

CIA 3 : PART-B

TOPIC: BUS RESERVATION SYSTEM

Submission Date: 8 September 2024

Under the guidance of

Dr.SHONEY SEBASTIAN

Submitted by

Jethro Jarvis Roy Jyrwa

2447122

Team Members:

1. Athira CM - 2447113
2. Ritam Maity- 2447141

Bus Reservation System

Our project titled Bus Reservation System, is designed to facilitate efficient bus ticket booking. The system is divided into 2 primary modules : Admin and User.

The admin module has the ability to manage bus details like adding new buses, viewing existing bus and so on. Moreover admin can add users and view user details.

The user module enables logged in users to book bus seats. Users can also view their tickets.

Features Used in The Program

1. Preprocessor Directives  
   These are the preprocessor directives that instruct the compiler before the commencement of compilation. They are: #include, #define, declarations of constants.

Usage:  
#include <stdio.h> - Includes the library for printf and scanf.  
#define MAXDROP 5: This declares the maximum number of drop points for a bus.

Example:  
#define MAXDROP 5

2. Constants  
Constants are unchangeable values in the code, declared using const.

Usage:  
const char ADMINID[6]="adm12"; //Constant Admin ID.  
const char ADMINPASS[10] = "admin123";.

Example:  
const char ADMINID[6] = "adm12";

3. Data Types & Variables  
Description: Variables store data of different types, such as int, char, float, and more. These types define the format and size of the data.

Usage:  
int NUMBER\_OF\_BUSES = 50; // Variable for count of buses.  
char user\_name[10]; // Character array to hold the user's name.

Example:  
int NUMBER\_OF\_BUSES = 50;

4. Structures  
A structure is a user-defined data-type that allows different types of data items to be grouped together to form a record.

Usage:  
Struct bus contains information about buses: bus number, boarding point, drop locations, and seat availability.  
struct ticket holds booked ticket information.

Example:  
struct bus {  
int busNo;  
char boarding\_point[20];  
int num\_drop\_loc;  
char drop\_locations[MAXDROP][20];  
int no\_of\_seats;  
int available\_seats;  
};

5. File Handling  
File handling allows a user to create, open, read, write, and close files with the use of functions such as fopen(), fwrite(), and fread().

Usage:  
FILE \*ptr;: declares a file pointer.  
ptr = fopen("bus", "ab");: Opens a file in binary append mode to add a bus record.  
fwrite(&b,sizeof(struct bus),1,ptr);: This writes the bus details to the file.

Example:  
FILE \*ptr = fopen("bus", "ab");  
fwrite(&b, sizeof(struct bus), 1, ptr);

6. Loops: for, do-while  
Loops execute a block of code repeatedly. Use For when the number of iterations is known and Do-While to make sure that the loop always runs once.

Usage:  
for(int i=0; i < b.num\_drop\_loc; i++): Loops through the bus drop locations.  
The do-while loop in the main menu guarantees one iteration.

Example:  
for(int i=0; i < b.num\_drop\_loc; i++);  
printf("Enter Destination %d: ", i+1);  
scanf("%s",b.drop\_locations[i]);  
}

7. Conditionals: if, else if, switch  
Description: Conditional statements are used to execute certain blocks of code based on specific conditions.

Usage:  
Checks for admin ID and password if (strcmp(ADMINID, adminID) == 0) && (strcmp(ADMINPASS, adminPass) == 0),.  
Switch allows having menu options, such as choosing between admin and user.

Example:  
if ((strcmp(ADMINID, adminID)== 0) && (strcmp(ADMINPASS, adminPass)== 0))  
// Admin authentication successful  
}

8. Functions  
Functions are blocks of code that perform a particular activity and can be called more than once. The main() function marks the entry point of this program.

Usage: The function main() governs the program, and addition of buses and users are done within the cases of switch statements.

Example:   
void main() {  
// Main program logic  
}

9. Pointers  
A pointer holds the memory address of a variable. Pointers are used with dynamic memory allocation, handling files, and the passing of large structures or arrays to functions.

Usage:   
FILE \*ptr;: A file pointer for operations.  
Dynamically allocate memory to the bus structure: `struct bus \*b3 = malloc(NUMBER\_OF\_BUSES \* sizeof(struct bus));`.

