Building Desktop Applications using PyQt6

CS/CE 355L/373L: Database Systems

Lab 08

Lab done by:
Syed Asghar Abbas Zaidi (sz07201)

Instructor:
Muhammad Umer Tariq
Saima Shaheen

August 26, 2024

1 Overview

My system's theme is "Dark Mode" by default. As such you will see the windows reflecting that change. Due to my Windows OS not being activated, I can't change that, hopefully it's not a issue.

I have attached the code that you can use to verify various conditions we were asked to fulfill. I can also be called out for a viva if verification of some sort is needed

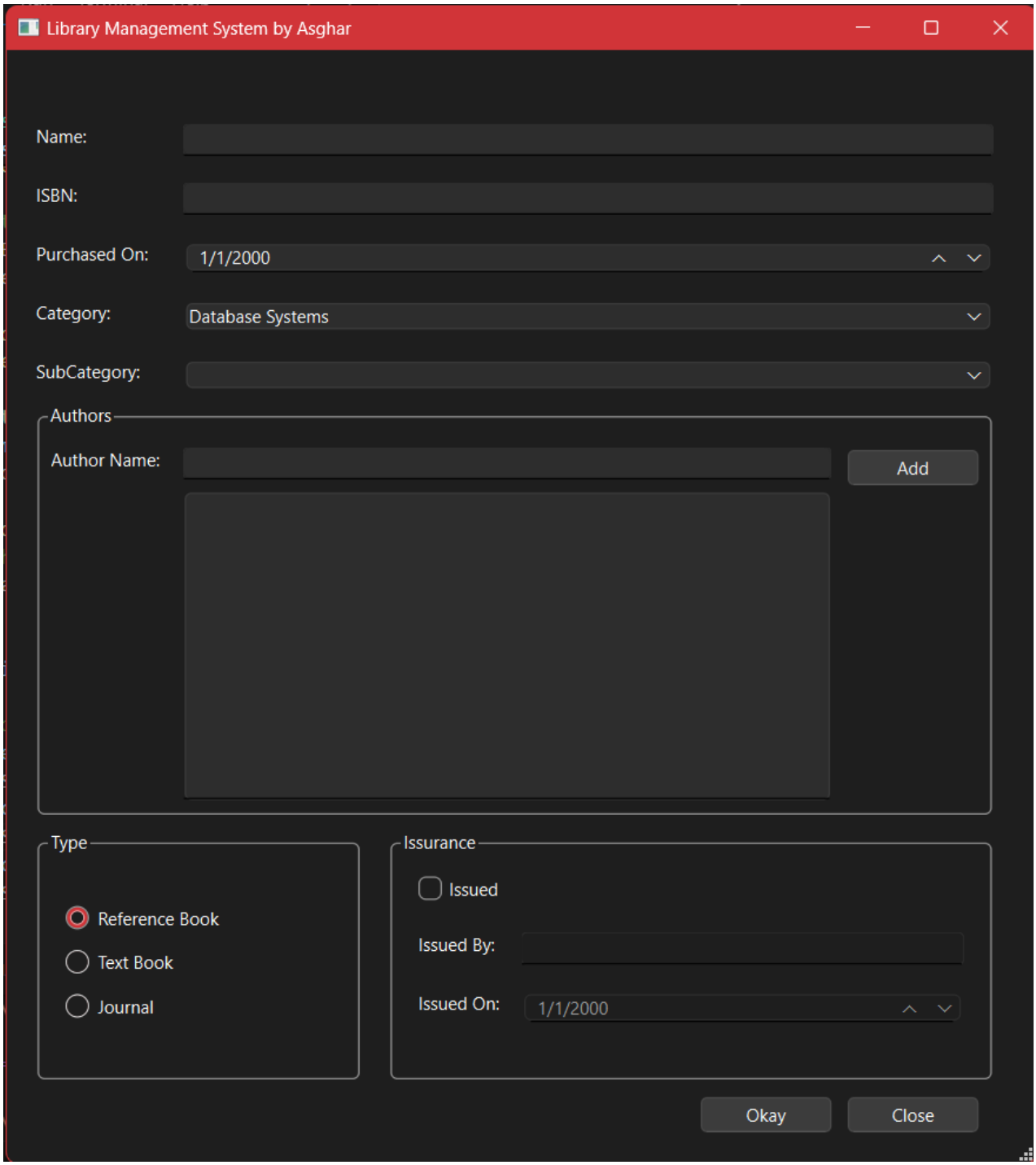
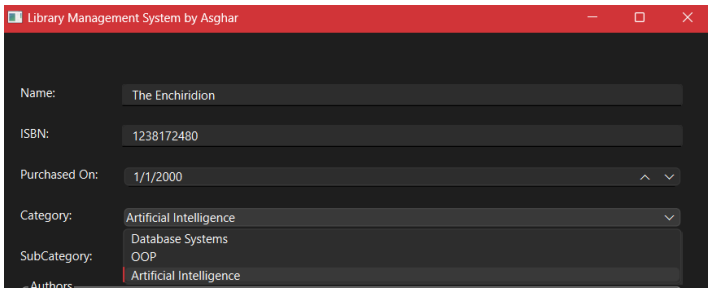


