Car Rental Management System

JAYESH OM SAHARAVAT

SAP ID - 590023634

December 2, 2025

Abstract

The Car Rental Management System is a console-based application written in C that helps maintain and manage car rental records efficiently. The system allows users to add new cars, display available cars, rent cars by recording the rental date, and return cars while generating an automated bill. The program is designed with modular programming concepts and structured data handling using arrays and struct. This project demonstrates understanding of C fundamentals such as structures, functions, file handling (optional), and time-based operations using the <time.h> library.

TABLE OF CONTENTS

INTRODUCTION

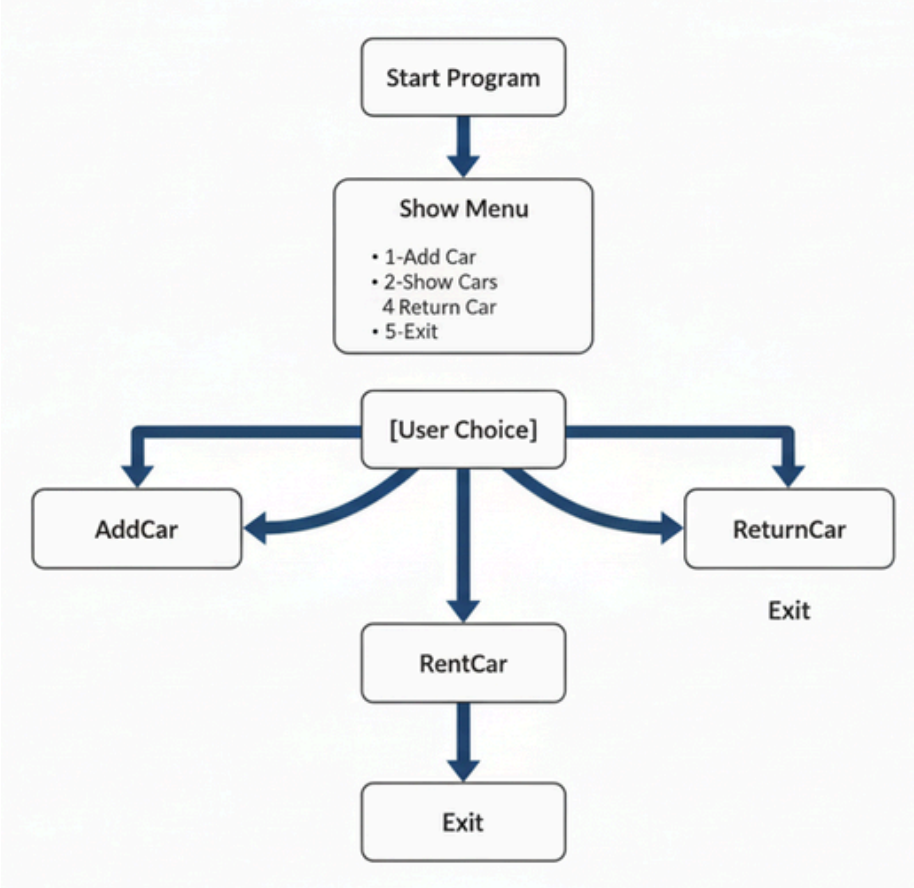
Car rental businesses require efficient management systems to track vehicle inventory, record rental transactions, and generate accurate bills. Traditional manual methods are prone to errors and inefficiency. Computer-based solutions provide automated, reliable, and scalable approaches to manage such operations.

Project Overview

The Car Rental Management System is a console-based C application that provides automated management of car rental operations. It demonstrates practical implementation of fundamental programming concepts including:

- Data structure design (structures and arrays)
- User interface development (menu-driven system)
- Input validation and error handling
- Timestamp-based date/time operations
- Dynamic billing calculations
- Core business logic implementation

3.2 Flowchart



4.1 Implementation Details

1. Data Structure

```
typedef struct {
    int id;
    char name[50];
    int rentPerDay;
    char status[20];
    time_t rentDate;
    int daysRented;
} Car;
```

2. Adding a New Car

```
void addCar(Car cars[], int *count) {
    printf("Enter Car ID: ");
    scanf("%d", &cars[*count].id);
    printf("Enter Car Name: ");
    scanf(" %[^\n]", cars[*count].name);
    printf("Enter Rent per Day (Rs): ");
    scanf("%d", &cars[*count].rentPerDay);
    strcpy(cars[*count].status, "Available");
    (*count)++;
}
```

