**VEHICLE PARKING RESERVATION SYSTEM**

Project submitted to the

SRM University – AP, Andhra Pradesh for the partial fulfilment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**ROSELIN THOMSI G.C (AP23110010272)**

**LOHITH KUMAR REDDY G (AP23110010261)**

**SANTHOSH (AP23110010255)**

**REVANTH (AP23110010273)**



Under the Guidance of

**K. KAVITHA RANI Ma’am**

**SRM University–AP**

**Neeru Konda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**November 2024**

# Certificate

Date: 16-Nov-24

This is to certify that the work present in this Project entitled “**VEHICLE PARKING RESERVATION SYSTEM**” has been carried out BY **ROSELIN THOMSI G.C (AP23110010272), LOHITH KUMAR REDDY G (AP23110010261)**, **SANTHOSH (AP23110010255)**, **REVANTH (AP23110010273)** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in **School of Engineering and Sciences**.

(Signature)

K. KAVITHA RANI

Associate Professor,

Affiliation.

**Acknowledgements:**

I would want to sincerely thank everyone who contributed to make this vehicle parking system project a success and to show my appreciation. Their assistance and direction have been crucial in forming our undertaking.

First and foremost, I would like to express my sincere gratitude to my project supervisor, "KAVITHA RANI.K ma’am," for their invaluable advice and steadfast support during this project's development. Their management, presentation, and programming experience has been a consistent source of motivation.

I would want to express my gratitude to every one of my friends and classmates who participated, helped, and offered insightful comments and suggestions throughout the development stages. Their managerial style, ideology, collaborative spirit, and constructive criticism have all been crucial in helping to improve the Vehicle Parking System's features and functionality.

Finally, but just as importantly, I want to thank my friends for their continuous support and patience during the project schedule. Without the combined efforts and encouragement of everyone listed above, this project would not have been achievable. I appreciate your contribution to this journey.

# Table of Contents

Certificate................................................................................................................................. ii

Acknowledgements...............................................................................................................iii

Table of Contents...................................................................................................................iv

Abstract.................................................................................................................................. v

Statement of Contributions...................................................................................................vi

Figures....................................................................................................................... vii

Flowcharts……………………………………………………………………………viii

1.Introduction..................................................................................................................... ix

1.1 Class Hierarchy...............................................................................................................x

1.2 Time Tracking............................................................................................................xi

1.3 User Interaction……………………………………………………………. xii

1.4 Fee Calculation………………………………………………………. xiii

1. Methodology................................................................................................................xvi
   1. Inheritance in C++...........................................................................................................xv
   2. **Ctime………………………………………………………………………………xvi**
   3. **Vectors……………………………………………………………………………xvii**
2. Discussion...................................................................................................................xviii
3. Concluding Remarks...................................................................................................... xix
4. Future Work.................................................................................................................. xx

References.............................................................................................................................xix

# Abstract

**The presented C++ code a simple Vehicle Parking System,**

**Accommodating both cars and bikes. The system leverages object-oriented**

**Programming principles and utilizes time tracking for parking duration** **Calculations. The parking slots for cars and bikes are managed separately, with functionalities for parking, retrieving, and displaying the current parking status. The code employs vectors to store entry times, aiding in calculating parking durations for fee computations.**

**Key features include:**

**Car Parking:**

**Reservation and retrieval of parking slots for cars.**

**Time tracking for entry and exit timestamps.** **Calculation of parking duration and associated fees.**

**Bike Parking:**

**Parking and retrieval of bike slots.**

**Time tracking for entry and exit times.**

**Computation of parking duration and corresponding charges.**

**The system is user-interactive, with a menu-driven interface for users to park or retrieve vehicles and check the current parking status. The fee calculation is based on a predefined pricing constant, and the system ensures slot availability is maintained.**

**This Vehicle Parking System provides a foundation for expanding functionalities and incorporating additional features, making it adaptable for more complex parking management scenarios.**

# Statement of Contributions

**I am incredibly appreciative and happy to be involved with this project. With the assistance of my friends and teammates, I have given it my all. As a team member and project associate for this project, I am presenting the following roles and contributions.**

