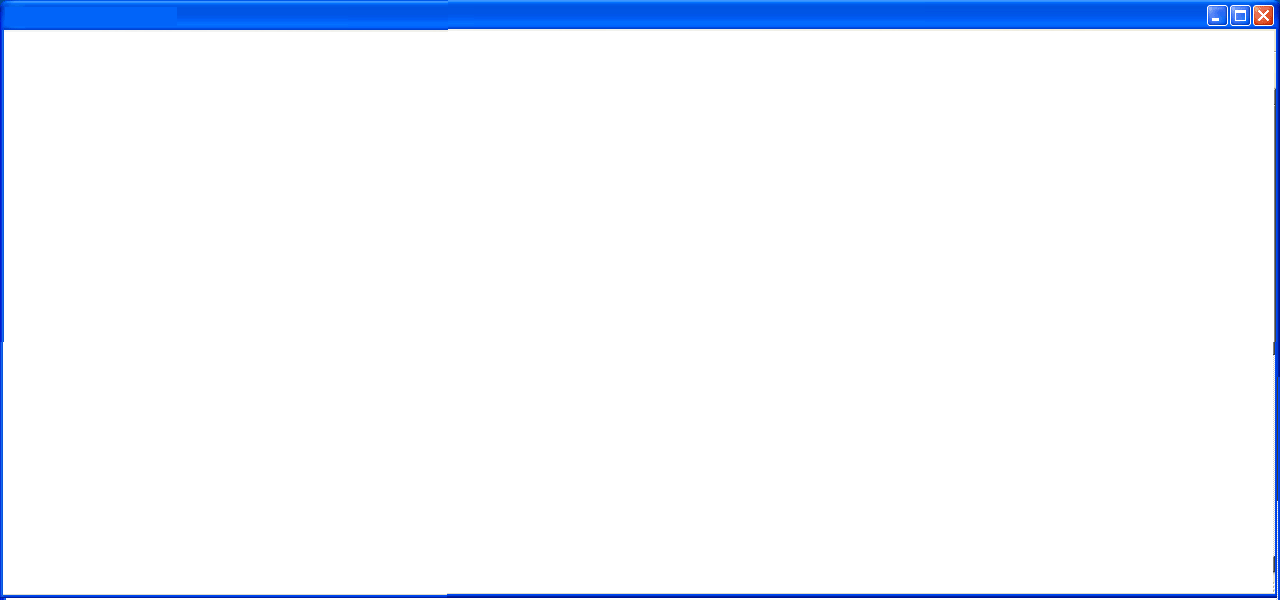


Gems millennium school

Sharjah



Computer

Project

**Name**

**XII-B1**

**2019-2020**

***Certificate Page***

***Acknowledgement***



***Contents***

01 Aim

02 User Documentation

03 User Defined Functions

04 Algorithms

05 Flow Charts

06 Source Code

07 Screen Shots

08 Bibliography

******

***AIM***

**The Main Aim Of This Project Is To Implement A Real Life Application Of Python And SQL To Create A User Friendly Intuitive Program For A Movie Booking System. This Program Allows The User To Book And Display The Details Of Each Booking. The Program Also Allows The User To Choose The Timings, Seats, Day And Date Of Each Booking. It Displays And Sends The Bill Via An Email Bill Containing All Details Along With The Price, Reference Number And Movie Name.**

******

***USER DOCUMENTATION***

1. **tkinter –**

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI.

In this program, we use 13 Tkinter module functions/ classes :

1. **title() -**

It is used to assign a title to the window.

1. **label() –**

The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

1. **button() –**

The Button widget is used to display buttons in your application.

1. **pack() –**

This geometry manager organizes widgets in blocks before placing them in the parent widget.

1. **grid() –**

This geometry manager organizes widgets in a table-like structure in the parent widget.

******

1. **mainloop() –**

It is used when you are ready for the application to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

1. **destroy() –**

Destroys the widget. The widget is removed from the screen, and all resources associated with the widget are released.

1. **optionmenu() –**

The OptionMenu class is a helper class that creates a popup menu, and a button to display it.

1. **set() –**

It is used to set and change the value stored within a tkinter variable.

1. **get() –**

It is used to fetch the current entry text.