3. Renting a Car

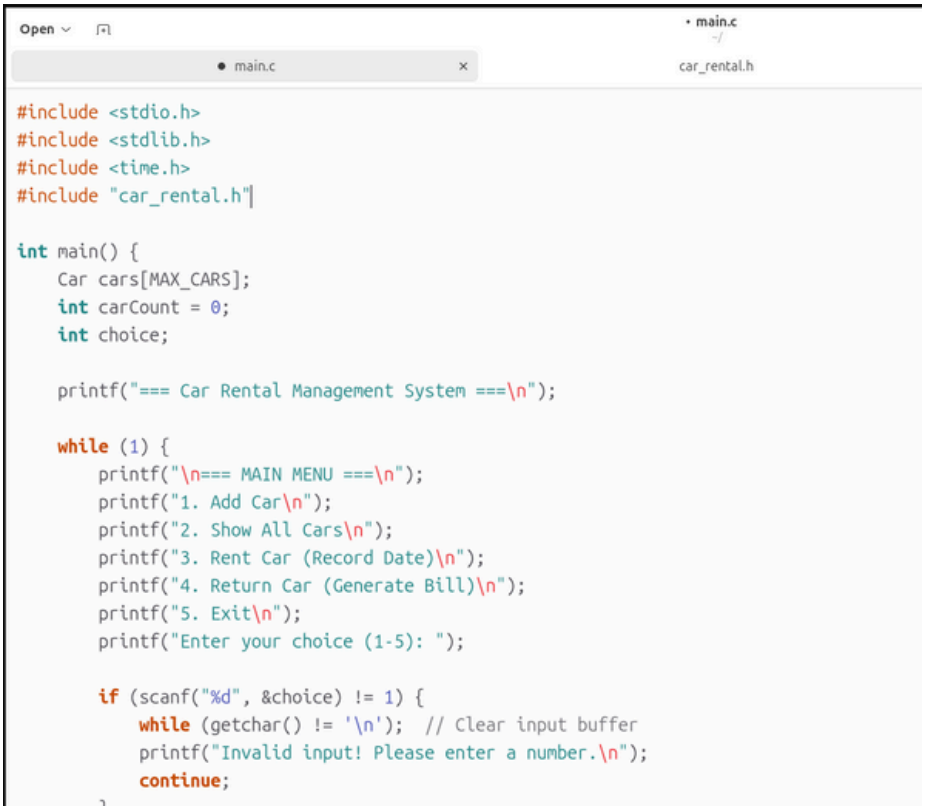
```
void rentCar(Car cars[], int count) {
    int id;
    printf("Enter car ID to rent: ");
    scanf("%d", &id);
    for (int i = 0; i < count; i++) {
        if (cars[i].id == id && strcmp(cars[i].status, "Available") == 0) {
            strcpy(cars[i].status, "Rented");
            cars[i].rentDate = time(NULL);
            printf("Car rented successfully on %s", ctime(&cars[i].rentDate));
        }
    }
}
```

4. Returning a Car

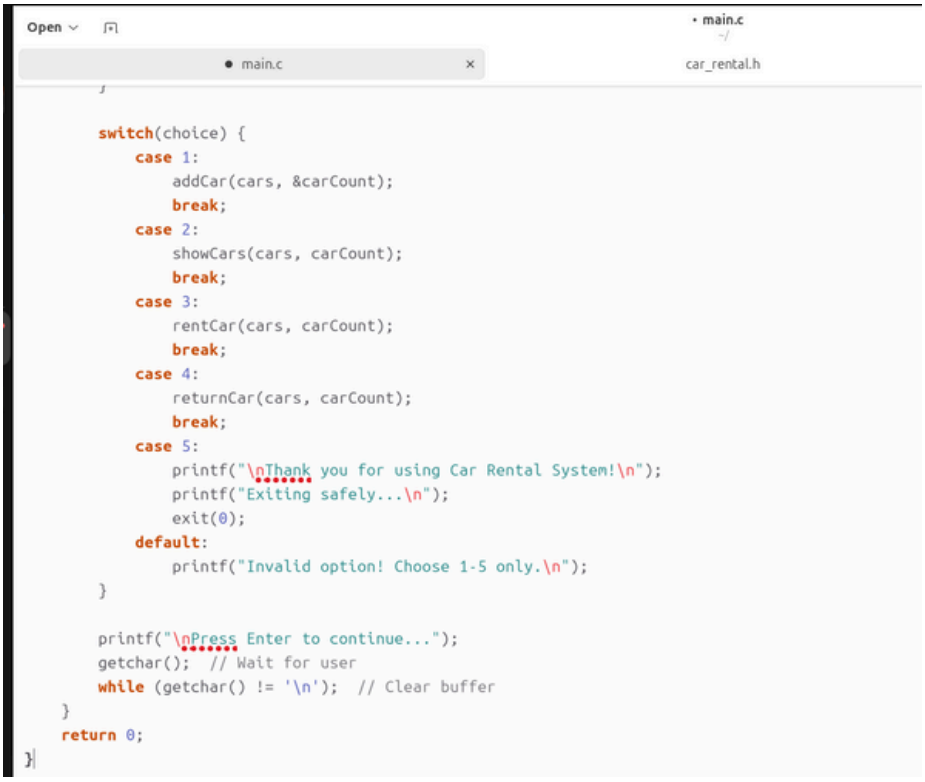
```
void returnCar(Car cars[], int count) {
    int id;
    printf("Enter car ID to return: ");
    scanf("%d", &id);
    for (int i = 0; i < count; i++) {
        if (cars[i].id == id && strcmp(cars[i].status, "Rented") == 0) {
            time_t now = time(NULL);
            double days = difftime(now, cars[i].rentDate)/(60*60*24);
            int cost = (int)(days + 0.5) * cars[i].rentPerDay;
            printf("Total Bill: Rs. %d\n", cost);
            strcpy(cars[i].status, "Available");
        }
    }
}
```