Ideology Development. – {Roselin}

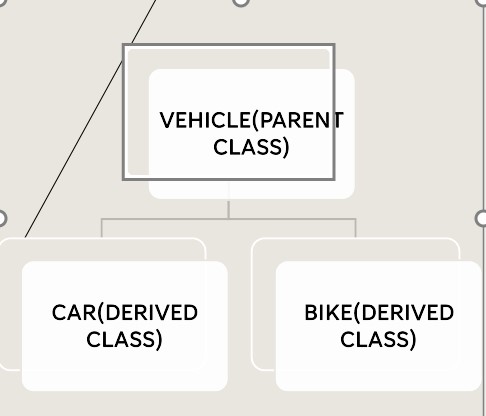
PowerPoint Presentation. – [Lohith}

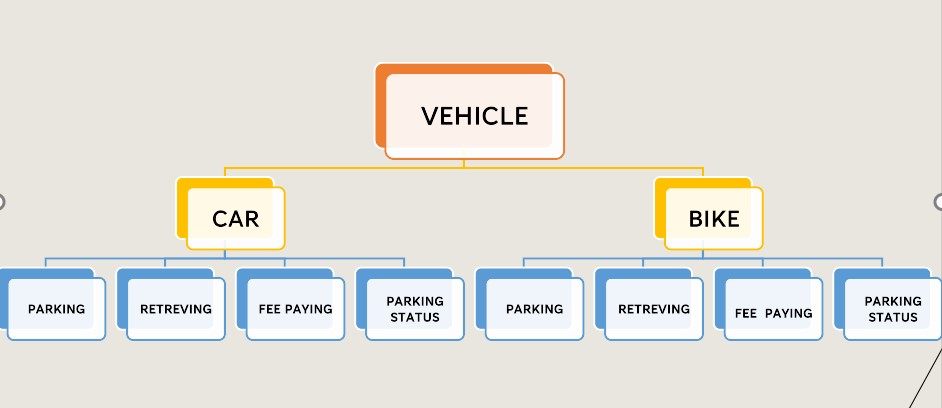
Coding /Programming of The Project-{Roselin, Lohith, Santhosh, Revanth}

Project Presentation. {Roselin, Lohith, Santhosh, Revanth}

Project Report. – {Roselin, Lohith}

# Figures





7

# Introduction

The provided C++ code introduces a Vehicle Parking System designed to manage parking slots for both cars and bikes. The implementation employs object-oriented programming principles to create a modular and extensible solution. The system utilizes time tracking mechanisms to record entry and exit times, facilitating the calculation of parking durations for fee computation.

Key Components:

1. Class Hierarchy:

Parking System Class: Serves as the base class, encapsulating common properties and methods for both cars and bikes, such as parking slots and availability.

Car Class (Derived from Parking System): Manages specific functionalities related to car parking, including reservation, retrieval, and status display. It incorporates time tracking for entry and exit timestamps.

Bike Class (Derived from Parking System): Similar to the Car class, it handles bike specific parking operations, tracking entry and exit times for duration calculation.

1. Time Tracking:

The `time\_t` data type is utilized to record entry and exit times for both cars and bikes. These timestamps are stored in vectors (`p` and `q`) to enable efficient calculation of parking durations.

1. User Interaction:

The main function engages users through a command-line interface, allowing them to choose between parking a car, retrieving a car, checking car parking status, parking a bike, retrieving a bike, and checking bike parking status.

1. Fee Calculation:

The system calculates parking fees based on a predefined constant (`PRICE`). The fee computation considers the parking duration and differentiates between short-term and long-term stays.

Usage:

Users are prompted to input the type of vehicle they intend to park (Car or Bike).

Depending on the vehicle type, the user can then choose from a menu of options, including parking, retrieving, or checking the parking status of the respective vehicle.

Methodology

**Inheritance in C++**

* **Inheritance is a feature or a process in which new classes are created from the existing classes.**
* **The new class created is called “derived class” or “child class” and the existing class is known as the “base class” or “parent class”.**

* **The derived class now is said to be inherited from the base class.**

* **When we say derived class inherits the base class, it means, the derived class inherits all the properties of the base class, without changing the properties of base class and may add new features to its own.**

* **These new features in the derived class will not affect the base class. The derived class is the specialized class for the base class.**

* **Sub Class: The class that inherits properties from another class is called Subclass or Derived Class.**

* **Super Class: The class whose properties are inherited by a subclass is called Base Class or Superclass.**

