PARKING-LOT-MANAGEMENT PROJECT REPORT

SCHOOL OF COMPUTER SCIENCE
COMPUTER SCIENCE ENGINEERING
(CSEG1041)
(BATCH – 2025-2029)

SUBMITTED BY:
Swastik
SAP ID: 590027150
ENROLLMENT NUMBER: 25010103164
SEMESTER: 1st

SUBMITTED TO:
Dr. Prashant Trivedi

2. ABSTRACT:

A car-oriented computer program manages parking facilities by streamlining and automating the tracking and control of vehicles in garages. Modern systems for managing traditional parking use automated records instead of relying solely on human effort, thereby reducing inaccuracies, speeding up processes, and making it easier to access data quickly. Our endeavor seeks to address these constraints through an organized, effective, and accessible approach designed specifically for managing every crucial aspect of parking activities.

The mechanism ensures precise documentation for every car arriving and leaving the garage by noting down information like plate numbers, types, arrival times, and departure times. Employing organized data management alongside temporal computations allows this software to determine the overall period spent on parking and subsequently calculate the appropriate parking charge. The application employs core elements of C programming like structuring data, utilizing arrays for storage, managing files through extension if necessary, incorporating conditional decision-making processes, and implementing modular design principles to enhance readability and expandability.

The venture includes crucial elements like registering vehicle entries, removing them, computing fees, showing information on every parking spot, and locating particular vehicular data. These features facilitate efficient use of resources while minimizing attendant tasks. Moreover, this system reduces mistakes made by humans through automation of task recording and cost calculation processes.

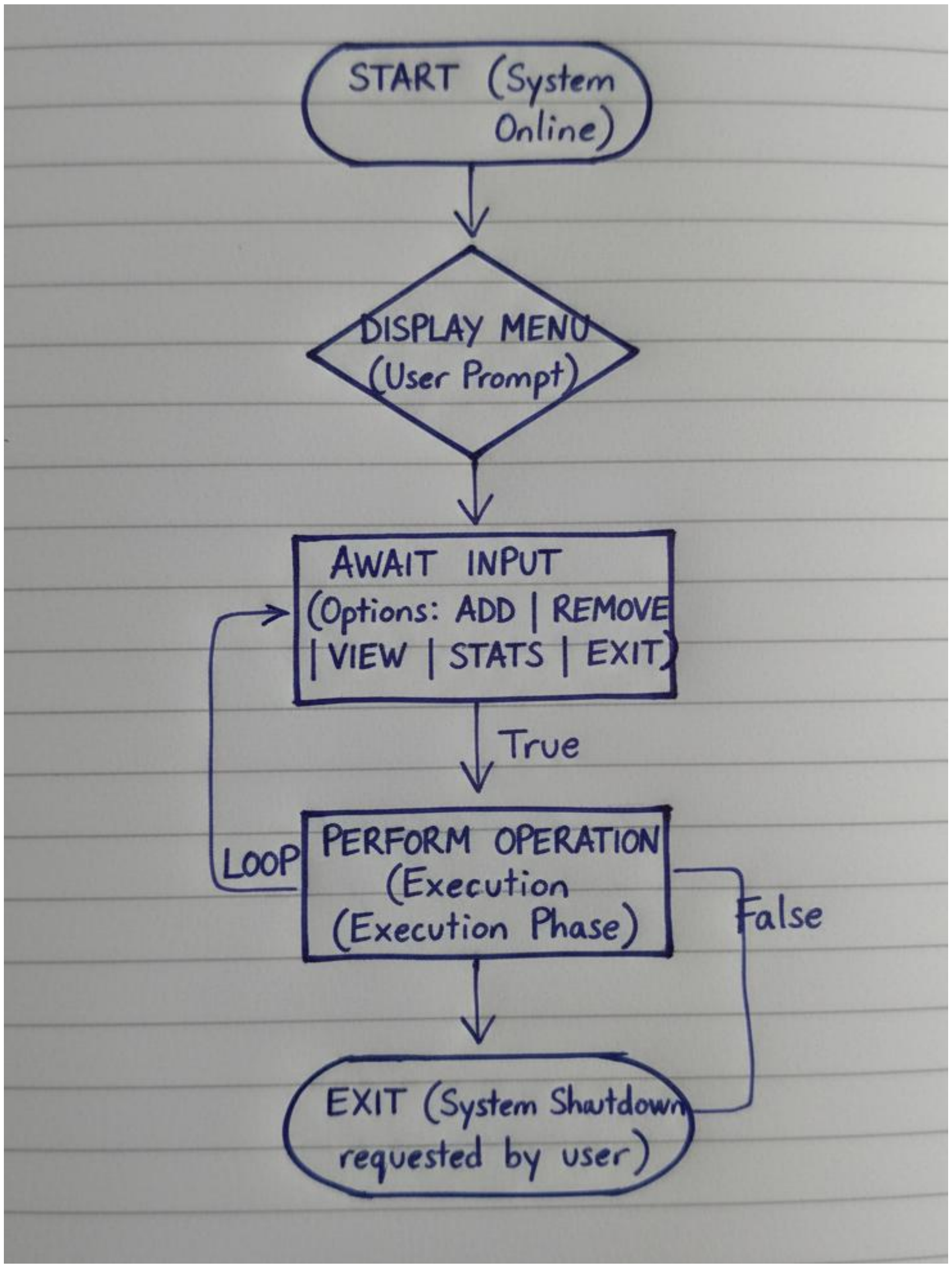
In summary, this system ensures efficient management of parking spaces by offering dependable functionality, swift operations, and structured organization in parking procedures. This showcases how C programming is applied practically for tackling real-life challenges; it could also benefit by incorporating graphical user interfaces, data storage through databases, and robust cybersecurity measures. This venture acts as an integral base for comprehending systems architecture, managing information effectively, and executing algorithms within computer programming contexts

3. PROBLEM DEFINITION:

Operating parking areas independently requires significant effort and increases chances of mistakes. Concerns encompass flawed documentation practices, challenges in monitoring open spots, and errors in calculating earnings. This endeavor aims at creating an automated parking lot management system capable of capturing information about vehicles including their registration numbers, types, and arrival times. Count both occupied spaces and remaining spots in all designated car lots. Permit drivers to depart by removing their vehicles while calculating fees accordingly. Ensure a comprehensive collection of all revenues. Show every car that is currently stationary in your viewfinder. Enhance precision in operations while optimizing resource utilization.

4. SYSTEM DESIGN:

4.1 Flowchart:


START (System Online)
DISPLAY MENU (User Prompt)
AWAIT INPUT (Options: ADD | REMOVE | VIEW | STATS | EXIT)
True
LOOP
PERFORM OPERATION (Execution (Execution Phase)
False
EXIT (System Shutdown requested by user)

4.2 ALGORITHM:

Algorithm: Add Vehicle

Check if filledSlots < totalSlots.
If not, display "Parking Full."
Alternatively, record the license plate, type of the vehicle, and input its arrival timestamp next.
Store details in vehicles array.
Increment filledSlots.

Algorithm: Remove Vehicle

Accept vehicle number from user.
Search vehicle in array.
If found: Calculate parking charges.
Add amount to totalRevenue.
Remove vehicle and shift array.
Decrement filledSlots.
Else, print "Vehicle Not Found".

Algorithm: display vehicles

Check if there’s any vehicle parked,
If true then print vehicle num, type , time
Else print no vehicle parked.

Algorithm: parking lot status

Print totalSlots
Print filledSlots
Calculate available slots = totalSlots – filledSlots
Print available slots

Algorithm: search

Take vehicle number as input
Loop through vehicles array
If match found Print vehicle details
If no match found Print "Vehicle Not Found"

Algorithm: calculate fee

```
f type = Car then rate = 20
Else if type = Bike then rate = 10
Else  rate = 30 (Truck)
fee = rate × hours
Return fee
```

Algorithm: save data to file

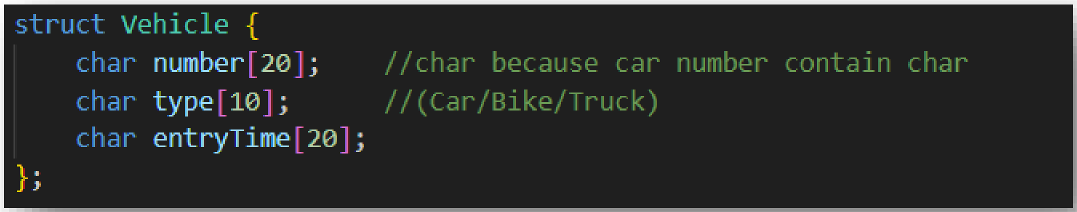
```
Open file "parkingdata.txt" in write mode
For each parked vehicle
Write number, type, entryTime into file
Close file
```

Algorithm: load data

```
Open file "parkingdata.txt" in read mode
If file does not exist then return
While data exists in file
Read number, type, time into vehicles array
Increment filledSlots
Close file
```