5.Source Code

main.c



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "car_rental.h"

int main() {
    Car cars[MAX_CARS];
    int carCount = 0;
    int choice;

    printf("=== Car Rental Management System ===\n");

    while (1) {
        printf("\n=== MAIN MENU ===\n");
        printf("1. Add Car\n");
        printf("2. Show All Cars\n");
        printf("3. Rent Car (Record Date)\n");
        printf("4. Return Car (Generate Bill)\n");
        printf("5. Exit\n");
        printf("Enter your choice (1-5): ");

        if (scanf("%d", &choice) != 1) {
            while (getchar() != '\n');  // Clear input buffer
            printf("Invalid input! Please enter a number.\n");
            continue;
```



```c
        switch(choice) {
            case 1:
                addCar(cars, &carCount);
                break;
            case 2:
                showCars(cars, carCount);
                break;
            case 3:
                rentCar(cars, carCount);
                break;
            case 4:
                returnCar(cars, carCount);
                break;
            case 5:
                printf("\nThank you for using Car Rental System!\n");
                printf("Exiting safely...\n");
                exit(0);
            default:
                printf("Invalid option! Choose 1-5 only.\n");
        }

        printf("\nPress Enter to continue...");
        getchar();  // Wait for user
        while (getchar() != '\n');  // Clear buffer
    }
    return 0;
}
```

car_operation.c

Open car_operation.c ~/

main.c | car_rental.h

```c
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "car_rental.h"

time_t getCurrentTime() {
    return time(NULL);
}

void addCar(Car cars[], int *count) {
    if (*count >= MAX_CARS) {
        printf("Maximum cars limit reached!\n");
        return;
    }

    Car *newCar = &cars[*count];
    printf("Enter Car ID: ");
    scanf("%d", &newCar->id);
    printf("Enter Car Name: ");
    scanf(" %[^\n]", newCar->name);  // Read string with spaces
    printf("Enter Rent per Day (Rs): ");
    scanf("%d", &newCar->rentPerDay);

    strcpy(newCar->status, "Available");
    newCar->rentDate = 0;
    newCar->daysRented = 0;

    (*count)++;
    printf("Car added successfully!\n");
```

Open car_operation.c ~/

main.c | car_rental.h

```c
    printf("Car added successfully!\n");
}

void showCars(Car cars[], int count) {
    if (count == 0) {
        printf("No cars available!\n");
        return;
    }

    printf("\n=== CARS LIST ===\n");
    printf("ID | Name           | Rent/Day | Status    | Days Rented\n");
    printf("---|----------------|----------|-----------|------------\n");

    for (int i = 0; i < count; i++) {
        printf("%-2d| %-14s | %-8d | %-9s | %-10d\n",
               cars[i].id, cars[i].name, cars[i].rentPerDay,
               cars[i].status, cars[i].daysRented);
    }
}

void rentCar(Car cars[], int count) {
    if (count == 0) {
        printf("No cars available to rent!\n");
        return;
    }

