VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

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

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.

1.3 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.

1.4 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.

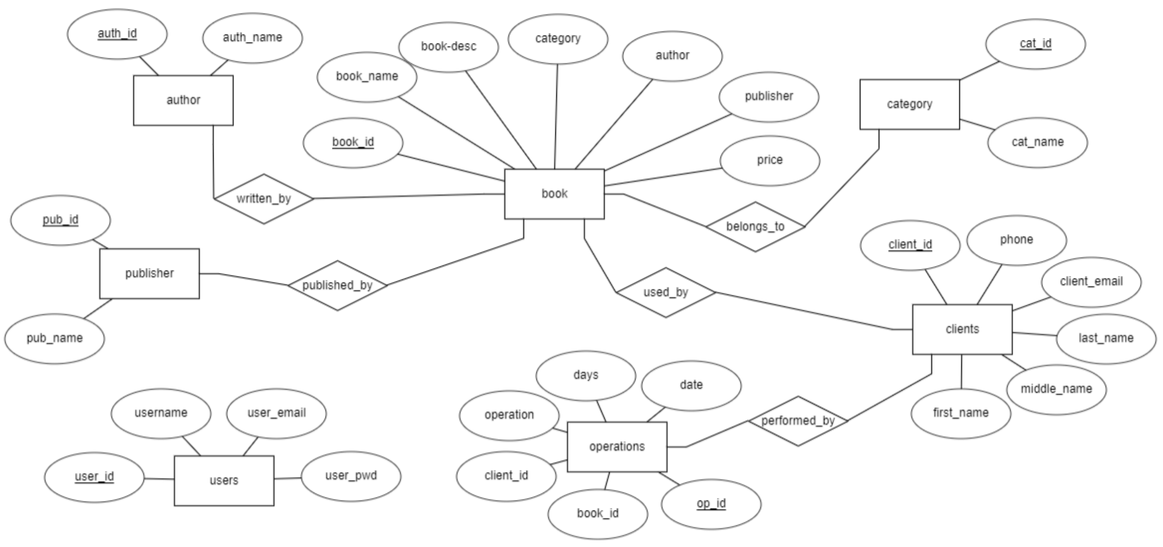
LITERATURE SURVEY

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.	

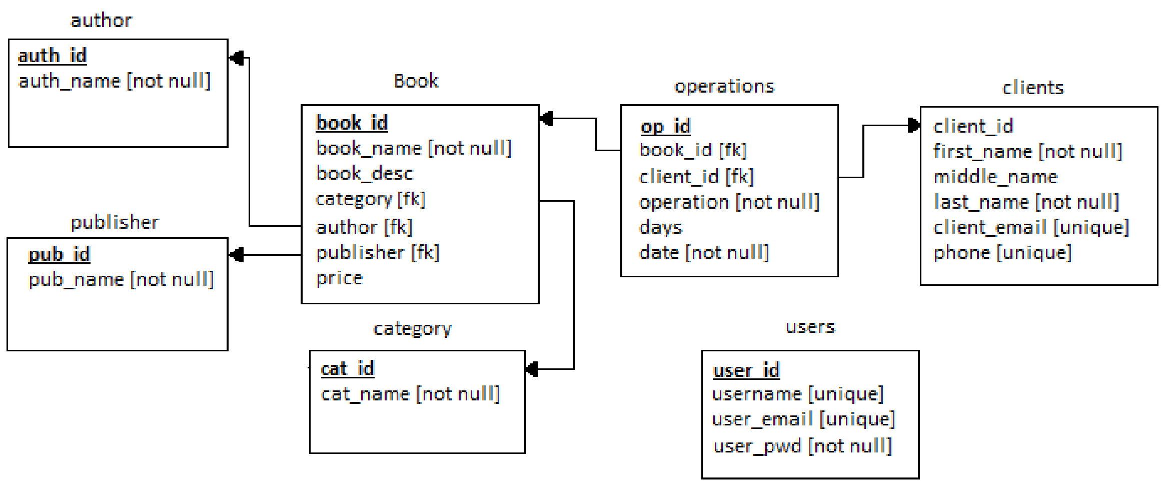
Lal Bahadur Chouhan (2010)	Provides open source software for library management study like KOHA, GSDL and open journal system (OJS) to study all functions and requirements of library automation.	
Ayodeji Iwayemi (2019)	Development of robust library manage system. They have made a website that allows 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 (<u>book_id</u>, book_name, book_desc, book_cat, author, publisher, price)

Users (<u>user_id</u>, username, user_email, user_pwd)

operations (<u>op_id</u>, book_id, client_id, operation days, date)

clients (<u>client_id</u>, first_name, middle_name, last_name, client_email, phone)

category (<u>cat_id</u>, cat_name)

author (<u>auth_id</u>, auth_name)

publisher (<u>pub_id</u>, 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:

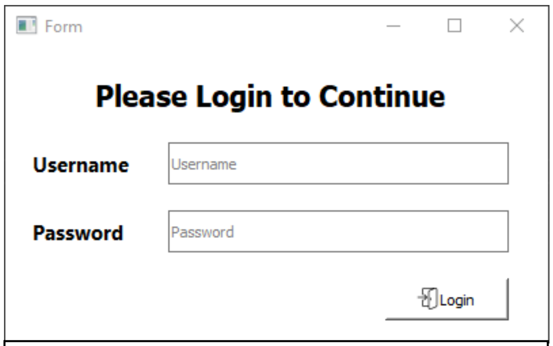


Image: *Main Login Page*

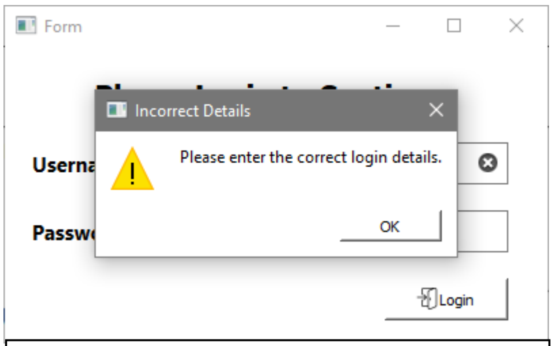


Image: *Error upon entering wrong login details*

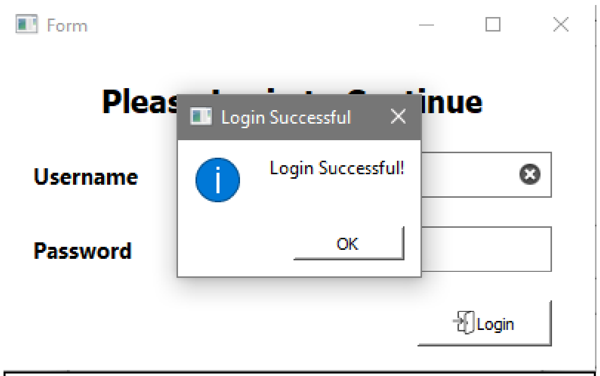
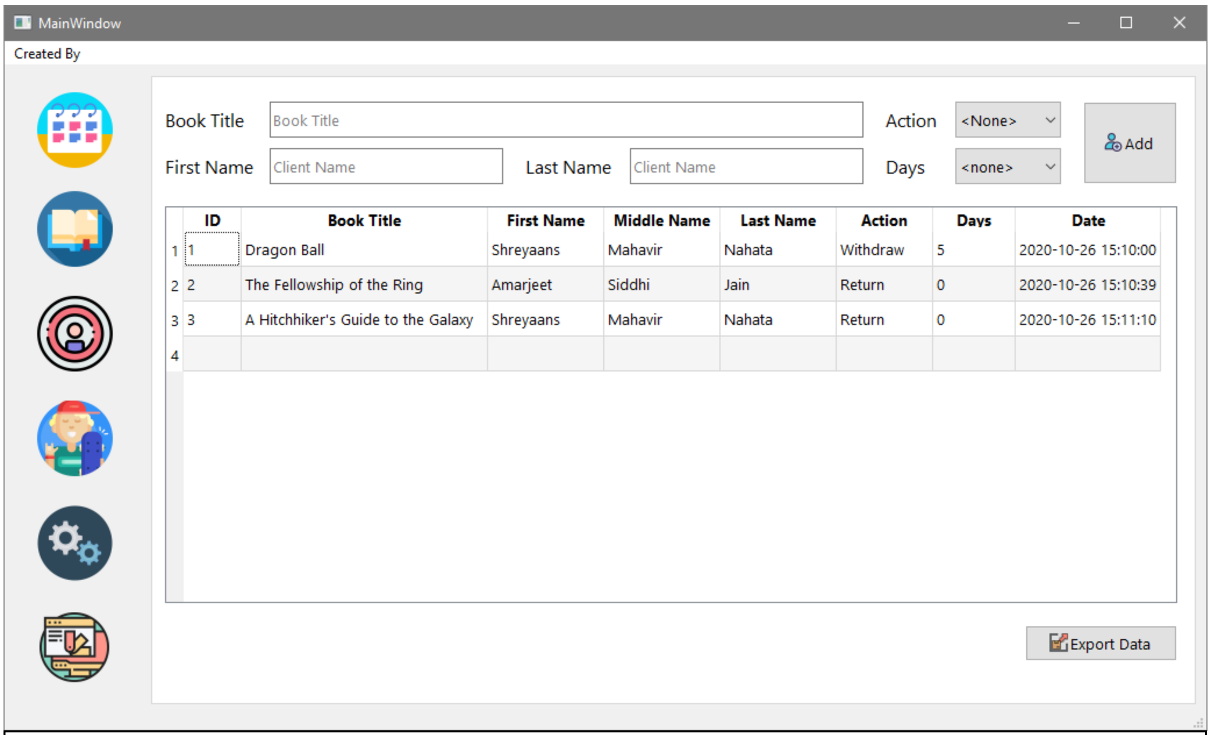
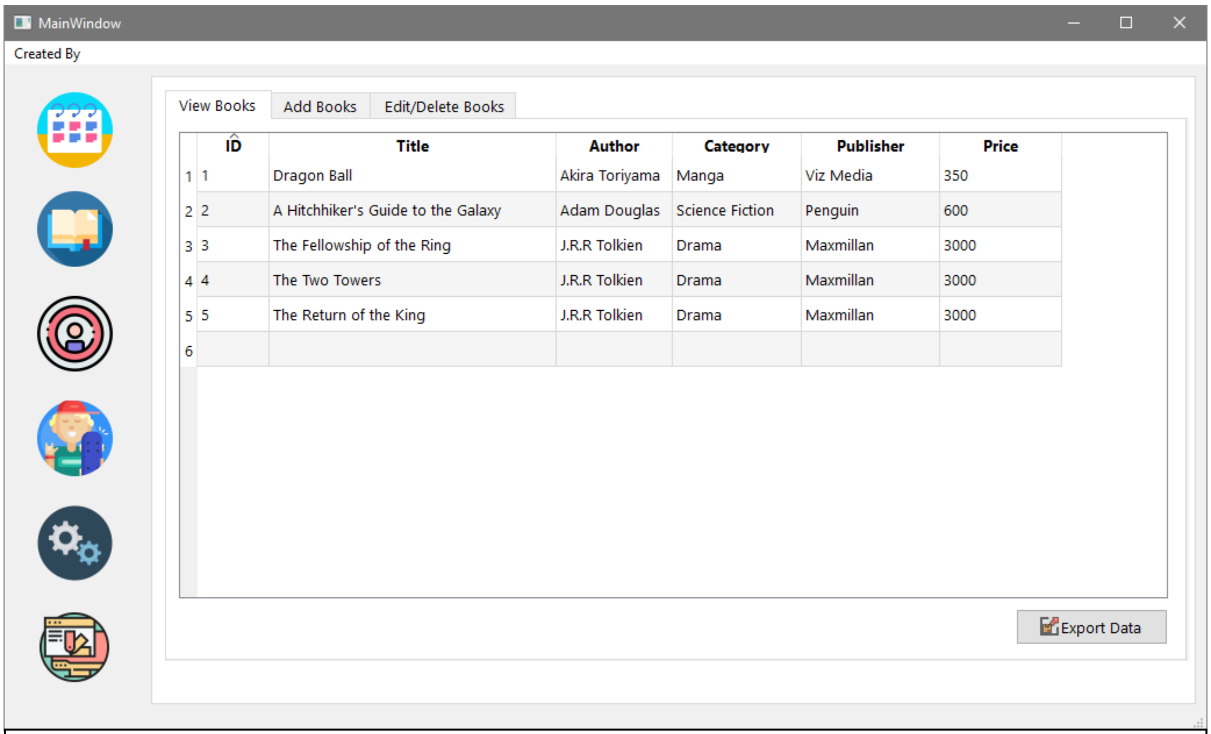