Figure 1: Library Management System App

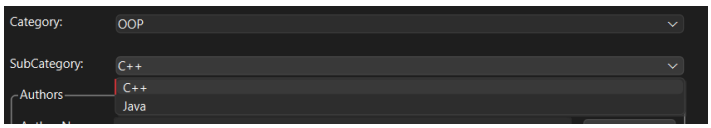
2 Proofs

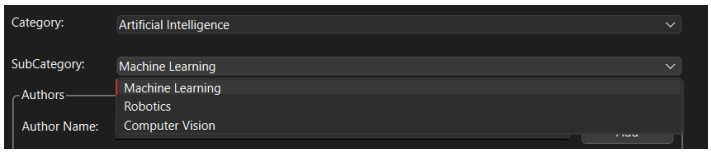
I will now be showcasing few screenshots that proofs that my code is in-fact properly working.

2.0.1 Category Combo

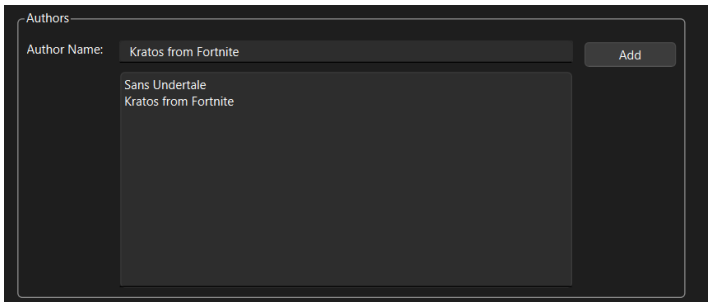


2.0.2 Subcategory





2.0.3 Adding Author's name



3 Error Messages

Attaching few error messages, I have attached the code in text form in Section 5. You can test-run my code using that as well

