

A Project Report
On
MOVIE RESERVATION SYSTEM

Submitted By

Annie Rana 16CSU049

Anshum Gill 16CSU052



(Formerly ITM University, Gurugram)

**Department of CSE & IT
The Northcap University
Gurugram**

A Project Report
On
MOVIE RESERVATION SYSTEM

Submitted in partial fulfillment of the requirement for the award of the degree

Of

**Bachelor of Technology
In Computer Science Engineering and Information Technology**

By

Annie Rana 16CSU049

Anshum Gill 16CSU052

Under the Supervision of
Ms. Kanika Gupta



(Formerly ITM University, Gurugram)

Department of CSE & IT

The Northcap University, Gurugram

April 2018

CERTIFICATE

This is to certify that the Project Report entitled, "Movie Ticket Reservation System" submitted by Annie Rana and Anshum Gill to The Northcap University, Gurugram, India, is a record of bonafide Project work carried out by him/her under my supervision and guidance and is worthy of consideration for the award of degree of Bachelor of Technology in Computer Science Engineering and Information Technology of the institute.

Date:

Supervisor Name

Designation

Acknowledgment

We would like to thank our teacher Kanika Gupta for providing us with this opportunity also for their guidance and constant supervision as well as providing necessary information regarding the project and also for their support in completing the project.

Annie Rana 16CSU049

Anshum Gill 16CSU052

Abstract

The movie ticket reservation system was developed in Python. GUI was made with the help of the tkinter module in Python and sqlite3 was used for backend database storage and modification.

The movie ticket reservation system eases the process of reserving a seat for a movie in the theater.

On the initial screen the movie names are presented on the left as buttons, upon clicking the desired movie name the poster, synopsis and the show times are shown in the right frame. The show times are also developed as buttons, upon clicking those buttons a new window opens which gives the user the opportunity to select a desired seat which are also buttons. On clicking the desired seat, the program checks if the seat is already reserved or not, if already reserved then the program will give a message informing user to try again, if the seat is available the program will input the necessary details and confirm the reservation prompting the user with his/her reservation ID.

Table of Contents

S.NO.	TOPIC	PAGE NUMBER
1	Certificate	3
2	Acknowledgement	4
3	Abstract	5
4	Introduction	7
5	Chapter 1: Tkinter	8
	• Introduction	8
	• Definition	8
	• Widgets	8
	• Geometry Managers	9
	• Windows	10
	• Organizing Using Paned Windows	10
6	Chapter 2: SQLite3	11
	• Introduction	11
	• Connection	11
	• Iterating through extracted data	11
7	Chapter 3: Program and Tables	12
	• Program	12
	• Table 'Movie'	16
	• Table 'Reservation'	16
	• Table 'Screen'	16
	• Table 'Seats'	17
	• Table 'Seats_reserved'	17
	• Table 'Show'	18
8	Output	19

Introduction

The Movie Ticket Reservation System allows users to pick their desired seats in a movie theater.

Upon running the program the window is divided in 3 paned windows. The top one contains the title of the program, the left one contains the movie names as well as the instructions on how to use the program. The third one and the largest one contains the all the synopsis for the selected movie, its title and poster, show times and a clear button. The clear button will clear that particular frame.

Once a movie name is clicked the program stores the selected movie name as a string and processes a query using the sqlite .execute function which then extracts all the information regarding the movie from the 'movie' table. The timings variable stores the show timings of the movie which is extracted from the 'show' table using the movieID in the form of a list of tuples.

When desired show timing is selected which were presented as buttons a new window opens up with the layout of the screen. Each screen has different number of seats and rows which is extracted from the table 'screen' which is printed out using a for loop.

When a desired seat is clicked on a new window opens up with a two entry fields for name and phone number along with all the information regarding the seat and the show. When the submit button is clicked a message box is displayed which informs the user of their reservation ID and confirms the reservation.

Chapter 1

Tkinter

Introduction

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk. Tkinter is not the only GUI Programming toolkit for Python. It is however the most commonly used one.

