DEPARTMENT OF COMPUTER TECHNOLOGY,

MIT CAMPUS, ANNA UNIVERSITY, CHENNAI – 44

INDUSTRIAL ORIENTED COURSE

IOT PROJECT

SMART CART

TEAM GYRO GUARDIANS

TEAM MEMBERS:

- DURAISINGH J – 2023503051
- KARTHIK RAJA A – 2023503522
- GOKUL K – 2023503051
- LOKESH R N – 2023503513
- KISHOR M – 2023503514
- SRIRAM S – 2023503058
- ARUMUGA PERUMAL – 2023503031
- AAJEETH D R – 2023503585
- ADITHYA B – 2023503520
- ARUN ULAGAPPAN S – 2023503016
- INBANATHAN M - 2023503538

TABLE OF CONTENTS:

EXECUTIVE SUMMARY

This report documents the design and development of a Smart Cart system using RFID and Firebase integration, aimed at simplifying the shopping experience. The system automatically detects products placed in or removed from the cart using RFID tags and synchronizes real-time data with Firebase. This project addresses the need for efficient billing and inventory management in retail environments.

PROJECT OBJECTIVE

The objective of the Smart Cart project is to:

- Develop a prototype that tracks products using RFID tags.
- Automatically update product status (added/removed) in a real-time cloud database.
- Notify users of successful additions, removals, or unknown products using buzzer feedback.
- Eliminate manual checkout lines and improve customer convenience.

SCOPE

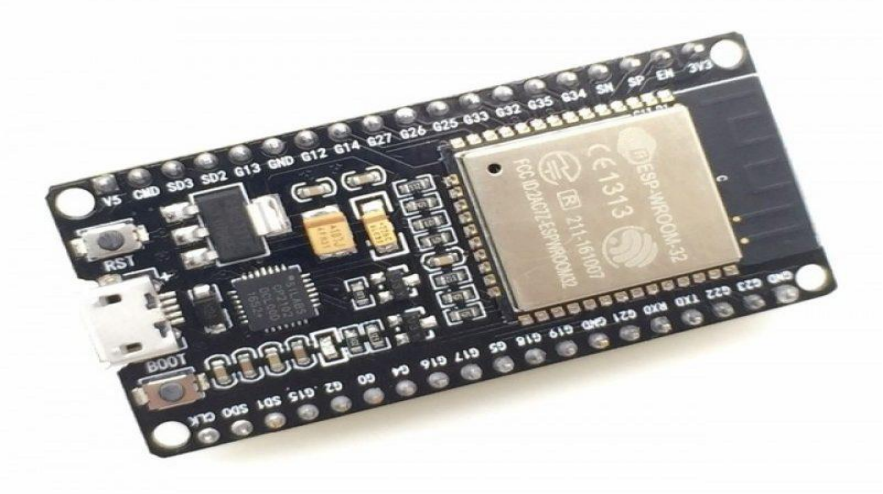
- Applicable in retail supermarkets and hypermarkets.
- Focused on real-time item detection and inventory tracking.
- Uses Firebase for cloud-based database management.
- Covers only detection and tracking, not payment processing.

METHODOLOGY

1. Integrate MFRC522 RFID reader with ESP32 microcontroller.
2. Use Firebase Realtime Database to store product information and update states.
3. Establish WiFi communication for real-time sync.
4. Program logic to detect RFID tag presence, validate against database, and update item status.
5. Use buzzer feedback for user interaction.
6. Implement debounce logic to avoid multiple readings.

