Lab 2: Building Desktop Applications with Multiple Forms in PyQt6

CS355/CE373 Database Systems
Habib University
Fall 2023

1 Instructions

- This lab will contribute 1% towards the final grade.
- The deadline for this lab is one week after your lab ends. **However, you must also submit all the work completed in the lab at the end of the lab.**
- The lab must be submitted online via CANVAS. You are required to submit a zip file that contains both *.py* and *.ui* files.
- The zip file should be named as *Lab_02_aa1234.zip* where *aa1234* will be replaced with your student id.
- **Files that don't follow the appropriate naming convention will not be graded.**

1.1 Marking scheme

This lab will be marked out of 100.

- 50 Marks are for the completion of the lab.
- 10 Marks are for filling the feedback form within the lab timings.
- 40 Marks are for progress and attendance during the lab.

1.2 Late submission policy

No late submissions are allowed for this lab.

2 Objective

The objective of this lab is to enable students to work with multiple GUI forms in their application and exchange data across them. This lab will cover an example that will demonstrate how to transfer data between two forms. The main task of this lab is to build a library management system.

3 Example: Transferring the Data Between Multiple Forms

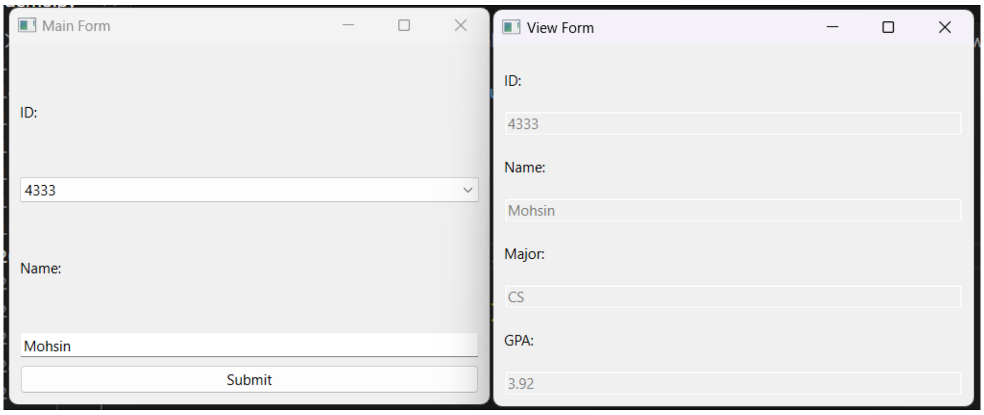
In this example, we will consider an application that consists of two forms: MainForm and ViewForm. Once the application is started, the MainForm will load. In the MainForm, the student will enter the Name and ID, and then click on submit. Once the submit button is clicked, the GPA and Major of the student along with Name and ID, will be displayed in the ViewForm which is a read-only form.

In this application, the data will be fetched from a hardcoded Python List containing 3 records which can be seen in Table 1

Name	ID	Major	GPA
Ahmed	4289	CS	3.85
Hammad	4305	CS	3.53
Mohsin	4333	CS	3.92

Table 1: Student Entries

In the MainForm, the ID will be a combo box field. Once the ID is selected, the name text box will be populated. Furthermore, once the submit button is clicked all the corresponding details will be transferred to the ViewForm, which will then display them in a view-only form. The MainForm and ViewForm can be viewed in Figure 1