**Types of Inheritance in C++**

**There are 5 types of inheritance in C++ programming language:**

**1.Single Inheritance**

**2.Multiple Inheritance**

**3.Hierarchical Inheritance**

**4.Multilevel Inheritance**

**5.Hybrid Inheritance**

**Hierarchical Inheritance:**

**The inheritance in which a single base class inherits multiple derived classes is known as the Hierarchical Inheritance. This inheritance has a tree-like structure since every class acts as a base class for one or more child classes. The visibility mode for each derived class is specified separately during the inheritance and it accesses the data members accordingly.**

**Syntax class class\_A**

**{**

**// class definition**

**};**

**class class\_B: visibility\_mode class\_A**

**{**

**// class definition**

**};**

**class class\_C: visibility\_mode class\_A**

**{**

**// class definition**

**};**

**class class\_D: visibility\_mode class\_B**

**{**

**// class definition**

**};**

**class class\_E: visibility\_mode class\_C**

**{**

**// class definition**

**};**

**The subclasses class\_B and class\_C inherit the attributes of the base class class\_A. Further, these two subclasses are inherited by other subclasses class\_D and class\_E respectively.**

**Example**

**The following example illustrates Hierarchical Inheritance in C++:**

**#include <iostream> using namespace std;**

**// base class class electronicDevice**

**{**

**public:**

**// constructor of the base class 1 electronicDevice()**

**{**

**cout << "I am an electronic device.\n\n";**

**}**

**};**

**// derived class inheriting base class class Computer: public electronicDevice**

**{};**

**// derived class inheriting base class class Linux\_based : public electronicDevice**

**{};**

**int main ()**

**{**

**// create object of the derived classes**

**Computer obj1; // constructor of base class will be called Linux\_based obj2; // constructor of base class will be called return 0;**

**}**

**Output**

**I am an electronic device.**

**I am an electronic device.**

**CTIME:**

**In C++, the <ctime> header provides several functions for working with time related operations. Here are some commonly used functions from this header:**

**1.time\_t time (time\_t\* timer):**

**Returns the current calendar time as a time\_t object.**

**If timer is not a null pointer, the result is also stored in the object it points to.**

**Example:**

**time\_t now = time (NULL);**

**2.char\* ctime(const time\_t\* timer):**

**Converts the given time represented as time\_t to a string in the format "Day Mon dd hh:mm:ss yyyy".**

**The returned string is a pointer to an internal static array, so it should be used or copied before subsequent calls to ctime.**

**Example:**

**time\_t now = time (NULL); char\* dt = ctime(&now);**

**3.struct tm\* localtime(const time\_t\* timer):**

**Converts the given time represented as time\_t to a tm structure representing local time.**

**The tm structure has fields like year, month, day, hour, minute, second, etc.**

**Example: time\_t now = time (NULL); struct tm\* localTime = localtime(&now);**

**4.struct tm\* gmtime(const time\_t\* timer):**

**Similar to localtime, but converts the given time to Coordinated Universal Time (UTC). Example: time\_t now = time (NULL); struct tm\* utcTime = gmtime(&now);**

**5.size\_t strftime(char\* str, size\_t count, const char\* format, const struct tm\* timeptr):**

**Formats the time represented by the tm structure into a character string. The format is specified by the format string.**

**The result is stored in the character array pointed to by str, and at most count characters are stored.**

**Example: time\_t now = time (NULL); struct tm\* localTime = localtime(&now); char buffer [80]; strftime(buffer, sizeof(buffer), "%Y-%m-%d %H: %M: %S", localTime); These functions are part of the C++ Standard Library and are commonly used for tasks such as timestamping, logging, and handling date and time information in various applications.**

**VECTOR:**

In C++ programming, the <vector> header file provides the implementation of a dynamic array, which is known as a vector. Vectors are part of the Standard Template Library (STL) and offer dynamic arrays with some additional features compared to traditional arrays in C++.

Here are some key aspects of the <vector> header file:

1. **Dynamic Array Implementation:**

Vectors are essentially dynamic arrays that can resize themselves automatically when elements are added or removed. This dynamic resizing makes them more flexible than static arrays.

1. **Template Class:**

The vector is a template class, meaning it can store elements of any data type. This makes it a versatile container that can hold integers, floating-point numbers, custom objects, and more.

