## horizontal line



Computer Science Project Report

**────────────────────────────────────**

**Name: R. Arun Hariharan**

**Class: XII**

**Roll Number: 17643535**

**Subject Code: 083**

# Index

# 

| **Sno.** | **Segment** |
| --- | --- |
| **1.** | **Certificate** |
| **2.** | **Acknowledgement** |
| **3.** | **Declaration** |
| **4.** | **Introduction** |
| **5.** | **Objectives and Benefits** |
| **6.** | **Tool/Platform used** |
| **7.** | **Flow of Data** |
| **8.** | **Input & Output** |
| **9.** | **Bibliography** |
| **10.** | **Glossary** |

# 

# 

**CERTIFICATE**

**Amity International School, Sec-43, Gurugram, Haryana**



This is to certify that **R. Arun Hariharan** CBSE Roll No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ has satisfactorily completed the project in

**Computer Science(083) Python** laid down in the regulations of CBSE for the purpose of **AISSCE Practical Examination** \_\_\_\_\_ in **Class XII** to be held in **Amity International School, Sector - 43** on\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

(Ms. Jeevan Jyothi)

*PGT for Computer Science*

Examiners:

1. **Name**: ***Signature****:*

(**Internal**)

1. **Name**:

(**External**) ***Signature****:*

**Acknowledgement**

I would like to take this opportunity to express my profound sense of gratitude and respect to all those who helped me throughout the duration of this project. My school Amity International School, Sector-43 has been the source of inspiration for me. I acknowledge the efforts of those who have contributed significantly to my project. I express my sincere gratitude and thankfulness towards Ms. Jeevan Jyoti, Department of Computer Science,Amity International School, Sector-43 for her valuable time and guidance for this project, especially the precious suggestions and encouragement. Finally I would like to thank my parents for constantly supporting my decisions regarding this project.

**Declaration**

I hereby declare that this project and titled life assistant has been done by me at Amity International School, Sector 43, Gurugram for the fulfillment of the requirement for the award of the class 12 board examination 2022-23 and no part of the project has been submitted by me for any other purpose.

**Introduction**

Calling all movie enthusiasts! Are you ready to revolutionize your cinematic adventures? CineSnack, a Python-powered masterpiece, invites you to embark on a journey of convenience and delight. Picture this: no more endless lines, no more agonizing seat scrambles, no more missing the previews due to popcorn procurement. CineSnack seamlessly blends the magic of movies with the power of Python to deliver an unparalleled movie-going experience right at your fingertips. In this project, we'll explore the elegance of Python, a language renowned for its readability, versatility, and extensive libraries.

We'll witness its ability to create a user-friendly booking system that not only simplifies ticket purchases but also elevates movie nights with delectable snacks—all within a few simple code commands. Get ready to unravel the cinematic possibilities unleashed by Python, and pave the way for a future where movie magic is just a click away! This project will help you easily book tickets and in a seamless and effective manner.

**Objectives and Benefits**

Project Objectives:

* Develop a Python-based system for effortless movie ticket booking and snack ordering.
* Implement a menu-driven interface for user-friendly navigation.
* Integrate seat selection functionality with a graphical representation of the theater.
* Store booking information effectively.
* Generate clear and informative output for users.

Benefits of the Project:

* Convenience: Streamlines the movie-going process, saving time and effort.
* Personalization: Empowers users to choose their preferred seats and snacks.
* Efficiency: Automates ticket and snack sales, reducing manual work.
* Enhanced User Experience: Promotes a seamless and enjoyable movie experience.

Benefits of Python:

* Readability: Clear and concise syntax, making it easy to learn and understand.
* Versatility: Applicable to a wide range of tasks, from web development to data analysis.
* Extensive Libraries: Offers a rich collection of pre-written code for various functionalities.
* Cross-Platform Compatibility: Runs smoothly on Windows, macOS, and Linux.
* Ideal for Beginners: User-friendly nature makes it an excellent choice for those new to programming.

