Source-code of c

```
//main.c(main program)

#include <stdio.h>

#include <stdlib.h>

#include "functions.h"

#include <time.h>


int main() {

   loadCar();  // call of loadCar function

   int choice;

   do {

      clearScreen();  // call of clearScreen function

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

      // menu of differnt user will be display

      printf("1. Admin\n2. Passenger\n3. Exit\nChoice: ");

      scanf("%d", &choice);

      flushInput(); // To clear the input buffer after scanf


      switch (choice) {

         case 1:

            login();  //calling the login function

            break;

         case 2:

            pass(); //calling the  (passanger) pass function

            break;

         case 3:

            printf("Bye!Have a nice day \n");

            saveCars();    // call of saveCars function
            exit(0);

         default:

            printf("erorrrrrrr\n");

            getchar();

      }

   } while (1);


   return 0;

}
```

```
//function.h(header file)

//macro are define (define at one place and used in different place)

#ifndef FUNCTIONS_H  // if not define(if the header is not define)

#define FUNCTIONS_H  // defining the header file name as FUNCTION_H


//pre-defining the size and easier to modify

#define MAX_CARS 100 //maximum number of cars that can be stored

#define MAX_STR 100  // maximum length of string (name,route,etc)

#define MAX_BOOKINGS 100 //maximum numbers of bookings


 //car file name (holds the the data of cars )in txt formate

#define CAR_FILE "cars.txt"

 //  booking file (holds the data of bookin and other details) in csv formate (comma seprated value)

#define BOOKING_FILE "bookings.csv"


// struture of car containing  --> modal,type,route...etc

typedef struct {

   int id;

   char carModel[MAX_STR];

   char carType[MAX_STR];

   char category[MAX_STR];

   char route[MAX_STR];

   float price;

   int available;

} Car;


// struture of booking containing  --> id,name,carid,dates(from to return,also actual date of return),..etc

typedef struct {

   char bookingID[MAX_STR];
```

```
    char userName[MAX_STR];

    char carID[MAX_STR];

    char bookingDate[MAX_STR];

    char returnDate[MAX_STR];

    int returned;

    char actualReturnDate[MAX_STR]; // for present date car is returning

} Booking;


//structure of date-->storing in the formate of date,month no. and year[dd-mm-yyyy]

typedef struct {

    int day, month, year;

} Date;


// these are utility function help in performing the code effectively

void clearScreen(); // help to clear the screen or help load new screen

void flushInput(); //flushing out the buffer

void getStringInput(const char *prompt, char *buffer, int size);


// function for loading and saving prioe,new and update data to the function

void loadCar();

void saveCars();

void saveNewCar(Car newCar);


// admin section (all the function belongs to or access by admin only )

void login();

void adminMenu();

void add();

void revenue_report();

void arrival_R();
void modify();

void car_stat();

void user_d();

void modifyActualReturnDate();


// passenger function (all these funcion belongs to passenger or accessible to passenger)

void pass(void);

void book(void);

void return_c(void);

void avail(void);


#endif // end of macro
```

```
//function.c(containing all the function used in program )

//the header files of c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

#include <time.h>

#include "functions.h"


// array of company and there car (categories in order -->hatchback,suv,sudan)

const char* companies[] = {"Tata", "Honda", "Hyundai", "Kia", "Maruti Suzuki"};

const char* tataCars[] = {"Altroz ", "Tigor ", "Harrier "};

const char* hondaCars[] = {"Jazz ", "City ", "CR-V "};

const char* hyundaiCars[] = {"i10 ", "Verna ", "Creta "};

const char* kiaCars[] = {"Picanto ", "K5 ", "Seltos "};

const char* marutiCars[] = {"Swift ", "Ciaz ", "Brezza "};


//defining admin name and password

#define ADMIN_NAME "admin"

#define ADMIN_PASS "password"


static Car cars[MAX_CARS]; //private array of car records, hold upto MAX_CARS(which is 100)

static int carCount = 0;  //private counter for the number of added cars


// Utility functions

void clearScreen() //it help to clear the terminal or console screen

 { system("cls||clear"); }
```

```
void flushInput() //it helps to clears any unwanted characters left in the input buffer.

