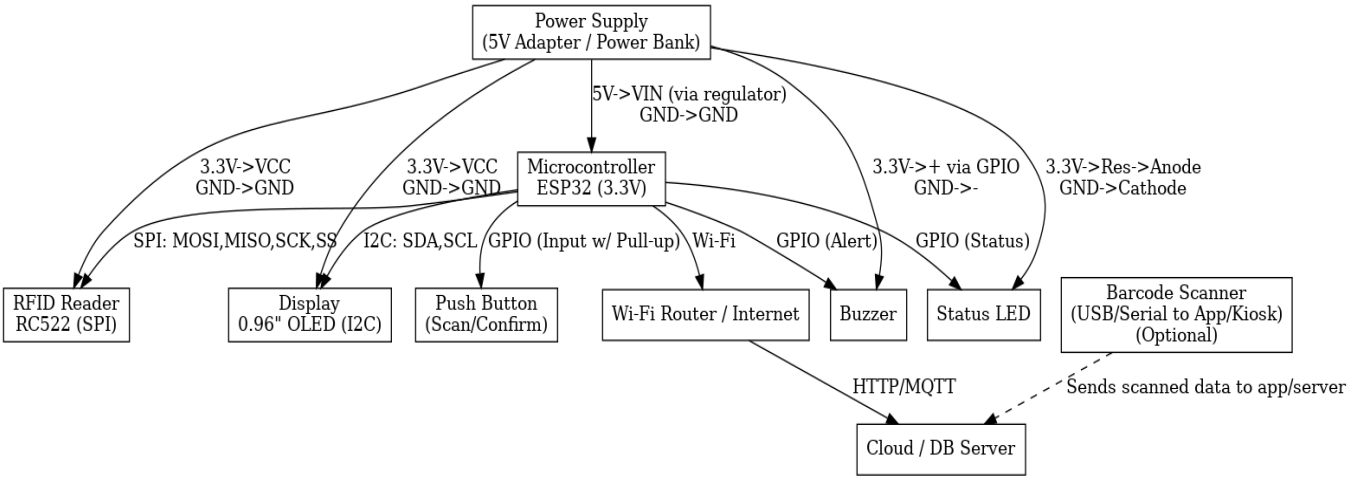
3.4 Software Requirements

- Operating System: Windows/Linux
- Programming Language: Python / C / Embedded C (for IoT device)
- Database: MySQL / Firebase / MongoDB
- IoT Middleware: MQTT or HTTP Protocols
- Frontend: HTML, CSS, JavaScript (or Mobile App Framework)
- Backend: Flask/Django (Python) or Node.js

3.5. Working Principle/ Block Diagram



3.6. Circuit Diagrams and Screenshots



3.7 Code Explanation

The code for the IoT-Based Smart Pricing and Shopping Assistant is divided into several modules for clarity and functionality:

1. **Product Scanning Module**
    - Reads product data from ESP – 32 cam module .
    - Sends the product ID to the microcontroller.

2. **Database & Cloud Communication Module**
    - Connects to the Cloud Database (MySQL/Firebase/MongoDB) via Wi-Fi using HTTP/MQTT protocol.
    - Fetches product details: name, price, expiry date, and category.
3. **Smart Pricing Engine Module**
    - Compares per-unit prices of similar products.
    - Tracks customer budget and alerts if exceeded.
    - Generates recommendations in Economy, Value, or Premium modes.
4. **Output & Alert Module**
    - Displays recommendations, price details, and budget status on the OLED/mobile app.
    - Provides audio (buzzer) or visual (LED) alerts when budget crosses the limit.
5. **Retailer Dashboard Module**
    - Retailers can log in to update offers, expiry discounts, and sponsored deals.
    - Dynamic bundles are generated automatically to improve sales.

Flow of Execution:

- Customer scans product → ESP – 32 cam module reads Product_ID → Controller sends to Cloud → Database returns product details → Smart Pricing Engine processes → Recommendation displayed to customer and updates retailer dashboard.

3.8 Summary

This chapter detailed the functional and non-functional requirements of the IoT Smart Pricing and Shopping Assistant, along with the necessary hardware and software specifications. The system workflow was explained using a block diagram and supported by a circuit design for practical implementation. The code modules were discussed to highlight the logical structure of the project, covering product scanning, database communication, smart pricing engine, output display, and retailer dashboard. Overall, this chapter bridges the gap between theoretical design and practical implementation, laying the foundation for the next phase of development and testing.

Chapter 4: Implementation and Testing

4.1 Modules

4.1.1 Descriptions of Modules

1. **Product Scanning Module**
    - ESP 32 – cam module to capture product details.
    - Sends product ID to the microcontroller for further processing.
2. **Database & Cloud Module**
    - Manages product information (name, price, expiry date, category).
    - Connects via Wi-Fi (MQTT/HTTP) to retrieve data in real time.
3. **Smart Pricing Engine Module**
    - Implements recommendation logic (Economy, Value, Premium modes).
    - Tracks budget and provides alerts when limits are exceeded.
4. **Customer Interface Module**
    - Displays product info, price comparisons, and recommendations on OLED/mobile app.
    - Provides alerts (buzzer/LED) for overspending.
5. **Retailer Dashboard Module**
    - Allows retailers to update offers, bundles, and expiry-based promotions.
    - Provides insights for maximizing sales and reducing wastage.