1. **Common Operations:**

The <vector> header provides various member functions for common operations like adding elements (push\_back), removing elements (pop\_back), accessing elements by index (at, operator []), getting the size of the vector (size), and checking if the vector is empty (empty).

#include <iostream>

#include<vector>

Using namespace std;

int main () {

// Declare a vector of integers

vector<int> myVector; // Add elements to the vector

myVector.push\_back(10);

myVector.push\_back(20);

myVector.push\_back(30); // Access elements

cout << "Element at index 1: " << myVector.at (1) << endl;

// Iterate through the vector

for (int i : myVector) {

cout << i << " ";

}

return 0;

}

4**. Iterator Support:**

Vectors support iterators, which allow you to iterate through the elements of the vector. This provides a convenient way to loop through the vector and perform operations on its elements.

#include <vector>

#include <iostream> Using namespace std;

int main () {

vector<int> myVector = {10, 20, 30, 40, 50}; // Using iterators to traverse the vector

for (auto it = myVector.begin(); it! = myVector.end(); ++it) {

cout << \*it << " ";

}

return 0;

}

5. **Efficient Dynamic Memory Management:**

Vectors manage memory efficiently, automatically handling memory allocation and deallocation. This can save developers from manual memory management concerns.

The <vector> header file is part of the C++ Standard Library, and including it allows you to use the vector class and its associated functions in your C++ programs.

Discussion:

#include <iostream>

#include <string>

#include<ctime> #include<vector> using namespace std;

const int MAX\_CARPARKING\_SLOTS = 10; const int MAX\_BIKEPARKING\_SLOTS =10;

const int PRICE = 100;

class ParkingSystem {public:

string parkingCarSlots[MAX\_CARPARKING\_SLOTS];

int availableCarSlots=10;

vector<time\_t>p;

string parkingBikeSlots[MAX\_BIKEPARKING\_SLOTS];

int availableBikeSlots=10; vector<time\_t>q;

public:

ParkingSystem () { for (int i = 0; i < MAX\_CARPARKING\_SLOTS; ++i) {

parkingCarSlots[i] = "";

}

for (int i = 0; i < MAX\_BIKEPARKING\_SLOTS; ++i) { parkingBikeSlots[i] = "";

}

}

};

class Car:public ParkingSystem

{

public: time\_t entrytime;

void parkCar() { entrytime=time (NULL); time\_t now = time (0); char\* dt = ctime(&now); cout << "Entry Time: " << dt <<"\n\n\n\n\n"<< endl;

if (availableCarSlots > 0) { cout << "Enter car registration number: "; string regNumber; cin >> regNumber; for (int i = 0; i < MAX\_CARPARKING\_SLOTS; ++i) {

if (parkingCarSlots[i] == "") { parkingCarSlots[i] = regNumber;

availableCarSlots--; cout << "Car parked at slot " << i + 1 << endl;

p.insert(p.begin() +i,entrytime);

return;

}

}

} else {

cout << "Parking is full. No available slots." << endl;

}

}

void retrieveCar() {

time\_t exittime=time (NULL);

cout << "Enter slot number to retrieve the car: "; int slotNumber; cin >> slotNumber;

if (slotNumber >= 1 && slotNumber <= MAX\_CARPARKING\_SLOTS) {

if (parkingCarSlots[slotNumber - 1]! = "") {

time\_t now = time (0);

char\* dt = ctime(&now); cout << "Exit Time: " << dt << endl; cout << "Car with registration number " << parkingCarSlots[slotNumber - 1]

<< " retrieved from slot " << slotNumber << endl; double n=difftime(exittime,p[slotNumber-1]); cout<<"Duration:"<<difftime(exittime,p[slotNumber-1]) <<endl; cout<<"Fee to be paid..Rs:"<<endl;

if (n < 10)

{

cout<<"\t\t~~~~~~~~ The total charges ~~~~~~~~~~~\t\t"<<endl; cout<<PRICE;

}

else

{

cout<<(n/10) \*PRICE;

}

parkingCarSlots[slotNumber - 1] = "";

availableCarSlots++;

} else {

cout << "No car found at slot " << slotNumber << endl;

}

} else {

cout << "Invalid slot number. Please enter a valid slot number." << endl;

