**SMART**

**PARKING SYSTEM**

**A PROJECT REPORT**

**Submitted By :- Dheeraj Kumar**

**Date :- 28-08-2024**

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **TOPIC** | **PAGE NO** |
| *1* | Introduction | 1 |
| *2* | Objective | 3 |
| *3* | Tools/Environment | 4 |
| *4* | Program Code | 7 |
| *5* | Input and Output Screens | 17 |
| *6* | Limitations of the Project | 23 |
| *7* | Future Applications of the Project | 24 |
| *8* | Bibliography | 25 |

**INTRODUCTION**

A Smart Parking System is an advanced software platform designed to optimize the management and operation of parking facilities. Parking spaces are essential in urban areas, helping to manage the flow of vehicles and reduce congestion. Effective management of these spaces is crucial to ensuring that drivers can find parking easily and efficiently. A Smart Parking System provides parking facilities with tools to automate the monitoring, allocation, and tracking of parking spaces, ultimately enhancing the user experience and improving operational efficiency.

SMART PARKING SYSTEM

The evolution of parking facilities from simple, manual operations to sophisticated, technology-driven environments has created the need for advanced management systems. A Smart Parking System addresses this need by providing a comprehensive suite of tools and functionalities that cover a wide range of tasks, including real-time parking space monitoring, automated payment processing, and user management. By integrating various parking operations into a single platform, a Smart Parking System reduces inefficiencies, enhances accuracy, and significantly improves the overall user experience.

**KEY FEATURES OF THE SMART PARKING SYSTEM**

Smart parking systems offer a range of features designed to enhance the efficiency of parking management and improve the user experience. Here are some of the key features:

1. Real-Time Occupancy Detection:
   * Sensors: Utilizes various sensors (ultrasonic, infrared, magnetic) to detect whether parking spaces are occupied or available.
   * Updates: Provides real-time updates on space availability, allowing drivers to find open spots quickly.
2. Mobile Applications:
   * Navigation Assistance: Offers directions to available parking spaces via a mobile app.
   * Reservations: Allows users to reserve and pay for parking spots in advance.
3. Dynamic Pricing:
   * Pricing Models: Implements variable pricing based on demand, time of day, or location.
   * Revenue Optimization: Adjusts prices to maximize revenue and manage parking space utilization effectively.
4. Digital Signage:
   * Real-Time Information: Displays available parking spaces and directions on electronic signs located throughout the city.
   * Guidance: Helps drivers navigate to the nearest available parking spots.
5. Data Analytics:

Usage Patterns: Analyzes parking data to identify peak usage times, high-demand areas, and overall space utilization.

**OBJECTIVES**

The objectives of smart parking systems are designed to address common parking challenges and leverage technology to improve urban mobility. Here are the primary objectives:

1. Reduce Parking Search Time:
   * Efficiency: Minimize the time drivers spend searching for available parking spots, leading to a more efficient and stress-free parking experience.
2. Alleviate Traffic Congestion:
   * Flow Improvement: Decrease traffic congestion caused by drivers circling to find parking by providing real-time information on space availability.
3. Optimize Space Utilization:
   * Maximize Use: Improve the utilization of available parking spaces through better management and dynamic pricing strategies.
4. Increase Revenue:
   * Revenue Generation: Enhance the revenue potential from parking fees by implementing dynamic pricing models and efficient space management.
5. Enhance User Experience:
   * Convenience: Provide a user-friendly experience with features such as mobile apps for space reservations, real-time notifications, and navigation assistance.
6. Support Sustainable Urban Development:
   * Environmental Impact: Reduce vehicle emissions and pollution by minimizing the time vehicles spend idling while searching for parking.

**TOOLS AND ENVIORNMENT**

**HARDWARE REQUIREMENTS**

Processor: Minimum Pentium IV 2.4 GHZ

RAM: At Least 100 MB

Disk Space: At Least 500 MB

**SOFTWARE REQUIREMENTS**

Operating System: Windows,IOS,LINUX,Etc.

Code Compiler : Visual Code Studio / Dev C++/ Turbo C++/Etc.