4.2 Testing

4.2.1 Test Cases and Results

Test Case	Input	Expected Output	Actual Result	Status
Scan product	Product ID via ESP-32 cam	Display product name & price	Displayed correctly	Pass

Budget alert	Add items exceeding set budget	Alert buzzer/LED & message	Alert triggered	Pass
Recommendation (Economy Mode)	Select mode = Economy	Show cheapest option	Correctly displayed	Pass
Recommendation (Premium Mode)	Select mode = Premium	Show branded/high-quality option	Correctly displayed	Pass
Expiry promotion	Product near expiry in DB	Discount applied automatically	Displayed correctly	Pass
Retailer update	Add new offer via dashboard	Offer reflected in recommendations	Updated successfully	Pass

Chapter 5: Results and Discussion

5.1 Results

The IoT-Based Smart Pricing and Shopping Assistant was successfully implemented and tested. The following key outcomes were observed:

1. **Product Scanning:**
    - The ESP – 32 cam module accurately detected products and retrieved details from the database in real time.
2. **Smart Pricing Engine:**
    - Economy, Value, and Premium modes provided correct recommendations.
    - Customers were able to compare per-unit prices instantly without manual calculations.
3. **Budget Monitoring:**
    - The system tracked shopping cart totals effectively.
    - Alerts (buzzer + message) were triggered when the budget was exceeded.

4. **Retailer Dashboard:**
    - Retailers could add offers, bundles, and expiry-based promotions.
    - Updated data was reflected immediately in customer recommendations.
5. **System Integration:**
    - Communication between IoT devices, cloud database, and user interfaces worked seamlessly.
    - Both customers and retailers benefited simultaneously.

5.2 Discussion

The developed system effectively bridges the gap between consumer convenience and retailer profitability, which is a limitation in most existing solutions. Customers saved time and money by receiving instant price comparisons and recommendations, while retailers gained higher sales through dynamic bundling and expiry-based promotions.

The project demonstrates that IoT technology can be applied in real-world retail scenarios to improve efficiency and user experience. However, some challenges remain:

- **Scalability:** The system needs to handle large product datasets and simultaneous users in supermarket environments.
- **Hardware Limitations:** Low-cost IoT devices like ESP32 or Raspberry Pi may face processing delays with big data.
- **Security Concerns:** Data protection and secure communication must be ensured to avoid misuse of customer and retailer data.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

The IoT-Based Smart Pricing and Shopping Assistant successfully addresses the challenges faced by both customers and retailers in the modern retail environment. By integrating IoT devices, cloud storage, and a smart pricing engine, the system enables customers to compare prices in real time, monitor their budget, and receive tailored recommendations. At the same time, retailers benefit from dynamic bundling, expiry-based promotions, and additional revenue opportunities through sponsored recommendations.

The project demonstrates how IoT technology can transform traditional shopping into a more **intelligent, efficient, and balanced ecosystem**, ensuring customer satisfaction and retailer profitability.

6.2 Future Work

Although the prototype shows promising results, there is scope for further development:

1. **Integration with Payment Systems:**
    - Enable auto-billing and seamless payment through mobile wallets or UPI.
2. **AI-Powered Recommendations:**
    - Incorporate machine learning models to provide more personalized product suggestions based on shopping history.
3. **Scalability Improvements:**
    - Optimize the system to handle large databases and multiple concurrent users in supermarket environments.
4. **Enhanced Security:**
    - Implement encryption and secure authentication to protect customer data and retailer transactions.
5. **Mobile Application Expansion:**
    - Develop a dedicated mobile app with push notifications for discounts, promotions, and budget alerts.

References for the report (APA Format)

1. "IoT-Based Smart Shopping Cart for Supermarkets," International Journal of Advanced Research in Computer Science, 2021 https://www.researchgate.net/publication/324271203_An_IOT_Based_Smart_Shopping_Cart_for_Smart_Shopping

2. Smart Auto-billing Shopping Cart using Raspberry Pi + Giveaway https://youtu.be/IqZSBzPKAco?si=xmNvKp_Dziv-Tc8l

---

Appendices

Appendix 1: Screenshots

- Screenshots of architecture diagram, ER diagram, block diagram, circuit connections, and system interface.

Appendix 2: Source Code

- Python/Embedded C code for RFID scanning, database communication, smart pricing engine, and customer/retailer interface.

