INFORMATICS PRACTICES PROJECT

2020-21

 

***‘LIGHTS, CAMERA, AND POPCORN’!***

An Authentic Experience :)

NAME = MOHSIN MURTAZA SIDDIQUI

CLASS = XII B SCIENCE

CBSE ROLL NUMBER = \*\*\*\*9898

INDEX

|  |  |  |
| --- | --- | --- |
| SERIAL NO. | DESCRIPTION | PAGE NO. |
| 1. | Introduction | 3 |
| 2. | Objective | 4 |
| 3. | Development Series | 5 |
| 4. | Flowchart | 6 |
| 5. | Hardware/Software Requirements | 7 |
| 6. | Bibliography | 8 |
| 7. | Coding | 9 |
| 8A. | SQL | 9 |
| 8B. | Python | 13 |

INTRODUCTION

Cinema- An early 20th Century word, derived from the French word *cinématographe,* is the 21st Century synonym for somewhere to go when there is a need to hang out with family, friends, and experience the many emotions one experiences when watching a film at a movie theatre.

The ‘Lights, Camera and Popcorn!’ application is here to make that entire experience easier for you; with a user friendly-interface, simple set-up instructions, easy-to-access information about currently on-show movies, options to cancel if necessary, and more features. A more detailed explanation is available in the later pages of this document.

OBJECTIVE

The objective of this project is to let the students apply the programming knowledge into real- world problems and will help expose the students to how programming skills help in developing good software.

1. Write programs utilizing modern software tools.

2. Apply object-oriented programming principles effectively,

3. Write effective procedural code to solve problems.

4. Demonstrate a breadth of knowledge in Python and MySQL, as exemplified in the areas of systems, theory and software development.

5. Students will demonstrate ability to conduct a research or applied Informatics Practices project, requiring writing and presentation skills which exemplify scholarly style in Informatics Practices.

DEVELOPMENT SERIES

Following were the steps implemented during the design of this project:

* ANALYSIS

Identifying an opportunity to improve the way of doing business of an organization that would use such a program; Identifying significant assumptions and constraints on solutions to figuring out how to implement that way.

* PLANNING

Deciding how to go about designing the interface; features; and the practicality of the project; figuring out how much percentage of my time should be devoted to what part of the program, singling out ways to improve efficiency in code writing, and implementing those effectively.

* DESIGN

Carefully overseeing the design of the program so as to ensure efficiency of sorting out issues, managing each module so as to ensure program design is streamlined and easy-to-understand, for any future analysts of the program.

* IMPLEMENTATION

Writing/typing the coding.

* SUPPORT

Asking for solutions i.e. help, from the project supervisor, in this case being the school teacher, as well as from online sources such as webpages & documents. Offline sources such as guide books were also consulted to a certain degree.

FLOWCHART

OPEN PROGRAM

YOUR ACCOUNT CONTINUE NO APPLICATION

IS REGISTERED TO APP? CLOSES

YES

REGISTER

ENTER REGISTRATION LOGIN OR REGISTER?

DETAILS

SIGN UP

LOGIN

MOVIE INFORMATION ENTER LOGIN DETAILS

SUCCESSFUL LOGIN

MOVIES ON-SHOW HOME SCREEN BUY TICKETS

ABOUT US VIEW PREVIOUS PURCHASES/ LOG-OUT

CANCEL PURCHASE

VIEW PREVIOUS PURCHASE CANCEL PURCHASE

VIEW PREVIOUS CANCEL PURCHASES

PURCHASES SCREEN SCREEN

END PROGRAM

HARDWARE & SOFTWARE REQUIREMENTS

HARDWARE

* PROCESSOR: PENTIUM(ANY) OR AMD ATHALON(3800+- 4200+ DUAL CORE)
* RAM: 512MB+
* HARD DISK : SATA 40 GB OR ABOVE
* MONITOR, KEYBOARD, MOUSE AS REQUIRED

SOFTWARE

* OPERATING SYSTEM: WINDOWS 7 OR LATER
* PYTHON 3.5.7 OR LATER
* SQL 8.0 OR NEWER

BIBLIOGRAPHY

LIST OF WEBSITES

* https://www.wikipedia.org
* <https://www.python.org/>
* https://python.mykvs.in/
* https://stackoverflow.com/
* https://www.w3schools.com/
* https://www.dreamincode.net/
* https://www.dbader.org/
* https://www.geeksforgeeks.org/
* https://www.dev.mysql.com/
* https://www.semicolonworld.com
* https://www.techbeamers.com
* https://www.python-course.eu/
* https://www.tutorialspoint.com/
* https://www.forums.mysql.com

LIST OF BOOKS

* INFORMATICS PRACTICES *BY SUMITA ARORA*

CODING

SQL

-- MySQL dump 10.13 Distrib 8.0.21, for Win64 (x86\_64)

--

-- Host: localhost Database: ipproject

-- ------------------------------------------------------

-- Server version 8.0.21

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8mb4 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `movies`

--

DROP TABLE IF EXISTS `movies`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `movies` (

`Movie` varchar(50) DEFAULT NULL,

`Price` smallint DEFAULT NULL,

`ShowTimings` varchar(255) DEFAULT NULL,

`NUMBER` int NOT NULL,

PRIMARY KEY (`NUMBER`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `movies`

--

LOCK TABLES `movies` WRITE;

/\*!40000 ALTER TABLE `movies` DISABLE KEYS \*/;

INSERT INTO `movies` VALUES ('007: No Time To Die',12,'3:00 PM',1),('Black Widow',12,'4:00 PM',2),('Venom: Let there be Carnage',12,'5:00 PM',3),('Mission Impossible: 07',11,'6:00 PM',4),('Wonder Woman 1984',13,'7:00 PM',5),('The Batman',13,'8:00 PM',6),('Marvel\'s Eternals',12,'9:00 PM',7),('Mulan',14,'10:00 PM',8),('FlashPoint',10,'10:45 PM',9),('Suicide Squad 2',13,'11:30 PM',10),('UnCharted',15,'12:15 AM',11),('Crimes Of Grindelwald',16,'1:00 AM',12),('Jurassic World: Dominion',12,'1:45 AM',13);

/\*!40000 ALTER TABLE `movies` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `ticketssold`

--

DROP TABLE IF EXISTS `ticketssold`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `ticketssold` (

`Email` varchar(255) DEFAULT NULL,

`MovieNumber` varchar(255) DEFAULT NULL,

`DateOfShowing` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `ticketssold`

--

LOCK TABLES `ticketssold` WRITE;

/\*!40000 ALTER TABLE `ticketssold` DISABLE KEYS \*/;

INSERT INTO `ticketssold` VALUES ('mohsinsiddiqui002@gmail.com','3','2020-12-25'),('mohsinsiddiqui002@gmail.com','3','2020-12-23'),('mohsinsiddiqui002@gmail.com','12','2021-01-12'),('mohsinsiddiqui002@gmail.com','9','2021-01-10'),('mohsinsiddiqui002@gmail.com','3','2020-11-09');

/\*!40000 ALTER TABLE `ticketssold` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `users`

--

DROP TABLE IF EXISTS `users`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `users` (

`Email` varchar(100) DEFAULT NULL,

`Name` varchar(50) DEFAULT NULL,

`Password` varchar(20) DEFAULT NULL,

UNIQUE KEY `Email` (`Email`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `users`

--

LOCK TABLES `users` WRITE;

/\*!40000 ALTER TABLE `users` DISABLE KEYS \*/;

