25COA122
Introduction to Programming and Databases

Coursework Assignment Specification
(50% of the module assessment)
Submission Deadline: **11am on Wednesday 14th January 2026**

Games Store Management System



Statement on the use of GenAI Tools: GenAI tools are permitted in an assistive role.
Appropriate use includes: (i) Finding literature sources. (ii) Help writing code. (iii) Text proofreading.
Inappropriate use includes: (i) Text/code generation. (ii) Interpretation and discussion of results.

Reminder: Take the Academic Integrity Quiz on Learn.
Reminder: You are responsible for fact checking any AI generated material.
Student Handbook: Acknowledging, describing and referencing the use of GenAI tools .

1 Problem Description

1.1 Overview

Your task is to develop a board/video games management system for a new Gaming Store in Loughborough town centre. The system should enable a store manager to manage game rentals and face-to-face group sessions based on customer subscriptions. The manager should be able to check the availability of board games and video games and rent them out to customers and maintain a database of all games in the inventory. The store manager should also use a booking system to allow subscribers to attend face-to-face sessions at the Store between 2pm and 10pm each day. Two sessions are bookable: 2pm-6pm and 6pm-10pm. Subscribers are allowed to bring up to three guests each.

For each board game, the following information should be stored: name of game, number of players, genre, purchase date and a unique ID number. For each video game: title, platform, genre, purchase date and unique ID number. There must be at least ten board games and ten video games on the system. We expect to see different games for each student.

You MUST use **Google Colab** to create an ipynb notebook for your submission. The provided modules **'subscriptionManager.pyc,'** to manage customer subscriptions, and **'feedbackManager.pyc'** to collect feedback, must be saved in the same folder as your ipynb notebook when you submit.

1.2 Customer

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

1.3 Searching for Games

Your program should include functionality to search for games based on whether it is a board or video game with title and genre. Given a search term, your program should return a complete list of games records with all their associated information including its availability.

1.4 Renting Games

To rent out a game, the store manager should provide a customer ID (e.g., "coab") and the game's ID (e.g., "art01"). 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 or return a message indicating why the rental cannot proceed.

1.5 Booking Face-to-Face Sessions

To book a session, the store manager should provide a customer ID (e.g., "coab"). There is a limit of 50 persons who can attend the Store at any one time. Your program should:

- Validate the input and use the provided **'subscriptionManager.pyc'** to check the customer's subscription status.
- Depending on available time slots, either allow a booking or indicate why the booking cannot proceed.

1.6 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 not returnable, the program should return an error message.

1.7 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 game records. Based on this information, the program should suggest which game records could be removed from the inventory to optimise space and resources.

1.8 Database Requirements

The system must use five delimited text files for the main inventory and transaction log, which keeps the rental history of game records. You should not edit Game_Feedback.txt, your program will populate it for you.

File 1: Board_Game_Info.txt

Game ID	Name	No. Players	Genre	Purchase Date
art01	Articulate	4-20	Word	2025-08-01
bac02	Backgammon	2	Strategy	2024-08-01
chs03	Chess	2	Strategy	2024-08-01
...	...	...	...	...

File 2: Video_Game_Info.txt

Game ID	Name	Platform	Genre	Purchase Date
ff701	Final Fantasy VII	PlayStation	Fantasy	2025-08-01
gta02	Grand Theft Auto 5	Xbox	Action	2025-04-01
mnc03	Minecraft	Nintendo Switch	Sandbox	2025-06-01
...	...	...	...	...

File 3: Rental.txt

Game ID	Rental Date	Return Date	Rental Customer ID
art01	2025-09-01	2025-10-01	coab
bac02	2025-09-01	2025-10-01	efgh
chs03	2025-10-01		mnop
...	...	...	...

Each line in File 3 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.

File 4: Booking.txt

User ID	Booking Date	Time	Number of Guests
coab	2025-09-01	2pm	1
efgh	2025-09-02	2pm	2
mnop	2025-09-03	6pm	3
...	...	...	...

Customers can book slots 2pm-6pm or 6pm-10pm and can bring up to 3 guests.

2 How to Structure Your Program

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

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

Board/Video_Game_Info.txt: Stores initial example data for the details of the games-record inventories. You must populate this text files with realistic data (minimum 10 records each).

Rental.txt: Stores initial example data for the rental history of game-records. You must populate this text file with realistic data (minimum 50 records). You may write a program to generate this data.

Booking.txt: Stores initial example data for the booking system. You must populate this text file with realistic data (minimum 50 records). You may write a program to generate this data.

Python Cells – You MUST label the menu cells as follows:

gameSearch: 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: A Python module containing functions that prompt the store manager for the customer's ID and the ID of the games-record(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: A Python module containing functions that prompt the store manager for the ID of the game-record(s) they wish to return and collect feedback if applicable.

inventoryPruning: A Python module containing functions used to identify game-records 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: A Python module containing common functions that the games-record search, rent, return, and inventory pruning modules use to interact with the data files.

menu: The main cell 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

You can provide any special instructions or warnings to the user (or assessor!) 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) in the Markdown cells of the notebook.

All the files (notebook and data files) should be compressed into a zip file and submitted electronically as directed. Using the incorrect version of a Jupyter notebook will result in loss of marks.

4 Notes on Expectations

You will be marked according to your overall achievement, and 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 notebook should show mastery of what you have been taught.

Design: Your notebook should be well structured for the task in hand so that it is as easy as possible for:
- A store manager 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, the code in a related application.

Clarity and Self-Documentation: Given the structure of your notebook, it should be as easy to read and understand as possible. 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 each cell does and how they should be used. Also state who (**your ID only** for anonymity) 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.

Restrictions:

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 Google Colab to prepare your work.
5) Your code must use **ONLY** standard Python libraries and MatPlotLib.
6) You must only use iPyWidgets for your GUI.
7) Briefly discuss **five** security CODE issues.

GOOD LUCK!
Professor Stephen Lynch NTF FIMA SFHEA