}

}

void displayParkingStatus() { cout << "Parking Status:" << endl; for (int i = 0; i < MAX\_CARPARKING\_SLOTS; ++i) {

cout << "Slot " << i + 1 << ": "; if (parkingCarSlots[i]! = "") { cout << parkingCarSlots[i];

} else {

cout << "Empty";

}

cout << endl;

}

}

};

class Bike: public ParkingSystem

{

public: time\_t entrybiketime;

void parkBike() {

entrybiketime=time (NULL); time\_t now = time (0);

char\* dt = ctime(&now); cout << "Entry Time: " << dt << endl; if (availableBikeSlots > 0) { cout << "Enter bike registration number: "; string regNumber; cin >> regNumber;

for (int i = 0; i < MAX\_BIKEPARKING\_SLOTS; ++i) {

if (parkingBikeSlots[i] == "") { parkingBikeSlots[i] = regNumber;

availableBikeSlots--; cout << "Bike parked at slot " << i + 1 << endl;

q.insert(q.begin() +i,entrybiketime);

return;

}

}

} else {

cout << "Parking is full. No available slots." << endl;

}

}

void retrieveBike() {

time\_t exitbiketime=time (NULL);

cout << "Enter slot number to retrieve the bike: "; int slotNumber; cin >> slotNumber;

if (slotNumber >= 1 && slotNumber <= MAX\_BIKEPARKING\_SLOTS) {

if (parkingBikeSlots[slotNumber - 1]! = "") {

time\_t now = time (0);

char\* dt = ctime(&now); cout << "Exit Time: " << dt << endl; cout << "Bike with registration number " << parkingBikeSlots[slotNumber - 1]

<< " retrieved from slot " << slotNumber << endl; double K=difftime(exitbiketime,q[slotNumber-1]); cout<<"Duration:"<<difftime(exitbiketime,q[slotNumber-1]) <<endl; cout<<"Fee to be paid..Rs:"<<endl;

if (K < 10)

{

cout<<"~~~~~~~~ The total charges ~~~~~~~~~~~"<<endl; cout<<PRICE<<endl;

}

else

{

cout<<(K/10) \*PRICE<<endl;

}

parkingBikeSlots[slotNumber - 1] =""; availableBikeSlots++;

}

else {

cout << "No bike found at slot " << slotNumber << endl;

}

} else {

cout << "Invalid slot number. Please enter a valid slot number." << endl;

}

}

void displayParkingStatus() {

cout << "Parking Status:" << endl; for (int i = 0; i < MAX\_BIKEPARKING\_SLOTS; ++i) {

cout << "Slot " << i + 1 << ": "; if (parkingBikeSlots[i]! = "") { cout << parkingBikeSlots[i];

} else {

cout << "Empty";

}

cout << endl;

}

}

};

int main () {

Car c; Bike b;

while (1)

{

string s; cout<<endl;

cout<<"\t\t\t\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~WELCOME!!~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\t\t\t\t"<<endl

<<"\n"<<"Enter the vehicle you bought (CAR/BIKE):"<<endl; cin>>s;

if(s=="CAR"||s=="car")

{

int choice;

do {

cout << "\nMenu:\n1. Park Car\n2. Retrieve Car\n3. Display Parking Status\n4. Exit\n"; cout << "Enter your choice: "; cin >> choice; cout<<endl;

switch (choice) {

case 1:

c.parkCar(); break; case 2:

c.retrieveCar(); break; case 3:

c.displayParkingStatus(); break; case 4: cout << "Exiting the program.\n"; break; default:

cout << "Invalid choice. Please enter a valid option.\n";

}

} while (choice! = 4);

}

else if(s=="BIKE"||s=="bike")

{

int choice;

do {

cout << "\nMenu:\n1. Park bike\n2. Retrieve bike\n3. Display Parking Status\n4. Exit\n";

cout << "Enter your choice: "; cin >> choice;

switch (choice) { case 1:

b.parkBike(); break; case 2:

b.retrieveBike(); break; case 3:

b.displayParkingStatus(); break; case 4: cout << "Exiting the program.\n"; break; default:

cout << "Invalid choice. Please enter a valid option.\n";

}

} while (choice! = 4);