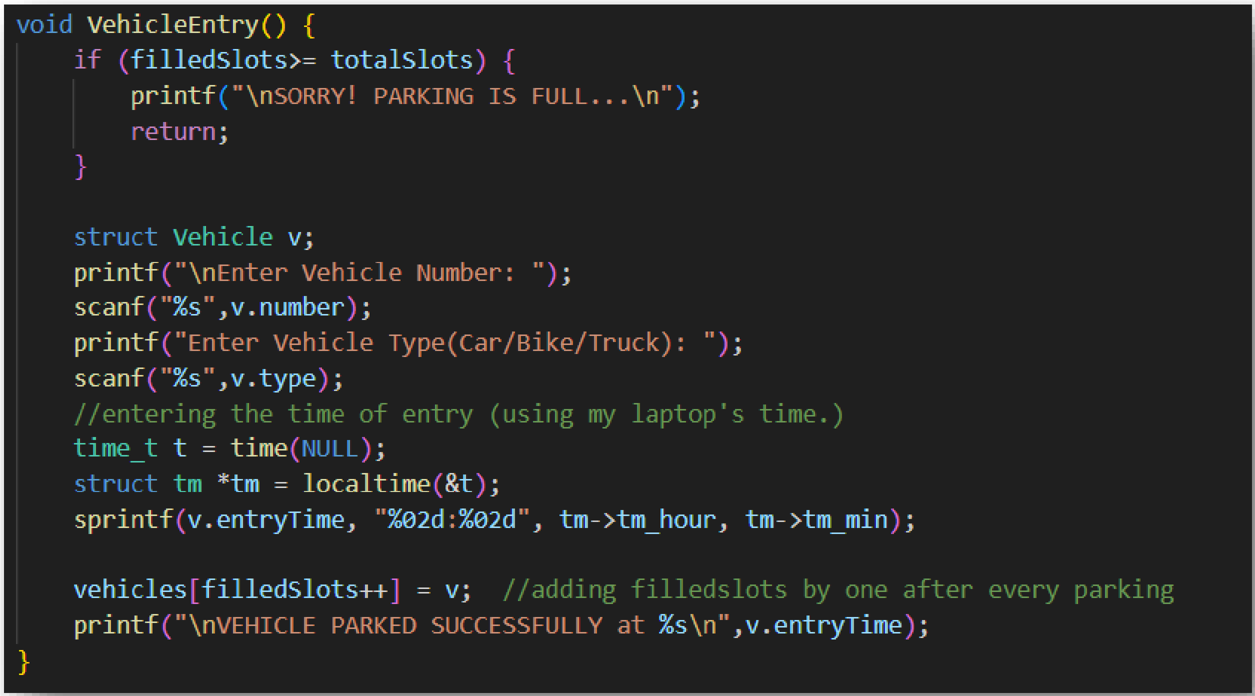
5. IMPLEMENTATION DETAILS:

5.1 Key Data Structure Used:



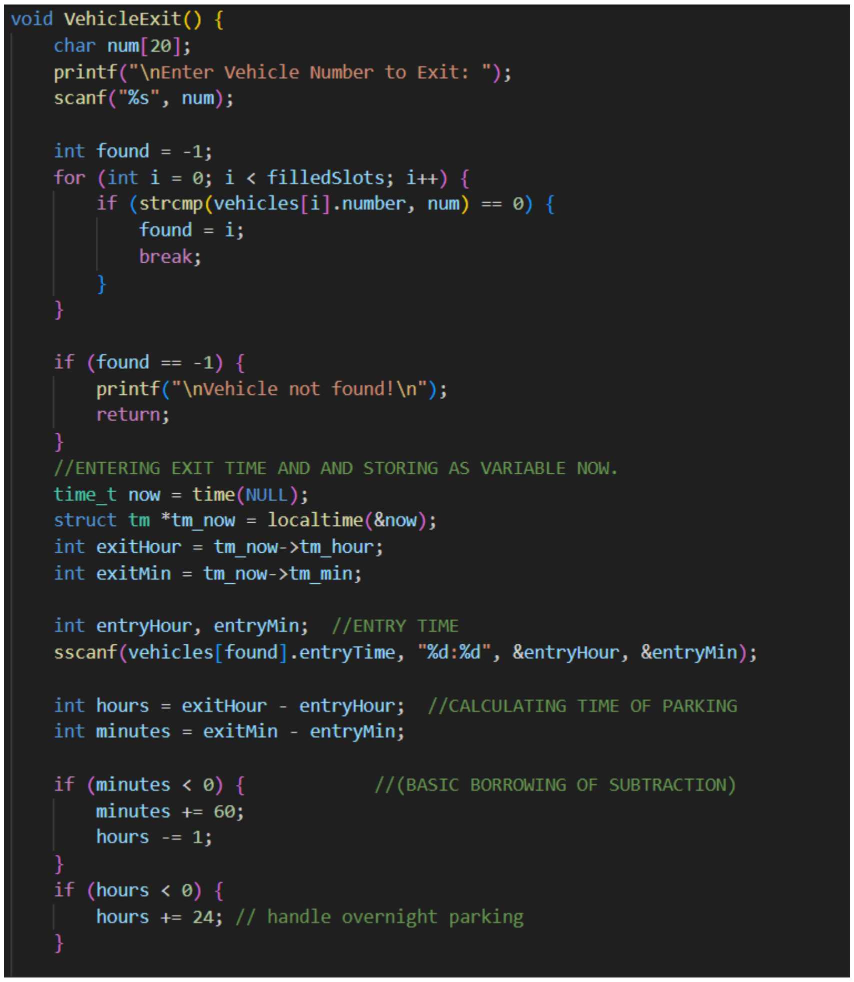
```
struct Vehicle {
    char number[20];    //char because car number contain char
    char type[10];      //(Car/Bike/Truck)
    char entryTime[20];
};
```

