

***LIBRARY MANAGEMENT SYSTEM USING PYTHON AND MySQL***

**REVIEW REPORT**

*Submitted by*

**Shreyaans Nahata** [*19BCE2686*]

**M B Srinidhi** [*18BCB0104*]

**Suyasha Agrawal** [*19BCE0321*]

For

**DATABASE MANAGEMENT SYSTEMS (*CSE2004*)**

**PROJECT COMPONENT**

Submitted to

**Prof. Nancy Victor**

**School of Computer Science and Engineering**

**ABSTRACT**

With the need of informationization and modernization of colleges, more and more colleges choose to move towards the direction of digital library management. As a transmission center of information in colleges, library plays a vital role in the dissemination of knowledge and spiritual civilization. The level of library construction is closely related to the quality of teaching in colleges

The Library Management System is an application for assisting a librarian in managing a book library. The system would provide basic set of features to add/update members, add/update books, and manage check in specifications for the systems based on the client’s statement of need.

A Library management system is a typical management Information system (MIS), its Development include the establishment and maintenance of back-end database and front-end application development aspects. For the former require the establishment of data consistency and integrity of the strong data security and good libraries. As for the latter requires the application fully functional, easy to use and so on.

One such effective system is our Library Management System which will be designed using the latest open-source technology. Our focus is to provide a lightweight application which is easy to use even for the least experienced user and provides basic functionality such as add books, remove books, edit book details, edit user details, add users, add/remove authors, publishers, categories and maintain records for various activities such as borrowed books and due dates and such.

We also intend to draw focus on presenting the database information in an easy and intelligible manner. The application will also feature password authentication so that the users can access their records and issue books from the library but will not be able to change any database details.

**TABLE OF CONTENTS**

|  |
| --- |
| **CHAPTER PAGE NO** |
| **Abstract 02** |
| 1. **Introduction 04-05**  1.1 Overview 04  1.2 Motivation 04  1.3 Objective 04  1.4 Organization of the Project 05 |
| 2. **Project Resource Requirements 06**  2.1.1 Software Requirements (on Developer Machine) 06  2.1.2 Software Requirements (on Client Machine) 06  2.2 Hardware Requirements 06 |
| 3. **Literature Survey 07-08** |
| 4. **Design of the Project 09-10**  4.1 ER Diagram 09  4.2 ER to Relational Mapping (Schema Diagram) 09  4.3 Tables and Constraints 10 |
| 5. **Normalized Tables 11-13** |
| 6. **Output 14-38** |
| 7. **Conclusion 39** |
| 8. **Code 40-56** |
| **References 57** |

**INTRODUCTION**

**1.1 Overview**

With the rapid development of computer technology, the application of computer technology in all walks of life has been widely popular. The development of modern information technology has led to the progress of the library in the direction of automation, network, and digitization. Due to the increase in the collection of library books and the increasing demand for information, the traditional manual management methods have many shortcomings, the main performance is that the efficiency of handling of borrowing books and returning books process is very low, obviously it cannot adapt to the current information society.

**1.2 Motivation**

We chose to implement a Library Management System since even though there are a lot of proprietary library management applications out there, most of which have a very unintuitive User Interface (UI) and may also lack many functions. We wish to make an application with an *intuitive UI*, along with functionality such as simplicity in use, lightweight and minimal use of system resources, ease of installation and other features such as *Dark Mode* and *Light Mode* support.

* 1. **Objective**

The Library Management System is an application for assisting a librarian in managing a book library. The system would provide basic set of features to add/update members, add/update books, and manage check in specifications for the systems based on the client’s statement of need.

The proposed system is an automated Library Management System. Through our application, user can add books, search books, update/edit information, add/edit information about authors, publishers, and various categories.

Our proposed system has the following advantages:

* User Friendly Interface.
* Fast access to Database.
* More Storage Capacity.
* Low and Efficient resource use.
* Ability to choose between multiple themes.
* Ability to export the data as excel (.xlsx) files.

All the manual difficulties in managing the Library will be rectified by implementing computerization.

The application will be made entirely using MySQL, Python and PyQt5 (a Python framework). MySQL comprises of the back-end, which holds the database. Python will be used for both front-end and back-end while the GUI for the application will be made using PyQt5.

* 1. **Organization of the Project**

|  |  |  |
| --- | --- | --- |
| **Reg. No.** | **Name** | **Work Assigned** |
| 19BCE2686 | Shreyaans Nahata | Creating GUI using PyQt5 and linking to the Python modules |
| 18BCB0104 | M B Srinidhi | Creating the database schema and populating with sample data |
| 19BCE0321 | Suyasha Agrawal | Creating Python modules and linking with MySQL database |

**PROJECT RESOURCE REQUIREMENTS**

**2.1.1 Software Requirements (on Developer Machine):**

GUI Created Using: Qt Designer

Database Created Using: MySQL Workbench 8.0 Community Edition

Connected GUI to DB Using: Python v3.8.6

Required python packages:

PyQt5: To connect python code to UI

mysqlclient: To connect python code to MySQL DB

xlrd: To extract data from Excel spreadsheets

xlsxwriter: To write data onto Excel spreadsheets

datetime: To get the system date-time related information.

**2.1.2 Software Requirements (On Client Machine):**

***MySQL*** to manage the Database on the client machine.

An ***Office Desktop Editor*** (like MSOffice) to access the exported xlsx files.

***Python3*** (preferably over 3.7.x)

*Following Python* ***pip*** *packages*:

PyQt5, mysqlclient, xlsxwriter, xlrd, datetime

**2.2 Hardware Requirements**

A laptop/desktop with at least:

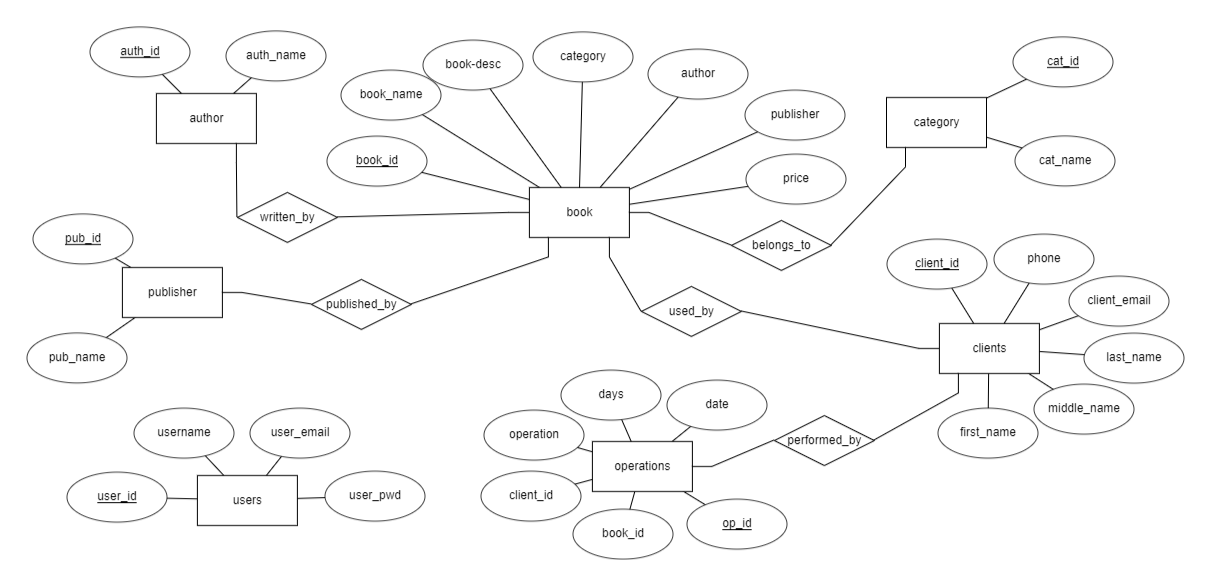
* A dual-core CPU.
* 4 GB RAM.
* Windows/MacOS/Linux with MySQL installed.
* 100MB free storage.

**LITERATURE SURVEY**

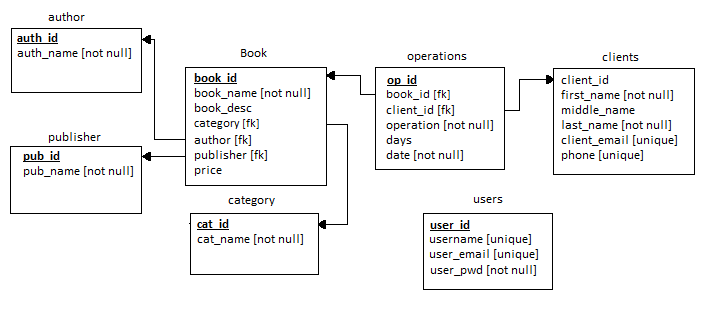
|  |  |  |
| --- | --- | --- |
| **Author** | **Contribution** | **Research Gap** |
| *Amin (2003)* | Provides information about various open source software for use in libraries like, software tools for automation, software tools for value added services, software tools for digital library initiatives, miscellaneous supporting tools. | Doesn’t mention the cost and resources required to develop and maintain the software. |
| *Eby (2007)* | Provides information on some of the available open source library management systems, digital library software, metasearch, link resolvers, federated search engines and OPAC software. | Doesn’t mention the limitations for development of the software. |
| *Hoffman & Yang (2012)* | Studies the current usage of next generation online public access catalogues and discovery tools in academic libraries in the USA and Canada. They also report that use of discovery tool is increasing. The author also provides update on next generation catalogue and discovery tool usage in academic libraries of both countries. |  |
| *Dartmouth College Library report (2013)* | Describes shortcomings of the present generation of library management systems and suggests improvements and inclusion of features in next generation systems like discovery, personalization, Reuse, collection development, collection management, electronic resource management system integration. |  |
| *Yang (2013)* | Describes advanced features of next generation library management systems such as interoperability, electronic resource management, role-based login, and other features such as support for different record formats, integration with other system. | Doesn’t mention the vulnerabilities and security risks associated with the software. |
| *Palmer & Choi (2014)* | Assesses the state of open source software research in the library context by employing descriptive literature review. They found that most of the significant areas of research are digital repository software, OPAC and integrated library systems. |  |
| *Lal Bahadur Chouhan (2010)* | Provides open source software for library management study like KOHA, GSDL and open journal system (OJS) to study all functions and requirements of library automation. |  |
| *Ayodeji Iwayemi (2019)* | Development of robust library manage system. They have made a website that allows students and teachers to access the library easily and at the same time automate library processes by keeping record of library resources and allocating them automatically. | Doesn’t mention the difficulties in accommodating any modifications in the system. |
| *A.Sanni (2013)* | The design of an Integrated library system with internet security solution. They have developed a library management system which not only provides easy access to all the books and information but in a secure and safe way. |  |
| *Yujun Li (2012)* | Development of a web-based library management system. They have made a three-layer architecture and applying model building language. Using JSP technique for front end and SQL server 2005 for backend. | Doesn’t mention the efficiency rate and performance of the developed system. |

**DESIGN OF THE PROJECT**

**4.1 Entity-Relationship Diagram**



**4.2 ER to Relational Mapping (Schema Diagram)**



Book (book\_id, book\_name, book\_desc, book\_cat, author, publisher, price)

Users (user\_id, username, user\_email, user\_pwd)

operations (op\_id, book\_id, client\_id, operation days, date)

clients (client\_id, first\_name, middle\_name, last\_name, client\_email, phone)

category (cat\_id, cat\_name)

author (auth\_id, auth\_name)

publisher (pub\_id, pub\_name)

**4.3 Tables and Constraints**

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **DATATYPE** | **CONSTRAINT** |
| **Book** | book\_id | int | primary key, auto\_increment |
| book\_name | varchar | not null |
| book\_desc | varchar |  |
| Category | int | foreign key category (cat\_id) |
| Author | int | foreign key author (auth\_id) |
| Publisher | int | foreign key pub(pub\_id) |
| Price | int |  |
|  | | | |
| **Users** | user\_id | int | primary key, auto\_increment |
| username | varchar | Unique |
| user\_email | varchar | unique |
| user\_pwd | varchar | not null |
|  | | | |
| **Clients** | Client\_id | int | Primary key, auto\_increment |
| First\_name | varchar | Not null |
| Middle\_name | Varchar |  |
| Last\_name | Varchar | Not null |
| Client\_email | Varchar | Unique |
| phone | varchar | unique |
|  | | | |
| **operations** | op\_id | int | primary key, auto\_increment |
| Book\_id | int | foreign key Book (book\_id) |
| Client\_id | int | foreign key clients(client\_id) |
| operation | varchar | foreign key Users (user\_id) |
| days | int |  |
| date | datetime | Not null |
|  | | | |
| **category** | cat\_id | varchar | primary key, auto\_increment |
| cat\_name | varchar | not null |
|  | | | |
| **author** | auth\_id | varchar | primary key, auto\_increment |
| auth\_name | varchar | not null |
|  | | | |
| **publisher** | pub\_id | varchar | primary key, auto\_increment |
| pub\_name | varchar | not null |

**NORMALIZED TABLES**

Since there are *no attributes* that aren’t on the RHS of any functional dependency, the *Candidate Keys* must comprise of all the *attributes on the LHS* of all functional dependencies.

Since all the candidate keys are super keys as well, we can conclude that ***all tables*** are in ***Boyce-Codd Normal Form (BCNF).***

**Table:** *book*

**Functional Dependencies:**

Book\_id 🡪 {book\_id, book\_name, book\_desc, category, author, publisher, price}

Book\_name 🡪 {book\_id, book\_name, book\_desc, category, author, publisher, price}

Book\_desc 🡪 {book\_id, book\_name, book\_desc, category, author, publisher, price}

**Candidate Keys:** {book\_id, book\_name, book\_desc}

|  |  |  |
| --- | --- | --- |
| *ATTRIBUTE* | DATATYPE | CONSTRAINT |
| Book\_id | Int | Primary key, auto\_increment |
| Book\_name | Varchar | Unique |
| Book\_desc | varchar | Unique |
| Category | Int | Fk category(cat\_id) |
| Author | Int | Fk author(auth\_id) |
| Publisher | Int | Fk publisher(pub\_id) |
| Price | int |  |

**Table:** *users*

**Functional Dependencies:**

User\_id 🡪 {user\_id, username, user\_email, user\_pwd}

Username 🡪 {user\_id, username, user\_email, user\_pwd}

User\_email 🡪 {user\_id, username, user\_email, user\_pwd}

**Candidate Keys:** {user\_id, username, user\_email}

|  |  |  |
| --- | --- | --- |
| *ATTRIBUTE* | DATATYPE | CONSTRAINT |
| User\_id | Int | Primary key, auto\_increment |
| username | Varchar | Unique |
| user\_email | varchar | Unique |
| User\_pwd | Int | Not null |

**Table:** *clients*

**Functional Dependencies:**

Client\_id 🡪 {client\_id, first\_name, middle\_name, last\_name, client\_email, phone}

Client\_email 🡪 {client\_id, first\_name, middle\_name, last\_name, client\_email, phone}

**Candidate Keys:** {client\_id, client\_email}

|  |  |  |
| --- | --- | --- |
| ATTRIBUTE | DATATYPE | CONSTRAINT |
| client\_id | Int | Primary key, auto\_increment |
| first\_name | Varchar | not null1 |
| Middle\_name | varchar |  |
| Last\_name | Varchar | Not null |
| Client\_email | Varchar | Unique |
| phone | varchar | Unique |

**Table:** *operations*

**Functional Dependencies:**

Op\_id 🡪 {op\_id, book\_id, client\_id, operation, days, date}

**Candidate Keys:** {op\_id}

|  |  |  |
| --- | --- | --- |
| ATTRIBUTE | DATATYPE | CONSTRAINT |
| op\_id | Int | Primary key, auto\_increment |
| Book\_id | Int | Fk book(book\_id) |
| Client\_id | Int | Fk clients(client\_id) |
| operation | Varchar | Not null |
| Days | Int |  |
| date | datetime | Not null |

**Table:** *author*

**Functional Dependencies:**

Auth\_id 🡪 {auth\_id, auth\_name}

Auth\_name 🡪 {auth\_id, auth\_name}

**Candidate Keys:** {auth\_id, auth\_name}

|  |  |  |
| --- | --- | --- |
| ATTRIBUTE | DATATYPE | CONSTRAINT |
| Auth\_id | Int | Primary key, auto\_increment |
| Auth\_name | Varchar | Not null |

**Table:** *publisher*

**Functional Dependencies:**

Pub\_id 🡪 {pub\_id, pub\_name}

Pub\_name 🡪 {pub\_id, pub\_name}

**Candidate Keys:** {pub\_id, pub\_name}

|  |  |  |
| --- | --- | --- |
| ATTRIBUTE | DATATYPE | CONSTRAINT |
| Pub\_id | Int | Primary key, auto\_increment |
| Pub\_name | Varchar | Not null |

**Table:** *category*

**Functional Dependencies:**

cat\_id 🡪 {cat\_id, cat\_name}

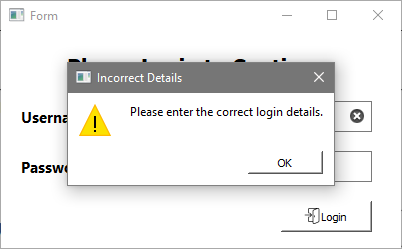
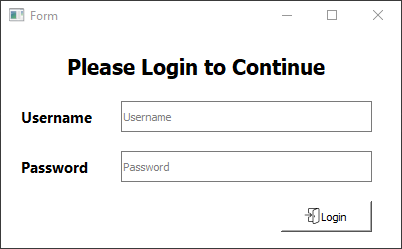
cat\_name 🡪 {cat\_id, cat\_name}

**Candidate Keys:** {cat\_id, cat\_name}

|  |  |  |
| --- | --- | --- |
| ATTRIBUTE | DATATYPE | CONSTRAINT |
| Cat\_id | Int | Primary key, auto\_increment |
| Cat\_name | Varchar | Not null |