}

else

{

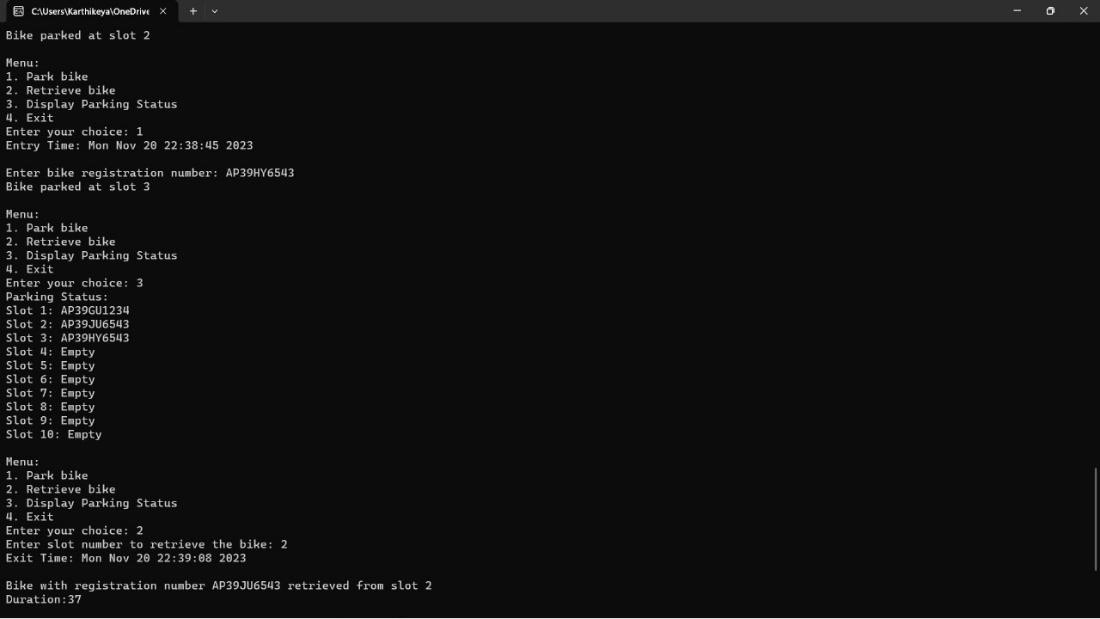
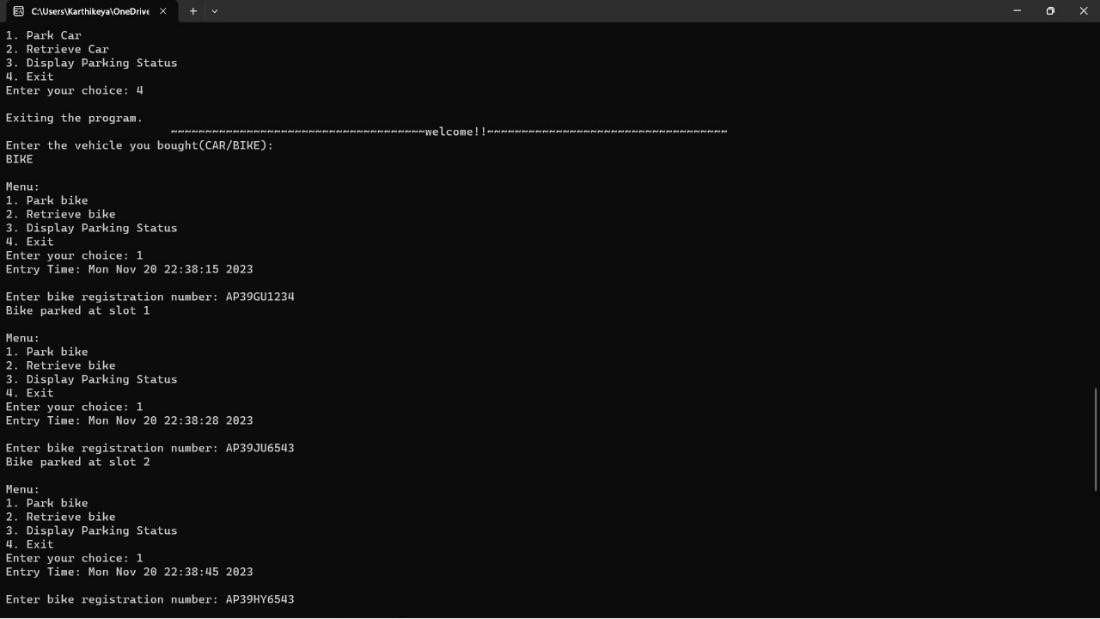
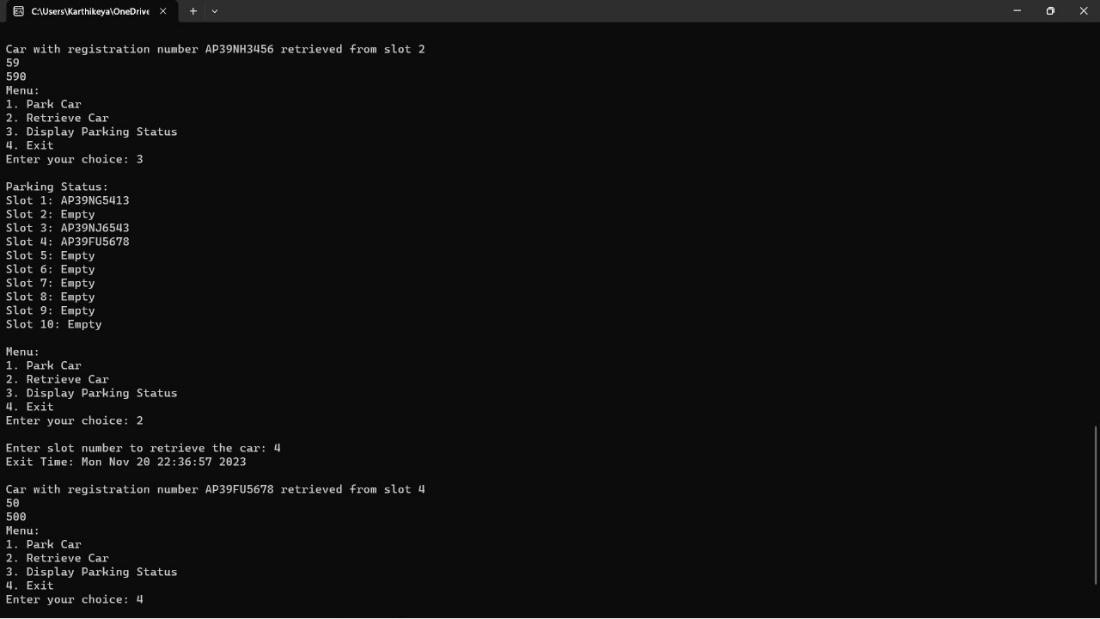