5.2 Code Snippet – Adding a Vehicle:

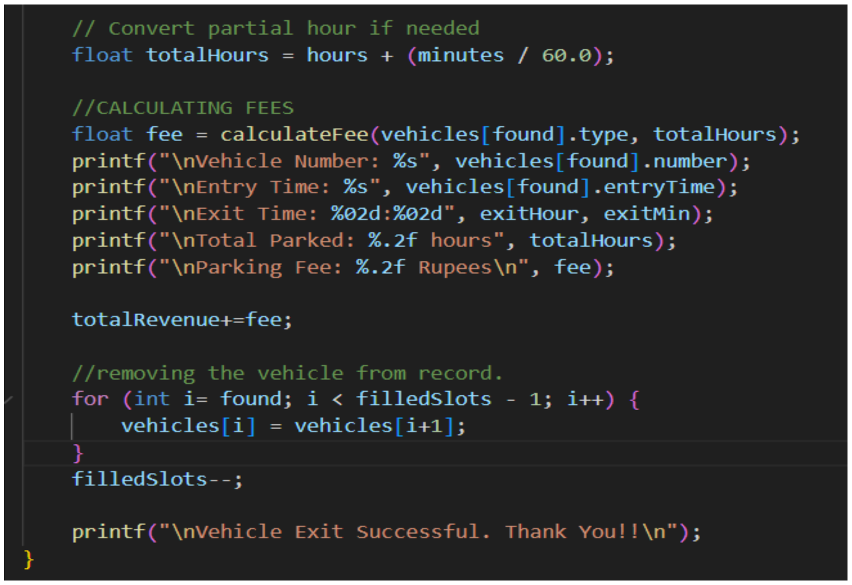


```
void VehicleEntry() {
    if (filledSlots>= totalSlots) {
        printf("\nSORRY! PARKING IS FULL...\n");
        return;
    }

    struct Vehicle v;
    printf("\nEnter Vehicle Number: ");
    scanf("%s",v.number);
    printf("Enter Vehicle Type(Car/Bike/Truck): ");
    scanf("%s",v.type);
    //entering the time of entry (using my laptop's time.)
    time_t t = time(NULL);
    struct tm *tm = localtime(&t);
    sprintf(v.entryTime, "%02d:%02d", tm->tm_hour, tm->tm_min);

    vehicles[filledSlots++] = v;  //adding filledslots by one after every parking
    printf("\nVEHICLE PARKED SUCCESSFULLY at %s\n",v.entryTime);
}
```

5.3 Code Snippet: Removing a Vehicle

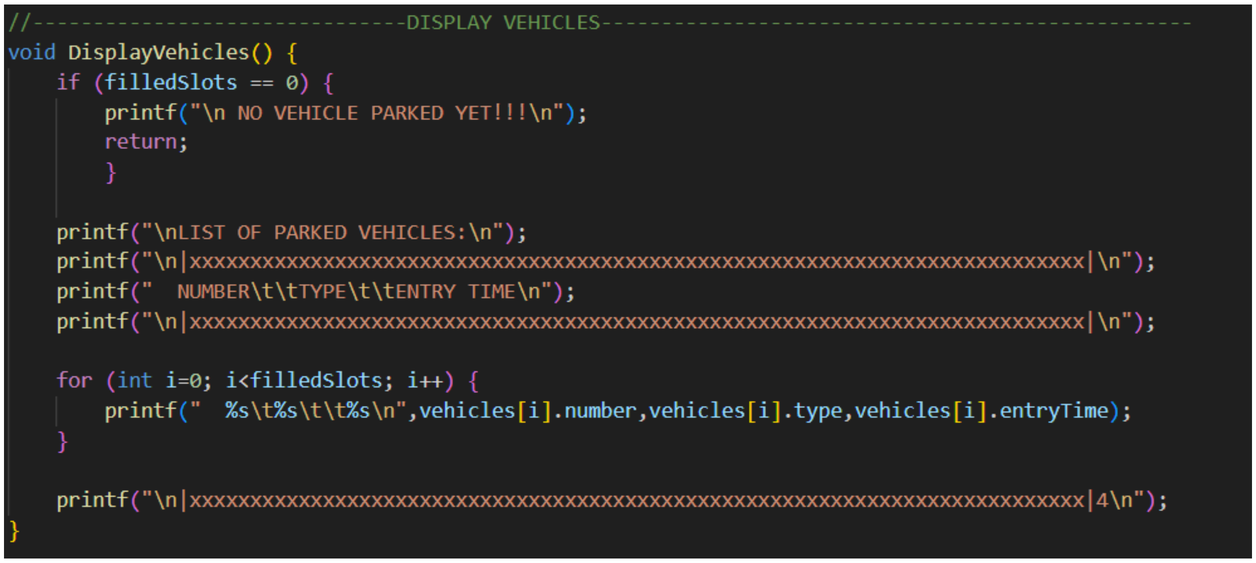


```c
void VehicleExit() {
    char num[20];
    printf("\nEnter Vehicle Number to Exit: ");
    scanf("%s", num);

    int found = -1;
    for (int i = 0; i < filledSlots; i++) {
        if (strcmp(vehicles[i].number, num) == 0) {
            found = i;
            break;
        }
    }

    if (found == -1) {
        printf("\nVehicle not found!\n");
        return;
    }
    //ENTERING EXIT TIME AND AND STORING AS VARIABLE NOW.
    time_t now = time(NULL);
    struct tm *tm_now = localtime(&now);
    int exitHour = tm_now->tm_hour;
    int exitMin = tm_now->tm_min;

    int entryHour, entryMin;  //ENTRY TIME
    sscanf(vehicles[found].entryTime, "%d:%d", &entryHour, &entryMin);

    int hours = exitHour - entryHour;  //CALCULATING TIME OF PARKING
    int minutes = exitMin - entryMin;

    if (minutes < 0) {            //(BASIC BORROWING OF SUBTRACTION)
        minutes += 60;
        hours -= 1;
    }
    if (hours < 0) {
        hours += 24; // handle overnight parking
    }
```



