Level – 2: Medium (Orange Problem)

# Problem 1: Smart Book Selection Assistant

PES Library has digitized its book inventory. Create a Python and Tkinter application to help users filter books based on their preferences.

## Instructions:

1. Create a GUI using Tkinter, allowing users to select preferences such as title, genre, author, year, ratings and language using checkboxes or dropdown menus.
2. Filter and display books from a CSV file containing book details based on user preferences (triggered by a button).
3. Users should be able to reset their preferences to start over.
4. Only books that match **all selected preferences** should be displayed.
5. If no books match the selected preferences, display the top five books in the library based on popularity.
6. Highlight the most popular book from the filtered list.

## Deliverable:

* + A .py file containing the complete code.

# Tools/Technologies:

* Language: Python 3.10 or above.
* Concepts to Apply:
  + Data Structures: Use lists or dictionaries to store and organize movie data.
  + Control Structures: Use loops and conditionals to manipulate and analyze data.
  + File Handling: Read and process the CSV dataset.
  + GUI Programming: Design and implement a user-friendly interface using Tkinter.

## Methodology:

1. Import Necessary Libraries:
2. Load the Dataset:
   * Load the book dataset from a CSV file named book\_dataset.csv.
   * Use Python's csv.reader to parse the file and load the data into a list of dictionaries.
   * Ensure the dataset contains the following columns:
     + **Title**: The name of the book.
     + **Author**: The author of the book.
     + **Year**: The year the book was released.
     + **Genres**: Separate columns for each genre .
     + **Ratings**: The ratings given by the readers
     + **Language**: The language in which the book is written.
3. Data Exploration
4. Detailed Analysis:

* Filtering Logic:
  + Implement a function to filter books based on user-selected genres using loops and conditional statements.
  + Include only book with 1 in the corresponding genre columns for all selected genres.
* Default Recommendations:
  + If no books match the preferences, provide the top 5 ratings as a fallback.
* Highlighting the Top Book:
  + Sort the filtered books by their ratings in descending order.
  + Highlight the best-rated book from the filtered list as the "Top Pick.”

1. Build the GUI:

* Design the Interface:
  + Add a heading titled "SMART BOOK SELECTION ASSISTANT."
  + Include checkboxes for genre selection.
  + Add buttons for "Recommend" and "Clear."
* Dynamic Results Display:
  + Display the top-rated book ("Top Pick") and the filtered book list in a Text

widget.

* + Format the output for readability in a well-organized layout.

**Code:**

import csv

import tkinter as t

from tkinter import messagebox

# function to load book datasheet

def load\_books(file\_name):

books=[]

try:

with open(r"D:\Hamza\Python\OrangeProblem\book\_dataset.csv",'r',newline='\r\n') as f:

reading = csv.DictReader(f)

for row in reading:

# parse the row, converting numeric fields to proper ty

book = {

"Title": row["Title"],

"Release Year": int(row["Release Year"]),

"Rating": float(row["Rating"])

}

# add genre information (one-hot encoding)

for genre in genres:

book[genre] = int(row[genre])

books.append(book)

except FileNotFoundError:

messagebox.showerror("Error", f"File '{file\_name}' not found!")

except Exception as e:

messagebox.showerror("Error", f"Error loading file: {e}")

return books

# function to filter books based on user preference

def recommended\_books():

selected\_genres = [genre for genre, var in genre\_vars.items() if var.get() == 1]

if not selected\_genres:

messagebox.showinfo("No selection, please select genre")

return

filtered\_books=[

book for book in books

if all(book[genre]==1 for genre in selected\_genres)

]

# display results

result\_text.delete(1.0, t.END)

if filtered\_books:

# sort by rating (highest rating first)

filtered\_books .sort(key=lambda x:x["Rating"], reverse=True)

# display the top match

top\_book = filtered\_books[0]

result\_text.insert(

t.END,

f"Top Pick: {top\_book["Title"]} ({top\_book['Release Year']}) - Rating: {top\_book['Rating']}\n\n"

)

# display other recommendations

result\_text.insert(t.END, "Recommended Book:\n\n")

for book in filtered\_books[:5]:

result\_text.insert(

t.END,

f"{book["Title"]} ({book['Release Year']}) - Rating: {book['Rating']}\n\n"

)

else:

books.sort(key= lambda x:x["Rating"], reverse = True)

result\_text.insert(t.END,"No matches found. Showing top rated books:\n\n")

for book in books[:5]:

result\_text.insert(

t.END,

f"{book["Title"]} ({book['Release Year']}) - Rating: {book['Rating']}\n\n"

)

# function to clear all preferences

def clear\_preferences():

for var in genre\_vars.values():

var.set(0)

result\_text.delete(1.0, t.END)

# load the book data

genres = ["Action", "Comedy", "Drama", "Sci-Fi","Romance", "Animation"]

books = load\_books("book\_datasheet.csv")

# create main tkinter window

root = t.Tk()

root.title("Book Finder Assistant")

root.geometry("1500x900")

root.resizable(False,False)

# to add heading

t.Label(root,text = "Select your favourite genres", font=("Arial",16,"bold")).pack(pady=11)

# UI components

t.Label(root,text = "Select genres:", font=("Arial",14)).pack(pady=10)

# checkbox for each genre

genre\_vars = {genre: t.IntVar() for genre in genres}

for genre, var in genre\_vars.items():

t.Checkbutton(root,text = genre, variable = var).pack(anchor = 'w',padx=20)

# adding buttons

t.Button(root, text= "Recommended", command= recommended\_books).pack(pady=10)

t.Button(root, text= "Clear", command= clear\_preferences).pack(pady=10)

# displaying the result

result\_text = t.Text(root, wrap="word", height=20, width=50)

result\_text.pack(pady=10)

# run the application

root.mainloop()

### Output:





