SUJAY HAZRA

1NH17CS127

III B

Topic: Movie Booker

SEM: III

ABSTRACT

"Movie Booker" as the name suggests is a program that helps people book movie tickets on their own, without any assistance. This project is best suited for Movie theatre owners. While it can also be implemented for booking other venues/facilities that requires monitoring of seats.
This Project fastens the booking process.
Customer do not have to wait in long queues.

What makes this program unique from the rest of the booking system projects is the seat arrangement display which dynamically updates itself according to the booking of the user, this makes it easier for the user to know the available seats and its location at the theater.

This program can be cited in the following steps:-

1) Owner Initializes the booking.
2) User Can view movie details and seat details and book their desired seats.
3) User Can cancel their booked seats.
4) At the End Owner can calculate and get the statistics related to booking including details of customers on no. os seats booked and revenue.

Further improvements can be made by making the program online, tickets can be booked from homes, Higher-end Graphics can be provided making it a GUI, movie trailers can be displayed alongside complete movie info, e-tickets could be sent via SMS, payments can be made online, this program could be implemented in multiplexes and the code can be altered accordingly to facilitate multiple screenings at the same time.

CHAPTER 1

INTRODUCTION

In this project, The owner specifies the movie details, price, max no. of seats available for a single screening and based on it no. of tickets to be sold can be limited and when the seats are full, further booking is no longer available and a House Full Message is displayed.

Once the initialization is done, The Project allows customers to view the details (Name of the movie, rating, rotten tomatoes, price) of the Movie available for screening and the seating arrangement of the theater, with booking status of each seat("0" means the seat has been booked) and lets them book the tickets by specifying their Name, No. of Seats and Seat numbers.
This Program is used for on the spot booking of tickets. Customer can view booked seats & select the desired seat based on seat numbers available. Once done the user can go for calculation, The total Seats sold, and money accumulated can be calculated along with details of each customer about no. of seats booked by each and the price.
The Objective of this Project is to provide digital facilities to Theatre Owners to facilitate ticket booking process with ease, providing an interface to book tickets.

The User can also view their details and cancel their booking if required.

CHAPTER 2

ANALYSIS AND DESIGN

2.1 OBJECTIVES OF THE PROJECT

Our main objective here is to use Linked Lists in unique ways to facilitate booking process with ease, Linked list makes it easier to perform various operations on Data. The main Operations being : Addition & Removal of data, Displaying of data in a matrix format, Matchmaking & Searching, Calculation and Modification.

2.2 REQUIREMENT SPECIFICATION

2.2.1. Hardware Requirement

- Processor : X86 Compatible processor with 1.7 GHz Clock speed
- RAM : 512 MB or more
- Hard disk : 500 GB

2.2.2. Software Requirements

- Windows 10 Operating System
- Basic C compiler (Turbo C)

2.2.3.Pre-Requisite Knowledge

- Basic knowledge on Data Structures, C Programming, Sorting Methods.
- Basic Level of Mathematics.
- An Idea on how movies are booked.

CHAPTER 3

DESIGN

3.1 ALGORITHM

Step1: Start
Step2: Choose
1.Initialize
2.Start Booking
3.Finish & Calculate
4.Quit Program
Step3: Initialize
Enter the following data
1.Movie Info
2.Price per Ticket
3.No of seats per Row
4.No of Row
5.Save (yes/No)
6.Back to Step 2
Step4: Start Booking
Movie info is displayed
4.1-Book
Seating Arrangement Displayed along with current seating arrangement showing booking status for each seat, “0” for booked seats and number for available seats.
4.1.1-Proceed
1. Enter Name
2. Enter No. of seats
3. Enter Seat Numbers
Booking successful, amount displayed.
Returned to step4.
If any fault “Book” is repeated.
4.1.2-Exit
Go Back to Step4.
4.2-Cancel Booking
4.2.1-Enter Name Under which it was booked.
4.2.2-Display seats booked under name if matched else re enter name.
4.2.3-Select No of Tickets to be cancelled
4.2.4-Enter seat numbers to be removed
4.2.5-Display success or not, and amount refunded. Go back to Step4.

