**A Project On**

**CINEMA TICKET BOOKING APPLICATION**



**Created By:**

**VISHAKHA MISHRA**

**INDEX:**

|  |  |  |
| --- | --- | --- |
| **SR.NO** | **CONTENT** | **PG.NO** |
|  |  |  |
| **1.** | **AIM** |  |
| **2.** | **OBJECTIVE AND SCOPE OF THE PROJECT** |  |
| **3.** | **SYSTEM REQUIREMENTS** |  |
| **4.** | **SALIENT FEATURES** |  |
| **5.** | **PROGRAM SOURCE CODE** |  |
| **6.** | **A WALK THROUGH (OUTPUT SCREEN)** |  |
| **7.** | **BIBLIOGRAPHY** |  |

**AIM:**

The aim of the Cinema Ticket Booking application is to provide a comprehensive functionality that is required to booking a show. It uses python language for programming, and it’s pandas data-frame libraries for storing and retrieving all the details of ticket booking in csv format.

**OBJECTIVE AND SCOPE OF THE PROJECT:**

* The objective of this project is to allow a customer of the application to book cinema show.

* It presents users with choices to select cities, show timings, show format, theater, seat, and food choices.

* It also shows the seats that are already sold out, and the orientation of the seat from screen, its category (like Gold, Silver, Platinum), and choices of variety of food and beverages their combos and prices.

**SYSTEM**

**REQUIREMENTS:**

Recommended System Requirements

* Processors:
* Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM
* Intel® Xeon® processor E5-2698 v3 at 2.30 GHz (2 sockets, 16 cores each, 1 thread per core), 64 GB of DRAM
* Intel® Xeon Phi™ processor 7210 at 1.30 GHz (1 socket, 64 cores, 4 threads per core), 32 GB of DRAM, 16 GB of MCDRAM (flat mode enabled)
* Disk space: 2 to 3 GB
* Operating systems: Windows® 10, macOS\*, and Linux\*

Minimum System Requirements

* Processors: Intel Atom® processor or Intel® Core™ i3 processor
* Disk space: 1 GB
* RAM 512 MB
* Operating systems: Windows\* 7 or later, macOS, and Linux
* Python\* versions: 3.6.X

**SALIENT FEATURES:**

*Interactive python program that prompts for choices, by showing the menu items at each stage. The program prompts user to confirm seat and food choices, thus allowing user to reenter if he doesn’t want to confirm previous selection or change this choice for reconsideration.*

**1. Project details**

The python program that is created for booking the ticket has the following modules and functions.

 **Main Module:**

This module has a lot of functions that form the entire cinema ticket booking application, some of the core methods are:

1. **\_\_main\_\_**

This method is the entry point of the application it assembles all the other methods to form a single application

1. **printSeatMap**:

This function prints the map of the seat available to be picked for sitting. It also shows the available and taken seats.

--------------------------------------BOX OFFICE------------------------------

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

A [ ] [ ] [ ] [ ] [ ] [x] [ ] [x] [ ] [x] [ ] [ ] [x] [ ] [ ] [ ] [ ] B [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] C [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

**3. printSummary**:

This method prints the final summary of all the choices made by the user, it is typically used for printing the tickets with all the details

------------------------------- | BOOKING SUMMARY |

------------------------------ | Booking Id : 0123A1 | | Movie : Jurrasic Park | | Theater : Inox |

| Show Time : 03:00 PM |

| Show Format : IMAX |

| Seat : a10 |

| Food : 2 L Coke |

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

| Total Cost : ₹ 1100 |

------------------------------- Enjoy your show!!

1. **chooseSeats**:

This method allocates the seats chosen by the ticket booker, and it internally uses getCostOfSeat method to calculate the total price for the booking of the seats.

