Tkinter Application with DB Activity

Create a Moviehouse app that allows the user to register a customer and reserve a room for them. For this app, create a database with 4 tables:

- a table named `room` which contains the following columns:
    - id (Primary Key)
    - cost

NOTE: Manually enter 4 data into this table. The cost can be any amount.

- a table named `movie` which contains the following columns:
    - id (Primary Key)
    - title of type VARCHAR
    - genre of type VARCHAR
    - is_deleted of type TINYINT, INTEGER, or BOOLEAN with default value set to 0.
    - cost of type real or double
- a table named `room_record` which contains the following columns:
    - id (Primary Key)
    - room_id (Foreign Key)
    - total_cost of type real or double
    - is_finished of type TINYINT, INTEGER, or BOOLEAN with default value set to 0.
- a table named `room_movie_record` which contains the following columns:
    - id (Primary Key)
    - movie_id (Foreign Key)
    - room_record_id (Foreign Key)

NOTE: Use Sqlite database and sqlite3 python module for this task. Then follow the step-by-step instructions below.

1.) Create a file named `classes.py` which contains the following classes:

- A class called ***Movie*** which contains the following attributes:
    - id
    - title
    - genre
    - cost

This class should also contain a constructor that contains a parameter for each attributes. It should also contain getters and setters. This class should also override the `__str__` function to display the movie's id and title, separated by a dash, just like in the example below.

```
1 - The Movie
```

- A class called ***Room*** which contains the following attributes:
    - id
    - cost

- A class called ***Record*** which contains the following attributes:
    - id
    - room_id
    - total_cost
    - movies - a list of **Movie**

Both **Room** and **Record** classes should contain a constructor that contains a parameter for each attributes. They should also contain getters and setters.

2.) Create a file named `database_manager.py` which contains the following class:

- A class called ***MovieHouseDatabaseManager*** which has a single attribute called *database_file* of type String and has the following function:
    - a constructor that has a parameter of type String called *database_file*.
    - a function named `get_connection`. This function should allow the app to connect to the database using the 3 attributes. This function should be used throughout the class to connect to the database.
    - a function named `register_movie` which has a parameter of `title`, `genre` and `cost`. This function will save the values to the `movie` table in their corresponding columns. If the student was saved successfully, this function should return true. Otherwise, return false.
    - a function named `remove_movie` which has a parametor of `id`. This method will update the is_deleted column of the movie with the same id as the parameter to true.
    - a function named `retrieve_movies` which has a parameter of `genres`, which is a List of strings. This function will retrieve all movies that has genres found in the genres parameter. If the array is empty, this function will instead retrieve all movies. This will return a list of **Movie**.

    - a function named retrieve_rooms. This function will retrieve all rooms from the database. This will return a list of **Movie**.
    - a function named retrieve_record which has a parameter of room_id. This function will retrieve the latest record of that room that is not yet finished. If there are no records found, this will return an empty **Record** object. This function will also retrieve all movies associated with that record.
    - a function named check_in which has a parameter of room_id and movies, which is a List of **Movie** objects. This function will create a room_record for with the parameters as its values. This will also create a room_movie_record for each movie in the movies list. This function will return true if the record was successfully created. Otherwise, return false.
    - a function named check_out which has a parameter of id. This function will update the is_finished column of the record with the same id as the parameter. This function will return true if the record was successfully updated. Otherwise, return false.

3.) Lastly, create the main.py which should contain the following classes:

- A class called ***RecordWindow*** which inherits from the *Tkinter Window* class and contains the following attributes:
    - room - a **Room** object
    - db_manager - a **MovieHouseDatabaseManager** object
    - record - a **Record** object

This class should also contain the following components:

- 2 listboxes named movies_list and movies_to_view_list
- 4 buttons with the following functionality:
    - add_movie_button will add movies to movies_to_view_list
    - remove_movie_button will remove movies from movies_to_view_list
    - check_in_button will create a record for the room with its movies being the movies in movies_to_view_list y calling the *check_in* function of **MovieHouseDatabaseManager**
    - check_out_button will checkout the record of this class by calling the *check_out* function of **MovieHouseDatabaseManager** class.
- a label named total_cost_label which will display the total cost of the room + movies to view.