4.3-Exit (back to step 2)

Step5: Finsih & Calculation

5.1-Display the Following details

1.Total Seat Available

2.No, Of Seats Booked

3.Total revenue

4.Percentage of Booked Seats

5.Record of Customers

a.Name

b.No. Of Tickets

c.Cost

6.Back to Step2

Step6: Exit Program

Step7: END

3.2 DATA STRUCTURES

1. Linked_list:

   A linked list is a data structure which is collection of zero or more node where each node has some information. Every node consists of two fields namely Data and Link. Given address of a node we can easily obtain addresses of subsequent nodes. Linked list is the simplest of data types. Its Advantage over array include No fixed size, Insertion is easier, Deletion is easier, Continuous memory chunk is not required, etc. A linked list is usually created with an user defined data type that is created with the help of structures.

2. Array:

   Arrays are a kind of data structure that can store a fixed size of sequential collection of elements of similar type. Instead of declaring individual variables like no.0, no.1…. Etc., we can declare one array such as numbers and use Number [0], number [2] … and so on. All arrays consist of contiguous memory locations.
   Declaration of Arrays in C :
   Type arrayName [arraySize];
   int a [10] ;

   In this program the array is used to store all the seat numbers to be displayed in the form of matrix. And to take data such as name, etc from the user.

CHAPTER 4

IMPLEMENTATION

1. Structure of Linked List

```
struct node
{
 char name[50];
 int seat[50],nos,cost;
 struct node *link;
}*first=NULL;
```

2. Booking Seats

a. Displaying Seating Arrangement:-

[1]

```
void book()
{
 int i,j,h,match=0,choice,z=1,k=1;
 struct node* temp=first;
 clrscr();
 printf("Seating Arrangement\n");
 for(i=0;i<no_row;i++)
  {
   for(j=0;j<no_seat;j++)
   {
    printf("%d\t",k);
    k++;
   }
   printf("\n");
```

```
 }
```

The Above Code Displays a Normal Matrix With seat Numbers having row=no_row and column=no_column which was supposed to be initialized at the start.

```
 [2]
k=1;
 printf("current seating arrangement\n");
for(i=0;i<no_row;i++)
 {
  for(j=0;j<no_seat;j++)
   {
    while(temp!=NULL)
    {
      for(h=0;h<temp->nos;h++)
       {
                                        if(temp->seat[h]==k)
                                                {
                                           printf("0\t");
                                              k++;
                                            match=1;
                                             goto m;
                                              }
       }
    temp=temp->link;
   }
  m: temp=first;
   if(match==0)
   {
```

```
    printf("%d\t",k);
    k++;
   }
   match=0;
  }
  printf("\n");
 }
```

The Above code is used to display current seating arrangement, whenever a seat from the linked list is matched to the matrix containing all the seat numbers it is displayed as "0", this allows the user to understand that the seat has been booked and hence has to book a different seat.

b. Adding Data

[1]

```
void add()
{  struct node *temp,*temp2=first;
  int i,j;
  temp=(struct node*)malloc(sizeof(struct node));
  printf("Enter Full Name:\n");
  scanf("%s",temp->name);
  ll: printf("Enter No. of Seats:\n");
  scanf("%d",&temp->nos);
  if(temp->nos > (no_row*no_seat)-booked_seats)
  {
    printf("No of seats exceeds capacity please enter again\nseats available is
%d\n",(no_row*no_seat)-booked_seats);
    goto ll;
```

```
    }
```

The above code takes data from the user, and displays a message if the no. of seats being selected by the user exceeds the available seats.

[2]

```
 if(temp->nos==0)
    { printf("\nNUMBER CANT BE ZERO\n");
      goto ll;
    }
   printf("Enter seat nos.\n");
```

This Makes sure that number of seats isnt zerro.

[3]

```
 for(i=0;i<temp->nos;i++)
   {
     C: scanf("%d",&temp->seat[i]);
     if(temp->seat[i]>(no_row*no_seat))
       {
                        printf("Seat non existant\nEnter Again\n");
                                          goto C;
       }
     if(temp->seat[i]==0)
      {
       printf("NUMBER CANT BE ZERO\n Re Enter!!\n");
       goto C;
      }
```

The Above two if conditions rules out the possibility of a seat number not existing in the seating arrangement getting stored(i.e, greater than the max seat no. or zero)

```
while(temp2!=NULL)
    { for(j=0;j<temp2->nos;j++)
          {
              if(temp->seat[i]==temp2->seat[j])
             {
                printf("Seat Already Booked, Please Enter Different Seat!!\n");
                goto C;
              }
            }
         temp2=temp2->link;
    }
    temp2=first;
    booked_seats++;
}
```

The above code checks if the seat number entered by the user is already booked or not.

```
  temp->cost=price*temp->nos;
  clrscr();
  printf("Booking Succesful!!\nPay %d at the counter",temp->cost);
  temp->link=first;
  first=temp;
  getch();
 }
```

After all discrepencies are avoided, booking is succesful and the data is updated in the linked list, the cost is also calculated and displayed to the user.