1. **chooseFood**:

This method sums all the choices picked by the users, and sums it up to determine the total price of all the food items selected by the movie watcher

 **Utility Modules:**

**1. Printing**

**a. Print header**

The menu of each of the choices is printed using header. Together with print centered it creates a box that is aligned centrally on the console

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| |

| Theaters Menu |  **Header** (printed in Center)

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  **Separator**

| |

| 1: PVR |

| 2: Inox |

| 3: Big Cinemas |

| 4: Fame |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  **Separator**

| 0: Exit the Application |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

# b. Print Food Menu

This method prints the Food Menu that lists all the snack items available to be used during the cinema like Popcorn, ThumsUp, etc

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

| Food Menu |Large|Medium|Reg|

|----------------------------------------------------------|-------|--------|-----|

| 1: Coke/Fanta/Mirinda/Pepsi/Sprite/Thums Up |₹ 300|₹ 250 |₹ 200|

| 2: Pop Corn [ a. Cheese b. Caramel c. Salted ] |₹ 400|₹ 300 |₹ 200|

| 3: Nachos | | |₹ 200|

| 4: Samosa/Patties [ 2 pieces ] | | |₹ 200|

| 5: Sandwich | | |₹ 200|

|----------------------------------------------------------------------------------|

| 6: Combo #1 | Large Pop Corn + Large Coke/Pepsi/Fanta/Sprite |₹ 600|

| 7: Combo #2 | Large Pop Corn + Samosa/Patties |₹ 500|

| 8: Combo #3 | Large Coke/Pepsi/Fanta/Sprite + Samosa/Patties |₹ 400|

| 9: Combo #4 | Medium Pop Corn + Sandwich + Reg Coke/Mirinda |₹ 600|

|----------------------------------------------------------------------------------| | 0: No Food Needed I am already full |

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

1. **Print separator**

This function separates the main menu choices with the default choice to quit the application at any time (See 0: Exit the Application above)

1. **Print centered**

This function determines the menu name and it fits it centered in the box that is used to display the menu title

**2. Data Storage and Retrieval in CSV using pandas Dataframe**

This module uses to the panda’s library, it creates csv files for all the different choices presented to user for selection like movies.csv, cities.csv, theaters.csv, show\_times.csv, show\_format.csv, and bookings.csv to manage the storage and retrieval of all the necessary information required for cinema ticket booking

# a. getMovies()

→ This function gets the list of all the movies running across all cinema theaters of different cities

# b. getCities()

→ Gets the list of all the cities where the application can book the movies

# c. getTheaters()

→ Gets the list of all the theaters where one can book cinema tickets

# d. getShowTimes()

→ Lists all the slots when the movie is playing from morning, matinee, to second show

# e. getShowFormat()

→ Lists all the formats in which the shows are available like 2D, 3D, 4D, IMAX etc

# f. getSeats()

→ Lists all the seats available for booking in different categories like Box office, Gold, Platinum, Silver etc

# g. insertBooking()

→ This function stores the booking for a show, this helps in ensuring no two people book same seats for the same show at the same time in the same theatre

# 3. Summary

The Cinema Ticket Booking Application is a python program that is modularized into two modules one for helping to print menus and choices, and the second for data interaction.

**PROGRAM SOURCE CODE:**

'''

CinemaTicketBookingApplication.py: Project File Name

Creator: Vishakha Mishra

Project Title: Cinema Ticket Booking Application

Project Details:

This project in python consisting of multiple functions that together form an application for booking cinema tickets

Some of the salient features of this application is booking the movie of choice booking the movie in theater of choice booking the seats of choice booking the timing of choice booking the movie in city of choice booking the movie on day of choice booking the food of choice

'''

# Import the tempfile system module for finding temporary directory in an OS import pandas as pd

# The below imports are used to find today's date and add days to get next days from datetime import datetime, timedelta

from print\_util import \* from panda\_util import \*

#Constants not to be changed in the program, works only in linux

#COLOR = {"BLUE":"\033[34m","RED":"\033[31m","GREEN":"\033[32m","NONE":"\033[m"}

# Define your own movies, theaters, timings, format and types here

ROW\_MIN = ord('A') # A ord:to retrive the ASCII values of given character

ROW\_MAX = ord('W') # W

COL\_MIN = 1 # 1

COL\_MAX = 17 # 17

PLATINUM\_START\_ROW = ord('D')

GOLD\_START\_ROW = ord('I')

SILVER\_START\_ROW = ord('P')

BOX\_OFFICE\_PRICE = 500

reservedSeats = [] # Seats booked in previous sessions

def setReserved(seats): if seats.empty: return df = seats['seat'] for seat in df: for eachSeat in seat.split(','):

reservedSeats.append(eachSeat.upper())

def isAvailable(seat1): seat = seat1.upper() if seat in reservedSeats: return "x" return " "