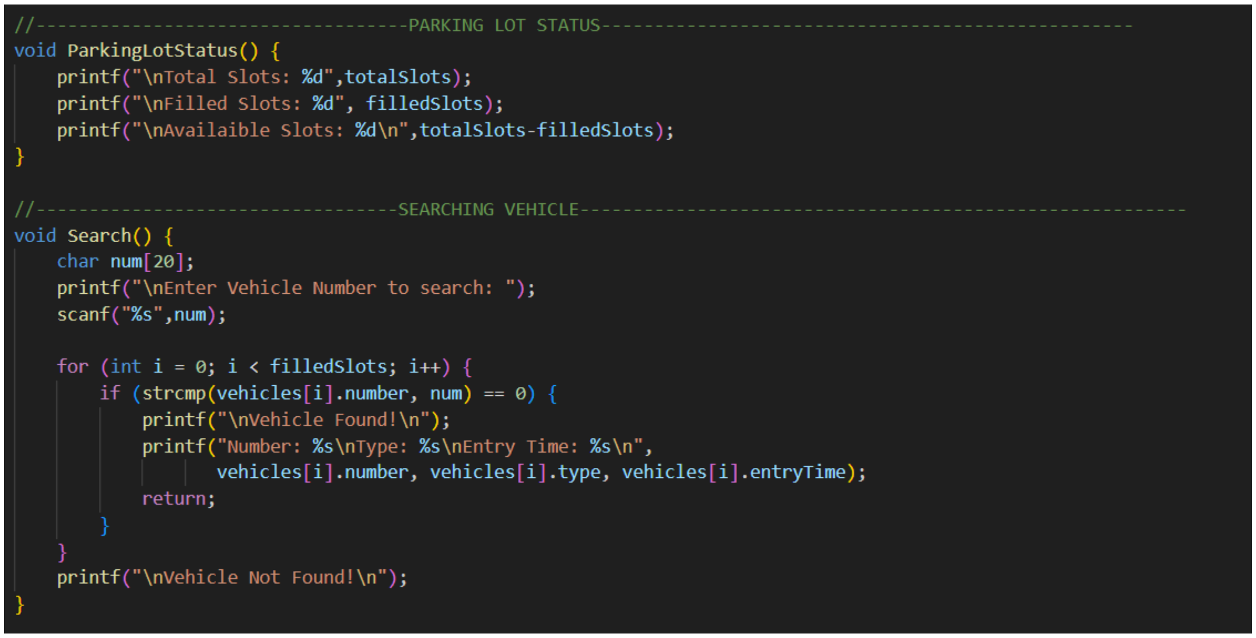
```c
    // Convert partial hour if needed
    float totalHours = hours + (minutes / 60.0);

    //CALCULATING FEES
    float fee = calculateFee(vehicles[found].type, totalHours);
    printf("\nVehicle Number: %s", vehicles[found].number);
    printf("\nEntry Time: %s", vehicles[found].entryTime);
    printf("\nExit Time: %02d:%02d", exitHour, exitMin);
    printf("\nTotal Parked: %.2f hours", totalHours);
    printf("\nParking Fee: %.2f Rupees\n", fee);

    totalRevenue+=fee;

    //removing the vehicle from record.
    for (int i= found; i < filledSlots - 1; i++) {
        vehicles[i] = vehicles[i+1];
    }
    filledSlots--;

    printf("\nVehicle Exit Successful. Thank You!!\n");
}
```