1.1 Definitions

- **Window** - This term has different meanings in different contexts, but in general it refers to a rectangular area somewhere on your display screen.
- **Top-level Window** - A window that exists independently on your screen. It will be decorated with the standard frame and controls for your system's desktop manager. You can move it around on your desktop. You can generally resize it, although your application can prevent this
- **Widget** - The generic term for any of the building blocks that make up an application in a graphical user interface. Examples of widgets: buttons, radiobuttons, text fields, frames, and text labels.
- **Frame** - In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets.
- **Child, Parent** - When any widget is created, a parent-child relationship is created. For example, if you place a text label inside a frame, the frame is the parent of the label.

1.2 Widgets

1.2.1 Frame

A frame is a widget that displays just as a simple rectangle. Frames are primarily used as a container for other widgets, which are under the control of a geometry manager such as grid, place, pack.

Example: `frame = ttk.Frame(parent)`

1.2.2 Label

A label is a widget that displays text or images, typically that the user will just view but not otherwise interact with. Labels are used for such things as identifying controls or other parts of the user interface, providing textual feedback or results, etc.

Example: `label = ttk.Label(parent, text='Full name:')`

1.2.3 Button

A button, unlike a frame or label, is very much designed for the user to interact with, and in particular, press to perform some action. Like labels, they can display text or images, but also have a whole range of new options used to control their behavior.

Example: `button = ttk.Button(parent, text='Okay', command=submitForm)`

1.2.4 Checkbutton

A checkbutton is like a regular button, except that not only can the user press it, which will invoke a command callback, but it also holds a binary value of some kind (i.e. a toggle). Checkbuttons are used all the time when a user is asked to choose between e.g. two different values for an option.

Example: `measureSystem = StringVar()`

```
check = ttk.Checkbutton(parent, text='Use Metric',
                        command=metricChanged, variable=measureSystem,
                        onvalue='metric', offvalue='imperial')
```


1.2.5 Radiobutton

A radiobutton lets you choose between one of a number of mutually exclusive choices; unlike a checkbox, it is not limited to just two choices. Radiobuttons are always used together in a set and are a good option when the number of choices is fairly small, e.g. 3-5.

```
Example: phone = StringVar()
        home = ttk.Radiobutton(parent, text='Home', variable=phone, value='home')
        office = ttk.Radiobutton(parent, text='Office', variable=phone,
value='office')
        cell = ttk.Radiobutton(parent, text='Mobile', variable=phone, value='cell')
```

1.2.6 Entry

An entry presents the user with a single line text field that they can use to type in a string value. These can be just about anything: their name, a city, a password, social security number, and so on.

```
Example: username = StringVar()
        name = ttk.Entry(parent, textvariable=username)
```

1.2.7 Combobox

A combobox combines an entry with a list of choices available to the user. This lets them either choose from a set of values you've provided (e.g. typical settings), but also put in their own value (e.g. for less common cases you don't want to include in the list).

```
Example: countryvar = StringVar()
        country = ttk.Combobox(parent, textvariable=countryvar)
```

1.3 Geometry Managers

1.3.1 Pack

Pack is the easiest to use of the three geometry managers of Tk and Tkinter. Instead of having to declare precisely where a widget should appear on the display screen, we can declare the positions of widgets with the pack command relative to each other. The pack command takes care of the details. Though the pack command is easier to use, this layout managers is limited in its possibilities compared to the grid and place managers. For simple applications it is definitely the manager of choice. For example, simple applications like placing a number of widgets side by side, or on top of each other.

1.3.2 Place

The Place geometry manager allows you explicitly set the position and size of a window, either in absolute terms, or relative to another window. The place manager can be accessed through the place method. It can be applied to all standard widgets.

1.3.3 Grid

The Grid geometry manager places the widgets in a 2-dimensional table, which consists of a number of rows and columns. The position of a widget is defined by a row and a column number. Widgets with the same column number and different row numbers will be above or below each other. Correspondingly, widgets with the same row number but different column numbers will be on the same "line" and will be beside of each other, i.e. to the left or the right.