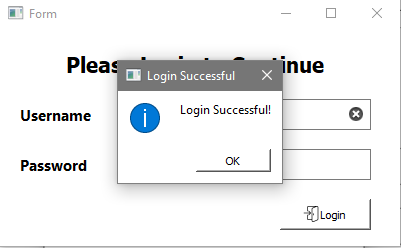
**OUTPUT**

**Main Application Pages:**

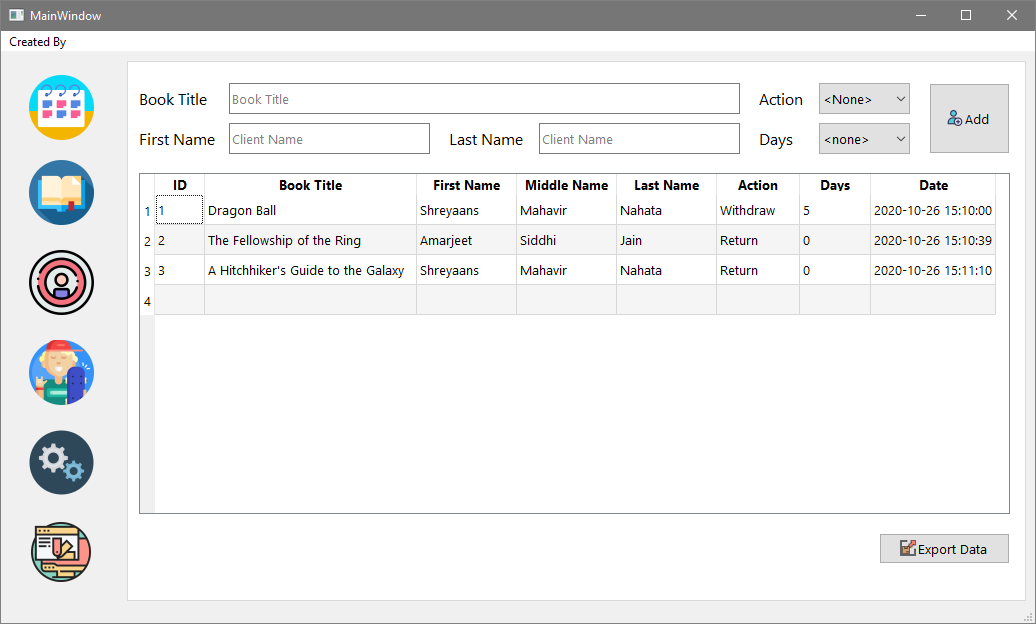


***Image:*** *Error upon entering wrong login details*

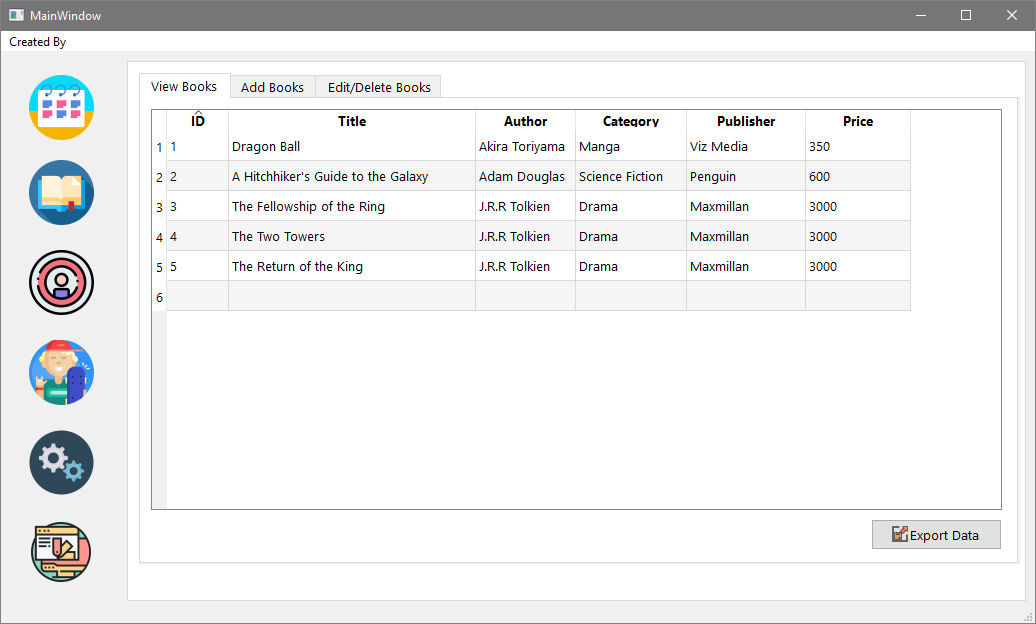
***Image:*** *Main Login Page*



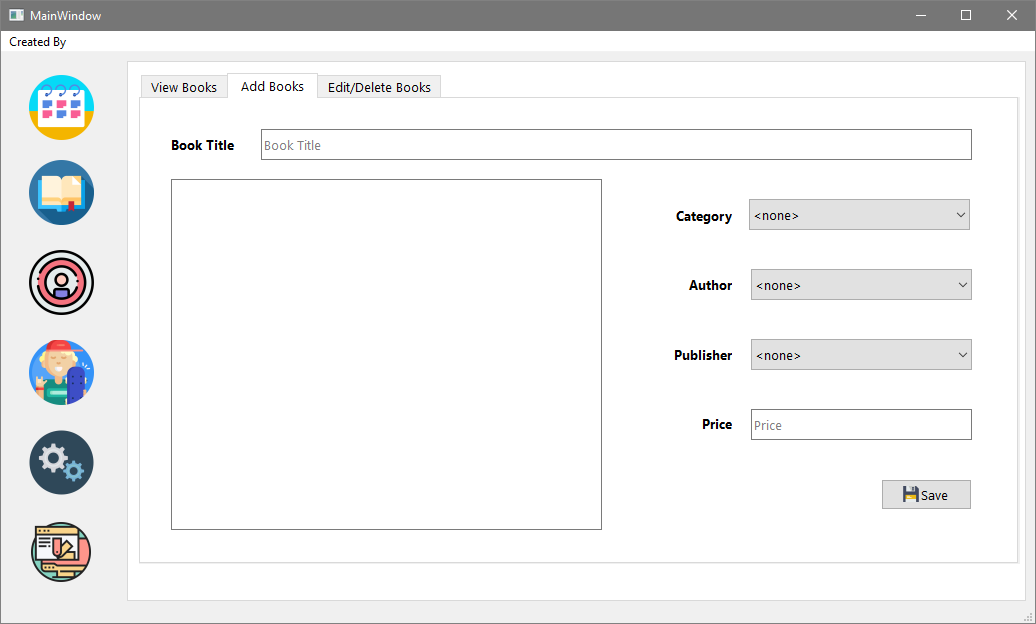
***Image:*** *Login successful confirmation*



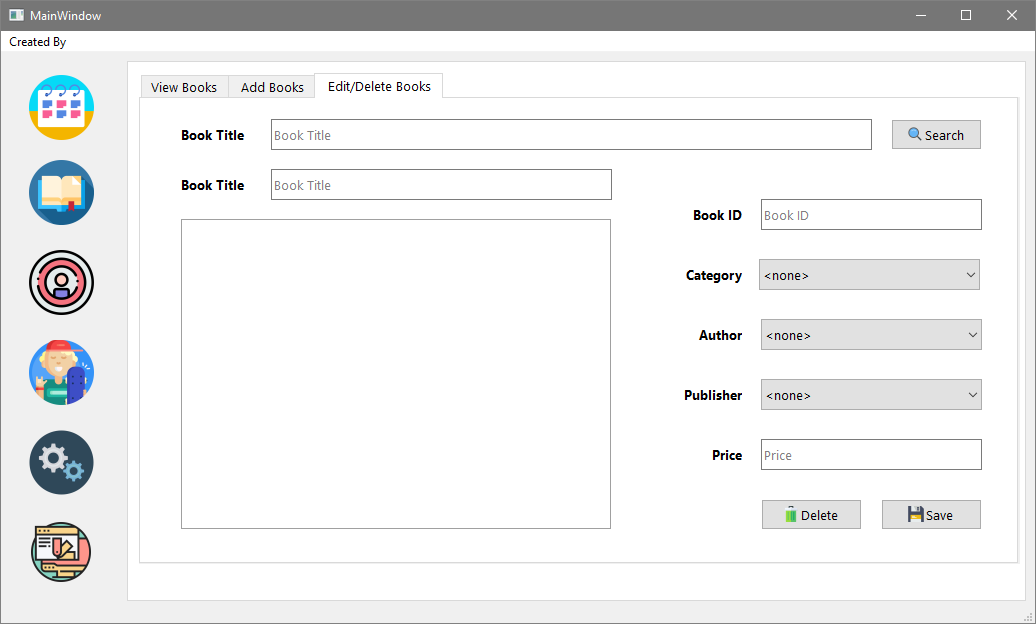
***Image:*** *Main Page of the Application. Shows all the library operation details.*



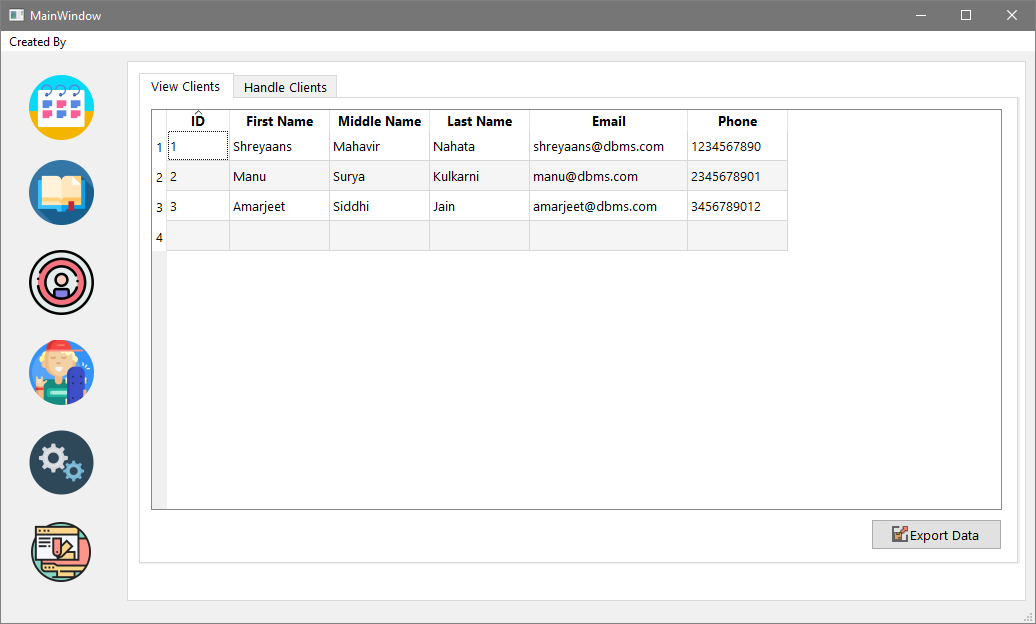
***Image:*** *View Books Tab. Shows all the books in the library.*



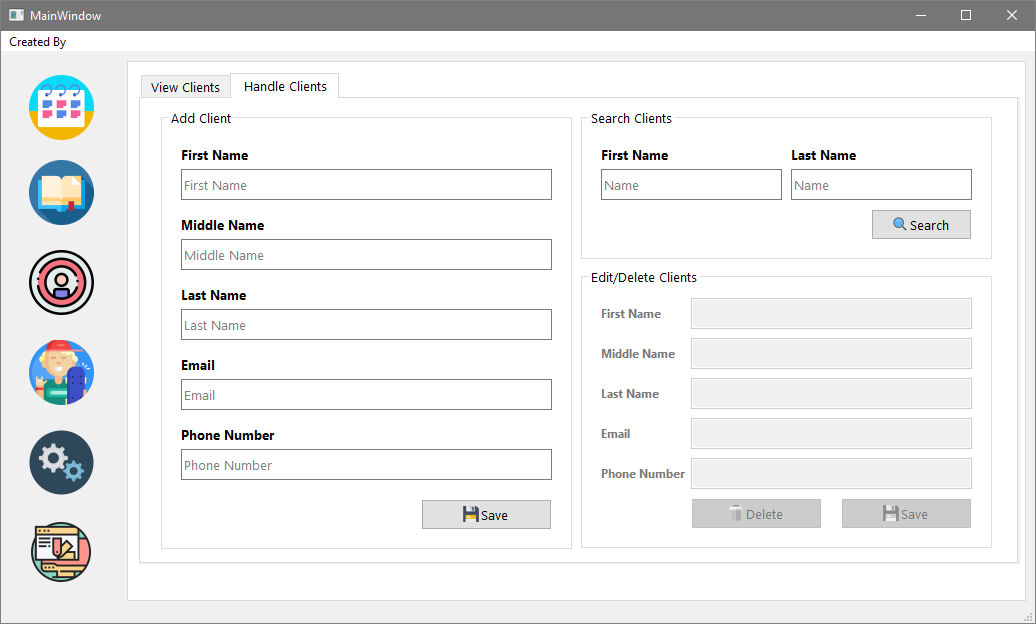
***Image:*** *Add Books Tab. Allows the users to add new books in the library.*



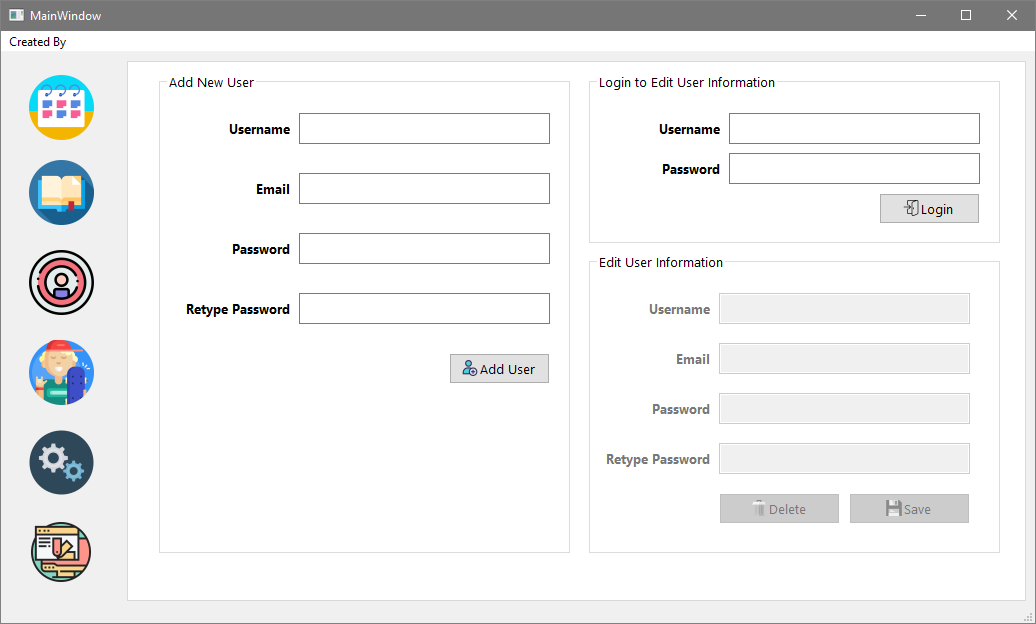
***Image:*** *Edit/Delete Books Tab. Allows the users to edit/delete existing books in the library.*



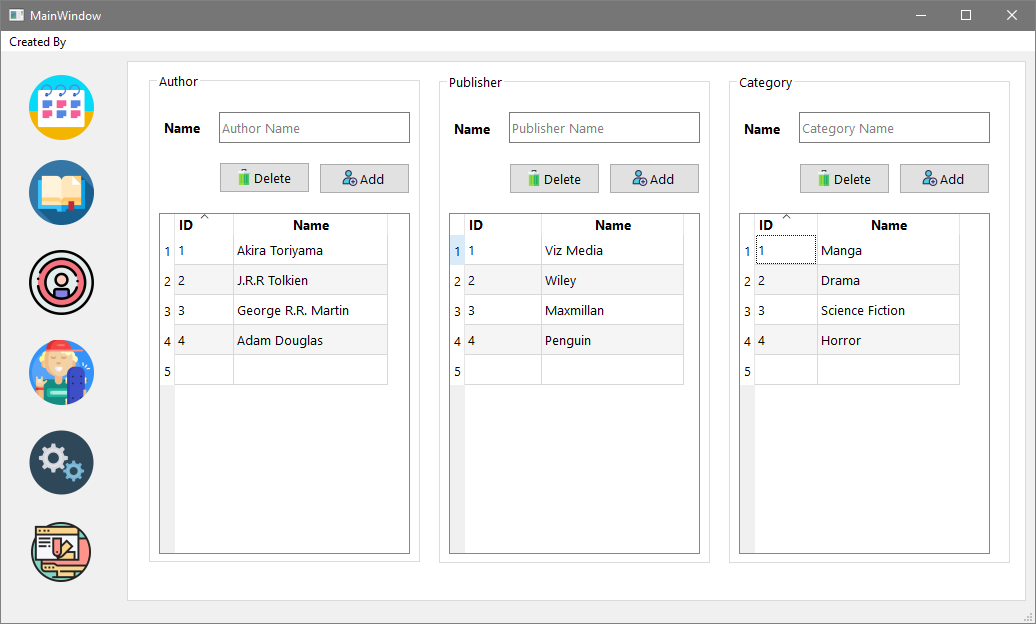
***Image:*** *View Clients tab. Shows all the library client details.*



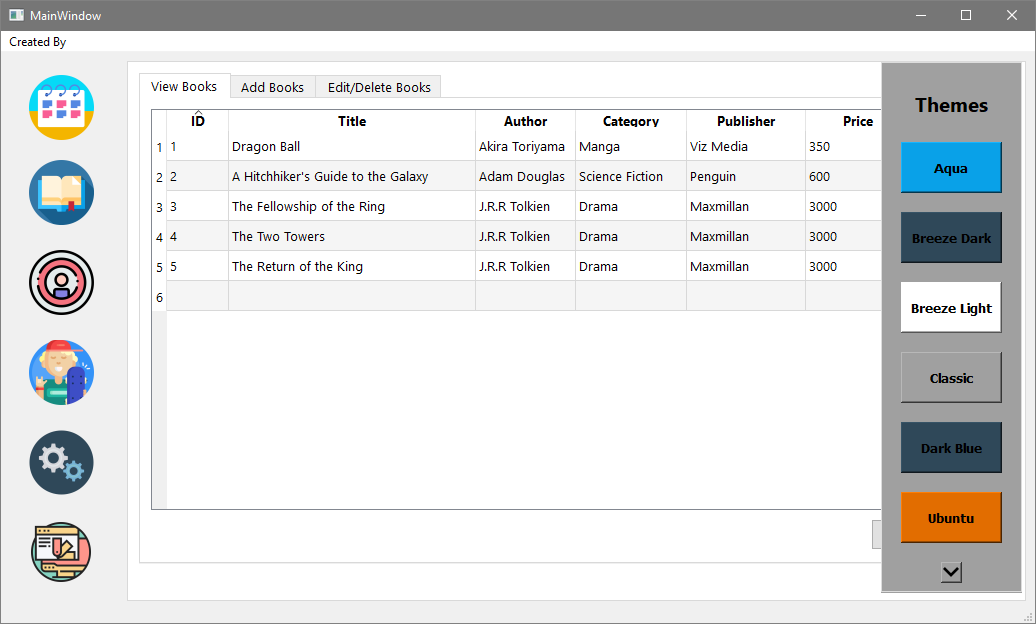
***Image:*** *Users Tab. Add/Search/Edit/Delete Clients.*