5.4 Code snippet: Display Vehicles



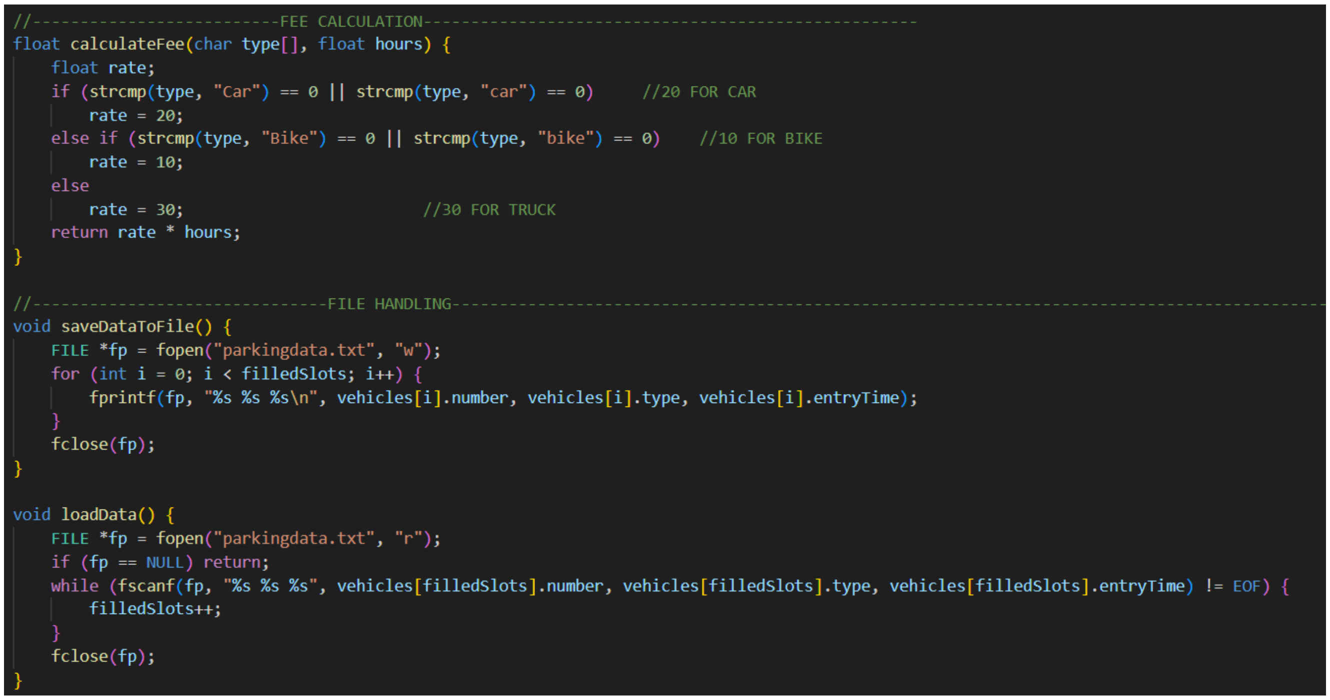
```c
//------------------------------DISPLAY VEHICLES----------------------------------------------
void DisplayVehicles() {
    if (filledSlots == 0) {
        printf("\n NO VEHICLE PARKED YET!!!\n");
        return;
        }

    printf("\nLIST OF PARKED VEHICLES:\n");
    printf("\n|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|\n");
    printf("  NUMBER\t\tTYPE\t\tENTRY TIME\n");
    printf("\n|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|\n");

    for (int i=0; i<filledSlots; i++) {
        printf("  %s\t%s\t\t%s\n",vehicles[i].number,vehicles[i].type,vehicles[i].entryTime);
    }

    printf("\n|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|4\n");
}
```

5.5 Code Snippet: Parking lot status and Searching Vehicle



```c
//---------------------------------PARKING LOT STATUS-------------------------------------------------
void ParkingLotStatus() {
    printf("\nTotal Slots: %d",totalSlots);
    printf("\nFilled Slots: %d", filledSlots);
    printf("\nAvailaible Slots: %d\n",totalSlots-filledSlots);
}

//-------------------------------SEARCHING VEHICLE-------------------------------------------------------
void Search() {
    char num[20];
    printf("\nEnter Vehicle Number to search: ");
    scanf("%s",num);

    for (int i = 0; i < filledSlots; i++) {
        if (strcmp(vehicles[i].number, num) == 0) {
            printf("\nVehicle Found!\n");
            printf("Number: %s\nType: %s\nEntry Time: %s\n",
                   vehicles[i].number, vehicles[i].type, vehicles[i].entryTime);
            return;
        }
    }
    printf("\nVehicle Not Found!\n");
}
```

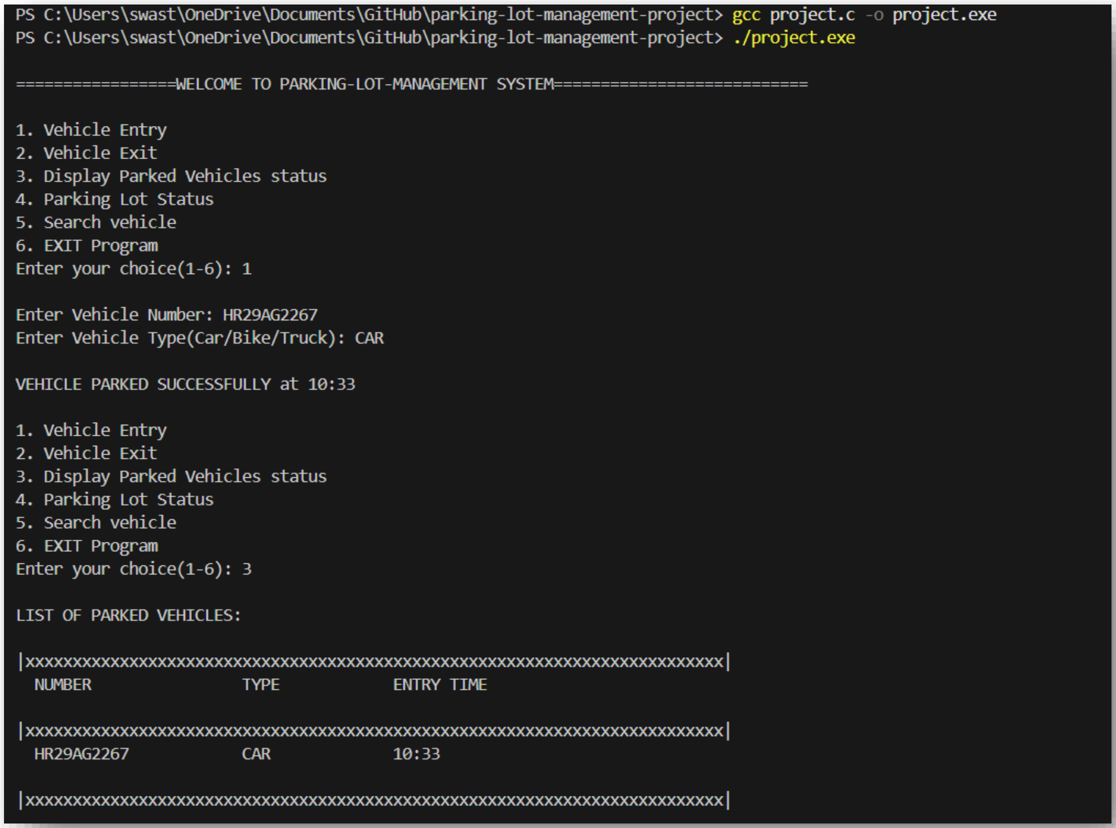
5.6 Code Snippet: Fee calculation and file handling



```
//--------------------------FEE CALCULATION----------------------------------------------------
float calculateFee(char type[], float hours) {
    float rate;
    if (strcmp(type, "Car") == 0 || strcmp(type, "car") == 0)     //20 FOR CAR
        rate = 20;
    else if (strcmp(type, "Bike") == 0 || strcmp(type, "bike") == 0)    //10 FOR BIKE
        rate = 10;
    else
        rate = 30;                              //30 FOR TRUCK
    return rate * hours;
}

//------------------------------FILE HANDLING-------------------------------------------------------------------------------------
void saveDataToFile() {
    FILE *fp = fopen("parkingdata.txt", "w");
    for (int i = 0; i < filledSlots; i++) {
        fprintf(fp, "%s %s %s\n", vehicles[i].number, vehicles[i].type, vehicles[i].entryTime);
    }
    fclose(fp);
}

void loadData() {
    FILE *fp = fopen("parkingdata.txt", "r");
    if (fp == NULL) return;
    while (fscanf(fp, "%s %s %s", vehicles[filledSlots].number, vehicles[filledSlots].type, vehicles[filledSlots].entryTime) != EOF) {
        filledSlots++;
    }
    fclose(fp);
}
```