'''

This method generates 9 days starting from tomorrow

Inputs:

======

None

Outputs:

=======

List: Dates of 9 days from tomorrow

Example:

=======

The Menu Separator that separates choices and Exit option

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

'''

def generateNextNineDaysFromTomorrow():

dates = [] for i in range(1,10):

dates.append( str( datetime.now() + timedelta(days=i)).split(" ")[0]) return dates

# The food price is statically copied from above printFoodMenu function def getFoodPrice(item, size): if item == 1: if size == 'L':

return 300 elif size == 'M': return 250 return 200 elif item == 2: if size == 'L': return 400 elif size == 'M': return 300 return 200 elif item == 3 or item == 4 or item == 5:

return 200 elif item == 6 or item == 9:

return 600 elif item == 7: return 500 elif item == 8: return 400 return 0

'''

--------------------------------------BOX OFFICE------------------------------

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

1. [ ] [ ] [ ] [ ] [ ] [x] [ ] [x] [ ] [x] [ ] [ ] [x] [ ] [ ] [ ] [ ]
2. [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] C [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] '''

def printSeatMap(selectedDate, selectedCity, selectedMovie, selectedTheater,

selectedTiming, selectedFormat):

setReserved(getSeats(selectedDate, selectedCity, selectedMovie,

selectedTheater, selectedTiming, selectedFormat)) printCentered('-','BOX OFFICE (\u20B9 ' + \ str(getCostOfSeat(selectedTiming,ROW\_MIN)) + ")") print("\n 1 2 3 4 5 6 7 8 9 10 11 12 13 14 " + \

"15 16 17\n") rowStr = ""

for row in range(ROW\_MIN, ROW\_MAX+1): if row == PLATINUM\_START\_ROW:

print()

printCentered('-', 'PLATINUM (\u20B9 ' + \ str(getCostOfSeat(selectedTiming,PLATINUM\_START\_ROW)) \

+ ")")

print()

elif row == GOLD\_START\_ROW:

print() printCentered('-', 'GOLD (\u20B9 ' + \ str(getCostOfSeat(selectedTiming,GOLD\_START\_ROW)) + \

")")

print()

elif row == SILVER\_START\_ROW:

print() printCentered('-', 'SILVER (\u20B9 ' + \ str(getCostOfSeat(selectedTiming,SILVER\_START\_ROW)) + \

")")

print()

rowStr = chr(row) + " "

for col in range(COL\_MIN, COL\_MAX+1):

if col == 4 or col == 16:

rowStr = rowStr + " " # Add col separator for stair case rowStr = rowStr + "[" +isAvailable(chr(row).upper()+ str(col)) + "]" + " " print(rowStr)

rowStr = "" for i in range(80): rowStr = rowStr + "-" print("\n" + rowStr)

printCentered(' ',"\ /")

printCentered(' '," \ Screen This Way /") printCentered(' '," \\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/")

def getCostOfSeat(selectedTiming, row):

baseCost = BOX\_OFFICE\_PRICE if selectedTiming == "09:00AM": baseCost = baseCost - 100

if row < PLATINUM\_START\_ROW:

return baseCost

elif row < GOLD\_START\_ROW: return baseCost - 100 elif row < SILVER\_START\_ROW:

return baseCost - 200

else:

return baseCost - 300

def chooseSeat(selectedTiming):

choiceText = None

failed = True

while failed:

choiceText = input("Choose the seats of your choice (Ex: A10,B7,C9) :")

seats = choiceText.split(',') totalCost = 0 bookedSeats = []

for seat in seats:

try:

seat = seat.upper() row = ord(seat[0]) except Exception as ex:

print("Invalid seat choice:", seat, "please retry", ex) failed = True break

if row < ROW\_MIN or row > ROW\_MAX:

print("No Row Exists of series", seat[0], "invalid choice, " + \ "retry with valid row", chr(ROW\_MIN), "-", chr(ROW\_MAX))

failed = True break else: try:

col = int(seat[1:len(seat)])

if col < COL\_MIN or col > COL\_MAX:

print("No Column Exist of the seat", seat, "please retry " + \

"with valid col", COL\_MIN, "-", COL\_MAX) failed = True break else:

# Now that the row/col combination is valid, see if it is pre-booked if isAvailable(seat) != " ": # Not reserved print("Seat", seat, "is already booked, please try other seat") failed = True break else:

#print("Seat", seat, "is ok for booking") if seat not in bookedSeats: bookedSeats.append(seat) totalCost = totalCost + getCostOfSeat(selectedTiming, row)

failed = False continue except Exception as ex:

print("Invalid seat choice:", seat, "please retry", ex) failed = True break if not failed: print('Booking accepted for seats :', choiceText, 'Total Cost is : ₹', \ totalCost)

choiceText2 = input('Do you want to continue with the selection Y or ' + \

'replan N?')

if choiceText2 == 'N' or choiceText2 == 'n':

failed = True return choiceText, totalCost

'''

This method creates a menu item and displays the choices

Inputs:

======

1. optList - The list of options to be used for creating menu
2. option - It is the heading of the menu Ex: Movie, Ticket

Outputs:

=======

1. Return the number choosen by the user Ex: 1 corresponds to 0th element

Example:

=======

The showFormat Menu

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| |

| ShowFormat Menu |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| |

| 1: 2D |

| 2: 3D |

| 3: 4D |

| 4: IMAX |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| 0: Exit the Application |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

'''

def chooseOptions(optList, option):

i = 0

maxLen = len("0: Exit the Application") for opt in optList:

# Using ternary operator in python maxLen = (len(opt) > maxLen) and len(opt) or maxLen

printHeader(option, maxLen)

# For each option, generate a sequencial choice number Ex: 1,2,3,4... for opt in optList:

i = i + 1

printPretty(str(i) + ": " + opt, maxLen) printSeparator(maxLen) printPretty("0: Exit the Application", maxLen) printSeparator(maxLen)

while True:

choiceText = None

try:

choiceText = input("Enter you choice: ") choice = int(choiceText) except: print("Invalid choice, [" + choiceText + "], please retry") continue

if choice == 0:

print("Good Bye!!")

exit()

elif choice < 1 or choice > len(optList):

print("Invalid choice, [" + choiceText + "], please retry") continue break # The choices were correct, so exit the while loop return choice

def chooseFood(): foodSummary = ""

totalCost = 0 failed = True while failed:

choiceText = input("Choose the food (Ex: 2\*1cL [2 coke Large], 3\*2aR[3 " + \

"cheese pop corn reg], 6) :") choices = choiceText.split(',') for choice in choices:

qty = 0 item = 0 x = choice.split("\*") if len(x) == 1: # No quantity specified assume one

qty = 1 item = x[0] else: try:

qty = int(x[0])

except Exception as ex:

print("Invalid quantity entered for choice", choice, "please retry") failed = True

totalCost = 0 foodSummary = ""

break

if qty < 1 or qty > 1000:

print("Quantity should be atleast 1 and less than 1000")

failed = True totalCost = 0

foodSummary = ""

break item = x[1]

# Quantity is Fine, now lets decipher the item

try:

itemCode = int(item[0]) except Exception as ex:

print("Invalid choice of food", choice, "please retry")

failed = True totalCost = 0

foodSummary = ""

break

if itemCode == 0: # Application Quit Selected failed = False totalCost =0 foodSummary = ""

break

elif itemCode == 1: # Coke/Fanta/Mirinda/Pepsi/Sprite/Thums Up Selected

if len(item) != 3:

print('Invalid format for choice #1[', choice, '], please retry') failed = True

totalCost = 0 foodSummary = ""

break size = item[2] if size == 'l' or size == 'L':

size = 'L'

elif size == 'm' or size == 'M':

size = 'M' elif size == 'r' or size == 'R' or size == 's' or size == 'S':

size = 'R' else:

print("Invalid item size selected", choice, "please retry")

failed = True totalCost = 0

foodSummary = ""

break

if item[1] == 'c' or item[1] == 'C': # Coke Selected foodSummary = foodSummary + str(qty) + " " + size + " Coke, " elif item[1] == 'f' or item[1] == 'F': # Fanta Selected foodSummary = foodSummary + str(qty) + " " + size + " Fanta, " elif item[1] == 'm' or item[1] == 'M': # Mirinda Selected foodSummary = foodSummary + str(qty) + " " + size + " Mirinda, " elif item[1] == 'p' or item[1] == 'P': # Pepsi Selected foodSummary = foodSummary + str(qty) + " " + size + " Pepsi, " elif item[1] == 's' or item[1] == 'S': # Sprite Selected foodSummary = foodSummary + str(qty) + " " + size + " Sprite, " elif item[1] == 't' or item[1] == 'T': # Thums Up Selected

