Lab 2: Building Desktop Applications with Multiple Forms in PyQt6

CS355/CE373 Database Systems
Fall 2025

Dhanani School of Science and Engineering
Habib University

Contents

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: Lab completion.
- 50 marks: Progress and attendance (validated via viva).

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

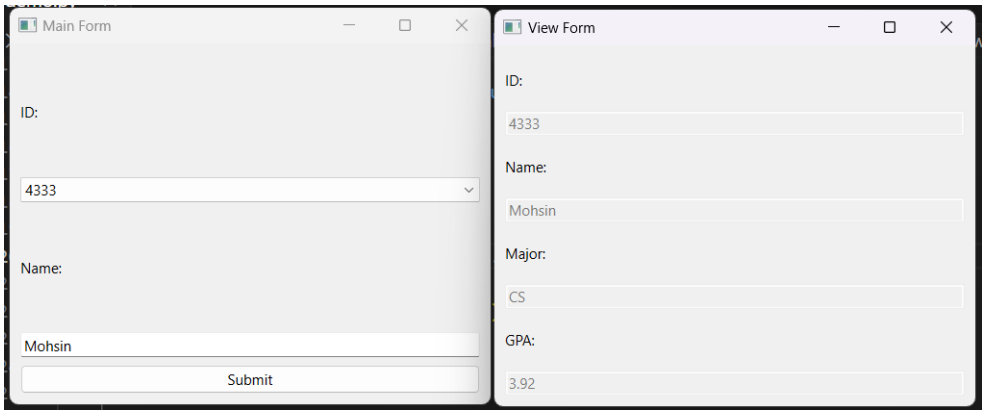

Figure 1: Main Form & View Form in PyQt6

The code and ui 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 under the **Example** header. The code for this application can also be viewed in Listing 1

4 Exercise

For the following exercise, you are provided 3 files that can be accessed from **Lab 2 - Building Desktop applications with multiple forms in PyQt6** module of your lab section on LMS under the **Exercise** header. You are required to work with them only.

- `Lab02.ui` (no need to edit, this is the main form)
- `view.ui` (no need to edit, this is the view form)
- `app.py` (This is the skeleton file which you will edit. Only make changes where it is indicated by `...`)

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.

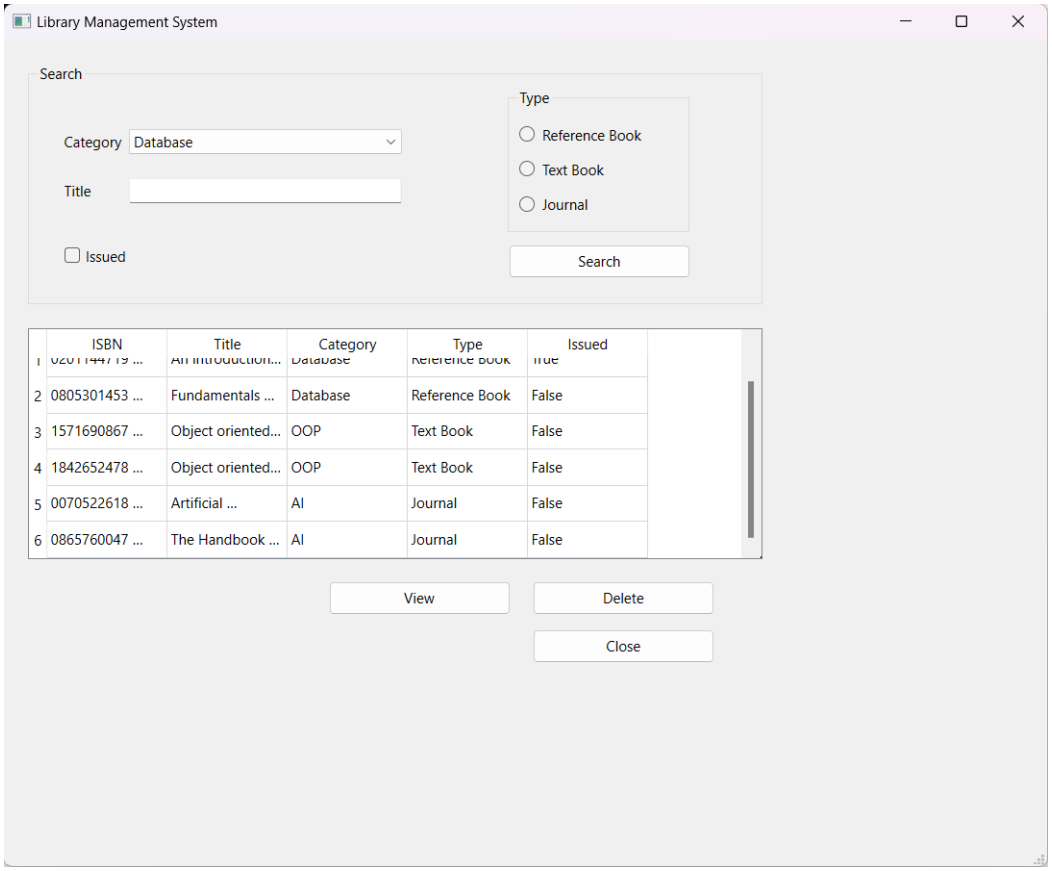

Figure 2: Library Management System using PyQt6

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.