1.4 Windows

We have already seen that all Tk programs start out with a root toplevel window, and then widgets are created as children of that root window. Creating new toplevel windows works almost exactly the same as creating new widgets.

Creating: `t = Toplevel(parent)`

Destroying: `t.destroy()`

Changing Window Behavior and Styles

1.4.1 Window Title

```
window.title('New title')
```

1.4.2 Size and Location

```
widthxheight±x±y
```

```
window.geometry('300x200-5+40')
```

1.4.3 Stacking Order

```
window.lift()
```

```
window.lift(otherwin)
```

```
window.lower()
```

```
window.lower(otherwin)
```

1.4.4 Resizing Behavior

```
window.resizable(FALSE, FALSE)
```

1.4.5 Iconifying and Withdrawing

```
thestate = window.state()
```

```
window.state('normal')
```

```
window.iconify()
```

```
window.deiconify()
```

1.5 Organizing using Paned Windows

A `panedwindow` widget lets you stack two or more resizable widgets above and below each other (or to the left and right). The user can adjust the relative heights (or widths) of each pane by dragging a sash located between them. Typically, the widgets you're adding to a `panedwindow` will be frames containing many other widgets.

```
p = ttk.Panedwindow(parent, orient=VERTICAL)
# first pane, which would get widgets gridded into it:
f1 = ttk.Labelframe(p, text='Pane1', width=100, height=100)
f2 = ttk.Labelframe(p, text='Pane2', width=100, height=100) # second pane
p.add(f1)
p.add(f2)
```

Chapter 2

SQLite3

Introduction

SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

2.1 Connection

To use the module, you must first create a Connection object that represents the database.

Example: `conn = sqlite3.connect('example.db')`

Once we have a Connection, we can create a Cursor object and call its `execute()` method to perform SQL commands.

Example: `c=conn.cursor()`

```
c.execute("SELECT * FROM table_name")
c.execute("INSERT INTO table_name VALUES(value1,value2....)")
c.commit()
c.close()
```

It is necessary to commit after any modifications to the table in order to save the changes.

2.2 Iterating through the extracted data

Whenever a SELECT query is executed the we have 3 options,

1. Use the cursor as iterator

Example: `for row in c.execute('SELECT * FROM stocks ORDER BY price'):`
`print(row)`

```
('2006-01-05', 'BUY', 'RHAT', 100, 35.14)
('2006-03-28', 'BUY', 'IBM', 1000, 45.0)
('2006-04-06', 'SELL', 'IBM', 500, 53.0)
('2006-04-05', 'BUY', 'MSFT', 1000, 72.0)
```

2. Use `.fetchone()`

Fetches the next row of a query result set, returning a single sequence, or None when no more data is available.

3. Use `.fetchall()`

Fetches all (remaining) rows of a query result, returning a list. Note that the cursor's `arraysize` attribute can affect the performance of this operation. An empty list is returned when no rows are available.

Chapter 3

Program and Tables

3.1 Program

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from sqlite3 import *
import uuid

conn=connect("movieticketreservation.db")
root=Tk()
root.title("Movie Ticket Reservation System")
root.geometry("960x540+200+50")
root.resizable(False,False)
style=ttk.Style()
style.configure('TFrame',background="#FFE4C4")
style.configure('TButton',background="#FFE4C4",font=15)
style.configure('TLabel',background="#FFE4C4",font=15)
style.configure('TPanedwindow',background="#FFE4C4")

pwin=ttk.Panedwindow(root,orient=VERTICAL)
pwin.pack(fill=BOTH)
pwindow=ttk.Panedwindow(root,orient=HORIZONTAL)
pwindow.pack(fill=BOTH)

info=ttk.Frame(pwin,width=960,height=140,relief=RIDGE)
info.pack()

movNames=ttk.Frame(pwindow,width=300,height=540,relief=RIDGE)
movNames.pack()
movNames.config(padding=(15,15))

showTime=ttk.Frame(pwindow,width=660,height=540,relief=RIDGE)
showTime.pack()

pwindow.add(movNames)
pwindow.add(showTime)

pwin.add(info)
pwin.add(pwindow)

infoLabel=ttk.Label(info,text="Movie Ticket Reservation System",font=35).pack()
```