INSERT INTO `users` VALUES ('mohsinsiddiqui002@gmail.com','Mohsin Siddiqui','Mohsin123'),('testing@gmail.com','Mohsin Siddiqui','password1'),('mohsin@gmail.com','','Mohsin2323'),('mohsin10@gmail.com','Mohsin Siddiqui','mohsin342');

/\*!40000 ALTER TABLE `users` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `yourmovie`

--

DROP TABLE IF EXISTS `yourmovie`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `yourmovie` (

`Movie` varchar(50) DEFAULT NULL,

`Cast` varchar(255) DEFAULT NULL,

`Director` varchar(255) DEFAULT NULL,

`NUMBER` int NOT NULL,

PRIMARY KEY (`NUMBER`),

CONSTRAINT `yourmovie\_ibfk\_1` FOREIGN KEY (`NUMBER`) REFERENCES `movies` (`NUMBER`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `yourmovie`

--

LOCK TABLES `yourmovie` WRITE;

/\*!40000 ALTER TABLE `yourmovie` DISABLE KEYS \*/;

INSERT INTO `yourmovie` VALUES ('007: No Time To Die','Daniel Craig, Rami Malek, Ana De Armas','Cary JF',1),('Black Widow','Scarlett Johansson, Florence Pugh, Rachel Weisz','Cate Shortland',2),('Venom: Let there be Carnage','Tom Hardy, Woody Harrelson, Michelle Williams','Andy Serkis',3),('Mission Impossible: 07','Tom Cruise, Hayley Atwell, Vanessa Kirby','Christopher McQuarrie',4),('Wonder Woman 1984','Pedro Pascal, Gal Gadot, Connie Nielsen','Patty Jenkins',5),('The Batman','Robert Pattinson, Zoë Kravitz, Paul Dano','Matt Reeves',6),('Marvel\'s Eternals','Angelina Jolie, Richard Madden, Kumail Nanjiani','Chloé Zhao',7),('Mulan','Yifei Liu, Donnie Yen, Tzi Ma','Niki Caro',8),('FlashPoint','Ezra Miller, tba','Andrés Muschietti',9),('Suicide Squad 2','Margot Robbie, John Cena, Idris Elba','James Gunn',10),('UnCharted','Tom Holland, Mark Wahlberg, Antonio Banderas','Ruben Fleischer',11),('Crimes Of Grindelwald','Eddie Redmayne, Johnny Depp, Katherine Waterston','David Yates',12),('Jurassic World: Dominion','Bryce Dallas Howard, Chris Pratt, Jake Johnson','Colin Trevorrow',13);

/\*!40000 ALTER TABLE `yourmovie` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

PYTHON

**1) IMPORTING RELEVANT PACKAGES**

from tkinter import \*

from tkinter import messagebox as msgbox

import mysql.connector as mycon

import pandas as pd

import tkinter as tk

con1 = mycon.connect(host="localhost", user="root", password="1234", database="ipproject")

if con1.is\_connected():

print("Connected")

**2) WINDOW**

**CODING TO DISPLAY WELCOME PICTURE + LOGO OF 'LIGHTS, CAMERA AND POPCORN' APP**

if con1.is\_connected():

welcomewindow = Tk()

welcomewindow.title("Welcome to Lights Camera Popcorn!")

welcomewindow.configure(background="black")

welcomephoto=PhotoImage(file= "C:\\Users\\mohsi\\Desktop\\welcomeimage.png")

Label (welcomewindow, image=welcomephoto, bg ="black") .grid(row=0, column=0, sticky=W)

**3) WINDOW**

**CODING TO OPEN THE SCREEN FROM WHERE TO CHOOSE BETWEEN LOGGING-IN, IN CASE OF BEING AN OLD USER, AND REGISTERING A NEW ACCOUNT, IN CASE OF A NEW USER.**

def openloginorregisterscreen():

startappwindow.destroy()

global loginorregisterscreen

loginorregisterscreen = Toplevel()

loginorregisterscreen.title('Welcome To the Login/Register Screen!')

loginorregisterscreen.geometry("900x500")

loginorregisterscreen.configure(background="black")

l2 = Label(loginorregisterscreen, text="OLD USER", bg='firebrick4', fg='grey99', font='fixedsys',width = 23)

l2.place(x=190,y=175)

l3 = Label(loginorregisterscreen, text="NEW USER", bg='firebrick4', fg='grey99',font='fixedsys',width = 23)

l3.place(x=410,y=175)

LogIn = Button(loginorregisterscreen,text = "Log-In",bg='medium blue',font='fixedsys',fg='white',width =23, command=openloginwindow)

LogIn.place(x=190,y=225)

Register = Button(loginorregisterscreen,text = "Register",bg='medium blue', font='fixedsys',fg='white',width =23, command=openregisterwindow)

Register.place(x=410, y=225)

**4) COMMAND**

**CODING TO CLOSE 'LIGHTS, CAMERA AND POPCORN' APP**

def closestartappwindow():

root.destroy()

startappwindow.destroy()

**5) WINDOW**

**CODING TO OPEN WINDOW FROM WHERE CHOOSE BETWEEN CONTINUING TO 'LIGHTS, CAMERA AND POPCORN' APP**

if con1.is\_connected():

startappwindow = Toplevel()

startappwindow.title("Would you like to continue to the App?")

startappwindow.configure(background="black")

startappwindow.geometry("800x500")

l1 = Label(startappwindow, text="Continue to App?", bg='yellow', fg='grey1',font='fixedsys', width =60)

l1.place(x=140,y=175)

b1 = Button(startappwindow,text = "Yes, continue to app.",bg='midnight blue',font='fixedsys',fg='grey99',width =50, command=openloginorregisterscreen)

b1.place(x=175,y=225)

b2 = Button(startappwindow,text = "No, do not continue to app.",bg='red4',font='fixedsys',fg='grey99',width =50, command=closestartappwindow)

b2.place(x=175,y=275)

**6) WINDOW**

**CODING FOR OPENING THE REGISTER WINDOW**

def openregisterwindow():

registerscreen = Toplevel()

registerscreen.title('Register Screen')

registerscreen.geometry("900x500")

registerscreen.configure(background="black")

emaillabel = Label(registerscreen, text="Enter Email", bg='chocolate1',font='fixedsys',fg='white',width =23)

emaillabel.place(x=125, y =175)

passwordlabel = Label(registerscreen, text="Enter Password", bg='chocolate1',font='fixedsys',fg='white',width =23)

passwordlabel.place(x=125, y =225)

namelabel = Label(registerscreen, text="Enter Name", bg='chocolate1',font='fixedsys',fg='white',width =23)

namelabel.place(x=125, y =275)

global emailentry

emailentry = Entry(registerscreen, font='fixedsys', justify = CENTER, fg='white', bg='light sea green', selectborderwidth = 3, width = 23)

emailentry.place(x=375, y =175)

global passwordentry

passwordentry = Entry(registerscreen, font='fixedsys', justify = CENTER, fg='white', bg='light sea green', selectborderwidth = 3, width = 23)

passwordentry.place(x=375, y =225)

global nameentry

nameentry = Entry(registerscreen, font='fixedsys', justify = CENTER, fg='white', bg='light sea green', selectborderwidth = 3, width = 23)

nameentry.place(x=375, y =275)

checkemailbutton = Button(registerscreen, font='fixedsys', text = "Check Email", bg='firebrick4', fg='white', width =23, command = checkingemail)

checkemailbutton.place(x = 625, y = 175)

checkpasswordbutton = Button(registerscreen, font='fixedsys', text = "Check password", bg='firebrick4', fg='white', width =23, command = checkingpassword)

checkpasswordbutton.place(x = 625, y = 225)

submitdetailsbutton = Button(registerscreen, font='fixedsys', text = "Submit Details", bg='grey33', fg='white', width =23 , command = submitregistrationdetails)

submitdetailsbutton.place(x = 375, y = 325)

registerherelabel = Label(registerscreen, text="Register by entering details as instructed below.", bg='grey33',font='fixedsys',fg='white',width =86)

registerherelabel.place(x=125, y =75)

registerinfolabel = Label(registerscreen, text="Email must contain '@' and '.com', Password must contain atleast 8 characters and atleast one uppercase letter or numerical digit.", bg='grey99',font='fixedsys',fg='grey1', wraplength =600)

registerinfolabel.place(x=200, y =112)