**Tools/Platforms Used**

* Programming language: Python 3.x
* Platforms: Compatible with Python interpreters on any major operating system
* Functions and Libraries used:
  + tkinter: A standard GUI (Graphical User Interface) library for Python, providing tools to create windows, buttons, labels, etc.
  + PhotoImage: A class in tkinter for displaying images.
  + lambda function: An anonymous function used for creating small, one-time-use functions, often for event handling in GUIs.
  + subsample method: Used to resize images in tkinter.
  + global keyword: Declares a variable as global, allowing it to be accessed and modified globally.
  + grid method: Organizes widgets (buttons, labels, etc.) in a table-like structure.
  + Checkbutton widget: A tkinter widget for creating checkboxes.
  + pack\_forget method: Hides a tkinter frame.
  + Font class: Allows customization of text font in tkinter.
  + Label widget: Displays text or images in tkinter.
  + Button widget: Creates clickable buttons in tkinter.
  + configure method: Modifies attributes of tkinter widgets dynamically.

**Software Development Cycle**

1. **Requirements:** Understanding the requirements, including movie selection, seat selection, snack selection, and ticket generation.
2. **Design:** Designing the graphical user interface using tkinter. Creating functions for movie, seat, snack, and ticket handling.
3. **Implementation:** Writing the code, handling global variables for selected items, and managing frame transitions.
4. **Testing:** Iterative testing of each frame and button to ensure proper functionality.
5. **Debugging:** Identifying and fixing errors, ensuring smooth transitions and accurate data handling.
6. **Deployment:** The program is ready for deployment as an executable or script for end-users.
7. **Maintenance:** Periodic updates or improvements based on user feedback or additional features.

**Flow of Data**

* **Movie Selection:** User selects a movie by clicking on the corresponding button in the movie frame.
* **Movie Frame Transition:** The on\_movie\_selection function is triggered, storing the selected movie globally (selected\_movie). The movie frame is then hidden, and the seat frame is shown.
* **Seat Selection:** User selects seats by clicking on buttons in the seat frame.
* **Seat Frame Transition:** The on\_done\_seat function is triggered, hiding the seat frame and showing the snack frame.
* **Snack Selection:** User selects snacks using checkboxes in the snack frame.
* **Snack Frame Transition:** The on\_done\_snack function is triggered, hiding the snack frame and showing the ticket frame.
* **Ticket Generation:** The generate\_ticket function generates a ticket with details of the selected movie, seats, snacks, and the total price.
* **Ticket Frame Transition:** The ticket frame is displayed.
* **Program Completion:** User clicks the "Done" button in the ticket frame, hiding the ticket frame and completing the program.

**Key Data Elements**:

* **Movies:**
  + **Attributes:**
    - Name (e.g., "Interstellar")
    - Image path (file path to the movie poster or logo)
    - Width and height (dimensions for resizing)
  + **Data Structure:**
    - List of dictionaries, where each dictionary represents a movie.
* **Selected Movie:**
  + **Attributes:**
    - Name
    - Image path
    - Width and height
  + **Data Structure:**
    - Dictionary representing the currently selected movie.
* **Selected Seats:**
  + **Attributes:**
    - Row and column information for each selected seat.
  + **Data Structure:**
    - List of strings, where each string represents a seat (e.g., "A1", "B3").
* **Selected Snacks:**
  + **Attributes:**
    - Name and price information for each selected snack.
  + **Data Structure:**
    - List of strings, where each string represents a snack (e.g., "Popcorn", "Coke").
* **Total Price:**
  + **Attributes:**
    - The total cost, including movie tickets and selected snacks.
  + **Data Structure:**
    - Integer representing the total price in currency (e.g., INR).

These data elements are crucial for tracking and managing the state of the booking process, ensuring that the user's choices are recorded accurately and used to generate the final ticket information. Global variables, like selected\_movie, selected\_seats, selected\_snacks, and total\_price, are used to maintain the selected state across different frames and functions.