ESP 32:

```
#include <WiFi.h>

#include <HTTPClient.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include "esp_camera.h"


// ===== WiFi =====

const char* ssid = "Alfiya";

const char* password = "12102006";


// ===== Server URLs =====

String serverPost = "http://10.87.244.101/iot_project/frontend/esp_post.php";   // change IP!

String serverCheckout = "http://10.87.244.101/iot_project/frontend/checkout.php"; // change IP!


// ===== LCD =====

LiquidCrystal_I2C lcd(0x27, 16, 2);


// ===== Buttons =====
```

```
#define SCAN_BUTTON 13
#define CHECKOUT_BUTTON 12

float totalAmount = 0.0;

// ===== Camera Pins =====
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

void setup() {
  Serial.begin(115200);

  // --- LCD Setup ---
  Wire.begin(14, 15);
  lcd.init();
```

```
lcd.backlight();
lcd.clear();
lcd.print("Smart Cart");
lcd.setCursor(0, 1);
lcd.print("Starting...");

// --- Buttons ---
pinMode(SCAN_BUTTON, INPUT_PULLUP);
pinMode(CHECKOUT_BUTTON, INPUT_PULLUP);

// --- WiFi ---
WiFi.begin(ssid, password);
lcd.clear();
lcd.print("Connecting WiFi");
Serial.print("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) {
  delay(200);
  Serial.print(".");
}
lcd.clear();
lcd.print("WiFi Connected!");
Serial.println("\nWiFi connected!");
delay(1000);

// --- Camera Config ---
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
```

```
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
config.frame_size = FRAMESIZE_QVGA;
config.jpeg_quality = 12;
config.fb_count = 1;

if (esp_camera_init(&config) != ESP_OK) {
  Serial.println("Camera init failed");
  lcd.clear();
  lcd.print("Cam Error!");
  while (true);
}

lcd.clear();
```

```
  lcd.print("Ready to Scan!");
}

void loop() {
  if (digitalRead(SCAN_BUTTON) == LOW) {
    sendBarcodeToServer("1234567890"); // sample barcode, replace later
    delay(1500);
  }

  if (digitalRead(CHECKOUT_BUTTON) == LOW) {
    handleCheckout();
    delay(2000);
  }
}

void sendBarcodeToServer(String barcode) {
  lcd.clear();
  lcd.print("Sending...");
  Serial.println("Sending barcode to server...");

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverPost);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    String postData = "barcode=" + barcode;
    int httpResponseCode = http.POST(postData);

    if (httpResponseCode == 200) {
```

```
      String payload = http.getString();
      Serial.println(payload);

      String name = extractValue(payload, "name");
      float price = extractValue(payload, "price").toFloat();
      totalAmount += price;

      lcd.clear();
      lcd.print(name.substring(0, 16));
      lcd.setCursor(0, 1);
      lcd.print("Rs:");
      lcd.print(price, 2);
      lcd.print(" Tot:");
      lcd.print(totalAmount, 2);
    } else {
      lcd.clear();
      lcd.print("HTTP Err:");
      Serial.println("HTTP Error: " + String(httpResponseCode));
    }
    http.end();
  } else {
    lcd.clear();
    lcd.print("WiFi Lost!");
  }
}

void handleCheckout() {
  lcd.clear();
  lcd.print("Checkout...");
```

```
  Serial.println("Checkout...");

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = serverCheckout + "?total=" + String(totalAmount);
    http.begin(url);
    http.GET();
    http.end();
  }

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Done! Total:");
  lcd.setCursor(0, 1);
  lcd.print("Rs:");
  lcd.print(totalAmount, 2);

  Serial.println("Checkout Done! Total: " + String(totalAmount));
  delay(4000);

  totalAmount = 0.0;
  lcd.clear();
  lcd.print("Ready to Scan!");
}

String extractValue(String json, String key) {
  int start = json.indexOf("\"" + key + "\":");
  if (start == -1) return "";
  start += key.length() + 3;
```

```
  int end = json.indexOf(",", start);
  if (end == -1) end = json.indexOf("}", start);
  String value = json.substring(start, end);
  value.replace("\"", "");
  return value;
}
```

DBMS Code:

```
CREATE DATABASE smart_shop;
USE smart_shop;
drop database smart_shop;
CREATE TABLE products (
  product_id INT AUTO_INCREMENT PRIMARY KEY,
  barcode VARCHAR(20) UNIQUE NOT NULL,
  name VARCHAR(100),
  brand VARCHAR(50),
  category VARCHAR(50),
  mrp DECIMAL(8,2),
  discount DECIMAL(5,2),
  final_price DECIMAL(8,2),
  pack_size VARCHAR(20),
  expiry_date DATE,
  sponsored_flag BOOLEAN DEFAULT 0,
  store_margin DECIMAL(5,2)
);

CREATE TABLE bundles (
  bundle_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  bundle_name VARCHAR(50),
  total_price DECIMAL(8,2),
  discount_text VARCHAR(100)
);

CREATE TABLE bundle_items (
  bundle_item_id INT AUTO_INCREMENT PRIMARY KEY,
  bundle_id INT,
  product_id INT,
  FOREIGN KEY (bundle_id) REFERENCES bundles(bundle_id),
  FOREIGN KEY (product_id) REFERENCES products(product_id)
);

CREATE TABLE customers (
  customer_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100),
  phone VARCHAR(15)
);

CREATE TABLE cart (
  cart_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT,
  product_id INT,
  quantity INT,
  price DECIMAL(8,2),
  mode_selected VARCHAR(20),
  timestamp DATETIME,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
```

```
  FOREIGN KEY (product_id) REFERENCES products(product_id)
);


CREATE TABLE promotions (
  promo_id INT AUTO_INCREMENT PRIMARY KEY,
  product_id INT,
  promo_type VARCHAR(20),
  discount_percent DECIMAL(5,2),
  valid_till DATE,
  FOREIGN KEY (product_id) REFERENCES products(product_id)
);

CREATE TABLE transactions (
  transaction_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT,
  total_amount DECIMAL(8,2),
  mode_selected VARCHAR(20),
  payment_status VARCHAR(20),
  timestamp DATETIME,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

ALTER TABLE promotions
ADD FOREIGN KEY (product_id) REFERENCES products(product_id);

ALTER TABLE cart
ADD FOREIGN KEY (product_id) REFERENCES products(product_id);
```

```
ALTER TABLE bundle_items

ADD FOREIGN KEY (bundle_id) REFERENCES bundles(bundle_id),

ADD FOREIGN KEY (product_id) REFERENCES products(product_id);


INSERT INTO customers (name, email, phone) VALUES

('Chinmayee', 'chinmayee@gmail.com', '9876543210'),

('Rahul', 'rahul@gmail.com', '9123456789');


INSERT INTO products (barcode, name, brand, category, mrp, discount, final_price, pack_size,
expiry_date, sponsored_flag, store_margin) VALUES

('1001', 'Dove Shampoo 180ml', 'Unilever', 'Haircare', 250.00, 10.00, 225.00, '180ml', '2026-01-15',
0, 12.5),

('1002', 'Clinic Plus Shampoo 180ml', 'HUL', 'Haircare', 220.00, 5.00, 209.00, '180ml', '2025-12-10',
0, 10.0),

('1003', 'Pears Soap 100g', 'Unilever', 'Skincare', 80.00, 10.00, 72.00, '100g', '2026-05-30', 1, 15.0),

('1004', 'Colgate Paste 200g', 'Colgate', 'Oralcare', 150.00, 15.00, 127.50, '200g', '2026-04-25', 0,
8.0);


INSERT INTO promotions (product_id, promo_type, discount_percent, valid_till) VALUES

(1, 'Festive Offer', 15.00, '2025-12-31'),

(3, 'Buy 2 Get 1', 0.00, '2025-11-30');


INSERT INTO bundles (bundle_name, total_price, discount_text) VALUES

('Daily Care Combo', 380.00, 'Buy Dove Shampoo + Pears Soap and Save ₹45'),

('Oral & Skin Combo', 180.00, 'Colgate + Pears Combo – Save ₹20');


INSERT INTO bundle_items (bundle_id, product_id) VALUES

(1, 1),  -- Dove Shampoo

(1, 3),  -- Pears Soap

(2, 3),  -- Pears Soap

(2, 4);  -- Colgate Paste
```

INSERT INTO cart (customer_id, product_id, quantity, price, mode_selected, timestamp) VALUES

(1, 1, 1, 225.00, 'Economy', NOW()),

(1, 3, 2, 144.00, 'Value', NOW()),

(2, 4, 1, 127.50, 'Premium', NOW());

INSERT INTO transactions (customer_id, total_amount, mode_selected, payment_status, timestamp) VALUES

(1, 369.00, 'Value', 'Paid', NOW()),

(2, 127.50, 'Premium', 'Pending', NOW());

Fronted:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Retail Store - Frontend Demo</title>
  <style>
    :root{
      --bg:#0f1724; /* dark navy */
      --panel:#0b1220;
      --muted:#94a3b8;
      --accent:#06b6d4; /* teal */
      --accent-2:#7c3aed; /* purple */
      --card:#071029;
      --glass:rgba(255,255,255,0.04);
      --success:#10b981;
      --danger:#ef4444;
      --radius:14px;
      --card-padding:18px;
      font-family: Inter, ui-sans-serif, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue",
Arial;
    }
    *{box-sizing:border-box}
    html,body{height:100%}
    body{
      margin:0;
      background: linear-gradient(180deg, #071427 0%, #051126 100%);
      color:#e6eef7;
      -webkit-font-smoothing:antialiased;
      -moz-osx-font-smoothing:grayscale;
      padding:28px;
    }

    .app{
```

```
      max-width:1200px;
      margin:0 auto;
      display:grid;
      grid-template-columns: 1fr 360px;
      gap:22px;
    }
```

```
    header{
      grid-column:1/-1;
      display:flex;
      align-items:center;
      justify-content:space-between;
      gap:16px;
      margin-bottom:6px;\    }
    .brand{
      display:flex;align-items:center;gap:14px
    }
    .logo{
      width:56px;height:56px;border-radius:12px;background:linear-gradient(135deg,var(--accent),var(--accent-
2));display:flex;align-items:center;justify-content:center;font-weight:700;color:#051124;font-size:20px
    }
    h1{margin:0;font-size:20px}
    .controls{display:flex;align-items:center;gap:8px}
```

```
    .search{
      display:flex;align-items:center;background:var(--glass);padding:8px 12px;border-
radius:12px;gap:8px;color:var(--muted)
    }
    .search input{background:transparent;border:0;outline:0;color:inherit;width:220px}
    .filters{display:flex;gap:8px}
    .btn{
      background:linear-gradient(90deg,var(--accent),var(--accent-2));border:0;padding:9px 12px;border-
radius:10px;color:#061022;cursor:pointer;font-weight:600
    }
    .ghost{background:transparent;border:1px solid rgba(255,255,255,0.06);padding:8px 10px;border-
radius:10px;color:var(--muted);cursor:pointer}
```

```
    .main{
      background:linear-gradient(180deg, rgba(255,255,255,0.02), transparent);padding:18px;border-
radius:var(--radius);box-shadow:0 10px 30px rgba(2,6,23,0.6)
    }
```