    int id;
    printf("Enter car ID to rent: ");
    if (scanf("%d", &id) != 1) {
        printf("Invalid ID!\n");
```


Open
• car_operation.c
~/
main.c
car_rental.h
printf("Invalid ID!\n");
return;
}
for (int i = 0; i < count; i++) {
if (cars[i].id == id && strcmp(cars[i].status, "Available") == 0) {
strcpy(cars[i].status, "Rented");
cars[i].rentDate = getCurrentTime();
cars[i].daysRented = 0;
printf("Car rented successfully on %s", ctime(&cars[i].rentDate));
return;
}
}
printf("Car ID %d not found or not available!\n", id);
}S
void returnCar(Car cars[], int count) {
if (count == 0) {
printf("No cars to return!\n");
return;
}
int id;
printf("Enter car ID to return: ");
if (scanf("%d", &id) != 1) {
printf("Invalid ID!\n");
return;
}
for (int i = 0; i < count; i++) {


Open
• car_operation.c
~/
main.c
car_rental.h
for (int i = 0; i < count; i++) {
if (cars[i].id == id && strcmp(cars[i].status, "Rented") == 0) {
time_t currentTime = getCurrentTime();
double daysDiff = difftime(currentTime, cars[i].rentDate) / (60*60*24);
cars[i].daysRented = (int)(daysDiff + 0.5); // Round to nearest day
int totalCost = cars[i].rentPerDay * cars[i].daysRented;
int lateFee = (daysDiff > 7) ? LATE_FEE : 0;
if (lateFee > 0) totalCost += lateFee;
printf("\n=== BILL RECEIPT ===\n");
printf("Car: %s (ID: %d)\n", cars[i].name, cars[i].id);
printf("Rented: %s", ctime(&cars[i].rentDate));
printf("Returned: %s", ctime(&currentTime));
printf("Days Rented: %d\n", cars[i].daysRented);
printf("Rent per Day: Rs. %d\n", cars[i].rentPerDay);
printf("Late Fee (>7 days): Rs. %d\n", lateFee);
printf("TOTAL BILL: Rs. %d\n", totalCost);
printf("====================\n");
strcpy(cars[i].status, "Available");
printf("Car returned successfully!\n");
return;
}
}
printf("Car ID %d not found or not rented!\n", id);
}
int id;

car_rental.h

Open

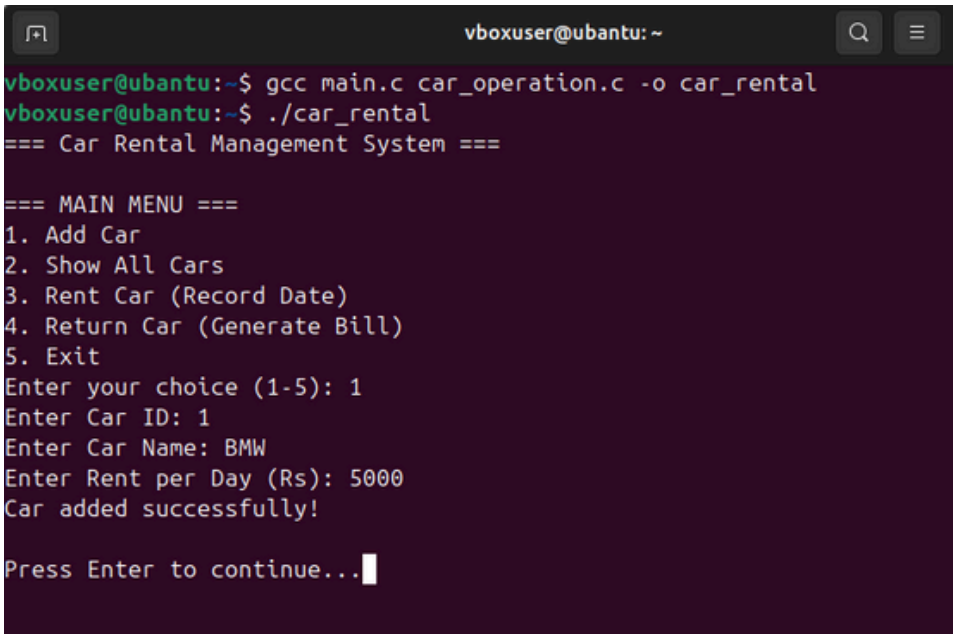
car_rental.h
~/

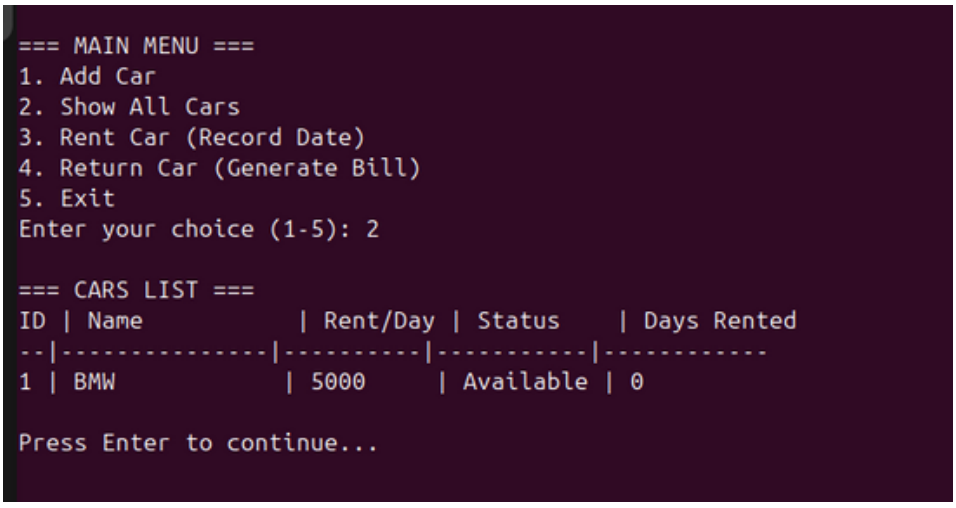
• main.c

car_rental.h