```

def button_movieName(Mname):
    showTimes=ttk.Frame(showTime,width=800,height=540)
    showTimes.place(x=10,y=10)
    timings=conn.execute("SELECT ShowTime, ShowID FROM show as t1 INNER JOIN movie as
t2 ON t2.MovieID=t1.MovieID WHERE t2.Name=?", (Mname,)).fetchall()
    clear=ttk.Button(showTimes,text="Clear",command=
showTimes.destroy).place(x=350,y=450)
    MovieName=ttk.Label(showTimes,text=Mname).place(x=400,y=10)
    synopsis=conn.execute("SELECT Synopsis FROM movie WHERE
Name=?", (Mname,)).fetchone()
    syn=str(synopsis[0])
    pth=conn.execute("SELECT Poster FROM movie WHERE Name=?", (Mname,)).fetchone()
    path=str(pth[0])

MovieSyn=ttk.Label(showTimes,text=syn,wraplength=500,justify=CENTER).place(x=250,y=60)
    img=PhotoImage(file=path)
    Poster=ttk.Label(showTimes,image=img)
    Poster.image=img
    Poster.place(x=10,y=10)
    k=0
    for i in timings:
        item=i[0]
        showid=str(i[1])
        timeButton=ttk.Button(showTimes,text=item,command=lambda x=item,y=showid:
seatselect(x,y)).place(x=10+k,y=370)
        k+=120

def seatselect(time,showid):
    window=Toplevel(root)
    window.geometry("400x500+200+50")
    window.title("Select Your Seat")
    window.resizable(False,False)
    window.configure(background="#FFE4C4")
    movieid=conn.execute("SELECT MovieID FROM show WHERE
ShowID=?", (showid,)).fetchone()
    movieid=str(movieid[0])
    screen=conn.execute("SELECT ScreenID FROM show WHERE
ShowID=?", (showid,)).fetchone()
    screen=str(screen[0])
    seats=conn.execute("SELECT Seats FROM screen WHERE
ScreenID=?", (screen,)).fetchone()
    seats=int(seats[0])
    seatnums=conn.execute("SELECT SeatNum FROM seats WHERE
ScreenID=?", (screen,)).fetchall()
    seatnums=[i[0] for i in seatnums]
    cols=max(seatnums)
    screenlabel=ttk.Label(window,text="****Screen This
Way****").grid(row=0,column=0,columnspan=cols)
    j=0

```

```

    for i in range(1,cols+1):
        columnnum=ttk.Label(window,text=i).grid(row=1,column=j)
        j+=1
    j=0
    k=2
    for i in range(1,seats+1):
        seatButton=Button(window,text="| _ |",command= lambda a=(k-
1),b=(j+1),c=screen,d=showid,e=movieid,f=time:
seatidfinder(a,b,c,d,e,f),height=2,width=3).grid(row=k,column=j)
        if(i%cols==0):
            rownum=ttk.Label(window,text=(k-1)).grid(row=k,column=j+1)
            k+=1
            j-=(cols-1)
        else:
            j+=1

def seatidfinder(row,column,screen,showid,movieid,time):
    seatid=conn.execute("SELECT SeatID FROM seats WHERE Row=? AND SeatNum=? AND
ScreenID=?", (row,column,screen,)).fetchone()
    seatid=str(seatid[0])
    reserved=conn.execute("SELECT SeatID FROM seats_reserved WHERE
SeatID=?", (seatid,)).fetchone()
    if reserved is None:
        window=Toplevel(root)
        window.geometry("400x250+200+50")
        window.title("Enter Your Details")
        window.resizable(False,False)
        window.configure(background="#FFE4C4")
        def reciept():
            reservationid=str(uuid.uuid4())
            reservationid=reservationid.upper()
            reservationid=reservationid.replace("-","")
            reservationid=reservationid[0:10]
            addRow=conn.execute("INSERT INTO seats_reserved
VALUES(?,?,?,?)", (seatid,screen,movieid,showid,))
            addRow2=conn.execute("INSERT INTO reservation
VALUES(?,?,?,?)", (reservationid,nameEntry.get(),phoneEntry.get(),seatid,))
            conn.commit()
            messagebox.showinfo(title="Success", message="Your reservation
was successful, Thank You\nPlease make a note of your reservation ID: "+reservationid)
            window.destroy()

