PROJECT REPORT

DATA STRUCTURES AND ALGORITHMS (UCS049)

TOPIC: IMPLEMENTATION OF RAILWAY TICKET RESERVATION SYSTEM

SUBMITTTED BY: ANKIT KUMAR (101804042)
SAMRIDH SHARMA (101804046)
ARSH GUPTA (101804066)
SAHARSH HANDA (101804067)

SUBMITTED TO: DR. DIWAKAR TRIPATHI

SECTION: ELE-3 & ELE-4

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

2021
Electrical and Instrumentation Engineering Department
Thapar Institute of Engineering & Technology, Patiala
(Declared as Deemed-to-be-University u/s 3 of the UGC Act., 1956)
Post Bag No. 32, Patiala – 147004
Punjab (India)

TOPIC:

Write the assumptions and implementation about railway ticket reservation system (as in Indian railways five preferences as Lower, middle, upper, side lower and side upper seat).

ABSTRACT:

The system is based on a concept to reserve train tickets. Here, the program is executed, there's a system where the user is asked to select the seat according to their preference from the choices available and By choosing reservation of ticket, user has to enter their details like name, gender, email etc. There are three cases designed for each type of seat wherein on selection of particular case the user can allocate, cancel or check booked seat details. The system then displays the status of booked seats and user can enter the seat number.The concept is based that on selecting the seat number we assume payment to be processed with payment receipt sent on registered mail id. Further seat number and ticket number are displayed as selected by the user. For the cancellation, user is asked to enter their seat number and the payment is assumed to be refunded upon this. This shows the systematic presentation of the complete ticket booking system.

ASSUMPTIONS:

1. The maximum number of seats taken for each preference (Lower, Middle, Upper, Side Lower, Side Upper) are 100 and as follows:
   1-100 Lower seats
   101-200 Middle seats
   201-300 Upper seats
   301-400 Side Lower seats
   401-500 Side Upper seats

2. The last 20 seats of Lower and Side Lower section are reserved only for the disable people.

3. The payment process for a seat reservation is considered as successful and the payment receipt is sent to the registered email with the booking of a particular seat by the user.

IMPLEMENTATION:

Each of these preference (Lower, Middle, Upper, Side Lower, Side Upper) are implemented using Linked Lists along with a particular structure and corresponding seat allocation and seat cancellation functions.
For example:

For **Lower seat section**:

1. Firstly, the **structure** is created with name **Lower_seat** and the information like name, gender, email, disability(Yes/No) is to be entered by the user.

2. Then, the functions like **allocate_Lower_seat()** for allocation of Lower section seats, **cancel_Lower_seat()** for cancellation of reserved Lower section seat, **booked_Lower_seats_details()** for showing the details of the people who booked the Lower section seats are created.

Similarly, for the other preferences, similar structure and functions are created.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Lower_seat
{
	char name_ls[20];
	char gender[6];
```

```
        char email[30];
        char any_disability[3];
        int seat_number;
        struct Lower_seat *link;
};
struct Lower_seat *start1=NULL;
struct Middle_seat
{
        char name_ms[20];
        char gender[6];
        char email[30];
        char any_disability[3];
        int seat_number;
        struct Middle_seat *link;
};
struct Middle_seat *start2=NULL;
struct Upper_seat
{
        char name_us[20];
        char gender[6];
        char email[30];
        char any_disability[3];
        int seat_number;
        struct Upper_seat *link;
};
struct Upper_seat *start3=NULL;
struct Side_Lower_seat
{
        char name_sls[20];
        char gender[6];
        char email[30];
        char any_disability[3];
        int seat_number;
        struct Side_Lower_seat *link;
};
```

```
struct Side_Lower_seat *start4=NULL;
struct Side_Upper_seat
{
	char name_sus[20];
	char gender[6];
	char email[30];
	char any_disability[3];
	int seat_number;
	struct Side_Upper_seat *link;
};
struct Side_Upper_seat *start5=NULL;
void booked_ls_seats()
{
	struct Lower_seat *p=start1;
	if (start1==NULL)
	{
	printf("All seats are unbooked\n");}
	else
	{
	while(p->link!=NULL)
	{
		printf("%d,",p->seat_number);
		p=p->link;
	}
	printf("%d\n",p->seat_number);
}
}
void booked_ms_seats()
{
	struct Middle_seat *p=start2;
	if (start2==NULL)
	{
	printf("All seats are unbooked\n");}
	else
	{
```

