Smart Parking System with Real-Time Slot Availability

PROJECT REPORT

Submitted by

C. VENKATA SOMNATH (192212181)
P. HARIKRISHNA (192212358)

Under the guidance of

Dr. R. DHANALAKSHMI

in partial fulfilment for the completion of course

CSA0279 – C PROGRAMMING FOR BEGINEERS

NOVEMBER-2024

TITLE:

Smart Parking System with Real-Time Slot Availability

PROBLEM STATEMENT:

Urban areas are increasingly facing challenges related to traffic congestion and parking shortages, leading to frustration among drivers and increased pollution levels. Traditional parking systems often lack real-time information about available parking slots, forcing drivers to waste time searching for parking and contributing to overall inefficiency in city traffic management. This not only affects the convenience of finding a parking space but also results in increased carbon emissions and stress for motorists.

TASKS :

- Deployment of IoT sensors to monitor parking slot occupancy in real-time.
- A mobile application for users to view available parking spaces and navigate to them.
- Automated updates on slot availability and reservation capabilities for drivers. Integration with city traffic management systems to optimize traffic flow.
- User-friendly interface for parking facility owners to manage their lots and monitor usage.
- Payment processing options for seamless transactions and reservations.
- Data analytics features for analyzing parking trends and optimizing space usage.

OUTCOME :

The outcome of this project is to develop an automated Consumer Complaint Handling System that accepts customer complaints, categorizes them based on the type of issue,

assigns them to the appropriate department or personnel, and tracks the resolution progress. The system should also allow for feedback on the resolution process to ensure continuous improvement. By automating the complaint handling process, businesses can enhance customer experience, reduce manual workload, and improve the quality of support services.

AIM

The aim of the project is to design and implement an intelligent parking solution that leverages IOT technology and mobile applications to address the growing challenges of parking in urban areas. The system focuses on providing real-time information about parking slot availability to drivers, ensuring efficient parking space utilization, reducing traffic congestion, and minimizing the environmental impact caused by vehicle idling and searching for parking spaces.

ABSTRACT

Urban traffic congestion and parking shortages are pressing issues that result in driver frustration, wasted time, and increased carbon emissions. Traditional parking systems lack the capability to provide real-time updates on parking availability, further exacerbating these problems. This project proposes a Smart Parking System that leverages IoT sensors to monitor parking slot occupancy and a mobile application to display real-time slot availability to drivers. By integrating IoT technology and data analytics, the system aims to streamline parking management, optimize the use of parking spaces, and enhance the overall urban driving experience. This solution not only improves convenience for drivers but also contributes to sustainable urban development by reducing unnecessary traffic and emissions.

The Smart Parking System with Real-Time Slot Availability is an innovative solution to address the growing challenges of parking management in urban areas. With the increasing number of vehicles and limited parking spaces, drivers often face frustration and delays while searching for available parking. This contributes to unnecessary traffic congestion, fuel wastage, and heightened pollution levels. The proposed system leverages Internet of Things (IoT) technology to monitor parking slot occupancy and provide real-time updates to drivers through a mobile application. IoT sensors installed in parking slots detect vehicle presence and transmit the data to a centralized server, where it is processed and displayed

to users. This real-time information allows drivers to locate available parking spaces efficiently, saving time and reducing stress. Additionally, the system helps optimize the utilization of parking spaces and supports environmental sustainability by minimizing vehicle idling and emissions. By integrating advanced technologies with a user centric design, the project offers a practical approach to improving urban mobility and aligns with the vision of creating smarter and more sustainable cities.

INTRODUCTION

Parking in urban areas has become a critical issue as cities continue to expand and vehicle ownership rises. The limited availability of parking spaces, combined with inefficient traditional parking systems, often leaves drivers struggling to find a place to park. This not only wastes time but also exacerbates traffic congestion and increases fuel consumption. The cumulative effect of these inefficiencies contributes significantly to environmental degradation in the form of higher carbon emissions and air pollution.

Current parking systems lack the ability to provide real-time information about slot availability, forcing drivers to search aimlessly for parking, which can lead to frustration and stress. With the increasing focus on smart cities and sustainable development, there is a growing need for intelligent systems that can address these challenges effectively.