***Image:*** *Users Tab. Add/Search/Edit/Delete Users.*

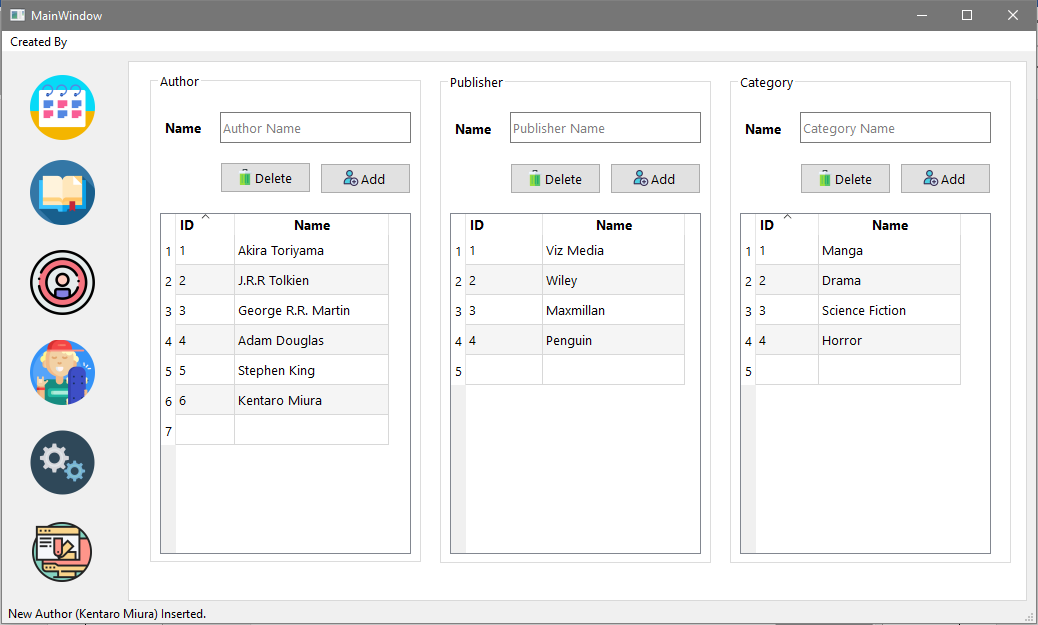


***Image:*** *Settings Tab. Allows the users to add/delete authors, publishers, categories.*



***Image:*** *Themes Tab. Allows the users to choose between various themes.*

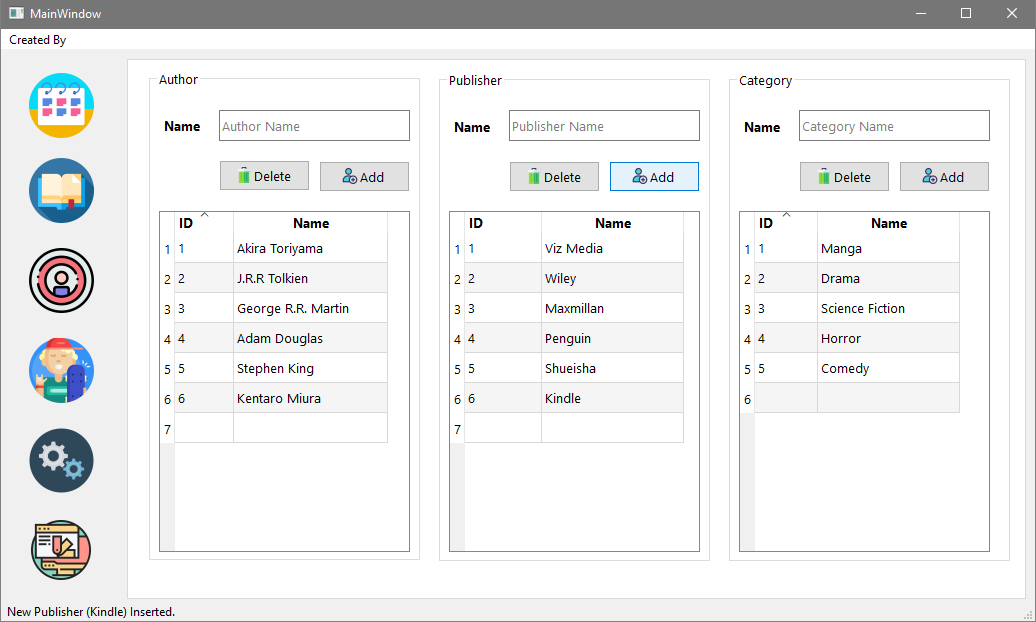
**Adding New Author:**



***Confirmation on success***

***Image:*** *Add New Author in Database.*

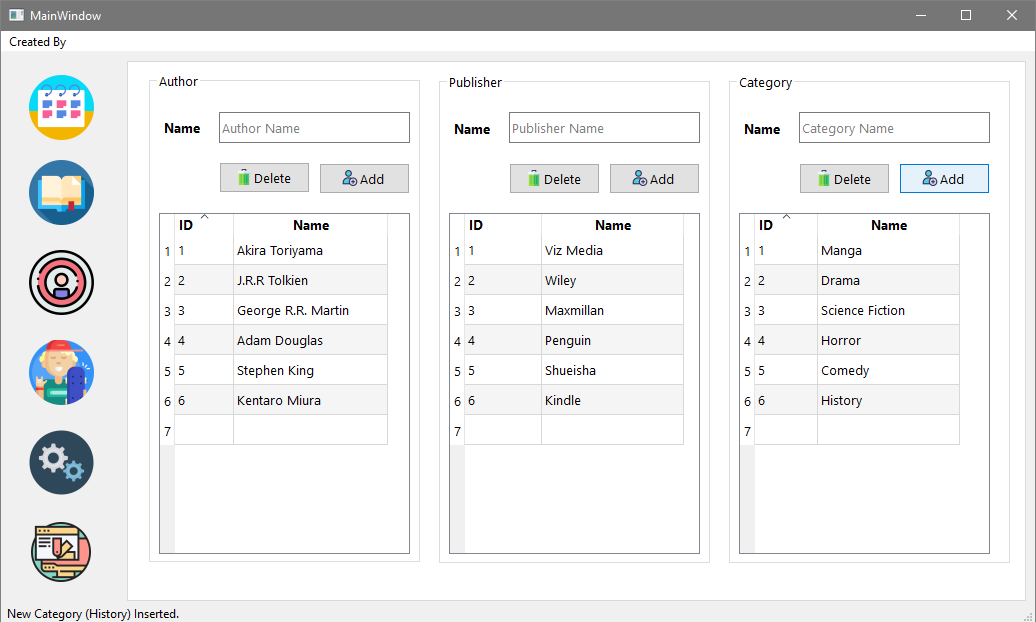
**Add New Publisher:**



***Confirmation on success***

***Image:*** *Add New Publisher in Database.*

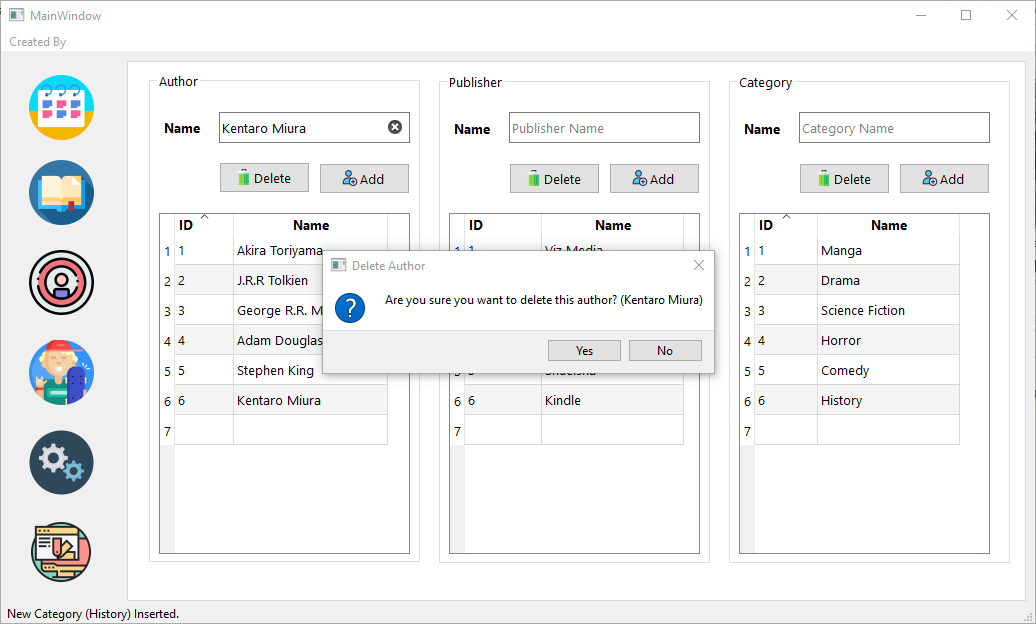
**Add New Category:**



***Confirmation on success***

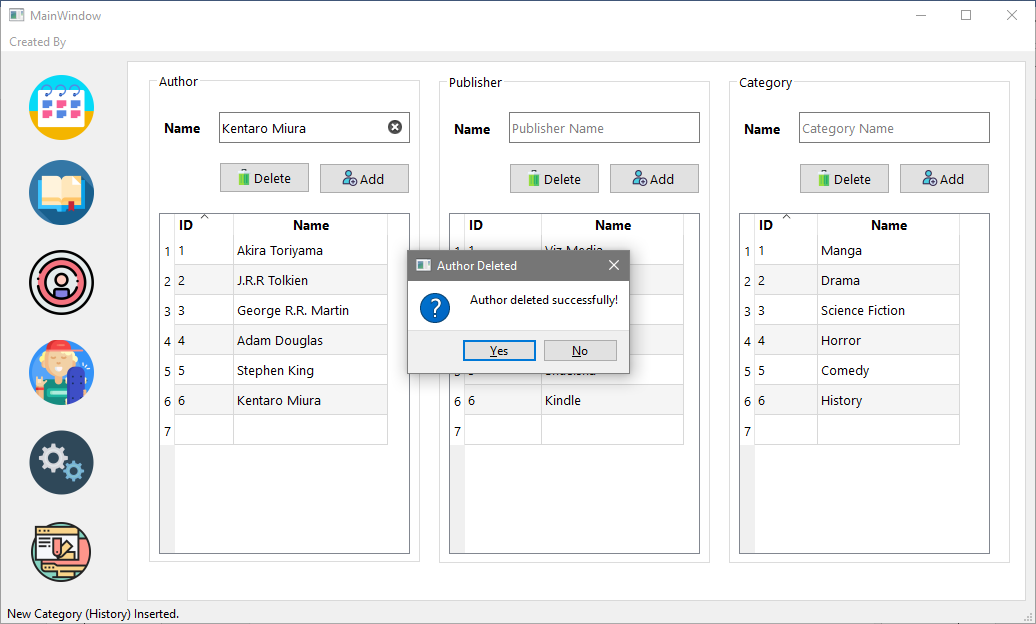
***Image:*** *Add New Category in Database.*

**Delete Existing Author:**



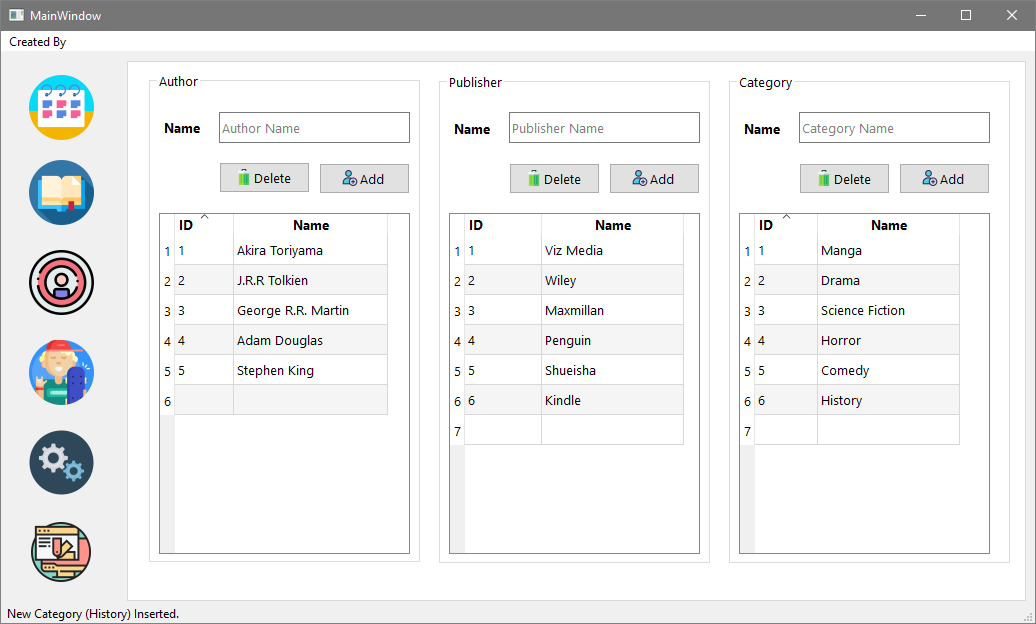
***Asks confirmation from user***

***Image:*** *Delete Author from database.*



***Successful deletion message***

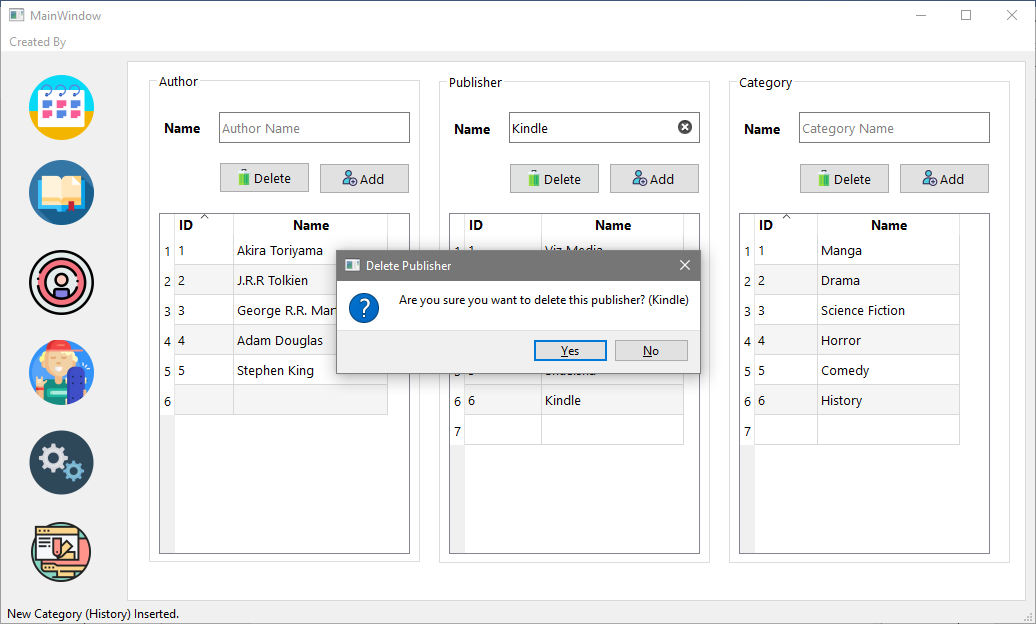
***Image:*** *Deletion is successful.*



***Deleted Author cannot be seen***

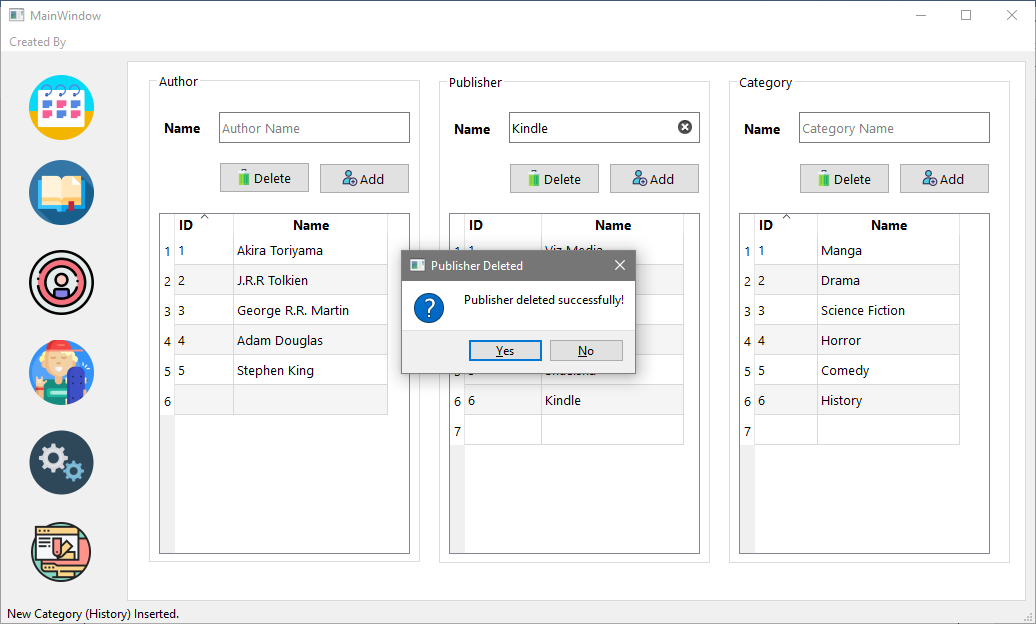
***Image:*** *Author is deleted from the database.*

**Delete Existing Publisher:**



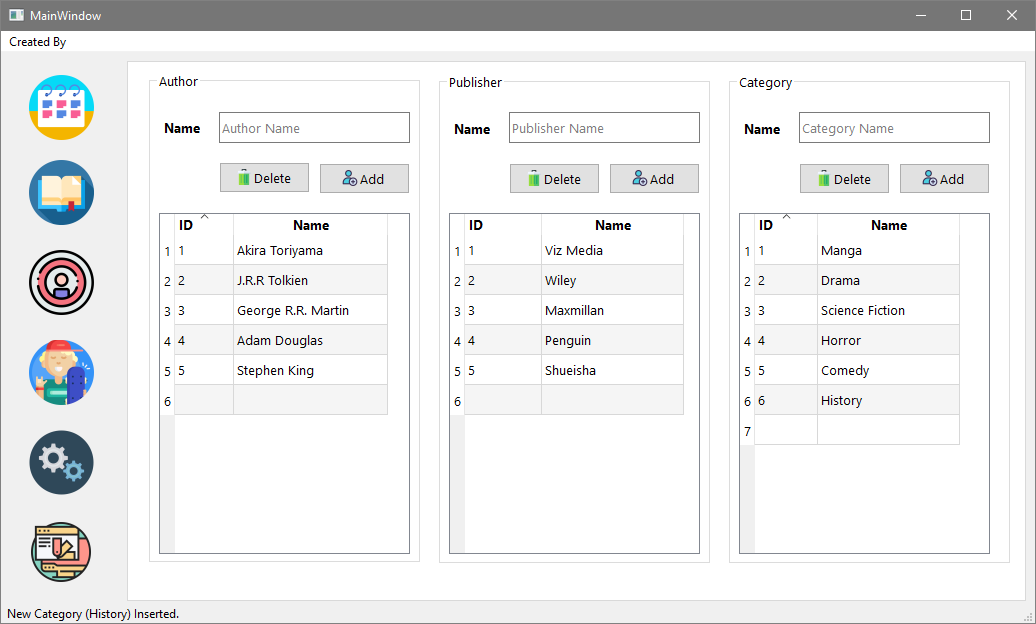
***Asks confirmation from user***

***Image:*** *Delete Publisher from database.*



***Successful deletion message***

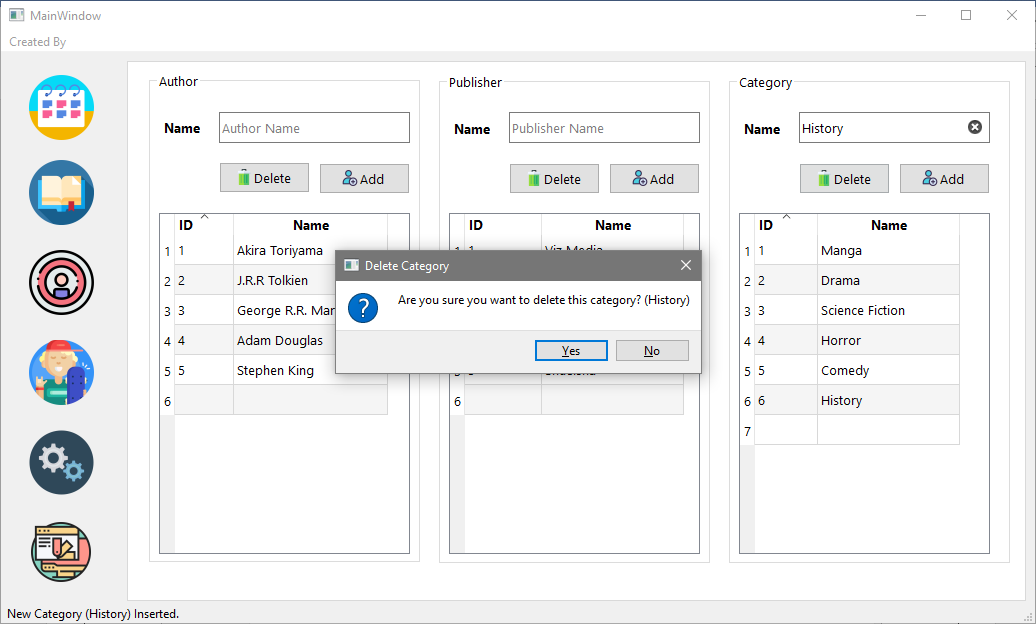
***Image:*** *Deletion is successful.*



***Deleted Publisher cannot be seen***

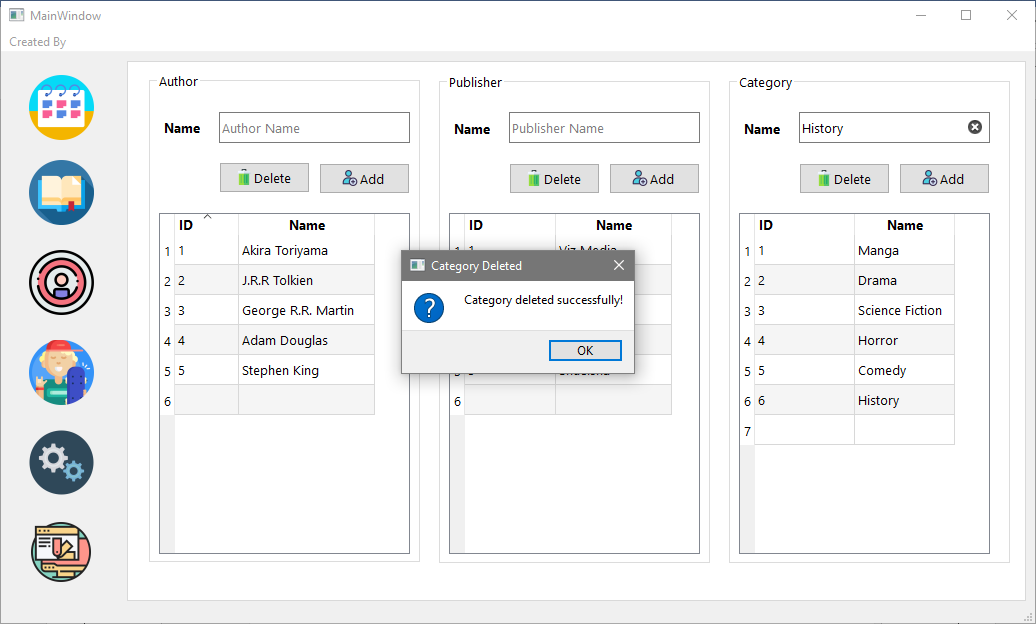
***Image:*** *Publisher is deleted from the database.*

