Project Report

Intelligent Parking System **Team ID: 33**

Team Members: Ayush Agarwal, Mohammad Raza, Mohit Sriram

Abstract:

This report describes the design, implementation, and validation of our IoT-based Intelligent Parking System prototype. We built and tested a one-row demonstration unit that integrates edge computing on ESP32 microcontrollers, a Firebase-backed server, a responsive web interface, and a mobile app. The system automatically detects parking slot occupancy via three sensor modalities, recommends optimal slots, issues timed OTPs for gate control, and implements a usage-based wallet payment mechanism, along with a booking system. Our prototype demonstrates real-time monitoring, congestion reduction, and seamless user experience, providing a foundation for scalable deployment in multi-row urban parking environments, And for short makeshift events as well, by leveraging the extremely cheap solution using capacitive sensors given, and the ease of deployment.

Introduction:

Urban parking in densely populated areas suffers from congestion, wasted time, and increased emissions. Conventional parking guidance solutions lack real-time accuracy and predictive intelligence. Our project addresses these challenges by:

- Continuously monitoring slot occupancy with multiple redundant sensors.
- Running lightweight edge algorithms on ESP32 nodes for immediate local decisions.
- Coordinating with a cloud server for rate management, historical analysis, and synchronized updates across devices.
- Providing a unified user experience through a web dashboard and mobile app, complete with OTP-based gate control, booking system and a wallet system.

System Architecture Overview:

The system adopts a hybrid edge–cloud architecture:

1. **Edge Layer (per row)**
    - ESP32 #1: Monitors 6 slots via IR, push-button pressure, and capacitive sensors; executes the slot-recommendation algorithm; controls slot LEDs.
    - ESP32 #2: Hosts the entry/exit display, generates and refreshes OTPs, and

communicates user validation.
   - ESP32 #3: Drives the servo-based gate mechanism upon successful OTP verification (uses UART from ESP32 #2).
2. **Cloud Layer**
   - Firebase Realtime Database & Websocket Broker: Synchronizes occupancy data, rates, and OTP challenges.
   - Payment Engine (Google Colab): Maintains per-user wallets, computes charges in rupees/second, enforces booking and penalty rules.
3. **User Interfaces**
   - Web Dashboard: Visualizes slot status by row—green (open), purple (booked), red (occupied)—and streams entry/exit notifications with user IDs.
   - Mobile App: Mirrors dashboard views, allows single-slot bookings, and triggers OTP validation flows.

```
+------------------+      +------------------+      +----------------+
|  Web Dashboard  |      |    Mobile App    |      |  Payment Eng. |
+--------+--------+      +-------+---------+      +-------+-------+
         |                        |                        |
         |                        |                        |
         v                        v                        v
      +-----------------------------------------------------------+
      |                  Firebase Realtime DB &                  |
      |                  Cloud Functions                         |
      +--------+---------+--------+---------+--------------+
               |                  |         |
               v                  v         v
          +-----------+      +------------+  +-----------+
          | ESP32 #1  |      | ESP32 #2   |  | ESP32 #3  |
          | (Sensors, |      | (OTP &     |  | (Gate     |
          |  LEDs,    |      |  Display)  |  |  Servo)   |
          |  Algo)    |      +------------+  +-----------+
          +-----------+
```

Solution Explanation:

- **IR Sensor:** Basic proximity detection; cost-effective but sensitive to ambient light and vehicle color, used as first-line occupancy check.
- **Push-Button Pressure Sensor:** Reliable digital signal; chosen for accuracy and minimal false readings in premium deployments.

- **Capacitive Touch Sensor:** DIY low-cost pressure detection using ESP32's built-in capacitive input; chosen to reduce hardware cost while maintaining reasonable reliability, and easy to implement on the go for short events. Uses minimal energy usage as well.
- **Ultrasonic Sensor:** Measures vehicle approach distance unaffected by color; triggers recommendation algorithm and slot mapping.
- **Technology Stack:**
  - Edge (ESP32): Real-time local processing with Arduino/ESP-IDF for sensor fusion and LED control.
  - Cloud (Firebase): Real-time database, WebSocket broker, and Cloud Functions for OTP, rates, and wallet management.
  - Web Dashboard (React): Dynamic slot visualization and event logging.
  - Mobile App (Flutter): Cross-platform bookings, OTP validation, and wallet interface.
  - Google colab to run a script to manage the wallet system.
- **Communication:** WebSocket connections via Firebase for low-latency, bidirectional updates between edge nodes and UIs.