{ int c; while ((c = getchar()) != '\n' && c != EOF);

}

void getStringInput(const char *prompt, char *buffer, int size) //it  help displays a message (prompt), takes a full line of input from the user, and removes the newline character at the end.

{

   printf("%s", prompt);

   fgets(buffer, size, stdin);

   buffer[strcspn(buffer, "\n")] = '\0';

}


void saveUserDetails(const char *name, const char *phone, const char *aadhar) {

   FILE *fp = fopen("user_details.txt", "a");

   if (!fp) {

      printf("Error opening user_details.txt for writing.\n");

      return;

   }

   fprintf(fp, "%s,%s,%s\n", name, phone, aadhar);

   fclose(fp);

}


// function checking for leap year or not

// Returns 1 if the given year is a leap year, 0 otherwise.

int isLeap(int year) { return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0); }


// Returns the number of days in a given month and year

int getMonthDays(int month, int year) {

   int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

   if (month == 2 && isLeap(year)) return 29;

   return daysInMonth[month - 1];
```

```
}

// Checks if a given date falls within the range of start and end dates (inclusive).
int isDateInRange(Date d, Date start, Date end) {
  if ((d.year > start.year || (d.year == start.year && d.month > start.month) ||
    (d.year == start.year && d.month == start.month && d.day >= start.day)) &&
    (d.year < end.year || (d.year == end.year && d.month < end.month) ||
    (d.year == end.year && d.month == end.month && d.day <= end.day))) {
    return 1;
  }
  return 0;
}

// Loads all car records from the car file into the cars array.
void loadCar() {
  FILE *fp = fopen(CAR_FILE, "r");
  carCount = 0;
  if (fp) {
    while (fscanf(fp, "%d,%[^,],%[^,],%[^,],%[^,],%f,%d\n",
           &cars[carCount].id,
           cars[carCount].carModel,
           cars[carCount].carType,
           cars[carCount].category,
           cars[carCount].route,
           &cars[carCount].price,
           &cars[carCount].available
    ) == 7 && carCount < MAX_CARS) {
      carCount++;
    }
```

```
        fclose(fp);
    }
}

// Saves all cars in the cars array to the car file (overwrites file).
void saveCars() {
    FILE *fp = fopen(CAR_FILE, "w");
    if (!fp) return;
    for (int i = 0; i < carCount; i++) {
        fprintf(fp, "%d,%s,%s,%s,%s,%.2f,%d\n",
                cars[i].id, cars[i].carModel, cars[i].carType,
                cars[i].category, cars[i].route, cars[i].price, cars[i].available);
    }
    fclose(fp);
}

// Appends(add from the bottom of the text or file) a new car record to the car file(cars.txt)
void saveNewCar(Car newCar) {
    FILE *fp = fopen(CAR_FILE, "a");
    if (!fp) {
        printf("Error: Could not open %s for appending.\n", CAR_FILE);
        return;
    }
    fprintf(fp, "%d,%s,%s,%s,%s,%.2f,%d\n",
            newCar.id,
            newCar.carModel,
            newCar.carType,
            newCar.category,
            newCar.route,
        newCar.price,
        newCar.available);
  fclose(fp);
}

//  admin login function(before entering to admin menu,check the credentional)
void login() {
  char user[50], pass[50];
  int attempts = 0;
  while (attempts < 3) {
    clearScreen();
    printf("=== Admin Login (%d/3) ===\n", attempts + 1);
    printf("Username: "); scanf("%49s", user); flushInput();
    printf("Password: ");
    int i = 0; char ch;
    while ((ch = getch()) != 13) {
      if (ch == 8 && i > 0) { i--; printf("\b \b"); }
      else if (i < 49)     { pass[i++] = ch; printf("*"); }
    }
    pass[i] = '\0';
    printf("\n");
    if (!strcmp(user, ADMIN_NAME) && !strcmp(pass, ADMIN_PASS)) {
      printf("Login successful!\n"); getchar();
      adminMenu();
      return;
    }
    attempts++;
    printf("wrong details \n"); getchar();
  }
```

```
    printf("Too many attempts. sorry\n");
    exit(0);
}

// display the admin menu
void adminMenu() {
    int choice;
    do {
        clearScreen();
        printf("=== Admin Menu ===\n");
        printf("1. Add a Car\n");
        printf("2. Generate Revenue Report\n");
        printf("3. View Cars Status\n");
        printf("4. Modify/Delete Record\n");
        printf("5. User Details\n");
        printf("6. Car Arrival Status \n");
        printf("7. Return  \n");
        printf("Choice: "); scanf("%d", &choice); flushInput();
        switch (choice) {
            case 1: add(); break;
            case 2: revenue_report(); break;
            case 3: car_stat(); break;
            case 4: modify(); break;
            case 5: user_d(); break;
            case 6: arrival_R(); break;
            case 7: return;
            default: printf("errorrrrrrr#######\n"); getchar();
        }
    } while (1);
```

```c
}

// add a new car to the console (by menu driven option).