```
	while(p->link!=NULL)
	{
		printf("%d,",p->seat_number);
		p=p->link;
	}
	printf("%d\n",p->seat_number);
}
}
void booked_us_seats()
{
	struct Upper_seat *p=start3;
	if (start3==NULL)
	{
	printf("All seats are unbooked\n");}
	else
	{
	while(p->link!=NULL)
	{
		printf("%d,",p->seat_number);
		p=p->link;
	}
	printf("%d\n",p->seat_number);
}
}
void booked_sls_seats()
{
	struct Side_Lower_seat *p=start4;
	if (start4==NULL)
	{
	printf("All seats are unbooked\n");}
	else
	{
	while(p->link!=NULL)
	{
		printf("%d,",p->seat_number);
```

```
                p=p->link;
        }
        printf("%d\n",p->seat_number);
}
}
void booked_sus_seats()
{
        struct Side_Upper_seat *p=start5;
        if (start5==NULL)
        {
        printf("All seats are unbooked\n");}
        else
        {
        while(p->link!=NULL)
        {
                printf("%d,",p->seat_number);
                p=p->link;
        }
        printf("%d\n",p->seat_number);
}
}
void booked_Lower_seats_details()
{
struct Lower_seat *temp=start1;
if(temp==NULL)
{
        printf("Data can't be shown as all seats are unbooked right now\n");
}
else
{
        while(temp->link!=NULL)
        {
                printf("NAME: %s\n",temp->name_ls);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: L-%d\n",temp->seat_number);
                temp=temp->link;
         }
        printf("NAME: %s\n",temp->name_ls);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: L-%d\n",temp->seat_number);
}
}
void booked_Middle_seats_details()
{
struct Middle_seat *temp=start2;
if(temp==NULL)
{
        printf("Details can't be shown as all seats are unbooked right now\n");
}
else
{
        while(temp->link!=NULL)
        {
                printf("NAME: %s\n",temp->name_ms);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: M-%d\n",temp->seat_number);
                temp=temp->link;
         }
        printf("NAME: %s\n",temp->name_ms);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: M-%d\n",temp->seat_number);
}
```

```
}
void booked_Upper_seats_details()
{
struct Upper_seat *temp=start3;
if(temp==NULL)
{
        printf("Data can't be shown as all seats are unbooked right now\n");
}
else
{
        while(temp->link!=NULL)
        {
                printf("NAME: %s\n",temp->name_us);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: U-%d\n",temp->seat_number);
                temp=temp->link;
        }
       printf("NAME: %s\n",temp->name_us);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: U-%d\n",temp->seat_number);
}
}
void booked_Side_Lower_seats_details()
{
struct Side_Lower_seat *temp=start4;
if(temp==NULL)
{
        printf("Data can't be shown as all seats are unbooked right now\n");
}
else
{
        while(temp->link!=NULL)
        {
                printf("NAME: %s\n",temp->name_sls);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: SL-%d\n",temp->seat_number);
                temp=temp->link;
        }
       printf("NAME: %s\n",temp->name_sls);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: SL-%d\n",temp->seat_number);
}
}
void booked_Side_Upper_seats_details()
{
struct Side_Upper_seat *temp=start5;
if(temp==NULL)
{
        printf("Data can't be shown as all seats are unbooked right now\n");
}
else
{
        while(temp->link!=NULL)
        {
                printf("NAME: %s\n",temp->name_sus);
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: SU-%d\n",temp->seat_number);
                temp=temp->link;
        }
       printf("NAME: %s\n",temp->name_sus);
```

```
                printf("GENDER: %s\n",temp->gender);
                printf("E-mail: %s\n",temp->email);
                printf("ANY DISABILITY: %s\n",temp->any_disability);
                printf("SEAT NUMBER: SU-%d\n",temp->seat_number);
}
}
int check_lower_seat(int a)
{
        struct Lower_seat *t1=start1;
        if(start1==NULL)
        return 0;
        while(t1->link!=NULL)
        {
                if(t1->seat_number==a)
                {
                        return 1;
                }
                t1=t1->link;
        }
        if(t1->seat_number==a)
        return 1;
        else
        return 0;
}
int check_middle_seat(int a)
{
        struct Middle_seat *t1=start2;
        if(start2==NULL)
        return 0;
        while(t1->link!=NULL)
        {
                if(t1->seat_number==a)
                {
                        return 1;
                }
```