Image: *Login successful confirmation*



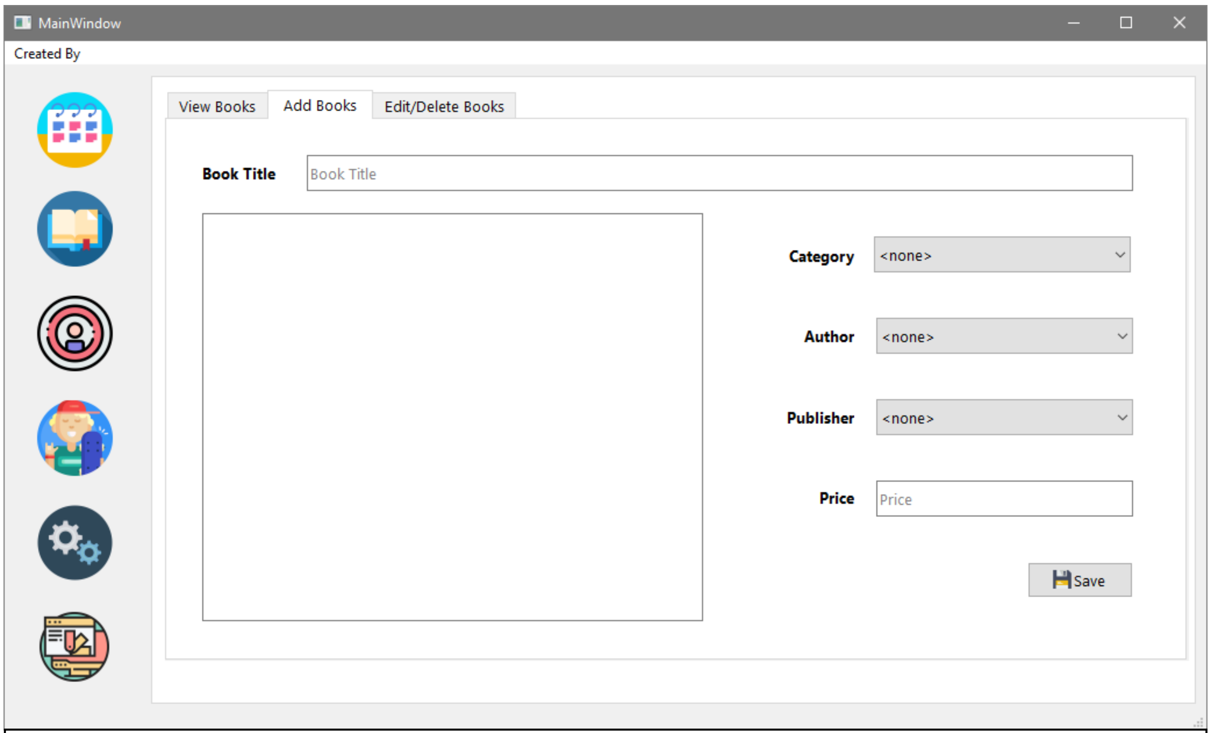
	ID	Book Title	First Name	Middle Name	Last Name	Action	Days	Date
1	1	Dragon Ball	Shreyaans	Mahavir	Nahata	Withdraw	5	2020-10-26 15:10:00
2	2	The Fellowship of the Ring	Amarjeet	Siddhi	Jain	Return	0	2020-10-26 15:10:39
3	3	A Hitchhiker's Guide to the Galaxy	Shreyaans	Mahavir	Nahata	Return	0	2020-10-26 15:11:10
4								