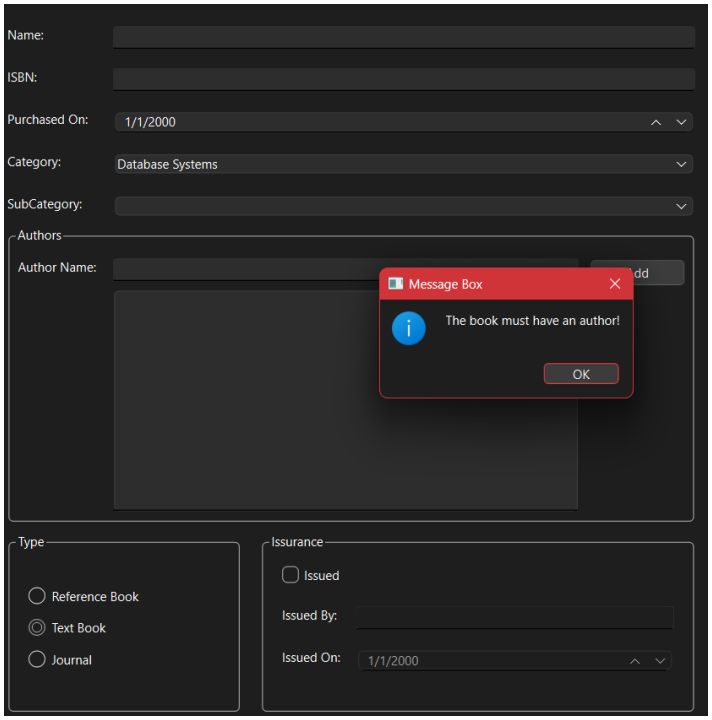


Figure 2: Error shown if author not picked if a Book is selected

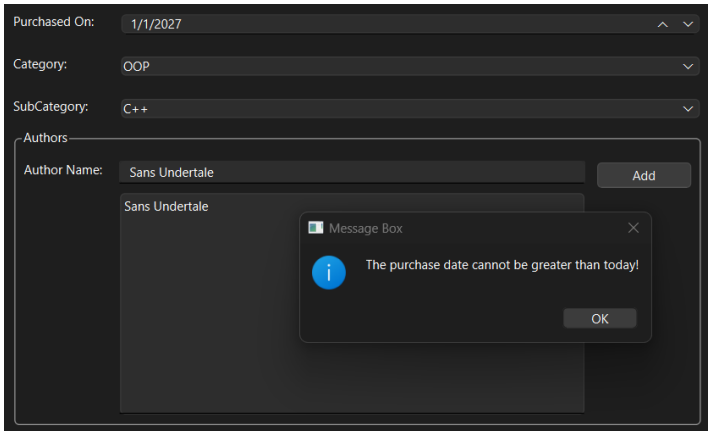


Figure 3: If you select purchase date far further in the future

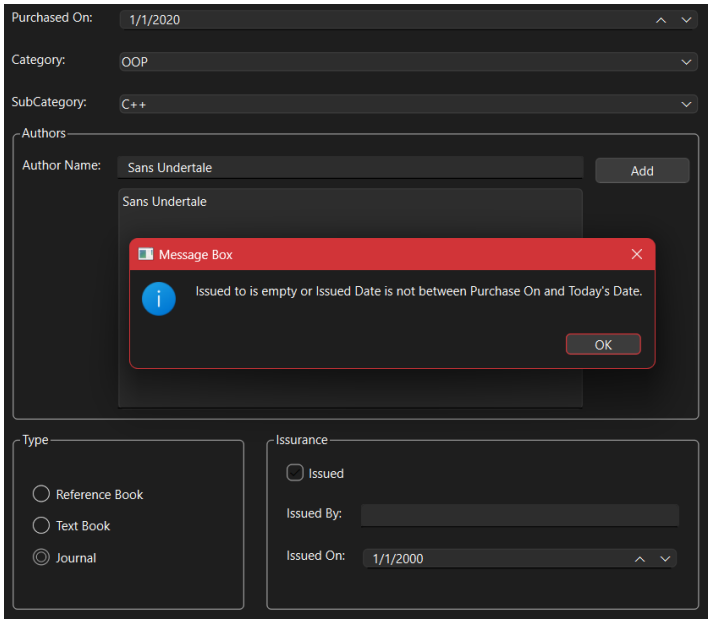


Figure 4: Issue Box not properly filled

4 Code Snippets

As we were asked to attach our code snippets as well. I will be doing that.