```
		t1=t1->link;
	}
	if(t1->seat_number==a)
	return 1;
	else
	return 0;
}
int check_upper_seat(int a)
{
	struct Upper_seat *t1=start3;
	if(start3==NULL)
	return 0;
	while(t1->link!=NULL)
	{
		if(t1->seat_number==a)
		{
			return 1;
		}
		t1=t1->link;
	}
	if(t1->seat_number==a)
	return 1;
	else
	return 0;
}
int check_side_lower_seat(int a)
{
	struct Side_Lower_seat *t1=start4;
	if(start4==NULL)
	return 0;
	while(t1->link!=NULL)
	{
		if(t1->seat_number==a)
		{
			return 1;
		}
		t1=t1->link;
	}
	if(t1->seat_number==a)
	return 1;
	else
	return 0;
}
int check_side_upper_seat(int a)
{
	struct Side_Upper_seat *t1=start5;
	if(start5==NULL)
	return 0;
	while(t1->link!=NULL)
	{
		if(t1->seat_number==a)
		{
			return 1;
		}
		t1=t1->link;
	}
	if(t1->seat_number==a)
	return 1;
	else
	return 0;
}
void allocate_Lower_seat()
{
	struct Lower_seat *temp=(struct Lower_seat *)malloc(sizeof(struct
Lower_seat));
	printf("enter your name for lower seat reservation(limit 20 letters and
use underscore instead of space)\n");
	scanf("%s",temp->name_ls);
	printf("enter your gender(Male/Female)\n");
	scanf("%s",temp->gender);
	printf("enter your email id(maximum 30 characters and no spaces are 
allowed)\n");
	scanf("%s",temp->email);
	printf("Any type of Disability(Yes/No)\n");
	scanf("%s",temp->any_disability);
	printf("Total seats available are 100 from 1-100 (last 20 seats are 
reserved for disable persons only) and already booked seats are:\n");
	booked_ls_seats();
	printf("enter your seat number from the remaining seats in 1-100 (Note: 
seats 81-100 are for disable persons only)\n");
    scanf("%d",&temp->seat_number);
    if(temp->seat_number<1 || temp->seat_number>100 || 
check_lower_seat(temp->seat_number))
    {
    printf("Seat number is not avaible or already reserved so again enter your 
seat number from the remaining seats in 1-100 (Note: seats 81-100 are for 
disable persons only)\n");
    scanf("%d",&temp->seat_number);
	}
    printf("Payment processed \nPayment receipt has beeen sent on the 
registered email id\nAllocated seat number and ticket number are L-%d and 
%d respectively\n",temp->seat_number,temp->seat_number);
	temp->link=NULL;
	if(start1==NULL)
	{
		start1=temp;
}
else
{
	struct Lower_seat *ptr=start1;
	while(ptr->link!=NULL)
	{
		ptr=ptr->link;
	}
	ptr->link=temp;
```

```
}
}
void allocate_Middle_seat()
{
        struct Middle_seat *temp=(struct Middle_seat *)malloc(sizeof(struct
Middle_seat));
        printf("enter your name for middle seat reservation(limit 20 letters and
use underscore instead of space)\n");
        scanf("%s",temp->name_ms);
        printf("enter your gender(Male/Female)\n");
        scanf("%s",temp->gender);
        printf("enter your email id(maximum 30 characters and no spaces are
allowed)\n");
        scanf("%s",temp->email);
        printf("Any type of Disability(Yes/No)\n");
        scanf("%s",temp->any_disability);
        printf("Total seats available are 100 from 101-200 and already booked
seats are:\n");
        booked_ms_seats();
        printf("enter your seat number from the remaining seats in 101-200\n");
    scanf("%d",&temp->seat_number);
    if(temp->seat_number<101 || temp->seat_number>200 ||
check_middle_seat(temp->seat_number))
    {
    printf("Seat number is not avaible or already reserved so again enter your
seat number from the remaining seats in 101-200\n");
    scanf("%d",&temp->seat_number);
        }
    printf("Payment processed \nPayment receipt has beeen sent on the
registered email id\nAllocated seat number and ticket number are M-%d and
%d respectively\n ",temp->seat_number,temp->seat_number);
        temp->link=NULL;
        if(start2==NULL)
        {
                start2=temp;
```