1. **Tk() –**

The Tk class is instantiated without arguments. This creates a toplevel of Tk which usually is the main window of an application. Each instance has its own associated Tcl interpreter.

1. **stringvar() –**

A StringVar() is used most commonly to edit widget's text very quickly.

******

**m) photoimage() –**

The PhotoImage class is used to display images (either grayscale or true color images) in labels, buttons, canvases, and text widgets.

**2) mysql.connector –**

MySQL Connector/Python. MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification.

In this program, we use 5 mysql.connector module functions/ classes :

1. **connect() –**

The connect() constructor creates a connection to the MySQL server and returns a MySQLConnection object.

1. **is\_connected () –**

MySQLConnection.is\_connected() Method Re-ports whether the connection to MySQL Server is available.

1. **cursor() –**

Cursor objects interact with the MySQL server using a MySQLConnection object.

******

1. **execute() –**

It is used to execute SQL queries through Python.

1. **commit() –**

It is important to call this method after every transaction that modifies data for tables that use transactional storage engines.

**3) functools –**

The functools module is for higher-order functions: functions that act on or return other functions.

In this program, we use a mysql.connector module function/ class :

1. **partial() –**

Return a new partial object which when called will behave like func called with the positional arguments args and keyword arguments keywords.

**4) datetime –**

The datetime module supplies classes for manipulating dates and times in both simple and complex ways.

In this program, we use 2 datetime module functions/

***ALGORITHMS***

Step 1: Start

Step 2: Choose [Click to Enter] or [Exit]

If the choice is Click to Enter, go to STEP 3

If the choice is Exit, go to STEP 17

Step 3: Select the date

Step 4: Select the time

Step 5: Choose one of the four movies displayed and click [Book] under the selected movie

Step 6: Select the preferred seats

Step 7: Click [Proceed]

Step 8: Enter the mobile number

Step 9: Click [Enter]

If the mobile number pre-exists in the database, go to STEP 14

If the mobile number does not exist in the database, go to STEP 10

Step 10: Enter Name

******Step 11: Enter Address

Step 12: Enter E-mail

Step 13: Click [Submit]

Step 14: The receipt is displayed with Movie name, Date, Time, Seat(s), Total price and Reference number

Step 15: Click [Exit]

Step 16: An e-bill is sent to the entered email

Step 17: Stop

*** Booking a movie***

Start

Checks if the database exists or not and creates database accordingly

Option provided to start or exit

Stop

C1

Select Date, Time and Movie

1

******

1

2

An error window is displayed, further prompting the user to select seats accordingly

Proceed after selecting seats

Choose preferred seats, with restrictions on already booked ones

The SQL database is checked for any prior bookings, to avoid overlap of seats with current booking

******

2

Mobile number is collected from the user

The SQL database is checked for any record on the entered phone number

C2

A bill is displayed along with a referral number and the booking details

An e-mail address is received from the SQL database and an e-bill is sent

Stop

User receives

e-bill

***Dropdown for date and time***

C1

Receives the current date and time

Checks if the time is greater than the last show timing

If yes, the shows will be shown from next day

If not, displays show timings from the current date greater than the current time

***User validity***

A new record is created with name, address, e-mail, movie name, selected seats, date and time

If not, user is asked to enter Name, Address and E-mail address

C2

Checks if mobile number pre-exists or not in the SQL database

If yes, continue

***Source Code***

#All imported modules

from tkinter import \*

from functools import partial

from random import randint

from datetime import datetime, timedelta

import mysql.connector as mysql

import sys

import smtplib

#Creating/Accessing the database

db = mysql.connect(host = 'localhost', user = 'root', password = '', buffered = True)

if db.is\_connected():

dbc = db.cursor()

dbc.execute('show databases')

cursor = [i[0] for i in dbc]

****** if 'movie' in cursor:

dbc.execute('use movie')

else:

dbc.execute('create database movie')

dbc.execute('use movie')

dbc.execute('show tables')

cursor = [i[0] for i in dbc]

if 'movie' in cursor:

pass

else:

dbc.execute('create table movie(RefNum int primary key, Movie varchar(60), Seat varchar(500),'+

' Cost int, Date varchar(10), Time varchar(8),'+

' Name varchar(20), Address varchar(30), MobileNo varchar(20), EMail varchar(30))')

else:

print("MySQL connection failed")