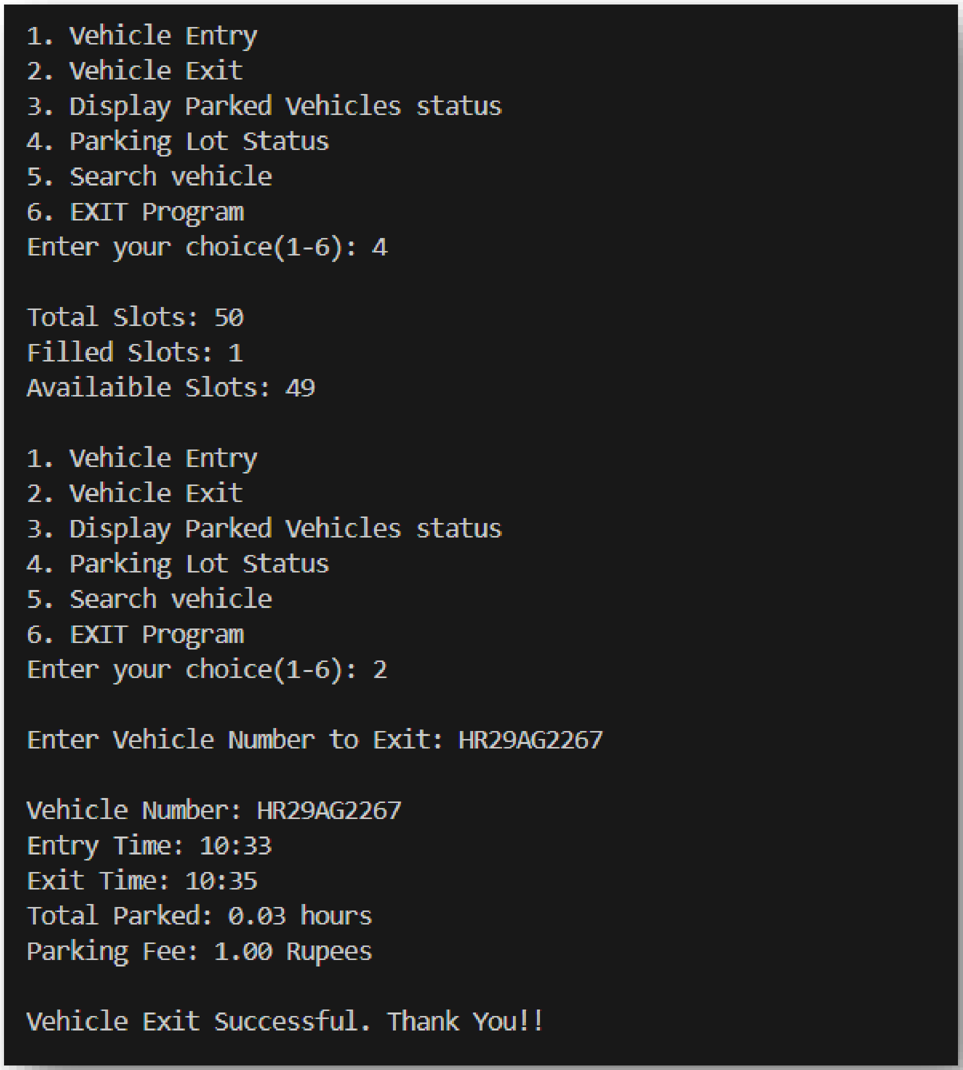
6. TESTING AND RESULTS:

Test case 1:



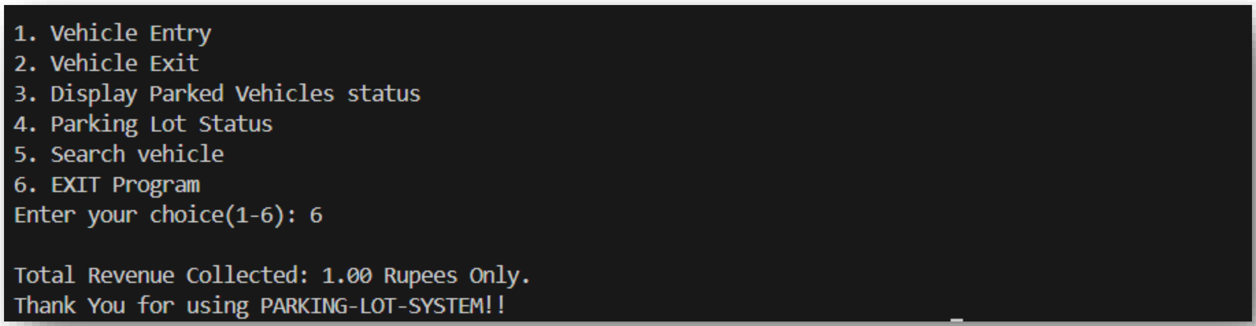
```
PS C:\Users\swast\OneDrive\Documents\GitHub\parking-lot-management-project> gcc project.c -o project.exe
PS C:\Users\swast\OneDrive\Documents\GitHub\parking-lot-management-project> ./project.exe

=================WELCOME TO PARKING-LOT-MANAGEMENT SYSTEM===========================

1. Vehicle Entry
2. Vehicle Exit
3. Display Parked Vehicles status
4. Parking Lot Status
5. Search vehicle
6. EXIT Program
Enter your choice(1-6): 1

Enter Vehicle Number: HR29AG2267
Enter Vehicle Type(Car/Bike/Truck): CAR

VEHICLE PARKED SUCCESSFULLY at 10:33

1. Vehicle Entry
2. Vehicle Exit
3. Display Parked Vehicles status
4. Parking Lot Status
5. Search vehicle
6. EXIT Program
Enter your choice(1-6): 3

LIST OF PARKED VEHICLES:

|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
  NUMBER               TYPE            ENTRY TIME

|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
  HR29AG2267           CAR             10:33

|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
```

Test case 2:



```
1. Vehicle Entry
2. Vehicle Exit
3. Display Parked Vehicles status
4. Parking Lot Status
5. Search vehicle
6. EXIT Program
Enter your choice(1-6): 4

Total Slots: 50
Filled Slots: 1
Availaible Slots: 49

1. Vehicle Entry
2. Vehicle Exit
3. Display Parked Vehicles status
4. Parking Lot Status
5. Search vehicle
6. EXIT Program
Enter your choice(1-6): 2

Enter Vehicle Number to Exit: HR29AG2267

Vehicle Number: HR29AG2267
Entry Time: 10:33
Exit Time: 10:35
Total Parked: 0.03 hours
Parking Fee: 1.00 Rupees

Vehicle Exit Successful. Thank You!!
```

Test case 3:



```
1. Vehicle Entry
2. Vehicle Exit
3. Display Parked Vehicles status
4. Parking Lot Status
5. Search vehicle
6. EXIT Program
Enter your choice(1-6): 6

Total Revenue Collected: 1.00 Rupees Only.
Thank You for using PARKING-LOT-SYSTEM!!
```

Overall Result:

The system compiles successfully, handles inputs gracefully, and meets all requirements.

7. CONCLUSION AND FUTURE WORK:

7.1 CONCLUSION:

Automated management of parking spaces enables efficient handling of vehicles by integrating technology into everyday parking operations. Such a system continuously monitors the availability of parking spots, keeping accurate and up-to-date records of occupied and vacant spaces. By collecting real-time data related to vehicle entries, exits, and parking durations, the system provides valuable insights into overall occupancy trends and usage patterns. This information helps in analyzing peak hours, optimizing space allocation, and determining daily or monthly earnings with improved accuracy.

The project is built using fundamental principles of C programming, making it both educational and practical. Core concepts such as structures are used to store vehicle-related data, while arrays help organize multiple records efficiently. Loops and conditionals manage repetitive tasks and decision-making processes, ensuring smooth operation of each module. Functions further enhance modularity and readability by breaking down the system into smaller, manageable components. Together, these programming techniques allow the system to operate reliably, automate manual tasks, reduce human errors, and present an organized solution for parking lot administration.

7.2 FUTURE WORK:

- ❖ We can add new slots for ev charging.
- ❖ We can add gui or web based interface.
- ❖ We can also add slot based parking instead of sequencial storage(array).

8. REFERENCES:

- ❖ Online C Language Resources (GeeksForGeeks, TutorialsPoint)
- ❖ The C Programming Language by Brian W. Kernighan and Dennis M. Ritchie