Laurent Valle

Project Proposal Updated

-PROJECT SUMMARY-

Proposed project title:

June in Country Music

Longer description of the project:

This program is designed as a tool for fans of country music. They will easily figure out what concerts of country music are available in June, in which state they take place and how much they will cost. This project is trying to answer efficiently these three questions:

1. What are the different concerts available in June in each state contained in the database?
2. Where are the different concerts available for a specific day of June?
3. What concerts are available for a specific range of price during the month of June?

Technologies:

In order to make this program pleasant for the user, I will use a GUI designed as following:

* The title of the GUI.
* A label describing what the program does.
* Two menu buttons, one for the states and one for the different days of June.
* Each menu buttons are displaying their respective menus containing labeled check-buttons.
* Two output labels showing the user what check-button he selected in the menu.
* A sequence of four check buttons which allows the user to filter its search by price.
* One button activating the search in the database.
* One output text field displaying the results.
* One button clearing the text field.
* One button clearing the output labels and turning the check buttons off.

Moreover, a database is connected to the program. Several tools are required to make the connection and these steps will be described in the -DATA DESIGN- section.

-USE CASE ANALYSIS-

The file caseAnalysis.pptx presents the screen the user will encounter. Moreover, seven slides describe the path from the input part of the screen to the output part of the screen.

-UI DESIGN-

As drawn in the file caseAnalysis.pptx, the graphic interface will get the title “Country Concert Selector (Data provided by SeatGeek)” Then it will be designed as the following:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| self.title("Country Concert Selector (Data provided by SeatGeek)") | | | | | |
| Label.grid**(row = 0, column = 0)** -> “empty label” to get an empty row in the GUI | | | | | |
| Label(self, text = "June in Country Music", font = self.headerFont).grid**(row = 1, columnspan = 6)** | | | | | |
| Label(self).grid**(row = 2, column = 0)** -> “empty label” | | | | | |
| Label(self, width = 1).grid**(row = 3, column = 0)** -> “empty label” to get an empty column in the GUI | stateMenuButton.grid**(row = 3, column = 1)** | Label(self, width = 1).grid**(row = 3, column = 2)** | self.buttonShowResults.grid**(row = 3, column = 3, rowspan = 2)** | self.buttonClearResults.grid**(row = 3, column = 4, rowspan = 2)** | Label(self, width = 1).grid**(row = 3, column = 5)** |
| self.outputSate.grid**(row = 4, column = 1)** |
| Label(self).grid**(row = 5, column = 1)** ->”empty label” | self.textConcert.grid**(row = 5, column = 3, columnspan = 2, rowspan = 25)** | |
| dayMenuButton.grid**(row = 6, column = 1)** |
| self.outputDay.grid**(row = 7, column)** |
| Label(self).grid**(row = 8, column = 1)** -> “empty label” |
| Label(self, text = "Price:", font = self.headerFont).grid**(row = 9, column = 1)** |
| priceCheckButton20.grid**(row = 10, column = 1)** |
| priceCheckButton100.grid**(row = 11, column = 1)** |
| priceCheckButton200.grid**(row = 12, column = 1)** |
| priceCheckButton400.grid**(row = 13, column = 1)** |
| Label(self).grid**(row = 14, column = 1) ->** “empty Label” |
| self.buttonClearSelection.grid**(row = 15, column = 1, rowspan = 2)** |
| Label(self, **height = 10**).grid**(row = 17, column = 1)** |
|  |  |

The button “Show Results” will display the results according the previous inputs. The button “Clear Results” will reinitialize the text field of the graphic interface and the button “Clear Selection” will reinitialize the two output label fields and will turn the check buttons off.

-DATA DESIGN-

The best way to represent these data is through a database. The diagram of the basic data structure and the SQLite database is presented in the file caseAnalysis.pptx.

My database is composed of three tables:

* The first table, the main one, is grouping all data required by the program.
* The second table is grouping the states.
* The third table is grouping the day of the month of June.

-ALGORITHM-

Considering my entire program will be a subclass of the build-in class Tkinter, class ConcertApp(Tk), my main object will be the instance of class app = ConcertApp(), which I will create in the main function to run the program.

* Define the data members - what are the key data elements of the class?

My data members will be designed into their method and will be assigned through the state of the check buttons. Each check buttons with a state “ON,” will differently direct the cursor in the database countryMusic.db

* Describe the initializer - Initializers always create and populate the data members. Will you read in parameters? Have default values? both?

However, all the methods interacting with the user interface will be initialized in the constructor for passing control to the methods. The only parameter I will have is the parameter “self” and no default value will be necessary considering that the only inputs available will be determined by the different state of the graphic interface check buttons.

* Describe any other housekeeping that may need to happen in the initializer:

All widgets will appear in the initializer.