**Input/Source Code**

import tkinter as tk

root = tk.Tk()

root.title("Movie Ticket Booking")

root.geometry("500x500")

root.configure(bg="#ADD8E6") # Light Blue background color

def show\_frame(frame):

frame.pack()

def hide\_frame(frame):

frame.pack\_forget()

def on\_movie\_selection(movie):

# Handle the movie selection logic here

global selected\_movie

selected\_movie = movie

print(f"Selected movie: {movie['name']}")

def on\_done\_movie():

hide\_frame(movie\_frame)

show\_frame(seat\_frame)

# Movies with resized images and larger dimensions

movies = [

{"name": "Interstellar", "image\_path": "D:/FULL CS PRJECT ARUN HARIHARAN 2024/interstellar.gif", "width": 200, "height": 200},

{"name": "Gladiator", "image\_path": "D:/FULL CS PRJECT ARUN HARIHARAN 2024/gladiator.gif", "width": 200, "height": 200},

{"name": "Inception", "image\_path": "D:/FULL CS PRJECT ARUN HARIHARAN 2024/inception.gif", "width": 400, "height": 400},

{"name": "Hacksaw Ridge", "image\_path": "D:/FULL CS PRJECT ARUN HARIHARAN 2024/hacksaw\_ridge.gif", "width": 400, "height": 400},

{"name": "Whiplash", "image\_path": "D:/FULL CS PRJECT ARUN HARIHARAN 2024/whiplash.gif", "width": 400, "height": 400},

{"name": "Dunkirk", "image\_path": "D:/FULL CS PRJECT ARUN HARIHARAN 2024/dunkirk.gif", "width": 400, "height": 400},

{"name": "Tenet", "image\_path": "D:/FULL CS PRJECT ARUN HARIHARAN 2024/tenet.gif", "width": 400, "height": 400},

{"name": "Oppenheimer", "image\_path": "D:/FULL CS PRJECT ARUN HARIHARAN 2024/oppenheimer.gif", "width": 400, "height": 400},

{"name": "The Dark Knight", "image\_path":"D:/FULL CS PRJECT ARUN HARIHARAN 2024/the\_dark\_knight.gif","width": 400, "height": 400}

]

# Movie Frame

movie\_frame = tk.Frame(root, bg="#ADD8E6") # Light Blue background color

for i, movie in enumerate(movies):

# Use PhotoImage to load and display images

img = tk.PhotoImage(file=movie["image\_path"])

# Resize the image by subsampling

img = img.subsample(5) # Change the factor as needed

button = tk.Button(movie\_frame, image=img, text=movie["name"], compound="top", command=lambda m=movie: on\_movie\_selection(m), font=("Helvetica", 12, "bold"))

button.img = img # To prevent image from being garbage collected

button.grid(row=i // 3, column=i % 3, padx=10, pady=10) # Reduced padding

done\_button\_movie = tk.Button(movie\_frame, text="Done", command=on\_done\_movie, font=("Helvetica", 12, "bold"))

done\_button\_movie.grid(row=(len(movies) // 3) + 1, column=0, columnspan=3, pady=10)

# Initial frame to show

show\_frame(movie\_frame)

# Seat Frame

seat\_frame = tk.Frame(root, bg="#FFDAB9") # Peachpuff

seat\_frame.grid\_rowconfigure(9, weight=1)

seat\_frame.grid\_columnconfigure(10, weight=1)

def create\_seat\_buttons():

seat\_buttons = []

for row in range(8):

for col in range(10):

button = tk.Button(seat\_frame, text=f"{chr(65 + row)}{col + 1}", bg="#FFE4B5", command=lambda r=row, c=col: select\_seat(r, c))

button.grid(row=row, column=col, padx=5, pady=5)

seat\_buttons.append(button)

return seat\_buttons

selected\_seats = []