**7) COMMAND**

**CODING TO CHECK EMAIL VALIDITY AND AVAILABILITY**

def checkingemail():

global emaileligibility

checkemailavailability = emailentry.get()

if "@" not in checkemailavailability:

msgbox.showerror("Error", "Does not contain '@', please re-enter.")

emaileligibility = "false"

elif ".com" not in checkemailavailability:

msgbox.showerror("Error", "Does not contain '.com', please re-enter.")

emaileligibility = "false"

else:

msgbox.showinfo("Email is of correct format")

emaileligibility = "correct"

**8) COMMAND**

**CODING TO CHECK IF PASSWORD IS OF CORRECT FORMAT.**

def checkingpassword():

pwd = passwordentry.get()

global eligibility

eligibility = "empty"

for x in pwd:

if x.isupper()==True or x.isnumeric()==True:

eligibility = "correct"

break

else:

eligibility = "false"

continue

if eligibility == "correct" and len(pwd) >= 8:

msgbox.showinfo("Password is of correct format")

eligibility = "correct"

else:

msgbox.showerror("Password is of incorrect format.")

**9) COMMAND**

**CODING TO SUBMIT E-MAIL, PASSWORD AND NAME FOR REGISTRATION.**

def submitregistrationdetails():

checkingemail()

if emaileligibility == "correct" and eligibility == "correct":

registersubmitemail = emailentry.get()

registersubmitname = nameentry.get()

registersubmitpassword = passwordentry.get()

sqlinsertemail = "insert into users values (%s,%s,%s)"

crsr = con1.cursor()

crsr.execute(sqlinsertemail, (registersubmitemail, registersubmitname, registersubmitpassword))

con1.commit()

msgbox.showinfo("Your account has been registered!")

else:

msgbox.showerror("Password or Email is of incorrect format.")

**10) WINDOW**

**CODING TO OPEN A WINDOW FROM WHERE TO LOG-IN**

def openloginwindow():

loginorregisterscreen.destroy()

global loginscreen

loginscreen = Toplevel()

loginscreen.title('Log-In Screen')

loginscreen.geometry("900x500")

loginscreen.configure(background="black")

registerherelabel = Label(loginscreen, text="Register by entering details as instructed below.", bg='yellow',font='fixedsys',fg='grey1',width =86)

registerherelabel.place(x=125, y =112)

loginemail = Label(loginscreen, text="Email", bg='grey25',font='fixedsys',fg='yellow',width =23)

loginemail.place(x=250, y =175)

loginpassword = Label(loginscreen, text="Password", bg='grey25',font='fixedsys',fg='yellow',width =23)

loginpassword.place(x=250, y =225)

global loginemailentry

loginemailentry = Entry(loginscreen, font='fixedsys', justify = CENTER, fg='yellow', bg='grey25', selectborderwidth = 3, width = 23)

loginemailentry.place(x=500, y =175)

global loginpasswordentry

loginpasswordentry = Entry(loginscreen, font='fixedsys', justify = CENTER, fg='yellow', bg='grey25', selectborderwidth = 3, width = 23, show = "\*")

loginpasswordentry.place(x=500, y =225)

loginbutton = Button(loginscreen, font='fixedsys', text = "Log-in", bg='yellow', fg='grey1', width =23, command=logintoapp)

loginbutton.place(x = 375, y = 275)

newaccountbutton = Button(loginscreen, font='fixedsys', text = "Sign Up", bg='firebrick4', fg='grey1', width =23, command=openregisterwindow)

newaccountbutton.place(x = 375, y = 365)

endprogrambutton = Button(loginscreen, font='fixedsys', text = "End Program", bg='grey1', fg='firebrick4', width =23, command=endprogram)

endprogrambutton.place(x = 375, y = 400)

**11) COMMAND**

**CODING TO END PROGRAM**

def endprogram():

msgbox.showinfo("Thankyou for using the - Lights, Camera, Popcorn! - app!")

loginscreen.destroy()

loginorregisterscreen.destroy()

**12) COMMAND**

**CODING TO SUBMIT DETAILS FOR LOG-IN**

def logintoapp():

global loginsubmitemail

loginsubmitemail = loginemailentry.get() #use this for future transactions

global loginsubmitpassword

loginsubmitpassword = loginpasswordentry.get()

sqllogin = "SELECT EMAIL, PASSWORD FROM USERS WHERE Email = %s AND Password = %s"

loginsql = con1.cursor()

loginsql.execute(sqllogin, (loginsubmitemail, loginsubmitpassword))

loginsql.fetchall()

count=loginsql.rowcount

if count > 0:

openhomescreen()

global useemailtransac

useemailtransac = loginsubmitemail

loginscreen.destroy()

else:

msgbox.showerror("Either user doesn't exist or password is incorrect!")

**13) WINDOW**

**HOME SCREEN FROM WHERE TO NAVIGATE TO SEEING WHICH MOVIES ARE ON SHOW, BUYING TICKETS, VIEWING PREVIOUS PURCHASES, AND 'ABOUT US'- INFORMATION ABOUT THE HISTORY AND FOUNDATION OF THIS APP/THEATRE FRANCHISE :)**

def openhomescreen():

global homescreen

homescreen = Toplevel()

homescreen.title('Menu/Home Screen')

homescreen.geometry("1000x700")

homescreen.configure(background="black")

viewmoviesbutton = Button(homescreen, text="Movies-On-Show", bg='grey25',font='fixedsys',fg='yellow',width =30, command = viewmovieswindow)

buyticketsbutton = Button(homescreen, text="Buy tickets", bg='grey25',font='fixedsys',fg='yellow',width =30, command = buyticketswindow)

yourmoviebutton = Button(homescreen, text="YourMovie™", bg='grey25',font='fixedsys',fg='yellow',width =30, command = yourmoviewindow)

infoaboutusbutton = Button(homescreen, text="About Us", bg='grey25',font='fixedsys',fg='yellow',width =30)

logoutbutton = Button(homescreen, text="Log-out", bg='firebrick4',font='fixedsys',fg='grey1',width =30, command = logoutcommand)

emaillabel = Label(homescreen, text=loginsubmitemail, bg='firebrick4',font='fixedsys',fg='grey99',width =30)

homescreenlabel = Label(homescreen, text="Lights, Camera, Popcorn!- HOME SCREEN", bg='yellow',font='fixedsys',fg='grey1',width =50)

viewmoviesbutton.place(x=375, y =175)

buyticketsbutton.place(x=375, y =225)

yourmoviebutton.place(x=375, y =275)

infoaboutusbutton.place(x=375, y =325)

logoutbutton.place(x = 375, y=375)

emaillabel.place(x=750,y=5)

homescreenlabel.place(x=300,y=125)

