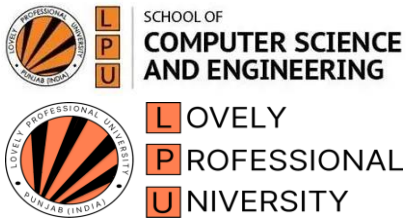

Summer Internship Training

(June 2025 - July 2025)

on

Fundamental of Data Structure using C++

Project Title: Car Parking Management System

Submitted by

Name of the Student: PONNAPALLI SURENDRA VARMA

Registration number of the Student: 12309779

Under the Guidance of

Dr. Om Prakash Yadav
Associate Professor

School of Computer Science and Engineering
Lovely Professional University

Jalandhar - Delhi G.T. Road, Phagwara, Punjab (India) – 144411

1. Introduction

This is a **C++ based Parking Management System** designed to efficiently manage vehicle entries and exits in a multi-lane parking lot. The system supports **5 lanes with 10 spots each**, allowing up to **50 vehicles** to be parked at a time. It assigns a **unique token** to each car upon entry, stores the entry time, and calculates the **parking fee** based on the duration of stay at a rate of ₹60/hour.

2. Objectives

- **Automate Vehicle Entry and Exit:**

  Streamline the process of parking by automatically assigning spots and managing vehicle movement using a structured system.

- **Generate and Manage Unique Tokens:**

  Issue a unique token to each car upon entry to ensure easy identification and secure exit operations.

- **Calculate Accurate Parking Fees:**

  Determine parking charges based on exact duration (₹60/hour), using real-time entry and exit timestamps.

- **Persist Data for Continuity:**

  Store all parked car details and revenue in external files (parked_cars.txt and revenue.txt) to maintain data across program restarts.

- **Prevent Duplicate Parking Entries:**

  Avoid errors by checking and disallowing the same vehicle from being parked multiple times simultaneously.

3. Advantages

- **Efficient Vehicle Management:**

  Automates the entire parking process, reducing manual effort and human error.

- **Accurate Time-Based Billing:**

Calculates charges precisely based on parking duration with ₹60/hour billing.

- **Data Persistence:**

  Saves all data in files, so information is not lost even if the program closes.

- **Scalable Design:**

  Easily extendable to more lanes or features due to its modular structure using C++ OOP.

4. Disadvantages / Limitations:

- **Console-Based Interface Only:**

  No graphical user interface (GUI), which may limit user-friendliness for non-technical users.

- **No Admin Login or Security:**

  Anyone can access or exit a car using a token — lacks authentication.

5. Project Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>
#include <ctime>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <cmath>

using namespace std;
```

```
struct Car {
   string number;
time_t entry_time;

   Car(string num, time_t entry = time(nullptr)) {
number = num;         entry_time = entry;
   }
};

struct ParkingSpot {
int lane;     int
position;     bool
occupied;
   Car* car;

   ParkingSpot(int l, int p) : lane(l), position(p), occupied(false), car(nullptr) {}
};

class ParkingLot {     const int lanes = 5;     const
int spots_per_lane = 10;
vector<vector<ParkingSpot*>> lanes_data;
unordered_map<string, ParkingSpot*> car_map;
unordered_map<string, string> token_to_car;
double total_revenue = 0.0;
   const double rate_per_second = 60.0 / 3600.0;

   string generateToken() { string
      token;
```

```
        do {
            token = to_string(rand() % 90000 + 10000);
} while (token_to_car.count(token));         return
token;
    }

    void loadRevenue() {
ifstream file("revenue.txt");
if (file.is_open()) {             file
>> total_revenue;
file.close();
        }
    }

    void loadParkedCars() {
ifstream file("parked_cars.txt");
if (!file.is_open()) return;         string
token, car_number;         int lane,
position;         time_t entry_time;
        while (file >> token >> car_number >> lane >> position >> entry_time) {
ParkingSpot* spot = lanes_data[lane][position];             spot->occupied = true;
            spot->car = new Car(car_number, entry_time);
car_map[car_number] = spot;
token_to_car[token] = car_number;             string
ignore_line;             getline(file, ignore_line);
        }
        file.close();
    }
```

```
public:     ParkingLot() {         srand(time(0));
for (int i = 0; i < lanes; ++i) {
vector<ParkingSpot*> row;             for (int j = 0;
j < spots_per_lane; ++j) {
row.push_back(new ParkingSpot(i, j));
          }
          lanes_data.push_back(row);
       }
       loadRevenue();
loadParkedCars();
   }

