SMART PARKING SYSTEM

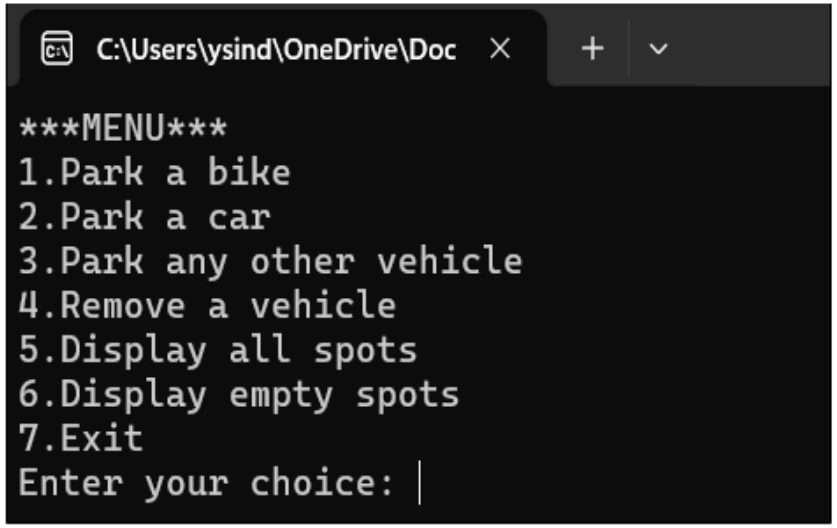
UNDER THE GUIDANCE OF : **K . KAVITHA RANI**

INTRODUCTION

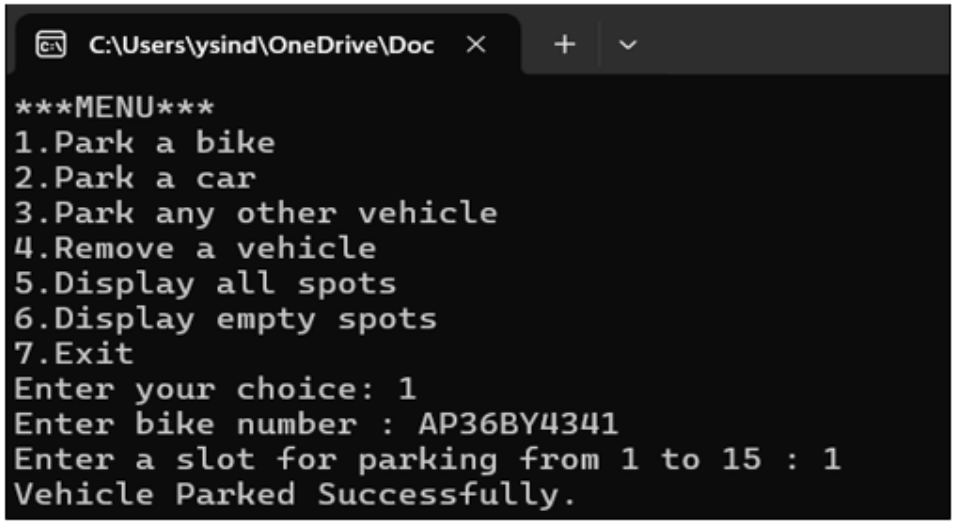
Efficient management of parking spaces is crucial in urban environments where parking availability is limited. The significant rise in vehicle numbers necessitates the implementation of structured and automated parking systems that can alleviate congestion, enhance space utilization, and provide a more streamlined parking experience for users. This project aims to develop a straightforward yet effective parking management system using C++ for a small-scale parking facility with a finite number of spots. The system accommodates various vehicle types, including bikes, cars, and others, offering a flexible and user-friendly interface for both parking and retrieving vehicles. By employing object-oriented programming (OOP) principles, the project encapsulates the characteristics and functionalities of different vehicle types, promoting modularity and code reusability.

Output explanation:-

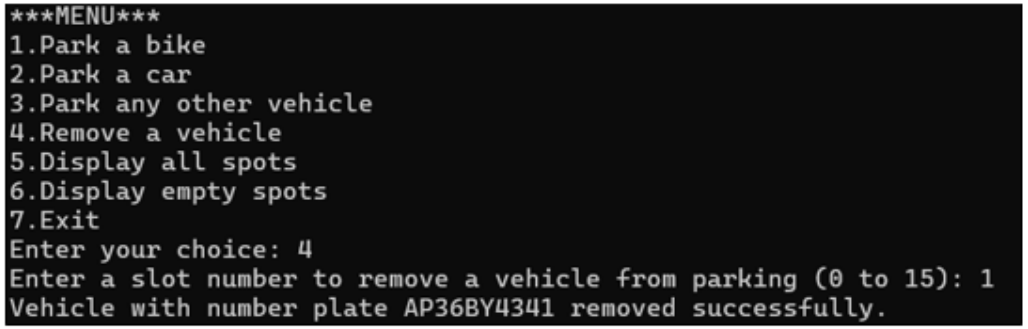
This is the main menu. User can enter their choice.