Example:   
struct bus \*b3 = malloc(NUMBER\_OF\_BUSES \* sizeof(\*b3));

10. Dynamic Memory Allocation (malloc)  
Dynamically allocates memory during runtime using the malloc() function.

Usage:  
struct bus \*b3 = malloc(NUMBER\_OF\_BUSES \* sizeof(struct bus));: Allocates memory for bus details dynamically.

Example:   
struct bus \*b3 = malloc(NUMBER\_OF\_BUSES \* sizeof(\*b3));

11. String Functions (strcmp, strcpy, strcat)  
String manipulation functions, such as strcmp(), strcpy(), and strcat() as well as some other functions, control the string data.

Usage:  
strcmp(ADMINID, adminID): Compares the user-entered admin ID with the constant.  
strcpy(t.user\_name, user); // copies username to the ticket structure.

Example:   
strcpy(t.user\_name, user);

12. Random Number Generation (rand())  
rand() function generates random numbers, often needed when a unique value needs to be created like ticket IDs.

Usage:  
t.ticket\_id = rand();: Assigns a random ticket ID for each booking.

Example:   
t.ticket\_id = random();

13. I/O Functions(printf, scanf)  
These standard functions display output and take user input.

Usage:  
printf("\nEnter choice (1-3): ");: Displays options to the user.  
scanf("%d", &ch); : scanf is used to.

Example: printf("\nChoose (1-3): "); scanf("%d", &ch);

14. Arrays

Arrays store multiple values of the same data type. They can be single-dimensional or multi-dimensional.

Usage: char drop\_locations[MAXDROP][20]; // holds drop locations - makes any amount.

Example: C Copy code char drop\_locations[MAXDROP][20];

Functionalities Of The Program

1. Admin Functionalities:

Admin Login: The admin logs in using predefined credentials (ADMINID and ADMINPASS).

Add a Bus: Admin can add new bus details, including bus number, boarding point, drop points, number of seats, and boarding time.

View Bus Details: Admin can view the list of buses in ascending order based on bus numbers, along with their details (boarding point, drop locations, number of seats).

Add a User: Admin can add new users to the system by entering a user ID, name, password, and phone number.

View Users: Admin can view a list of users, sorted by their username in ascending order.

2. User Functionalities:

User Login: Users can log in using their username and password.

Book Seats: Logged-in users can:

View available buses and select a boarding point and drop location.

Choose the number of seats to book.

Specify the travel date and time.

Book the seats and store the ticket information in a file.

Check Ticket: Users can view their ticket details, including ticket ID, boarding point, drop point, bus number, price, number of passengers, and travel date/time.

3. General Functionalities:

File Operations: The program reads from and writes to files (bus, userDetails, ticketDetails) for storing and retrieving bus, user, and ticket information.

Sorting: Both buses and users are displayed in ascending order (bus number for buses, and username for users).

Data Validation: Admin and user logins require correct credentials, and user inputs (boarding point, drop point, etc.) are validated before processing.

4. Exit:

The user can exit the system anytime by selecting the "EXIT" option from the main menu.

Source code:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

const char ADMINID[6] = "adm12"; //define admin id

const char ADMINPASS[10] = "admin123"; //define admin password

int NUMBER\_OF\_BUSES = 50;

int NUMBER\_OF\_TICKETS = 100; //store the number of tickets in the ticketdetails file

int NUMBER\_OF\_USERS = 50;

#define MAXDROP 5 //maximum drop point for a bus

struct user {

int user\_id;

char user\_name[10];

char user\_password[15];

char phone\_no[10];

};

struct bus {

int bus\_no;

char boarding\_point[20];

int num\_drop\_loc;

char drop\_locations[MAXDROP][20];

int no\_of\_seats;

int available\_no\_seats;

int boarding\_time[4];

};

struct ticket {

char user\_name[10];

int ticket\_id;

char board\_point[10];

char drop\_point[10];

int bus\_no;

float price;

int num\_passenger;

char date[10];

int time;

};

struct dates {

int month;

char date[3];

};

void main()