4333

Mohsin

4333

Mohsin

CS

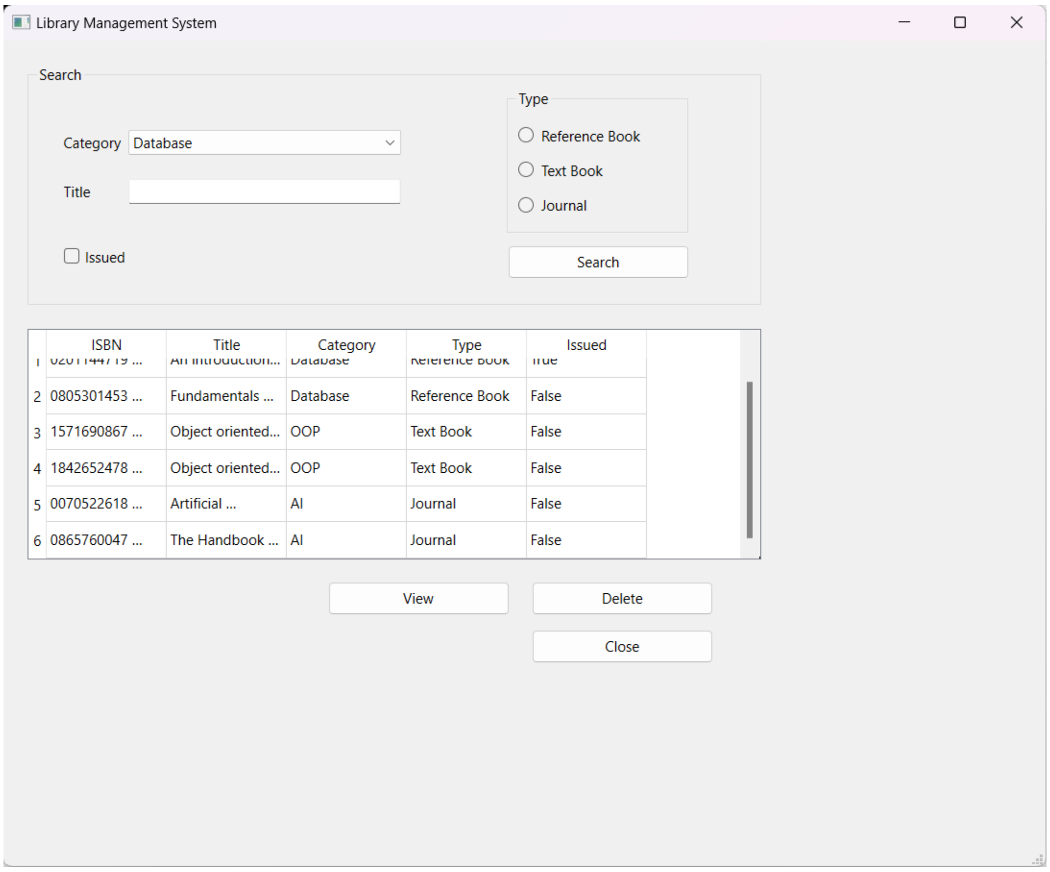
3.92

Figure 1: Main Form & View Form in PyQt6

The code for this sample application can be found in the **Lab 2 - Building Desktop applications with multiple forms in PyQt6** module of your lab section on LMS in the file `demo.py`. The code for this application can also be viewed Listing 1

4 Exercise

In this exercise, you have to develop a Library Management System that will allow the users to search for books by providing criteria, view the details of a particular book, and delete the book from the system.



Category: Database

Title:

☐ Issued

○ Reference Book
○ Text Book
○ Journal

	ISBN	Title	Category	Type	Issued
1	0201144719 ...	An introduction...	Database	Reference Book	True
2	0805301453 ...	Fundamentals ...	Database	Reference Book	False
3	1571690867 ...	Object oriented...	OOP	Text Book	False
4	1842652478 ...	Object oriented...	OOP	Text Book	False
5	0070522618 ...	Artificial ...	AI	Journal	False
6	0865760047 ...	The Handbook ...	AI	Journal	False

Figure 2: Library Management System using PyQt6

The Book Search Form will have the following functionality.