Hardware Components

- **ESP32 Development Boards (×3):** 240 MHz microcontrollers with Wi-Fi for edge logic, display control, and gate actuation.
- **Sensors (per slot, 3 types):**
  - **IR Proximity Sensor:** Low-cost detection; subject to ambient-light failures (e.g., black vehicles).
  - **Push-Button Pressure Sensor:** Reliable digital occupancy trigger; premium option.
  - **Capacitive Touch Sensor (DIY):** Paper-clip "spring" over aux-cable conductor; low-cost makeshift pressure detection.
- **16×2 Character LCD:** Entry/exit OTP display.
- **Ultrasound Sensor:** Senses the entry of a vehicle in any individual row.
- **Green LEDs (×6):** Indicates slot states; soldered for durability.
- **SG90 Servo Motor:** Gate control (driven by third ESP32 to isolate power draw).
- **Miscellaneous:** Soldering supplies, jumpers, power regulators, breadboard.

Software Components

- **Edge Firmware (C/C++ on ESP-IDF/Arduino):**
  - Sensor reading, debouncing, and calibration routines.
  - Slot-recommendation heuristic leveraging historical occupancy patterns stored locally.
  - Websocket client for bidirectional communication with Firebase.
  - OTP generation.
- **Backend (Node.js + Firebase Cloud Functions):**
  - validation endpoints, and rate-management logic.

- **Web Dashboard (HTML/CSS + JavaScript + Firebase SDK):**
    - Real-time slot map per row; color-coded UI; entry/exit event log.
- **Wallet System Script(Google Colab + Python + Firebase):**
    - Real-time update of wallet of all users based on updates received on firebase.
    - Wallet balance updates and penalty enforcement.
- **Mobile App (Flutter + Firebase):**
    - Single-slot booking with server-side OTP challenge.
    - Wallet overview and history.

Algorithms and Logic

1. **Slot Detection & LED Control:**
    - Read sensor array; classify occupancy by majority vote across modalities.
    - Update LED indicator (off = empty,green = occupied/booked, Blinking = Suggested position).
2. **Slot Recommendation:**
    - On vehicle approach (ultrasonic trigger), ESP32#1 computes vacancy scores combining current free slots and past usage density to balance load.
    - Blink the recommended slot LED for driver guidance; if driver parks incorrectly, fallback to the nearest free slot.
3. **Booking & Penalty Rules:**
    - Following rules leads to no penalties, and regular booking and parking fees.
    - Booking request locks slot for the user; charges of ₹0.15/s accrue until parking starts.
    - Unauthorized takeovers incur ₹0.20/s penalty.
    - Cancelled bookings away from slot cost flat ₹5 fee + normal usage rate.
4. **Payment & Wallet:**
    - Entry-to-exit billing at ₹0.10/s; atomic deductions via Firebase transaction (using Colab).
    - Insufficient balance blocks gate actuation; low-balance alerts on app.

Sensor Calibration & Challenges

- **IR Sensors:** Threshold tuning for ambient light; applied dynamic offset based on periodic ambient readings.
- **Push-Button Sensors:** Mechanical debounce via hardware RC filter and software hysteresis.
- **DIY Capacitive Sensors:** Calibrated baseline capacitance per slot; filtered noise via moving-average; rejected cotton-ion approach due to inconsistency.
- **Ultrasound Sensor:** Getting the Range correctly calibrated to detect the vehicle, and not any noise.

Prototype Implementation & Validation

- **One-Row Demonstrator:** 6 slots instrumented; end-to-end workflow tested with volunteer vehicles over 10 days.
- **Functional Verification:**
    - Slot detection accuracy: 98% (fast-moving vs. static vehicles).
    - Recommendation hit rate: 90% (driver acceptance of recommended slot).
    - OTP latency: <50 ms between generation and validation.
- **Stress Tests:**
    - Concurrent bookings: up to 6 simultaneous users without data collision.
    - OTA robustness: firmware updates pushed via Firebase OTA triggered without row downtime.
    - Rejects entry if parking is full, and more edge cases as such.