void add() {
    Car newCar;
    int companyChoice, modelChoice;
    const char** selectedModels = NULL;

    printf("=== Add Car ===\n");
    printf("Select Company:\n");
    for (int i = 0; i < 5; i++) {
        printf("%d. %s\n", i + 1, companies[i]);
    }
    printf("Enter choice (1-5): ");
    scanf("%d", &companyChoice); flushInput();

    if (companyChoice < 1 || companyChoice > 5) {
        printf("Invalid company choice!\n"); getchar(); return;
    }
    strcpy(newCar.carModel, companies[companyChoice - 1]);
    switch (companyChoice) {
        case 1: selectedModels = tataCars; break;
        case 2: selectedModels = hondaCars; break;
        case 3: selectedModels = hyundaiCars; break;
        case 4: selectedModels = kiaCars; break;
        case 5: selectedModels = marutiCars; break;
    }
    printf("Select Car Type:\n");
    for (int i = 0; i < 3; i++) {
        printf("%d. %s\n", i+1, selectedModels[i]);
    }
    printf("Enter choice (1-3): ");
    scanf("%d", &modelChoice); flushInput();

    if (modelChoice < 1 || modelChoice > 3) {
        printf("Invalid model choice!\n"); getchar(); return;
    }
    strcpy(newCar.carType, selectedModels[modelChoice - 1]);
    if (modelChoice == 1) strcpy(newCar.category, "Hatchback");
    else if (modelChoice == 2) strcpy(newCar.category, "Sedan");
    else strcpy(newCar.category, "SUV");

    getStringInput("Route: ", newCar.route, sizeof(newCar.route));
    printf("Price per day: "); scanf("%f", &newCar.price); flushInput();

    newCar.available = 1;
    newCar.id = carCount > 0 ? cars[carCount - 1].id + 1 : 1;
    cars[carCount++] = newCar;
    saveNewCar(newCar);

    printf("Car added successfully.\n"); getchar();
}

// generates  the revenue report with help of starting and till you want
void revenue_report() {
    clearScreen();
    printf("=== Revenue Report ===\n");
    Date startDate, endDate;
    printf("Enter start date (DD-MM-YYYY): ");
    scanf("%d-%d-%d", &startDate.day, &startDate.month, &startDate.year);
    printf("Enter end date (DD-MM-YYYY): ");
    scanf("%d-%d-%d", &endDate.day, &endDate.month, &endDate.year);
    flushInput();

    FILE *fp = fopen(BOOKING_FILE, "r");
    if (!fp) {
        printf("No bookings found.\n");
        getchar();
        return;
    }
    Booking b;
    float totalRevenue = 0;
    printf("\n%-10s %-10s %-10s %-12s %-12s %-16s\n", "CarID", "User", "Fare", "Book Date", "Return Date",
"Actual Return");
    printf("----------------------------------------------------------------------\n");

    while (fscanf(fp, " %[^,],%[^,],%[^,],%[^,],%[^,],%d,%[^\n]\n",
            b.bookingID, b.userName, b.carID,
            b.bookingDate, b.returnDate, &b.returned, b.actualReturnDate) != EOF) {

        if (b.returned != 1) continue;
        Date d;
        sscanf(b.bookingDate, "%d-%d-%d", &d.day, &d.month, &d.year);

        // Date in range?(chechk the range of date )
        if (!isDateInRange(d, startDate, endDate)) continue;

        float carPrice = 0;
        int days = 1;

        for (int i = 0; i < carCount; i++) {

            char idstr[16]; sprintf(idstr, "%d", cars[i].id);

            if (strcmp(idstr, b.carID) == 0) {

                carPrice = cars[i].price;

                // Calculate days between book and return

                int bDay, bMonth, bYear, rDay, rMonth, rYear;

                sscanf(b.bookingDate, "%d-%d-%d", &bDay, &bMonth, &bYear);

                sscanf(b.returnDate, "%d-%d-%d", &rDay, &rMonth, &rYear);

                struct tm t1 = {0}, t2 = {0};

                t1.tm_mday = bDay; t1.tm_mon = bMonth - 1; t1.tm_year = bYear - 1900;

                t2.tm_mday = rDay; t2.tm_mon = rMonth - 1; t2.tm_year = rYear - 1900;

                time_t time1 = mktime(&t1), time2 = mktime(&t2);

                days = (int)((difftime(time2, time1) / (60 * 60 * 24))) + 1;

                if (days < 1) days = 1;

                break;

            }

        }

        float fare = carPrice * days;

        printf("%-10s %-10s %-10.2f %-12s %-12s %-16s\n",

               b.carID, b.userName, fare, b.bookingDate, b.returnDate, b.actualReturnDate);