foodSummary = foodSummary + str(qty) + " " + size + " Thums Up, "

else:

print("Invalid item selected", choice, "please retry")

failed = True totalCost = 0

foodSummary= ""

break

totalCost = totalCost + qty \* getFoodPrice(itemCode, size)

failed = False elif itemCode == 2: # Pop Corn Selected

if len(item) != 3: print('Invalid format for choice #2[', choice, '], please retry') failed = True totalCost = 0

foodSummary = ""

break size = item[2] if size == 'l' or size == 'L':

size = 'L' elif size == 'm' or size == 'M':

size = 'M' elif size == 'r' or size == 'R' or size == 's' or size == 'S':

size = 'R' else:

print("Invalid item size selected", choice, "please retry") failed = True totalCost = 0 foodSummary = "" break

if item[1] == 'a' or item[1] == 'A': # Cheese Pop Corn Selected foodSummary = foodSummary + str(qty) + " " + size + \

" Cheese Pop Corn, "

elif item[1] == 'b' or item[1] == 'B': # Caramel Pop Corn Selected foodSummary = foodSummary + str(qty) + " " + size + \

" Caramel Pop Corn, " elif item[1] == 'c' or item[1] == 'C': # Salted Pop Corn Selected foodSummary = foodSummary + str(qty) + " " + size + \

" Salted Pop Corn, " else:

print("Invalid item selected", choice, item, "please retry")

failed = True totalCost = 0

foodSummary= ""

break

totalCost = totalCost + qty \* getFoodPrice(itemCode, size)

failed = False elif itemCode == 3: # Nachos selected foodSummary = foodSummary + str(qty) + " Nachos, " totalCost = totalCost + qty \* getFoodPrice(itemCode, 'R')

failed = False elif itemCode == 4: # Samosa/Pattice Selected foodSummary = foodSummary + str(qty) + " Samosa/Pattice, " totalCost = totalCost + qty \* getFoodPrice(itemCode, 'R')

failed = False elif itemCode == 5: # Sandwich Selected foodSummary = foodSummary + str(qty) + " Sandwich, " totalCost = totalCost + qty \* getFoodPrice(itemCode, 'R') failed = False

elif itemCode == 6: # Combo #1 Selected foodSummary = foodSummary + str(qty) + " Combo #1, " totalCost = totalCost + qty \* getFoodPrice(itemCode, 'R') failed = False

elif itemCode == 7: # Combo #2 Selected foodSummary = foodSummary + str(qty) + " Combo #2, " totalCost = totalCost + qty \* getFoodPrice(itemCode, 'R') failed = False

elif itemCode == 8: # Combo #3 Selected foodSummary = foodSummary + str(qty) + " Combo #3, " totalCost = totalCost + qty \* getFoodPrice(itemCode, 'R') failed = False

elif itemCode == 9: # Combo #4 Selected foodSummary = foodSummary + str(qty) + " Combo #4, " totalCost = totalCost + qty \* getFoodPrice(itemCode, 'R') failed = False else:

print("Invalid item selected", choice, "please retry")

failed = True totalCost = 0

foodSummary= ""

break

# Recontinue?

if not failed:

print('Food booking is complete :', foodSummary, 'Total Cost is : ₹', \

totalCost)

choiceText2 = input('Do you want to continue with the selection Y ' + \ 'or replan N?')

if choiceText2 == 'n' or choiceText2 == 'N':

totalCost = 0 failed = True

foodSummary = "" return foodSummary[0:len(foodSummary)-2], totalCost

def printSummary(bookingIdd, mobileNo, selectedDay, selectedCity, selectedMovie,

selectedTheater, selectedTiming, selectedFormat, selectedFood,

selectedSeat):

maxLen = len(' BOOKING SUMMARY') len2 = len('Seat : ' + selectedSeat) if len2 > maxLen: maxLen = len2 len2 = len("Food : " + selectedFood) if len2 > maxLen: maxLen = len2

strTmp = "" for i in range(maxLen+5): strTmp = strTmp + "-"