```
}
else
{
        struct Middle_seat *ptr=start2;
        while(ptr->link!=NULL)
        {
                ptr=ptr->link;
        }
        ptr->link=temp;
}
}
void allocate_Upper_seat()
{
        struct Upper_seat *temp=(struct Upper_seat *)malloc(sizeof(struct
Upper_seat));
        printf("enter your name for upper seat reservation(limit 20 letters and
use underscore instead of space)\n");
        scanf("%s",temp->name_us);
        printf("enter your gender(Male/Female)\n");
        scanf("%s",temp->gender);
        printf("enter your email id(maximum 30 characters and no spaces are
allowed)\n");
        scanf("%s",temp->email);
        printf("Any type of Disability(Yes/No)\n");
        scanf("%s",temp->any_disability);
        printf("Total seats available are 100 from 201-300 and already booked
seats are:\n");
        booked_us_seats();
        printf("enter your seat number from the remaining seats in 201-300\n");
    scanf("%d",&temp->seat_number);
    if(temp->seat_number<201 || temp->seat_number>300 ||
check_upper_seat(temp->seat_number))
    {
    printf("Seat number is not avaible or already reserved so again enter your
seat number from the remaining seats in 201-300\n");
```

```
   scanf("%d",&temp->seat_number);
       }
   printf("Payment processed \nPayment receipt has beeen sent on the
registered email id\nAllocated seat number and ticket number are U-%d and
%d respectively\n ",temp->seat_number,temp->seat_number);
       temp->link=NULL;
       if(start3==NULL)
       {
              start3=temp;
}
else
{
       struct Upper_seat *ptr=start3;
       while(ptr->link!=NULL)
       {
              ptr=ptr->link;
       }
       ptr->link=temp;
}
}
void allocate_Side_Lower_seat()
{
       struct Side_Lower_seat *temp=(struct Side_Lower_seat
*)malloc(sizeof(struct Side_Lower_seat));
       printf("enter your name for side lower seat reservation(limit 20 letters
and use underscore instead of space)\n");
       scanf("%s",temp->name_sls);
       printf("enter your gender(Male/Female)\n");
       scanf("%s",temp->gender);
       printf("enter your email id(maximum 30 characters and no spaces are
allowed)\n");
       scanf("%s",temp->email);
       printf("Any type of Disability(Yes/No)\n");
       scanf("%s",temp->any_disability);
       printf("Total seats available are 100 from 301-400 (last 20 seats are
reserved for disable persons only) and already booked seats are:\n");
       booked_sls_seats();
       printf("enter your seat number from the remaining seats in 301-400
(Note: seats 381-400 are for disable persons only)\n");
   scanf("%d",&temp->seat_number);
   if(temp->seat_number<301 || temp->seat_number>400 ||
check_side_lower_seat(temp->seat_number))
   {
   printf("Seat number is not avaible or already reserved so again enter your
seat number from the remaining seats in 301-400 (Note: seats 381-400 are for
disable persons only)\n");
   scanf("%d",&temp->seat_number);
       }
   printf("Payment processed \nPayment receipt has beeen sent on the
registered email id\nAllocated seat number and ticket number are SL-%d and
%d respectively\n ",temp->seat_number,temp->seat_number);
       temp->link=NULL;
       if(start4==NULL)
       {
              start4=temp;
}
else
{
       struct Side_Lower_seat *ptr=start4;
       while(ptr->link!=NULL)
       {
              ptr=ptr->link;
       }
       ptr->link=temp;
}
}
void allocate_Side_Upper_seat()
{
       struct Side_Upper_seat *temp=(struct Side_Upper_seat
*)malloc(sizeof(struct Side_Upper_seat));
```

```
	printf("enter your name for side upper seat reservation(limit 20 letters
and use underscore instead of space)\n");
	scanf("%s",temp->name_sus);
	printf("enter your gender(Male/Female)\n");
	scanf("%s",temp->gender);
	printf("enter your email id(maximum 30 characters and no spaces are
allowed)\n");
	scanf("%s",temp->email);
	printf("Any type of Disability(Yes/No)\n");
	scanf("%s",temp->any_disability);
	printf("Total seats available are 100 from 401-500 and already booked
seats are:\n");
	booked_sus_seats();
	printf("enter your seat number from the remaining seats in 401-500\n");
    scanf("%d",&temp->seat_number);
    if(temp->seat_number<401 || temp->seat_number>500 ||
check_side_upper_seat(temp->seat_number))
    {
    printf("Seat number is not avaible or alredy reserved so again enter your
seat number from the remaining seats in 401-500\n");
    scanf("%d",&temp->seat_number);
	}
    printf("Payment processed \nPayment receipt has beeen sent on the
registered email id\nAllocated seat number and ticket number are SU-%d and
%d respectively\n ",temp->seat_number,temp->seat_number);
	temp->link=NULL;
	if(start5==NULL)
	{
		start5=temp;
}
else
{
	struct Side_Upper_seat *ptr=start5;
	while(ptr->link!=NULL)
	{
```