```
    .grid{
      display:grid;grid-template-columns:repeat(3,1fr);gap:14px
    }
    .card{
      background:linear-gradient(180deg, rgba(255,255,255,0.02), rgba(255,255,255,0.01));padding:14px;border-
radius:12px;border:1px solid rgba(255,255,255,0.03);min-height:160px;display:flex;flex-
direction:column;justify-content:space-between
    }
    .product-head{display:flex;gap:12px;align-items:center}
    .thumb{width:64px;height:64px;border-radius:10px;background:linear-
gradient(135deg,#0b1220,#091424);display:flex;align-items:center;justify-content:center;font-
weight:700;color:var(--muted)}
    .meta{flex:1}
    .name{font-weight:700;margin:0}
    .brandSmall{font-size:12px;color:var(--muted);margin-top:6px}
    .priceRow{display:flex;align-items:center;justify-content:space-between;margin-top:12px}
    .final{font-weight:800;font-size:18px}
    .mrp{font-size:13px;color:var(--muted);text-decoration:line-through;margin-left:8px}
    .discountTag{background:rgba(255,255,255,0.04);padding:6px;border-radius:8px;font-weight:700;color:var(--
success);font-size:13px}
```

```
    /* right column / cart */
```

```
    .sidebar{
      background:linear-gradient(180deg, rgba(255,255,255,0.02), transparent);padding:18px;border-
radius:var(--radius);height:calc(100vh - 80px);position:sticky;top:28px;display:flex;flex-
direction:column;gap:12px;overflow:auto
    }
    .cartItem{display:flex;gap:12px;align-items:center;padding:10px;background:rgba(255,255,255,0.02);border-
radius:10px}
    .qty{display:flex;align-items:center;gap:8px}
    .qty button{background:transparent;border:1px solid rgba(255,255,255,0.04);padding:6px;border-
radius:6px;color:var(--muted);cursor:pointer}
    .totals{margin-top:auto;padding-top:10px;border-top:1px dashed rgba(255,255,255,0.03)}
    .totals .row{display:flex;justify-content:space-between;margin-top:8px}
```

```
    /* Bundles & Promotions */
    .sectionTitle{display:flex;align-items:center;justify-content:space-between;margin-bottom:8px}
    .bundleList{display:flex;flex-direction:column;gap:12px}
    .bundleCard{display:flex;flex-
direction:column;gap:6px;padding:12px;background:rgba(255,255,255,0.015);border-radius:10px;border:1px solid
rgba(255,255,255,0.02)}
    .promo{padding:10px;border-radius:10px;background:linear-gradient(90deg, rgba(124,58,237,0.1),
rgba(6,182,212,0.06));display:flex;align-items:center;justify-content:space-between}.promo small{color:var(--
muted)}
```

```
    footer{grid-column:1/-1;margin-top:18px;color:var(--muted);font-size:13px}
```

```
    /* responsive */
    @media (max-width:980px){
      .app{grid-template-columns:1fr;}
      .grid{grid-template-columns:repeat(2,1fr)}
      .sidebar{height:auto;position:relative;top:0}
    }
    @media (max-width:640px){
      .grid{grid-template-columns:1fr}
      .search input{width:120px}
    }
```

```
    /* small helpers */
    .muted{color:var(--muted)}
    .pill{padding:6px 8px;border-radius:8px;background:rgba(255,255,255,0.02);font-weight:600}
    .empty{opacity:0.6;color:var(--muted);text-align:center;padding:28px;border-radius:10px}
  </style>
</head>
<body>
  <div class="app">
    <header>
      <div class="brand">
        <div class="logo">RS</div>
        <div>
          <h1>Retail Store — Demo UI</h1>
          <div class="muted" style="font-size:13px">Products, Bundles, Promotions & Cart simulation</div>
        </div>
      </div>
```

```
      <div class="controls">
        <div class="search">
          🔎
          <input id="q" placeholder="Search products or brand..." />
        </div>
        <div class="filters">
          <select id="categoryFilter" class="ghost">
            <option value="">All categories</option>
          </select>
        </div>
        <button class="btn" id="clearBtn">Clear Cart</button>
```

```
      </div>
    </header>

    <main class="main">
      <section style="margin-bottom:18px;display:flex;gap:12px;align-items:flex-start">
        <div style="flex:1">
          <div class="sectionTitle">
            <h2 style="margin:0">Products</h2>
            <div class="muted">Showing <span id="count">0</span> products</div>
          </div>

          <div id="products" class="grid">
            <!-- product cards injected here -->
          </div>
        </div>

        <aside style="width:320px;display:flex;flex-direction:column;gap:12px">
          <div>
            <div class="sectionTitle"><h3 style="margin:0">Bundles</h3><div class="muted">Combos &
offers</div></div>
            <div id="bundles" class="bundleList"></div>
          </div>

          <div>
            <div class="sectionTitle"><h3 style="margin:0">Promotions</h3><div class="muted">Active
promotions</div></div>
            <div id="promos"></div>
          </div>
        </aside>
      </section>

      <section>
        <div class="sectionTitle"><h3 style="margin:0">Selected Cart</h3><div class="muted">Temporary
simulation cart</div></div>
        <div id="cartPreview"></div>
      </section>