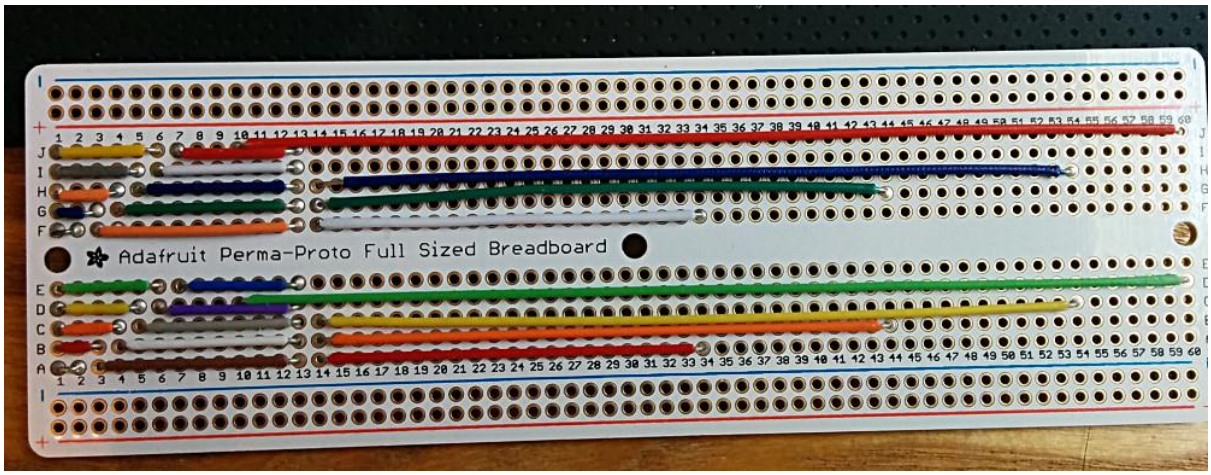
ARTIFACTS USED

- **Hardware:**
    - ESP32 Microcontroller



- MFRC522 RFID Reader
- RFID Tags

- Buzzer
- Jumper Wires
- Breadboard



TECHNICAL COVERAGE

- **RFID Detection:**
    - The MFRC522 reads RFID tags and retrieves UID in hexadecimal format.

```
// RFID_Detector.h
#ifndef RFID_DETECTOR_H
#define RFID_DETECTOR_H

#include <MFRC522.h>
#include <SPI.h>

#define SS_PIN 5
#define RST_PIN 22

class RFIDDetector {
  private:
    MFRC522 mfrc522;
    String lastTag;
    unsigned long lastReadTime;
    const unsigned long debounceTime = 1000;

  public:
    RFIDDetector() : mfrc522(SS_PIN, RST_PIN), lastReadTime(0) {}

    void begin() {
      SPI.begin();
      mfrc522.PCD_Init();
    }
```

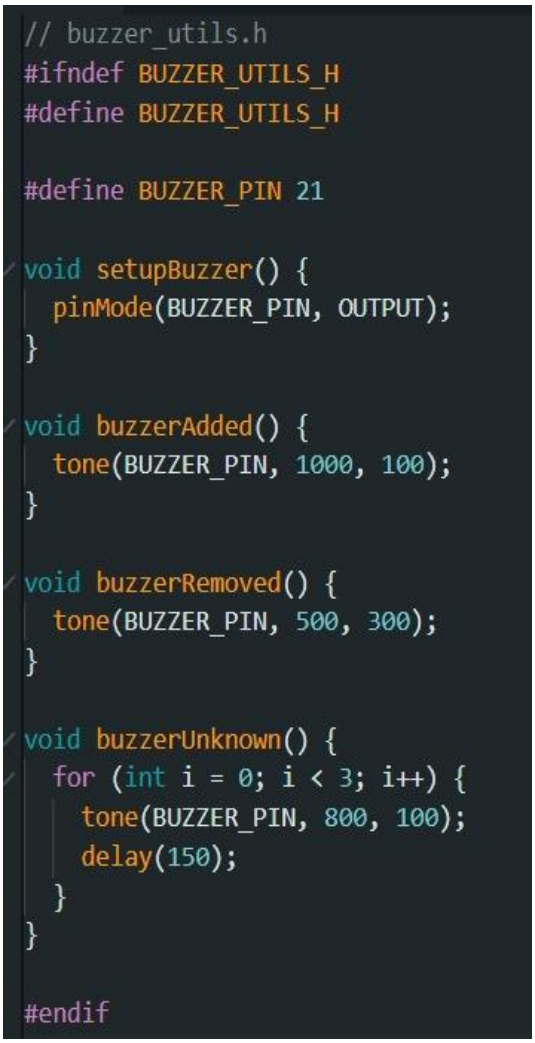
```
    String readTag() {
      if (lastTag != "" && millis() - lastReadTime > debounceTime) {
        lastTag = "";
      }

      if (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) {
        return "";
      }

      String tagUID = "";
      for (byte i = 0; i < mfrc522.uid.size; i++) {
        if (mfrc522.uid.uidByte[i] < 0x10) tagUID += "0";
        tagUID += String(mfrc522.uid.uidByte[i], HEX);
      }
      tagUID.toLowerCase();

      if (tagUID != lastTag || millis() - lastReadTime > debounceTime) {
        lastTag = tagUID;
        lastReadTime = millis();
        mfrc522.PICC_HaltA();
        mfrc522.PCD_StopCrypto1();
        return tagUID;
      }

      return "";
    }
};

#endif
```