```
                ptr=ptr->link;
        }
        ptr->link=temp;
}
}
void cancel_Lower_seat()
{
        int seat_num;
        printf("enter your allocated seat number for seat cancellation\n");
        scanf("%d",&seat_num);
                if(start1==NULL)
        {
                printf("No such seat is allocated yet\n");
        }
   struct Lower_seat *ptr=start1;
   struct Lower_seat *pq=start1->link;
   if(ptr->seat_number==seat_num)
{
        start1=ptr->link;
        free(ptr);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
}
else if(pq==NULL)
{
                printf("No such seat is allocated yet\n");
}
   else if(ptr->seat_number!=seat_num)
   {
   while(pq->link!=NULL)
   {
   if(pq->seat_number==seat_num)
   {
        struct Lower_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
        break;
    }
        ptr=ptr->link;
        pq=pq->link;
        }
        if(pq->link==NULL)
        {
                  if(pq->seat_number==seat_num)
    {
        struct Lower_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
    }
                else
                printf("No such seat is allocated yet\n");
        }
}
}
void cancel_Middle_seat()
{
        int seat_num;
        printf("enter your allocated seat number for seat cancellation\n");
        scanf("%d",&seat_num);
                if(start2==NULL)
        {
                printf("No such seat is allocated yet\n");
        }
    struct Middle_seat *ptr=start2;
    struct Middle_seat *pq=start2->link;
    if(ptr->seat_number==seat_num)
```

```
{
        start2=ptr->link;
        free(ptr);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
}
else if(pq==NULL)
{
                printf("No such seat is allocated yet\n");
}
    else if(ptr->seat_number!=seat_num)
    {
    while(pq->link!=NULL)
    {
        if(pq->seat_number==seat_num)
        {
                        struct Middle_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
        break;
                }
        ptr=ptr->link;
        pq=pq->link;
        }
if(pq->link==NULL)
{   if(pq->seat_number==seat_num)
        {
                                struct Middle_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
}
```

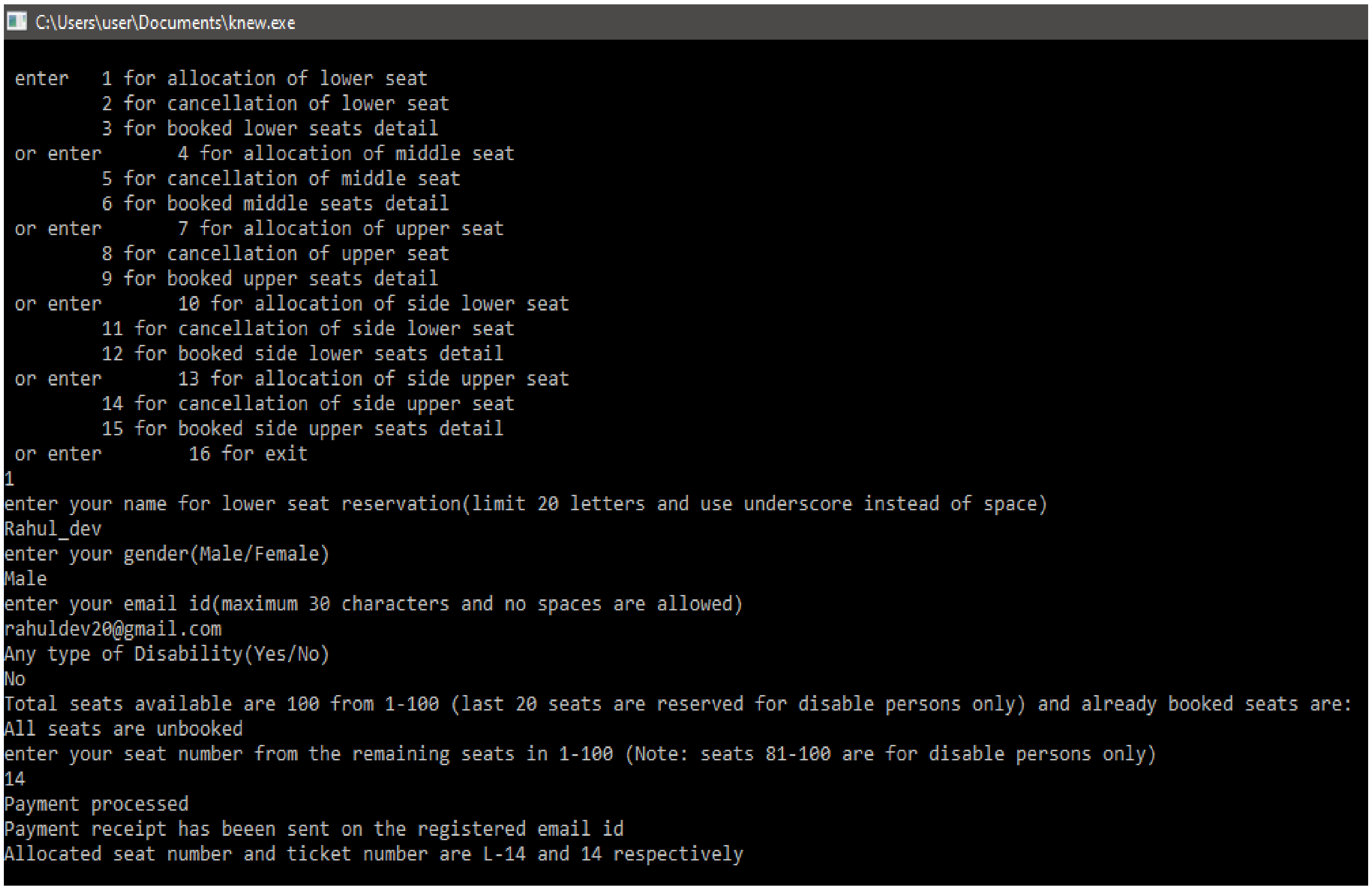
```
else
printf("No such seat is allocated yet\n");
}
}
}
void cancel_Upper_seat()
{
        int seat_num;
        printf("enter your allocated seat number for seat cancellation\n");
        scanf("%d",&seat_num);
                if(start3==NULL)
        {
                printf("No such seat is allocated yet\n");
        }
    struct Upper_seat *ptr=start3;
    struct Upper_seat *pq=start3->link;
    if(ptr->seat_number==seat_num)
    {
                start3=ptr->link;
        free(ptr);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
}
else if(pq==NULL)
{
                printf("No such seat is allocated yet\n");
}
else if(pq->seat_number!=seat_num){
    while(pq->link!=NULL)
    {
        if(pq->seat_number==seat_num)
        {
                struct Upper_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
        break;
                }
        ptr=ptr->link;
        pq=pq->link;
        }
        if(pq->link==NULL)
        {
                        if(pq->seat_number==seat_num)
        {
                struct Upper_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
                }
                else
                printf("No such seat is allocated yet\n");
        }
}
}
void cancel_Side_Lower_seat()
{
        int seat_num;
        printf("enter your allocated seat number for seat cancellation\n");
        scanf("%d",&seat_num);
                if(start4==NULL)
        {
                printf("No such seat is allocated yet\n");
        }
   struct Side_Lower_seat *ptr=start4;
   struct Side_Lower_seat *pq=start4->link;
   if(ptr->seat_number==seat_num)
   {
```