    </main>

    <aside class="sidebar">
      <div style="display:flex;justify-content:space-between;align-items:center">
        <div>
          <div class="muted">Your Cart</div>
          <div style="font-weight:800;font-size:20px">Checkout</div>
        </div>
        <div class="pill" id="cartCount">0 items</div>
      </div>

      <div id="cartList">
        <!-- cart items -->
      </div>

      <div class="totals">
        <div class="row"><div class="muted">Subtotal</div><div id="subtotal">₹0.00</div></div>
        <div class="row"><div class="muted">Discounts</div><div id="discountTotal">-₹0.00</div></div>
        <div class="row"><div class="muted">Tax (GST 0%)</div><div id="tax">₹0.00</div></div>
        <div class="row" style="margin-top:12px;font-weight:900;font-size:18px"><div>Total</div><div
id="grand">₹0.00</div></div>

        <div style="margin-top:14px;display:flex;gap:8px">
          <button class="btn" id="checkoutBtn">Proceed</button>
          <button class="ghost" id="exportBtn">Export JSON</button>
        </div>
```

```
      </div>
    </aside>
```

```
    <footer>Built for demo & education — Data is local to this page (no backend).</footer>
  </div>
```

```
  <script>
    /* ---------- Sample Data matching your schema ---------- */
    const products = [
      { product_id:1, barcode:'B10001', name:'AquaFresh Shampoo 200ml', brand:'CleanCo', category:'Shampoo',
mrp:199.00, discount:10.0, final_price:179.10, pack_size:'200ml', expiry_date:'2026-08-01',
sponsored_flag:false, store_margin:10},
      { product_id:2, barcode:'B10002', name:'CareSoap Bar 100g', brand:'SoftSkin', category:'Soap',
mrp:49.00, discount:0, final_price:49.00, pack_size:'100g', expiry_date:'2027-01-15', sponsored_flag:true,
store_margin:15},
      { product_id:3, barcode:'B10003', name:'Ultra Toothpaste 150g', brand:'SmileX', category:'Toothpaste',
mrp:129.00, discount:20, final_price:103.20, pack_size:'150g', expiry_date:'2026-04-22',
sponsored_flag:false, store_margin:12},
      { product_id:4, barcode:'B10004', name:'Herbal Conditioner 200ml', brand:'GreenLeaf',
category:'Conditioner', mrp:249.00, discount:15, final_price:211.65, pack_size:'200ml', expiry_date:'2026-11-
01', sponsored_flag:false, store_margin:11},
      { product_id:5, barcode:'B10005', name:'Kids Bubble Bath 500ml', brand:'HappyKids', category:'Bath',
mrp:349.00, discount:5, final_price:331.55, pack_size:'500ml', expiry_date:'2027-06-01', sponsored_flag:true,
store_margin:20},
      { product_id:6, barcode:'B10006', name:'Economy Dishwash 1L', brand:'HomeSafe', category:'Cleaning',
mrp:99.00, discount:0, final_price:99.00, pack_size:'1L', expiry_date:'2028-01-01', sponsored_flag:false,
store_margin:8}
    ];
```

```
    const bundles = [
      { bundle_id:1, bundle_name:'Haircare Combo', total_price:349.00, discount_text:'Save ₹100',
items:[1,4], description:'Shampoo + Conditioner combo' },
      { bundle_id:2, bundle_name:'Kids Bath Set', total_price:599.00, discount_text:'Buy 2 save ₹50',
items:[5], description:'Bubble Bath + Toys' }
    ];
```

```
    const promotions = [
      { promo_id:1, product_id:1, promo_type:'Festival', discount_percent:5, valid_till:'2026-12-31' },
      { promo_id:2, product_id:3, promo_type:'Expiry Clearance', discount_percent:30, valid_till:'2025-12-
31' }
    ];
```

```
    /* ---------- Application State ---------- */
    let cart = [];
    let activePromo = null;
```

```
    /* ---------- Utilities ---------- */
    function money(n){ return '₹'+Number(n).toFixed(2); }
```

```
    function findProduct(pid){ return products.find(p=>p.product_id===pid); }
```

```
    function calcFinalPrice(p){
      // compute from mrp and discount field; promotions applied separately
      const afterDiscount = p.mrp * (1 - (p.discount||0)/100);
      return Math.round(afterDiscount*100)/100;
    }
```

```
    function applyPromotionsToPrice(product){
      let price = calcFinalPrice(product);
      // check active promo that applies to this product and still valid
      const now = new Date();
      const applicable = promotions.filter(pr=>pr.product_id===product.product_id && new
Date(pr.valid_till) >= now);
```

```
    if(applicable.length){
      // choose highest discount
      const max = Math.max(...applicable.map(a=>a.discount_percent));
      price = price * (1 - max/100);
    }
    return Math.round(price*100)/100;
  }