**ENTITY RELATIONSHIP DIAGRAM**

**VEHICLE**

**OCCUPIES**

**PARKING SLOT**

**GENERATES**

**PARKING\_SESSION**

**DATA FLOW DIAGRAM**

SMART PARKING SYSTEM

DSPLAY AVAILABILTIY ENTER PARKING

**USER**

**VEHICLE**

PAYMENT EXIT PARKING

DATA DATA

DATABASE

**PROGRAM CODE**

#include <iostream>

#include <string>

#include <fstream>

#include <sys/stat.h> // For file existence check

using namespace std;

struct Vehicle {

string licensePlateNumber;

string vehicleType;

int parkingSpotNumber;

};

const int MAX\_VEHICLES = 50;

Vehicle parkingLot[MAX\_VEHICLES];

int numVehicles = 0;

void displayMenu() {

cout << "\n\*\*\*\*\*\*\*\*\*\*|| SMART PARKING SYSTEM ||\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\n1. Park Vehicle" << endl;

cout << "2. Unpark Vehicle" << endl;

cout << "3. Display Available Parking Spots" << endl;

cout << "4. Display Occupied Parking Spots" << endl;

cout << "5. Search for Vehicle" << endl;

cout << "6. Save and Exit" << endl;

cout << "-------------------" << endl;

}

bool fileExists(const string& filename) {

struct stat buffer;

return (stat(filename.c\_str(), &buffer) == 0);

}

void parkVehicle() {

if (numVehicles < MAX\_VEHICLES) {

cout << "Enter license plate number: ";

cin >> parkingLot[numVehicles].licensePlateNumber;

cout << "Enter vehicle type (car, motorcycle, bicycle): ";

cin >> parkingLot[numVehicles].vehicleType;

parkingLot[numVehicles].parkingSpotNumber = numVehicles + 1;

numVehicles++;

cout << "Vehicle parked successfully!" << endl;

} else {

cout << "Parking lot is full!" << endl;

}

}

void unparkVehicle() {

if (numVehicles == 0) {

cout << "No vehicles to unpark!" << endl;

return;

}

string licensePlateNumber;

cout << "Enter license plate number: ";

cin >> licensePlateNumber;

for (int i = 0; i < numVehicles; i++) {

if (parkingLot[i].licensePlateNumber == licensePlateNumber) {

for (int j = i; j < numVehicles - 1; j++) {

parkingLot[j] = parkingLot[j + 1];

}

numVehicles--;

cout << "Vehicle unparked successfully!" << endl;

return;

}

}

cout << "Vehicle not found!" << endl;

}

void displayAvailableParkingSpots() {

if (MAX\_VEHICLES - numVehicles == 0) {

cout << "No available parking spots!" << endl;

} else {

cout << "Available parking spots: " << MAX\_VEHICLES - numVehicles << endl;

}

}

void displayOccupiedParkingSpots() {

if (numVehicles == 0) {

cout << "No occupied parking spots!" << endl;

} else {

cout << "Occupied parking spots: " << endl;

for (int i = 0; i < numVehicles; i++) {

cout << "Parking spot " << parkingLot[i].parkingSpotNumber << ": " << parkingLot[i].licensePlateNumber << " (" << parkingLot[i].vehicleType << ")" << endl;

}

}

}

void searchForVehicle() {

if (numVehicles == 0) {

cout << "No vehicles in the parking lot!" << endl;

return;

}

string licensePlateNumber;

cout << "Enter license plate number: ";

cin >> licensePlateNumber;

for (int i = 0; i < numVehicles; i++) {

if (parkingLot[i].licensePlateNumber == licensePlateNumber) {

cout << "Vehicle found! Parking spot: " << parkingLot[i].parkingSpotNumber << endl;

return;

}

}

cout << "Vehicle not found!" << endl;

}

void saveDataToFile() {

ofstream file("parking\_lot\_data.txt");

if (file.is\_open()) {

file << numVehicles << endl;

for (int i = 0; i < numVehicles; i++) {

file << parkingLot[i].licensePlateNumber << " " << parkingLot[i].vehicleType << " " << parkingLot[i].parkingSpotNumber << endl;