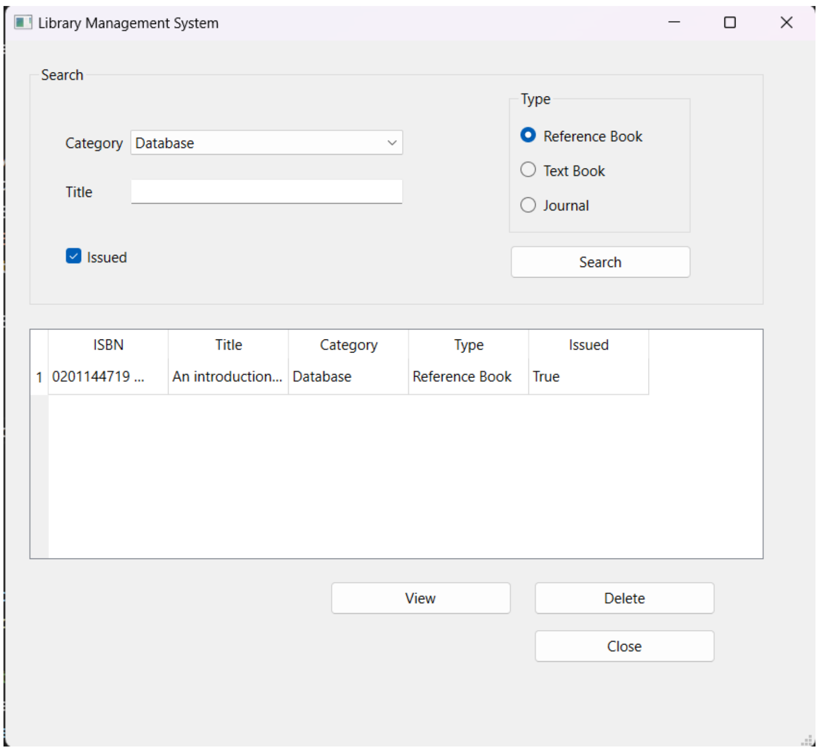
1. When the form is loaded, all the books which are stored in the **Python List** will be displayed in the table widget. The table widget has the following columns:

   - ISBN
   - Title
   - Category

- Type
- Issued

2. The Category combo box will show the following options:
   - Database
   - OOP
   - AI

3. Users can enter any search criteria, such as Category, Type, Title, and Issued, and then click ‘Search’ to find books that match those criteria. Only the books that match the search criteria will be loaded in the **table** widget. For example, if the Category is **Database**, the Title field is empty, the Type is Reference Book and the Issued Checkbox is Checked, the following will be the search result



Category: Database

Title:

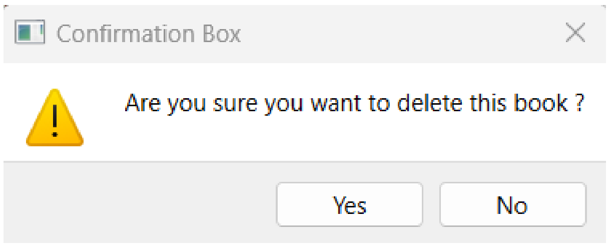
Type: [✓ Reference Book] Text Book Journal

[✓ Issued]

	ISBN	Title	Category	Type	Issued
1	0201144719 ...	An introduction...	Database	Reference Book	True

Figure 3: Search Output

4. The 'Delete' button will request the user for confirmation of deletion as follows:

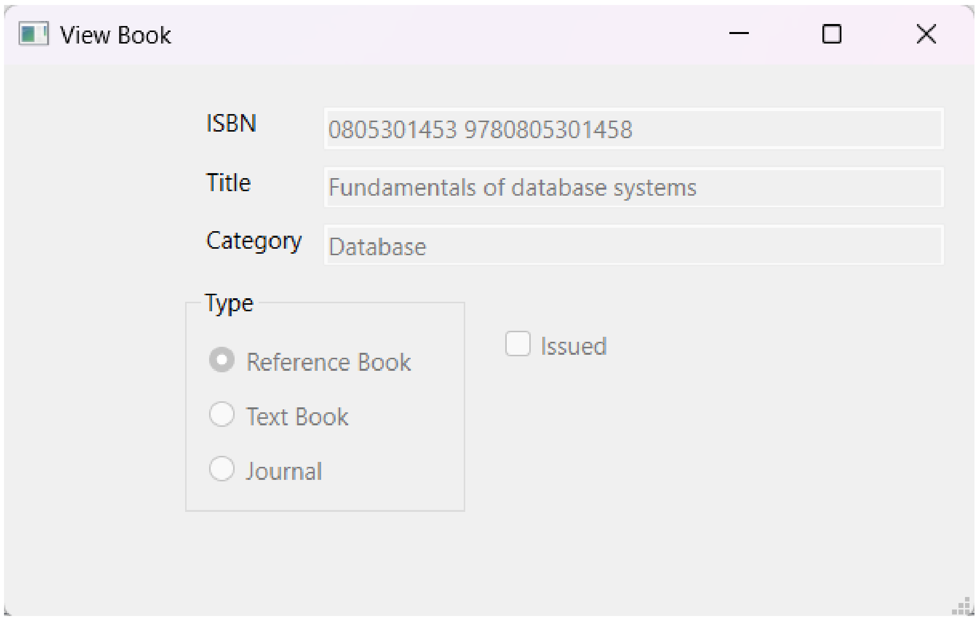

Confirmation Box

Are you sure you want to delete this book ?