This class should have a constructor that has a parameter for each attributes and implement the following behaviors:

- `check_out_button` should be disabled and `check_in_button` should be enabled if there are no records. Otherwise, `check_out_button` should be disabled and `check_in_button` hould be enabled.

- A class named **MovieHouseWindow** which inherits from the Tkinter window class. This class should have a single attribute named `_database_manager` which is of type **MovieHouseDatabaseManager**. The class' constructor should have a parameter of `database_file_name` and initializes the `_database_manager` with the parameter.
    - 5 checkboxes used for filtering `movies_list` that has the following details:
        - a checkbox named `adventure_checkbutton` and labeled "Adventure".
        - a checkbox named `comedy_checkbutton` and labeled "Comedy".
        - a checkbox named `fantasy_checkbutton` and labeled "Fantasy".
        - a checkbox named `romance_checkbutton` and labeled "Romance".
        - a checkbox named `tragedy_checkbutton` and labeled "Tragedy".
    - 4 ***buttons*** named `room1_button`, `room2_button`, `room3_button`, and `room4_button`. When a button is clicked, retrieve the corresponding record on a room and then show the RecordWindow.
    - a listbox named `movies_list` which will display the movies retrieved from the database
    - 3 entries named `movie_title_entry`, `cost_entry`, and `genre_entry`.
    - 2 buttons that do the following:
        - `add_button` which will add the movie to the database with its title, cost, and genre being the values of their respective components.
        - `delete_button` which will delete the movie from the database that has the same id as the movie selected from the list.

This class should also contain the following components:
Both of these buttons will cause `movies_list` to refetch data from the database.

The app should have this general flow:

- View all movies in the database and add filters accordingly.

- Should be able to add and delete movies in the app.
- Should be able to display all buttons for the **Rooms** (4)
- Should be able to check in a customer by clicking a room which will display the record window.
- Should be able to check out a customer by clicking a room which will display the record window.

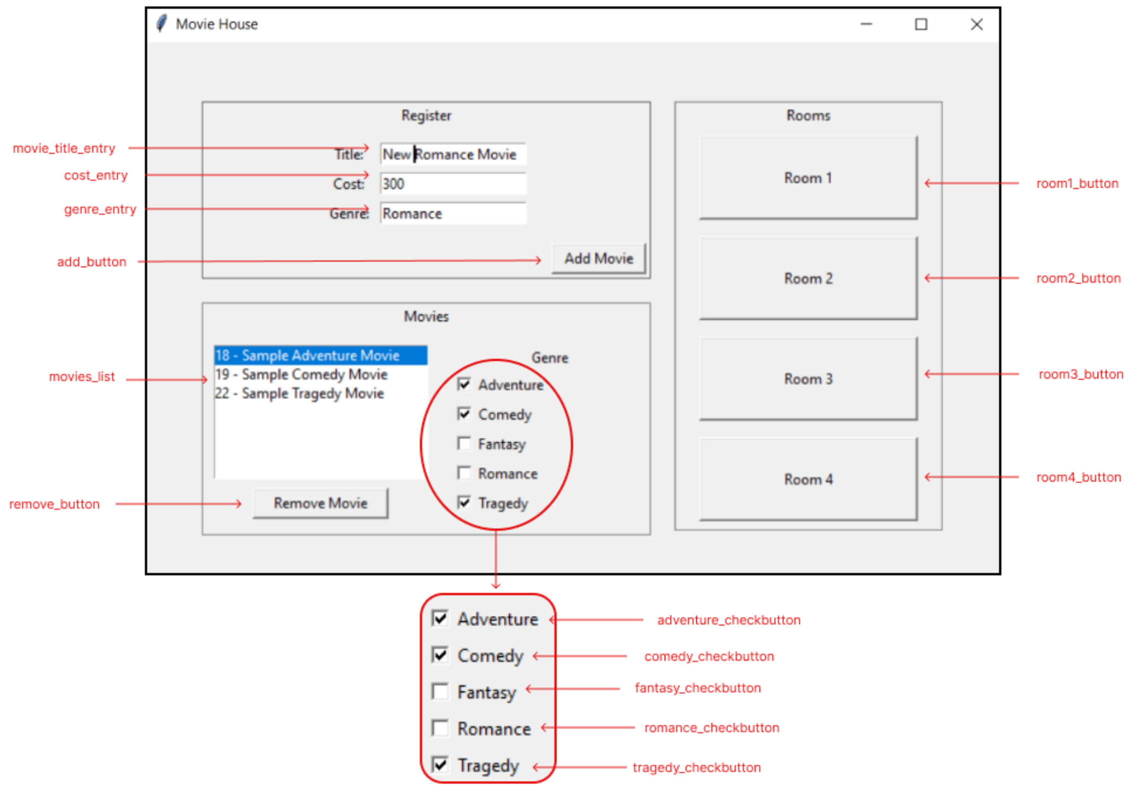
In addition, these following conditions should be followed:

- The app should retrieve all room and all movies data on app start. Then populate the necessary components of applicable.
- The RecordWindow should close on check in and check out.

Note: Do not include the mainloop() call in the constructors of both RecordWindow and MovieHouseWindow

Sample Output

MovieHouseWindow

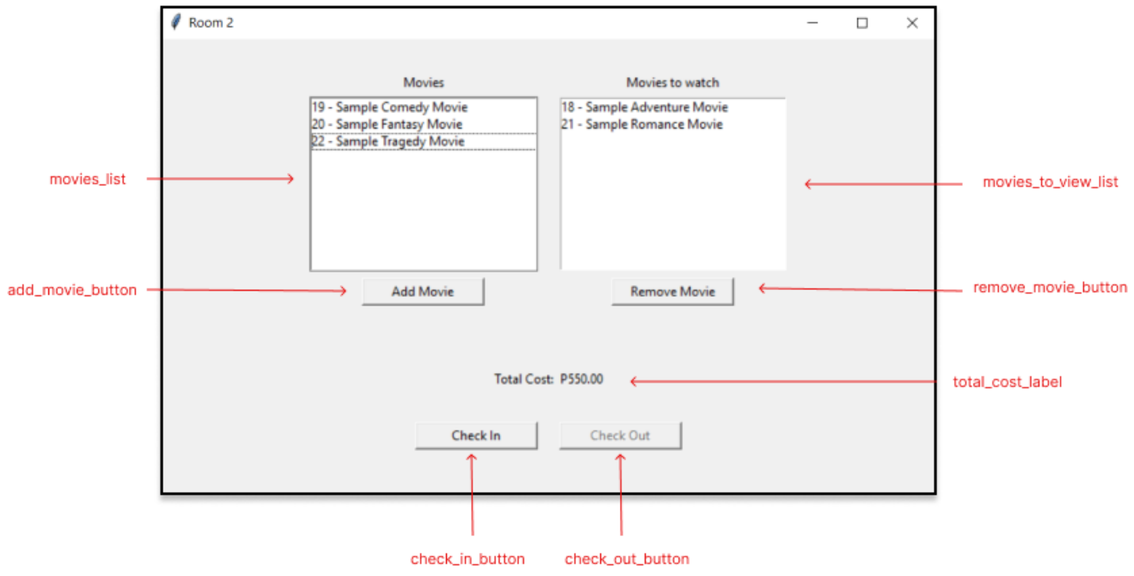


Title: New Romance Movie
Cost: 300
Genre: Romance

18 - Sample Adventure Movie
19 - Sample Comedy Movie
22 - Sample Tragedy Movie

[✓ Adventure]
[✓ Comedy]
Fantasy
Romance
[✓ Tragedy]

RecordWindow



Movies:
19 - Sample Comedy Movie
20 - Sample Fantasy Movie
22 - Sample Tragedy Movie

Movies to watch:
18 - Sample Adventure Movie
21 - Sample Romance Movie

Total Cost: P550.00