```
            start4=ptr->link;
        free(ptr);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
}
else if(pq==NULL)
{
            printf("No such seat is allocated yet\n");
}
else if(pq->seat_number!=seat_num)
{
   while(pq->link!=NULL)
   {  if(pq->seat_number==seat_num)
   {
            struct Side_Lower_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
        break;
        }
        ptr=ptr->link;
        pq=pq->link;
        }
if(pq->link==NULL)
{
        if(pq->seat_number==seat_num)
   {
            struct Side_Lower_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
        }
        else
```

```
                printf("No such seat is allocated yet\n");
}
}
}
void cancel_Side_Upper_seat()
{
        int seat_num;
        printf("enter your allocated seat number for seat cancellation\n");
        scanf("%d",&seat_num);
                if(start5==NULL)
        {
                printf("No such seat is allocated yet\n");
        }
    struct Side_Upper_seat *ptr=start5;
    struct Side_Upper_seat *pq=start5->link;
    if(ptr->seat_number==seat_num)
    {
                start5=ptr->link;
        free(ptr);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
}
else if(pq==NULL)
{
                printf("No such seat is allocated yet\n");
}
else if(pq->seat_number!=seat_num)
{
    while(pq->link!=NULL)
    {
        if(pq->seat_number==seat_num)
        {
                        struct Side_Upper_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
        break;
                }
        ptr=ptr->link;
        pq=pq->link;
        }
        if(pq->link==NULL)
        {
                        if(pq->seat_number==seat_num)
        {
                        struct Side_Upper_seat *a=pq;
        ptr->link=a->link;
        free(a);
        printf("Your allocated seat is cancelled and money will be refunded
according to the system norms\n");
                }
                else
                printf("No such seat is allocated yet\n");
        }
}
}
int main()
{
        int a;
        int ch;
    do
    {
    printf("enter    1 for Lower section seats detail \n \t 2 for middle section
seats detail \n \t 3 for upper section seats detail \n \t 4 for side lower section
seats detail \n \t 5 for side upper section seats detail \n \t 6 for exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("enter    11 for allocation of Lower seat \n \t 12 for
cancellation of Lower seat \n \t 13 for booked lower seats detail \n \t 14 for
exit");
            scanf("%d",&a);
            switch(a)
            {