- **Firebase Integration:**
    - API key and database URL are configured.
    - Anonymous sign-in method is used for authentication.
    - Product data is stored in paths like /Products/<tagUID>.

```
// Firebase_Handler.h
#ifndef FIREBASE_HANDLER_H
#define FIREBASE_HANDLER_H

#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <addons/TokenHelper.h>

#define WIFI_SSID "Redmi"
#define WIFI_PASSWORD "1234567890"
#define FIREBASE_API_KEY "AIzaSyDybdqtljX2xzXbYyeEtTtgVOIWKGGIPXg"
#define DATABASE_URL "https://smart-cart-c8262-default-rtdb.asia-southeast1.firebasedatabase.app/"

class FirebaseHandler {
  public:
    FirebaseData fbdo;
    FirebaseAuth auth;
    FirebaseConfig config;

    void begin() {
      Serial.print("Connecting to WiFi");
      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
      while (WiFi.status() != WL_CONNECTED) {
        delay(300);
        Serial.print(".");
      }
      Serial.println("\nWiFi connected");

      config.api_key = FIREBASE_API_KEY;
      config.database_url = DATABASE_URL;
```

```
      if (Firebase.signUp(&config, &auth, "", "")) {
        Serial.println("Firebase auth OK");
      } else {
        Serial.println("Auth failed: " + String(config.signer.signupError.message.c_str()));
      }
      config.token_status_callback = tokenStatusCallback;
      Firebase.begin(&config, &auth);
      Firebase.reconnectWiFi(true);
      unsigned long startTime = millis();
      Serial.print("Connecting to Firebase");
      while (!Firebase.ready() && millis() - startTime < 20000) {
        Serial.print(".");
        delay(300);
      }
      Serial.println(Firebase.ready() ? "\nFirebase connected!" : "\nFailed to connect to Firebase");
    }
    bool toggleProduct(String tagUID, bool &newStatus) {
      String path = "/Products/" + tagUID;
      String isAddedPath = path + "/isAdded";
      if (Firebase.RTDB.get(&fbdo, path) && fbdo.dataType() == "json") {
        bool currentStatus = false;
        if (Firebase.RTDB.getBool(&fbdo, isAddedPath)) {
          currentStatus = fbdo.boolData();
          newStatus = !currentStatus;
          return Firebase.RTDB.setBool(&fbdo, isAddedPath, newStatus);
        }
      }
      return false;
    }
    String getLastError() {
      return fbdo.errorReason();
    }}; #endif
```

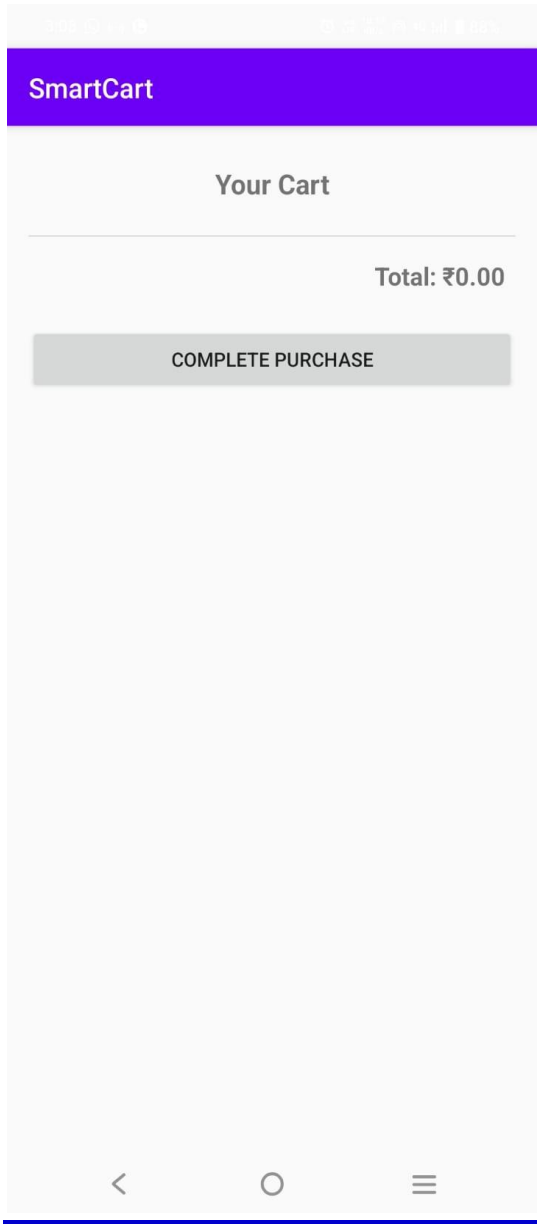
- **Debounce Logic:**
    - Prevents multiple scans within a short interval using time comparison.
- **Status Toggle Logic:**
    - Each scan toggles the isAdded field in Firebase.
- **User Feedback:**
    - Buzzer tones indicate product added, removed, or unknown tag.