**14) WINDOW**

**WINDOW SHOWING ALL MOVIES BEING SHOWN IN THEATRES**

def viewmovieswindow():

moviescurrentlyshowingscreen = Toplevel()

moviescurrentlyshowingscreen.title('Movies Currently on show')

moviescurrentlyshowingscreen.geometry("1000x700")

moviescurrentlyshowingscreen.configure(background="black")

currentmovieslabel = Label(moviescurrentlyshowingscreen, text="These are the movies currently showing on the - Lights, Camera, Popcorn! - App. Click on the movies below to view more information before buying your tickets.",

bg='grey25',font='fixedsys',fg='yellow', wraplength =650)

currentmovieslabel.place(x= 212, y =125)

notimetodiebutton = Button(moviescurrentlyshowingscreen, text=" #1: 007: No Time To Die",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = notimetodiescreen)

notimetodiebutton.place(x =100, y =175)

blackwidowbutton = Button(moviescurrentlyshowingscreen, text="#2: Black Widow",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = blackwidowscreen)

blackwidowbutton.place(x =100, y =225)

venombutton = Button(moviescurrentlyshowingscreen, text="#3: Venom: Let There Be Carnage",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = vltbcscreen)

venombutton.place(x =100, y =275)

mibutton = Button(moviescurrentlyshowingscreen, text="#4: Mission Impossible: 07",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = miscreen)

mibutton.place(x =100, y =325)

wonderwomanbutton = Button(moviescurrentlyshowingscreen, text="#5: Wonder Woman 1984",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = wwscreen)

wonderwomanbutton.place(x =400, y =175)

thebatmanbutton = Button(moviescurrentlyshowingscreen, text="#6: The Batman",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = tbscreen)

thebatmanbutton.place(x =400, y =225)

marvelseternalsbutton = Button(moviescurrentlyshowingscreen, text="#7: Marvel's Eternals",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = mescreen)

marvelseternalsbutton.place(x =400, y =275)

mulanbutton = Button(moviescurrentlyshowingscreen, text="#8: Mulan",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = mulanscreen)

mulanbutton.place(x =400, y =325)

flashpointbutton = Button(moviescurrentlyshowingscreen, text="#9: Flashpoint",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = flashscreen)

flashpointbutton.place(x =400, y =375)

ssbutton = Button(moviescurrentlyshowingscreen, text="#10: Suicide Squad 2",

bg='yellow',font='fixedsys',fg='grey25',width =30, command= ssscreen)

ssbutton.place(x =700, y =175)

unchartedbutton = Button(moviescurrentlyshowingscreen, text="#11: Uncharted",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = ucscreen)

unchartedbutton.place(x =700, y =225)

cogbutton = Button(moviescurrentlyshowingscreen, text="#12: Crimes Of Grindelwald",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = cogscreen)

cogbutton.place(x =700, y =275)

jwdbutton = Button(moviescurrentlyshowingscreen, text="#13: Jurassic World: Dominion",

bg='yellow',font='fixedsys',fg='grey25',width =30, command = jwdscreen)

jwdbutton.place(x =700, y =325)

#add all movie information boxes here

#13 to be added

**15 A) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, 007: NO TIME TO DIE**

def notimetodiescreen():

notimetodieinfo = Toplevel()

notimetodieinfo.title('007: No Time To Die')

notimetodieinfo.geometry("1000x700")

notimetodieinfo.configure(background="grey1")

nttdtitle = Label(notimetodieinfo, text="007: No Time To Die", bg='yellow',font='fixedsys',fg='grey1')

nttdtitle.place(x=150, y =75)

nttdsummary = Label(notimetodieinfo, text="SUMMARY: Recruited to rescue a kidnapped scientist, globe-trotting spy James Bond finds himself hot on the trail of a mysterious villain, who's armed with a dangerous new technology.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

nttdsummary.place(x=150, y =125)

nttdcast = Label(notimetodieinfo, text="STARRING: Daniel Craig as 'James Bond', Rami Malek as 'Safin', Ana De Armas as 'Paloma'", bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

nttdcast.place(x=150, y =175)

nttdtime = Label(notimetodieinfo, text="RUNTIME: 163 minutes", bg='blue4',font='fixedsys',fg='grey99')

nttdtime.place(x=150, y =200)

nttddirector = Label(notimetodieinfo, text="DIRECTOR: Cary JF", bg='blue4',font='fixedsys',fg='grey99')

nttddirector.place(x=150, y =225)

nttdrelease = Label(notimetodieinfo, text="RELEASE DATE: 2020-11-21", bg='blue4',font='fixedsys',fg='grey99')

nttdrelease.place(x=150, y =250)

## nttdphoto=PhotoImage(file= "C:\\Users\\mohsi\\Desktop\\nttd.png")

## Label (notimetodieinfo, image=nttdphoto, bg ="black") .grid(row=0, column=0, sticky=W)

**15 B) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, BLACK WIDOW**

def blackwidowscreen():

bwinfo = Toplevel()

bwinfo.title('Black Widow')

bwinfo.geometry("1000x700")

bwinfo.configure(background="grey1")

bwtitle = Label(bwinfo, text="Black Widow", bg='yellow',font='fixedsys',fg='grey1')

bwtitle.place(x=150, y =75)

bwsummary = Label(bwinfo, text="SUMMARY: Following the events of Captain America: Civil War (2016), Natasha Romanoff finds herself alone and forced to confront a dangerous conspiracy with ties to her past. Pursued by a force that will stop at nothing to bring her down, Romanoff must deal with her history as a spy and the broken relationships left in her wake long before she became an Avenger.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

bwsummary.place(x=150, y =125)

bwcast = Label(bwinfo, text="STARRING: Scarlett Johansson as 'Natasha Romanoff', Florence Pugh as 'Yelena Belova', Rachel Weisz as 'Melina Vostokoff'", bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

bwcast.place(x=150, y =200)

bwtime = Label(bwinfo, text="RUNTIME: 120+ minutes", bg='blue4',font='fixedsys',fg='grey99')

bwtime.place(x=150, y =240)

bwdirector = Label(bwinfo, text="DIRECTOR: Cate Shortland", bg='blue4',font='fixedsys',fg='grey99')

bwdirector.place(x=150, y =265)

bwrelease = Label(bwinfo, text="RELEASE DATE: 2020-11-06", bg='blue4',font='fixedsys',fg='grey99')

bwrelease.place(x=150, y =290)

**15 C) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, VENOM: LET THERE BE CARNAGE**

def vltbcscreen():

vltbcinfo = Toplevel()

vltbcinfo.title('Venom: Let there be Carnage')

vltbcinfo.geometry("1000x700")

vltbcinfo.configure(background="grey1")

vltbctitle = Label(vltbcinfo, text="Venom: Let there be Carnage", bg='yellow',font='fixedsys',fg='grey1')

vltbctitle.place(x=150, y =75)

vltbcsummary = Label(vltbcinfo, text="SUMMARY: Venom: Let There Be Carnage is an American superhero film based on the Marvel Comics character Venom, produced by Columbia Pictures in association with Marvel and Tencent Pictures.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

vltbcsummary.place(x=150, y =125)

vltbccast = Label(vltbcinfo, text="STARRING: Tom Hardy as 'Eddie Brock / Venom', Michelle Williams as 'Anne Weying', Woody Harrelson as 'Cletus Kasady / Carnage'",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

vltbccast.place(x=150, y =200)

vltbctime = Label(vltbcinfo, text="RUNTIME: 120+ minutes", bg='blue4',font='fixedsys',fg='grey99')

vltbctime.place(x=150, y =240)

vltbcdirector = Label(vltbcinfo, text="DIRECTOR: Andy Serkis", bg='blue4',font='fixedsys',fg='grey99')

vltbcdirector.place(x=150, y =265)

vltbcrelease = Label(vltbcinfo, text="RELEASE DATE: 2020-10-31", bg='blue4',font='fixedsys',fg='grey99')

vltbcrelease.place(x=150, y =290)