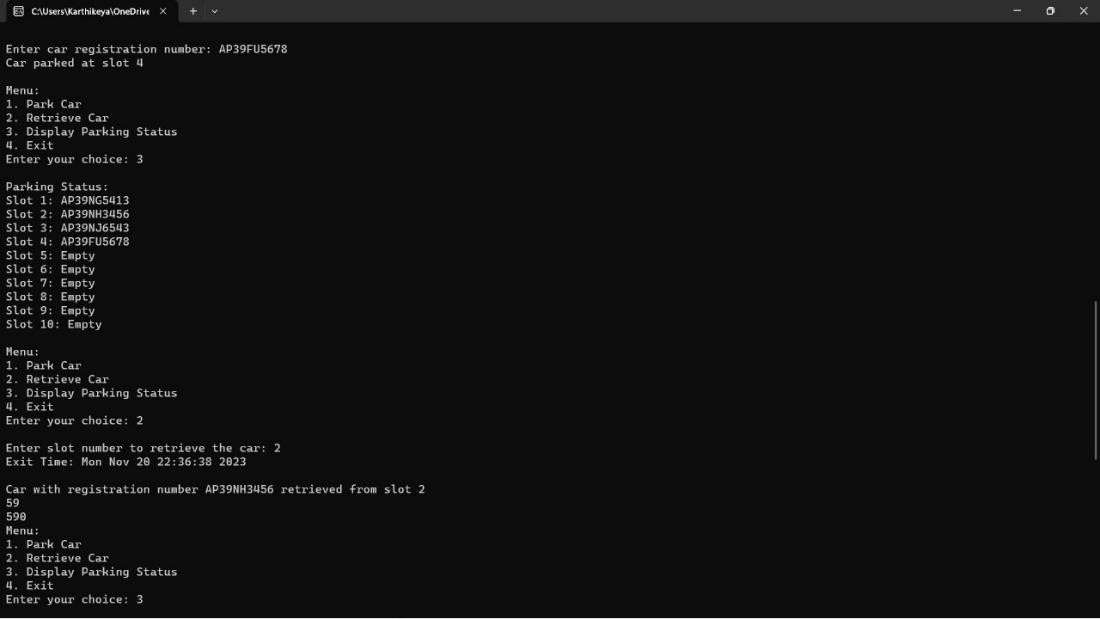
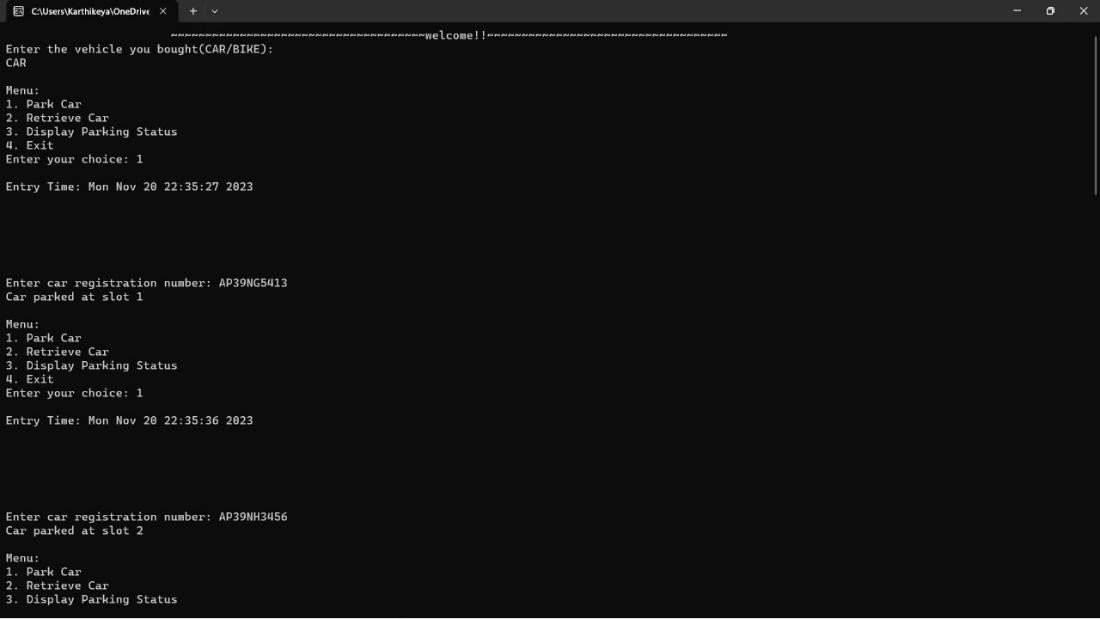
cout<<"~Invalid vehicle~"<<endl;