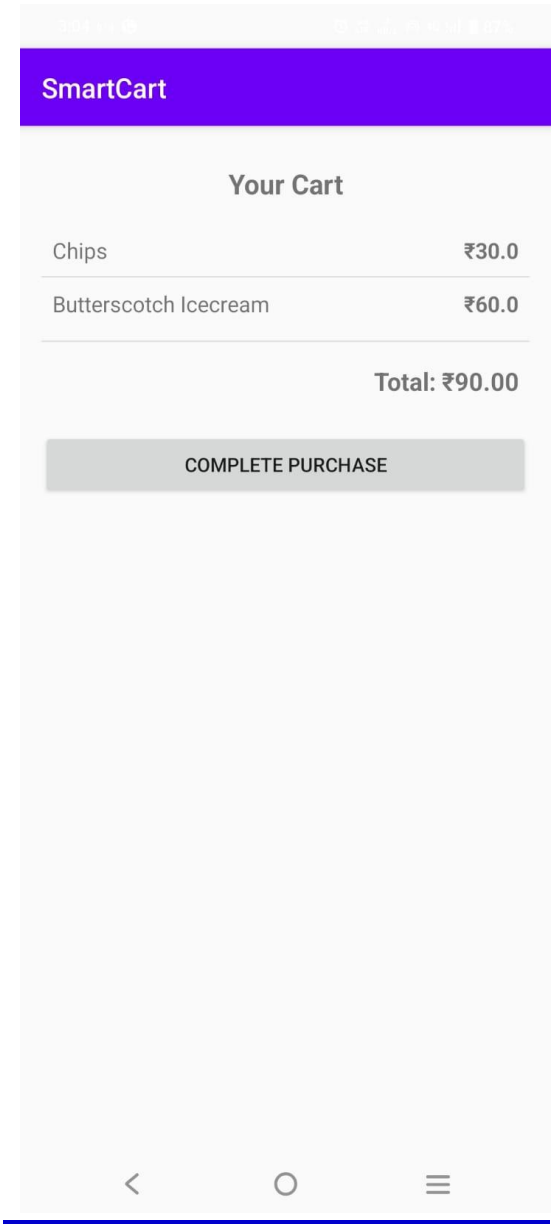


```
// buzzer_utils.h
#ifndef BUZZER_UTILS_H
#define BUZZER_UTILS_H

#define BUZZER_PIN 21

void setupBuzzer() {
  pinMode(BUZZER_PIN, OUTPUT);
}

void buzzerAdded() {
  tone(BUZZER_PIN, 1000, 100);
}

void buzzerRemoved() {
  tone(BUZZER_PIN, 500, 300);
}

void buzzerUnknown() {
  for (int i = 0; i < 3; i++) {
    tone(BUZZER_PIN, 800, 100);
    delay(150);
  }
}

#endif
```