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



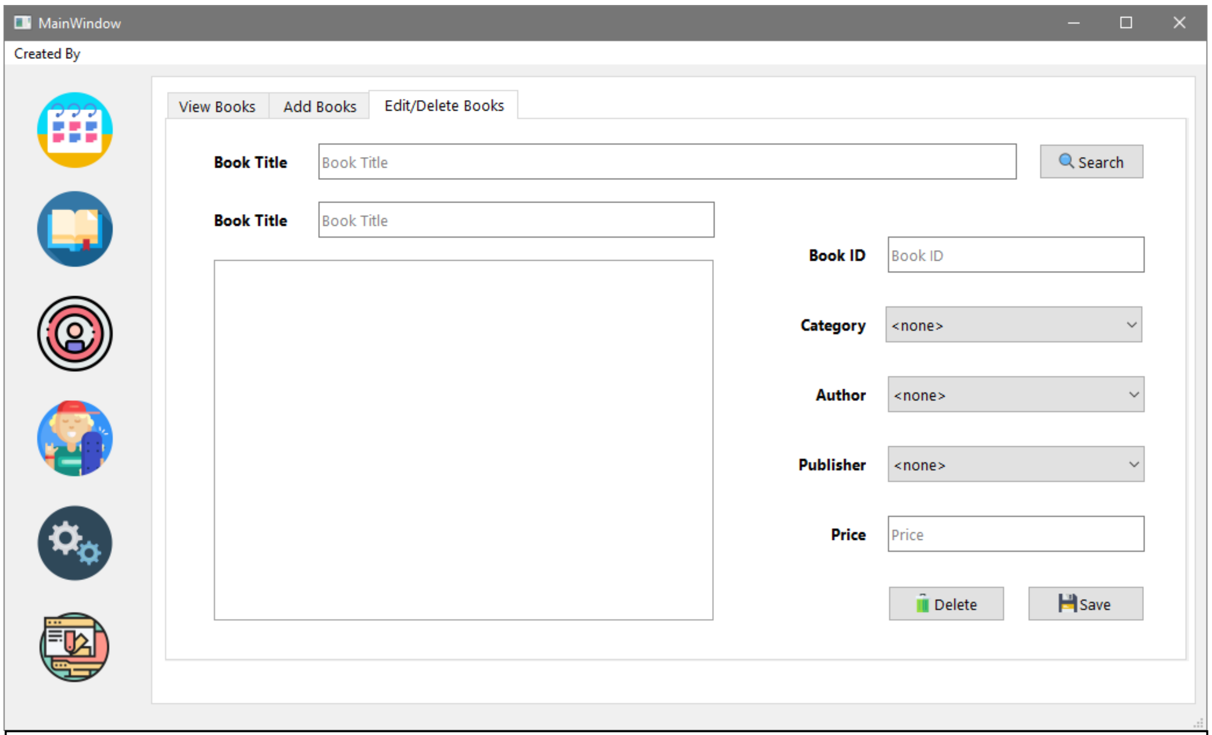
	ID	Title	Author	Category	Publisher	Price
1	1	Dragon Ball	Akira Toriyama	Manga	Viz Media	350
2	2	A Hitchhiker's Guide to the Galaxy	Adam Douglas	Science Fiction	Penguin	600
3	3	The Fellowship of the Ring	J.R.R Tolkien	Drama	Maxmillan	3000
4	4	The Two Towers	J.R.R Tolkien	Drama	Maxmillan	3000
5	5	The Return of the King	J.R.R Tolkien	Drama	Maxmillan	3000
6						

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



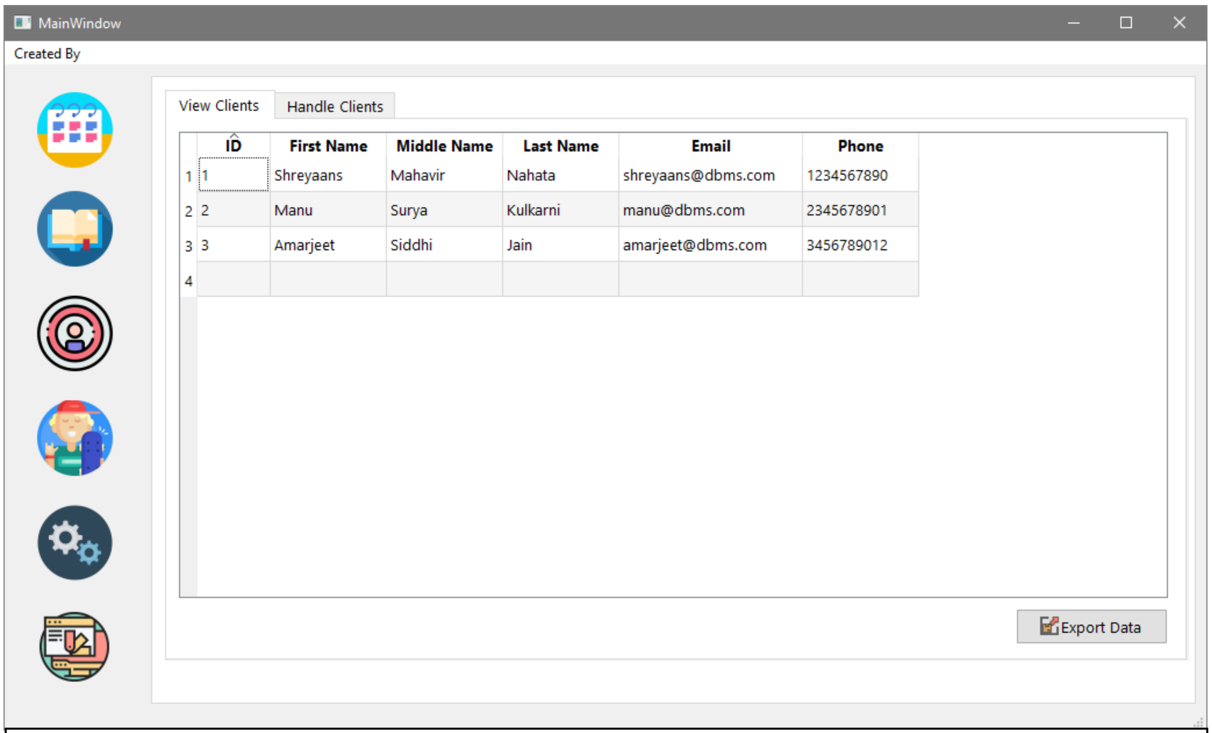
Book Title	Book Title
Category	<none>
Author	<none>
Publisher	<none>
Price	Price