**Delete Existing Category:**



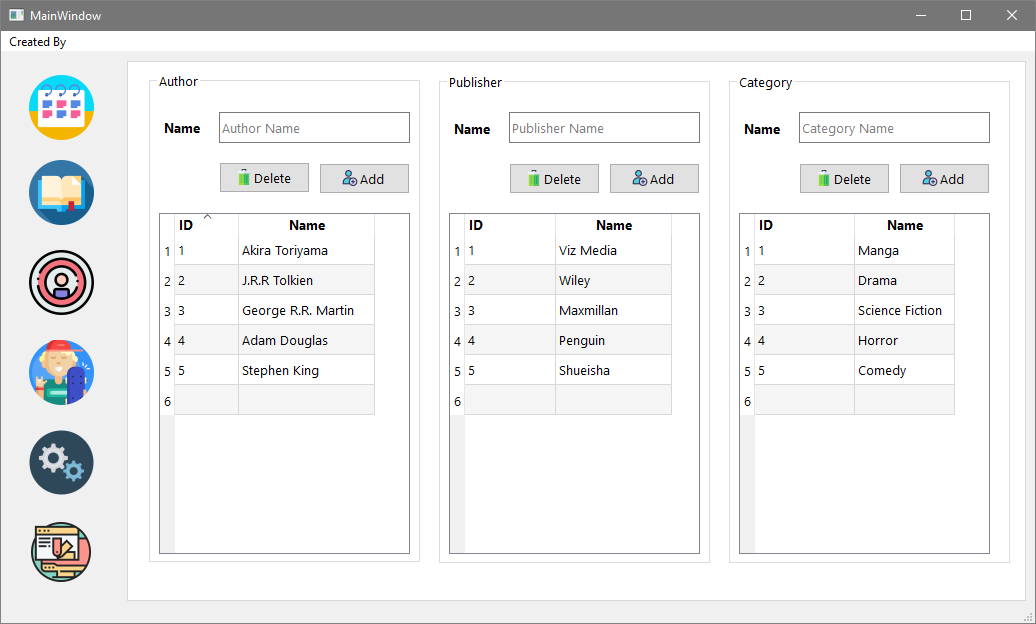
***Asks confirmation from user***

***Image:*** *Delete Category from database.*



***Successful deletion message***

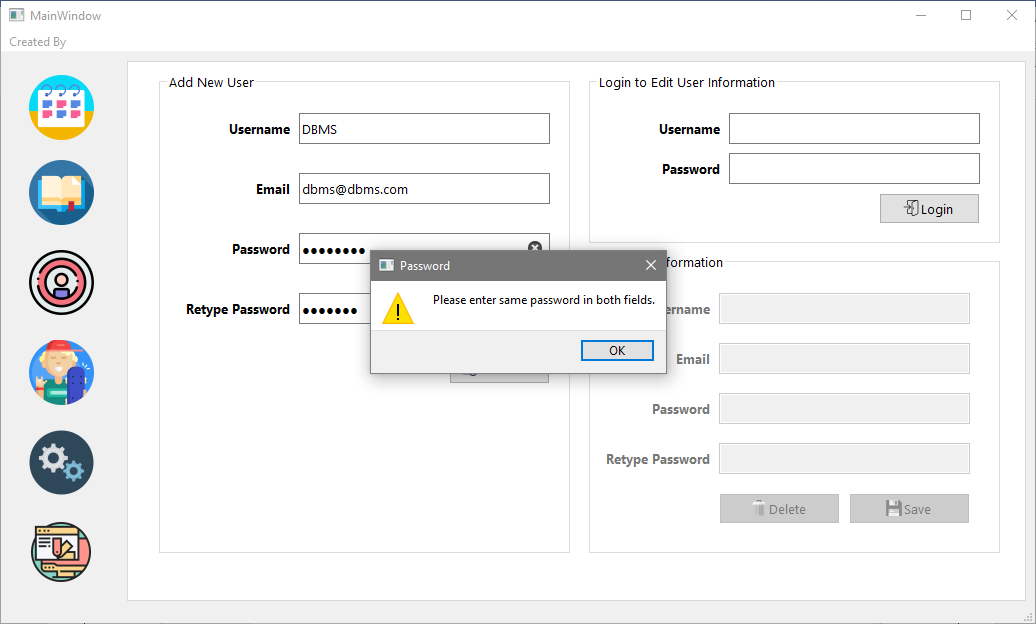
***Image:*** *Delete Publisher from database.*



***Image:*** *Category is deleted from the database.*

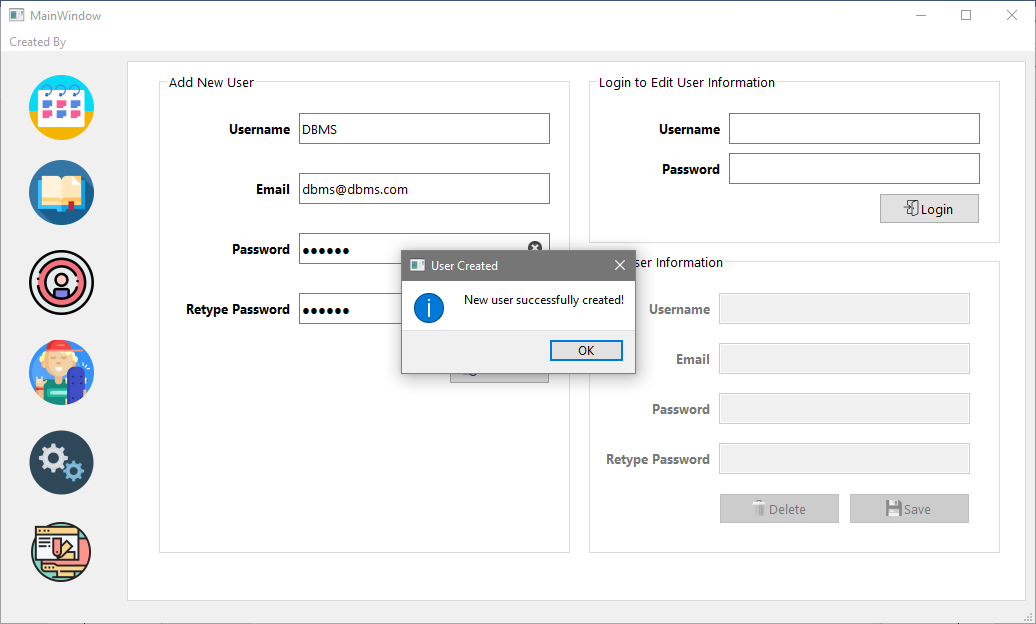
***Deleted Category cannot be seen***

**Creating New User:**



***Image:*** *Error Message if both passwords aren’t same*

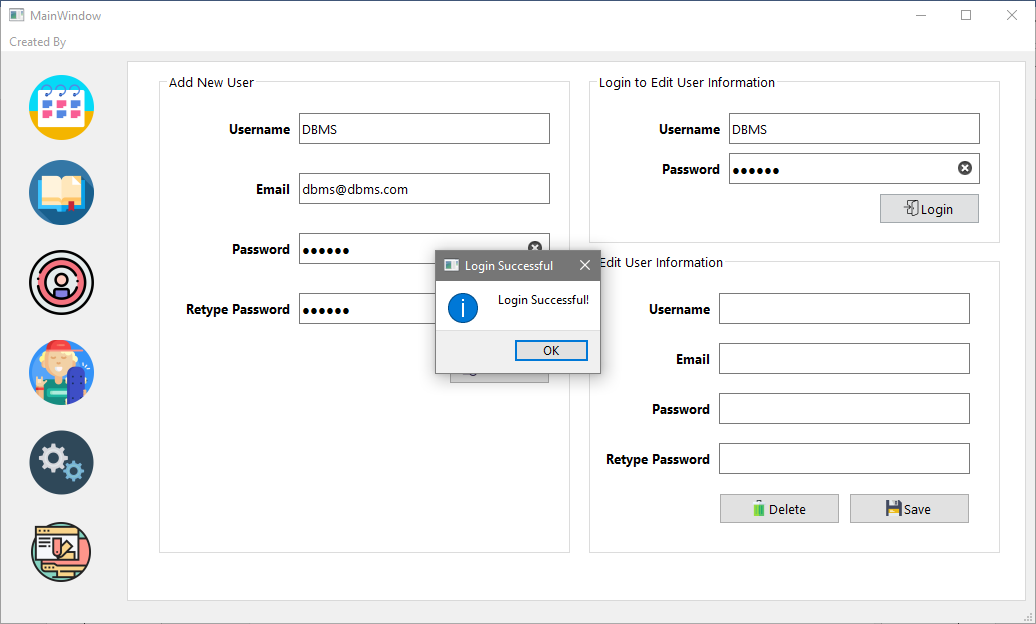
***Error Message***



***Group box is initially disabled***

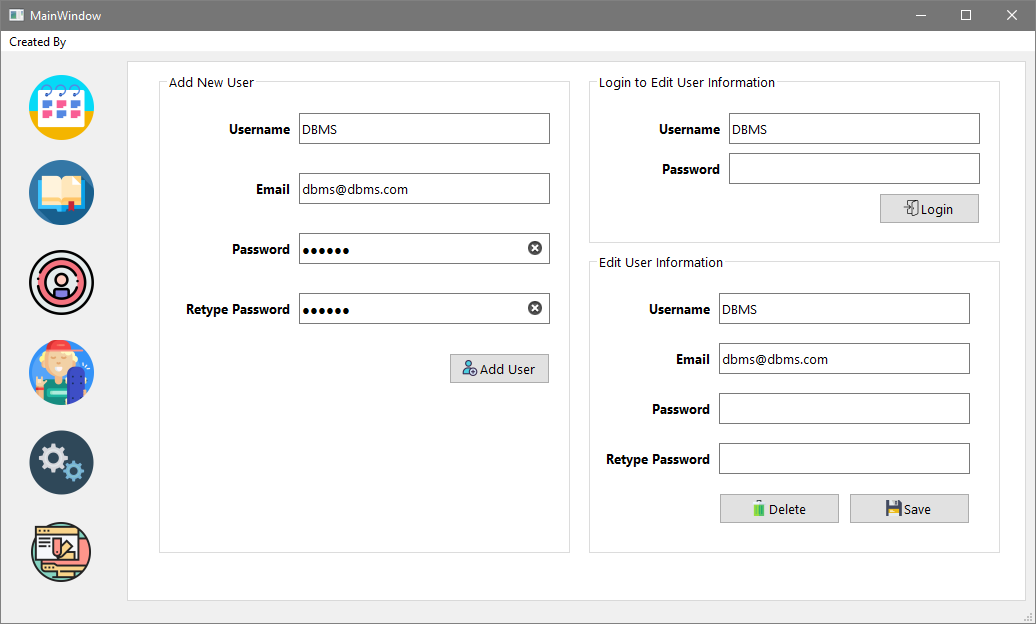
***Image:*** *New User has been created.*

**Login to Edit Information:**



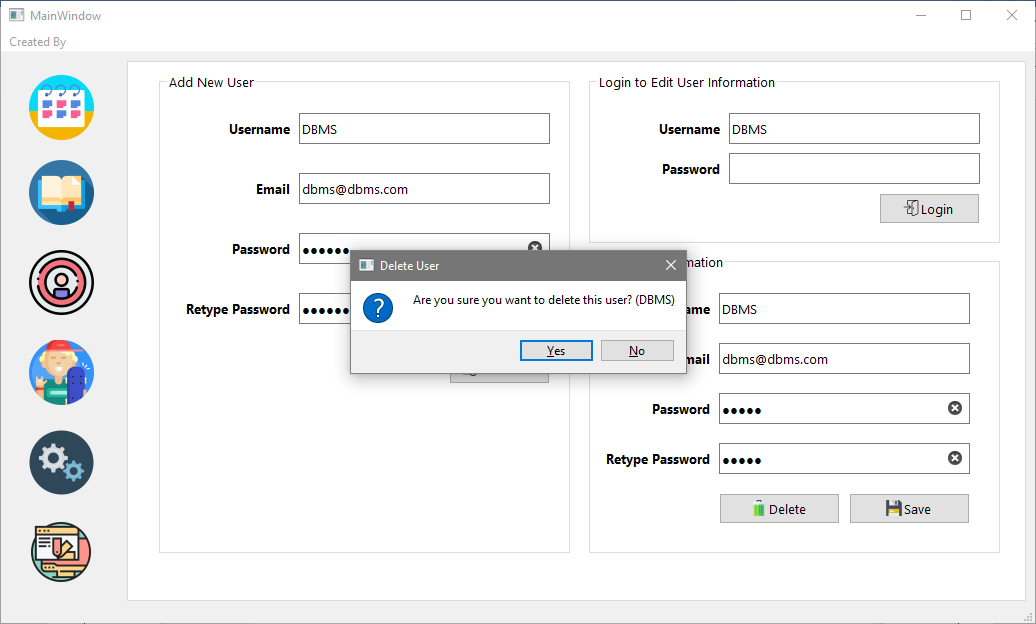
***Group box enables upon successful login***

***Image:*** *Login for users to be able to edit their data.*



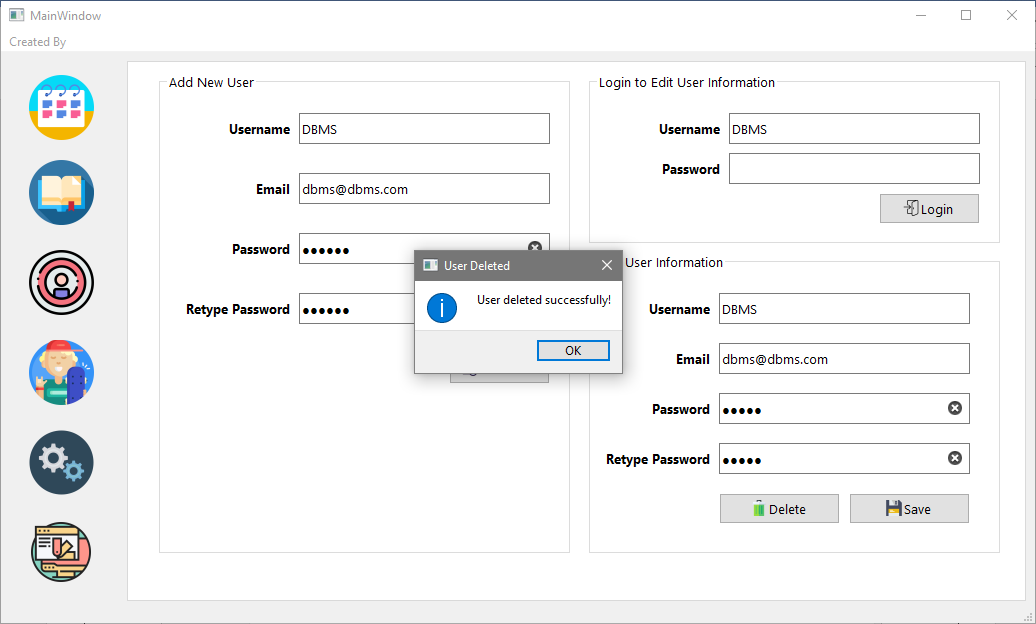
***Username cannot be edited***

***Image:*** *Edit/Delete User data from the database.*



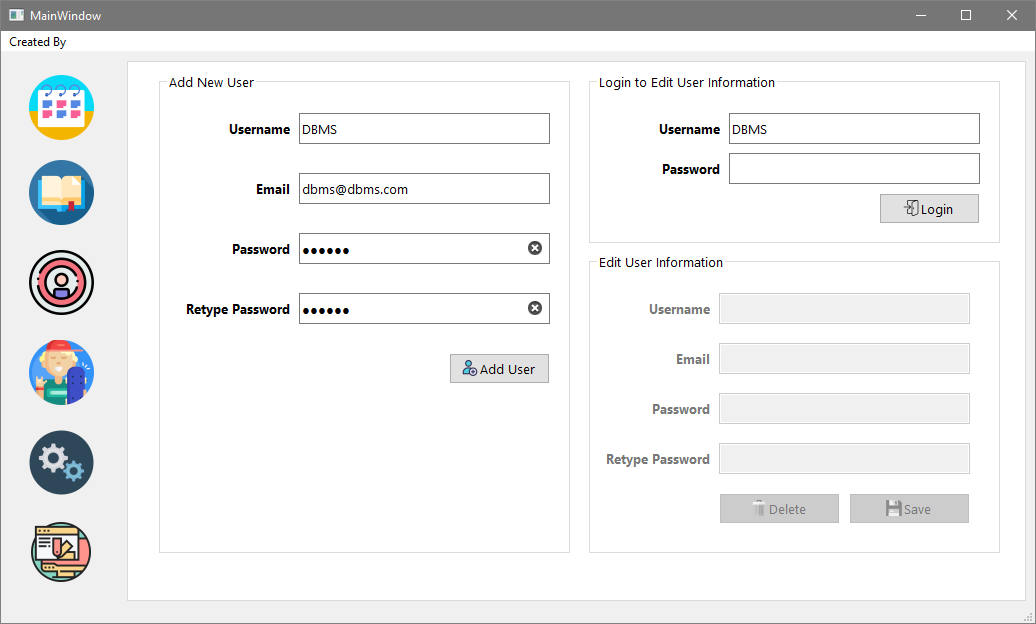
***Asks confirmation from user***

***Image:*** *Delete User from database.*



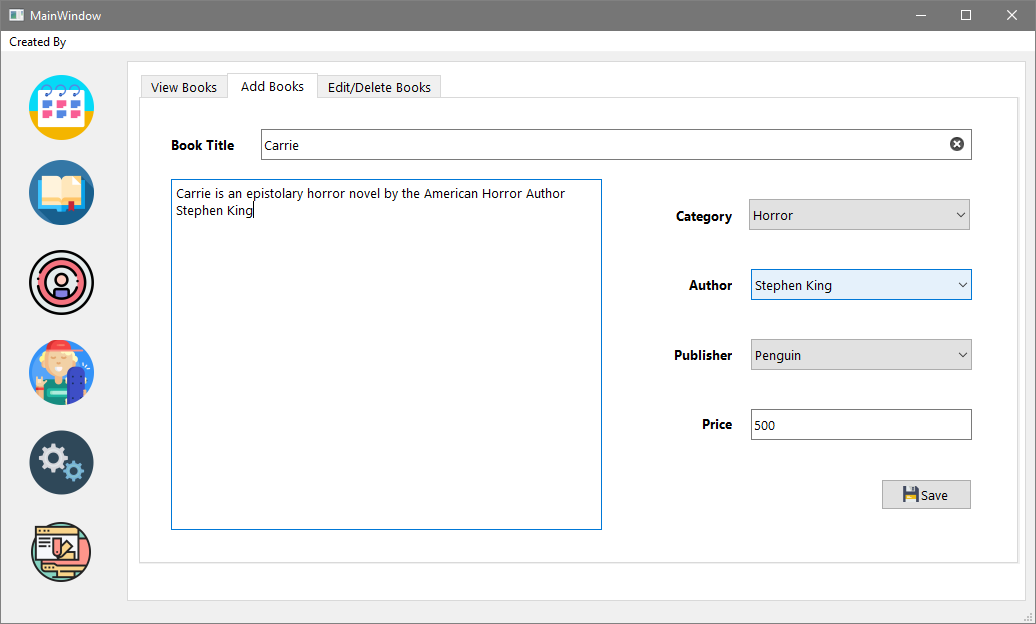
***Image:*** *User has been deleted from the database.*