  /* ---------- Rendering ---------- */
  function renderCategoryFilter(){
    const cat = [...new Set(products.map(p=>p.category))];
    const sel = document.getElementById('categoryFilter');
    cat.forEach(c=>{
      const o = document.createElement('option'); o.value=c; o.textContent=c; sel.appendChild(o);
    });
  }

  function renderProducts(filterQ=''){
    const container = document.getElementById('products'); container.innerHTML='';
    const q = (document.getElementById('q').value || '').toLowerCase();
    const cat = document.getElementById('categoryFilter').value;
    let list = products.slice();
    if(cat) list = list.filter(p=>p.category===cat);
    if(q) list = list.filter(p=> (p.name+ ' '+p.brand+' '+p.category).toLowerCase().includes(q));
    document.getElementById('count').textContent = list.length;

    if(list.length===0){ container.innerHTML='<div class="empty">No products found</div>'; return; }

    for(const p of list){
      const card = document.createElement('div'); card.className='card';
      card.innerHTML = `
        <div class="product-head">
          <div class="thumb">${p.brand[0]||'P'}</div>
          <div class="meta">
            <p class="name">${p.name}</p>
            <div class="brandSmall">${p.brand} • ${p.pack_size}</div>
          </div>
          <div style="text-align:right">
            <div class="discountTag">${p.discount}%</div>
          </div>
        </div>
        <div class="priceRow">
          <div>
            <div class="final">${money(applyPromotionsToPrice(p))}</div>
            <div class="muted" style="font-size:13px">MRP <span class="mrp">${money(p.mrp)}</span></div>
          </div>
          <div style="display:flex;flex-direction:column;gap:8px;align-items:flex-end">
            <button class="btn" data-pid="${p.product_id}">Add</button>
            <div class="muted" style="font-size:12px">${p.sponsored_flag? 'Sponsored' : ''}</div>
          </div>
        </div>
      `;
      container.appendChild(card);
    }

    // attach handlers
    container.querySelectorAll('.btn').forEach(b=>b.addEventListener('click', e=>{
      const pid = Number(e.currentTarget.dataset.pid);
      addToCart(pid,1);
    }));
  }

  function renderBundles(){
    const el = document.getElementById('bundles'); el.innerHTML='';
```

```
      for(const b of bundles){
        const node = document.createElement('div'); node.className='bundleCard';
        const items = b.items.map(id=>findProduct(id)).filter(Boolean).map(p=>p.name).join(', ');
        node.innerHTML = `
          <div style="display:flex;justify-content:space-between;align-items:center">
            <strong>${b.bundle_name}</strong>
            <div class="muted">${money(b.total_price)}</div>
          </div>
          <div class="muted" style="font-size:13px">${b.description}</div>
          <div style="display:flex;justify-content:space-between;align-items:center;margin-top:6px">
            <div class="muted" style="font-size:13px">${items}</div>
            <div style="display:flex;gap:8px"><button class="btn" data-bid="${b.bundle_id}">Add
Bundle</button></div>
          </div>
        `;
        el.appendChild(node);
      }
      el.querySelectorAll('.btn').forEach(b=>b.addEventListener('click',e=>{
        const bid = Number(e.currentTarget.dataset.bid);
        const bundle = bundles.find(x=>x.bundle_id===bid);
        bundle.items.forEach(id=>addToCart(id,1));
      }));
    }
```

```
    function renderPromotions(){
      const el = document.getElementById('promos'); el.innerHTML='';
      const now = new Date();
      const active = promotions.filter(p=> new Date(p.valid_till) >= now);
      if(active.length===0){ el.innerHTML='<div class="empty">No active promotions</div>'; return; }
      for(const p of active){
        const prod = findProduct(p.product_id);
        const node = document.createElement('div'); node.className='promo';
        node.innerHTML = `
          <div>
            <div style="font-weight:700">${prod?prod.name:'Product'} — ${p.discount_percent}%</div>
            <small class="muted">${p.promo_type} • valid till ${p.valid_till}</small>
          </div>
          <div style="display:flex;flex-direction:column;gap:6px;align-items:flex-end">
            <button class="ghost" data-pid="${p.product_id}">Apply</button>
            <div class="muted" style="font-size:12px">Promo ID: ${p.promo_id}</div>
          </div>
        `;
        el.appendChild(node);
      }
```

```
      el.querySelectorAll('.ghost').forEach(b=>b.addEventListener('click', e=>{
        const pid = Number(e.currentTarget.dataset.pid);
        // set activePromo to promo for that product (highest percent if multiple)
        const now = new Date();
        const applicable = promotions.filter(pr=>pr.product_id===pid && new Date(pr.valid_till)>=now);
        if(applicable.length){
          activePromo = applicable.reduce((a,b)=> a.discount_percent>b.discount_percent? a:b);
          alert('Applied promotion: ' + activePromo.discount_percent + '% for product id '+ pid);
          renderProducts(); renderCart();
        }
      }));
    }
```

```
    /* ---------- Cart Logic ---------- */
    function addToCart(product_id, qty=1){
      const existing = cart.find(c=>c.product_id===product_id);
      if(existing){ existing.quantity += qty; }
      else cart.push({ product_id, quantity:qty });
      renderCart(); renderCartPreview();
```