def select\_seat(row, col):

seat = f"{chr(65 + row)}{col + 1}"

if seat in selected\_seats:

selected\_seats.remove(seat)

seat\_buttons[row \* 10 + col].config(bg="#FFE4B5")

else:

selected\_seats.append(seat)

seat\_buttons[row \* 10 + col].config(bg="#90EE90") # Light Green

# Create seat buttons

seat\_buttons = create\_seat\_buttons()

done\_button\_seat = tk.Button(seat\_frame, text="Next", command=lambda: on\_done\_seat(), font=("Helvetica", 12, "bold"), bg="#90EE90") # Light Green

done\_button\_seat.grid(row=9, column=0, columnspan=10, sticky="nsew", padx=20, pady=10)

def on\_done\_seat():

hide\_frame(seat\_frame)

show\_frame(snack\_frame)

# Snack Frame

snacks = {"Popcorn": 150, "Coke": 50, "Nachos": 150, "Fries": 150, "Burger": 200}

snack\_frame = tk.Frame(root, bg="#87CEEB") # SkyBlue

snack\_frame.grid\_rowconfigure(len(snacks) + 2, weight=1)

selected\_snacks = []

total\_price = 0 # Initialize total\_price globally

def update\_price():

global total\_price

number\_of\_seats=len(selected\_seats)

total\_price = 350\*number\_of\_seats # Movie ticket cost

total\_price += sum(snacks[snack] for snack in selected\_snacks)

total\_price\_label.config(text=f"Total Price: ₹{total\_price}")

def on\_check(snack):

if snack in selected\_snacks:

selected\_snacks.remove(snack)

else:

selected\_snacks.append(snack)

update\_price()

frame\_label = tk.Label(snack\_frame, text="Select Snacks", font=("Helvetica", 20, "bold"), bg="#87CEEB", fg="white")

frame\_label.grid(row=0, column=0, columnspan=len(snacks), sticky="nsew")

# Create snack checkboxes

for i, (snack, price) in enumerate(snacks.items()):

checkbox = tk.Checkbutton(snack\_frame, text=f"{snack} - ₹{price}", variable=tk.IntVar(), command=lambda s=snack: on\_check(s), font=("Helvetica", 12), bg="#87CEEB", fg="white")

checkbox.grid(row=i + 1, column=0, padx=20, pady=10)

total\_price\_label = tk.Label(snack\_frame, text="Total Price: ₹0", font=("Helvetica", 14), bg="#87CEEB", fg="white")

total\_price\_label.grid(row=len(snacks) + 1, column=0, columnspan=len(snacks), pady=10)

done\_button\_snack = tk.Button(snack\_frame, text="Next", command=lambda: on\_done\_snack(), font=("Helvetica", 14, "bold"), bg="#90EE90") # Light Green

done\_button\_snack.grid(row=len(snacks) + 2, column=0, columnspan=len(snacks), sticky="nsew", padx=20, pady=10)

def on\_done\_snack():

hide\_frame(snack\_frame)

show\_frame(ticket\_frame)

generate\_ticket()

# Ticket Frame

ticket\_frame = tk.Frame(root, bg="#F08080") # Light Coral

ticket\_frame.grid\_rowconfigure(9, weight=1)

ticket\_frame.grid\_columnconfigure(0, weight=1)

def generate\_ticket():

global selected\_movie

ticket\_label = tk.Label(ticket\_frame, text="--- Movie Ticket ---\nMovie: {}\nSeats: {}\nSnacks: {}\nTotal Price: ₹{}".format(selected\_movie["name"], ", ".join(selected\_seats), ", ".join(selected\_snacks), total\_price), font=("Helvetica", 16), bg="#F08080", fg="white")

ticket\_label.pack()

done\_button\_ticket = tk.Button(ticket\_frame, text="Done", command=lambda: on\_done\_ticket(), font=("Helvetica", 12, "bold"), bg="#90EE90") # Light Green

done\_button\_ticket.pack(pady=20)

def on\_done\_ticket():

hide\_frame(ticket\_frame)

