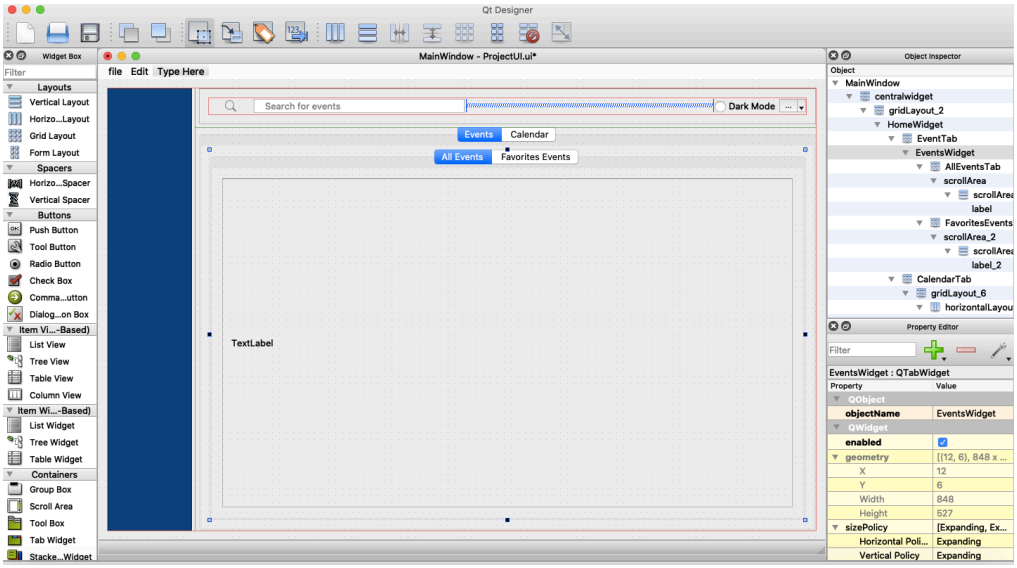
Abdullah Alshuaibi

CS 441

12/01/2019

Midterm Project Writeup

1. When we started our project, we decided to write it in python due to the easiness and capability of this language. I had little knowledge about python. But by the end of our project, I learned great things about this language. Also, beside that most of my work was on the UI. At first, we used Tkinter but it provides basic GUI. So, we had to use pyqt5 which has more capability of providing professional GUI. The good thing of pyqt5 is that it has a designer where we didn't have to code the user interface codes. So, I learned how to use the designer and used that knowledge to design our user interface. And then I used a special command to convert the designer file to python codes (pyuic5 -x test.ui -o test.py).



Also, I learned some stuff of what my teammates did. For example, I learned how to get some content from html pages using Beautifulsoup and Selenium. At design phase before we started the implementation phase, I knew some knowledge about UML from database class, but it was

worth knowing some new information about it. it was helpful. However, the new and important thing I learned was the use cases. They were really important before we started coding to know how the user would interact with the app.

2. I helped the group with the UI. Whenever someone needed a button or any part of the GUI, I did it for him with a nice layout for implementing it with his code.

3. We used use cases to split the requirements between us. But unfortunately, we didn't have time to finish some requirements. At the design phase, I did these use cases:

User Search & Filter

- 10.1 User click on the search bar
- 10.2 System enables user to type
- 10.3 User types the event he/she wants to go
- 10.4 System shows several events based on the user's request
- 10.5 If user wants to use filter
  - 10.5.1 User clicks on the filter button
  - 10.5.2 System opens a popup window with several search choices
  - 10.5.3 User selects one of the choices and then clicks done
  - 10.5.4 system shows several filtered events
- 10.5 User selects an event from the list
- 10.6 system opens the event page that was selected