        totalRevenue += fare;

    }

    fclose(fp);

    printf("\ntotal revenue in : Rs %.2f\n", totalRevenue);

    printf("\n Enter to return   ");

    getchar();

}
```

```
// Allows admin to modify car details or delete a car record.
void modify() {
    clearScreen();
    int id, idx = -1, ch;
    printf("Enter Car ID: ");
    scanf("%d", &id); flushInput();

    for (int i = 0; i < carCount; i++) {
        if (cars[i].id == id) {
            idx = i;
            break;
        }
    }
    if (idx < 0) {
        printf("Car not found.\n"); getchar(); return;
    }
    printf("1) Modify\n2) Delete\nChoice: "); scanf("%d", &ch); flushInput();
    if (ch == 1) {
        int done = 0;
        while (!done) {
            clearScreen();
            printf("Modify Car ID %d\n", cars[idx].id);
            printf("1. Change Company (current: %s)\n", cars[idx].carModel);
            printf("2. Change Model (current: %s)\n", cars[idx].carType);
            printf("3. Change Category (current: %s)\n", cars[idx].category);
            printf("4. Change Route (current: %s)\n", cars[idx].route);
            printf("5. Change Price (current: %.2f)\n", cars[idx].price);
            printf("6. Change Availability (current: %s)\n", cars[idx].available ? "Yes" : "No");
            printf("7. Finish\n");
```

```c
printf("Choice: ");
int opt; scanf("%d", &opt); flushInput();
switch (opt) {
  case 1:
    printf("Select Company:\n");
    for (int i = 0; i < 5; i++) printf("%d. %s\n", i + 1, companies[i]);
    printf("Enter choice (1-5): "); int c; scanf("%d", &c); flushInput();
    if (c >= 1 && c <= 5) strcpy(cars[idx].carModel, companies[c-1]);
    else printf("Invalid.\n");
    break;
  case 2: {
    printf("Enter new Model: "); getStringInput("", cars[idx].carType, MAX_STR);
    break;
  }
  case 3:
    printf("Enter new Category: "); getStringInput("", cars[idx].category, MAX_STR);
    break;
  case 4:
    printf("Enter new Route: "); getStringInput("", cars[idx].route, MAX_STR);
    break;
  case 5:
    printf("Enter new Price: "); scanf("%f", &cars[idx].price); flushInput();
    break;
  case 6:
    printf("Set Availability (1 = Yes, 0 = No): "); int av; scanf("%d", &av); flushInput();
    cars[idx].available = av ? 1 : 0;
    break;
  case 7:
    done = 1; break;
```

```
            default:
              printf("errorrrrrrrrrrr.\n"); break;
          }
        }
        saveCars();
        printf("Car modified.\n");
      } else if (ch == 2) {

      }
      getchar();
    }

    // displays the status of all cars, including their booking informations
    void car_stat() {
      clearScreen();
      printf("ID   Company      Model            Category    Route               Price   Available  Booked On     Return By\n");
      printf("----------------------------------------------------------------------------------------------------------------------
    \n");
      for (int i = 0; i < carCount; i++) {
        char bookingDateStr[32] = "-";
        char returnDateStr[32] = "-";
        char availStr[4] = "Yes";

        if (!cars[i].available) {
          strcpy(availStr, "No");
          FILE *fp = fopen(BOOKING_FILE, "r");
          if (fp) {
            Booking b;
            char idstr[16];
            sprintf(idstr, "%d", cars[i].id);
            while (fscanf(fp, " %[^,],%[^,],%[^,],%[^,],%[^,],%d\n",
                       b.bookingID, b.userName, b.carID,
                       b.bookingDate, b.returnDate, &b.returned) != EOF) {
              if (strcmp(b.carID, idstr) == 0 && b.returned == 0) {
                strncpy(bookingDateStr, b.bookingDate, sizeof(bookingDateStr) - 1);
                bookingDateStr[sizeof(bookingDateStr)-1] = '\0';
                strncpy(returnDateStr, b.returnDate, sizeof(returnDateStr) - 1);
                returnDateStr[sizeof(returnDateStr)-1] = '\0';
                break;
              }
            }
            fclose(fp);
          }
        }

        printf("%-4d %-13s %-21s %-11s %-25s %-9.2f %-10s %-12s %-12s\n",
          cars[i].id,
          cars[i].carModel,
          cars[i].carType,