```
    }

    function removeFromCart(product_id){ cart = cart.filter(c=>c.product_id!==product_id); renderCart();
renderCartPreview(); }

    function updateQuantity(product_id, q){ const it = cart.find(c=>c.product_id===product_id); if(!it)
return; it.quantity = Math.max(0, Number(q)||0); if(it.quantity===0) removeFromCart(product_id); else
renderCart(); renderCartPreview(); }

    function clearCart(){ cart = []; activePromo = null; renderCart(); renderCartPreview(); }

    function calculateTotals(){
      let subtotal = 0; let discountTotal = 0;
      for(const c of cart){
        const prod = findProduct(c.product_id);
        if(!prod) continue;
        const base = calcFinalPrice(prod) * c.quantity;
        let final = applyPromotionsToPrice(prod) * c.quantity;
        subtotal += base;
        discountTotal += (base - final);
      }
      subtotal = Math.round(subtotal*100)/100;
      discountTotal = Math.round(discountTotal*100)/100;
      const tax = 0;
      const grand = Math.round((subtotal - discountTotal + tax)*100)/100;
      return {subtotal, discountTotal, tax, grand};
    }

    function renderCart(){
      const el = document.getElementById('cartList'); el.innerHTML='';
      if(cart.length===0){ el.innerHTML='<div class="empty">Cart is empty. Add products on the left.</div>';
document.getElementById('cartCount').textContent='0 items'; return; }
      document.getElementById('cartCount').textContent = cart.reduce((s,i)=>s+i.quantity,0) + ' items';
      for(const c of cart){
        const p = findProduct(c.product_id);
        const node = document.createElement('div'); node.className='cartItem';
        node.innerHTML = `
          <div style="flex:1">
            <div style="font-weight:700">${p.name}</div>
            <div class="muted" style="font-size:13px">${p.brand} • ${p.pack_size}</div>
          </div>
          <div style="display:flex;flex-direction:column;align-items:flex-end;gap:6px">
            <div style="font-weight:800">${money(applyPromotionsToPrice(p) * c.quantity)}</div>
            <div class="qty">
              <button data-op="dec" data-pid="${p.product_id}">-</button>
              <div style="padding:6px 10px;background:rgba(255,255,255,0.02);border-
radius:8px">${c.quantity}</div>
              <button data-op="inc" data-pid="${p.product_id}">+</button>
            </div>
            <button class="ghost" data-remove="${p.product_id}">Remove</button>
          </div>
        `;
        el.appendChild(node);
      }

      // attach events
      el.querySelectorAll('[data-op]').forEach(b=>b.addEventListener('click',e=>{
        const pid = Number(e.currentTarget.dataset.pid);
        const op = e.currentTarget.dataset.op;
        const it = cart.find(x=>x.product_id===pid);
        if(!it) return;
        if(op==='inc') updateQuantity(pid, it.quantity+1);
        else updateQuantity(pid, it.quantity-1);
      }));
```

```
      el.querySelectorAll('[data-remove]').forEach(b=>b.addEventListener('click',e=>{
        const pid = Number(e.currentTarget.dataset.remove); removeFromCart(pid);
      }));
```

```
      // update totals
      const t = calculateTotals();
      document.getElementById('subtotal').textContent = money(t.subtotal);
      document.getElementById('discountTotal').textContent = '-'+money(t.discountTotal).replace('₹','');
      document.getElementById('tax').textContent = money(t.tax);
      document.getElementById('grand').textContent = money(t.grand);
```

```
      // checkout/export handlers
      document.getElementById('exportBtn').onclick = ()=>{
        const payload = { cart:cart.map(ci=> ({...ci, product: findProduct(ci.product_id)})), totals: t };
        const data = JSON.stringify(payload, null, 2);
        const blob = new Blob([data], {type:'application/json'});
        const url = URL.createObjectURL(blob);
        const a = document.createElement('a'); a.href = url; a.download = 'cart.json'; a.click();
URL.revokeObjectURL(url);
      };
```

```
      document.getElementById('checkoutBtn').onclick = ()=>{
        alert('Checkout simulated. Total: ' + document.getElementById('grand').textContent);
      };
    }
```

```
    function renderCartPreview(){
      const el = document.getElementById('cartPreview'); el.innerHTML='';
      if(cart.length===0){ el.innerHTML='<div class="empty">No items selected for simulation.</div>';
return; }
      const ul = document.createElement('div'); ul.style.display='grid';
ul.style.gridTemplateColumns='repeat(3,1fr)'; ul.style.gap='8px';
      for(const c of cart){ const p = findProduct(c.product_id); const block = document.createElement('div');
block.className='card'; block.style.padding='8px'; block.innerHTML = `<div style="font-
weight:700">${p.name}</div><div class="muted" style="font-size:12px">${c.quantity} ×
${money(applyPromotionsToPrice(p))}</div>`; ul.appendChild(block); }
      el.appendChild(ul);
    }
```

```
    /* ---------- Wiring & events ---------- */
    document.getElementById('q').addEventListener('input', ()=> renderProducts());
    document.getElementById('categoryFilter').addEventListener('change', ()=> renderProducts());
    document.getElementById('clearBtn').addEventListener('click', ()=> { if(confirm('Clear cart?'))
clearCart(); });
```

```
    document.getElementById('checkoutBtn').addEventListener('click', ()=>{ /* handled in renderCart to ensure
totals exist */ });
```

```
    // initial render
    renderCategoryFilter(); renderProducts(); renderBundles(); renderPromotions(); renderCart();
```

```
  </script>
</body>
</html>
```

Appendix 3: Plagiarism Details

-