**15 D) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, MISSION IMPOSSIBLE: 07**

def miscreen():

miinfo = Toplevel()

miinfo.title('Mission Impossible: 07')

miinfo.geometry("1000x700")

miinfo.configure(background="grey1")

mititle = Label(miinfo, text="Mission Impossible: 07", bg='yellow',font='fixedsys',fg='grey1')

mititle.place(x=150, y =75)

misummary = Label(miinfo, text="SUMMARY: Mission: Impossible 7 is an American action spy film written and directed by Christopher McQuarrie, and starring Tom Cruise, who reprises his role as Ethan Hunt.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

misummary.place(x=150, y =125)

micast = Label(miinfo, text="STARRING: Tom Cruise as 'Ethan Hunt', Ving Rhames as 'Luther Stickell', Vanessa Kirby as 'Alanna Mitsopolis'",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

micast.place(x=150, y =165)

mitime = Label(miinfo, text="RUNTIME: 120+ minutes", bg='blue4',font='fixedsys',fg='grey99')

mitime.place(x=150, y =205)

midirector = Label(miinfo, text="DIRECTOR: Christopher McQuarrie", bg='blue4',font='fixedsys',fg='grey99')

midirector.place(x=150, y =230)

mirelease = Label(miinfo, text="RELEASE DATE: 2020-11-19", bg='blue4',font='fixedsys',fg='grey99')

mirelease.place(x=150, y =255)

**15 E) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, WONDER WOMAN 1984**

def wwscreen():

wwinfo = Toplevel()

wwinfo.title('Wonder Woman 1984')

wwinfo.geometry("1000x700")

wwinfo.configure(background="grey1")

wwtitle = Label(wwinfo, text="Wonder Woman 1984", bg='yellow',font='fixedsys',fg='grey1')

wwtitle.place(x=150, y =75)

wwsummary = Label(wwinfo, text="SUMMARY: In 1984, during the Cold War, Diana Prince comes into conflict with two formidable foes—media businessman Maxwell Lord and friend-turned-enemy Barbara Ann Minerva while reuniting with Steve Trevor.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

wwsummary.place(x=150, y =125)

wwcast = Label(wwinfo, text="STARRING: Pedro Pascal as 'Maxwell Lord', Gal Gadot as 'Diana Prince / Wonder Woman', Chris Pine as 'Steve Trevor'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

wwcast.place(x=150, y =200)

wwtime = Label(wwinfo, text="RUNTIME: 151 minutes", bg='blue4',font='fixedsys',fg='grey99')

wwtime.place(x=150, y =240)

wwdirector = Label(wwinfo, text="DIRECTOR: Patty Jenkins", bg='blue4',font='fixedsys',fg='grey99')

wwdirector.place(x=150, y =265)

wwrelease = Label(wwinfo, text="RELEASE DATE: 2020-12-16", bg='blue4',font='fixedsys',fg='grey99')

wwrelease.place(x=150, y =290)

**15 F) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, THE BATMAN**

def tbscreen():

tbinfo = Toplevel()

tbinfo.title('The Batman')

tbinfo.geometry("1000x700")

tbinfo.configure(background="grey1")

tbtitle = Label(tbinfo, text="The Batman", bg='yellow',font='fixedsys',fg='grey1')

tbtitle.place(x=150, y =75)

tbsummary = Label(tbinfo, text="SUMMARY: In his second year of fighting crime, Batman explores the corruption that plagues Gotham City and how it may tie to his own family, in addition to coming into conflict with a serial killer known as the Riddler.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

tbsummary.place(x=150, y =125)

tbcast = Label(tbinfo, text="STARRING: Robert Pattinson as 'Bruce Wayne / The Batman', Zoë Kravitz as 'Selina Kyle / Catwoman', Paul Dano as 'Edward Nashton / Riddler'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

tbcast.place(x=150, y =200)

tbtime = Label(tbinfo, text="RUNTIME: 180+ minutes", bg='blue4',font='fixedsys',fg='grey99')

tbtime.place(x=150, y =240)

tbdirector = Label(tbinfo, text="DIRECTOR: Matt Reeves", bg='blue4',font='fixedsys',fg='grey99')

tbdirector.place(x=150, y =265)

tbrelease = Label(tbinfo, text="RELEASE DATE: 2020-12-07", bg='blue4',font='fixedsys',fg='grey99')

tbrelease.place(x=150, y =290)

**15 G) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, MARVEL'S ETERNALS**

def mescreen():

meinfo = Toplevel()

meinfo.title("Marvel's Eternals")

meinfo.geometry("1000x700")

meinfo.configure(background="grey1")

metitle = Label(meinfo, text="Marvel's Eternals", bg='yellow',font='fixedsys',fg='grey1')

metitle.place(x=150, y =75)

mesummary = Label(meinfo, text="SUMMARY: After an unexpected tragedy following the events of Avengers: Endgame (2019), the Eternals—an immortal alien race created by the Celestials who have secretly lived on Earth for over 7000 years—reunite to protect humanity from their evil counterparts, the Deviants.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

mesummary.place(x=150, y =125)

mecast = Label(meinfo, text="STARRING: Angelina Jolie as 'Thena', Richard Madden as 'Ikaris', Kumail Nanjiani as 'Kingo'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

mecast.place(x=150, y =200)

metime = Label(meinfo, text="RUNTIME: 120+ minutes", bg='blue4',font='fixedsys',fg='grey99')

metime.place(x=150, y =240)

medirector = Label(meinfo, text="DIRECTOR: Chloé Zhao", bg='blue4',font='fixedsys',fg='grey99')

medirector.place(x=150, y =265)

merelease = Label(meinfo, text="RELEASE DATE: 2020-11-05", bg='blue4',font='fixedsys',fg='grey99')

merelease.place(x=150, y =290)

**15 H) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, MULAN**

def mulanscreen():

mulaninfo = Toplevel()

mulaninfo.title("Mulan")

mulaninfo.geometry("1000x700")

mulaninfo.configure(background="grey1")

mulantitle = Label(mulaninfo, text="Mulan", bg='yellow',font='fixedsys',fg='grey1')

mulantitle.place(x=150, y =75)

mulansummary = Label(mulaninfo, text="SUMMARY: In Imperial China, Hua Mulan is an adventurous and active girl, to the disappointment of her parents, who hopes that one day she will be wed to a good husband. As a young woman, Mulan is forced to meet with a matchmaker to demonstrate her fitness as a future wife. Mulan, flustered, attempts to pour tea in front of the matchmaker, but a spider causes a panic that destroys the kettle, and the matchmaker calls her a disgrace in front of her family.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

mulansummary.place(x=150, y =125)

mulancast = Label(mulaninfo, text="STARRING: Yifei Liu as 'Mulan', Donnie Yen as 'Commander Tung', Tzi Ma as 'Hua Zhou'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

mulancast.place(x=150, y =215)

mulantime = Label(mulaninfo, text="RUNTIME: 115 minutes", bg='blue4',font='fixedsys',fg='grey99')

mulantime.place(x=150, y =240)

mulandirector = Label(mulaninfo, text="DIRECTOR: Niki Caro", bg='blue4',font='fixedsys',fg='grey99')

mulandirector.place(x=150, y =265)

mulanrelease = Label(mulaninfo, text="RELEASE DATE: 2020-10-04", bg='blue4',font='fixedsys',fg='grey99')

mulanrelease.place(x=150, y =290)