### ADMIN VARIABLES ###

******nrow = 8

ncol = 14

cinema = 'MIG Cinemas, Sharjah'

screen1 = 'Avengers: Endgame'

screen2 = 'Fast and Furious Presents: Hobbs and Shaw'

screen3 = 'Spider-Man: Into the Spider-Verse'

screen4 = 'Joker'

screen = [screen1, screen2, screen3, screen4]

#####

#Option menu (Time)

def change(event):

g = variable.get()

if g != dates[0]:

s = OptionMenu(mov, var, \*times)

var.set(times[0]) ## Default

else:

s = OptionMenu(mov, var, \*times[deft:])

****** var.set(times[deft])

s.grid(column = 1, row = 1, columnspan = 2)

#Welcome screen

stat=False

def exi():

wel.destroy()

sys.exit()

def contin():

global stat

stat=True

wel.destroy()

wel=Tk()

wel.title("Welcome")

welimg = PhotoImage(file = 'Wel.gif')

******Label(wel, image=welimg).grid(row=0, column=0)

Button(wel, text = 'Click to enter', command=contin).grid(column=0, row=0, sticky = S)

Label(wel, text="").grid(row=1, column=0)

Button(wel, text = 'Exit', command=exi).grid(column=0, row=2)

wel.mainloop()

if stat==False:

sys.exit()

mov = Tk()

mov.title("Movie selection")

k = datetime.now()

times = ['12:00 AM', '9:00 AM', '11:30 AM', '2:00 PM', '4:30 PM', '7:00 PM', '9:30 PM']

******# Drop down box for time

var = StringVar(mov)

hr = int(str(k)[11:13])

mi = int(str(k)[14:16])

### Below block of code assigns the default time based on the time of execution of the program

if hr > 12:

hr %= 12

if hr > 9 or (hr == 9 and mi >= 30):

deft = 0

elif hr >= 7 and mi >= 0:

deft = -1

elif hr > 4 or (hr == 4 and mi >= 30):

deft = -2

elif hr >= 2 and mi >= 0:

deft = -3

else:

deft = -4

****** hr += 12

else:

if hr < 9:

deft = 1

elif (hr > 11) or (hr == 11 and mi >= 30):

deft = 3

else:

deft = 2

## Last show is at 9, so if the time is past it then the booking starts from the next day

if deft == 0:

k = datetime.now()+timedelta(days=1)

dates = []

for i in range(7):

dates.append(str(k)[:11])

k+=timedelta(days=1)

******# Drop down box for dates

variable = StringVar(mov)

variable.set(dates[0]) ## Default

w = OptionMenu(mov, variable, \*dates, command = change)

Label(mov, text = 'Select the date: ', font = 'Times 10').grid(row = 0, column = 0)

w.grid(column = 1, row = 0, columnspan = 2)

Label(mov, text = 'Select the time: ', font = 'Times 10').grid(row = 1, column = 0)

s = OptionMenu(mov, var, \*times[deft:])

var.set(times[deft]) ## Default

s.grid(column = 1, row = 1, columnspan = 2)

def closemov():

mov.destroy()

mov.protocol("WM\_DELETE\_WINDOW", closemov)

******

### Imports images to variables screen(1,4)img

for i in screen:

globals()['screen'+str(screen.index(i)+1)+'img'] = PhotoImage(file = i[:3]+'.gif')

#####

K=[(screen1, screen1img), (screen2, screen2img), (screen3, screen3img), (screen4, screen4img)]

J = ['K['+str(i)+'][0]' for i in range(4)]

## Function takes argument and changes moviename, time and date based on the button pressed

def book(f):

global date, moviename, poster, time ## is clicked

moviename = eval(f)

date = variable.get()

mov.destroy()

time = var.get()

******

j = 0

for i in [3, 20]: #Coordinates for placing the posters in the mov window

argument = partial(book, J[j]) # function from another module to pass arguments into the entered function

Label(mov, image = K[j][1]).grid(row = i, columnspan = 10, sticky = W)

Label(mov, text=K[j][0]).grid(row = i+1)

globals()[J[j]] = Button(mov, text = 'Book', command = argument).grid(row = i+2)

Label(mov, image = K[j+1][1]).grid(row = i, column = 30, columnspan = 50)