break;

}

}

}



**Concluding Remarks:**

In conclusion, the implemented parking management system demonstrates the effective use of C++ programming concepts and features to create a user-friendly application for managing car and bike parking. The system allows users to park, retrieve, and display the parking status for both cars and bikes. The key functionalities of the system include real-time entry and exit time tracking, efficient slot allocation, and dynamic slot availability management.

In summary, the developed parking management system serves as a foundation for a practical and functional solution to address parking challenges. It successfully combines C++ programming principles with time and vector manipulation, providing a solid base for future improvements and expansions in functionality. The project demonstrates effective object-oriented design, encapsulation, and modularization, making it a valuable learning experience and a potential starting point for a more comprehensive parking management solution.

# Future Work

* The vehicle parking reservation system has a lot of potential for growth and expansion. Here are some possible areas for future development:
* Integration with smart city infrastructure to optimize parking availability and reduce traffic congestion.
* Expansion to additional locations, such as airports, shopping and public transportation hubs.
* Incorporation of electric vehicle charging stations to support the growing trend of eco-friendly transportation.
* Development of a mobile app for easy access and convenience for users.

# References

* For Information related to Inheritance in OOPS <https://www.geeksforgeeks.org/inheritance-in-c/>

* For Information related to CTIME <https://cplusplus.com/reference/ctime/time_t/>

<https://www.geeksforgeeks.org/ctime-function-in-c-c/>

* For Information related to <https://www.geeksforgeeks.org/vector-in-cpp-stl/amp/>

THE END