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



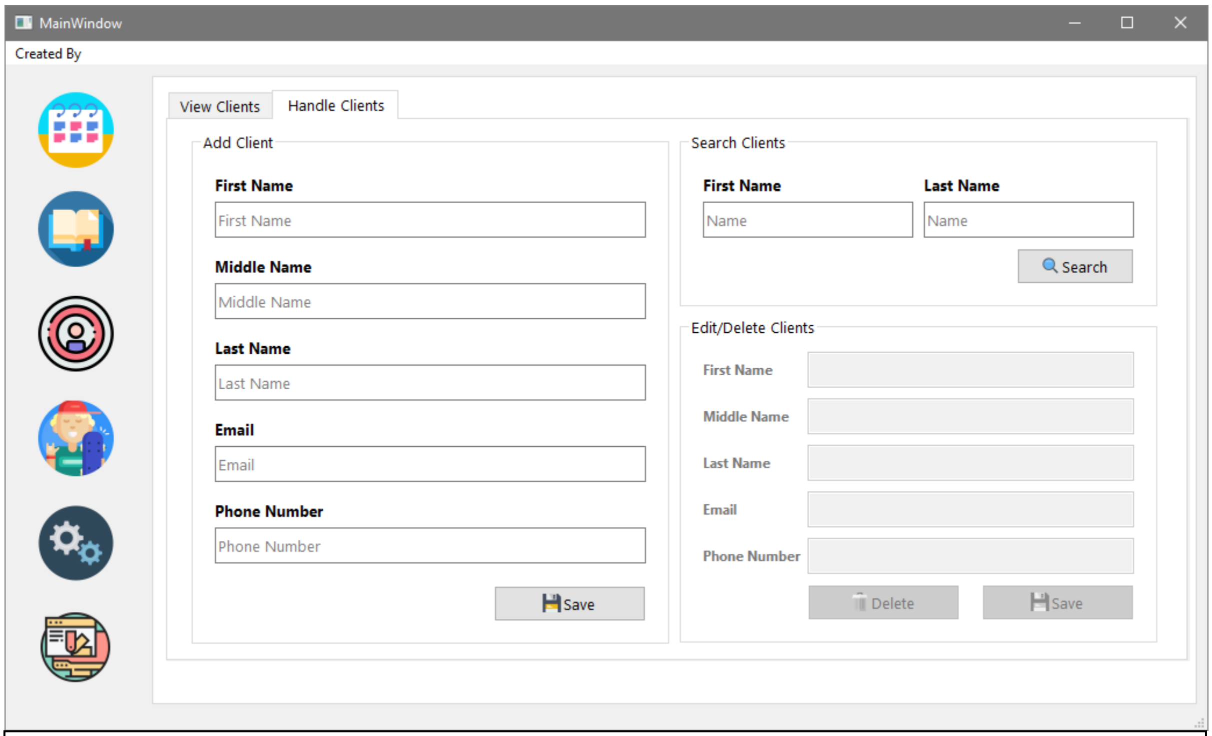
Book Title	Book Title
Book Title	Book Title
Book ID	Book ID
Category	<none>
Author	<none>
Publisher	<none>
Price	Price

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



	ID	First Name	Middle Name	Last Name	Email	Phone
1	1	Shreyaans	Mahavir	Nahata	shreyaans@dbms.com	1234567890
2	2	Manu	Surya	Kulkarni	manu@dbms.com	2345678901
3	3	Amarjeet	Siddhi	Jain	amarjeet@dbms.com	3456789012
4						