Example > app.py > UI > __init__

```python
# Import all the required libraries
from PyQt6 import QtWidgets, uic, QtGui, QtCore
from PyQt6.QtCore import QDate
from PyQt6.QtWidgets import QMessageBox  # Import QMessageBox correctly
import sys

class UI(QtWidgets.QMainWindow):
    def __init__(self):
        # Call the inherited classes __init__ method
        super(UI, self).__init__()

        # Load the .ui file
        uic.loadUi('LibraryManagementSystem.ui', self)

        # Show the GUI
        self.show()

        # Adding items to comboBox programmatically
        self.Category_ComboBox.addItem("Database Systems")
        self.Category_ComboBox.addItem("OOP")
        self.Category_ComboBox.addItem("Artificial Intelligence")

        # Event Handling
        self.Category_ComboBox.currentIndexChanged.connect(self.on_combobox_changed)
        self.AddAuthorButton.clicked.connect(self.handle_AuthorButtonclick)
        self.Issued_checkBox.toggled.connect(self.Issued_checkBox_Toggled)

        ##Initialize the state of the widgets based on the checkbox state
        self.Issued_checkBox_Toggled(self.Issued_checkBox.isChecked())

        ##OkayButton event
        self.OkayButton.clicked.connect(self.handle_OkayButtonclick)
        self.CloseButton.clicked.connect(self.close_window)

    def close_window(self):
        self.close()
```

```python
    def handle_AuthorButtonclick(self):
        Author_Name= self.AuthorName_LineEdit.text()
        self.AuthorName_textEdit.append(Author_Name)

    def on_combobox_changed(self):
        # Clear the contents of the second combo box
        self.SubCategory_ComboBox.clear()

        # Get the selected item from the first combo box
        selected_item = self.Category_ComboBox.currentText()

        # Conditional logic to change contents of the second combo box
        if selected_item == "Database Systems":
            self.SubCategory_ComboBox.addItems(["ERD", "SQL", "OLAP", "Data Mining"])
        elif selected_item == "OOP":
            self.SubCategory_ComboBox.addItems(["C++", "Java"])
        elif selected_item == "Artificial Intelligence":
            self.SubCategory_ComboBox.addItems(["Machine Learning", "Robotics", "Computer Vision"])

    def Issued_checkBox_Toggled(self, checked):
        self.IssuedBy_LineEdit.setEnabled(checked)
        self.IssuedOn_dateEdit.setEnabled(checked)

    def handle_OkayButtonclick(self):
        ISBN_LineEdit_text = self.ISBN_LineEdit.text() # Retrieve the text from the QLineEdit
        num_characters = len(ISBN_LineEdit_text) # Calculate the number of characters

        purchase_date = self.Purchase_DateEdit.date() # Retrieve the date from the QDateEdit
        today_date = QDate.currentDate() #Get today's date

        if purchase_date > today_date:
            msg_box = QMessageBox()
            msg_box.setWindowTitle("Message Box")
            msg_box.setText("The purchase date cannot be greater than today!")
            msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
```

```python
            msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
            msg_box.exec()  # Display the message box

        #Show error if number of characters are greater than 12 in ISBM
        elif num_characters > 12:
            msg_box = QMessageBox()
            msg_box.setWindowTitle("Message Box")
            msg_box.setText("The Length of ISBN can't be greater than 12!")
            msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
            msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
            msg_box.exec()  # Display the message box


        elif not self.Journal_RadioButton.isChecked():
            author_text = self.AuthorName_textEdit.toPlainText().strip()
            if not author_text:
                msg_box = QMessageBox()
                msg_box.setWindowTitle("Message Box")
                msg_box.setText("The book must have an author!")
                msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
                msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
                msg_box.exec()  # Display the message box

        elif self.Issued_checkBox.isChecked():
            issued_by_text = self.IssuedBy_LineEdit.text().strip()
            issued_on_date = self.IssuedOn_dateEdit.date()

            # Get today's date
            today_date = QDate.currentDate()

            # Validate the "Issued By" field
            if not issued_by_text or issued_on_date > today_date:
                msg_box = QMessageBox()
                msg_box.setWindowTitle("Message Box")
                msg_box.setText("Issued to is empty or Issued Date is not between Purchase On and Today's Date.")
```

```python
                msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
                msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
                msg_box.exec()  # Display the message box

        elif self.Journal_RadioButton.isChecked():
            msg_box = QMessageBox()
            msg_box.setWindowTitle("Message Box")
            msg_box.setText("Book Added Successfully")
            msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
            msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
            msg_box.exec()  # Display the message box

        else:
            msg_box = QMessageBox()
            msg_box.setWindowTitle("Message Box")
            msg_box.setText("Book Added Successfully")
            msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
            msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
            msg_box.exec()  # Display the message box


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)  # Create an instance of QtWidgets.QApplication
    window = UI()  # Create an instance of our class
    app.exec()  # Start the application
```

5 Appendix

Although I have attached the code screenshot snippets, I am also attaching the text

```python
# Import all the required libraries
from PyQt6 import QtWidgets, uic, QtGui, QtCore
from PyQt6.QtCore import QDate
from PyQt6.QtWidgets import QMessageBox  # Import QMessageBox correctly
import sys

class UI(QtWidgets.QMainWindow):
    def __init__(self):
        # Call the inherited classes __init__ method
        super(UI, self).__init__()

        # Load the .ui file
        uic.loadUi('LibraryManagementSystem.ui', self)

        # Show the GUI
        self.show()

        # Adding items to comboBox programmatically
        self.Category_ComboBox.addItem("Database Systems")
        self.Category_ComboBox.addItem("OOP")
        self.Category_ComboBox.addItem("Artificial Intelligence")

        # Event Handling
        self.Category_ComboBox.currentIndexChanged.connect(self.
           on_combobox_changed)
        self.AddAuthorButton.clicked.connect(self.handle_AuthorButtonclick)
        self.Issued_checkBox.toggled.connect(self.Issued_checkBox_Toggled)

        ##Initialize the state of the widgets based on the checkbox state
        self.Issued_checkBox_Toggled(self.Issued_checkBox.isChecked())

        ##OkayButton event
        self.OkayButton.clicked.connect(self.handle_OkayButtonclick)
        self.CloseButton.clicked.connect(self.close_window)

    def close_window(self):
        self.close()

    def handle_AuthorButtonclick(self):
        Author_Name= self.AuthorName_LineEdit.text()
        self.AuthorName_textEdit.append(Author_Name)

    def on_combobox_changed(self):
        # Clear the contents of the second combo box
        self.SubCategory_ComboBox.clear()

```

```
        # Get the selected item from the first combo box
        selected_item = self.Category_ComboBox.currentText()

        # Conditional logic to change contents of the second combo box
        if selected_item == "Database Systems":
            self.SubCategory_ComboBox.addItems(["ERD", "SQL", "OLAP", "Data Mining"])
        elif selected_item == "OOP":
            self.SubCategory_ComboBox.addItems(["C++", "Java"])
        elif selected_item == "Artificial Intelligence":
            self.SubCategory_ComboBox.addItems(["Machine Learning", "Robotics", "Computer Vision"])

    def Issued_checkBox_Toggled(self, checked):
        self.IssuedBy_LineEdit.setEnabled(checked)
        self.IssuedOn_dateEdit.setEnabled(checked)

    def handle_OkayButtonclick(self):
        ISBN_LineEdit_text = self.ISBN_LineEdit.text() # Retrieve the text from the QLineEdit
        num_characters = len(ISBN_LineEdit_text) # Calculate the number of characters

        purchase_date = self.Purchase_DateEdit.date() # Retrieve the date from the QDateEdit
        today_date = QDate.currentDate() #Get today's date

        if purchase_date > today_date:
            msg_box = QMessageBox()
            msg_box.setWindowTitle("Message Box")
            msg_box.setText("The purchase date cannot be greater than today!")
            msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
            msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
            msg_box.exec()  # Display the message box

        #Show error if number of characters are greater than 12 in ISBM
        elif num_characters > 12:
            msg_box = QMessageBox()
            msg_box.setWindowTitle("Message Box")
            msg_box.setText("The Length of ISBN can't be greater than 12!")
            msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
            msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
            msg_box.exec()  # Display the message box


```

```
        elif not self.Journal_RadioButton.isChecked():
            author_text = self.AuthorName_textEdit.toPlainText().strip()
            if not author_text:
                msg_box = QMessageBox()
                msg_box.setWindowTitle("Message Box")
                msg_box.setText("The book must have an author!")
                msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
                msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
                msg_box.exec()  # Display the message box

        elif self.Issued_checkBox.isChecked():
            issued_by_text = self.IssuedBy_LineEdit.text().strip()
            issued_on_date = self.IssuedOn_dateEdit.date()

            # Get today's date
            today_date = QDate.currentDate()

            # Validate the "Issued By" field
            if not issued_by_text or issued_on_date > today_date:
                msg_box = QMessageBox()
                msg_box.setWindowTitle("Message Box")
                msg_box.setText("Issued to is empty or Issued Date is not between Purchase On and Today's Date.")
                msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
                msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
                msg_box.exec()  # Display the message box

        elif self.Journal_RadioButton.isChecked():
            msg_box = QMessageBox()
            msg_box.setWindowTitle("Message Box")
            msg_box.setText("Book Added Successfully")
            msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
            msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
            msg_box.exec()  # Display the message box

        else:
            msg_box = QMessageBox()
            msg_box.setWindowTitle("Message Box")
            msg_box.setText("Book Added Successfully")
            msg_box.setIcon(QMessageBox.Icon.Information)  # Use an information icon
            msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  # Add OK button
            msg_box.exec()  # Display the message box
```

```python


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)  # Create an instance of QtWidgets.QApplication
    window = UI()  # Create an instance of our class
    app.exec()  # Start the application
```