PARKING LOT MANAGEMENT SYSTEM

Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

By

N.Vennela- 24KB1A05CA

N.Yasaswi- 24KB1A05CB

N.Haswitha- 24KB1AO5CC

O.Bhavana - 24KB1A05CJ

Under the Guidance of

SMT.B.Sruthi,Assistant Professer

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
N.B.K.R.I.S.T

NBKR Institute of Science and Technology

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Chapter 1

Chapter 2

CERTIFICATE

This is to certify that the project report entitled PARKING LOT MANAGEMENT SYSTEM being submitted by

N. Vennela (24KBA05CA),

N. Yasaawi (24KBA05CB),

N. Haswitha (24KBA05CC),

O. Bhavana (24KBA05CJ)

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Scie nce and Engineering to the Jawaharlal Nehru Technological University, Anantapur is a record of bonafide work carried out under my guidance and supervision.

B. Sruthi
Assistant Professor

A. Raja Sekhar Reddy
M.Tech,Ph.D
Head of the Department

DECLARATION

I hereby declare that the dissertation entitled PARKING LOT MANAGEMENT SYSTEM submitted for the B.TechDegree is my original work and the dissertation has not formed the basis for theaward of any degree, associateship, fellowship or any other similar titles.

Place: Vidyanagar
Date:

N. Vennela (24KBA05CA)
N. Yasaawi (24KBA05CB)
N. Haswitha (24KBA05CC)
O. Bhavana (24KBA05CJ)

ACKNOWLEDGEMENT

We sincerely thank our faculty guide, Ms. B. Sruthi, for her guidance and support throughout our project, "Parking Lot Management System." We also thank Dr. A. Raja Sekhar Reddy, Head of Department, for providing the necessary resources and encouragement.

Gratitude to our friends and teammates for their cooperation and motivation, and to our families for their constant support.

This project enhanced our knowledge of C Programming, especially in Arrays and Linked Lists, while giving us valuable insight into real-world application development.

Thank you all once again!

Abstract of project

The Parking Lot Management System is a software-based solution designed to streamline and automate the process of managing parking spaces in commercial, residential, or public areas. This system aims to reduce manual effort, prevent congestion, and optimize the use of available parking spaces. It provides real-time tracking of vehicle entries and exits, availability status of parking slots, and user-friendly interfaces for both administrators and users. The system also supports functionalities such as slot reservation, time-based billing, and vehicle categorization. By integrating technologies like sensors (optional), databases, and possibly mobile/web interfaces, this system enhances the overall efficiency and convenience of parking lot operations. The project is developed using modern programming tools and follows a modular design to ensure scalability and maintainability.

INTRODUCTION

In today's fast-paced urban environment, managing vehicle parking efficiently has become a major challenge due to the increasing number of vehicles and limited availability of parking spaces. Conventional manual parking systems often lead to issues such as time consumption, space mismanagement, traffic congestion, and human error. These limitations have highlighted the need for a systematic, automated solution to monitor and manage parking lots effectively.

The Parking Lot Management System is a software-based project designed to automate the management of parking areas using technology. The system aims to provide real-time information about parking slot availability, automate entry and exit tracking, calculate parking duration, and ensure optimal use of parking space. It can be implemented in commercial complexes, shopping malls, educational institutions, hospitals, and other public or private areas where parking space is a critical concern.

This system offers features such as digital slot booking, vehicle entry/exit time tracking, fee calculation, and user-friendly interfaces for both administrators and users. It reduces the need for manual supervision, eliminates paperwork, and enhances the overall parking experience. The system can also be further integrated with IoT-based sensors, mobile apps, or web applications for smart city environments.

Problem statement:

Traditional parking systems are inefficient, error-prone, and do not utilize available parking space effectively. Users waste time searching for parking, and administrators face difficulties managing and monitoring usage manually.

Scope of the Project:

The proposed system aims to automate the parking process from slot allocation to exit. It supports admin-controlled slot management, provides users with real-time updates, and can be extended to support online payment and mobile notifications. The system is scalable and can be customized based on the type and size of the parking facility.

Objectives:

To design and develop a user-friendly application to manage parking slots efficiently.

To reduce the time taken for parking and vehicle retrieval.

To ensure transparency and reliability in parking operations.

To enable easy monitoring and management for administrators.

To improve overall parking convenience for users.

Literature Survey / Existing System

Most small-scale parking systems use manual registers or token systems. These are time-consuming, prone to errors, and offer no real-time tracking.

Limitations of Existing Systems:

Manual entry can lead to inaccurate records. There is no immediate view of available slots, and the process is time-consuming for both users and administrators.
A command-line based system written in C that allows the user to interact with the parking lot through a menu. It handles a fixed number of slots using a structure array and offers clear vehicle-slot mapping.

Software Requirement Analysis

Functional Requirements:

The system should be able to initialize parking slots, park a vehicle by entering a license plate and choosing a slot, remove a vehicle from a slot, and display all available slots.

Non-Functional Requirements:

It should be a platform-independent C program with minimal memory usage and capable of running in any standard C compiler such as Turbo C or GCC.