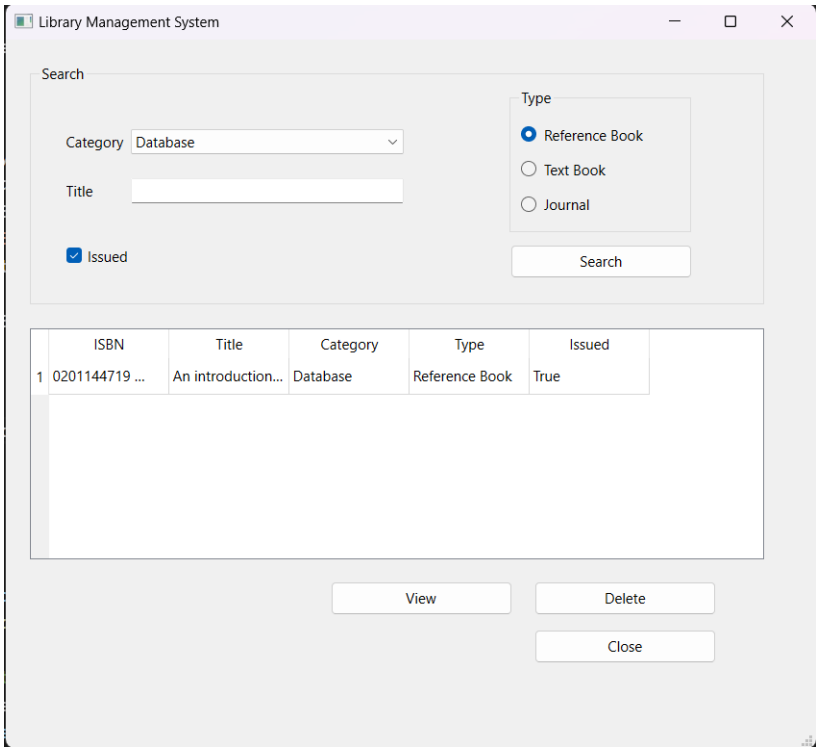

Figure 3: Search Output

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

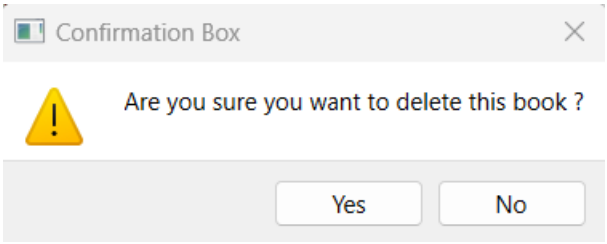

Figure 4: Confirmation Window for Deleting a Book

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.

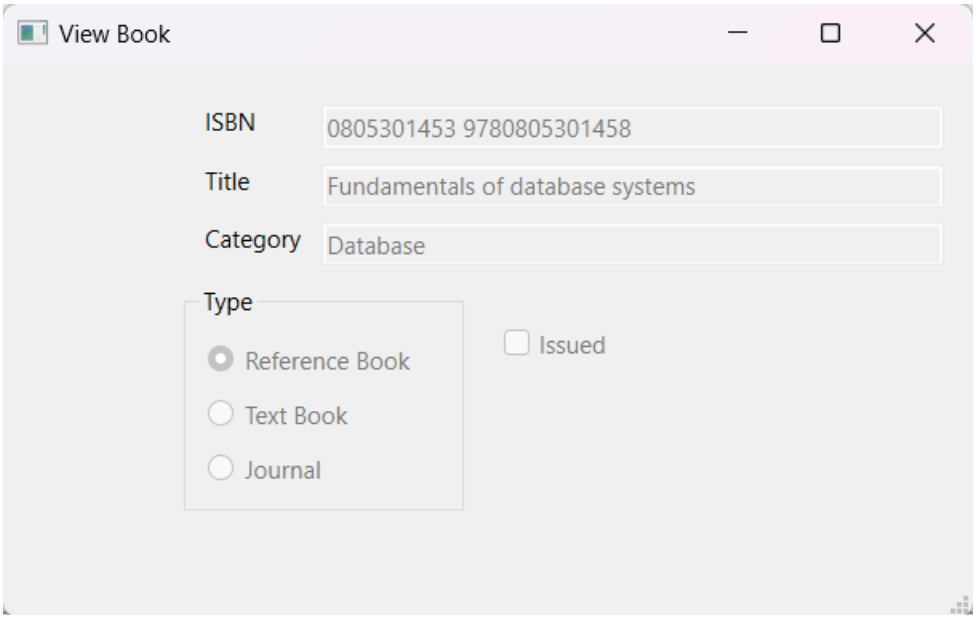

Figure 5: View Book Form in PyQt6

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**. 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 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 files are named exactly ("Lab02.ui") and ("view.ui") and the objectName of the table widget is (booksTableWidget)