{

int ch; //main menu choice

FILE \*ptr; //file pointer

struct bus b; //object of bus that stores records of file

int bp; //user boarding point

int user\_dp; //user drop point

struct ticket t;

struct dates d;

char user[10];

char mon[15];

char dropName[10];

do {

printf("\nChoose User"); //main menu

printf("\n1. Admin ");

printf("\n2. User");

printf("\n3. EXIT");

printf("\nEnter your choice:(1-3) ");

scanf("%d",&ch);

switch(ch)

{

case 1: //ADMIN

char adminID[6],adminPass[10];

printf("\nEnter AdminID: ");

scanf("%s",adminID);

printf("\nEnter Password: ");

scanf("%s",adminPass);

if((strcmp(ADMINID,adminID)==0) && (strcmp(ADMINPASS,adminPass)==0)) //admin authentication

{

int chA;

do {

printf("\nAdmin MENU");

printf("\n1. Add a Bus");

printf("\n2. View Bus Details (in ascending order)");

printf("\n3. Add a User");

printf("\n4. View Users (in ascending order)");

printf("\n5. Logout");

printf("\nEnter your choice (1-5): ");

scanf("%d",&chA);

switch(chA)

{

case 1: //Add a bus

int dp;

ptr = fopen("bus","ab");

if(ptr==NULL)

{

printf("File not found");

break;

}

printf("\nEnter Bus No: "); //write bus details to file

scanf("%d",&b.bus\_no);

printf("\nEnter Boarding Point: ");

scanf("%s",b.boarding\_point);

printf("\nEnter No of Drop Points: ");

scanf("%d",&b.num\_drop\_loc);

for(int i = 0; i<b.num\_drop\_loc; i++)

{

printf("Enter Destination %d: ",i+1);

scanf("%s",b.drop\_locations[i]);

}

printf("\nEnter No of Seats: ");

scanf("%d",&b.no\_of\_seats);

b.available\_no\_seats = b.no\_of\_seats;

for(int k=0; k<4; k++)

{

printf("\nEnter Boarding time %d: ",k+1);

scanf("%d",&b.boarding\_time[k]);

}

fwrite(&b,sizeof(struct bus),1,ptr);

fclose(ptr);

break;

case 2: //View Buses

struct bus \*b3 = malloc(NUMBER\_OF\_BUSES \* sizeof(struct bus));

ptr = fopen("bus","rb");

if(ptr==NULL)

{

printf("\nFile not found");

free(b3);

break;

}

size\_t resultB = fread(b3,sizeof(struct bus),NUMBER\_OF\_BUSES,ptr);

for(int i=0;i<resultB-1;i++)

{

for (int j = i + 1; j < resultB; j++)

{

if(b3[i].bus\_no > b3[j].bus\_no)

{

struct bus temp = b3[i];

b3[i] = b3[j];

b3[j] = temp;

}

}

}

for(int i=0; i<resultB; i++) //display bus details after reading from file

{

if(b3[i].bus\_no==0)

{

break;

}

printf("\nBus Number: %d\n",b3[i].bus\_no);

printf("\nBoarding Point: %s\n",b3[i].boarding\_point);

for(int j=0; j<b3[i].num\_drop\_loc; j++)

{

printf("\nDrop location: %d %s\n",j+1,b3[i].drop\_locations[j]);

}

printf("\nNumber of seats: %d",b3[i].no\_of\_seats);

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

}

free(b3);

fclose(ptr);

break;

case 3: //Add user

struct user u;

ptr = fopen("userDetails","ab");

if(ptr==NULL)

{

printf("\nFile not found");

break;

}

printf("\nEnter User ID: ");

scanf("%d",&u.user\_id);

printf("\nEnter UserName: ");

scanf("%s",u.user\_name);

printf("\nEnter User Password: ");

scanf("%s",u.user\_password);

printf("\nEnter User Phone No: ");

scanf("%s",u.phone\_no);

fwrite(&u,sizeof(struct user),1,ptr);

fclose(ptr);

break;

case 4: //displaying users in ascending order

struct user \*u2 = malloc(NUMBER\_OF\_USERS \* sizeof(struct user));

ptr = fopen("userDetails","rb");

if(ptr==NULL)

{

printf("\nFile not found");

break;

}

size\_t users = fread(u2,sizeof(struct user),NUMBER\_OF\_USERS,ptr);

fclose(ptr);

for(int i=0;i<users-1;i++)

{

for (int j = i + 1; j < users; j++)