          cars[i].category,
          cars[i].route,
          cars[i].price,
          availStr,
          bookingDateStr,
          returnDateStr
        );
      }
      printf("\nPress Enter to return...");
      flushInput();
```

```
    getchar();
}

// find and displays user details by phone number from user_details.txt
void user_d() {
    clearScreen();
    printf("=== View User Details by Phone Number ===\n");

    char searchPhone[32];
    getStringInput("Enter phone number: ", searchPhone, sizeof(searchPhone));

    FILE *fp = fopen("user_details.txt", "r");
    if (!fp) {
        printf("Error opening user_details.txt or file not found.\n");
        printf("Press Enter to return to the menu...");
        getchar();
        return;
    }
    char line[256];
    int found = 0;
    while (fgets(line, sizeof(line), fp)) {
        // Copy line for tokenizing
        char copy[256];
        strncpy(copy, line, sizeof(copy) - 1);
        copy[sizeof(copy) - 1] = '\0';

        char *name = strtok(copy, ",");
        char *phone = strtok(NULL, ",");
        char *aadhar = strtok(NULL, ",\n");
        if (phone && strcmp(phone, searchPhone) == 0) {
          printf("\nUser details found:\n");
          printf("Name     : %s\n", name ? name : "-");
          printf("Phone    : %s\n", phone);
          printf("Aadhar   : %s\n", aadhar ? aadhar : "-");
          found = 1;
          break;
        }
    }
    fclose(fp);

    if (!found) {
        printf("\nNo user found with phone number: %s\n", searchPhone);
    }
    printf("\nPress Enter to return to the menu...");
    getchar();
}

// shows a report of car arrival/return status for all bookings.
void arrival_R() {
    clearScreen();
    printf("=== Car Arrival Status Report ===\n");
    FILE *fp = fopen(BOOKING_FILE, "r");
    if (!fp) {
        printf("No bookings found.\n");
        printf("Press Enter to return...");
        getchar();
        return;
    }
    Booking b;

    printf("%-10s %-12s %-12s %-12s %-16s %-20s\n", "CarID", "User", "Book Date", "Return Date", "Actual Return",
"Status");

    printf("----------------------------------------------------------------------------------------------------\n");


    while (fscanf(fp, " %[^,],%[^,],%[^,],%[^,],%[^,],%d,%[^\n]\n",
            b.bookingID, b.userName, b.carID,
            b.bookingDate, b.returnDate, &b.returned, b.actualReturnDate) == 7) {
        char status[40] = "-";
        char actualReturnDisp[16] = "-";
        if (b.returned == 1 && strlen(b.actualReturnDate) > 0 && strcmp(b.actualReturnDate, "-") != 0) {
          strncpy(actualReturnDisp, b.actualReturnDate, sizeof(actualReturnDisp)-1);
          actualReturnDisp[sizeof(actualReturnDisp)-1] = '\0';


          // Compare actual return vs expected return
          int rDay=0, rMonth=0, rYear=0, aDay=0, aMonth=0, aYear=0;
          sscanf(b.returnDate, "%d-%d-%d", &rDay, &rMonth, &rYear);
          sscanf(b.actualReturnDate, "%d-%d-%d", &aDay, &aMonth, &aYear);
          struct tm ret_tm = {0}, act_tm = {0};
          ret_tm.tm_mday = rDay; ret_tm.tm_mon = rMonth - 1; ret_tm.tm_year = rYear - 1900;
          act_tm.tm_mday = aDay; act_tm.tm_mon = aMonth - 1; act_tm.tm_year = aYear - 1900;
          time_t t_ret = mktime(&ret_tm), t_act = mktime(&act_tm);
          int diff = (int)((difftime(t_act, t_ret)) / (60*60*24));
          if (diff > 0) sprintf(status, "Arrived Late (%d day%s)", diff, diff == 1 ? "" : "s");
          else if (diff < 0) sprintf(status, "Arrived Early (%d day%s)", -diff, diff == -1 ? "" : "s");
          else strcpy(status, "Arrived On Time");
        } else if (b.returned == 0) {
          strcpy(status, "Not Arrived");
          strcpy(actualReturnDisp, "-");
        } else {
            strcpy(actualReturnDisp, "-");
            strcpy(status, "-");
        }
        printf("%-10s %-12s %-12s %-12s %-16s %-20s\n", b.carID, b.userName, b.bookingDate, b.returnDate,
actualReturnDisp, status);
    }
    fclose(fp);
    printf("\nPress Enter to return...");
    getchar();
}


// displays user menu and runs selected booking/return/check options.