Software Requirements:

The software should run on Windows or Linux operating systems and be compatible with compilers like GCC, Turbo C, Code::Blocks, or Dev C++.

Software Design

Data Structure Design:

The system uses an array of structures defined as follows:

```c
typedef struct {
   char license_plate[15];
   int slot_number;
   int is_occupied;
} ParkingSlot;
```

Program Flow:

The main loop runs a menu repeatedly until the user chooses to exit. Based on the user's choice, appropriate functions are called, and vehicle and slot information is updated in memory.

Functions Used:

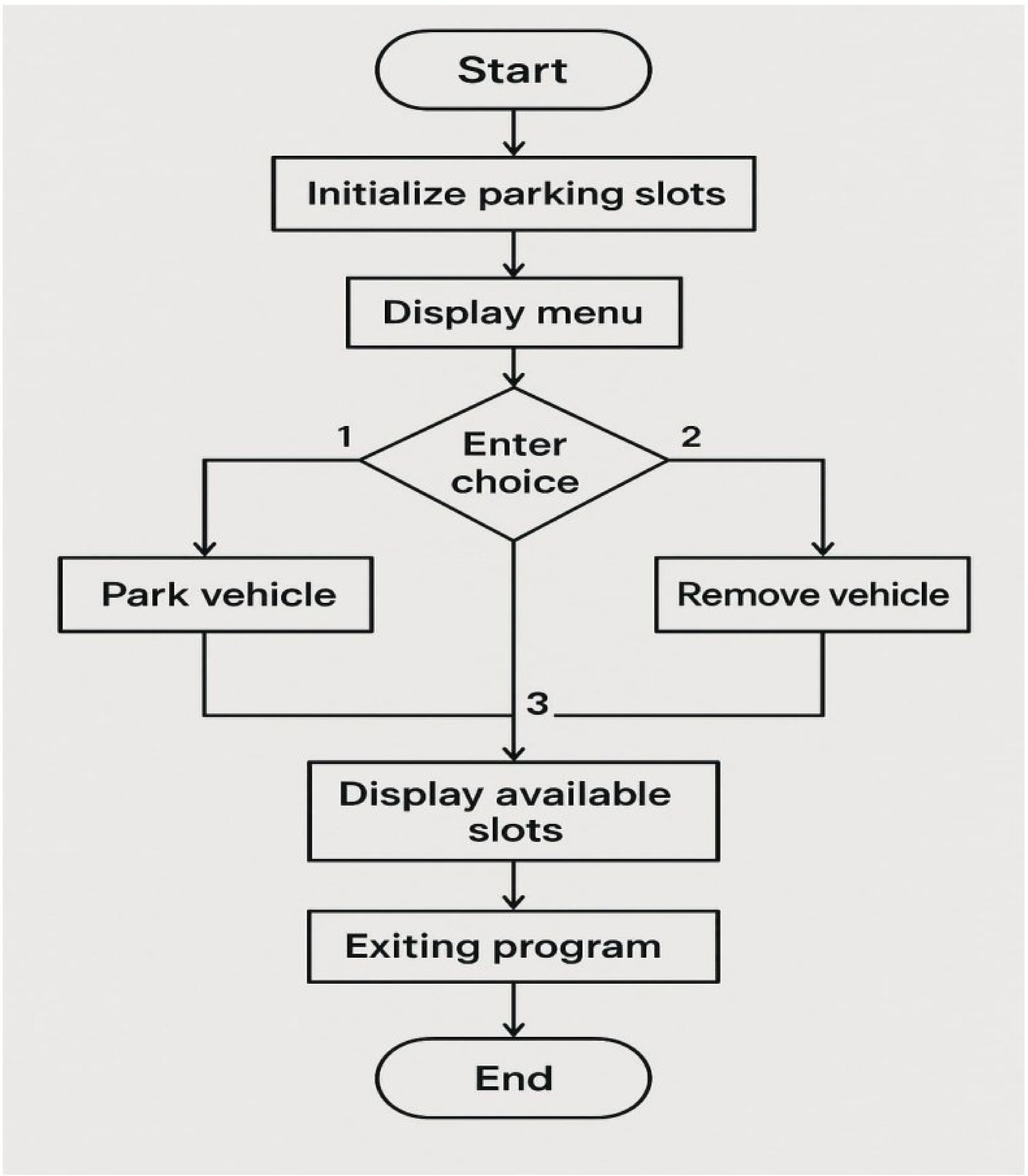
initialize_slots() – Initializes all slots as empty.

park_vehicle() – Allows parking by entering vehicle details.

remove_vehicle() – Empties a slot when the vehicle exits.

display_available_slots() – Lists all unoccupied slots.

Control Flow Diagram:



Proposed System:

The proposed system offers a streamlined approach to parking management using a simple terminal-based interface. It provides real-time updates on slot status, preventing duplicate slot occupancy. Its intuitive design ensures easy usability, while the modular code structure allows for straightforward maintenance and scalability.