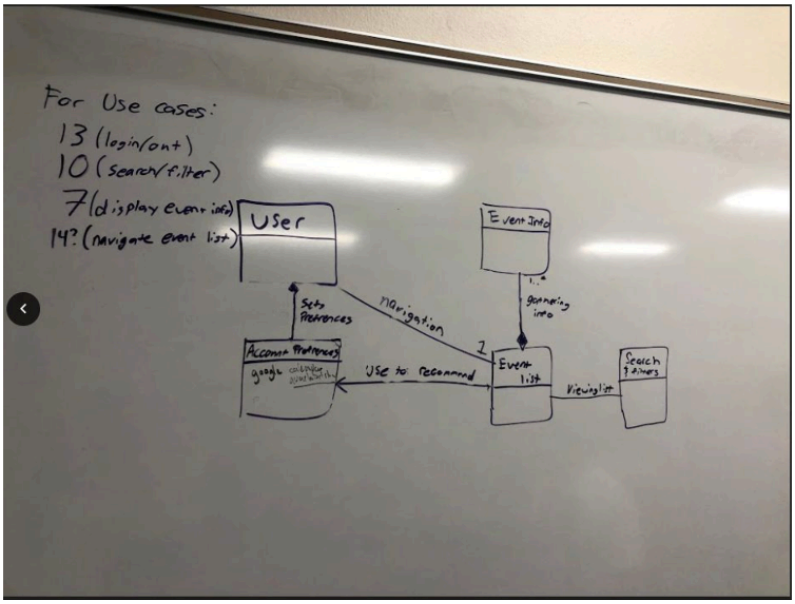
Rating & Reviewing

- 12.1 User selects an event
- 12.2 System Opens the event page that was selected
- 12.3 User click on Rating & Reviewing button
- 12.4 System opens the page of the event's rating and reviewing
- 12.5 User clicks on the review box
- 12.6 System enables user to type
- 12.7 User write a review
- 12.8 User rates the event from the five stars
- 12.9 System lights up the number of stars the user rates
- 12.10 User clicks on Done button
- 12.11 System sends the user to the event page

List & Grid View

- 11.1 When it's a list view
  - 11.1.1 User clicks on grid view icon
  - 11.1.2 System changes the events to grid view
- 11.2 When it's a grid view
  - 11.2.1 User clicks on list view icon
  - 11.2.2 system changes the events to list vie

4. The UML we came up with during the design phase is this one



5. The main thing I developed is the UI by using the designer. Each widget will be in a gird layout which makes them resizable when the user changes the size of the window. This design is converted to python codes in a separate file. I also did the favorite events page where when the user clicks on AddToFavorite button, the event will be added to the favorite page. And the user can do left click to get more info about the event or right click to remove the event from the favorite page. Also, when the user wants to add an event to favorite which exists in favorites, an error message will pop up to tell the user that the event already in favorites. Also, I did an additional feature which is dark mode. There is a radio button on the right top. When the user hits it, the darkMode method will be called, and if the button is on all of colors of the widgets, frames, labels, etc. will be set to dark colors. If the button is off, they will be set to default colors. Also, I helped connecting the events from the scraper with the labels.