Label(mov, text = K[j+1][0]).grid(row = i+1, column = 30, columnspan = 50)

argument = partial(book, J[j+1])

globals()[J[j]] = Button(mov, text = 'Book', command = argument).grid(row = i+2, column = 30, columnspan = 50)

j+=2

mov.mainloop()

******

win = Tk()

win.title("Seat selection")

def closewin():

win.destroy()

win.protocol("WM\_DELETE\_WINDOW", closewin)

#Image files for seat selection screen

screen = PhotoImage(file = 'screen.png')

cblue = PhotoImage(file = 'Screenshot\_1.png')

cred = PhotoImage(file = 'Screenshot\_2.png')

cX = PhotoImage(file = 'Screenshot\_3.png')

poster = PhotoImage(file = moviename[:3]+'.gif')

screen1 = Label(win, image = screen)

screen1.grid(columnspan = 25, rowspan = 15, sticky = N)

******Label(win, image = poster).grid(column = 26, columnspan = 30, rowspan = 10, row = 0)

Z=[]

tkrow = 21

for i in range(nrow):

row=[[tkrow, z+4, 'blue', chr(65+i)+str(z)] for z in range(1, ncol+1)]

Z.append(row)

tkrow += 1

col = 26

#Movie details

Q = [["Movie selected:", 11], ["Date and Time:", 13], ["Selected seat(s):", 16], [moviename, 12],

[date+", "+time, 14]]

for i in Q:

****** Label(win, text = i[0], font = "Times 10").grid(column = 26, row = i[1], columnspan = 28, sticky = W)

seat = []

seatselected = Label(win, text = '', font = "Times 10")

seatselected.grid(column = 26, row = 17, columnspan = 28, sticky = W)

### Returns a string containing all the booked seats

def seat\_display(seats):

strseat = ''

d = {}

for i in seat:

if i[0] not in d:

d[i[0]] = [int(i[1:])]

else:

d[i[0]].append(int(i[1:]))

for i in d:

d[i].sort()

strs = ''

****** for i in d:

k = d[i]

k.sort()

for j in k:

strs += i+str(j)+', '

return(strs[:-2])

### Function that changes the color of seats upon being selected

def color(j, q, p):

global cash, col, seat, seatselected

if p[q][2] == 'blue':

globals()[j].config(image = cred)

p[q][2] = 'red'

cash += 15

col += 1

seat.append(p[q][3])

elif p[q][2] == 'red':

globals()[j].configure(image = cblue)

****** p[q][2] = 'blue'

cash -= 15

col -= 1

seat.remove(p[q][3])

price.config(text = "Price Total: $"+str(cash))

R = seat\_display(seat)

seatselected.configure(text = R)

cash = 0

price = Label(win, text = "Price Total: $"+str(cash), font = "Times 10")

price.grid(column = 26, row = 15, columnspan = 28, sticky = W)

rcoord = [chr(i) for i in range(65, 65+26)]

rco = rcoord[:nrow+1]

cr = 0

### Below block of code returns the previously booked seats in order to prevent the current user to book them.

******dbc.execute("select seat from movie where Time='{}' and Date='{}' and movie = '{}'".format(time.strip(), date.strip(), moviename.strip()))

L = list(dbc)

bs = [i[0] for i in L]

booked\_seats = []

for i in bs:

temp = i.split(', ')

for j in temp:

booked\_seats.append(j)

for k in Z:

for i in k:

arg = partial(color, i[3], k.index(i), k)

if i[3] in booked\_seats:

globals()[i[3]] = Button(win, command = arg, state = 'disabled')

globals()[i[3]].config(image = cX)

else:

****** globals()[i[3]] = Button(win, command = arg)

globals()[i[3]].config(image = cblue)

globals()[i[3]].grid(row = i[0], column = i[1])

Label(win, text = k.index(i)+1).grid(row = 21+nrow, column = i[1])

Label(win, text = rco[cr]).grid(row = i[0], column = i[1]+1)

cr+=1

###

#Reference number generator

refno = randint(1000000, 9999999)

### Functions used to make sure that the entered characters are valid.

def on\_write(\*args):

global e1v, e2v

s1, s2 = e1v.get(), e2v.get()

if len(s1) > 2:

e1v.set(s1[:2])