3. Cancelation

[1]

```
void cancel()
 {
   struct node *temp=first;
   char Name[50];
   int y=1,cancel_amount=0,i,j,nus,*seats,flag=0;
   seats=(int *)calloc(nus,totseat);
   clrscr();
   printf("Enter any number to Continue\nPress 0 to go Back\n");
   scanf("%d",&y);
```

The above code initializes all the variables used for this function.

[2]

```
   while(y)
   {
   printf("Enter your Name (under which the ticket was previously
booked)\n");
   scanf("%s",Name);
   while(temp!=NULL)
   {
    if(strcmp(Name,temp->name)==0)
    {printf("Name Found, Booked tickets:\n");
     for(i=0;i<temp->nos;i++)
     {
      printf("%d\n",temp->seat[i]);
     }
     flag++;
```

```
    break;
    }
    temp=temp->link;
   }
  if(flag==0)
  {
   printf("Name not found please re enter!!\n");
   getch();
   cancel();
  }
```

The above code checks if the name entered by the user for cancelation of tickets already exists or not, if not then name has to be entered again or cancel option can be exited by entering "0", if the name is matched then the following code is executed.

[3]

```
  printf("Enter the no. of tickets to be Cancelled\n");
  u: scanf("%d",&nus);
  if(nus>temp->nos)
   {printf("No. of ticket to be cancelled cant be greater than tickets
booked\n");
    printf("Enter again\n");
    goto u;
   }
  if(nus==0)
  { printf("NUMBER CANT BE ZERO!! Re Enter!!\n");
    goto u;
   }
```

The above code checks for discrepencies such as seat number being entered being greater than the no of booked seats or being zero. Once the discrepencies has been eliminated following code is executed.

[4]

```
j=0;
flag=0;
printf("Enter Seat Numbers to Cancel\n");
for(i=0;i<nus;i++)
{
 mg: scanf("%d",seats);

 if(*seats==0)
 {
  printf("NO CANT BE ZERO!! RE ENTER!! \n");
  goto mg;
 }
```

***Checks for seat no. being zero, if yes then it has to be re-entered.**

```
 while(j<temp->nos)
 {
 if(*seats==temp->seat[j])
   {
  flag++;
  break;
   }
 j++;
 }
 if(flag!=0)
 {
```

```
      temp->seat[j]=0;
      booked_seats--;
      cancel_amount++;
      printf("Ticket succesfully cancelled\n");
      j=0;
      seats++;
     }
     if(flag==0)
     {
      printf("seat no. not found please re enter\n");
      j=0;
      goto mg;
     }
     flag=0;
    }
    temp->cost=(temp->cost)-(cancel_amount*price);
    clrscr();
    printf("Tickets are Cancelled\n Collect Rs%d\nPress Enter to go
Back",cancel_amount*price);
    getch();
    y=0;
   }


   booking();
     }
```

The Above code checks if the seat no. to be cancelled is available in the list of seats previously booked, if found then that seat is removed and a

success message is displayed and the user can now continue to delete if more seat has to be deleted or exit and return to the main menu.

If not found or any discrepencies occur(such as “0” being entered or non existant seat being entered) then the user is asked to re-enter proper seat numbers.

CHAPTER 6

CONCLUSION

This project has taught me a wider way to look at Linked Lists and exceeds the usage as its limited to our lab programs, It has given me a whole new insight on how pointers can be used to solve many problems.

This Project has provided me a great insight on problem solving and debugging because of its complex yet sipmle code and solutions.

There were many hurdles and limitations, But the joy of getting the output without any errors makes it feel worth it.

Finally I would like to conclude by saying that this project has helped me gain a wider knowledge and understning on how Linked list can be used, traversed in an out of the box way.

CHAPTER 7

BIBLIOGRAPHY

1. Wikibooks.org
2. C-Programming Techniques (PCD) – A.M.Padma Reddy
3. https://Stackoverflow.com/linked_lists#
4. https://Quora.in
5. htps://Hackerank.com/Data_structures/x0x#