   ~ParkingLot() {         for (auto& row :
lanes_data) {             for (auto& spot :
row) {                if (spot->car) delete
spot->car;                delete spot;
          }
       }
   }

   void saveAllData() {
ofstream file("parked_cars.txt"); if
(file.is_open()) {

          for (auto& entry : token_to_car) {
string token = entry.first;                string car_number
= entry.second;               ParkingSpot* spot =
car_map[car_number];               time_t entryTime =
spot->car->entry_time;               file << "Token
number is " << token
```

```
                << "\nCar number is " << car_number
                << "\nSpot->lane #" << spot->lane
                << "\nSpor->position #"<< spot->position
                << "\nDate and Time: " << ctime(&entryTime);
        }
file.close();
    }

    ofstream revenueFile("revenue.txt");
if (revenueFile.is_open()) {
        revenueFile << "Total Revenue till now: Rs." << total_revenue;
revenueFile.close();
    }
  }

  void enterParking(string car_number) {
if (car_map.count(car_number)) {
        cout << "Car '" << car_number << "' is already parked.\n";
return;
    }

    for (int i = 0; i < lanes; ++i) {          for
(int j = 0; j < spots_per_lane; ++j) {
if (!lanes_data[i][j]->occupied) {
string token = generateToken();
lanes_data[i][j]->occupied = true;
lanes_data[i][j]->car = new Car(car_number);
car_map[car_number] = lanes_data[i][j];
token_to_car[token] = car_number;
```

```
                    cout << "Car parked successfully!\n";
                    cout << "Location: Lane " << (i + 1) << ", Position " << (j + 1) <<
"\n";                cout << "Your token is: " << token << "\n";
        return;
                }
            }
        }

        cout << "Parking Full. No space available.\n";
    }

    void formatDuration(double seconds) {        int
total_seconds = static_cast<int>(seconds);        int
hours = total_seconds / 3600;        int minutes =
(total_seconds % 3600) / 60;        int secs =
total_seconds % 60;

        if (hours > 0) cout << hours << " hr ";
if (minutes > 0) cout << minutes << " min ";
cout << secs << " sec";
    }

    void exitParking(string token) {
if (!token_to_car.count(token)) {
cout << "Invalid token.\n";
return;
        }
```

```
        string car_number = token_to_car[token];
    ParkingSpot* spot = car_map[car_number];        time_t
    now = time(nullptr);
        double duration = difftime(now, spot->car->entry_time);
    if (duration < 1) duration = 1;

        double fee = round(duration * rate_per_second);
    total_revenue += fee;

        cout << "Car " << car_number << " is exiting from Lane " << spot->lane +
1 << ", Position " << spot->position + 1 << ".\n";        cout << "Duration Parked: ";
formatDuration(duration);        cout << "\n";
        cout << "Parking Fee: Rs." << fixed << setprecision(2) << fee << "\n";

        delete spot->car;        spot-
    >car = nullptr;        spot-
    >occupied = false;
    car_map.erase(car_number);
    token_to_car.erase(token);
      }

      void findCar(string token) {
        if (!token_to_car.count(token)) {
    cout << "Invalid token.\n";            return;
        }
        string car_number = token_to_car[token];
    ParkingSpot* spot = car_map[car_number];
        cout << "Car Number: " << car_number << " found at Lane " << spot>lane
+ 1 << ", Position " << spot->position + 1 << ".\n";
      }
```

```
    void viewAllParkedCars() {
if (car_map.empty()) {
        cout << "No cars are currently parked.\n";
return;
    }
    cout << "\nList of All Parked Cars:\n";
for (auto& entry : car_map) {          string
token = "";
        for (auto& pair : token_to_car) {
if (pair.second == entry.first) {
token = pair.first;                break;
            }
        }
        ParkingSpot* spot = entry.second;
        cout << "Car: " << entry.first << ", Token: " << token
            << ", Lane: " << spot->lane + 1
            << ", Position: " << spot->position + 1
            << ", Entry Time: " << ctime(&spot->car->entry_time);
    }
  }

  void showRevenue() {
    cout << "=====================================" << endl;
    cout << "Total Revenue Collected: Rs." << fixed << setprecision(2) <<
total_revenue << endl;      cout <<
"=====================================";
  }
};
```

