Coursework Assignment Specification
(50 % of the module assessment)
Submission Deadline: at 11:00 on 13th December 2023

Video Game Rental Management System

1 Problem Description

1.1 Overview

Your task is to develop a video game rental management system. The system should enable a store manager to manage video game rentals based on customer subscriptions. The manager should be able to check the availability of video games, rent them out to customers and maintain a database of all video games in the inventory.

For each video game, the following information should be stored: title, platform (e.g., PlayStation, Xbox), genre, publisher, purchase date, and a unique ID number to distinguish different copies of the same game.

The system should also integrate with provided **'subscriptionManager.pyc'** and **'feedbackManager.pyc'** modules to manage customer subscriptions and collect feedback, respectively.

1.2 Customer

Customers should be identified using their unique IDs, which consist of 4 alphabetic letters (e.g., "coai"). Only customers with active subscriptions, as verified by the provided '**subscriptionManager.pyc'** module, are considered valid for renting games.

1.3 Searching for Games

Your program should include functionality to search for a game based on its title, genre, or platform. Given a search term, your program should return a complete list of games with all their associated information including availability of the game.

1.4 Renting Games

To rent a game, the store manager should provide a customer ID (e.g., "coai") and the game's ID (e.g., "cod01"). Your program should:

- Validate the input and use the provided **subscriptionManager.pyc** to check the customer's subscription status.
- Depending on the game's availability and the customer's subscription limit, either allow the manager to rent the game by updating the related records in the database or return an appropriate message indicating why the rental cannot proceed.

1.5 Returning Games and Collecting Feedback

Upon the return of a game, the store manager should be able to enter a star rating and optional comments about the customer's experience. This feedback will be stored in an external database and managed by '**feedbackManager.pyc'**.

The main database should also be updated to reflect the game's return. If the ID is invalid, or the game is already available, the program should return an error message.

1.6 Inventory Pruning

Your program should have functionality to identify unpopular game titles that are not frequently rented. You should come up with good criteria to find unpopular games. Based on this information, the program should suggest which games could be removed from the inventory to optimise space and resources.

1.7 Database Requirements

The system must use two delimited text files for the main inventory and transaction log, which keeps the rental history of video games. A separate database mustn't be used for storing customer feedback, as managed by **feedbackManager.pyc**.

File 1: Game_Info.txt

ID	Platform	Genre	Title	Publisher	Purchase Date
cod01	PlayStation	Action	COD	Activision	01/08/2020
fifa02	Xbox	Sports	FIFA	EA Sports	01/08/2019
cod02	PlayStation	Action	COD	Activision	01/09/2021
…	…	….	….	….	…

File 2: Rental.txt

Game ID	Rental Date	Return Date	Rented Customer ID
cod01	11/09/2022		abcd
fifa02	12/09/2022	13/10/2022	efgh
tlo04	15/09/2022		mnop

Each line in File 2 shows transaction information. The 'Rental Date', 'Return Date', and 'Rented Customer ID' fields provide details about the actual rental transactions. The 'Return Date' field would be blank (Null) if the game is still rented.

2 How to Structure Your Program

The following structure is needed for the final submission of your project:

Data files stores all the data as specified in the previous section.

Game_Info.txt: Stores initial example data for the details of the video game inventory. Populate this text file with realistic data (minimum 20 records).

Rental.txt: Stores initial example data for the rental history of video games. Populate this text file with realistic data (minimum 100 records). You may write a program to generate this data.

Python Modules

gameSearch.py: A Python module containing functions that allow the store manager to input search terms as strings and return the output as described in the previous section.

gameRent.py: A Python module containing functions that prompt the store manager for the customer's ID and the ID of the game(s) they wish to rent. After performing validity checks and the functionality described earlier, the program should return a message indicating whether the game has been rented successfully.

gameReturn.py: A Python module containing functions that prompt the store manager for the ID of the game(s) they wish to return and collect feedback if applicable.

InventoryPruning.py: A Python module containing functions used to identify games for potential removal based on rental frequency. Before any pruning action, the module should provide suggestions and aid the decision-making process by offering visualisations.

database.py: A Python module containing common functions that the game search, rent, return, and inventory pruning modules use to interact with the data files.

menu.ipynb: A main program that provides the required menu options to the store manager. The menu may be based on the Graphical User Interface (GUI). You are only allowed to use IPyWidgets to build the GUI components. The GUI must use only one window and be easy to use.

3 What to Submit

In addition to the files mentioned in Section 2 you may write a short **text file** called README (max 500 words). This is to provide any special instructions or warnings to the user (or assessor!), or to draw attention to any aspects of the program that you are particularly proud of (please don't waste time by writing an excessive amount).

All the files (including sample data files) should be compressed into a zip file and submitted electronically as directed.

4 Notes on Expectations

You will be marked according to your overall achievement, marked according to the Assessment Matrix that will be provided to you separately from this document. However below follows a qualitative description of some general expectation that may help you understand the general level of expectation associated with this piece of coursework.

Technical mastery of Python Your programs should show mastery of what you have been taught.

Design Your programs should be well structured for the task in hand so that it is as easy as possible for:

- a user to use the program for any likely purpose,
- a programmer to understand the code structure and be able to develop it further,
- a programmer to be able to re-use as much as possible of the code in a related application.

Clarity and Self-Documentation Given the structure of your programs, they should be as easy to read and understand as possible. *Lay your code out* so that it can be listed sensibly on a variety of devices: avoid having any lines longer than 80 characters as these may wrap (to reduce the number of "problem lines" you should use 4 spaces for indentation rather than tabs). *Sensible names* should be chosen for all variables, methods etc. *Documentation strings* should be included for each:

Program Fully explain what the program does and how it should be used. Also state who (ID only) wrote it and when.

Function State what each function does and explain the roles of its parameters. In addition, you should include occasional comments in your code; these may be (a) to introduce a new section in the code, or (b) to explain something that is not obvious. Bear in mind that pointless comments make your code harder to read, not easier.

Restriction

1) Your code must **NOT** include any Class type definition.
2) Your code must **NOT** have any SQL statements.
3) Your code must **NOT** have any nested function declaration.
4) You must use Python v11.
5) Your code must use **ONLY** standard python libraries and Matplotlib.
6) IPyWidgets