***Successful deletion message***

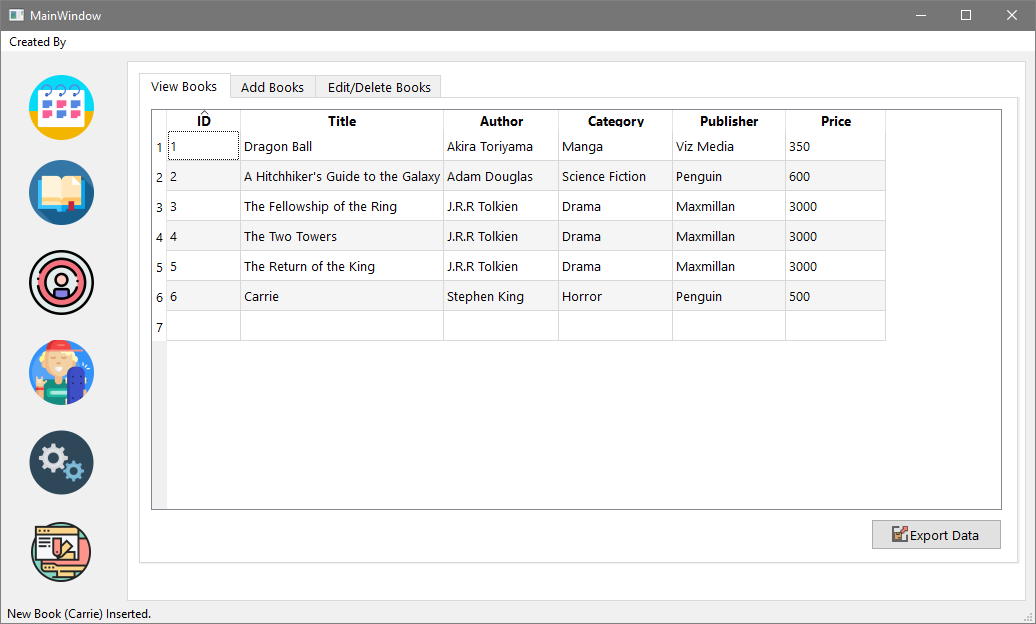


***Group Box disables upon successful Edit/Delete operation***

**Add New Book:**



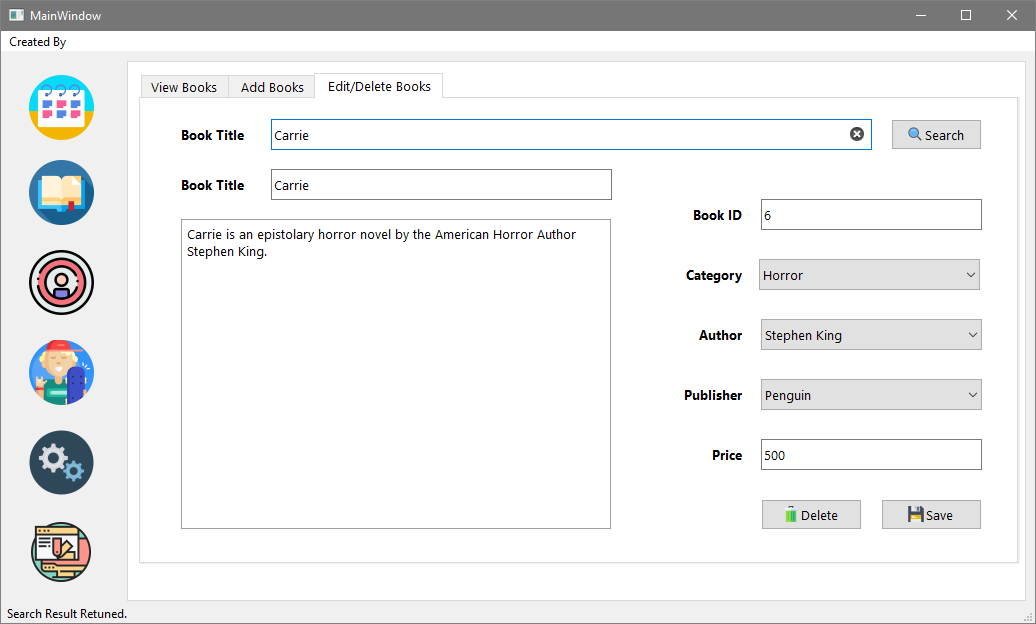
***Image:*** *Adding new book in the library.*



***Image:*** *Book successfully added.*

***Book can be seen in the list***

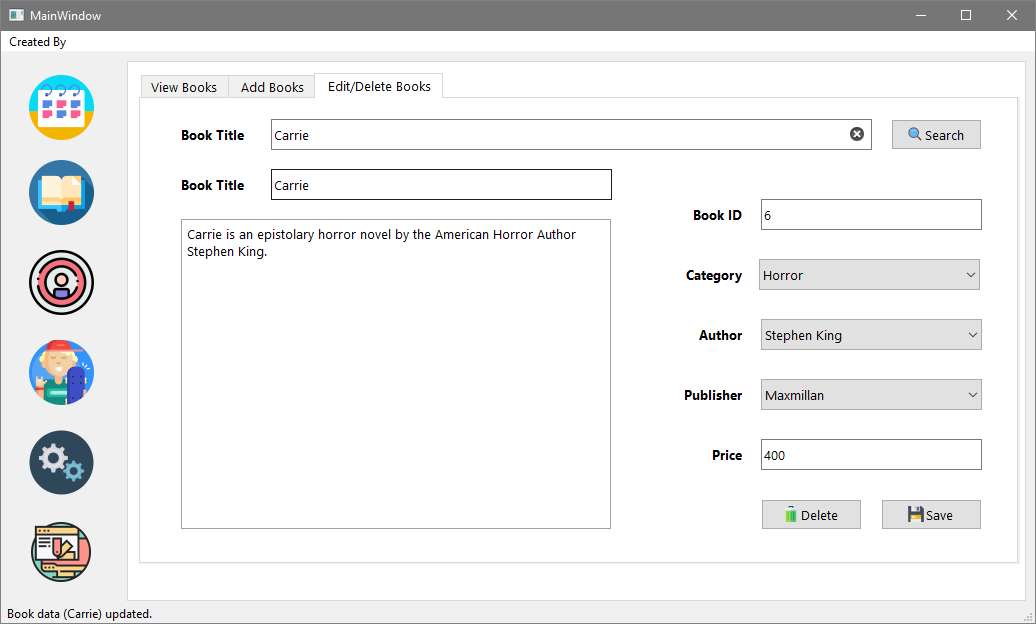
**Searching Books:**



***Image:*** *Book data successfully fetched.*

***Book data returned***

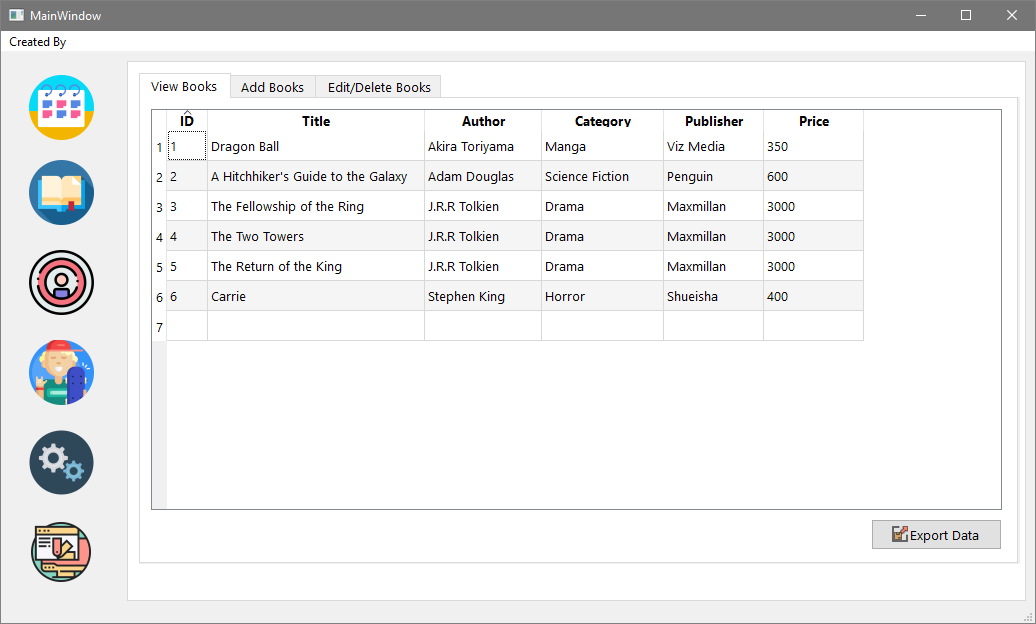
**Editing Book Data:**



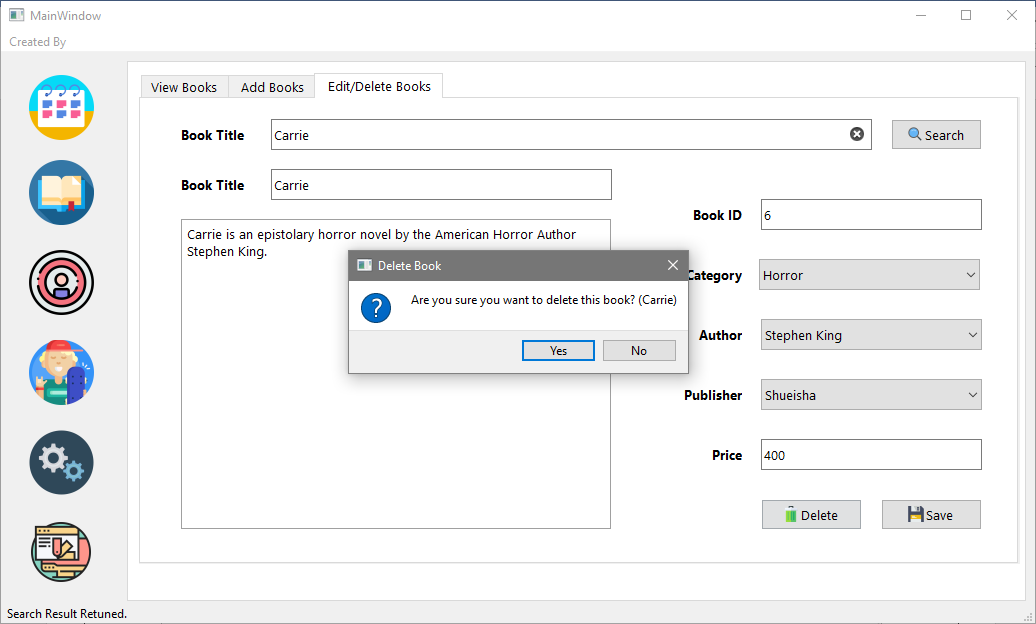
***Changing the publisher***

***Changing the price***

***Updation Success***



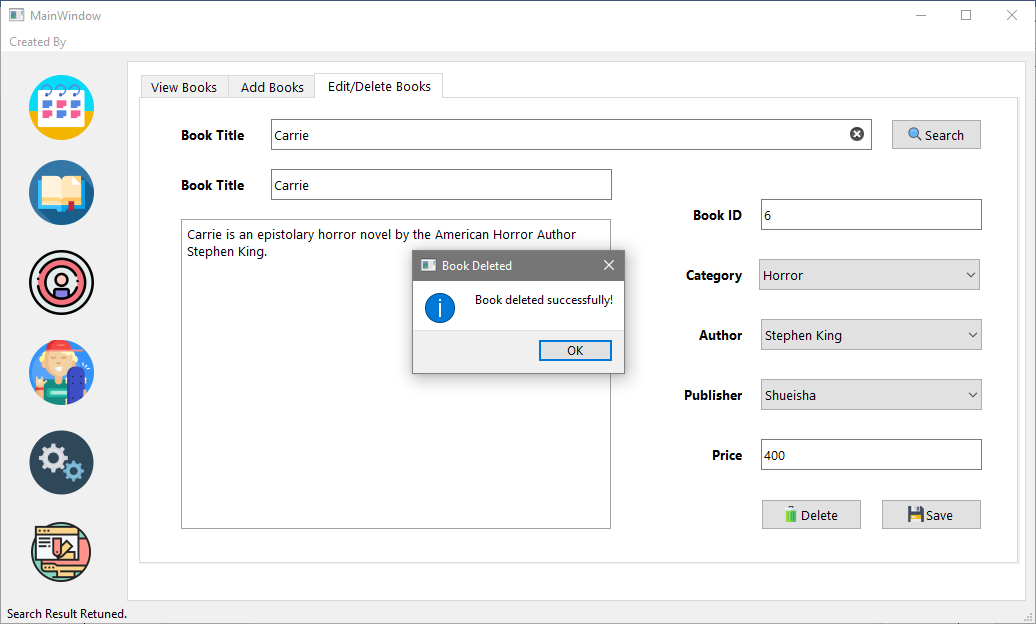
***Updated information can be seen in View Books tab***



***Image:*** *Delete Book from database.*

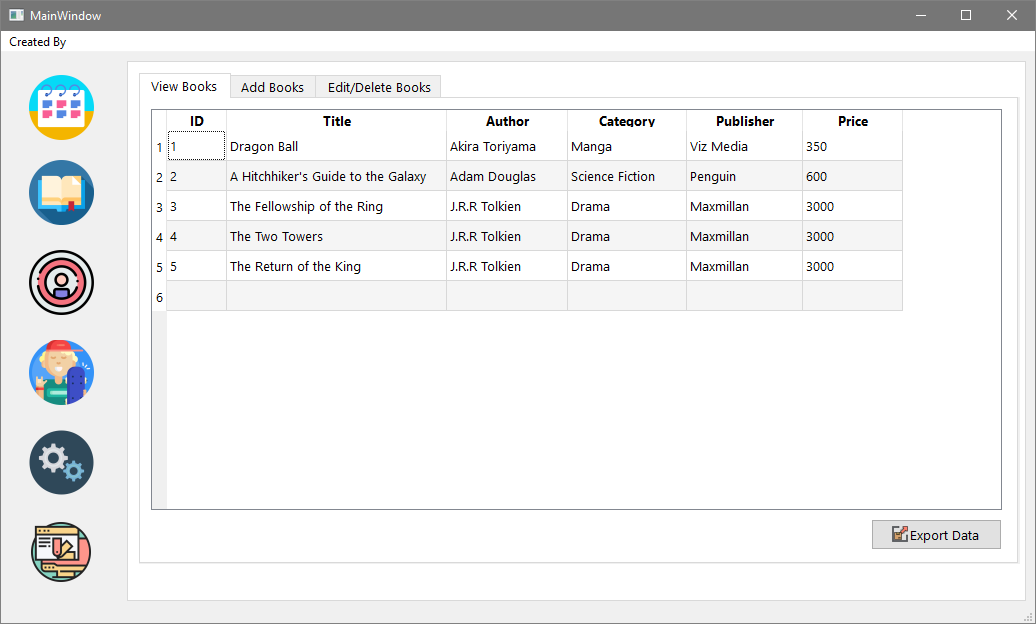
***Asks confirmation from user***

**Deleting a Book:**



***Successful deletion message***

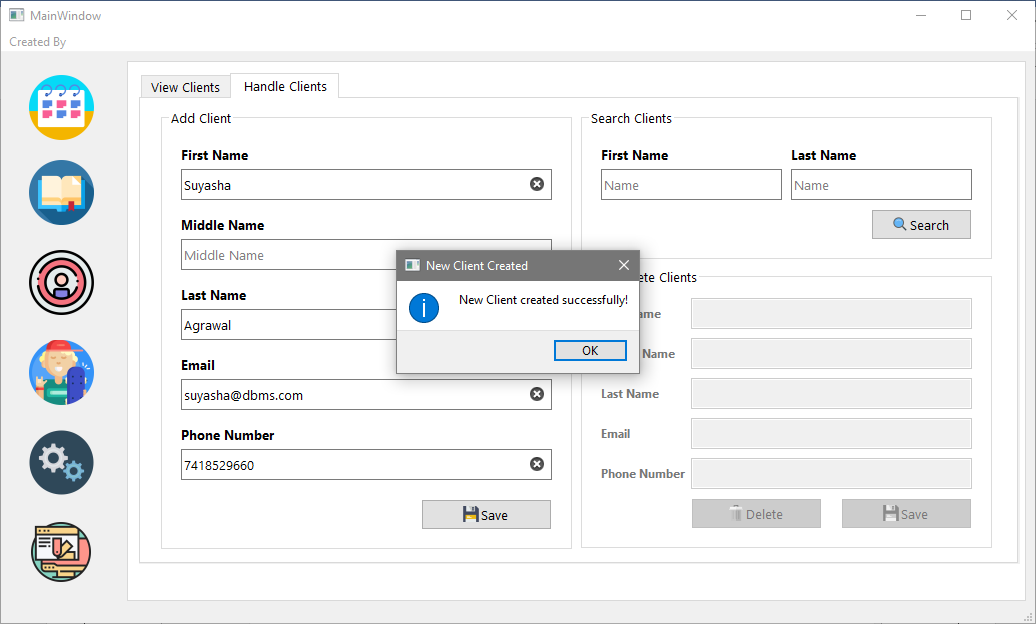
***Image:*** *Delete User from database.*