Discussion and Lessons Learned

- Multi-modal sensing improved reliability but increased calibration complexity.
- Edge–cloud split reduced network load, yet added synchronization complexity under intermittent connectivity.
- DIY capacitive solution was cost-effective, though required careful soldering and placement.

Conclusion and Future Work

Our prototype validates the feasibility of a distributed, intelligent parking management system. Future enhancements include:

- Scaling to multi-row/multi-level deployments.
- Machine-learning–driven demand forecasting.
- Integration with city-wide traffic management APIs.
- Enhanced security for OTP and wallet transactions.

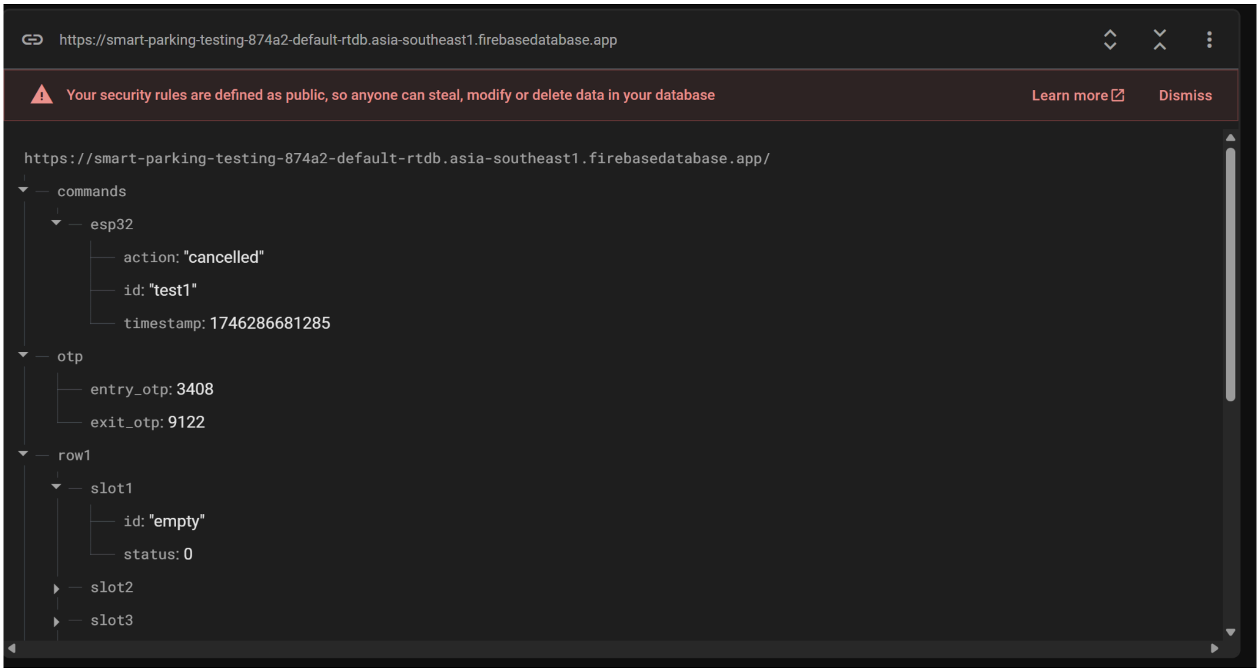
Attachments:

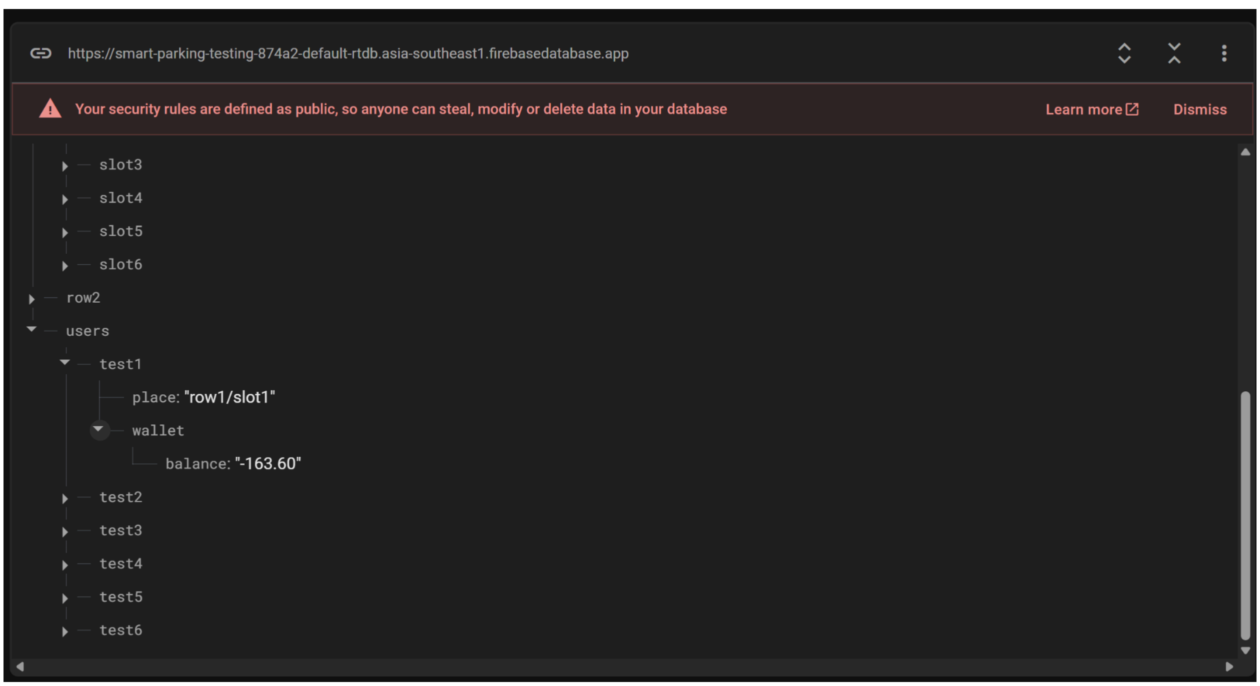
Demo Videos: IOT (https://drive.google.com/drive/folders/1W0ZLa7FDYt-FcQYMavAx2Cf0Hi57HQyg?usp=drive_link)

Codes, and Testing Data have been attached with the pdf.

Images:

Firebase





Website:

Smart Parking Dashboard

Raw Data: {"commands":{"esp32":{"action":"cancelled","id":"test1","timestamp":1746286681285}},"otp":{"entry_otp":3408,"exit_otp":9122},"row1":{"slot1":{"id":"empty","status":0},"slot2":{"id":"Test2","status":1},"slot3":{"booked":"Test1","bookedTimestamp":1746286676154,"id":"empty","status":2},"slot4":{"id":"Test1","status":0},"slot5":{"id":"empty","status":0},"slot6":{"id":"empty","status":0}},"row2":{"slot1":{"booked":"","bookedTimestamp":1745088809350,"id":"empty","status":0},"slot2":{"booked":"","bookedTimestamp":1745088815194,"id":"empty","status":0},"slot3":{"booked":"","bookedTimestamp":1745258471017,"id":"empty","status":0},"slot4":{"booked":"","bookedTimestamp":1745258459982,"id":"empty","status":0},"slot5":{"booked":"","bookedTimestamp":1745088826609,"id":"empty","status":0},"slot6":{"booked":"","bookedTimestamp":1745088830222,"id":"empty","status":0}},"users":{"test1":{"place":"row1/slot1","wallet":{"balance":"100"}},"test2":{"wallet":{"balance":"98.00"}},"test3":{"wallet":{"balance":"100"}},"test4":{"wallet":{"balance":"100"}},"test5":{"wallet":{"balance":"100"}},"test6":{"place":"row1/slot4","wallet":{"balance":"102.85"}}}}

Available Occupied Booked Occupied (Unknown ID)

User ID	Balance (₹)
test1	100.00
test2	98.00
test3	100.00
test4	100.00
test5	100.00
test6	102.85

Row ID	Parking Spots
row1	Test2 Test1
row2	

Mobile App (works on android and ios):