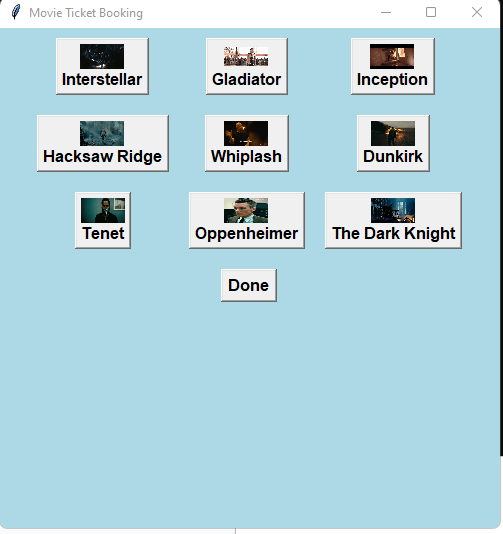
# Initial frame to show

show\_frame(movie\_frame)

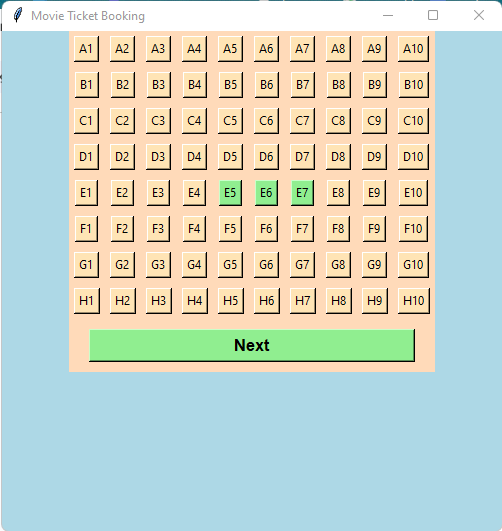
root.mainloop()

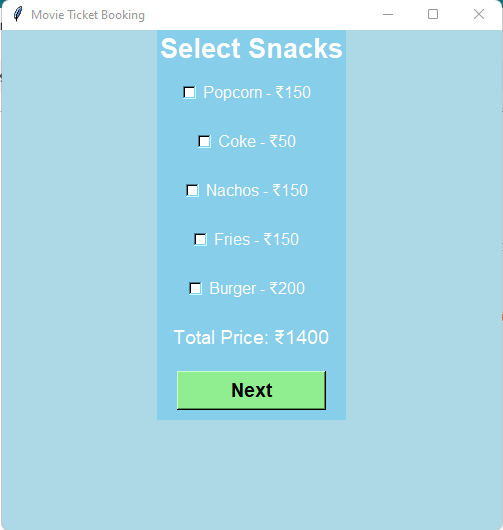
**Output**

*The above output image shows the window that opens after you execute the .py file. It includes the option of movies and the imagery to go along with it. Once you have selected your movie, you can press the ‘done’ button and finalize the choice.*

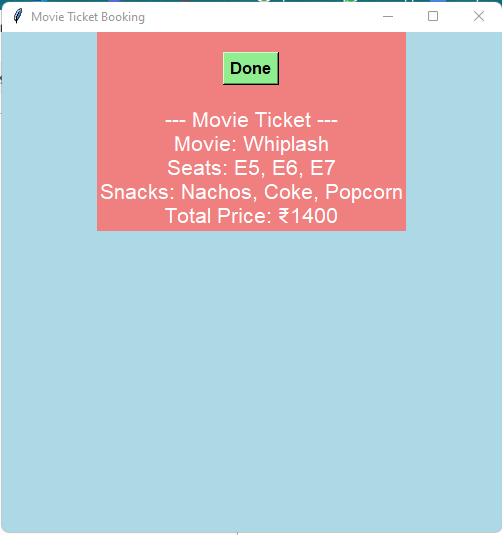
****

*The seats can be selected in this window which works on clicks. You can press any seat you want and select it. The color green indicates that the seat has been selected and pressing next finalizes the selection.*