* Define access methods for all data members. Build appropriate getters and setters:

According the program used as example gradCalc.py, the modification and the return of the value of properties are done through the user interface and the methods interacting with it (assignement depending on the state of the check buttons and return trough the text field and output labels). Thus, I am not sure how to set up getters, setters and virtual property in a subclass of the build-in class Tkinter.

* Identify any methods your class will need beyond access modifiers:

The different methods are:

* def stateMenu(self):
* def dayMenu(self):
* def priceMenu(self):
* def outputText(self):
* def addButton(self):
* def clearResult(self):
* def clearSelection(self):
* def selectionDisplayer(self):
* def resultDisplayer(self):
* Flesh out each method just like the function analysis below:
  + 1. The user opens the dropdown list of the menu button “State.”
    2. Clicking on the check button labeled with the name of the state, the user turns this check button ON and he also actions the command directing the program in the method def selectionDisplayer(self). Then, the user can immediately visualize into the output label just below the menu button, the name of the state he selected.
    3. The user does not action any check buttons of the menu of the menu button “Day”. All these check buttons are OFF.
    4. The user does not action any check buttons related to the range price selection. All these check buttons are OFF.
    5. Clicking the button “Show Results”, the user sends the program to the method def resultDisplayer(self). The program retrieves the state of each check buttons:
       - The selected check button of the menu of the menu button “State” is ON and its onvalue is the integer representing its index in the table stateTable.
       - The other check buttons are OFF and their offvalue are the integer zero.
    6. When check buttons are OFF, the program returns to the output label the text “no xxx selected.”
    7. Using SQL language, the program uses the onvalue to retrieve from the table stateTable, the state related to the index. It returns a string.
    8. Using SQL language, the program uses this string to retrieve from the table concertTable, all data required by the execution of the cursor.
    9. The contain of the cursor is unloaded according the function format() in the text field of the graphic interface.
    10. The user can read all concert taking place in the selected state during the month of June.
    11. Clicking the button “Clear Results,” the user send the program to the method def clearResults() and delete the contain of the text field.
    12. The user narrows his selection and select a day. In other word, he turns ON one of the check button of the menu of the menu button “Day”.
    13. Clicking the button “Show Results”, the user sends the program to the method def resultDisplayer(self). The program retrieves the state of each check buttons:
        - The selected check button of the menu of the menu button “State” is ON and its onvalue is the integer representing its index in the table stateTable.
        - The selected check button of the menu of the menu button “Day” is ON and its onvalue is the integer representing its index in the table dayTable.
        - The other check buttons are OFF and their offvalue are the integer zero.
    14. Using SQL language, the program uses the onvalue to retrieve from the table stateTable, the state related to the index. It returns a string.
    15. Using SQL language, the program uses the onvalue to retrieve from the table stateTable, the state related to the index. It returns a string.
    16. Using SQL language, the program uses these two strings to retrieve from the table concertTable, all data required by the execution of the cursor.
    17. The contain of the cursor is unloaded according the function format() in the text field of the graphic interface.
    18. The user can read all concert taking place in the selected state on the selected day during the month of June.
    19. Clicking the button “Clear Results,” the user send the program to the method def clearResults() and delete the contain of the text field.
    20. The user narrows his selection and selects a range price. In other word, he turns ON one of the check button of the range price selection.
    21. Clicking the button “Show Results”, the user sends the program to the method def resultDisplayer(). The program retrieves the state of each check buttons:
        - The selected check button of the menu of the menu button “State” is ON and its onvalue is the integer representing its index in the table stateTable.
        - The selected check button of the menu of the menu button “Day” is ON and its onvalue is the integer representing its index in the table dayTable.
        - One of the check button of the price area is ON and its onvalue is the maximum integer of its respective range
    22. Using SQL language, the program uses the onvalue to retrieve from the table stateTable, the state related to the index. It returns a string.
    23. Using SQL language, the program uses the onvalue to retrieve from the table stateTable, the state related to the index. It returns a string.
    24. The program recognizes in the list listMaxPrice the onvalue of the price range check button activated and returns its index from the list.
    25. Using SQL language, the program uses these two strings and the operators of comparison listMaxprice[index -1] < item of the table concertTable < listMaxPrice[index] to retrieve from the table concertTable, all data required by the execution of the cursor.
    26. The contain of the cursor is unloaded according the function format() in the text field of the graphic interface.
    27. The user can read all concert taking place in the selected state on the selected day, in the selected range price, during the month of June.
    28. Clicking the button “Clear Results,” the user send the program to the method def clearResults() and delete the contain of the text field.
    29. Clicking the button “Clear Selection,” the user send the program to the method def clearSelection(). Then, the program clears the two outputs label and turns all check buttons OFF.
    30. The user is ready for starting a new search.
    31. The user can proceed this search in any order he wants and display the results after each time he turns a check button ON.