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


MainWindow

Created By

View Clients | Handle Clients

Add Client

First Name

First Name

Middle Name

Middle Name

Last Name

Last Name

Email

Email

Phone Number

Phone Number

Save

Search Clients

First Name | Last Name

Name | Name

Search

Edit/Delete Clients

First Name

Middle Name

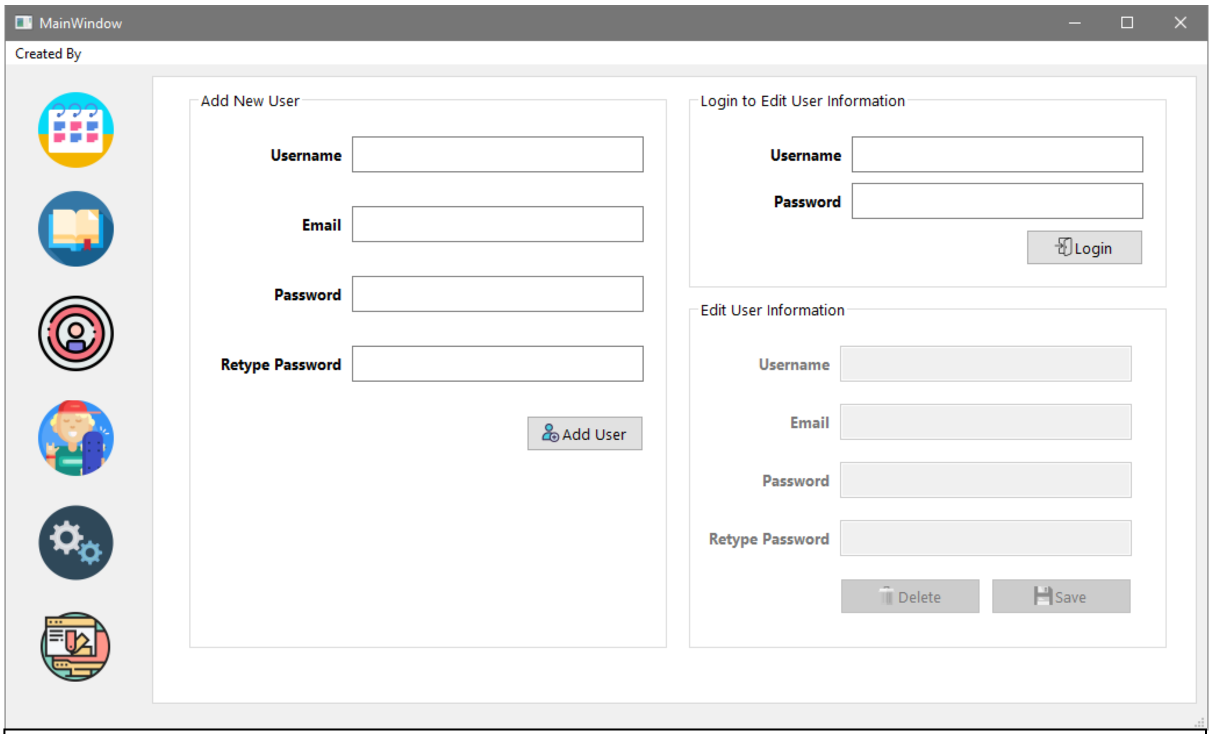
Last Name

Email

Phone Number

Delete | Save

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


MainWindow

Created By

Add New User

Username