****

****

*This window allows users to select the snacks they want. The Price index on the bottom reflects your choice in snacks, and the seats.*

****

*The ticket is finally generated. You can use this ticket to go into the theater and enjoy the movie.*

**Bibilography & Glossary**

**Bibliography:**

* "Python Crash Course" by Eric Matthes.
* "Programiz".
* "GeeksforGeeks python tutorial”.
* https://www.youtube.com/watch?v=yQSEXcf6s2I&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV

**Glossary:**

**Data Structures:**

* **List:** An ordered collection of items, used to store the movie list, seat map, and selected snacks.
* **Set:** An unordered collection of unique elements, used to track booked seats.
* **Dictionary:** A collection of key-value pairs, used to store snack names and prices.

**Variables:**

* **root:** tk.Tk() - The main window of the application.
* **selected\_movie:** Dictionary - Represents the currently selected movie.
* **selected\_seats:** List - Contains strings representing the selected seats.
* **selected\_snacks:** List - Contains strings representing the selected snacks.
* **total\_price:** Integer - Represents the total price of the movie tickets and snacks.
* **show\_frame:** Function - Displays a specific frame in the GUI.
* **hide\_frame:** Function - Hides a specific frame in the GUI.
* **on\_movie\_selection:** Function - Handles the logic when a movie is selected.
* **on\_done\_movie:** Function - Handles the logic when the movie selection is done.
* **movies:** List - Contains dictionaries representing different movies with attributes like name, image path, width, and height.
* **movie\_frame:** tk.Frame - Frame for displaying movie selection.
* **done\_button\_movie:** tk.Button - Button to move to the next step after selecting a movie.
* **seat\_frame:** tk.Frame - Frame for seat selection.
* **create\_seat\_buttons:** Function - Creates seat buttons dynamically.
* **seat\_buttons:** List - Contains seat buttons.
* **done\_button\_seat:** tk.Button - Button to move to the next step after selecting seats.
* **snacks:** Dictionary - Contains snack names and their prices.
* **snack\_frame:** tk.Frame - Frame for snack selection.
* **update\_price:** Function - Updates the total price based on selected seats and snacks.
* **on\_check:** Function - Handles the logic when a snack checkbox is checked.
* **total\_price\_label:** tk.Label - Label displaying the total price.
* **done\_button\_snack:** tk.Button - Button to move to the next step after selecting snacks.
* **ticket\_frame:** tk.Frame - Frame for displaying the generated ticket.
* **generate\_ticket:** Function - Generates the ticket information.
* **done\_button\_ticket:** tk.Button - Button to finish the ticketing process.

**Functions:**

* show\_frame(frame)
* hide\_frame(frame)
* on\_movie\_selection(movie)
* on\_done\_movie()
* create\_seat\_buttons()
* select\_seat(row, col)
* on\_done\_seat()
* update\_price()
* on\_check(snack)
* on\_done\_snack()
* generate\_ticket()
* on\_done\_ticket()

**Keywords:**

* global
* lambda
* subsample
* pack
* pack\_forget
* grid
* Text
* PhotoImage
* Button
* Label
* Frame
* Checkbutton
* IntVar
* StringVar
* mainloop
* configure
* title
* geometry
* Button
* Label
* Text
* subsample
* config
* command
* file
* compound
* font
* bg
* fg
* sticky
* columnspan
* rowspan
* text
* variable

**Additional Terms:**

* **Menu-based system:** A type of user interface that presents options in a menu format for user selection.
* **Seat arrangement:** A visual representation of the theater's seating, often using letters and numbers to denote seats.
* **Ticket generation:** The process of creating a printable or digital ticket with booking details.