The Smart Parking System with Real-Time Slot Availability aims to bridge this gap by combining IoT technology, data analytics, and mobile platforms to revolutionize the way parking is managed in urban areas. Using IoT sensors installed at parking slots, the system detects the occupancy status of each slot and updates the information in real time. This data is then transmitted to a central server and displayed on a user-friendly mobile application, allowing drivers to find available parking slots quickly and efficiently.

This innovative solution not only enhances convenience for drivers but also improves the overall efficiency of parking space utilization. By reducing the time spent searching for parking, the system helps decrease traffic congestion and fuel consumption, promoting a cleaner and greener environment. Furthermore, the integration of real-time data collection and analysis provides valuable insights for parking lot operators and city planners, enabling better resource allocation and long-term infrastructure improvements.

As urban areas continue to grow, implementing smart systems like this one is essential for creating more sustainable and livable cities. The Smart Parking System represents a significant step forward in transforming traditional parking into a modern, efficient, and environmentally friendly process.

CODE IMPLEMENTATION

Modules:

- Sensor Module
  *Simulates IoT sensors to detect whether a parking slot is occupied or vacant.
  *Updates the status of each parking slot dynamically.

- Server Module
  *Acts as the central system to store and process data received from sensors.
  *Provides real-time updates on slot statuses.

- Mobile Application Module (Simulated in C)
  *Displays the status of parking slots to users.
  *Allows users to view, search, or update slot statuses (for administrative purposes).

- User Interaction Module
  *Provides a menu-based interface for users to interact with the system.
  *Allows actions like checking slot availability, reserving slots, and releasing slots.

Data Storage:

The parking slot data is stored in an array or file, depending on the scale of implementation.

• Array Storage: Each slot is represented by an index in an array, where the value indicates occupancy (0 for vacant, 1 for occupied).

• File Storage: Data can be written to and read from a file for persistence (e.g., saving slot statuses after program termination).

Flow:

- Main menu for user interaction.
- Sub-menus for various functionalities.

Code Outline:

1. **Struct Definition**
2. **Functions**
    - Register Complaint
    - View Complaints
    - Update Complaint Status
3. **Main Menu**

PROGRAM

```
#include <stdio.h>
#define MAX_SLOTS 10 // Define maximum parking slots

void displaySlots(int slots[], int n) {
   printf("\nParking Slot Status:\n");
   for (int i = 0; i < n; i++) {
      printf("Slot %d: %s\n", i + 1, slots[i] == 0 ? "Available" : "Occupied");
   }
}

void updateSlot(int slots[], int slot, int status) {
   slots[slot - 1] = status;
   printf("\nSlot %d has been updated to %s.\n", slot, status == 0 ? "Available" :
"Occupied");
}

int main() {
   int slots[MAX_SLOTS] = {0}; // Initialize all slots as available
   int choice, slot, status;
```

```
    printf("Welcome to the Smart Parking System!\n");

    while (1) {
        printf("\nMenu:\n");
        printf("1. Display Parking Slots\n");
        printf("2. Update Slot Status\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                displaySlots(slots, MAX_SLOTS);
                break;
            case 2:
                printf("\nEnter slot number to update (1-%d): ", MAX_SLOTS);
                scanf("%d", &slot);
                if (slot < 1 || slot > MAX_SLOTS) {
                    printf("Invalid slot number! Try again.\n");
                    break;
                }
                printf("Enter status (0 for Available, 1 for Occupied): ");
                scanf("%d", &status);
                if (status != 0 && status != 1) {
                    printf("Invalid status! Use 0 or 1.\n");
                    break;
                }
                updateSlot(slots, slot, status);
                break;
            case 3:
                printf("Exiting the system. Have a great day!\n");
                return 0;
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
    return 0;
}
```

RESULT:

```
Welcome to the Smart Parking System!

Menu:
1. Display Parking Slots
2. Update Slot Status
3. Exit
Enter your choice: 1

Parking Slot Status:
Slot 1: Available
Slot 2: Available
Slot 3: Available
Slot 4: Available
Slot 5: Available
Slot 6: Available
Slot 7: Available
Slot 8: Available
Slot 9: Available
Slot 10: Available
```

```
Welcome to the Smart Parking System!

Menu:
1. Display Parking Slots
2. Update Slot Status
3. Exit
Enter your choice: 2

Enter slot number to update (1-10): 5
Enter status (0 for Available, 1 for Occupied): 1

Slot 5 has been updated to Occupied.
```