        movie=conn.execute("SELECT Name, RunTime, Language FROM movie WHERE
MovieID=?", (movieid,)).fetchone()
        showid=conn.execute("SELECT ShowID FROM show WHERE MovieID=? AND
ShowTime=?", (movieid,time,)).fetchone()

        showid=str(showid[0])

```

```

        moviename=str(movie[0])
        runtime=int(movie[1])
        lang=str(movie[2])

        nameEntry=StringVar(window)
        phoneEntry=StringVar(window)

        enter=ttk.Label(window,text="Please Enter your details:
").place(x=10,y=10)
        name=ttk.Label(window,text="Name-").place(x=10,y=30)
        nameEntry=ttk.Entry(window,width=24,textvariable=nameEntry)
        nameEntry.place(x=10,y=60)
        phn=ttk.Label(window,text="Phone Number-").place(x=200,y=30)
        phnEntry=ttk.Entry(window,width=24,textvariable=phoneEntry)
        phnEntry.place(x=200,y=60)
        info=ttk.Label(window,text="You have selected -").place(x=10,y=100)
        seatinfo=ttk.Label(window,text="Row: "+str(row)+" Seat Number:
"+str(column)+" of the "+str(time)+" show").place(x=10,y=130)
        movieinfo=ttk.Label(window,text="For: "+moviename+"\nRun Time:
"+str(runtime)+"\nLanguage: "+lang).place(x=10,y=160)

        submit=ttk.Button(window,text="Submit",command=reciept).place(x=150,y=220)
        else:
            messagebox.showinfo(title="Sorry", message="Seat alredy Reserved, Please
Try Again with another Seat")




        cursor=conn.execute("SELECT Name FROM movie")

        for i in cursor:
            item=i[0]
            button=ttk.Button(movNames, text=item,command=lambda x=item:
button_movieName(x)).grid()
        lbl=ttk.Label(movNames,text="1.Select Movie\n2.Select Show Time\n3.Select
Seat",wraplength=120,justify=CENTER).grid()