```c
#ifndef CAR_RENTAL_H
#define CAR_RENTAL_H

#include <time.h>

#define MAX_CARS 100
#define LATE_FEE 50

typedef struct {
    int id;
    char name[50];
    int rentPerDay;
    char status[20];
    time_t rentDate;      // Timestamp when rented
    int daysRented;       // Days rented (calculated on return)
} Car;

void addCar(Car cars[], int *count);
void showCars(Car cars[], int count);
void rentCar(Car cars[], int count);
void returnCar(Car cars[], int count);
time_t getCurrentTime();

#endif
```

6.Output


vboxuser@ubantu: ~
vboxuser@ubantu:~$ gcc main.c car_operation.c -o car_rental
vboxuser@ubantu:~$ ./car_rental
=== Car Rental Management System ===
=== MAIN MENU ===
1. Add Car
2. Show All Cars
3. Rent Car (Record Date)
4. Return Car (Generate Bill)
5. Exit
Enter your choice (1-5): 1
Enter Car ID: 1
Enter Car Name: BMW
Enter Rent per Day (Rs): 5000
Car added successfully!
Press Enter to continue...


=== MAIN MENU ===
1. Add Car
2. Show All Cars
3. Rent Car (Record Date)
4. Return Car (Generate Bill)
5. Exit
Enter your choice (1-5): 2
=== CARS LIST ===
ID	Name	Rent/Day	Status	Days Rented
1	BMW	5000	Available	0
Press Enter to continue...

```
=== MAIN MENU ===
1. Add Car
2. Show All Cars
3. Rent Car (Record Date)
4. Return Car (Generate Bill)
5. Exit
Enter your choice (1-5): 3
Enter car ID to rent: 1
Car rented successfully on Tue Dec  2 17:40:49 2025

Press Enter to continue...
```

```
=== MAIN MENU ===
1. Add Car
2. Show All Cars
3. Rent Car (Record Date)
4. Return Car (Generate Bill)
5. Exit
Enter your choice (1-5): 4
Enter car ID to return: 1

=== BILL RECEIPT ===
Car: BMW (ID: 1)
Rented: Tue Dec  2 17:40:49 2025
Returned: Tue Dec  2 17:41:18 2025
Days Rented: 0
Rent per Day: Rs. 5000
Late Fee (>7 days): Rs. 0
TOTAL BILL: Rs. 0
====================
Car returned successfully!

Press Enter to continue...
```

```
=== MAIN MENU ===
1. Add Car
2. Show All Cars
3. Rent Car (Record Date)
4. Return Car (Generate Bill)
5. Exit
Enter your choice (1-5): 5

Thank you for using Car Rental System!
Exiting safely...
vboxuser@ubantu:~$
```

7.Conclusion & Future Work

Conclusion

The Car Rental Management System successfully demonstrates how C programming can automate practical business tasks. It improves accuracy and reduces manual workload in small-scale rental operations.

FutureWork

- Add file handling to store data permanently.
- Implement customer records and authentication.
- Provide graphical UI using C graphics or migrate to C++/Python GUI.
- Integrate payment gateways for automated billing.

8.References

- Stack Overflow discussions and GeeksforGeeks for algorithm snippets.
- UPES Programming Lab Notes and Lecture Materials.