}

file.close();

cout << "Data saved to file successfully!" << endl;

} else {

cout << "Unable to open file!" << endl;

}

}

void loadDataFromFile() {

if (fileExists("parking\_lot\_data.txt")) {

ifstream file("parking\_lot\_data.txt");

if (file.is\_open()) {

file >> numVehicles;

for (int i = 0; i < numVehicles; i++) {

file >> parkingLot[i].licensePlateNumber >> parkingLot[i].vehicleType >> parkingLot[i].parkingSpotNumber;

}

file.close();

cout << "Data loaded from file successfully!" << endl;

} else {

cout << "Unable to open file!" << endl;

}

} else {

cout << "No previous data found, starting fresh." << endl;

}

}

int main() {

loadDataFromFile();

int choice;

while (true) {

displayMenu();

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

parkVehicle();

break;

case 2:

unparkVehicle();

break;

case 3:

displayAvailableParkingSpots();

break;

case 4:

displayOccupiedParkingSpots();

break;

case 5:

searchForVehicle();

break;

case 6:

saveDataToFile();

cout << "Exiting..." << endl;

return 0;

default:

cout << "Invalid choice. Please try again." << endl;

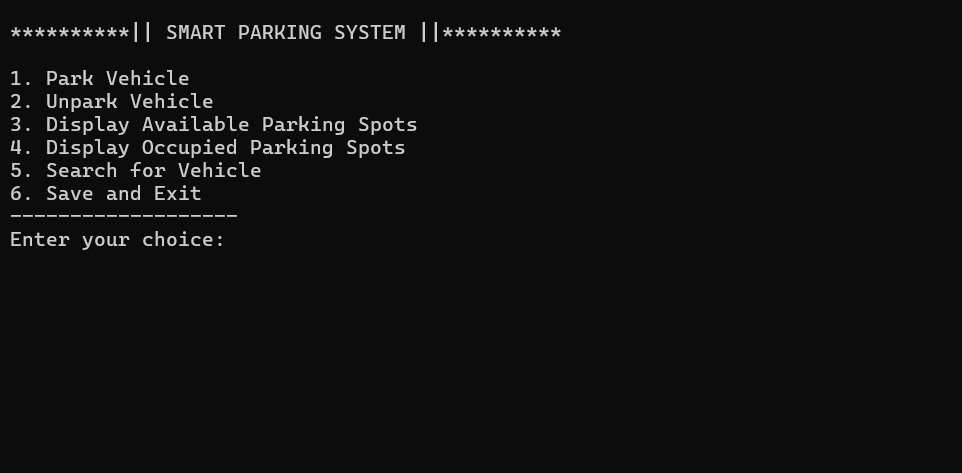
}

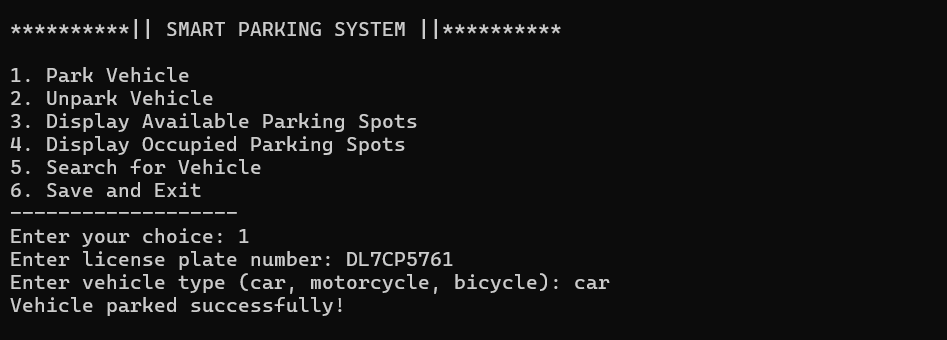
}

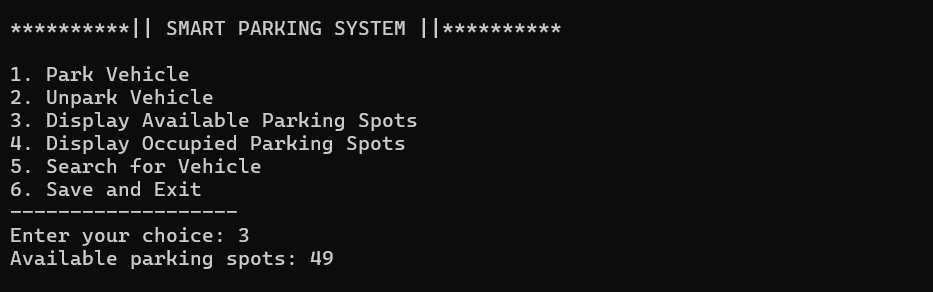
return 0;