6. We will see if we have time before the due date, we will try to do some of the use cases that we haven't done yet.

Here are some snippets of code I did or helped the group with.

```
from PySide2 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setWindowModality(QtCore.Qt.NonModal)
        MainWindow.resize(1034, 698)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())
        MainWindow.setSizePolicy(sizePolicy)
        MainWindow.setMinimumSize(QtCore.QSize(0, 0))
        MainWindow.setSizeIncrement(QtCore.QSize(0, 0))
        MainWindow.setBaseSize(QtCore.QSize(-1, 0))
        MainWindow.setTabShape(QtWidgets.QTabWidget.Rounded)
        MainWindow.setDockNestingEnabled(False)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setEnabled(True)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.centralwidget.sizePolicy().hasHeightForWidth())
        self.centralwidget.setSizePolicy(sizePolicy)
        self.centralwidget.setMinimumSize(QtCore.QSize(0, 0))
        self.centralwidget.setMouseTracking(False)
        self.centralwidget.setAutoFillBackground(False)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout_5 = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout_5.setObjectName("gridLayout_5")
        self.gridLayout_2 = QtWidgets.QGridLayout()
```

```
class MyQtAPP(ProjectUI.Ui_MainWindow, QtWidgets.QMainWindow):
    def __init__(self):
        super(MyQtAPP, self).__init__()
        self.setupUi(self)
        self.ui = ProjectUI2.Ui_EventWindow()
        self.window = QtWidgets.QMainWindow()

        #self.AddEventsCalndar.clicked.connect(self.GoogleAddEvent)
        self.PrintEventsCalendar.clicked.connect(self.GooglePrintEvent)

        self.labelList = []
        self.FavoritesLabelList = []
        self.FavoritesEventThere = []
        EventManager.FindEvents()
        self.EventPage()
        for i in range(len(self.labelList)):
            self.labelList[i].mousePressEvent = lambda event, x=i: self.oneEventInfo(x)

        self.DarkModeButton.toggled.connect(self.DarkMode)


    def EventPage(self):
        for e in EventManager.FullEventList:
            label = QtWidgets.QLabel(self.scrollAreaWidgetContents)
            label.setMinimumSize(QtCore.QSize(0, 40))
            label.setCursor(QtCore.Qt.PointingHandCursor)
            label.setStyleSheet("background-color: rgb(255, 255, 255);")
            label.setMargin(6)
            label.setWordWrap(True)
            label.setTextInteractionFlags(QtCore.Qt.LinksAccessibleByMouse | QtCore.Qt.TextSelectableByMouse)
            label.setObjectName("label")
            label.setText(e.name + "\n     " + e.date)
            self.verticalLayout_2.addWidget(label)
            self.labelList.append(label)
```

```
def oneEventInfo(self, i):
    self.ui.setupUi(self.window)
    EventManager.EventInfoDisplay(EventManager.FullEventList[i], self.ui.textBrowser,EventManager.FullEventList,i)
    self.window.show()
    self.ui.AddToFavoritesButton.clicked.connect(lambda: self.AddToFavorites(i))
    self.ui.AddToCalendarButton.clicked.connect(lambda: self.GoogleAddEvent(i))
```

```
def AddToFavorites(self, i):
    for x in self.FavoritesEventThere:
        if x == i:
            msg = QtWidgets.QMessageBox()
            msg.setIcon(QtWidgets.QMessageBox.Critical)
            msg.setText("This Event Is Already in Favorites")
            msg.setWindowTitle("Favorite Events")
            msg.exec_()
            return

    label = QtWidgets.QLabel(self.scrollAreaWidgetContents_2)
    label.setMinimumSize(QtCore.QSize(0, 40))
    label.setCursor(QtCore.Qt.PointingHandCursor)
    label.setStyleSheet("background-color: rgb(255, 255, 255);")
    label.setMargin(6)
    label.setWordWrap(True)
    label.setTextInteractionFlags(QtCore.Qt.LinksAccessibleByMouse | QtCore.Qt.TextSelectableByMouse)
    label.setObjectName("label")
    label.setText(EventManager.FullEventList[i].name + "\n    " + EventManager.FullEventList[i].date)
    self.verticalLayout.addWidget(label)
    self.FavoritesLabelList.append(label)
    self.FavoritesEventThere.append(i)

    msg = QtWidgets.QMessageBox()
    msg.setIcon(QtWidgets.QMessageBox.Information)