Coding:

```
#include <stdio.h>
#include <stdlib.h>
 #include <string.h>

#define MAX_SLOTS 10

typedef struct {
char license_plate[15];
int slot_number;
int is_occupied;
} ParkingSlot;

ParkingSlotparking_slots[MAX_SLOTS];

void initialize_slots()
{
for (int i = 0; i < MAX_SLOTS; i++)
 {
parking_slots[i].slot_number = i + 1;
parking_slots[i].is_occupied = 0;
strcpy(parking_slots[i].license_plate,"");
}
 }
```

```
void display_menu()
 {
printf("\nParking Lot Management System\n");
printf("1. Park Vehicle\n");
printf("2. Remove Vehicle\n");
printf("3. Display Available Slots\n");
printf("4. Exit\n");
 printf("Enter your choice: ");
}
void park_vehicle() {
int slot_number;
char license_plate[15];
printf("Enter slot number to park(1-%d): ",
MAX_SLOTS);
scanf("%d", &slot_number);

if (slot_number < 1 || slot_number >MAX_SLOTS) {
    printf("Invalid slot number.\n");
    return;
}

if (parking_slots[slot_number - 1].is_occupied) {
    printf("Slot is already occupied.\n");
    return;
}
printf("Enter vehicle license plate: ");
scanf("%s", license_plate);

parking_slots[slot_number-1].is_occupied = 1;
strcpy(parking_slots[slot_number - 1].license_plate,
license_plate);

printf("Vehicle parked successfully in slot %d.\n",
slot_number);
}
```

```
void remove_vehicle() {
int slot_number;
printf("Enter slot number to remove vehicle (1-%d): ",
MAX_SLOTS);
scanf("%d", &slot_number);

if (slot_number < 1 || slot_number>MAX_SLOTS) {
   printf("Invalid slot number.\n");
   return;
}

if (!parking_slots[slot_number - 1].is_occupied) {
   printf("Slot is already empty.\n");
   return;
}
parking_slots[slot_number - 1].is_occupied = 0;
strcpy(parking_slots[slot_number-1].license_plate, "");

printf("Vehicle removed from slot %d successfully.\n",
slot_number);

}
void display_available_slots()
{
printf("\nAvailable Parking Slots:\n"); for (int i = 0; i <
MAX_SLOTS; i++)
{
if (!parking_slots[i].is_occupied)
 {
printf("Slot %d: Available\n", i + 1);
}
}
 }
```

```
int main()
 {
 int choice; initialize_slots();
do {
    display_menu();
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            park_vehicle();
            break;
        case 2:
            remove_vehicle();
            break;
        case 3:
            display_available_slots();
            break;
        case 4:
            printf("Exiting program.\n");
            break;
        default:
 printf("Invalid choice. Please try again.\n");
    }
    }while (choice != 4);

    return 0;
    }
```

Testing:

Test Case 1: Park a vehicle

Input:
1 → 2 → KA01AB1234 → 4
Expected Output:
Vehicle parked successfully in slot 2.

Test Case 2: Park in an occupied slot

Input:
1 → 2 → KA02XY5678 → 4
Expected Output:
Slot is already occupied.

Test Case 3: Remove a vehicle

Input:
2 → 2 → 4
Expected Output:
Vehicle removed from slot 2 successfully.

Test Case 4: Remove from empty slot

Input:
2 → 5 → 4
Expected Output:
Slot is already empty.

OUTPUT :

```
Sample Output: Parking Lot Management System

Parking Lot Management System
1. Park Vehicle
2. Remove Vehicle
3. Display Available Slots
4. Exit
Enter your choice: 1
Enter slot number to park (1-10): 3
Enter vehicle license plate: AB123CD
Vehicle parked successfully in slot 3.

Parking Lot Management System
1. Park Vehicle
2. Remove Vehicle
3. Display Available Slots
4. Exit
Enter your choice: 1
Enter slot number to park (1-10): 3
Slot is already occupied.

Parking Lot Management System
1. Park Vehicle
2. Remove Vehicle
3. Display Available Slots
4. Exit
Enter your choice: 3

Available Parking Slots:
Slot 1: Available
Slot 2: Available
Slot 4: Available
Slot 5: Available
Slot 6: Available
Slot 7: Available
Slot 8: Available
Slot 9: Available
Slot 10: Available

Parking Lot Management System
1. Park Vehicle
2. Remove Vehicle
3. Display Available Slots
4. Exit
Enter your choice: 2
Enter slot number to remove vehicle (1-10): 3
Vehicle removed from slot 3 successfully.

Parking Lot Management System
1. Park Vehicle
2. Remove Vehicle
3. Display Available Slots
4. Exit
Enter your choice: 4
Exiting program.
```

CONCLUSION:

This project demonstrates the fundamentals of system-level programming and memory handling using C. The Parking Lot Management System fulfills the goal of efficient slot tracking and user interaction via a command-line interface.

Future Enhancements:

Timer-based parking duration tracking.

Dynamic fee calculation.

File I/O for storing vehicle records persistently.

GUI-based front-end for better interaction.

Integration with sensors or microcontrollers for hardware implementation.