print(strTmp) printPretty(' BOOKING SUMMARY', maxLen) print(strTmp) bookingId = str(bookingIdd) printPretty("Booking Id : " + bookingId, maxLen) printPretty("Mobile No : " + str(mobileNo), maxLen) printPretty("Date : " + selectedDay, maxLen) printPretty("City : " + selectedCity, maxLen) printPretty("Movie : " + selectedMovie, maxLen) printPretty("Theater : " + selectedTheater, maxLen) printPretty("Show Time : " + selectedTiming, maxLen) printPretty("Show Format : " + selectedFormat, maxLen) printPretty("Seat : " + selectedSeat, maxLen) printPretty("Food : " + selectedFood, maxLen) print(strTmp) printPretty("Total Cost : ₹ " + str(finalCost), maxLen) print(strTmp)

bookingOneLineSummary = "BookingId=" + bookingId bookingOneLineSummary = bookingOneLineSummary + "`Mobile=" + str(mobileNo) bookingOneLineSummary = bookingOneLineSummary + "`Date=" + selectedDay bookingOneLineSummary = bookingOneLineSummary + "`City=" + selectedCity bookingOneLineSummary = bookingOneLineSummary + "`Movie=" + selectedMovie bookingOneLineSummary = bookingOneLineSummary + "`Theater=" + selectedTheater bookingOneLineSummary = bookingOneLineSummary + "`ShowTime=" + selectedTiming bookingOneLineSummary = bookingOneLineSummary + "`ShowFormat=" +selectedFormat bookingOneLineSummary = bookingOneLineSummary + "`Seat=" + selectedSeat return bookingOneLineSummary

'''

To clear the previous bookings use the below command on the command prompt del %temp%\cinemaTicketBookingApplication.txt

'''

if \_\_name\_\_ == '\_\_main\_\_':

print("Welcome to Cinema Ticket Booking Application\n")

init()

# Take Mobile number as login for application

while True:

choiceText = input("Enter your mobile number for booking : ")

try:

mobileNo = int(choiceText) if mobileNo < 1000000000 or mobileNo > 10000000000: print("A valid mobile number should be 10 digits long " + \

"Ex: 9123456780, please re-enter") continue

# A Valid mobile number is entered, now exit the while loop

break

except Exception as ex:

print("Invalid mobile number entered, please re-enter")

# Take OTP as 4 digit number not including Ex: 0001 while True:

choiceText = input("Enter the OTP sent to your mobile for booking : ")

try:

otp = int(choiceText) if otp < 1000 or otp > 10000:

print("Invalid OTP entered, a valid OTP be 4 digits long Ex: 1234, " \

"retry") continue

# Valid OTP entered, exit the while loop break

except Exception as ex:

print("Invalid OTP number entered, please re-enter")

print("\nWhen do you want to watch the movie?") daysList = generateNextNineDaysFromTomorrow() choice = chooseOptions(daysList, "Dates") selectedDay = daysList[choice-1]

print("\nIn Which city do you want to watch the movie on [", selectedDay, \

"]?\n") cities = getCities() choice = chooseOptions(cities, "Cities") selectedCity = cities[choice-1]

print("\nWhich movie do you want to watch on [", selectedDay, \

"] in [", selectedCity, "]?\n") movies = getMovies() choice = chooseOptions(movies, "Movies") selectedMovie = movies[choice-1]

print("\nIn Which Theater do you want to watch [", selectedMovie, "] on [", \

selectedDay, "] in [", selectedCity, "]?\n") theaters = getTheaters() choice = chooseOptions(theaters, "Theaters") selectedTheater = theaters[choice-1]

print("\nWhen do you want to watch [", selectedMovie, "] at [", \ selectedTheater, "] in [", selectedCity, "] on [", selectedDay, "]?\n") show\_times = getShowTimes() choice = chooseOptions(show\_times, "Timings") selectedTiming = show\_times[choice-1]

print("\nWhat type of show do you want to watch [", selectedMovie, "] at [", \ selectedTheater, "] in [", selectedCity, "] on [", selectedDay, \ selectedTiming, "]?\n") show\_format = getShowFormat() choice = chooseOptions(show\_format, "ShowFormat") selectedFormat = show\_format[choice-1] print("\nWhat type of Seat do you want for [", selectedFormat, selectedMovie, \