ENGINEERING STANDARDS

1. **C Language Standards**
2. **Software Development Standards**
3. **Needs Assessment and Planning**
4. **Sensor Selection and Placement**
5. **Communication Network Design**
6. **Data Management and Processing**
7. **User Interface Development**
8. **Security Implementation**
9. **Power Management**
10. **Compliance with Standards and Regulations**
11. **Integration and Testing**

1. C Language Standard

I. ISO/IEC 9899:1990 (C90 or ANSI C)

- First standardization of the C language.
- Introduced features such as:
- Function prototypes.
- Standard library functions (e.g., stdio.h, stdlib.h).
- Improved type checking.
- Known as ANSI C when adopted by the American National Standards Institute (ANSI).

II. ISO/IEC 9899:1995 (C95)

- A minor revision of C90.
- Included:

- Normative corrections.
- Wide character support (wchar_t, <wchar.h>, <wctype.h>).
- New library functions (e.g., wprintf, mbtowc).

III. ISO/IEC 9899:1999 (C99)

- Significant update to the language.
- New features:
- Inline functions (inline keyword).
- Variable-length arrays (VLAs).
- restrict keyword for pointer optimization.
- Improved support for floating-point operations.
- New standard headers: <stdbool.h>, <stdint.h>, <complex.h>.
- Enhanced support for single-line comments (//).
- Initial declarations inside for loops.

IV. ISO/IEC 9899:2011 (C11)

- Focused on improving concurrency and safety.
- Added:
- Multi-threading support (e.g., <threads.h>, <atomic.h>).
- Anonymous structures and unions.
- _Generic keyword for generic programming.
- Improved bounds-checking functions (e.g., <stdnoreturn.h>).
- Static assertions (_Static_assert).

- Made VLAs optional for compiler implementation.

V. ISO/IEC 9899:2018 (C17 or C18)

- Minor revision of C11.
- Addressed defect reports (technical corrections).
- No new language features or libraries introduced.

VI. ISO/IEC 9899:2023 (C23)

- Latest revision focused on usability and modernization.
- Introduced:
- New keywords like typeof.
- Improved support for embedded systems.
- Added Unicode character literals and string prefixes (u8, u, U).
- Enhanced diagnostic capabilities.
- New library functions and macros for better standard compliance.

2. Software Development Standards

I. **ISO/IEC 12207:** Software Development Life Cycle (SDLC) Processes

- Defines a framework for software development processes, including planning, design, implementation, testing, deployment, and maintenance.
- Ensures that each stage of the project is systematically approached and thoroughly documented.

II. **IEEE 829:** Software Testing Documentation

- Establishes standards for creating test-related documents like test plans, test cases, and test reports.
- Ensures comprehensive testing to identify and address system errors or vulnerabilities.

III. **ISO/IEC 25010**: Software Quality Model

- Functionality: Ensures the system meets customer requirements, such as registering and tracking complaints.
- Reliability: The system should consistently handle a large volume of complaints without failures.
- Usability: Design an intuitive interface for ease of use by both customers and administrators.
- Performance Efficiency: Optimize the system for quick response times and low resource usage.
- Security: Protect sensitive consumer data from unauthorized access.
- Maintainability: Ensure the system can be easily updated or modified.
- Portability: Enable the system to run on various platforms.

IV. **ISO/IEC 26514:** User Documentation

- Provides guidelines for creating user-friendly documentation for both system operators and consumers.
- Ensures that users can easily navigate and operate the system with minimal training.

V. **IEEE 730:** Software Quality Assurance

- Outlines the process for ensuring that the software meets its quality objectives.
- Includes reviews, audits, and evaluations to maintain high-quality standards throughout development.

3. Needs Assessment and Planning

- **Identify Requirements: Determine the needs of the parking area.**

- **Feasibility Study: Assess the project's technical, financial, and operational feasibility**

4. Sensor Selection and Placement

- **Choose Sensors**: Select appropriate sensors such as ultrasonic, infrared, or RFID tags based on accuracy, range, and cost.
- **Placement Strategy**: Plan the optimal placement of sensors to cover all parking spots effectively, avoiding overlaps and blind spots.