Email

Password

Retype Password

Add User

Login to Edit User Information

Username

Password

Login

Edit User Information

Username

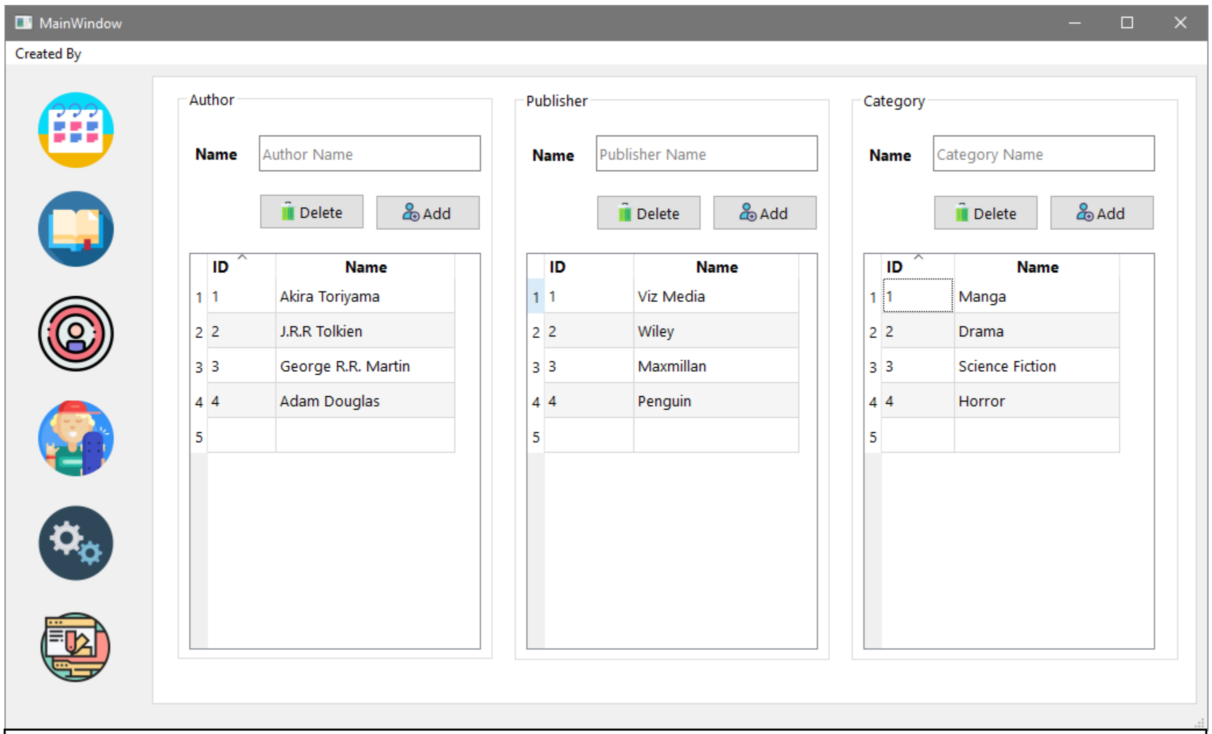
Email

Password

Retype Password

Delete | Save

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


MainWindow

Created By

Author

Name: Author Name

ID	Name
1	Akira Toriyama
2	J.R.R Tolkien
3	George R.R. Martin
4	Adam Douglas

Publisher

Name: Publisher Name

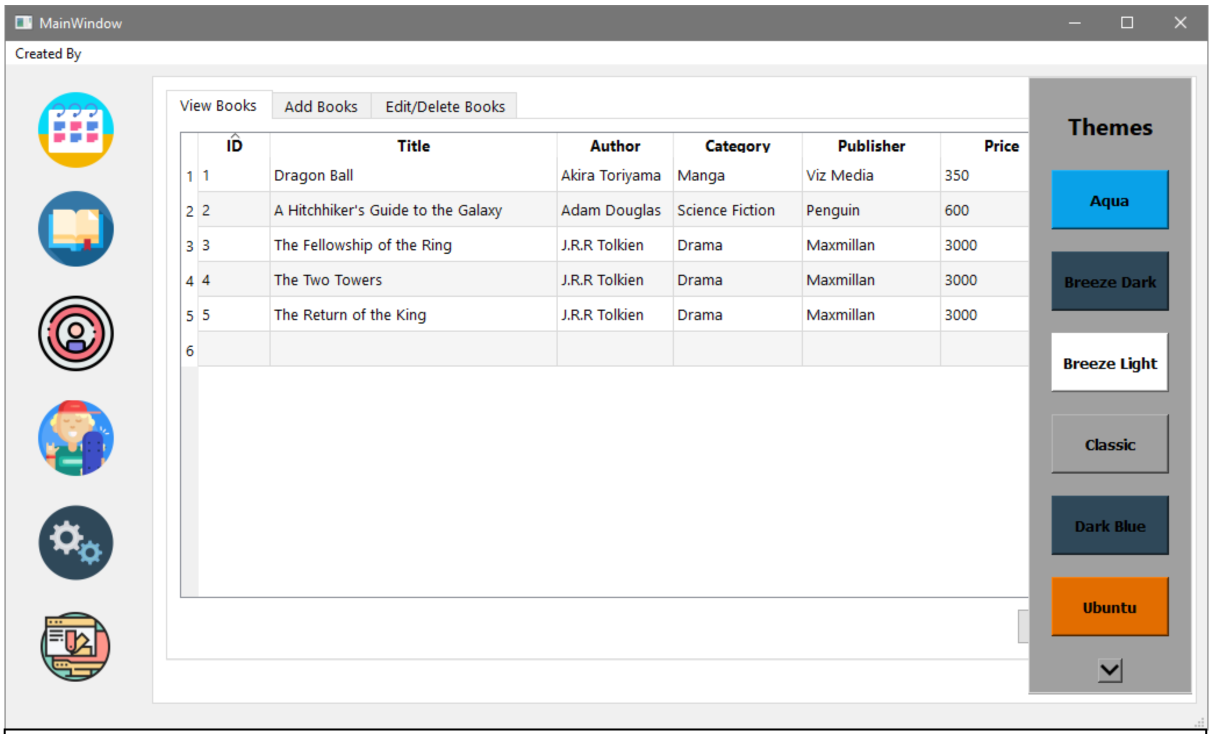
ID	Name
1	Viz Media
2	Wiley
3	Maxmillan
4	Penguin