***Deleted book cannot be viewed***

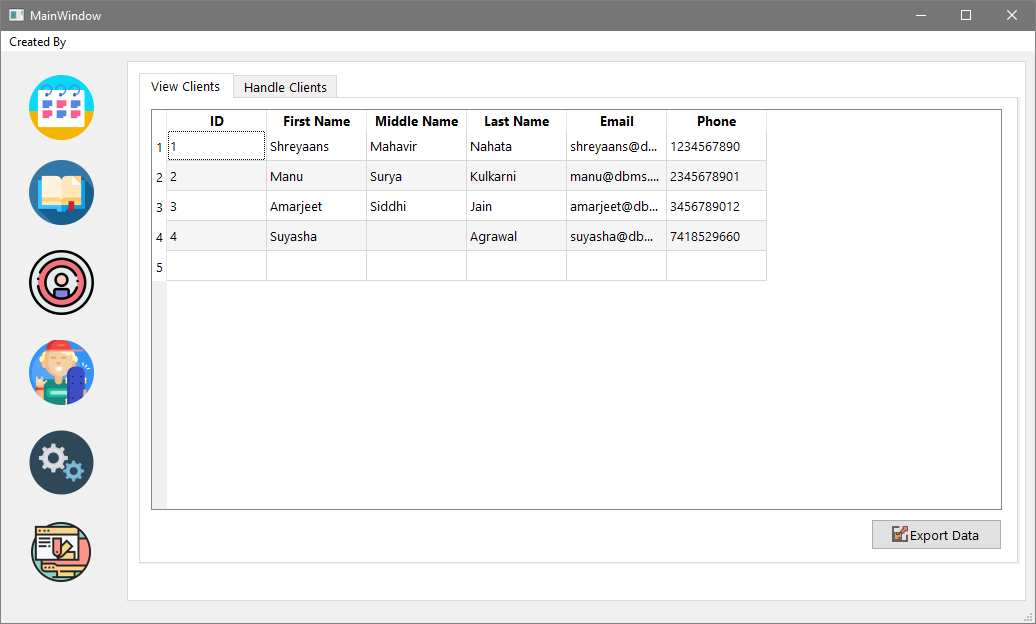
***Image:*** *View books in the database.*

**Adding a New Client:**



***New Client successfully created***

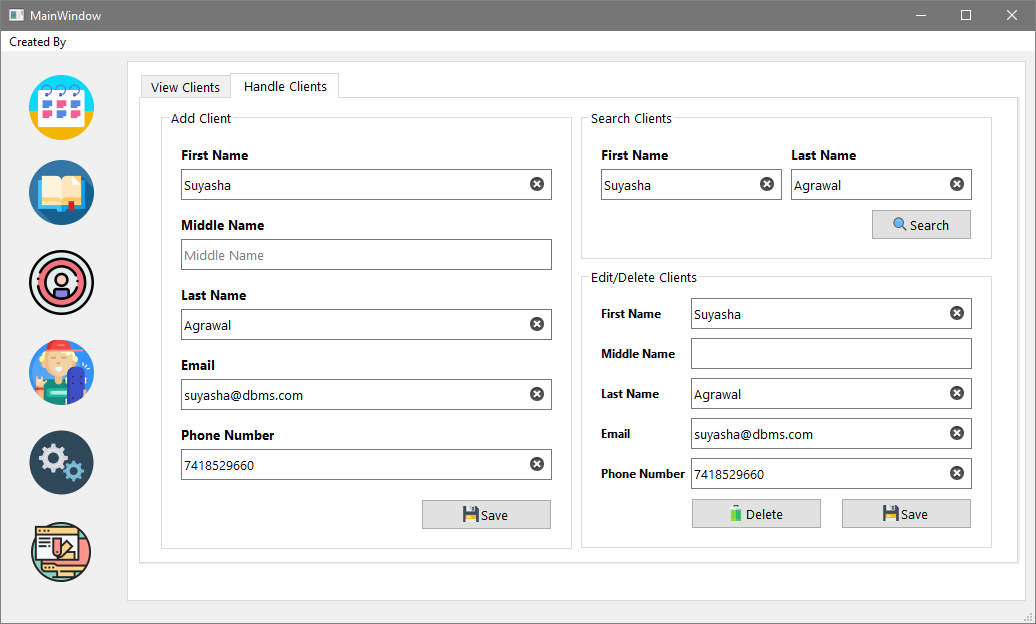
***Image:*** *Add new client to database.*



***New Client can be seen***

***Image:*** *View Clients in the database.*

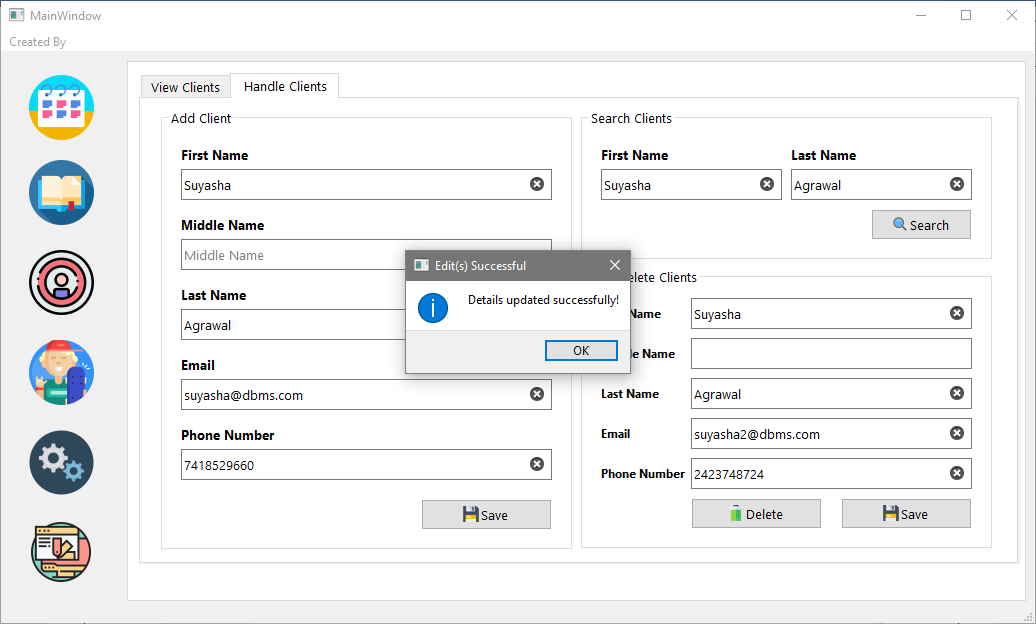
**Searching a Client:**



***Client Details Fetched***

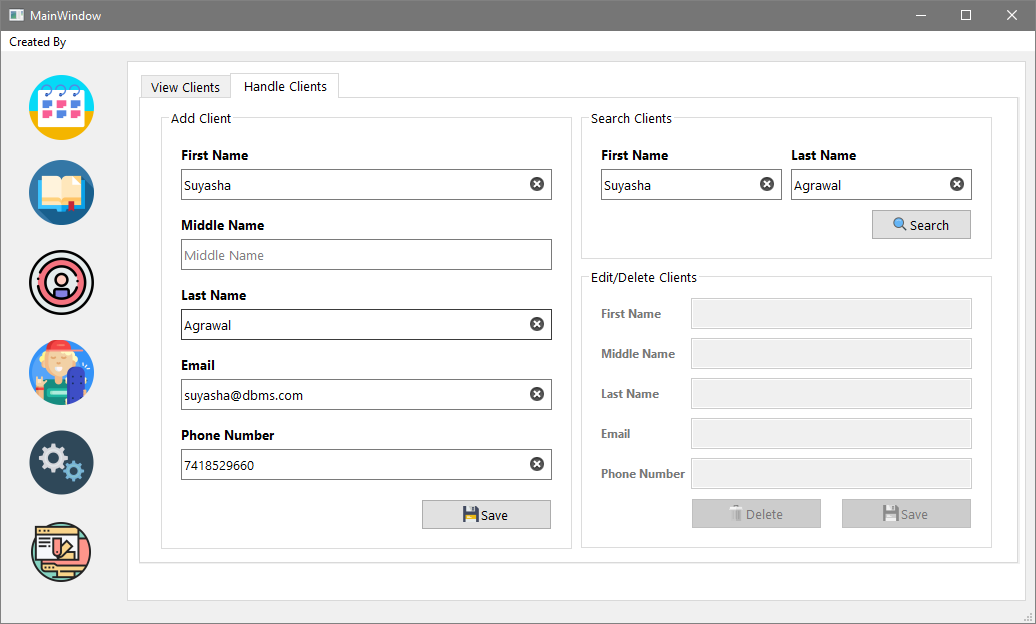
***Image:*** *Handle Clients tab: Search clients.*

**Editing Client Details:**



***Client Details Updated Successfully***

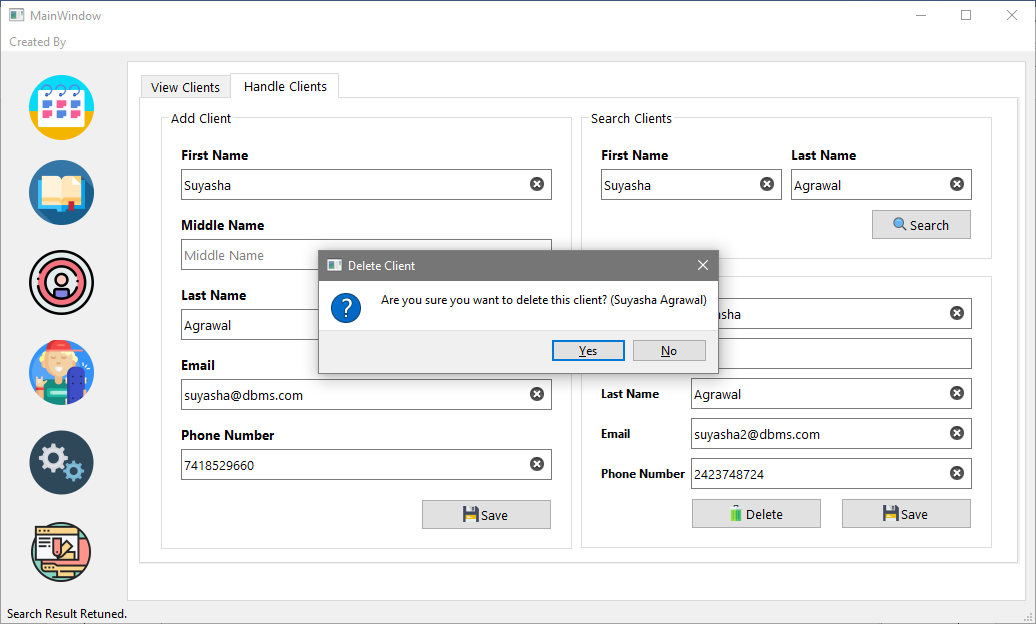
***Image:*** *Handle Clients tab: Edit client details.*



***Group Box disabled***

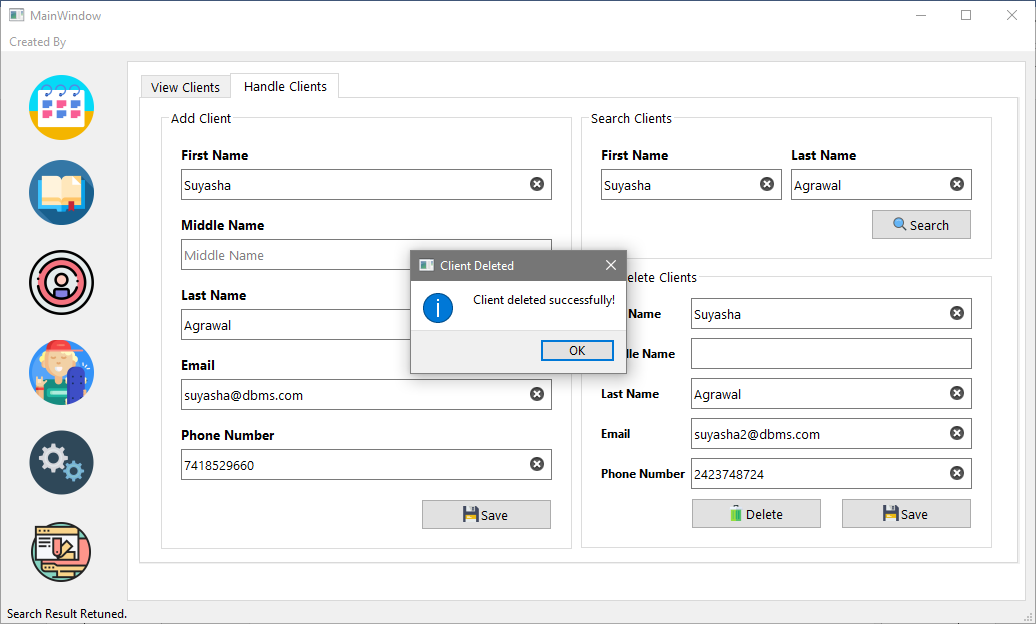
***Image:*** *Handle Clients tab: Post-edit success.*

**Deleting a Client:**



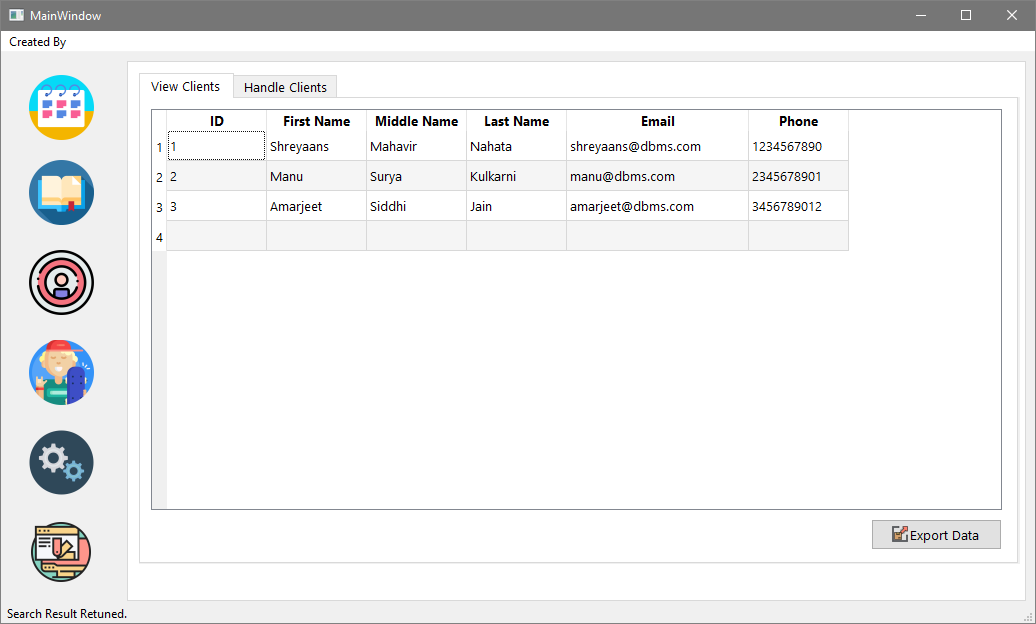
***Image:*** *Delete Client from database.*

***Asking the User for Confirmation***



***Successful deletion of client***

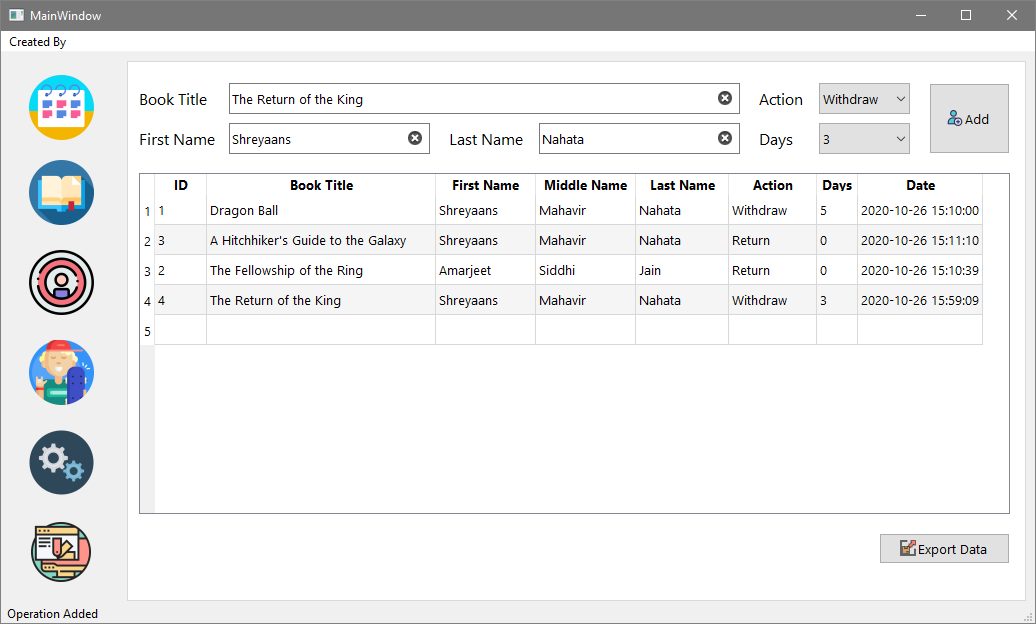
***Image:*** *Delete Client from database.*



***Image:*** *View Clients Tab.*

***Deleted Client cannot be seen***

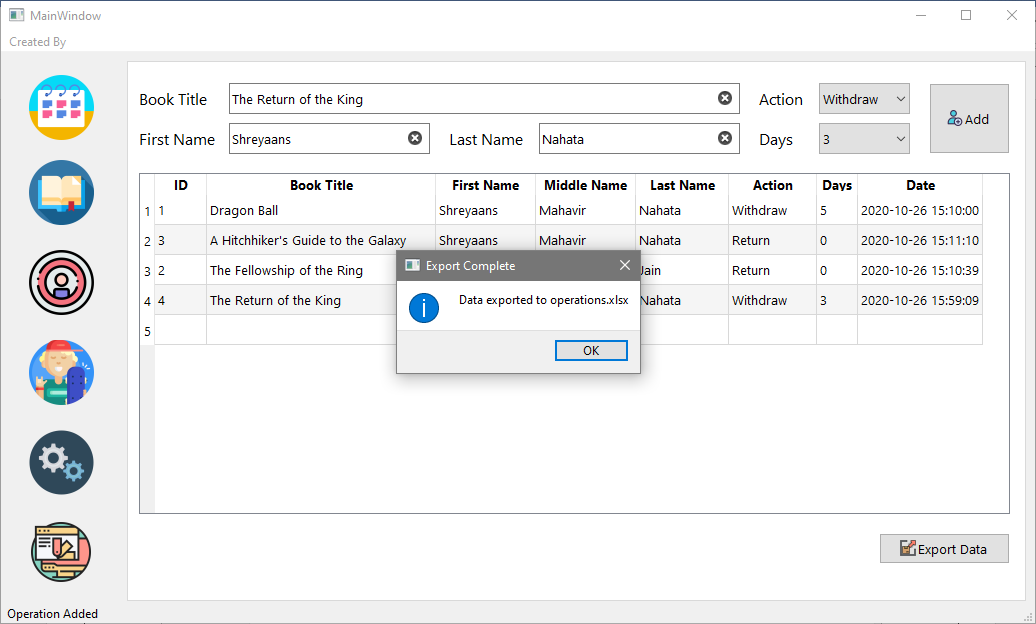
**Add a New Operation:**



***New Operation Added***

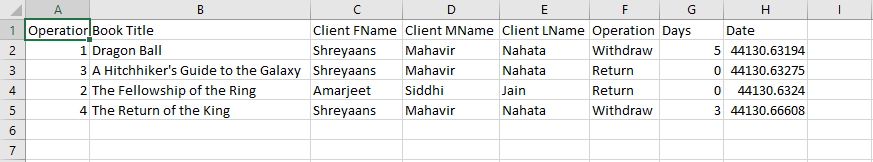
***Image:*** *Operations Page: Adding a new operation.*