void pass() {
    int choice;
    do {
        clearScreen();
        printf("=== User Menu ===\n");
        printf("1. Book a Car\n");
        printf("2. Return a Car\n");
        printf("3. Check Availability\n");
        printf("4. Return to Main Menu\n");
        printf("Choice: "); scanf("%d", &choice); flushInput();
        switch (choice) {
            case 1: book();         break;
            case 2: return_c();       break;
            case 3: avail(); break;
            case 4: return;
            default: printf("sorry errorrr.\n"); getchar();
        }
    } while (1);
```

```
}

// allows user to book a car, inputs user and booking details, and saves booking.
void book() {
  loadCar();

  printf("Available Cars:\n");
  printf("%-5s %-15s %-20s %-12s %-25s %-10s\n",
    "ID", "Company", "Model", "Category", "Route", "Price");
  int availableCount = 0;
  for (int i = 0; i < carCount; i++) {
    if (cars[i].available) {
      printf("%-5d %-15s %-20s %-12s %-25s %-10.2f\n",
        cars[i].id,
        cars[i].carModel,
        cars[i].carType,
        cars[i].category,
        cars[i].route,
        cars[i].price);
      availableCount++;
    }
  }
  if (availableCount == 0) {
    printf("No cars available for booking.\n");
    getchar();
    return;
  }

  int selectedId = 0, found = 0;
```

```
printf("Enter Car ID to book: ");
scanf("%d", &selectedId); flushInput();
for (int i = 0; i < carCount; i++) {
  if (cars[i].id == selectedId && cars[i].available) {
    found = 1;
    break;
  }
}
if (!found) {
  printf("Invalid or unavailable Car ID.\n");
  getchar();
  return;
}

Booking b;
char phone[20], aadhar[20];
getStringInput("Enter User Name: ", b.userName, MAX_STR);
getStringInput("Enter Phone Number: ", phone, sizeof(phone));
getStringInput("Enter Aadhar Number: ", aadhar, sizeof(aadhar));
saveUserDetails(b.userName, phone, aadhar);
snprintf(b.bookingID, MAX_STR, "B%s-%ld", phone, time(NULL));
snprintf(b.carID, MAX_STR, "%d", selectedId);
getStringInput("Enter Booking Date (dd-mm-yyyy): ", b.bookingDate, MAX_STR);
getStringInput("Enter Return Date (dd-mm-yyyy): ", b.returnDate, MAX_STR);
b.returned = 0;
strcpy(b.actualReturnDate, "-"); // Initialize as not returned yet

// date and fare calculation with validity
int bDay, bMonth, bYear, rDay, rMonth, rYear;
```

```
sscanf(b.bookingDate, "%d-%d-%d", &bDay, &bMonth, &bYear);
sscanf(b.returnDate, "%d-%d-%d", &rDay, &rMonth, &rYear);

if (bYear < 2025 || rYear < 2025 || bYear > 3000 || rYear > 3000) {
    printf("Year must be between 2025 and 3000. Please enter valid dates.\n");
    getchar();
    return;
}

struct tm start = {0}, end = {0};
start.tm_mday = bDay; start.tm_mon = bMonth - 1; start.tm_year = bYear - 1900;
end.tm_mday = rDay;   end.tm_mon = rMonth - 1;   end.tm_year = rYear - 1900;
start.tm_hour = start.tm_min = start.tm_sec = 0;
end.tm_hour = end.tm_min = end.tm_sec = 0;

time_t tStart = mktime(&start);
time_t tEnd = mktime(&end);
if (tStart == (time_t)(-1) || tEnd == (time_t)(-1)) {
    printf("Invalid dates entered. Please enter valid dates.\n");
    getchar();
    return;
}

int days = (int)((difftime(tEnd, tStart) / (60*60*24))) + 1;
if (days < 1) {
    printf("Return date must be the same or after the booking date.\n");
    getchar();
    return;
}
```

```
float fare = 0;
for(int i=0; i<carCount; i++) {
   if(cars[i].id == selectedId) {
      fare = cars[i].price * days;
      break;
   }
}

// Save booking info in bookings.csv WITH 7 FIELDS!