{

if(strcmp(u2[i].user\_name,u2[j].user\_name)>0)

{

struct user temp = u2[i];

u2[i] = u2[j];

u2[j] = temp;

}

}

}

for(int i=0;i<users;i++)

{

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

printf("\nUser %d",i+1);

printf("\nUser ID: %d",u2[i].user\_id);

printf("\nUserName: %s",u2[i].user\_name);

printf("\nPhoneNo: %s",u2[i].phone\_no);

}

free(u2);

break;

case 5: //Logout

strcpy(adminID,"");

strcpy(adminPass,"");

break;

default:

printf("\nInvalid Entry!!");

break;

}

} while(chA!=4);

}

else

{

printf("\nWrong ID and Password");

}

break;

case 2:

int flag = 0;

struct user u2;

char userName[10],userPass[15];

printf("\nEnter UserName: ");

scanf("%s",userName);

printf("\nEnter Password: ");

scanf("%s",userPass);

ptr = fopen("userDetails","rb");

for(int i = 0; i<NUMBER\_OF\_USERS; i++)

{

size\_t result = fread(&u2,sizeof(struct user),1,ptr);

if((strcmp(u2.user\_name,userName)==0) && (strcmp(u2.user\_password,userPass)==0))

{

flag = 1;

strcpy(user,u2.user\_name);

}

}

fclose(ptr);

if(flag==0)

{

printf("\nUser not found!");

break;

}

else

{

printf("\nLogin Successful");

}

int chU;

if(flag==1)

{

do {

printf("\n\nUser MENU");

printf("\n1. Book seats");

printf("\n2. Check My Ticket");

printf("\n3. Logout");

printf("\nEnter your choice: ");

scanf("%d",&chU);

switch(chU)

{

case 1:

struct bus \*b2 = malloc(NUMBER\_OF\_BUSES \* sizeof(struct bus)); //bus structure defined

int busNo,chT;

ptr = fopen("bus","rb");

if(ptr==NULL)

{

printf("\nFile not found");

break;

}

size\_t result = fread(b2,sizeof(struct bus),NUMBER\_OF\_BUSES,ptr);

if(!result==NUMBER\_OF\_BUSES)

{

printf("\nError reading the data");

}

fclose(ptr);

for(int i=0; i<result; i++) //to display all the boarding points

{

if(strcmp(b2[i].boarding\_point, "") == 0)

{

break;

}

else

{

printf("\n%d.%s",i+1,b2[i].boarding\_point);

}

}

printf("\nEnter your boarding point(1-%zu): ",result); //result stores number of records to retrieve and %zu is the format specifier for size\_t data type

scanf("%d",&bp); //user is choosing which boarding point

int flagBP = 0,no\_drops;

for(int i=0; i<NUMBER\_OF\_BUSES; i++)

{

if(strcmp(b2[i].boarding\_point,b2[bp-1].boarding\_point)==0)

{

for(int j=0; j<b2[i].num\_drop\_loc; j++) //display the drop locations based on the chosen boarding point

{

printf("\n%d. %s",j+1,b2[i].drop\_locations[j]);

no\_drops++;

}

break;

}

if(i==(result-1))

{

printf("\nInvalid Boarding Point. Try again!");

flagBP = 1;

break;

}

}

if(flagBP==1) //if flag is 1 then the boarding point is invalid and needs to restart

{

break;

}

printf("\nEnter the drop location(1-%d): ",no\_drops);

scanf("%d",&user\_dp);

for(int i=0; i<NUMBER\_OF\_BUSES; i++) //display drop points based on the boarding point

{

if((strcmp(b2[i].boarding\_point,b2[bp-1].boarding\_point)==0) && (user\_dp < result))

{

busNo = b2[i].bus\_no;

printf("\nBus\_No is: %d", b2[i].bus\_no);

printf("\nYour Boarding Point is: %s",b2[bp-1].boarding\_point);

printf("\nYour Drop Location is: %s", b2[i].drop\_locations[user\_dp-1]);

strcpy(dropName,b2[i].drop\_locations[user\_dp-1]);

break;

}

if(i==(result-1))

{

printf("\nInvalid Dropping Point. Try again!");

flagBP = 1;

break;

}

}

if(flagBP==1) //if flag is 1 means that the drop location is invalid and needs to restart

{

break;