"] at [", selectedTheater, "] in [", selectedCity, "] on [", \ selectedDay, selectedTiming, "]?\n")

printSeatMap(selectedDay, selectedCity, selectedMovie, selectedTheater, selectedTiming, selectedFormat)

selectedSeat, seatCost = chooseSeat(selectedTiming) print("\nWhat type of Food you want enjoy with the show \n") printFoodMenu() selectedFood, foodCost = chooseFood() finalCost = seatCost + foodCost bookingId = insertBooking(mobileNo,selectedDay, selectedCity, selectedMovie,

selectedTheater, selectedTiming, selectedFormat,

selectedSeat)

bookingOneLineSummary = printSummary(bookingId, mobileNo, selectedDay,

selectedCity, selectedMovie, selectedTheater, selectedTiming, selectedFormat, selectedFood, selectedSeat) print("Enjoy your show!!")

**PYPLOT SOURCE-CODE:**

❖ **GRAPH-I (MONTHLY INCOME OF DIFFERENT CITIES)**

import matplotlib.pyplot as plt import numpy as np months=["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"]

Pune=[26.0,62.0,33.0,46.0,75.0,85.0,69.5,28.5,61.5,12.5,83.4,26.2]

Bengaluru=[32,45,4.7,6.2,77.8,79.3,11.6,7,2,8,11,13.7]

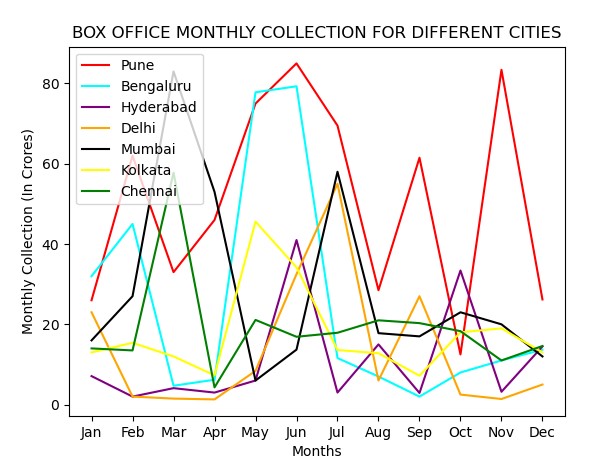
Hyderabad=[7.1,2,4.1,3,6,41,3,15,2.9,33.4,3.2,14.5]

Delhi=[23,2,1.5,1.3,8.4,32.5,55,6,27,2.5,1.4,5]

Mumbai=[16,27,83,53,6,13.7,58,17.8,17,23,20,12]

Kolkata=[13,15.4,12,7.3,45.6,34.2,13.6,12.8,7.2,18,19,13] Chennai=[14.0,13.5,57.8,4.3,21.1,16.9,17.9,21,20.3,18.3,11,14.6] plt.plot(months,Pune,label="Pune",color='red') plt.plot(months,Bengaluru,label="Bengaluru",color='cyan') plt.plot(months,Hyderabad,label="Hyderabad",color='purple') plt.plot(months,Delhi,label="Delhi",color='orange') plt.plot(months,Mumbai,label="Mumbai",color='k') plt.plot(months,Kolkata,label="Kolkata",color='yellow') plt.plot(months,Chennai,label="Chennai",color='green') plt.xlabel("Months") plt.ylabel("Monthly Collection (In Crores)") plt.title("BOX OFFICE MONTHLY COLLECTION FOR DIFFERENT CITIES") plt.legend(loc="upper left",frameon=True) plt.show()

**OUTPUT OF LINEPLOT GRAPH-I:**



* **GRAPH-II (MONTHLY INCOME OF DIFFERENT CINEMAS):**
* import matplotlib.pyplot as plt import numpy as np months=["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct ","Nov","Dec"]

PVR=[76.0,52.0,33.0,46.0,75.0,85.0,69.5,38.5,61.5,12.5,83.4,26.2]

Inox=[23,45,14.7,16.2,77.8,79.3,11.6,7,2,8,11,13.7]

Big\_Cinemas=[7.1,2,4.1,3,6,41,3,15,4.9,33.4,3.2,14.5]

Fame=[13,2,11.5,1.3,18.4,32.5,55,6,27,2.5,1.4,5]