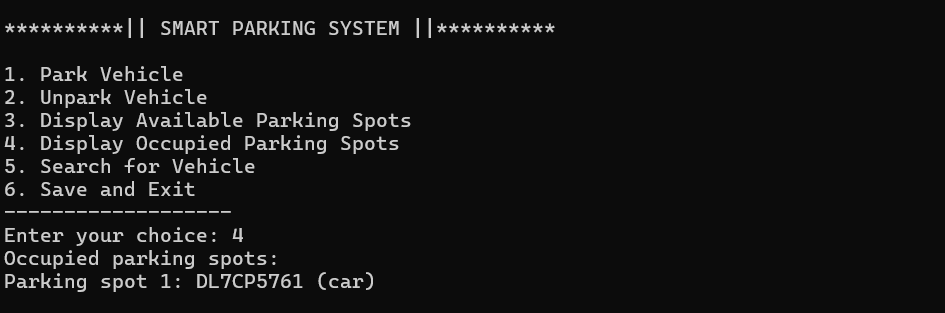
}

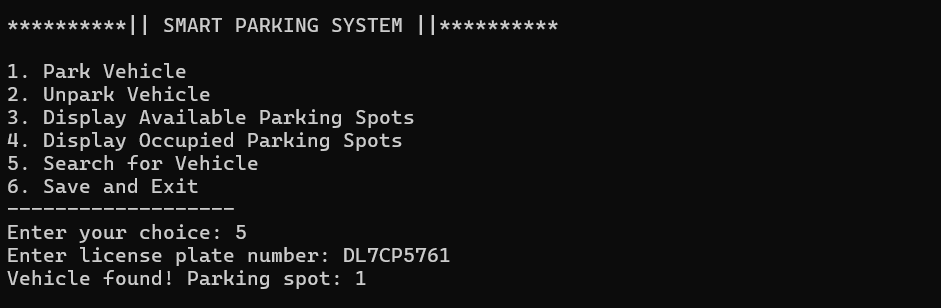
**INPUT / OUTPUT SCREENS**

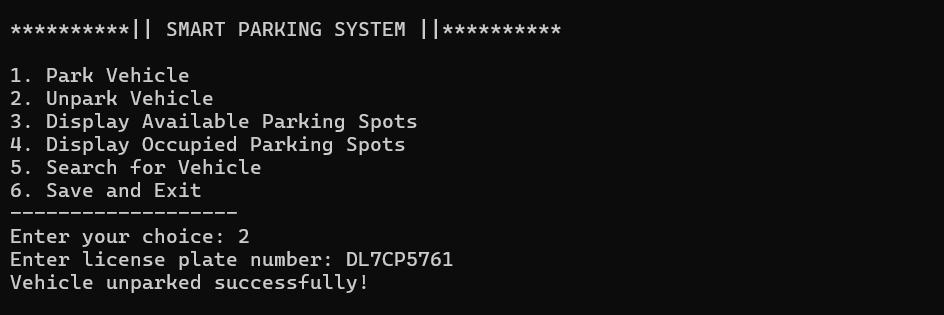
**MAIN MANU**

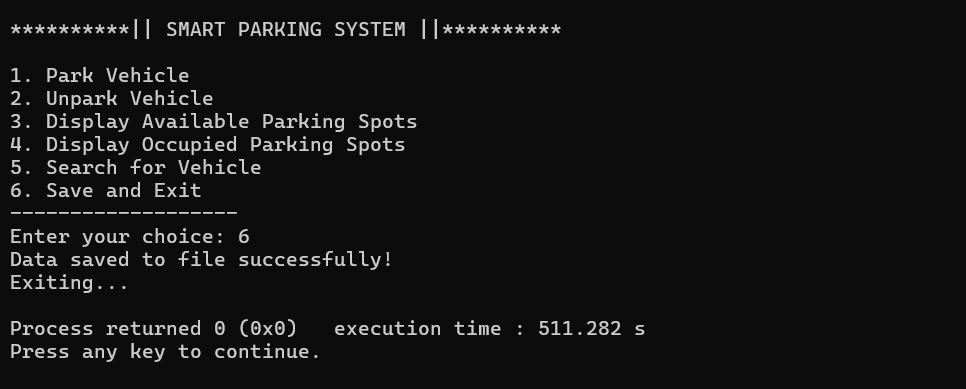
**PARK VEHICLE** 

**DISPLAY AVAILABLE PARKING SPOTS** 

**DISPLAY OCCUPIED PARKING SPOTS**

**SEARCH FOR VEHICLE** 

**UNPARK VEHICLE** 

**SAVE AND EXIT**

**LIMITATIONS OF SMART PARKING SYSTEM**

Despite their numerous advantages, smart parking systems also come with certain limitations that can impact their effectiveness and adoption. Here are some of the key limitations:

1. High Initial Costs:
   * Installation Expenses: The deployment of sensors, communication networks, and management software requires significant upfront investment.
   * Maintenance Costs: Ongoing maintenance, including sensor calibration and system upgrades, adds to the overall expense.
2. Complex Implementation:
   * Infrastructure Challenges: Retrofitting existing parking infrastructure with smart technologies can be complex and time-consuming, particularly in older cities.
   * Integration Issues: Integrating smart parking systems with existing urban infrastructure, such as traffic management and public transportation systems, can be challenging.
3. Data Privacy Concerns:
   * Personal Data: The collection and storage of user data, including parking habits and location information, raise privacy concerns and require robust data protection measures.
4. System Reliability:
   * Sensor Malfunction: Sensors and other hardware components can fail or provide inaccurate data, leading to incorrect information about space availability.

**FUTURE APPLICATION OF THE PROJECT**

The future applications of smart parking systems are vast and will likely evolve alongside advancements in technology, urban planning, and the increasing complexity of smart cities. Here are some potential future applications:

1. **Integration with Autonomous Vehicles:**
   * **Self-Parking Vehicles:** Smart parking systems could work in tandem with autonomous vehicles, allowing them to identify available parking spaces and park themselves without human intervention.
   * **Vehicle Retrieval:** Autonomous vehicles could be summoned from parking spaces to pick up passengers, reducing the need for dedicated parking near high-demand areas.
2. **Smart Mobility Hubs:**
   * **Multimodal Integration:** Smart parking systems could be integrated into smart mobility hubs that combine parking with access to public transportation, bike-sharing, and ride-hailing services, promoting seamless urban mobility.
   * **EV Charging Stations:** Future parking systems could incorporate more advanced electric vehicle (EV) charging infrastructure, allowing for dynamic allocation of spaces based on charging needs.
3. **Dynamic Urban Space Utilization:**
   * **Flexible Parking Spaces:** As urban areas continue to densify, smart parking systems could enable dynamic reconfiguration of parking spaces, converting them into other uses (like pop-up retail or green spaces) during off-peak hours.
   * **Real-Time Space Allocation:** Systems could allocate parking spaces dynamically based on real-time demand, allowing for temporary repurposing of spaces for events or delivery zones.

**BIBLIOGRAPHY**

**Books**

* Let Us C by Yashavant Kanetkar.
* Let us C++ by Yashavant Kanetkar.
* C in Depth by S.K Srivastava.
* The C++ Programming Language By Bjarne Stroustrup.

**Websites**

* [www.google.com](http://www.google.com)
* [www.youtube.com](http://www.youtube.com)
* [www.w3schools.com](http://www.w3schools.com)
* [www.geeksforgeeks.com](http://www.geeksforgeeks.com)