}

float cost = 100 \* user\_dp; //calculate cost per ticket

printf("\nCost per ticket: %.2f", cost);

char confirm; //store user confirmation whether to book or not

printf("\nDo you want to book a seat?(Y/N)\nDISCLAIMER: Ticket is not cancellable nor refundable! ;D\n\n");

scanf("%s",&confirm);

int seats;

if(confirm == 'Y') //if user confirms then get date,time

{

printf("\nHow many seats do you want to book? ");

scanf("%d",&seats);

printf("\nOn what date do you want to travel (1-31): ");

scanf("%s",d.date);

printf("\nIn what month do you want to travel (1-12): ");

scanf("%d",&d.month);

if(d.month == 1)

{

strcpy(mon," January");

}

else if(d.month == 2)

{

strcpy(mon," February");

}

else if(d.month == 3)

{

strcpy(mon," March");

}

else if(d.month == 4)

{

strcpy(mon," April");

}

else if(d.month == 5)

{

strcpy(mon," May");

}

else if(d.month == 6)

{

strcpy(mon," June");

}

else if(d.month == 7)

{

strcpy(mon," July");

}

else if(d.month == 8)

{

strcpy(mon," August");

}

else if(d.month == 9)

{

strcpy(mon," September");

}

else if(d.month == 10)

{

strcpy(mon," October");

}

else if(d.month == 11)

{

strcpy(mon," November");

}

else if(d.month == 12)

{

strcpy(mon," December");

}

else {

printf("\nInvalid Entry!!");

}

int z;

for(int i=0; i<NUMBER\_OF\_BUSES; i++)

{

if(b2[i].bus\_no == busNo)

{

printf("\nChoose Boarding Time: ");

for(z=0; z<4; z++)

{

printf("\n%d. %d",z+1,b2[i].boarding\_time[z]);

}

}

}

printf("\nChoose the time (1-%d): ",z);

scanf("%d",&chT);

for(int i=0; i<NUMBER\_OF\_BUSES; i++)

{

if(b2[i].bus\_no == busNo)

{

t.time = b2[i].boarding\_time[chT-1];

}

}

for(int i=0; i<NUMBER\_OF\_BUSES; i++) //reducing number of seats required from the available number of seats

{

if(b2[i].bus\_no == busNo)

{

b2[i].available\_no\_seats -= seats;

}

}

strcpy(t.user\_name, user);

t.ticket\_id=rand(); //using rand function to generate ticket id for each user

strcpy(t.board\_point, b2[bp-1].boarding\_point);

strcpy(t.drop\_point,dropName);

t.bus\_no=busNo;

t.price=cost;

t.num\_passenger=seats;

strcpy(t.date,d.date);

strcat(t.date,mon);

ptr = fopen("ticketDetails","ab");

if(ptr==NULL)

{

printf("\nFile not found");

break;

}

fwrite(&t,sizeof(struct ticket),1,ptr);

fclose(ptr);

printf("\nYour ticket has been booked");

}

else{

printf("\n\nSorry to hear that. Check our menu to proceed next!");

}

break;

case 2:

struct ticket \*t1 = malloc(NUMBER\_OF\_TICKETS \* sizeof(struct ticket));

ptr = fopen("ticketDetails","rb");

if(ptr==NULL)

{

printf("\nError in opening the file");

}

size\_t tickets = fread(t1, sizeof(struct ticket),NUMBER\_OF\_TICKETS,ptr);

for(int i=0; i<NUMBER\_OF\_TICKETS; i++)

{

if(strcmp(t1[i].user\_name,user)==0)

{

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

printf("\nTicket ID: %d",t1[i].ticket\_id);

printf("\nBoarding Point: %s",t1[i].board\_point);

printf("\nDropping Point: %s",t1[i].drop\_point);

printf("\nBus No: %d",t1[i].bus\_no);

printf("\nPrice: %.2f",t1[i].price);

printf("\nNo of Seats: %d",t1[i].num\_passenger);

printf("\nDate: %s",t1[i].date);

printf("\nTime: %d",t1[i].time);

}

}

free(t1);

fclose(ptr);

break;

case 3:

strcpy(user,"");

break;

}

} while(chU!=3);

}

break;

case 3:

break;

}

} while(ch!=3);

}

OUTPUT:











