Project 3: GUI for Book Recommendations

CS 1410

Background

The purpose of this project is to get experience with event-driven programming with Graphical User Interfaces (GUIs). You will use the **breezypythongui** module, which is based on Python's **tkinter** module, to design interactive windows for the Book Recommendation System from Project 1. You will also add functionality to the system.

Important: You must use the *breezypythongui.py* file in Canvas, not the one in MindTap!

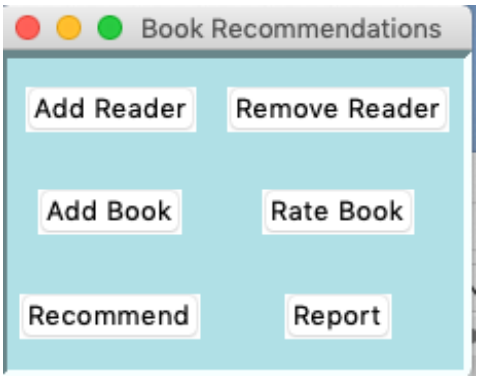
Requirements

Provide the following options to users, available to the user through simple buttons in a small main window. The corresponding windows/dialogs appear and disappear independently from the main window.

- **Add Reader**. This option receives the name of a new reader and initializes the new user's ratings for each book in the current book list to 0.
- **Add Book**. This option appends a new book (author and title) to the list of books and appends a rating of 0 to the ratings for *all readers*. (Remember, the connection between readers and books is by list position.) Use a dialog box for this to get all the data at once.
- **Rate Book**. This option prompts the user for the reader and the book. Allow these to be selected from list boxes. Then get the rating from the user and update your data structures accordingly. You can put all of this in a dialog.
- **Remove Reader**. This option removes a reader from the corresponding data structure. (We do not remove books in this assignment.)
- **Recommend**. This option opens a window that receives a reader name and then displays recommendations for that user in a message box. Validate that the reader exists in the system.
- **Report**. This option displays the same output from `bookrecs.main()` as you did in Project 1, but in a message box. Just use the default of `nfriends=2`.

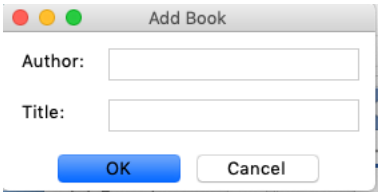
None of the options above modify the original *booklist.txt* and *ratings.txt* files. Additions and deletions of books or readers only affect the current execution.

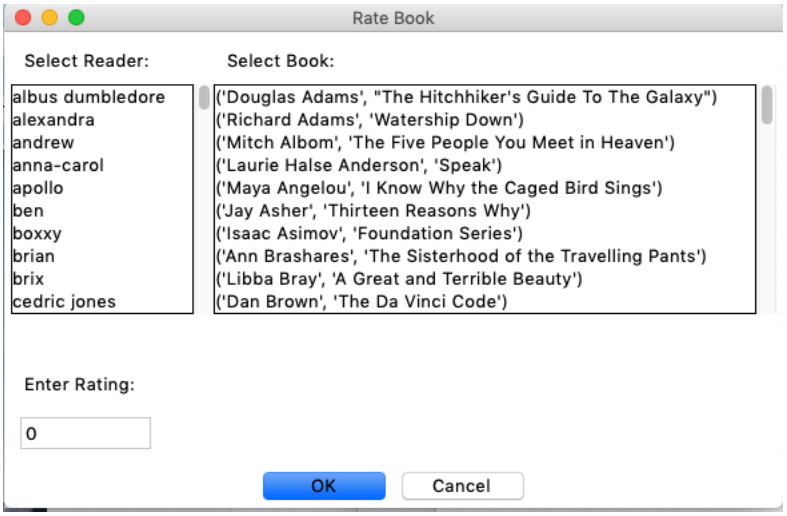
Here is a sample display, using "powder blue" ("#B0E0E6") as the background color:

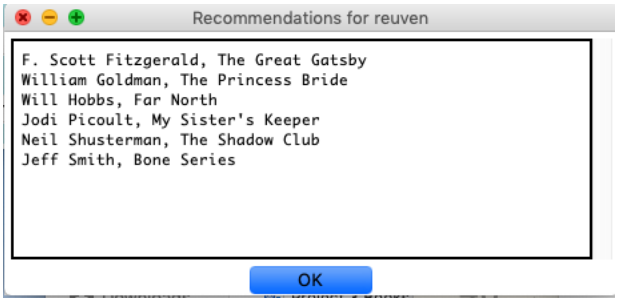


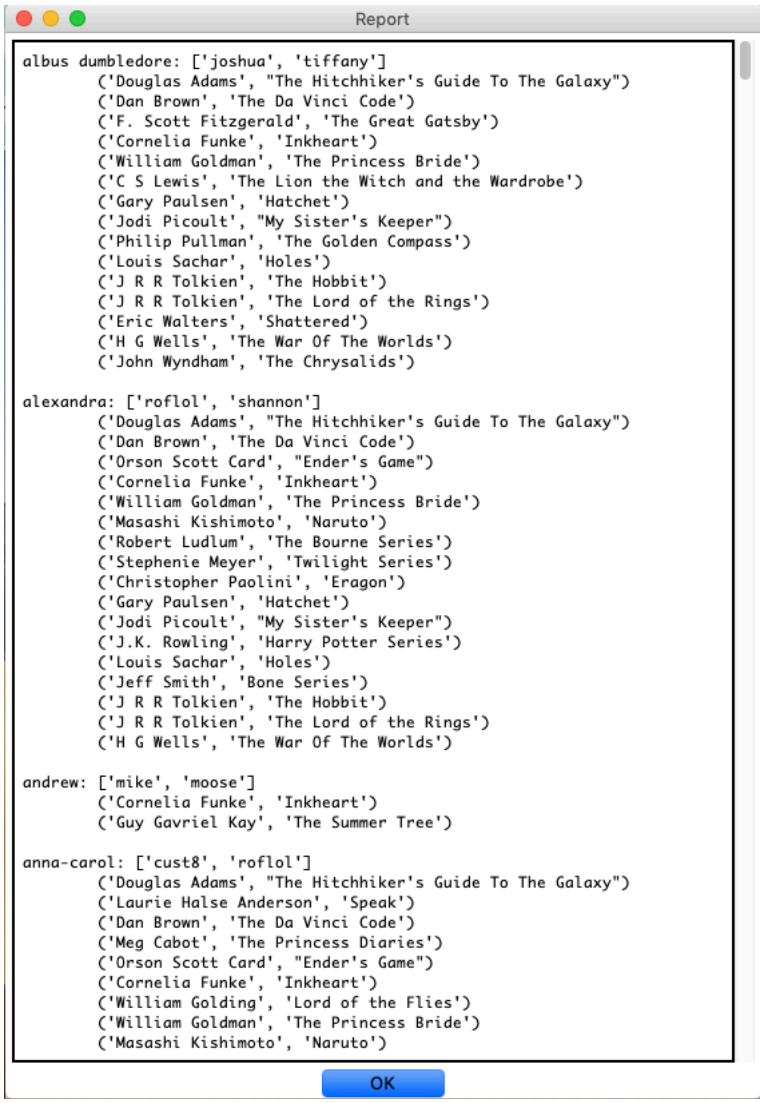
You can different background color codes at https://www.rapidtables.com/web/color/RGB_Color.html.

Here are some other suggested screens:









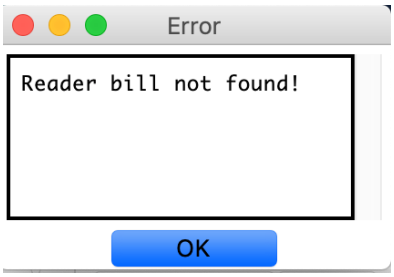
Implementation Notes

Implement Recommend and Report first, since these do not require any new logic as far as the data is concerned beyond what you did in Program 1.

It is important to keep the data-processing logic of a program independent from its user interaction as much as possible. Therefore, for most of the options above, you should have a *function* that modifies the data that is decoupled from the way the data is displayed. It should be possible to write a console-based program using your functions, independent from the GUI you design. This is sometimes called the Model/View architecture. The data model and associated logic is developed with little regard to how it will appear to a user on the screen, allowing the data and screen models to progress independently. For larger or more complex projects, there is a often third component that manages the communication between the model and the view, called Model/View/Controller architecture (MVC). Many web sites use MVC.

You may have to make changes to your bookrecs.py from Project 1 for this project. Your data structures that hold the books, ratings, and similarity scores need to be at the module level so you

can access them from your module for this project. Since you will be adding and/or removing readers and ratings, you will need a way to recompute the similarity scores, so that should be in a function in bookrecs.py that you call in this project's module. You will also need to be able to check to see if a reader exists when responding to Remove Reader or Recommend. Respond with message boxes when errors occur, such as:



If you have time, feel free to experiment a little and have some fun with this project. You can modify/enhance the user interface design as long as all required functionality is supported through a GUI. For example, you can use menus if you like instead of buttons.