**15 I) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, FLASHPOINT**

def flashscreen():

flashinfo = Toplevel()

flashinfo.title("FlashPoint")

flashinfo.geometry("1000x700")

flashinfo.configure(background="grey1")

flashtitle = Label(flashinfo, text="FlashPoint", bg='yellow',font='fixedsys',fg='grey1')

flashtitle.place(x=150, y =75)

flashsummary = Label(flashinfo, text="SUMMARY: Feature film based on the comic book superhero, The Flash. Stars Ezra Miller as Central City Police Department Detective Bartholomew 'Barry' Henry Allen, who is given the superpowers of superspeed after being struck by lightning during a science experiment by the city's most renknowned Laboratory that goes wrong.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

flashsummary.place(x=150, y =125)

flashcast = Label(flashinfo, text="STARRING: Ezra Miller as 'Barry Allen / The Flash'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

flashcast.place(x=150, y =215)

flashtime = Label(flashinfo, text="RUNTIME: 120+ minutes", bg='blue4',font='fixedsys',fg='grey99')

flashtime.place(x=150, y =240)

flashdirector = Label(flashinfo, text="DIRECTOR: Andrés Muschietti", bg='blue4',font='fixedsys',fg='grey99')

flashdirector.place(x=150, y =265)

flashrelease = Label(flashinfo, text="RELEASE DATE: 2020-12-25", bg='blue4',font='fixedsys',fg='grey99')

flashrelease.place(x=150, y =290)

**15 J) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, SUICIDE SQUAD 2**

def ssscreen():

ssinfo = Toplevel()

ssinfo.title("Suicide Squad 2")

ssinfo.geometry("1000x700")

ssinfo.configure(background="grey1")

sstitle = Label(ssinfo, text="Suicide Squad 2", bg='yellow',font='fixedsys',fg='grey1')

sstitle.place(x=150, y =75)

sssummary = Label(ssinfo, text="SUMMARY: The further adventures of Harley Quinn, Rick Flag and their team of assembled supervillains.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 900)

sssummary.place(x=150, y =125)

sscast = Label(ssinfo, text="STARRING: Margot Robbie as 'Harley Quinn', John Cena as 'Peacemaker', Idris Elba as 'Bloodsport'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

sscast.place(x=150, y =150)

sstime = Label(ssinfo, text="RUNTIME: 120+ minutes", bg='blue4',font='fixedsys',fg='grey99')

sstime.place(x=150, y =175)

ssdirector = Label(ssinfo, text="DIRECTOR: James Gunn", bg='blue4',font='fixedsys',fg='grey99')

ssdirector.place(x=150, y =200)

ssrelease = Label(ssinfo, text="RELEASE DATE: 2020-12-29", bg='blue4',font='fixedsys',fg='grey99')

ssrelease.place(x=150, y =225)

**15 K) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, UNCHARTED**

def ucscreen():

ucinfo = Toplevel()

ucinfo.title('UnCharted')

ucinfo.geometry("1000x700")

ucinfo.configure(background="grey1")

uctitle = Label(ucinfo, text="UnCharted", bg='yellow',font='fixedsys',fg='grey1')

uctitle.place(x=150, y =75)

ucsummary = Label(ucinfo, text="SUMMARY: An American Action adventure film based on the PlayStation exclusive videogame, Uncharted 4: A thief's End.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

ucsummary.place(x=150, y =125)

uccast = Label(ucinfo, text="STARRING: Tom Holland as 'Nathan Drake', Sophia Taylor Ali as 'Chloe Frazer', Mark Wahlberg as 'Victor Sully Sullivan'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

uccast.place(x=150, y =175)

uctime = Label(ucinfo, text="RUNTIME: 120+ minutes", bg='blue4',font='fixedsys',fg='grey99')

uctime.place(x=150, y =215)

ucdirector = Label(ucinfo, text="DIRECTOR: Ruben Fleischer", bg='blue4',font='fixedsys',fg='grey99')

ucdirector.place(x=150, y =240)

ucrelease = Label(ucinfo, text="RELEASE DATE: 2020-12-31", bg='blue4',font='fixedsys',fg='grey99')

ucrelease.place(x=150, y =265)

**15 L) WINDOW**

**SHOWS ALL INFORMATION ABOUT THE MOVIE, CRIMES OF GRINDELWALD**

def cogscreen():

coginfo = Toplevel()

coginfo.title('Crimes Of Grindelwald')

coginfo.geometry("1000x700")

coginfo.configure(background="grey1")

cogtitle = Label(coginfo, text="Crimes Of Grindelwald", bg='yellow',font='fixedsys',fg='grey1')

cogtitle.place(x=150, y =75)

cogsummary = Label(coginfo, text="SUMMARY: Gellert Grindelwald plans to raise an army of wizards to rule over non-magical beings. In response, Newt Scamander's former professor, Albus Dumbledore, seeks his help to stop him.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

cogsummary.place(x=150, y =125)

cogcast = Label(coginfo, text="STARRING: Eddie Redmayne as 'Newt Scamander', Johnny Depp as 'Gellert Grindelwald', Ezra Miller as 'Credence Barebone', Jude Law as 'Albus PVB Dumbledore', Katherine Waterston as 'Tina Goldstein'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

cogcast.place(x=150, y =165)

cogtime = Label(coginfo, text="RUNTIME: 134 minutes", bg='blue4',font='fixedsys',fg='grey99')

cogtime.place(x=150, y =205)

cogdirector = Label(coginfo, text="DIRECTOR: David Yates", bg='blue4',font='fixedsys',fg='grey99')

cogdirector.place(x=150, y =230)

cogrelease = Label(coginfo, text="RELEASE DATE: 2020-12-26", bg='blue4',font='fixedsys',fg='grey99')

cogrelease.place(x=150, y =255)

**15 M) WINDOW**

**#SHOWS ALL INFORMATION ABOUT THE MOVIE, JURASSIC WORLD: DOMINION**

def jwdscreen():

jwdinfo = Toplevel()

jwdinfo.title('Jurassic World: Dominion')

jwdinfo.geometry("1000x700")

jwdinfo.configure(background="grey1")

jwdtitle = Label(jwdinfo, text="Jurassic World: Dominion", bg='yellow',font='fixedsys',fg='grey1')

jwdtitle.place(x=150, y =75)

jwdsummary = Label(jwdinfo, text="SUMMARY: Sequel to the 2018 film 'Jurassic World: Fallen Kingdom'; the plot will revolve around the mutual co-existence of dinosaurs and humans, following the climax of the prequel.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

jwdsummary.place(x=150, y =125)

jwdcast = Label(jwdinfo, text="STARRING: Bryce Dallas Howard as 'Claire Dearing', Chris Pratt as 'Owen Grady', Sam Neill as 'Dr. Alan Grant', Laura Dern as 'Dr. Ellie Sattler', Jeff Goldblum as 'Dr. Ian Malcolm'.",

bg='blue4',font='fixedsys',fg='grey99', wraplength = 800)

jwdcast.place(x=150, y =165)

jwdtime = Label(jwdinfo, text="RUNTIME: 180+ minutes", bg='blue4',font='fixedsys',fg='grey99')

jwdtime.place(x=150, y =205)

jwddirector = Label(jwdinfo, text="DIRECTOR: Colin Trevorrow", bg='blue4',font='fixedsys',fg='grey99')

jwddirector.place(x=150, y =230)

jwdrelease = Label(jwdinfo, text="RELEASE DATE: 2021-01-09", bg='blue4',font='fixedsys',fg='grey99')

jwdrelease.place(x=150, y =255)