A Python Code File for the Sample Application

```python
import sys
from PyQt6 import QtWidgets, uic

# 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(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi("main_form.ui", self)

        # Populate combo box
        self.id_combo.addItems([i[1] for i in data])
        self.name_input.setText(data[0][0])

        # Connect signals
        self.submit_button.clicked.connect(self.open_view_form)
        self.id_combo.activated.connect(self.handle_id_toggle)

    def open_view_form(self):
        index = self.id_combo.currentIndex()
        student_id, name, major, gpa = data[index][1], data[index][0], data[
    index][2], data[index][3]

        self.view_form = ViewForm(student_id, name, major, gpa)
        self.view_form.show()

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

class ViewForm(QtWidgets.QMainWindow):
    def __init__(self, student_id, name, major, gpa):
        super().__init__()
        uic.loadUi("view_form.ui", self)

        # Fill fields
        self.id_input.setText(student_id)
        self.id_input.setDisabled(True)
        self.name_input.setText(name)
        self.name_input.setDisabled(True)
        self.major_input.setText(major)
        self.major_input.setDisabled(True)
        self.gpa_input.setText(str(gpa))
        self.gpa_input.setDisabled(True)

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    window = MainForm()
    window.show()
    sys.exit(app.exec())
```

Listing 1: Skeleton Python file (`demo.py`)

B Skeleton File for Lab Task

```
from PyQt6 import QtWidgets, uic
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import QApplication, QMainWindow, QVBoxLayout, QWidget, QPushButton, QComboBox, QLabel, QTableWidgetItem, QTableWidget, QHBoxLayout
from PyQt6.QtWidgets import QTableWidget, QTableWidgetItem

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"],
]

category=["Database","OOP","AI"]
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)


        # Populate the comboBox
        ...

        # Set current selection of comboBox to None
        self.comboBox.setCurrentIndex(-1)

        # 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):
        """
        Function to search and filter the booksTableWidget
        based on user input from comboBox, lineEdit, radioButtons, and checkBox
```

```
    .
        """

        # Get user input
        selected_combo = self.comboBox.currentText()
        selected_title = ...
        sel_radio_4 = ...
        sel_radio_5 = ...
        sel_radio_6 = ...
        issued = ...

        # Go through each row in the table
        for row in range(self.booksTableWidget.rowCount()):
            # Get values from the current row
            combo_compare = self.booksTableWidget.item(row, 2)
            title_compare = ...
            radio_compare = ...
            issued_compare = ...

            # Check if the user input and row data matches
            combo_match = selected_combo in {"", combo_compare.text()}
            title_match = ...
            issued_match = ...

            radio_match = False
            # Implement radio_match conditions below
            ...

            # Show row if all conditions are satisfied, otherwise hide it
            match = combo_match and title_match and radio_match and
    issued_match
            self.booksTableWidget.setRowHidden(row, not match)

        pass

    def view(self):
        # Get the currently selected row
        curr_row = self.booksTableWidget.currentRow()

        if curr_row >= 0:
            # Extract values from the selected row
            isbn = self.booksTableWidget.item(curr_row,0).text()
            title = ...
            cat = ...
            rad = ...
            issue = ...

            # Create and show the detailed view window
            self.view = ...
            self.view.show()

            # Reset current selection
            self.booksTableWidget.setCurrentItem(None)
        else:
            # Show a warning if no row is selected
            ...

        pass

    def delete(self):
        # Get the currently selected row
        row_delete = ...

```

```
        if row_delete >= 0:
            # Ask for confirmation before deleting
            confirmation = QtWidgets.QMessageBox.warning(
                ...
            )

            # If user confirms, delete the row and set current item to None
            if confirmation == QtWidgets.QMessageBox.StandardButton.Yes:
                ...
                self.booksTableWidget.setCurrentItem(None)
        else:
            # Show warning if no row is selected
            ...
        pass

    def close(self):
        # Close the form
        ...
        pass


class ViewBook(QtWidgets.QMainWindow):
    def __init__(self, isbn, title, cat, rad, issue):
        super(ViewBook, self).__init__()
        # Load the UI file
        uic.loadUi('view.ui', self)

        # Disable inputs (make them read-only)
        ...

        # Set values in fields
        ...

        # Select the correct radio button
        if rad == ...:
            ...
        elif rad == ...:
            ...
        elif rad == ...:
            ...

        # Set checkbox if issued
        if issue == ...:
            ...


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    window = UI()
    window.show()
    sys.exit(app.exec())
```

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