To park a bike or car enter 1 or 2 and enter the details of the bike or car i.e, number plate and spot number.



To remove a vehicle from the parking spot enter 4.And enter the spot number



To display the empty spots enter 6

```
***MENU***
1.Park a bike
2.Park a car
3.Park any other vehicle
4.Remove a vehicle
5.Display all spots
6.Display empty spots
7.Exit
Enter your choice: 6
Displaying empty spots:
Spot: 2 : Empty
Spot: 4 : Empty
Spot: 6 : Empty
Spot: 7 : Empty
Spot: 9 : Empty
Spot: 10 : Empty
Spot: 11 : Empty
Spot: 13 : Empty
```

To display all the spots enter 5

```
***MENU***
1.Park a bike
2.Park a car
3.Park any other vehicle
4.Remove a vehicle
5.Display all spots
6.Display empty spots
7.Exit
Enter your choice: 5
All spots:
Spot : 1BIKE:
 Number: AP23BY4341 Type: Bike
Spot : 2 Empty Spot
Spot : 3OTHER:
 Number: AP45CF2312 Type: Other vehicle
Spot : 4 Empty Spot
Spot : 5CAR:
 Number: AP28CV0046 Type: Car
Spot : 6 Empty Spot
Spot : 7 Empty Spot
Spot : 8OTHER:
 Number: TS24BY4459 Type: Other vehicle
Spot : 9 Empty Spot
Spot : 10 Empty Spot
Spot : 11 Empty Spot
Spot : 12BIKE:
 Number: AP23BG4029 Type: Bike
Spot : 13 Empty Spot
Spot : 14BIKE:
 Number: AP21BH7187 Type: Bike
Spot : 15CAR:
 Number: AP23YU2341 Type: Car
```

For exiting the program enter 7

```
***MENU***
1.Park a bike
2.Park a car
3.Park any other vehicle
4.Remove a vehicle
5.Display all spots
6.Display empty spots
7.Exit
Enter your choice: 7
Exiting the program...

--------------------------------
Process exited after 748.4 seconds with return value 0
Press any key to continue . . .
```

• FUTURE WORK:

- The parking management system developed in this project provides a basic framework, but there are several potential improvements for better functionality and scalability. Future enhancements could include integrating a database to store vehicle and parking records, adding timed parking with fee calculations based on duration or vehicle type, and enabling automated spot allocation to optimize space usage. Additionally, user authentication and security measures could protect sensitive data. A mobile or web interface would allow users to check availability or reserve spots remotely, while a graphical user interface (GUI) could make the system more user-friendly. Integration with IoT sensors could automate spot updates, and data analytics could provide insights into parking patterns to improve decision-making and space management.

- **Program Flow:**
- The program initializes an array of parking spot objects to manage up to 12 parking slots.• It runs in an infinite loop, presenting users with a menu.• Based on the user's choice, appropriate operations (park, remove, display) are performed.• The loop continues until the user chooses to exit.
- **Code Structure:**
- 1. Header Files and Macros : Includes standard libraries (bits/stdc++.h) and defines the maximum number of parking spots (max = 12).
- 2. Class Definitions : vehicle (Base Class): Stores vehicle details (number, type) with a virtual display() method. Derived Classes (bike, car, other): Inherit from vehicle and override display() for specific vehicle types.parkingspot: Manages parking slots, with attributes to track occupancy (occupied) and a pointer to a vehicle object.
- 3. Main Function : Initializes an array of parking spots (arr[max]).

• Conclusion:

- The parking management system effectively applies key programming and object-oriented principles to solve a practical problem. By using classes, inheritance, polymorphism, and dynamic memory management, the system efficiently handles tasks like parking, removing vehicles, displaying spot status, and identifying vehicle types. The use of a simple array for parking spots makes the system easy to manage and understand. The project achieves its goals by providing a functional, user-friendly command-line interface for real-time parking management. Its design is flexible, allowing for future feature expansions without major changes. Potential enhancements include database integration, automated spot allocation, fee calculation, and IoT capabilities. Overall, this project offers valuable experience in C++ and object-oriented design, demonstrating how software can improve efficiency and user experience in real-world applications.

THANK YOU

- **Project by :**
- **Sk.mohammad suheal (ap23110010513)**
- **Ch.Varshini(ap23110010547)**
- **y.kalyan(ap23110010521)**
- **k.Vamsi chandu(ap23110010545)**