```
int main() {
ParkingLot lot;
int choice;     string
input;

   do {
      cout << "\n======= Parking System =======\n";
      cout << "1. Park Car\n2. Exit Car\n3. Find Car\n4. View All Parked Cars\n5. Show Revenue\n6. Exit\n";          cout << "Enter your choice: ";          cin >> choice; cin.ignore();

      switch (choice) {
case 1:
            cout << "Enter Car Number: ";
getline(cin, input);
lot.enterParking(input);              break;
case 2:
            cout << "Enter Token: ";
getline(cin, input);
lot.exitParking(input);
break;          case 3:
            cout << "Enter Token: ";
getline(cin, input);
lot.findCar(input);
break;          case 4:
            lot.viewAllParkedCars();
break;          case 5:
            lot.showRevenue();
break;          case 6:
```

```
            lot.saveAllData();
            cout << "Exiting... Data saved.\n";
break;          default:
            cout << "Invalid choice.\n";
        }

    } while (choice != 6);

    return 0;
}
```

********** code ended **********

6. Screen shot:

- This image shows car parking and after parked the car they give a token using that token we exit car parking or find car.

```
C:\Users\prave\OneDrive\Desktop\DSA PROJECT 2025>g++ project.cpp -o project

C:\Users\prave\OneDrive\Desktop\DSA PROJECT 2025>project

======= Parking System =======
1. Park Car
2. Exit Car
3. Find Car
4. View All Parked Cars
5. Show Revenue
6. Exit
Enter your choice: 1
Enter Car Number: AP A8754
Car parked successfully!
Location: Lane 1, Position 1
Your token is: 36402

======= Parking System =======
1. Park Car
2. Exit Car
3. Find Car
4. View All Parked Cars
5. Show Revenue
6. Exit
Enter your choice: 1
Enter Car Number: UP C2314
Car parked successfully!
Location: Lane 1, Position 2
Your token is: 37681

======= Parking System =======
1. Park Car
2. Exit Car
3. Find Car
4. View All Parked Cars
5. Show Revenue
6. Exit
Enter your choice:
```

- This image shows all parked cars

```
======= Parking System =======
1. Park Car
2. Exit Car
3. Find Car
4. View All Parked Cars
5. Show Revenue
6. Exit
Enter your choice: 4

List of All Parked Cars:
Car: UP C2314, Token: 37681, Lane: 1, Position: 2, Entry Time: Wed Jul 09 10:36:24 2025
Car: AP A8754, Token: 36402, Lane: 1, Position: 1, Entry Time: Wed Jul 09 10:36:10 2025
```

- If we enter the correct token the car successful exit from the parking

```
======= Parking System =======
1. Park Car
2. Exit Car
3. Find Car
4. View All Parked Cars
5. Show Revenue
6. Exit
Enter your choice: 2
Enter Token: 37681
Car UP C2314 is exiting from Lane 1, Position 2.
Duration Parked: 9 min 35 sec
Parking Fee: Rs.10.00
```

- Using the token we will check the car parking spot

```
======= Parking System =======
1. Park Car
2. Exit Car
3. Find Car
4. View All Parked Cars
5. Show Revenue
6. Exit
Enter your choice: 3
Enter Token: 27042
Car Number: AP N9183 found at Lane 1, Position 4.
Parked At: Wed Jul 09 10:53:50 2025
```

- We are only charge money Rs.60 for hour and this image show revenue

```
======= Parking System =======
1. Park Car
2. Exit Car
3. Find Car
4. View All Parked Cars
5. Show Revenue
6. Exit
Enter your choice: 5
=========================================
Total Revenue Collected: Rs.8.00
=========================================
```

7. Future Scope:

- **Integration with a Graphical User Interface (GUI):**

Replace the current console-based interface with a modern GUI using frameworks like Qt or web technologies (HTML/CSS + C++ backend) for better user experience.

- **Mobile App or Web-Based Access:**

  Extend the system to allow users to reserve or check parking slots remotely through a mobile app or website.

- **Database Integration:**

  Replace file-based storage with a robust database system (like MySQL, SQLite, or MongoDB) for better scalability, query capabilities, and secure data handling.

- **SMS/Email Notification:**

  Notify users with their token, slot number, or fee details via SMS or email using integrated APIs.

8. LinkedIn & Github (post links):

- https://www.linkedin.com/in/surendra-varma-ponnapalli/
- https://github.com/surendra7438/gcn123i4re

9. References:

- C++ Documentation – cppreference.com https://en.cppreference.com
- W3Schools – File I/O and Time in C++ https://www.w3schools.com/cpp
- YouTube Tutorials on C++ Projects