**Exporting Data to .xlsx format:**

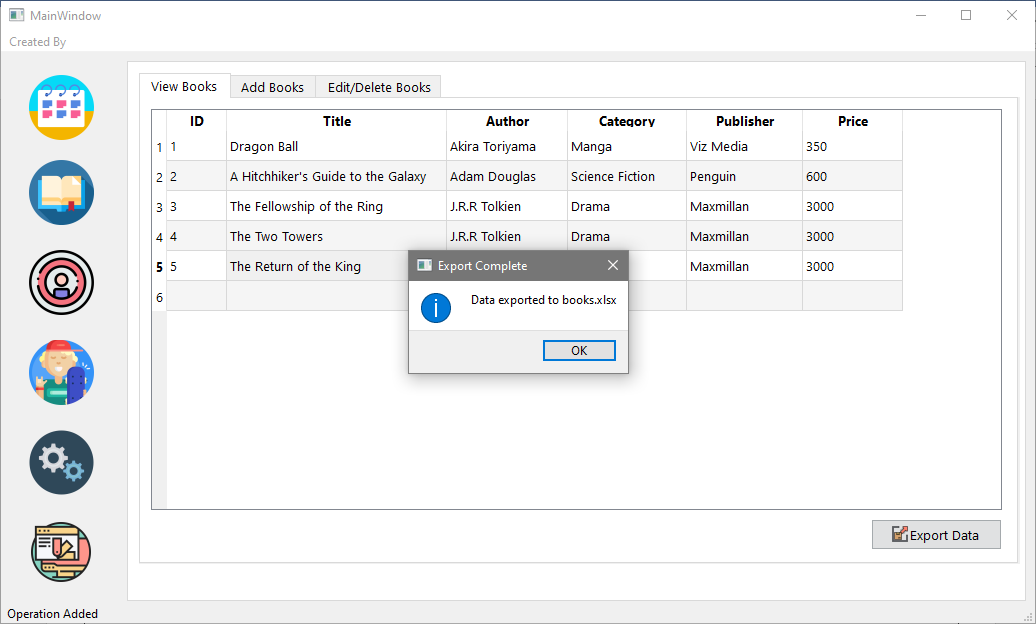


***Export completion message***

***Image:*** *Exporting operations data.*

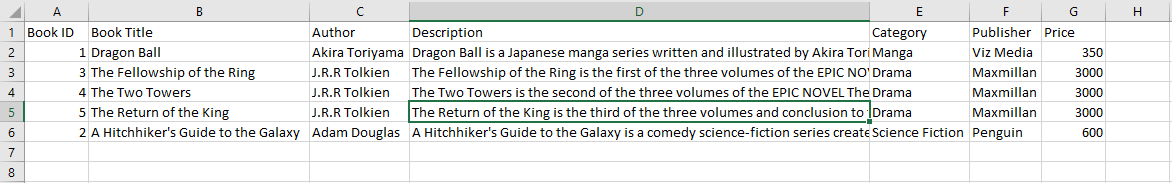


***Image:*** *Operations data exported to* ***operations.xlsx****.*

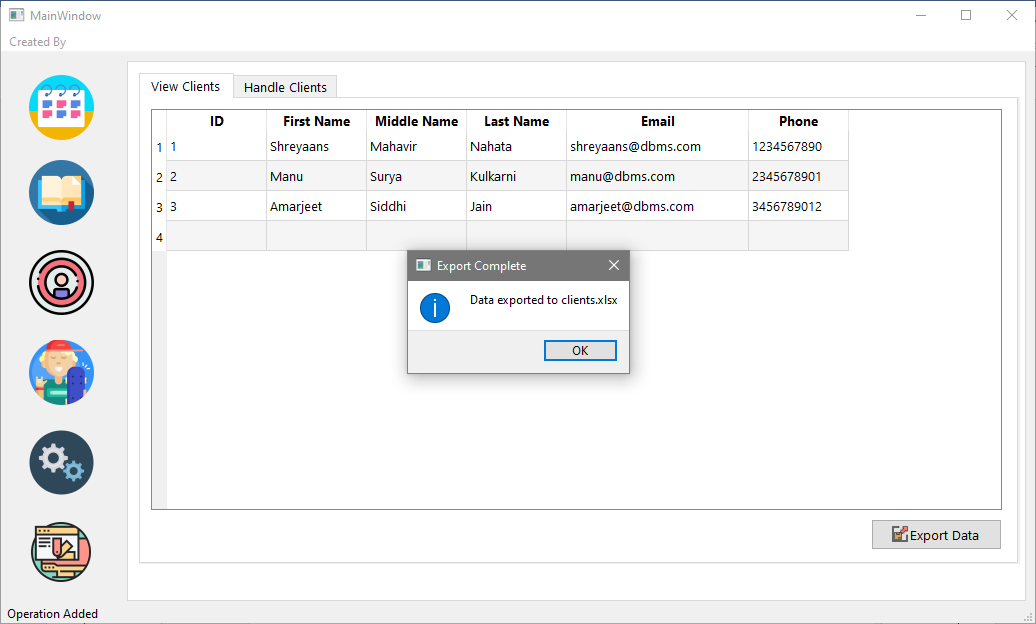


***Export completion message***

***Image:*** *Exporting books in the library.*

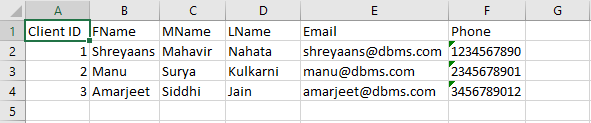


***Image:*** *Books data exported to* ***books.xlsx****.*



***Export completion message***

***Image:*** *Exporting client data.*



***Image:*** *Client data exported to* ***clients.xlsx****.*

**CONCLUSION**

Our library management System allows the user to store a large amount of information effectively and removes data redundancy. It also provides a user-friendly interface to let the end user interact with the database effectively without having to familiarize himself with the underlying technical details. This implementation of the system reduces the data entry time by a significant margin compared to using a more basic system. Another advantage is the reduction in human errors and the increase in efficiency. This overall reduces the human effort required.

The books are uniquely identified in our system and all the logs efficiently recorded. Hence making the search operation possible and effortless requiring just the press of a button. The information is thus accessed correctly and is without errors.

The system hence overcomes many of its predecessors’ shortcomings with its simplistic approach to its underlying database and minimalistic as well as aesthetic user interface providing functionalities such as light and dark mode for user comfort. Hence it is expected that this project will go a long way in satisfying user requirements and will increase efficiency while decreasing the stress of the database end users improving the overall human resources utilization.

**CODE**

**Find the code on:** <https://github.com/IAmOZRules/Library-Management-System>

# PyQt5 used to link to .ui modules

from PyQt5.QtWidgets import \*

from PyQt5.QtCore import \*

from PyQt5.QtGui import \*

# Used to read and write to .xlsx files

from xlsxwriter import \*

from xlrd import \*

import datetime             # Used to obtain the current date/time

import MySQLdb              # Connects the python and .ui to the Database

import sys

# Loads the main UI of the application

from PyQt5.uic import loadUiType

# Calls the UIs

ui, \_ = loadUiType('library.ui')

login, \_ = loadUiType('login.ui')

# Handles the Login Processes and UI

class Login (QWidget, login):

    # Loads the login UI

    def \_\_init\_\_(self):

        QWidget.\_\_init\_\_(self)

        self.setupUi(self)

        # Logs the user in on button click

        self.pushButton.clicked.connect(self.Handle\_Login)

    # Handles the Login process

    def Handle\_Login(self):

        # Connect to the MySQL Database

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        # Calls a cursor on the DB to access the data

        self.cur = self.db.cursor()

        # Get inserted information from the GUI as processable text

        username = self.lineEdit.text()

        password = self.lineEdit\_2.text()

        # SQL query to be executed in MySQL

        sql = ''' SELECT \* FROM users'''

        # Executes the SQL command

        self.cur.execute(sql)

        # Stores the result of the executes SQL query

        data = self.cur.fetchall()

        # If no users in DB, allow login by default

        if data == ():

            # Displays confirmation if login successful

            info = QMessageBox.information(self, 'Login Successful','Login Successful!', QMessageBox.Ok)

            # If 'Ok' is clicked, performs the specified actions

            if info == QMessageBox.Ok:

                # Open the Main App Window if login successful

                self.window2 = MainApp()

                self.close()                    # Close the login window

                self.window2.show()             # Shows the MainApp window

        # Iterates through the dat

        for row in data:

            # Enables login via username and email both

            if row[1] == username or row[2] == username:

                # Checks if the password is correct

                if row[3] ==  password:

                    # Gives a success message

                    info = QMessageBox.information(self, 'Login Successful','Login Successful!', QMessageBox.Ok)

                    if info == QMessageBox.Ok:

                        # Opens the Main App window

                        self.window2 = MainApp()

                        self.close()

                        self.window2.show()

                else:

                    warning = QMessageBox.warning(self, 'Incorrect Details','Please enter the correct login details.', QMessageBox.Ok)

                    if warning == QMessageBox.Ok:

                        self.lineEdit\_2.setText('')

# Handles the Main Application UI

class MainApp (QMainWindow, ui):

    # Handles all the main functions after application loading

    def \_\_init\_\_(self):

        QMainWindow.\_\_init\_\_(self)

        self.setupUi(self)

        # Calling the functions required on application startup

        self.Handle\_UI\_Changes()

        self.Handle\_buttons()

        # Shows the associated data in the respective tables

        self.Show\_Operations()

        self.Show\_Category()

        self.Show\_Publisher()

        self.Show\_Author()

        self.Show\_Books()

        self.Show\_Clients()

        # Shows the associated data in the respective comboboxes

        self.Combobox\_Author()

        self.Combobox\_Category()

        self.Combobox\_Publisher()

    # Handles the UI changes

    def Handle\_UI\_Changes(self):

        self.Hide\_Themes()

        self.tabWidget.tabBar().setVisible(False)

    # Handles the buttons

    def Handle\_buttons(self):

        # Handle the Operations

        self.pushButton\_3.clicked.connect(self.Handle\_Operations)

        # Show/Hide Themes

        self.pushButton\_8.clicked.connect(self.Show\_Themes)

        self.pushButton\_21.clicked.connect(self.Hide\_Themes)

        # Toggle between the various themes

        self.pushButton\_18.clicked.connect(self.aqua)

        self.pushButton\_19.clicked.connect(self.breezedark)

        self.pushButton\_20.clicked.connect(self.breezelight)

        self.pushButton\_27.clicked.connect(self.classic)

        self.pushButton\_28.clicked.connect(self.darkblue)

        self.pushButton\_31.clicked.connect(self.ubuntu)

        # Navigate between tabs

        self.pushButton.clicked.connect(self.Open\_Operations)

        self.pushButton\_2.clicked.connect(self.Open\_Books)

        self.pushButton\_26.clicked.connect(self.Open\_Clients)

        self.pushButton\_6.clicked.connect(self.Open\_Users)

        self.pushButton\_7.clicked.connect(self.Open\_Settings)

        # Add New Author, Publisher, Category

        self.pushButton\_14.clicked.connect(self.Add\_Author)

        self.pushButton\_15.clicked.connect(self.Add\_Publisher)

        self.pushButton\_16.clicked.connect(self.Add\_Category)

        # Delete an existing Author, Publisher, Category

        self.pushButton\_23.clicked.connect(self.Delete\_Author)

        self.pushButton\_24.clicked.connect(self.Delete\_Publisher)

        self.pushButton\_25.clicked.connect(self.Delete\_Category)

        # Book related operations

        self.pushButton\_4.clicked.connect(self.Add\_New\_Book)

        self.pushButton\_9.clicked.connect(self.Search\_Book)

        self.pushButton\_5.clicked.connect(self.Edit\_Book)

        self.pushButton\_10.clicked.connect(self.Delete\_Book)

        # Client related operations

        self.pushButton\_17.clicked.connect(self.Add\_Client)

        self.pushButton\_33.clicked.connect(self.Search\_Client)

        self.pushButton\_34.clicked.connect(self.Edit\_Client)

        self.pushButton\_35.clicked.connect(self.Delete\_Client)

        # User related operations

        self.pushButton\_11.clicked.connect(self.Add\_Users)

        self.pushButton\_12.clicked.connect(self.Login)

        self.pushButton\_13.clicked.connect(self.Edit\_User)

        self.pushButton\_22.clicked.connect(self.Delete\_User)

        # Export operations

        self.pushButton\_36.clicked.connect(self.Export\_Operations)

        self.pushButton\_29.clicked.connect(self.Export\_Books)

        self.pushButton\_30.clicked.connect(self.Export\_Clients)

    # Shows the themes tab

    def Show\_Themes(self):

        self.groupBox\_6.show()

    # Hides the themes tab

    def Hide\_Themes(self):

        self.groupBox\_6.hide()

    ################# Toggle between various tabs via buttons #################

    # Uses the Tab Indices to switch between tabs

    def Open\_Operations(self):

        self.tabWidget.setCurrentIndex(0)

    def Open\_Books(self):

        self.tabWidget.setCurrentIndex(1)

    def Open\_Clients(self):

        self.tabWidget.setCurrentIndex(2)

    def Open\_Users(self):

        self.tabWidget.setCurrentIndex(3)

    def Open\_Settings(self):

        self.tabWidget.setCurrentIndex(4)

    ################# Book Operations #################

    # Adds a new book

    def Add\_New\_Book(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        book\_title = self.lineEdit\_2.text()

        book\_desc = self.textEdit.toPlainText()

        book\_category = self.comboBox\_3.currentIndex()

        book\_author = self.comboBox\_4.currentIndex()

        book\_publisher = self.comboBox\_5.currentIndex()

        book\_price = self.lineEdit\_4.text()

        self.cur.execute('''

            INSERT INTO book (book\_name, book\_desc, category, author, publisher, price)

            VALUES (%s, %s, %s, %s, %s, %s)

        ''', (book\_title, book\_desc, book\_category, book\_author, book\_publisher, book\_price))

        self.db.commit()

        # Shows a confirmation message in the status bar

        self.statusBar().showMessage("New Book ({0}) Inserted.".format(book\_title))

        # Resets the respcetive fields once entry is done

        self.lineEdit\_2.setText('')

        self.textEdit.setPlainText('')

        self.comboBox\_3.setCurrentIndex(0)

        self.comboBox\_4.setCurrentIndex(0)

        self.comboBox\_5.setCurrentIndex(0)

        self.lineEdit\_4.setText('')

        # Updates the 'View Books' tab to show the recently added book(s)

        self.Show\_Books()

    # Search for an existing book in the DB

    def Search\_Book(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        book\_title = self.lineEdit\_8.text()

        # If search bar empty, display message

        if (book\_title == ''):

            self.statusBar().showMessage("No Book Found.")

        # Execute the search

        else:

            sql = ''' SELECT \* FROM book where book\_name = %s'''

            self.cur.execute(sql, [(book\_title)])

            # Fetch only one entry from the database

            data = self.cur.fetchone()

            # Returns the formatted and processed data in the respective fields

            self.lineEdit\_6.setText(data[1])

            self.textEdit\_2.setPlainText(data[2])

            self.lineEdit\_7.setText(str(data[0]))

            self.comboBox\_7.setCurrentIndex(data[3])

            self.comboBox\_6.setCurrentIndex(data[4])

            self.comboBox\_8.setCurrentIndex(data[5])

            self.lineEdit\_5.setText(str(data[6]))

            self.statusBar().showMessage("Search Result Retuned.")

    # Edit details for an existing book

    def Edit\_Book(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        search\_book\_title = self.lineEdit\_8.text()

        book\_id = self.lineEdit\_7.text()

        book\_title = self.lineEdit\_8.text()

        book\_desc = self.textEdit\_2.toPlainText()

        book\_category = self.comboBox\_7.currentIndex()

        book\_author = self.comboBox\_6.currentIndex()

        book\_publisher = self.comboBox\_6.currentIndex()

        book\_price = self.lineEdit\_5.text()

        self.cur.execute('''

            UPDATE book SET book\_id=%s, book\_name=%s, book\_desc=%s, category=%s, author=%s, publisher=%s, price=%s WHERE book\_name = %s

        ''', (book\_id, book\_title, book\_desc, str(book\_category), str(book\_author), str(book\_publisher), str(book\_price), search\_book\_title))

        self.db.commit()

        self.statusBar().showMessage("Book data ({0}) updated.".format(search\_book\_title))

        self.Show\_Books()

    # Delete an existing book

    def Delete\_Book(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        search\_book\_title = self.lineEdit\_8.text()

        warning = QMessageBox.question(self, 'Delete Book','Are you sure you want to delete this book? ({0})'.format(search\_book\_title), QMessageBox.Yes | QMessageBox.No)

        # Asks for confirmation before deleting

        if (warning == QMessageBox.Yes):

            sql = ''' DELETE FROM book WHERE book\_name = %s '''

            self.cur.execute(sql, [(search\_book\_title )])

            self.db.commit()

            QMessageBox.information(self, 'Book Deleted','Book deleted successfully!', QMessageBox.Ok)

        self.Show\_Books()

    ################# Client Operations #################

    # Add a new Client to the DB

    def Add\_Client(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        first\_name = self.lineEdit\_3.text()

        middle\_name = self.lineEdit\_23.text()

        last\_name = self.lineEdit\_25.text()

        client\_email = self.lineEdit\_20.text()

        phone = self.lineEdit\_26.text()

        self.cur.execute('''

            INSERT INTO clients (first\_name, middle\_name, last\_name, client\_email, phone) VALUES (%s, %s, %s, %s, %s)

        ''', (first\_name, middle\_name, last\_name, client\_email, phone))

        self.db.commit()

        self.db.close()

        QMessageBox.information(self, 'New Client Created','New Client created successfully!', QMessageBox.Ok)

        self.Show\_Clients()

    # Search for an existing client

    def Search\_Client(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        first\_name = self.lineEdit\_34.text()

        last\_name = self.lineEdit\_35.text()

        self.cur.execute('''

            SELECT \* FROM clients WHERE first\_name = %s AND last\_name = %s

        ''',(first\_name, last\_name))

        data = self.cur.fetchone()

        # Enables the group box after a successful search

        self.groupBox\_10.setEnabled(True)

        self.lineEdit\_32.setText(data[1])

        self.lineEdit\_36.setText(data[2])

        self.lineEdit\_37.setText(data[3])

        self.lineEdit\_33.setText(data[4])

        self.lineEdit\_38.setText(str(data[5]))

        self.statusBar().showMessage("Search Result Retuned.")