****** for i in s1:

if not i.isdigit():

e1v.set(s1[:-1])

if len(s2) > 7:

e2v.set(s2[:7])

for i in s2:

if not i.isdigit():

e2v.set(s2[:s2.index(i)])

def det\_write(\*args):

global name, email

n=name.get()

e=email.get()

for i in n:

if i.isspace():

continue

elif not i.isalpha():

****** name.set(n[:n.index(i)])

for i in e:

if i in '@.\_-':

continue

elif not i.isalpha() and not i.isdigit():

email.set(e[:e.index(i)])

######

def error(t):

'''Prints an error message with the dialogue as the entered parameter'''

error = Tk()

error.title("Error")

Label(error, text = t, font = "Times 10").pack()

Button(error, text = "Okay", command = error.destroy).pack()

### Function that adds the entered data to the user database in MySQL.

def submit():

****** global detail, name, address, email, dbc, con, e

#SQL queries to insert user data

n=name.get()

a=address.get()

e=email.get()

if len(n)>0 and len(a)>0 and len(e)>=6:

con = True

dbc.execute("insert into movie values({}, '{}', '{}', {}, '{}', '{}', '{}', '{}', '{}', '{}')".format(refno, moviename,

seat\_display(seat), cash, date, time, n, a, phno, e))

db.commit()

detail.destroy()

else:

error("Please re-check your details and ensure they're correct.")

******### Function that asks the user to input Name, Address and Email.

def detail\_num():

global detail, name, address, email, dbc, phno, con, e

phno = '+971 '+str(e1v.get())+' '+str(e2v.get())

if len(str(e1v.get())+str(e2v.get()))<9:

error('Please enter valid phone number')

else:

name = StringVar()

address = StringVar()

email = StringVar()

dbc.execute("select mobileno from movie")

cur = list(dbc)

cur = [i[0] for i in cur]

if phno in cur:

con = True

detail.destroy()

****** dbc.execute("select name, address, email from movie where mobileno='{}'".format(phno))

L = list(dbc)

L = [i for i in L[0]]

dbc.execute("insert into movie values({}, '{}', '{}', {}, '{}', '{}', '{}', '{}', '{}', '{}')".format(refno,

moviename, seat\_display(seat), cash, date, time, L[0], L[1], phno, L[2]))

db.commit()

e = L[2]

else:

Button(detail, text = 'Enter', state = 'disabled').grid(column=0, row=2, sticky = E, columnspan = 1)

Label(detail, text='Name: ').grid(column = 0, row = 3, sticky = E)

Entry(detail, width = 20, textvariable = name).grid(column = 1, row = 3)

****** Label(detail, text='Address: ').grid(column = 0, row = 4, sticky = E)

Entry(detail, width = 20, textvariable = address).grid(column = 1, row = 4)

Label(detail, text='E-mail: ').grid(column = 0, row = 5, sticky = E)

Entry(detail, width = 20, textvariable = email).grid(column = 1, row = 5)

Button(detail, text = 'Submit',command=submit).grid(column=0, row=6, sticky = E)

name.trace\_variable('w', det\_write)

email.trace\_variable('w', det\_write)

### Function that asks the user to enter his phone number.

def details():

global detail, on\_write, e1v, e2v

detail = Tk()

****** detail.title('Details')

e1v, e2v = StringVar(), StringVar()

e1v.trace\_variable('w', on\_write)

e2v.trace\_variable('w', on\_write)

Label(detail, text='Mobile number: +971-').grid(column = 0, row = 1)

Entry(detail, width = 2, textvariable = e1v).grid(column = 1, row = 1)

Label(detail, text='-').grid(column = 2, row = 1)

Entry(detail, width = 7, textvariable = e2v).grid(column = 3, row = 1)

Button(detail, text = 'Enter', command = detail\_num).grid(column=0, row=2, sticky = E, columnspan = 1)

****** detail.mainloop()

### Function that prints the bill

def proceed():

global billwin, con

if cash == 0:

error("No seats are selected. Please do so before proceeding further.")

else:

win.destroy()

con = False

details()

if con == True:

billwin = Tk()

billwin.title("Receipt:")