Figure 4: Confirmation Window for Deleting a Book

The book will be deleted if you choose 'Yes', but choosing 'No' will keep it in the system.

5. The view button will redirect the user to the View Book form, where the book's details will be displayed. The View button will open the View Book form on top of the Search Book form. In addition, the book details will be displayed in view-only mode and the user cannot edit them.


View Book

ISBN: 0805301453 9780805301458

Title: Fundamentals of database systems

Category: Database

Type: [✓ Reference Book] Text Book Journal

☐ Issued

Figure 5: View Book Form in PyQt6

6. The application will close upon clicking the close button.

Since you have not yet studied database connectivity, the details of all the books are hardcoded in a Python List **books**. Furthermore, the code for populating the Table widget once the application is run is provided for you and can be accessed from **Lab 2 - Building Desktop applications with multiple forms in PyQt6** module of your lab section on LMS in the file `app.py`. You are required to work on the same file (`app.py`) in this lab and implement the following functions:

1. The **search** function implements the search functionality. In this function, the user selection is gathered from the Title, Category, Type, and Issued fields. Afterward, the data is filtered on the basis of the user selection. Finally, the filtered data is then displayed on the table widget.
2. The **view** function implements the view book functionality. It first checks if a row in the table widget is selected, else it does not do anything. If a row is selected, the book details are extracted from the row and then passed on to the ViewBook Form and then displayed (Refer to the code in Listing 1).
3. The **delete** function implements the delete book functionality. It first checks if a row in the table widget is selected, else it does not do anything. If a row is selected, then it asks the user asks for confirmation. If the user says yes, then the book is deleted from the **books** Python List, and all the book details are re-displayed on the table widget.
4. The **close** function implements the close form functionality. Once the close button is clicked, the entire application is closed.

Note: In order for your skeleton file `app.py` to execute successfully, you need to ensure that the UI file is named exactly ("Lab02.ui") and the objectName of the table widget is (booksTableWidget)