    # Edit details for an existing client

    def Edit\_Client(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        search\_fname = self.lineEdit\_34.text()

        search\_lname = self.lineEdit\_35.text()

        first\_name = self.lineEdit\_32.text()

        middle\_name = self.lineEdit\_36.text()

        last\_name = self.lineEdit\_37.text()

        client\_email = self.lineEdit\_33.text()

        phone = self.lineEdit\_38.text()

        self.cur.execute('''

            UPDATE clients SET first\_name = %s, middle\_name = %s, last\_name = %s, client\_email = %s, phone = %s WHERE first\_name = %s AND last\_name = %s

        ''', (first\_name, middle\_name, last\_name, client\_email, phone, search\_fname, search\_lname))

        self.db.commit()

        QMessageBox.information(self, 'Edit(s) Successful','Details updated successfully!', QMessageBox.Ok)

        self.lineEdit\_32.setText('')

        self.lineEdit\_36.setText('')

        self.lineEdit\_37.setText('')

        self.lineEdit\_33.setText('')

        self.lineEdit\_38.setText('')

        # Disables the group box after a successful edit

        self.groupBox\_10.setEnabled(False)

        self.Show\_Clients()

    # Delete an exiting client

    def Delete\_Client(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        first\_name = self.lineEdit\_34.text()

        last\_name = self.lineEdit\_35.text()

        warning = QMessageBox.question(self, 'Delete Client','Are you sure you want to delete this client? ({0})'.format(

            first\_name+' '+last\_name), QMessageBox.Yes | QMessageBox.No)

        if (warning == QMessageBox.Yes):

            self.cur.execute(''' DELETE FROM clients  WHERE first\_name = %s AND last\_name = %s''', (first\_name, last\_name))

            self.db.commit()

            QMessageBox.information(self, 'Client Deleted','Client deleted successfully!', QMessageBox.Ok)

            self.lineEdit\_32.setText('')

            self.lineEdit\_36.setText('')

            self.lineEdit\_37.setText('')

            self.lineEdit\_33.setText('')

            self.lineEdit\_38.setText('')

            self.groupBox\_10.setEnabled(False)

        self.Show\_Clients()

    ################# User Operations #################

    # Add a new user

    def Add\_Users(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        username = self.lineEdit\_9.text()

        email = self.lineEdit\_10.text()

        password = self.lineEdit\_12.text()

        password2 = self.lineEdit\_11.text()

        # Adds the user only if both passwords are same

        if password == password2:

            self.cur.execute(''' INSERT INTO users (username, user\_email, user\_pwd) VALUES (%s, %s, %s)

            ''', (username, email, password))

            self.db.commit()

            QMessageBox.information(self, 'User Created','New user successfully created!', QMessageBox.Ok)

        # Throws error if both passwords not same

        else:

            warning = QMessageBox.warning(self, 'Password','Please enter same password in both fields.', QMessageBox.Ok)

            # Clears the password fields

            if warning == QMessageBox.Ok:

                self.lineEdit\_12.setText('')

                self.lineEdit\_11.setText('')

    # Login feature for the user to be able to edit their information

    def Login(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        username = self.lineEdit\_13.text()

        password = self.lineEdit\_14.text()

        sql = ''' SELECT \* FROM users'''

        self.cur.execute(sql)

        # Fetches all the rows from the DB

        data = self.cur.fetchall()

        for row in data:

            if row[1] == username:

                if row[3] ==  password:

                    # Enable the group box after successful login

                    self.groupBox\_7.setEnabled(True)

                    QMessageBox.information(self, 'Login Successful','Login Successful!', QMessageBox.Ok)

                    self.lineEdit\_17.setText(row[1])

                    self.lineEdit\_15.setText(row[2])

                else:

                    warning = QMessageBox.warning(self, 'Incorrect Details','Please enter the correct login details.', QMessageBox.Ok)

                    if warning == QMessageBox.Ok:

                        self.lineEdit\_14.setText('')

        self.lineEdit\_14.setText('')

    # Edit existing user information

    def Edit\_User(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        username = self.lineEdit\_13.text()

        email = self.lineEdit\_15.text()

        password = self.lineEdit\_18.text()

        password2 = self.lineEdit\_16.text()

        if password == password2:

            self.cur.execute('''

                UPDATE users SET user\_email = %s, user\_pwd = %s WHERE username = %s

            ''', (email, password, username))

            self.db.commit()

            QMessageBox.information(self, 'Edit(s) Successful','Details updated successfully!', QMessageBox.Ok)

            self.lineEdit\_17.setText('')

            self.lineEdit\_15.setText('')

            self.lineEdit\_18.setText('')

            self.lineEdit\_16.setText('')

            self.lineEdit\_13.setText('')

            self.groupBox\_7.setEnabled(False)

        else:

            warning = QMessageBox.warning(self, 'Password','Please enter same password in both fields.', QMessageBox.Ok)

            if warning == QMessageBox.Ok:

                self.lineEdit\_18.setText('')

                self.lineEdit\_16.setText('')

    # Delete existing user

    def Delete\_User(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        username = self.lineEdit\_17.text()

        password = self.lineEdit\_18.text()

        password2 = self.lineEdit\_16.text()

        if password == password2:

            warning = QMessageBox.question(self, 'Delete User','Are you sure you want to delete this user? ({0})'.format(username), QMessageBox.Yes | QMessageBox.No)

            if (warning == QMessageBox.Yes):

                self.cur.execute(''' DELETE FROM users  WHERE username = %s ''', (username,))

                self.db.commit()

                QMessageBox.information(self, 'User Deleted','User deleted successfully!', QMessageBox.Ok)

                self.lineEdit\_17.setText('')

                self.lineEdit\_15.setText('')

                self.lineEdit\_18.setText('')

                self.lineEdit\_16.setText('')

                self.groupBox\_7.setEnabled(False)

        else:

            warning = QMessageBox.warning(self, 'Password','Please enter same password in both fields.', QMessageBox.Ok)

            if warning == QMessageBox.Ok:

                self.lineEdit\_18.setText('')

                self.lineEdit\_16.setText('')

    ################# Settings Operations #################

    # Add a new Author

    def Add\_Author(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        auth\_name = self.lineEdit\_19.text()

        self.cur.execute('''

            INSERT INTO author (auth\_name) VALUES (%s)

        ''', (auth\_name,))

        self.db.commit()

        self.statusBar().showMessage("New Author ({0}) Inserted.".format(auth\_name))

        self.lineEdit\_19.setText('')

        self.Show\_Author()

        self.Combobox\_Author()

    # Add a new Publisher

    def Add\_Publisher(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        pub\_name = self.lineEdit\_22.text()

        self.cur.execute('''

            INSERT INTO publisher (pub\_name) VALUES (%s)

        ''', (pub\_name,))

        self.db.commit()

        self.statusBar().showMessage("New Publisher ({0}) Inserted.".format(pub\_name))

        self.lineEdit\_22.setText('')

        self.Show\_Publisher()

        self.Combobox\_Publisher()

    # Add a new category

    def Add\_Category(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        cat\_name = self.lineEdit\_24.text()

        self.cur.execute('''

            INSERT INTO category (cat\_name) VALUES (%s)

        ''', (cat\_name,))

        self.db.commit()

        self.statusBar().showMessage("New Category ({0}) Inserted.".format(cat\_name))

        self.lineEdit\_24.setText('')

        self.Show\_Category()

        self.Combobox\_Category()

    # Delete an existing author

    def Delete\_Author(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        auth\_name = self.lineEdit\_19.text()

        warning = QMessageBox.question(self, 'Delete Author','Are you sure you want to delete this author? ({0})'.format(auth\_name), QMessageBox.Yes | QMessageBox.No)

        if (warning == QMessageBox.Yes):

            self.cur.execute(''' DELETE FROM author WHERE auth\_name = %s ''', (auth\_name,))

            self.db.commit()

            QMessageBox.question(self, 'Author Deleted','Author deleted successfully!', QMessageBox.Ok)

            self.lineEdit\_19.setText('')

            self.Show\_Author()

            self.Combobox\_Author()

    # Delete an existing publisher

    def Delete\_Publisher(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        pub\_name = self.lineEdit\_22.text()

        warning = QMessageBox.question(self, 'Delete Publisher','Are you sure you want to delete this publisher? ({0})'.format(pub\_name), QMessageBox.Yes | QMessageBox.No)

        if (warning == QMessageBox.Yes):

            self.cur.execute(''' DELETE FROM publisher WHERE pub\_name = %s ''', (pub\_name,))

            self.db.commit()

            QMessageBox.question(self, 'Publisher Deleted','Publisher deleted successfully!', QMessageBox.Ok)

            self.lineEdit\_22.setText('')

            self.Show\_Publisher()

            self.Combobox\_Publisher()

    # Delete an existing category

    def Delete\_Category(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        cat\_name = self.lineEdit\_24.text()

        warning = QMessageBox.question(self, 'Delete Category','Are you sure you want to delete this category? ({0})'.format(cat\_name), QMessageBox.Yes | QMessageBox.No)

        if (warning == QMessageBox.Yes):

            self.cur.execute(''' DELETE FROM category WHERE cat\_name = %s ''', (cat\_name,))

            self.db.commit()

            QMessageBox.question(self, 'Category Deleted','Category deleted successfully!', QMessageBox.Ok)

            self.lineEdit\_24.setText('')

            self.Show\_Category()

            self.Combobox\_Category()

    ################# Operation Functions #################

    # Handles the Day-to-Day functioning of the library

    def Handle\_Operations(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        book\_title = self.lineEdit.text()

        first\_name = self.lineEdit\_27.text()

        last\_name = self.lineEdit\_21.text()

        action = self.comboBox.currentText()

        day = self.comboBox\_2.currentIndex()

        date = datetime.datetime.now()

        date = date.strftime("%Y-%m-%d %H:%M:%S")

        self.cur.execute(''' INSERT INTO operations(book\_id, client\_id, operation, days, date)

            VALUES ((SELECT book\_id FROM book WHERE book\_name = %s), (

                SELECT client\_id FROM clients WHERE first\_name = %s AND last\_name = %s), %s, %s, %s);

        ''', (book\_title, first\_name, last\_name, action, day, date))

        self.db.commit()

        self.statusBar().showMessage('Operation Added')

        self.Show\_Operations()

    ################# Show Data in Tables #################

    # Shows all Authors in the database

    def Show\_Author(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute(''' SELECT \* FROM author''')

        data = self.cur.fetchall()

        if data:

            self.tableWidget\_2.setRowCount(0)

            self.tableWidget\_2.insertRow(0)

            for row, form in enumerate(data):

                for column, item in enumerate(form):

                    self.tableWidget\_2.setItem(row, column, QTableWidgetItem(str(item)))

                    column += 1

                row\_position = self.tableWidget\_2.rowCount()

                self.tableWidget\_2.insertRow(row\_position)

    # Shows all publishers in the database

    def Show\_Publisher(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute(''' SELECT \* FROM publisher''')

        data = self.cur.fetchall()

        if data:

            self.tableWidget\_3.setRowCount(0)

            self.tableWidget\_3.insertRow(0)

            for row, form in enumerate(data):

                for column, item in enumerate(form):

                    self.tableWidget\_3.setItem(row, column, QTableWidgetItem(str(item)))

                    column += 1

                row\_position = self.tableWidget\_3.rowCount()

                self.tableWidget\_3.insertRow(row\_position)

    # Shows all categories in the database

    def Show\_Category(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute(''' SELECT \* FROM category''')

        data = self.cur.fetchall()

        if data:

            self.tableWidget\_4.setRowCount(0)

            self.tableWidget\_4.insertRow(0)

            for row, form in enumerate(data):

                for column, item in enumerate(form):

                    self.tableWidget\_4.setItem(row, column, QTableWidgetItem(str(item)))

                    column += 1

                row\_position = self.tableWidget\_4.rowCount()

                self.tableWidget\_4.insertRow(row\_position)

    # Shows all the info about the Books in the database

    def Show\_Books(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute('''SELECT book\_id, book\_name, auth\_name, cat\_name, pub\_name, price FROM book INNER JOIN category, publisher, author

                            WHERE book.category = category.cat\_id AND book.author = author.auth\_id AND book.publisher = publisher.pub\_id;''')

        data = self.cur.fetchall()

        if data:

            self.tableWidget\_5.setRowCount(0)

            self.tableWidget\_5.insertRow(0)

            for row, form in enumerate(data):

                for column, item in enumerate(form):

                    self.tableWidget\_5.setItem(row, column, QTableWidgetItem(str(item)))

                    column += 1

                row\_position = self.tableWidget\_5.rowCount()

                self.tableWidget\_5.insertRow(row\_position)

    # Shows all the clients in the database

    def Show\_Clients(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute('''SELECT \* FROM clients''')

        data = self.cur.fetchall()

        if data:

            self.tableWidget\_6.setRowCount(0)

            self.tableWidget\_6.insertRow(0)

            for row, form in enumerate(data):

                for column, item in enumerate(form):

                    self.tableWidget\_6.setItem(row, column, QTableWidgetItem(str(item)))

                    column += 1

                row\_position = self.tableWidget\_6.rowCount()

                self.tableWidget\_6.insertRow(row\_position)

    # Shows all the operations of the library

    def Show\_Operations(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute('''SELECT op\_id, book.book\_name, clients.first\_name, clients.middle\_name, clients.last\_name, operation, days, date

            FROM operations INNER JOIN book, clients

            WHERE operations.book\_id = book.book\_id AND operations.client\_id = clients.client\_id

        ''')

        data = self.cur.fetchall()

        if data:

            self.tableWidget.setRowCount(0)

            self.tableWidget.insertRow(0)

            for row, form in enumerate(data):

                for column, item in enumerate(form):

                    self.tableWidget.setItem(row, column, QTableWidgetItem(str(item)))

                    column += 1

                row\_position = self.tableWidget.rowCount()

                self.tableWidget.insertRow(row\_position)

    # Shows the authors in the database in the combobox for easier use

    def Combobox\_Author(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute(''' SELECT auth\_name FROM author ''')

        data = self.cur.fetchall()

        # Clears the combo box to prevent repetition of values after addition of new values

        self.comboBox\_4.clear()

        self.comboBox\_6.clear()

        # Adds '<none>' to the combobox for entering the data more consistently  in the DB

        self.comboBox\_4.addItem('<none>')

        self.comboBox\_6.addItem('<none>')

        for author in data:

            for i in author:

                self.comboBox\_4.addItem(i)

                self.comboBox\_6.addItem(i)

    # Shows the publishers in the database in the combobox for easier use

    def Combobox\_Publisher(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute(''' SELECT pub\_name FROM publisher ''')

        data = self.cur.fetchall()

        self.comboBox\_5.clear()

        self.comboBox\_8.clear()

        self.comboBox\_5.addItem('<none>')

        self.comboBox\_8.addItem('<none>')

        for publisher in data:

            for i in publisher:

                self.comboBox\_5.addItem(i)

                self.comboBox\_8.addItem(i)

    # Shows the categories in the database in the combobox for easier use

    def Combobox\_Category(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute(''' SELECT cat\_name FROM category ''')

        data = self.cur.fetchall()

        self.comboBox\_3.clear()

        self.comboBox\_7.clear()

        self.comboBox\_3.addItem('<none>')

        self.comboBox\_7.addItem('<none>')

        for category in data:

            for i in category:

                self.comboBox\_3.addItem(i)

                self.comboBox\_7.addItem(i)

    ################# Export functions ################

    # Exports the operations data into the 'operations.xlsx' file

    def Export\_Operations(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute('''SELECT op\_id, book.book\_name, clients.first\_name, clients.middle\_name, clients.last\_name, operation, days, date

            FROM operations INNER JOIN book, clients

            WHERE operations.book\_id = book.book\_id AND operations.client\_id = clients.client\_id ''')

        data = self.cur.fetchall()

        # Creates a workbook

        wb = Workbook('operations.xlsx')

        # Adds a worksheet to the workbook

        sheet1 = wb.add\_worksheet()

        # Create the header to better understand the exported file

        sheet1.write(0,0,'Operation ID')

        sheet1.write(0,1,'Book Title')

        sheet1.write(0,2,'Client FName')

        sheet1.write(0,3,'Client MName')

        sheet1.write(0,4,'Client LName')

        sheet1.write(0,5,'Operation')

        sheet1.write(0,6,'Days')

        sheet1.write(0,7,'Date')

        #### Iterate through 'data' and add a new row one by one

        # Writes the file from row-1 since row-0 has the respective headers

        row\_number = 1

        for row in data:

            column\_number = 0

            for item in row:

                sheet1.write(row\_number, column\_number, item)

                column\_number += 1

            row\_number += 1

        # Closes the open file

        wb.close()

        # Export completion confirmation

        QMessageBox.information(self, 'Export Complete','Data exported to operations.xlsx')

    # Exports the books data into the 'books.xlsx' file

    def Export\_Books(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute('''SELECT book\_id, book\_name, auth\_name, book\_desc, cat\_name, pub\_name, price FROM book INNER JOIN category, publisher, author

            WHERE book.category = category.cat\_id AND book.author = author.auth\_id AND book.publisher = publisher.pub\_id;''')

        data = self.cur.fetchall()

        wb = Workbook('books.xlsx')

        sheet1 = wb.add\_worksheet()

        sheet1.write(0,0,'Book ID')

        sheet1.write(0,1,'Book Title')

        sheet1.write(0,2,'Author')

        sheet1.write(0,3,'Description')

        sheet1.write(0,4,'Category')

        sheet1.write(0,5,'Publisher')

        sheet1.write(0,6,'Price')

        row\_number = 1

        for row in data:

            column\_number = 0

            for item in row:

                sheet1.write(row\_number, column\_number, item)

                column\_number += 1

            row\_number += 1

        wb.close()

        QMessageBox.information(self, 'Export Complete','Data exported to books.xlsx')

    # Exports the clients data into the 'clients.xlsx' file

    def Export\_Clients(self):

        self.db = MySQLdb.connect(host='localhost', user='root', password='2187', db='librarysys')

        self.cur = self.db.cursor()

        self.cur.execute('''SELECT \* FROM clients''')

        data = self.cur.fetchall()

        wb = Workbook('clients.xlsx')

        sheet1 = wb.add\_worksheet()

        sheet1.write(0,0,'Client ID')

        sheet1.write(0,1,'FName')

        sheet1.write(0,2,'MName')

        sheet1.write(0,3,'LName')

        sheet1.write(0,4,'Email')

        sheet1.write(0,5,'Phone')

        row\_number = 1

        for row in data:

            column\_number = 0

            for item in row:

                sheet1.write(row\_number, column\_number, item)

                column\_number += 1

            row\_number += 1

        wb.close()

        QMessageBox.information(self, 'Export Complete','Data exported to clients.xlsx')

    ################# Apply Themes ################

    # Aqua Theme

    def aqua(self):

        style = open('themes/aqua.css', 'r')        # Open the CSS syle sheet as read only

        style = style.read()                        # Reads the style sheet

        self.setStyleSheet(style)             # Set the style sheet for the application UI

    # Breeze Dark Theme

    def breezedark(self):

        style = open('themes/breezedark.css', 'r')

        style = style.read()

        self.setStyleSheet(style)

    # Breeze Light Theme

    def breezelight(self):

        style = open('themes/breezelight.css', 'r')

        style = style.read()

        self.setStyleSheet(style)

    # Classic Theme

    def classic(self):

        style = open('themes/classic.css', 'r')

        style = style.read()

        self.setStyleSheet(style)

    # Dark Blue Theme

    def darkblue(self):

        style = open('themes/darkblue.css', 'r')

        style = style.read()

        self.setStyleSheet(style)

    # Ubuntu Theme

    def ubuntu(self):

        style = open('themes/ubuntu.css', 'r')

        style = style.read()

        self.setStyleSheet(style)

# Driver code for the application

def main():

    app = QApplication(sys.argv)

    # Loads the Login page by default

    window = Login()

    window.show()

    app.exec()

# Calling the main function

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**REFERENCES**

[**01**] *Shasha Yu, Enhai Qiu and Mei Zhou*, “**Research on Library Management System Based on Java**”, Advances in Computer Science Research, Vol. 82, pp. 946-949, 2017

[**02**] *Roknuzzaman M, Kanai H, Umemoto K*, “**Integration of knowledge management process into digital library system**”, Library Review, 2013.

[**03**] *Seena S T, Pillaiw K G S*, “**A study of ICT skills among library professionals in the Kerala University Library System**”, Annals of Library & Information Studies, 2014.

[**04**] *Taole N, Dick A L*, “**Implementing a common library system for the Lesotho Library Consortium**”. Electronic Library, 2013.

[**05**] *Chen M, Cai W, Ma L*, “**Cloud Computing Platform for an Online Model Library System**”, Mathematical Problems in Engineering, 2013.

[**06**] *Hall K, Ames C M, Brice J*, “**Open Source Library Software Development in a Small Rural Library System**”, Code4lib Journal, 2013.

[**07**] *Uppal V, Chindwani G*, “**An Empirical Study of Application of Data Mining Techniques in Library System**”, Journal of Bacteriology, 2014.

[**08**] *Rao N S, Kumari N N*, “**Revitalisation of Public Library System in India: A CSR Perspective**”, Desidoc Journal of Library & Information Technology, 2013.

[**09**] *Pu Y H, Chiu P S, Chen T S*, et al. “**The design and implementation of a Mobile Library APP system**”. Library Hi Tech, 2015.

[**10**] *Iorio A D, Schaerf M*, “**The Organization information integration in the management of a Digital Library System**”, Digital Libraries. IEEE.

[**11**] *Fems, Seimiekumo Solomon, Zifawei O. Kennedy, George Deinbofa, Oberhiri Oruma Godwin* (2019), “**Design And Implementation Of Digital Library Management System. A Case Study Of The Niger Delta University, Bayelsa State**”, International Journal of Scientific and Research Publications (IJSRP) 9(12)

[**12**] *Li, H., & Cai, Z.-Q*, “**Design and implementation of the mobile library app based on smart phone**”, 2016 International Conference on Machine Learning and Cybernetics (ICMLC), 2016.