**16) WINDOW**

**WINDOW FROM WHERE TO BUY TICKETS**

def buyticketswindow():

buyticketsscreen = Toplevel()

buyticketsscreen.title('Buy your tickets here!')

buyticketsscreen.geometry("1000x700")

buyticketsscreen.configure(background="maroon")

buyticketslabel = Label(buyticketsscreen, text="Buy Tickets here!", bg='grey99',font='fixedsys',fg='grey1',width =50)

buyticketslabel.place(x=525, y =175)

#column of movies available to buy tickets for

allmoviesavailablelabel = Label(buyticketsscreen, text="MOVIES SHOWING", bg='grey1',font='fixedsys',fg='grey99', width = 30)

allmoviesavailablelabel.place(x=25, y =175)

notimetodielabel = Label(buyticketsscreen, text="#1: 007: No Time To Die", bg='grey1',font='fixedsys',fg='grey99',width= 30)

notimetodielabel.place(x=25, y =200)

bwlabel = Label(buyticketsscreen, text="#2: Black Widow", bg='grey1',font='fixedsys',fg='grey99',width =30)

bwlabel.place(x=25, y =220)

vltbclabel = Label(buyticketsscreen, text="#3: Venom: Let There Be Carnage", bg='grey1',font='fixedsys',fg='grey99',width =30)

vltbclabel.place(x=25, y =240)

milabel = Label(buyticketsscreen, text="#4: Mission Impossible: 07", bg='grey1',font='fixedsys',fg='grey99',width =30)

milabel.place(x=25, y =260)

wwlabel = Label(buyticketsscreen, text="#5: Wonder Woman 1984", bg='grey1',font='fixedsys',fg='grey99',width =30)

wwlabel.place(x=25, y =280)

tblabel = Label(buyticketsscreen, text="#6: The Batman", bg='grey1',font='fixedsys',fg='grey99',width =30)

tblabel.place(x=25, y =300)

melabel = Label(buyticketsscreen, text="#7: Marvel's Eternals", bg='grey1',font='fixedsys',fg='grey99',width =30)

melabel.place(x=25, y =320)

mulanlabel = Label(buyticketsscreen, text="#8: Mulan", bg='grey1',font='fixedsys',fg='grey99',width =30)

mulanlabel.place(x=25, y =340)

flashlabel = Label(buyticketsscreen, text="#9: FlashPoint", bg='grey1',font='fixedsys',fg='grey99',width =30)

flashlabel.place(x=25, y =360)

sslabel = Label(buyticketsscreen, text="#10: Suicide Squad 2", bg='grey1',font='fixedsys',fg='grey99',width =30)

sslabel.place(x=25, y =380)

uclabel = Label(buyticketsscreen, text="#11: UnCharted", bg='grey1',font='fixedsys',fg='grey99',width =30)

uclabel.place(x=25, y =400)

coglabel = Label(buyticketsscreen, text="#12: Crimes Of Grindelwald", bg='grey1',font='fixedsys',fg='grey99',width =30)

coglabel.place(x=25, y =420)

jwdlabel = Label(buyticketsscreen, text="#13: Jurassic World: Dominion", bg='grey1',font='fixedsys',fg='grey99',width =30)

jwdlabel.place(x=25, y =440)

#column of show timings

alltimingslabel = Label(buyticketsscreen, text="SHOW TIMINGS", bg='grey1',font='fixedsys',fg='grey99')

alltimingslabel.place(x=275, y =175)

notimetodieshowtime = Label(buyticketsscreen, text="3:00 PM", bg='grey1',font='fixedsys',fg='grey99',width= 12)

notimetodieshowtime.place(x=275, y =200)

bwshowtime = Label(buyticketsscreen, text="4:00 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

bwshowtime.place(x=275, y =220)

vltbcshowtime = Label(buyticketsscreen, text="5:00 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

vltbcshowtime.place(x=275, y =240)

mishowtime = Label(buyticketsscreen, text="6:00 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

mishowtime.place(x=275, y =260)

wwshowtime = Label(buyticketsscreen, text="7:00 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

wwshowtime.place(x=275, y =280)

tbshowtime = Label(buyticketsscreen, text="8:00 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

tbshowtime.place(x=275, y =300)

meshowtime = Label(buyticketsscreen, text="9:00 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

meshowtime.place(x=275, y =320)

mulanshowtime = Label(buyticketsscreen, text="10:00 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

mulanshowtime.place(x=275, y =340)

flashshowtime = Label(buyticketsscreen, text="10:45 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

flashshowtime.place(x=275, y =360)

ssshowtime = Label(buyticketsscreen, text="11:30 PM", bg='grey1',font='fixedsys',fg='grey99',width =12)

ssshowtime.place(x=275, y =380)

ucshowtime = Label(buyticketsscreen, text="12:15 AM", bg='grey1',font='fixedsys',fg='grey99',width =12)

ucshowtime.place(x=275, y =400)

cogshowtime = Label(buyticketsscreen, text="1:00 AM", bg='grey1',font='fixedsys',fg='grey99',width =12)

cogshowtime.place(x=275, y =420)

jwdshowtime = Label(buyticketsscreen, text="1:45 AM", bg='grey1',font='fixedsys',fg='grey99',width =12)

jwdshowtime.place(x=275, y =440)

#column of show prices

allpriceslabel = Label(buyticketsscreen, text="PRICES", bg='grey1',font='fixedsys',fg='grey99')

allpriceslabel.place(x=387, y =175)

notimetodieprice = Label(buyticketsscreen, text="12 USD", bg='grey1',font='fixedsys',fg='grey99',width= 6)

notimetodieprice.place(x=387, y =200)

bwprice = Label(buyticketsscreen, text="12 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

bwprice.place(x=387, y =220)

vltbcprice = Label(buyticketsscreen, text="12 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

vltbcprice.place(x=387, y =240)

miprice = Label(buyticketsscreen, text="11 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

miprice.place(x=387, y =260)

wwprice = Label(buyticketsscreen, text="13 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

wwprice.place(x=387, y =280)

tbprice = Label(buyticketsscreen, text="13 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

tbprice.place(x=387, y =300)

meprice = Label(buyticketsscreen, text="12 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

meprice.place(x=387, y =320)

mulanprice = Label(buyticketsscreen, text="14 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

mulanprice.place(x=387, y =340)

flashprice = Label(buyticketsscreen, text="10 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

flashprice.place(x=387, y =360)

ssprice = Label(buyticketsscreen, text="13 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

ssprice.place(x=387, y =380)

ucprice = Label(buyticketsscreen, text="15 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

ucprice.place(x=387, y =400)

cogprice = Label(buyticketsscreen, text="16 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

cogprice.place(x=387, y =420)

jwdprice = Label(buyticketsscreen, text="12 USD", bg='grey1',font='fixedsys',fg='grey99',width =6)

jwdprice.place(x=387, y =440)

#entry and selection of tickets

EntryMovieNumberLabel = Label(buyticketsscreen, text="Choose Movie here, by typing in Movie# ONLY", bg='grey99',font='fixedsys',fg='grey1')

EntryMovieNumberLabel.place(x=475, y =225)

global EntryMovieNumberEntry

EntryMovieNumberEntry = Entry(buyticketsscreen, font='fixedsys', justify = CENTER, fg='grey1', bg='grey99', selectborderwidth = 3, width = 15)

EntryMovieNumberEntry.place(x=850, y =225)

EntryDateLabel = Label(buyticketsscreen, text="Choose the Date on which you wish to watch the Movie, by typing the Date in YYYY-MM-DD format ONLY",

bg='grey99',font='fixedsys',fg='grey1', wraplength = 375)