5. Communication Network Design

- **Select Communication Protocols**: Choose the right communication protocols like Wi-Fi, LoRaWAN, Zigbee, or 5G, considering range, power consumption, and data needs.
- **Network Topology**: Design a network topology that ensures reliable and efficient data transmission between sensors and the central system.

6. Data Management and Processing

- **Cloud and Edge Computing**: Implement cloud computing for data storage and edge computing for real-time processing.
- **Database Setup**: Establish a robust database management system (DBMS) for efficient data storage and retrieval.

7. User Interface Development

- **Mobile Applications**: Develop user-friendly mobile apps for real-time updates, reservations, and payments.
- **Web Portals**: Create web portals for administrative control and system monitoring.

8. Security Implementation

- **Data Encryption**: Ensure secure data transmission and storage through encryption.
- **Access Control**: Implement access control mechanisms to restrict unauthorized access.
- **Regular Audits**: Conduct security audits to maintain compliance with standards.

9. Power Management

- **Energy-Efficient Devices**: Use low-power sensors and devices to minimize energy consumption.
- **Renewable Energy**: Consider renewable energy sources, like solar panels, for sustainable power solutions.

10. Compliance with Standards and Regulations

- **ISO Standards**: Adhere to relevant ISO standards, such as ISO/IEC 20922 (LoRaWAN), ISO/IEC 27001 (Information Security Management), and ISO/IEC 30141 (IoT Reference Architecture).
- **Local Regulations**: Ensure compliance with local laws and guidelines for smart city infrastructure and data privacy.

11. Integration and Testing

- **APIs and Interoperability**: Develop APIs to integrate the system with other smart city components like traffic management.
- **System Testing**: Conduct comprehensive testing of the entire system, including sensors, communication networks, data management, and user interfaces.

12. Deployment and Maintenance

- **Gradual Rollout**: Deploy the system in phases to monitor performance and address any issues promptly.

FUTURE SCOPE:

1. Integration with Advanced IoT Technology

- **Vehicle Detection**: Use high-precision sensors like LiDAR or advanced cameras to detect the type and size of vehicles.
- **Optimized Slot Allocation**: Implement algorithms that allocate parking slots based on vehicle type, ensuring efficient space utilization and easy navigation.

2. Mobile Application Development

- **User-Friendly Interface**: Design intuitive mobile applications for users to view real-time parking availability, book slots, and navigate to available parking spots.
- **Features**: Include features like slot reservations, payment options, and navigation assistance.

3. AI and Machine Learning

- **Predictive Analytics**: Use machine learning algorithms to analyze historical parking data and predict future parking demand.
- **Optimal Parking Suggestions**: Provide users with recommendations on the best parking locations based on current and predicted availability.

4. Scalability

- **Modular Design**: Develop the system with a modular approach to easily scale up for larger parking lots.
- **Integration**: Ensure the system can integrate with city-wide traffic management and other smart city systems for a cohesive urban infrastructure.

5. Smart Payment Systems

- **Automated Payment Solutions**: Implement RFID or QR code-based payment systems for seamless transactions.
- **Payment Integration**: Integrate with various payment platforms to offer multiple payment options to users.

6. Sustainability Focus

- **Solar-Powered IoT Devices**: Incorporate solar panels to power the IoT devices, reducing dependency on conventional power sources.
- **Green Initiatives**: Track and reduce carbon emissions by promoting eco-friendly practices like preferential parking for electric vehicles and carpooling.

CONCLUSION:

The Smart Parking System with Real-Time Slot Availability provides an innovative solution to the challenges of parking management in urban areas. By leveraging IoT sensors and mobile applications, the system offers real-time data on parking slot availability, enabling drivers to save time, reduce stress, and minimize their environmental footprint.

This project not only addresses the inefficiencies of traditional parking systems but also contributes to reducing traffic congestion and carbon emissions, promoting sustainable urban living. The implementation of such systems can significantly enhance the convenience and efficiency of city transportation infrastructure, paving the way for smarter, greener cities. Future enhancements, such as AI integration, predictive analytics, and seamless payment systems, will further expand the capabilities and benefits of this system.