FILE *bfp = fopen(BOOKING_FILE, "a");
if (bfp) {
   fprintf(bfp, "%s,%s,%s,%s,%s,%d,%s\n",
      b.bookingID, b.userName, b.carID, b.bookingDate, b.returnDate, b.returned, b.actualReturnDate);
   fclose(bfp);
} else {
   printf("Error saving booking!\n");
}

// Update car availability and save
for (int i = 0; i < carCount; i++) {
   if (cars[i].id == selectedId) {
      cars[i].available = 0;
      break;
   }
}
saveCars();

printf("Car booked successfully!\nBooking ID: %s\n", b.bookingID);
```

```
    printf("Total fare to pay: Rs %.2f for %d day(s) at Rs %.2f/day\n", fare, days, fare/days);
    getchar();
}

// function handles process of returning a car using booking ID, updates records and car availability.
void return_c() {
    char bookingID[MAX_STR];
    getStringInput("Enter Booking ID to return: ", bookingID, MAX_STR);

    FILE *fp = fopen(BOOKING_FILE, "r");
    FILE *temp = fopen("temp.csv", "w");

    if (!fp || !temp) {
        perror("Error opening file");
        if (fp) fclose(fp);
        if (temp) fclose(temp);
        return;
    }

    Booking b;
    int found = 0;
    char carIdMatched[MAX_STR] = "";

    // Read and write 7 fields for each booking
    while (fscanf(fp, " %[^,],%[^,],%[^,],%[^,],%[^,],%d,%[^\n]\n",
               b.bookingID, b.userName, b.carID,
               b.bookingDate, b.returnDate, &b.returned, b.actualReturnDate) == 7) {
        if (strcmp(b.bookingID, bookingID) == 0 && b.returned == 0) {
            b.returned = 1;
        getStringInput("Enter Actual Return Date (dd-mm-yyyy): ", b.actualReturnDate, MAX_STR);
        strcpy(carIdMatched, b.carID);
        found = 1;
    }
    fprintf(temp, "%s,%s,%s,%s,%s,%d,%s\n",
            b.bookingID, b.userName, b.carID, b.bookingDate, b.returnDate, b.returned, b.actualReturnDate);
}

fclose(fp);
fclose(temp);

remove(BOOKING_FILE);
rename("temp.csv", BOOKING_FILE);

if (found) {
    // Update car availability
    int carid = atoi(carIdMatched);
    loadCar();
    for (int i = 0; i < carCount; i++) {
        if (cars[i].id == carid) {
            cars[i].available = 1;
            break;
        }
    }
    saveCars();

    printf("Car returned successfully!\n");
} else {
    printf("Booking ID not found or already returned.\n");
    }
    getchar();
}

//this function help in showing all currently available cars for booking.
void avail() {
    loadCar();
    printf("Available Cars:\n");
    printf("%-5s %-15s %-20s %-12s %-25s %-10s\n",
        "ID", "Company", "Model", "Category", "Route", "Price");
    int availableCount = 0;
    for (int i = 0; i < carCount; i++) {
        if (cars[i].available) {
            printf("%-5d %-15s %-20s %-12s %-25s %-10.2f\n",
                cars[i].id,
                cars[i].carModel,
                cars[i].carType,
                cars[i].category,
                cars[i].route,
                cars[i].price);
            availableCount++;
        }
    }
    if (availableCount == 0) {
        printf("No cars available.\n");
    }
    printf("\nPress Enter to return...");
    flushInput();
    getchar();}
```