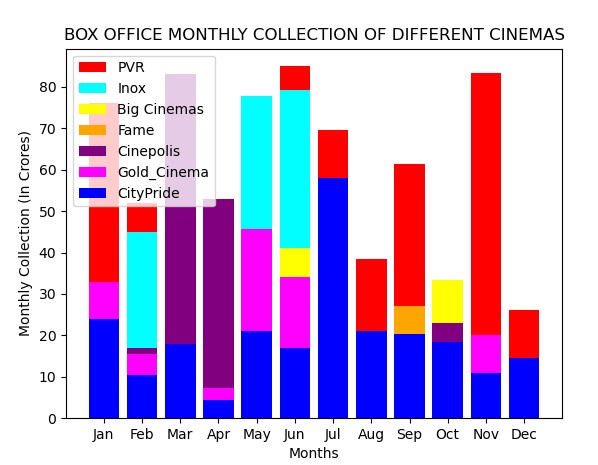
Cinepolis=[6.7,17,83,53,6,13.7,58,17.8,17,23,20,12]

Gold\_Cinema=[33,15.4,12,7.3,45.6,34.2,33.6,12.8,17.2,18,20,13] CityPride=[24.0,10.5,17.8,4.3,21.1,16.9,57.9,21,20.3,18.3,11,14.6] plt.bar(months,PVR,label="PVR",color='red') plt.bar(months,Inox,label="Inox",color='cyan') plt.bar(months,Big\_Cinemas,label="Big Cinemas",color='yellow') plt.bar(months,Fame,label="Fame",color='orange') plt.bar(months,Cinepolis,label="Cinepolis",color='purple') plt.bar(months,Gold\_Cinema,label="Gold\_Cinema",color='magenta') plt.bar(months,CityPride,label="CityPride",color='blue') plt.xlabel("Months") plt.ylabel("Monthly Collection (In Crores)")

plt.title("BOX OFFICE MONTHLY COLLECTION OF DIFFERENT CINEMAS")

plt.legend(loc="upper left",frameon=True) plt.show()

**OUTPUT OF BAR CHART-I:**



❖ **GRAPH-III (MOVIE RATINGS)**

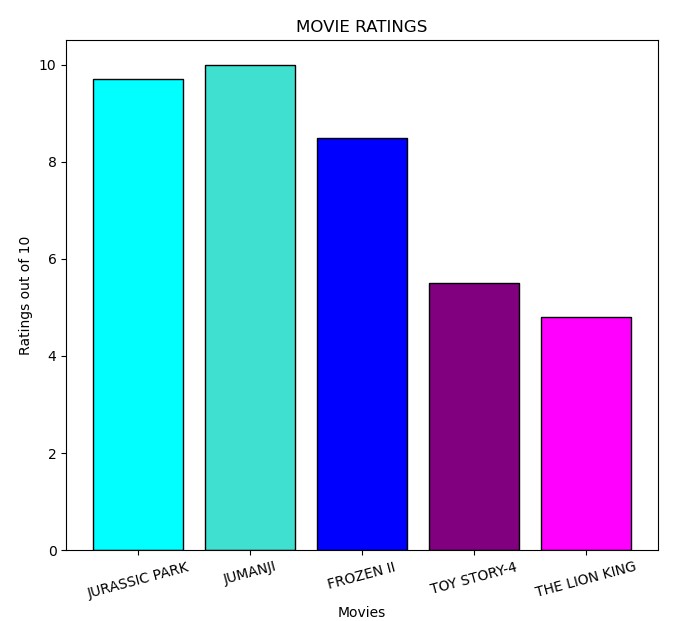
import matplotlib.pyplot as plt import numpy as np

movies=["JURASSIC PARK","JUMANJI","FROZEN II","TOY STORY-4","THE LION KING"]

ratings=[9.7,10,8.5,5.5,4.8]

plt.bar(movies,ratings,color=["cyan","turquoise","blue","purple","mage nta"],edgecolor="k") plt.xlabel("Movies") plt.ylabel("Ratings out of 10") plt.xticks(rotation=15) plt.title("MOVIE RATINGS") plt.show()

**OUTPUT OF BAR CHART-II:**



**A WALK THROUGH**

**(SCREENSHOT OF OUTPUT SCREEN):**