A Python Code File for the Sample Application

```python
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QWidget,
    QVBoxLayout, QLabel, QPushButton, QLineEdit, QRadioButton,
    QComboBox

# Hard coded Python for storing student entries
data = [["Ahmed","4289","CS",3.85],["Hammad","4305","CS"
    ,3.53],["Mohsin","4333","CS",3.92]]

class MainForm(QMainWindow):
    def __init__(self):
        super().__init__()

        # Set Window Title
        self.setWindowTitle('Main Form')
        # Set Dimensions
        self.setGeometry(100, 100, 400, 300)

        # Set Central Widget
        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        # Set Layout
        layout = QVBoxLayout()
        central_widget.setLayout(layout)

        # ID Combo Box
        self.id_label = QLabel('ID:')
        self.id_combo = QComboBox()
        self.id_combo.addItems([i[1] for i in data])

        # ID Label & Input
        self.name_label = QLabel('Name:')
        self.name_input = QLineEdit()
        self.name_input.setReadOnly(True)
        self.name_input.setText(data[0][0])

        # Submit Push Button
        self.submit_button = QPushButton('Submit')

        # Add Widgets to Layout
        layout.addWidget(self.id_label)
        layout.addWidget(self.id_combo)
        layout.addWidget(self.name_label)
        layout.addWidget(self.name_input)
        layout.addWidget(self.submit_button)

        # Connect Submit Button to Event Handling Code
        self.submit_button.clicked.connect(self.open_view_form)
        # Connect ID Combo Box to Event Handling Code
        self.id_combo.activated.connect(self.handle_id_toggle)

    def open_view_form(self):
        index = self.id_combo.currentIndex()
        # Get id of student
        id = data[index][1]
        # Get value from name_input
        name = data[index][0]
        # Get value from radio_button
        major = data[index][2]
        # Get value from combo_box
        gpa = data[index][3]

        # Pass all the data to view form as parameters
        self.view_form = ViewForm(id,name,major,gpa)
        self.view_form.show()

    def handle_id_toggle(self):
        id = self.id_combo.currentIndex()
        self.name_input.setText(data[id][0])

class ViewForm(QMainWindow):
    def __init__(self, id,name,major,gpa):
        super().__init__()
        # Receive Data from the Main Form
        self.name = name
        self.id = id
        self.major = major
        self.gpa = gpa


        # Set Window Title
        self.setWindowTitle('View Form')
        # Set Dimensions
        self.setGeometry(100, 100, 400, 300)

        # Set Central Widget
        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        # Set Layout
        layout = QVBoxLayout()
        central_widget.setLayout(layout)

        # ID Label & Input
        self.id_label = QLabel('ID:')
        self.id_input = QLineEdit()
        # Set ID Value
        self.id_input.setText(self.id)
        # Make ID read only
        self.id_input.setDisabled(True)

        # Name Label & Input
        self.name_label = QLabel('Name:')
        self.name_input = QLineEdit()
        # Set Name Value
        self.name_input.setText(self.name)
        # Make Name read only
        self.name_input.setDisabled(True)

        # Major Label & Input
        self.major_label = QLabel('Major:')
        self.major_input = QLineEdit()
        # Set Name Value
        self.major_input.setText(self.major)
        # Make Major read only
        self.major_input.setDisabled(True)

        # GPA Label & Input
        self.gpa_label = QLabel('GPA:')
        self.gpa_input = QLineEdit()
        # Set Name Value
        self.gpa_input.setText(str(self.gpa))
        # Make GPA read only
        self.gpa_input.setDisabled(True)

        # Add Widgets to Layout
        layout.addWidget(self.id_label)
        layout.addWidget(self.id_input)
        layout.addWidget(self.name_label)
        layout.addWidget(self.name_input)
        layout.addWidget(self.major_label)
        layout.addWidget(self.major_input)
        layout.addWidget(self.gpa_label)
        layout.addWidget(self.gpa_input)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    main_form = MainForm()
    main_form.show()
    sys.exit(app.exec())
```

Listing 1: Skeleton Python file (app.py)

B Skeleton File for Lab Task

```python
from PyQt6 import QtWidgets, uic
from PyQt6.QtCore import Qt
import sys

books = [
["0201144719 9780201144710","An introduction to database
    systems","Database","Reference Book","True"],
["0805301453 9780805301458","Fundamentals of database systems",
    "Database","Reference Book","False"],
["1571690867 9781571690869","Object oriented programming in
    Java","OOP","Text Book","False"],
["1842652478 9781842652473","Object oriented programming using
    C++","OOP","Text Book","False"],
["0070522618 9780070522619","Artificial intelligence","AI","
    Journal","False"],
["0865760047 9780865760042","The Handbook of artificial
    intelligence","AI","Journal","False"],
]

class UI(QtWidgets.QMainWindow):
    def __init__(self):
        # Call the inherited classes __init__ method
        super(UI, self).__init__()
        # Load the .ui file
        uic.loadUi('Lab02.ui', self)



        self.booksTableWidget.setRowCount(len(books))
        for i in range(len(books)):
            for j in range(5):
                item = QtWidgets.QTableWidgetItem(books[i][j])
                # Make the items non-editable
                item.setFlags(Qt.ItemFlag.ItemIsEnabled | Qt.
    ItemFlag.ItemIsSelectable)
                self.booksTableWidget.setItem(i,j,item)
        # Connect the search function with the search button.

        # Connect the view function with the view button.

        # Connect the delete function with the delete button.

        # Connect the close function with the close button.

    def search(self):
        # TO BE IMPLEMENTED
        pass

    def view(self):
        # TO BE IMPLEMENTED
        pass

    def delete(self):
        # TO BE IMPLEMENTED
        pass
    def close(self):
        # TO BE IMPLEMENTED
        pass


class ViewBook(QtWidgets.QMainWindow):
    pass
```

```python

app = QtWidgets.QApplication(sys.argv) # Create an instance of
    QtWidgets.QApplication
window = UI() # Create an instance of our
window.show()
app.exec() # Start the application
```

Listing 2: Skeleton Python file (`app.py`)