        root.mainloop()
        conn.close()




```

3.2 Table 'Movie'

Table:  movie  




	MovieID	Name	RunTime	Language	Synopsis	Poster
	Filter	Filter	Filter	Filter	Filter	Filter
1	BG02	Baaghi 2	145	Hindi	A battle-hard...	Posters\baag...
2	BLP01	Black Panther	134	English	After the even...	Posters\black...
3	HCH01	Hichki	116	Hindi	"Hichki" is a st...	Posters\hichki...
4	RAI01	Raid	128	Hindi	Raid is based ...	Posters\raid.gif
5	RGT03	Rangroot	142	Hindi	Set against th...	Posters\rangr...

3.3 Table 'Reservation'

Table:  reservation  

	ReservationID	Name	PhoneNumber	SeatID
	Filter	Filter	Filter	Filter
1	6FC3A8A482	Anshum	9811290033	BD46D
2	BF4DCA0560	anshum	788451	26B88
3	F03BCDF355	allwin	7011578115	53C60
4	5CE8F93853	Aman	7845120	4021F

3.4 Table 'Screen'

Table:  screen  

	ScreenID	Name	Seats
	Filter	Filter	Filter
1	FF01	Audi3	80
2	FF02	Audi4	50
3	GF01	Audi1	60
4	GF02	Audi2	100
5	SF01	Audi5	70

3.5 Table 'Seats'

Table: seats




	SeatID	Row	SeatNum	ScreenID
	Filter	Filter	Filter	Filter
267	8B216	8	7	GF02
268	0C6B8	8	8	GF02
269	E7DEF	8	9	GF02
270	621B0	8	10	GF02
271	7D2F5	9	1	GF02
272	4F88A	9	2	GF02
273	A32B3	9	3	GF02
274	F348A	9	4	GF02
275	084D7	9	5	GF02
276	F5421	9	6	GF02
277	C46BE	9	7	GF02
278	8EBF5	9	8	GF02
279	1CA55	9	9	GF02
280	4331F	9	10	GF02
281	BB463	10	1	GF02
282	DBD2B	10	2	GF02
283	4B7FE	10	3	GF02