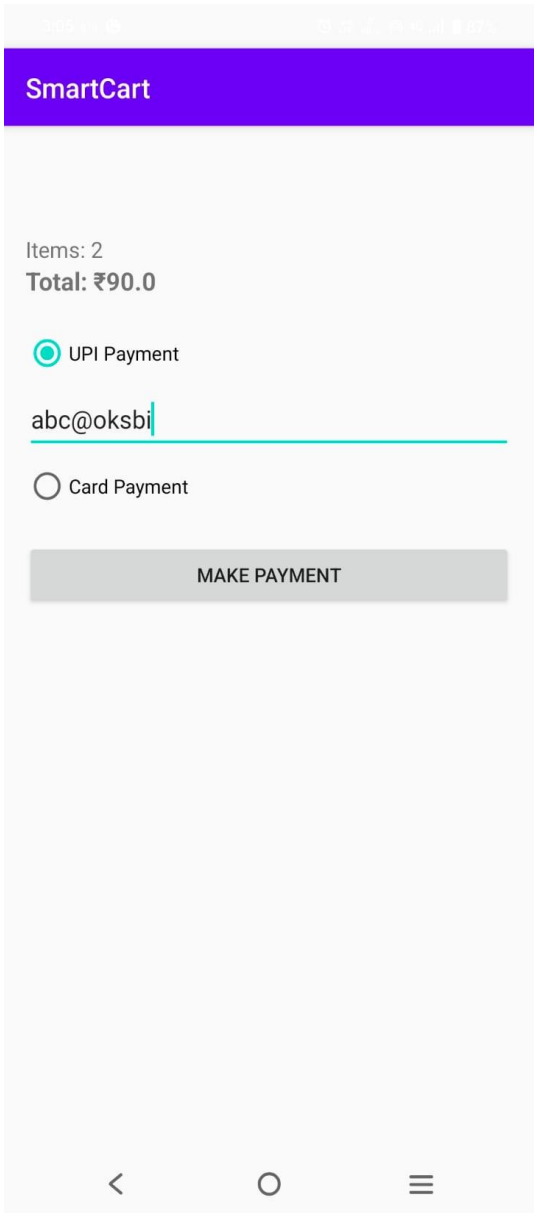
MOBILE APPLICATION

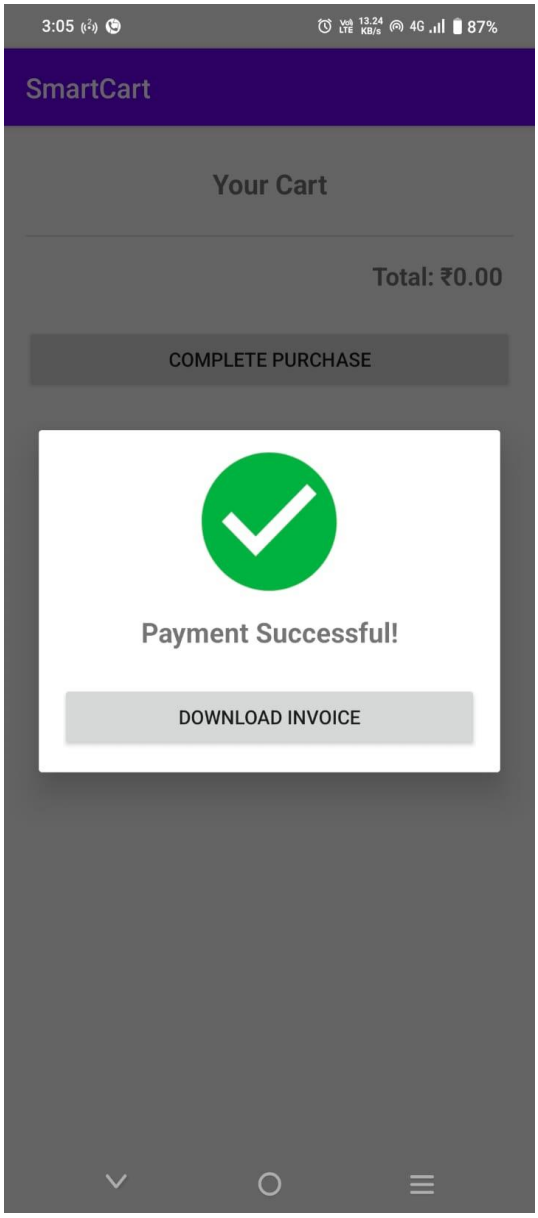
Scanning and adding items to Bill:





Payment Processing:





RESULTS

- RFID tags are successfully detected and differentiated.
- Real-time database updates occur without noticeable delay.
- Buzzer tones provide clear auditory feedback.
- Unknown tags trigger a distinct triple-beep warning.
- Reliable operation under typical shopping conditions.

CHALLENGES AND RESOLUTION

- **WiFi Connectivity Issues:**
  - Resolved by implementing a connection timeout and retry logic.
- **Multiple Scans of the Same Tag:**
  - Fixed using a debounce timer to avoid redundant reads.
- **Firebase Authentication Failures:**
  - Addressed by ensuring proper API key and anonymous auth setup.
- **Power Supply Fluctuations:**
  - Solved by using a regulated power source for ESP32 and peripherals.

CONCLUSION

The Smart Cart project successfully demonstrates how IoT technologies can automate and enhance retail experiences. By integrating RFID with real-time cloud updates and feedback systems, the solution offers an efficient and scalable approach to smart shopping carts. Future enhancements could include display modules, mobile app integration, and secure payment gateways.