    msg.setText("The Event Was Added to Favorites")
    msg.setWindowTitle("Favorite Events")
    msg.exec_()

    for i in range(len(self.FavoritesEventThere)):
        self.FavoritesLabelList[i].mousePressEvent = lambda event, y=i, z=self.FavoritesEventThere[i]: self.RemoveFromFavorites(event, y, z)

    self.DarkMode()
```

```
def RemoveFromFavorites(self,event, y, z):
    if event.button() == QtCore.Qt.RightButton:
        msg = QtWidgets.QMessageBox()
        msg.setIcon(QtWidgets.QMessageBox.Question)
        msg.setText("Do You Want To Remove This Event?")
        msg.setWindowTitle("Favorite Events")
        msg.setStandardButtons(QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No)
        msg.setDefaultButton(QtWidgets.QMessageBox.No)
        reply = msg.exec_()
        if reply == QtWidgets.QMessageBox.Yes:
            self.FavoritesLabelList[y].hide()
            self.FavoritesLabelList.pop(y)
            self.FavoritesEventThere.pop(y)
        else:
            return
    elif event.button() == QtCore.Qt.LeftButton:
        msg = QtWidgets.QMessageBox()
        msg.setIcon(QtWidgets.QMessageBox.Question)
        msg.setText("Do You Need More Info about This Event?")
        msg.setWindowTitle("Favorite Events")
        msg.setStandardButtons(QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No)
        msg.setDefaultButton(QtWidgets.QMessageBox.No)
        reply = msg.exec_()
        if reply == QtWidgets.QMessageBox.Yes:
            self.oneEventInfo(z)
        else:
            return
```

```
def DarkMode(self):
    if self.DarkModeButton.isChecked() == True:
        self.centralwidget.setStyleSheet("background-color: rgb(51, 51, 51);")
        self.frame.setStyleSheet("background-color: rgb(25, 25, 25);\n"
                                 "border-color: rgb(204, 204, 204);\n"
                                 "")
        self.SearchButton.setStyleSheet("background-color: rgb(204, 204, 204);")
        self.searchBar.setStyleSheet("background-color: rgb(127, 127, 127);\n"
                                     "color: rgb(230, 230, 230);\n"
                                     "selection-background-color: rgba(102, 204, 255, 243);")
        self.DarkModeButton.setStyleSheet("color: rgb(230, 230, 230);\n"
                                          "")
        self.toolButton.setStyleSheet("background-color: rgb(204, 204, 204);")
        self.HomeWidget.setStyleSheet("background-color: rgb(76, 76, 76);")
        self.AllEventsTab.setStyleSheet("background-color: rgb(76, 76, 76);")
        self.scrollArea.setStyleSheet("background-color: rgb(76, 76, 76);")
        self.scrollAreaWidgetContents.setStyleSheet("background-color: rgb(76, 76, 76);")
        self.scrollArea_2.setStyleSheet("background-color: rgb(76, 76, 76);")
        self.scrollAreaWidgetContents_2.setStyleSheet("background-color: rgb(76, 76, 76);")
        self.textBrowser_2.setStyleSheet("background-color: rgb(128, 128, 128);\n"
                                         "color: rgb(255, 255, 255);")
        #self.AddEventsCalndar.setStyleSheet("background-color: rgb(128, 0, 64);\n"
        #                                    "color: rgb(255, 255, 255);")
        self.PrintEventsCalendar.setStyleSheet("background-color: rgb(128, 0, 64);\n"
                                               "color: rgb(255, 255, 255);")
        self.frame_2.setStyleSheet("background-color: rgb(25, 25, 25);")
        for i in self.labelList:
            i.setStyleSheet("background-color: rgb(128, 128, 128);\n"
                            "color: rgb(255, 255, 255);")
        for i in self.FavoritesLabelList:
            i.setStyleSheet("background-color: rgb(128, 128, 128);\n"
                            "color: rgb(255, 255, 255);")

    else:
        self.centralwidget.setStyleSheet("default")
        self.frame.setStyleSheet("default")
        self.SearchButton.setStyleSheet("default")
```

Links:

https://www.youtube.com/watch?v=FVpho_UiDAY
https://www.youtube.com/watch?v=Dmo8eZG5I2w&t=618s

https://www.youtube.com/watch?v=dRRpbDFnMHI&t=575s

https://www.w3schools.com/python/python_lambda.asp