EntryDateLabel.place(x=475, y =265)

global EntryDateEntry

EntryDateEntry = Entry(buyticketsscreen, font='fixedsys', justify = CENTER, fg='grey1', bg='grey99', selectborderwidth = 3, width = 15)

EntryDateEntry.place(x=850, y =265)

reserveticketsbutton = Button(buyticketsscreen, text="Reserve Ticket",bg='firebrick4',font='fixedsys',fg='grey99',width =25, command = buytickets)

reserveticketsbutton.place(x =637, y =425)

**17) COMMAND**

**BUYS THE TICKETS**

def buytickets():

#first part of insert query, not executed until the condition is left unfulfilled later

submitemail = useemailtransac

submitmovie = EntryMovieNumberEntry.get()

submitdate = EntryDateEntry.get()

buyticketquery = "Insert into ticketssold values (%s,%s,%s)"

submitqry = con1.cursor()

#counts rows, i.e. tickets already sold for same showing

checkmovie = EntryMovieNumberEntry.get()

checkdate = EntryDateEntry.get()

checksqlquery = "SELECT \* FROM ticketssold WHERE MovieNumber = %s AND DateOfShowing = %s"

checkquery = con1.cursor()

checkquery.execute(checksqlquery, (checkmovie, checkdate))

checkquery.fetchall()

checkcount=checkquery.rowcount

#checks if less than 100 tickets have been sold

if checkcount > 99:

#if 100 have been sold

msgbox.showerror("No Tickets available for this show :( Try a different movie/date")

else:

#if less than 100 have been sold

#insert query is completed

submitqry.execute(buyticketquery, (submitemail, submitmovie, submitdate))

con1.commit()

msgbox.showinfo("Your ticket has been reserved!")

**18) WINDOW**

**WINDOW WHERE USER CAN MANAGE BOOKINGS, REFUND PURCHASES**

def yourmoviewindow():

yourmoviemenu = Toplevel()

yourmoviemenu.title('YourMovie')

yourmoviemenu.geometry("900x500")

yourmoviemenu.configure(background="grey1")

checkprevbutton = Button(yourmoviemenu, text="Check Previous Purchases",bg='firebrick4',font='fixedsys',fg='grey99',width =30, command = previouspurchaseswindow)

checkprevbutton.place(x=325, y =175)

refundbutton = Button(yourmoviemenu, text="Refunds/Cancelling",bg='firebrick4',font='fixedsys',fg='grey99',width =30, command = refundwindow)

refundbutton.place(x=325, y =225)

**19) WINDOW**

**WINDOW WHERE USER CAN CHECK PREVIOUS PURCHASES**

def previouspurchaseswindow():

ppw = Toplevel()

ppw.title('Previous Purchases Search')

ppw.geometry("900x500")

ppw.configure(background="grey1")

findpplabel = Label(ppw, text="Find Previous Purchases by entering in the fields given below", bg='grey99',font='fixedsys',fg='grey1',wraplength = 600)

findpplabel.place(x=175, y =100)

entermovienumberlabel = Label(ppw, text="Enter Movie Number", bg='grey99',font='fixedsys',fg='grey1',width = 30)

entermovienumberlabel.place(x=175, y =125)

enterdatelabel = Label(ppw, text="Enter Date Of Showing", bg='grey99',font='fixedsys',fg='grey1',width = 30)

enterdatelabel.place(x=175, y =150)

global movieentry

movieentry = Entry(ppw, font='fixedsys', justify = CENTER, fg='white', bg='light sea green', selectborderwidth = 3, width = 24)

movieentry.place(x=475, y =125)

global dateentry

dateentry = Entry(ppw, font='fixedsys', justify = CENTER, fg='white', bg='light sea green', selectborderwidth = 3, width = 24)

dateentry.place(x=475, y =150)

findppbutton = Button(ppw, text="Search",bg='yellow',font='fixedsys',fg='grey1',width =30, command = findppsearchcommand)

findppbutton.place(x=320, y =175)

**20) COMMAND**

**SEARCHES FOR PREVIOUS PURCHASES**

def findppsearchcommand():

ppwr = Toplevel()

ppwr.title('Previous Purchases Result')

ppwr.geometry("900x500")

ppwr.configure(background="grey1")

mepp = movieentry.get()

dapp = dateentry.get()

checkpp = "SELECT MovieNumber, DateOfShowing FROM ticketssold WHERE Email = %s AND MovieNumber = %s AND DateOfShowing = %s"

checkppsql = con1.cursor()

checkppsql.execute(checkpp, (loginsubmitemail, mepp, dapp))

checkppresult = checkppsql.fetchall()

toshowresultlabel = Label(ppwr, text=checkppresult, bg='grey99',font='fixedsys',fg='grey1',wraplength = 125)

toshowresultlabel.place(x=100,y=100)

ticketsboughtlabel = Label(ppwr, text='Tickets Bought', bg='yellow',font='fixedsys',fg='grey1',width = 20)

ticketsboughtlabel.place(x=100,y=25)

movieanddatelabel = Label(ppwr, text='Movie Number Date Of showing', bg='yellow',font='fixedsys',fg='grey1',width = 30)

movieanddatelabel.place(x=100,y=50)

**21) WINDOW**

**WINDOW WHERE USER CAN TAKE BACK THEIR RESERVATIONS**

def refundwindow():

rsw = Toplevel()

rsw.title('Refund Search')

rsw.geometry("900x500")

rsw.configure(background="grey1")

findrlabel = Label(rsw, text="Choose which Tickets you want to cancel by entering in the fields given", bg='grey99',font='fixedsys',fg='grey1',wraplength = 600)

findrlabel.place(x=175, y =100)

entermovienumberforrefund = Label(rsw, text="Enter Movie Number", bg='grey99',font='fixedsys',fg='grey1',width = 30)

entermovienumberforrefund.place(x=175, y =125)

enterdateforrefund = Label(rsw, text="Enter Date Of Showing", bg='grey99',font='fixedsys',fg='grey1',width = 30)

enterdateforrefund.place(x=175, y =150)

global movieentryrefund

movieentryrefund = Entry(rsw, font='fixedsys', justify = CENTER, fg='white', bg='light sea green', selectborderwidth = 3, width = 24)

movieentryrefund.place(x=475, y =125)

global dateentryrefund

dateentryrefund = Entry(rsw, font='fixedsys', justify = CENTER, fg='white', bg='light sea green', selectborderwidth = 3, width = 24)

dateentryrefund.place(x=475, y =150)

refundbutton = Button(rsw, text="Refund",bg='yellow',font='fixedsys',fg='grey1',width =30, command = refundcommand)

refundbutton.place(x=320, y =175)

notelabel = Label(rsw, text="Note: All tickets reserved on the entered date for the entered movie will be refunded. YourMovie™ points used to buy these tickets will not be returned.",

bg='firebrick4',font='fixedsys',fg='grey1',wraplength = 700)

notelabel.place(x=175, y =225)

**22) COMMAND**

**COMMAND THAT UN-RESERVES THE TICKETS**

def refundcommand():

rcmer = movieentryrefund.get()

rcder = dateentryrefund.get()

refundsqlquery = "delete from ticketssold where email = %s AND MovieNumber = %s AND dateofshowing = %s"

refundsql = con1.cursor()

refundsql.execute(refundsqlquery, (loginsubmitemail, rcmer, rcder))

con1.commit()

msgbox.showinfo("Your tickets have been un-reserved!")

**23) Command**

**Command to logout**

def logoutcommand():

msgbox.showinfo("You have successfully logged out.")

loginsubmitemail = ''

homescreen.destroy()

openloginwindow()

mainloop()