                case 11: allocate_Lower_seat(); break;
                case 12: cancel_Lower_seat(); break;
                case 13: booked_Lower_seats_details(); break;
                case 14: exit(0);
                default: printf("enter correct choice\n");
            }
            break;
            case 2:printf("enter    11 for allocation of Middle seat \n \t 12 for
cancellation of Middle seat \n \t 13 for booked Middle seats detail \n \t 14 for
exit");
            scanf("%d",&a);
            switch(a)
            {
                case 11: allocate_Middle_seat(); break;
                case 12: cancel_Middle_seat(); break;
                case 13: booked_Middle_seats_details(); break;
                case 14: exit(0);
                default: printf("enter correct choice\n");
            }
            break;
            case 3:printf("enter    11 for allocation of Upper seat \n \t 12 for
cancellation of Upper seat \n \t 13 for booked Upper seats detail \n \t 14 for
exit");
            scanf("%d",&a);
            switch(a)
            {
                case 11: allocate_Upper_seat(); break;
                case 12: cancel_Upper_seat(); break;
                case 13: booked_Upper_seats_details(); break;
                case 14: exit(0);
```

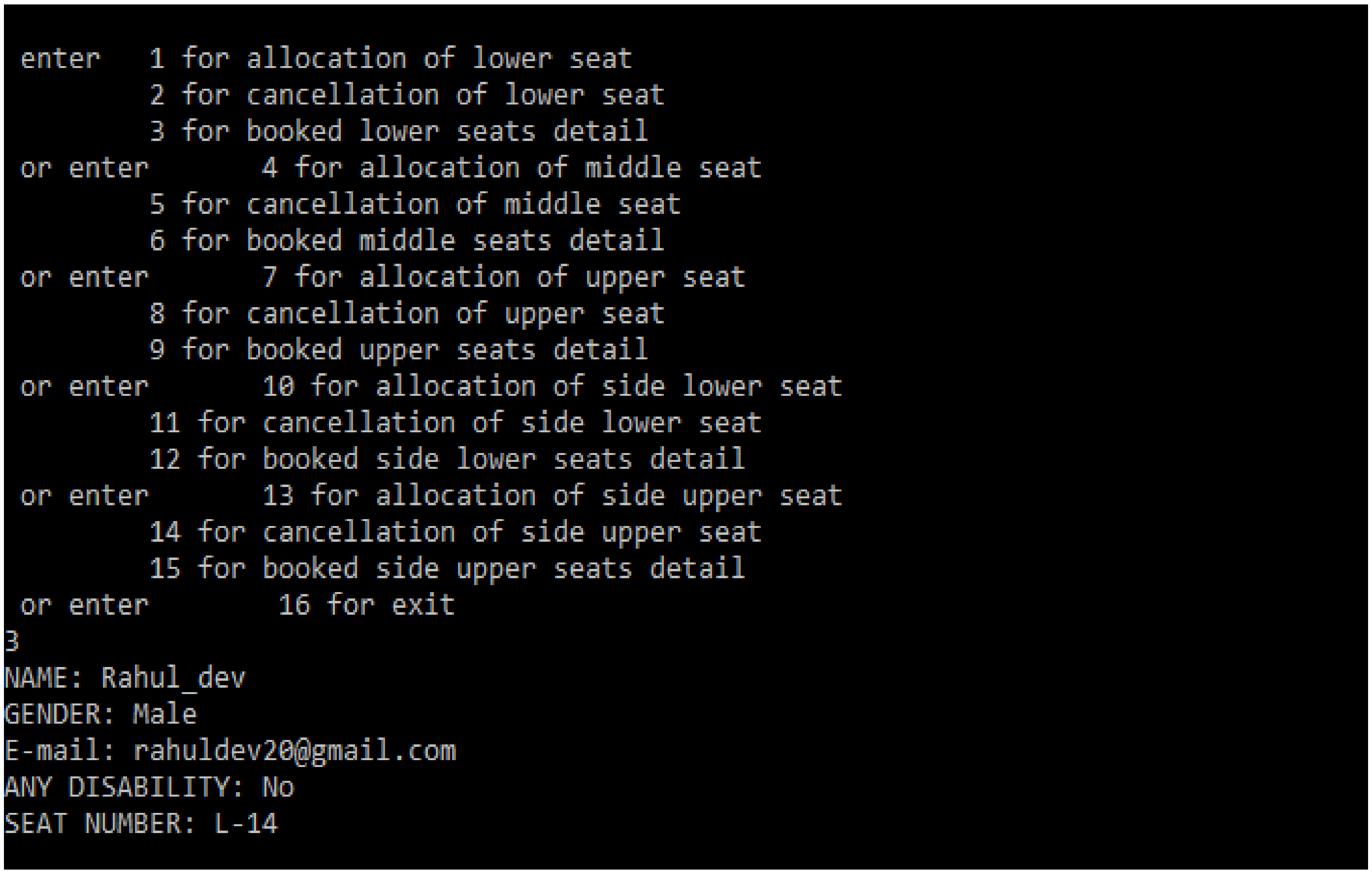
```
            default: printf("enter correct choice\n");
        }
        break;
        case 4:printf("enter    11 for allocation of Side Lower seat \n \t 12 for
cancellation of Side Lower seat \n \t 13 for booked Side Lower seats detail \n
\t 14 for exit");
        scanf("%d",&a);
        switch(a)
        {
            case 11: allocate_Side_Lower_seat(); break;
            case 12: cancel_Side_Lower_seat(); break;
            case 13: booked_Side_Lower_seats_details(); break;
            case 14: exit(0);
            default: printf("enter correct choice\n");
        }
        break;
        case 5:printf("enter    11 for allocation of Side Upper seat \n \t 12 for
cancellation of Side Upper seat \n \t 13 for booked Side Upper seats detail \n
\t 14 for exit");
        scanf("%d",&a);
        switch(a)
        {
            case 11: allocate_Side_Upper_seat(); break;
            case 12: cancel_Side_Upper_seat(); break;
            case 13: booked_Side_Upper_seats_details(); break;
            case 14: exit(0);
            default: printf("enter correct choice\n");
        }
        break;
            case 6: exit(0); break;
            default: printf("enter correct choice\n");
     }
}while(ch!=6);
}
```

OUTPUT:



C:\Users\user\Documents\knew.exe