******

billwininfo = [('Receipt', '', 'Times 14'), ('Movie: ', moviename, 'Times 11'), ('Date: ', date, 'Times 11'),

('Time: ', time, 'Times 11'), ('Cinema: ', cinema, 'Times 11'),

('Seat(s): ', seat\_display(seat), 'Times 11'), ('Price Total: $', str(cash), 'Times 11'),

('Ref No: #', str(refno), 'Times 11'), ('ENJOY!', '', 'Times 13')]

j=0

for i in billwininfo:

Label(billwin, text=i[0]+i[1], font=i[2]).grid(row=j)

j+=1

Button(billwin, text='Exit', command=closebill).grid(row=j+1)

billwin.mainloop()

einfo=''

****** for i in billwininfo:

einfo += i[0]+i[1]+'\n'

info = 'Subject: {}\n\n{}'.format('Your Booking - MIG Cinemas', einfo)

server = smtplib.SMTP\_SSL('smtp.gmail.com', 465)

server.login("migcinemas@gmail.com", "migcinemas123")

server.sendmail("migcinemas@gmail.com", e, info)

server.quit()

else:

error("Server crashed")

def closebill():

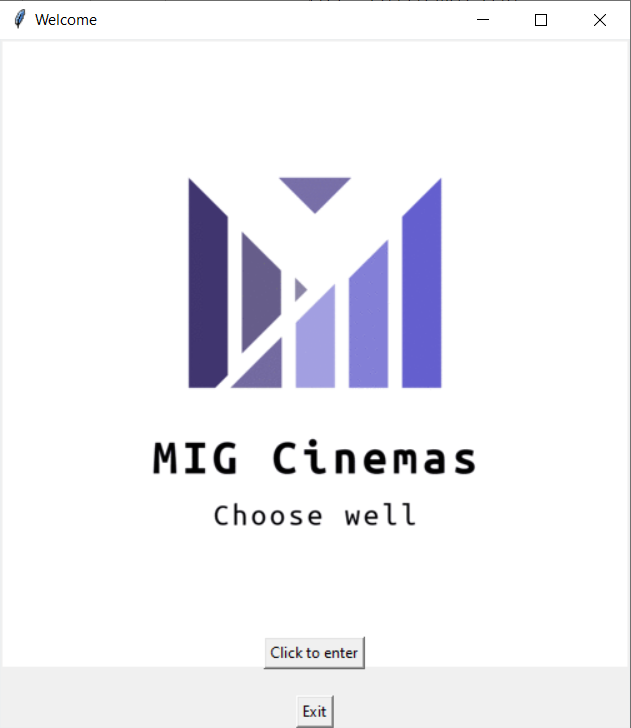
billwin.destroy()

Button(win, text = "Proceed", command = proceed).grid(column = 26, row = 20, columnspan = 28, sticky = W)

win.mainloop()

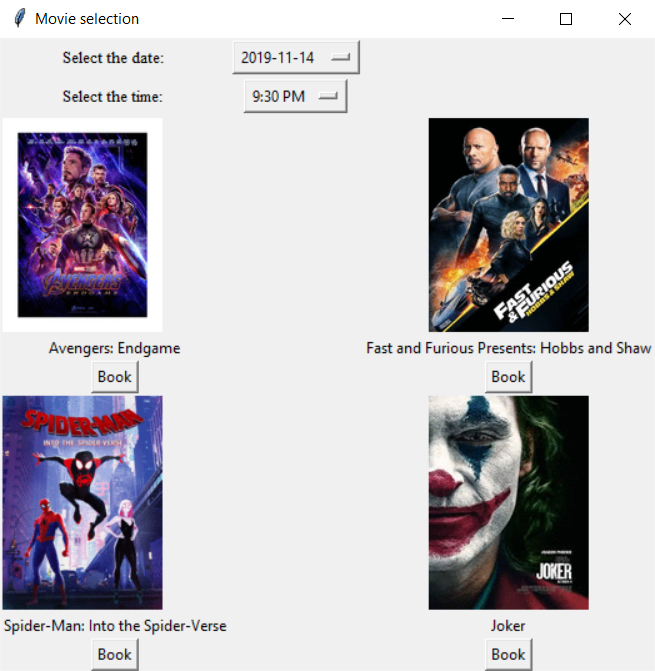
******

***Screen Shots***



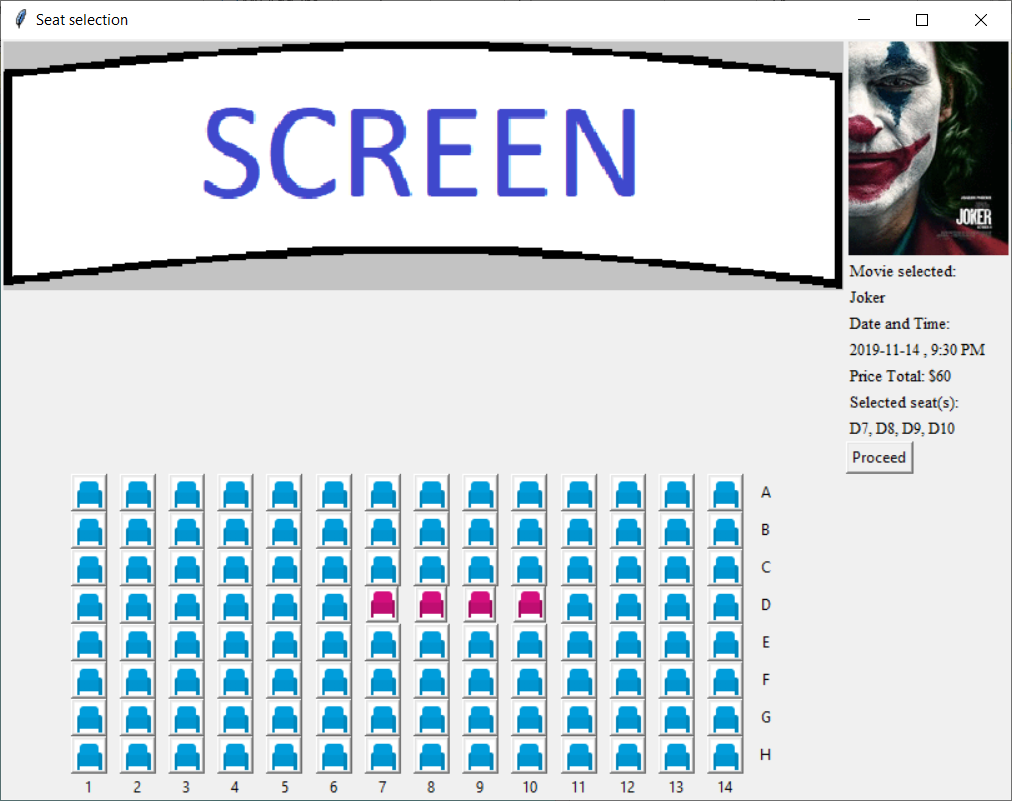
******

***Screen Shots***



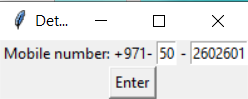
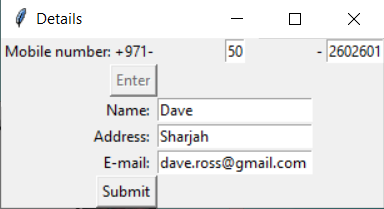
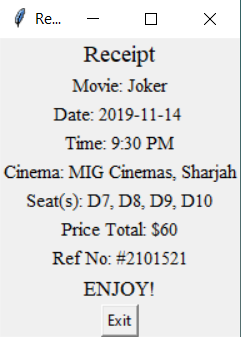
******

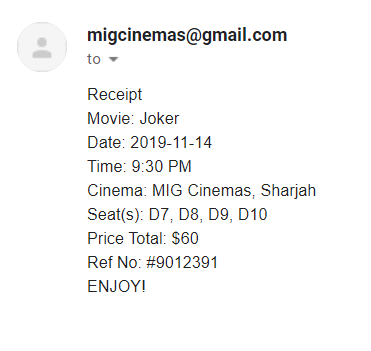
***Screen Shots***



******

***Screen Shots***



******

***Screen Shot of the E-mail***

***BIBLIOGRAPHY***

All definitions and user documentation source

-<https://docs.python.org/>

Tkinter source/YouTube tutorial

-<https://www.youtube.com/watch?v=_lSNIrR1nZU>

Other sources

-<https://stackoverflow.com/>