284	78287	10	4	GF02

3.6 Table 'Seats_reserved'

Table: seats_reserved

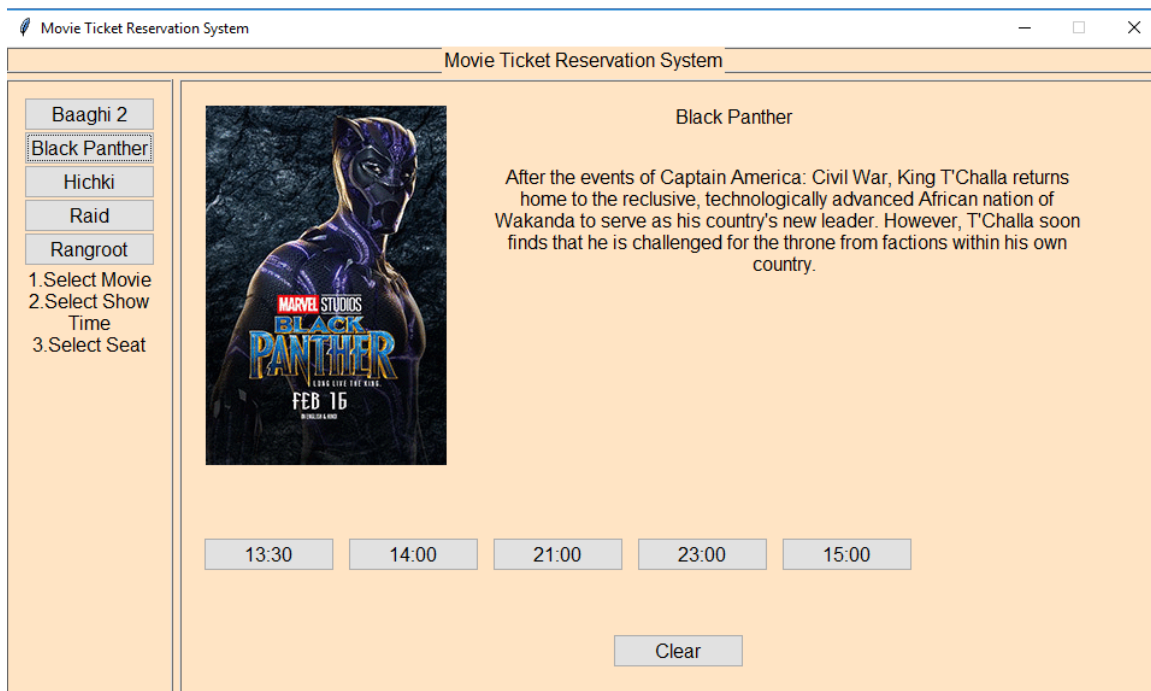
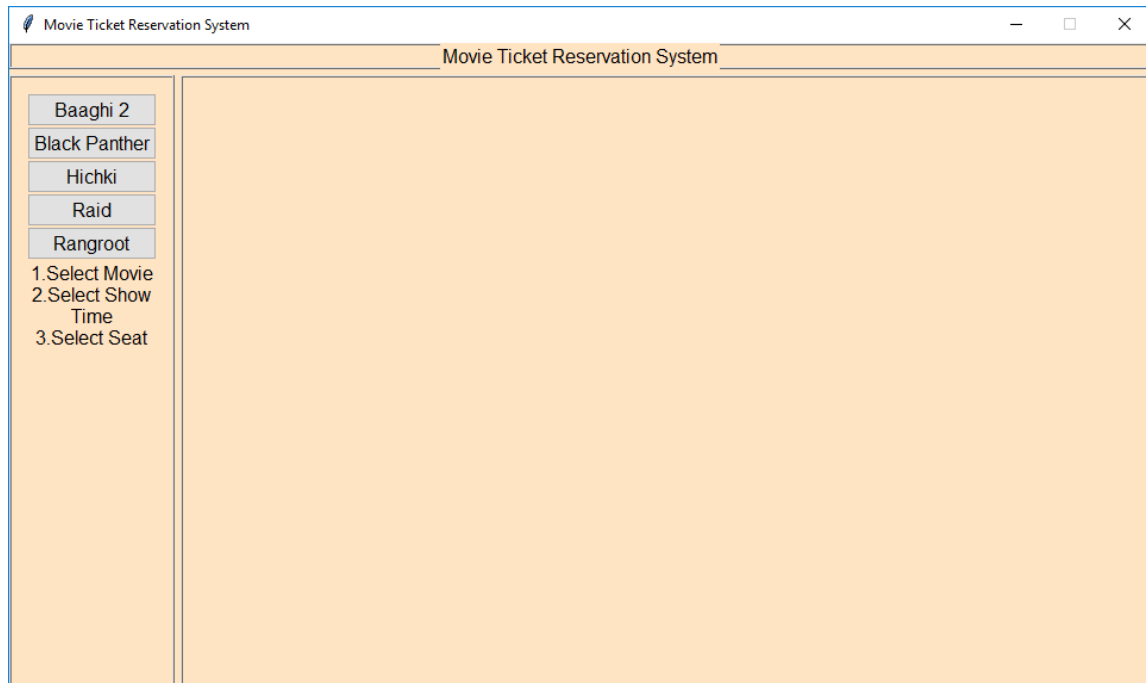
	SeatID	ScreenID	MovieID	ShowID
	Filter	Filter	Filter	Filter
1	BD46D	GF01	HCH01	36AC8
2	26B88	SF01	RGT03	A9BD0
3	53C60	GF01	RGT03	E16C8
4	4021F	GF01	BLP01	C0EB3

3.7 Table 'Show'

Table:  show  

	ShowID	ShowTime	MovieID	ScreenID
	Filter	Filter	Filter	Filter
1	7070C	09:30	BG02	FF01
2	363DD	11:00	BG02	FF02
3	73A3F	15:00	BG02	FF02
4	1876B	17:00	BG02	FF01
5	879C3	21:30	BG02	SF01
6	C0EB3	13:30	BLP01	GF01
7	20AE9	14:00	BLP01	SF01
8	C94D8	21:00	BLP01	FF02
9	2AC33	23:00	BLP01	FF01
10	6079B	15:00	BLP01	SF01
11	36AC8	09:30	HCH01	GF01
12	8C7F9	11:30	HCH01	SF01
13	7242A	14:30	HCH01	FF01
14	E1049	17:30	HCH01	SF01
15	0A10C	10:30	RAI01	GF02
16	DAE6B	14:30	RAI01	GF02
17	F219C	21:30	RAI01	GF02
18	C8D83	17:00	RAI01	FF01

Output



Select Your Seat

****Screen This Way****

1	2	3	4	5	
_	_	_	_	_	1
_	_	_	_	_	2
_	_	_	_	_	3
_	_	_	_	_	4
_	_	_	_	_	5
_	_	_	_	_	6
_	_	_	_	_	7
_	_	_	_	_	8
_	_	_	_	_	9
_	_	_	_	_	10

Enter Your Details

Please Enter your details:

Name- Phone Number-


You have selected -

Row: 7 Seat Number: 5 of the 21:00 show

For: Black Panther
Run Time: 134
Language: English

Submit

Success

 Your reservation was successful, Thank You
Please make a note of your reservation ID: 8665246385

OK