```
 enter   1 for allocation of lower seat
         2 for cancellation of lower seat
         3 for booked lower seats detail
 or enter       4 for allocation of middle seat
         5 for cancellation of middle seat
         6 for booked middle seats detail
 or enter       7 for allocation of upper seat
         8 for cancellation of upper seat
         9 for booked upper seats detail
 or enter       10 for allocation of side lower seat
         11 for cancellation of side lower seat
         12 for booked side lower seats detail
 or enter       13 for allocation of side upper seat
         14 for cancellation of side upper seat
         15 for booked side upper seats detail
 or enter        16 for exit
1
enter your name for lower seat reservation(limit 20 letters and use underscore instead of space)
Rahul_dev
enter your gender(Male/Female)
Male
enter your email id(maximum 30 characters and no spaces are allowed)
rahuldev20@gmail.com
Any type of Disability(Yes/No)
No
Total seats available are 100 from 1-100 (last 20 seats are reserved for disable persons only) and already booked seats are:
All seats are unbooked
enter your seat number from the remaining seats in 1-100 (Note: seats 81-100 are for disable persons only)
14
Payment processed
Payment receipt has beeen sent on the registered email id
Allocated seat number and ticket number are L-14 and 14 respectively
```



```
 enter   1 for allocation of lower seat
         2 for cancellation of lower seat
         3 for booked lower seats detail
 or enter       4 for allocation of middle seat
         5 for cancellation of middle seat
         6 for booked middle seats detail
 or enter       7 for allocation of upper seat
         8 for cancellation of upper seat
         9 for booked upper seats detail
 or enter       10 for allocation of side lower seat
         11 for cancellation of side lower seat
         12 for booked side lower seats detail
 or enter       13 for allocation of side upper seat
         14 for cancellation of side upper seat
         15 for booked side upper seats detail
 or enter        16 for exit
3
NAME: Rahul_dev
GENDER: Male
E-mail: rahuldev20@gmail.com
ANY DISABILITY: No
SEAT NUMBER: L-14
```