Category

Name: Category Name

ID	Name
1	Manga
2	Drama
3	Science Fiction
4	Horror

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


MainWindow

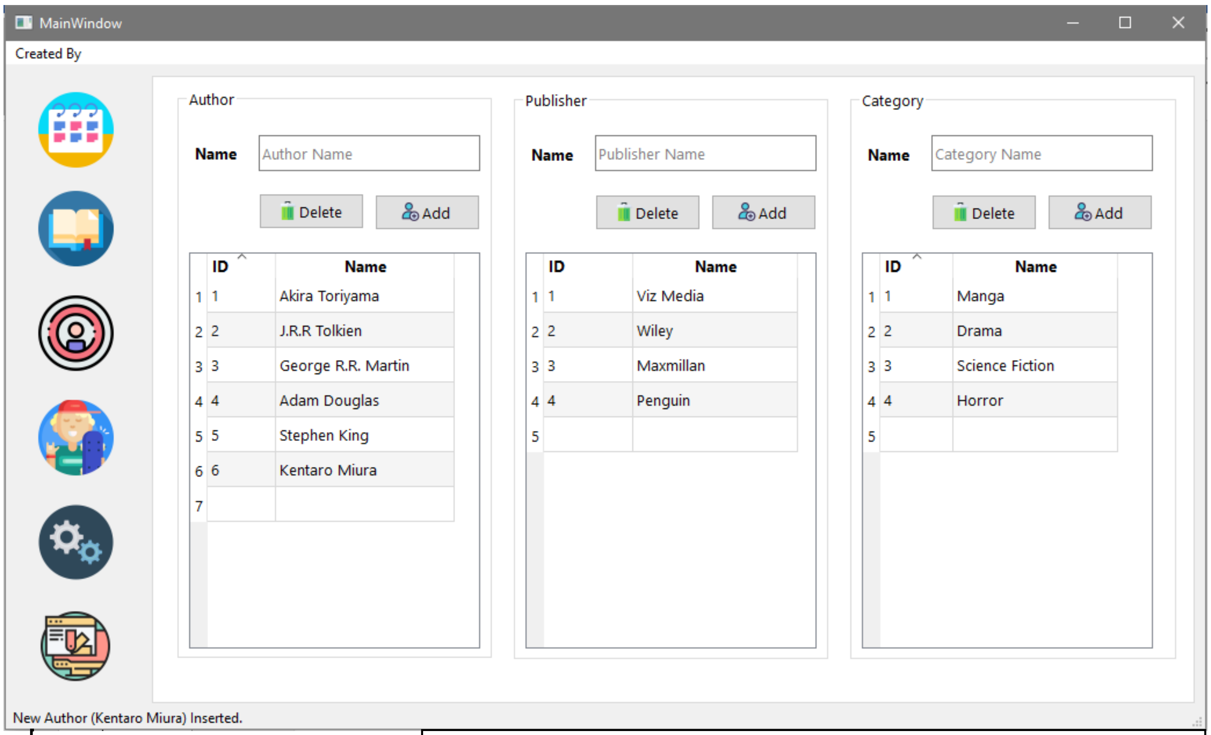
Created By

View Books | Add Books | Edit/Delete Books

ID	Title	Author	Category	Publisher	Price
1	Dragon Ball	Akira Toriyama	Manga	Viz Media	350
2	A Hitchhiker's Guide to the Galaxy	Adam Douglas	Science Fiction	Penguin	600
3	The Fellowship of the Ring	J.R.R Tolkien	Drama	Maxmillan	3000
4	The Two Towers	J.R.R Tolkien	Drama	Maxmillan	3000
5	The Return of the King	J.R.R Tolkien	Drama	Maxmillan	3000

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

Adding New Author:


MainWindow

Created By

Author

Name: Author Name

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6	6	Kentaro Miura
7		

Publisher

Name: Publisher Name

	ID	Name
1	1	Viz Media
2	2	Wiley
3	3	Maxmillan
4	4	Penguin
5		

Category

Name: Category Name

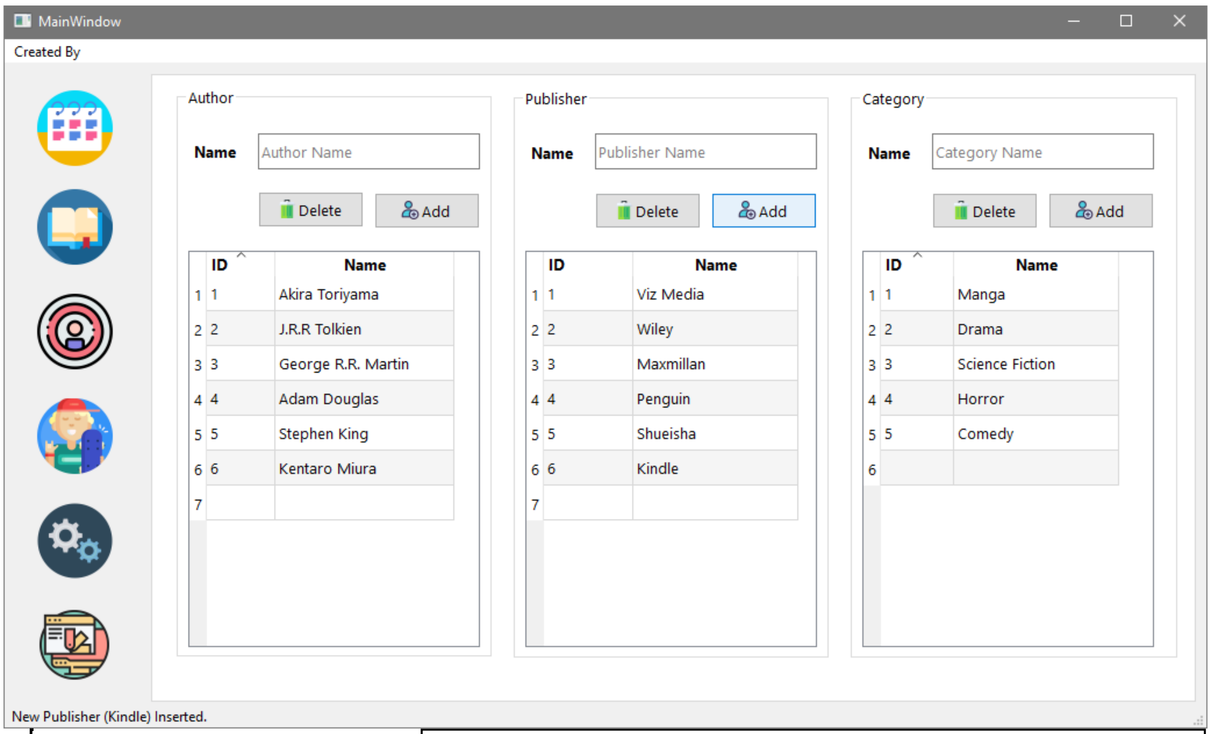
	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5		

New Author (Kentaro Miura) Inserted.

Confirmation on success

Image: *Add New Author in Database.*

Add New Publisher:


MainWindow

Created By

Author

Name: Author Name

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6	6	Kentaro Miura
7		

Publisher

Name: Publisher Name

	ID	Name
1	1	Viz Media
2	2	Wiley
3	3	Maxmillan
4	4	Penguin
5	5	Shueisha
6	6	Kindle
7		

Category

Name: Category Name

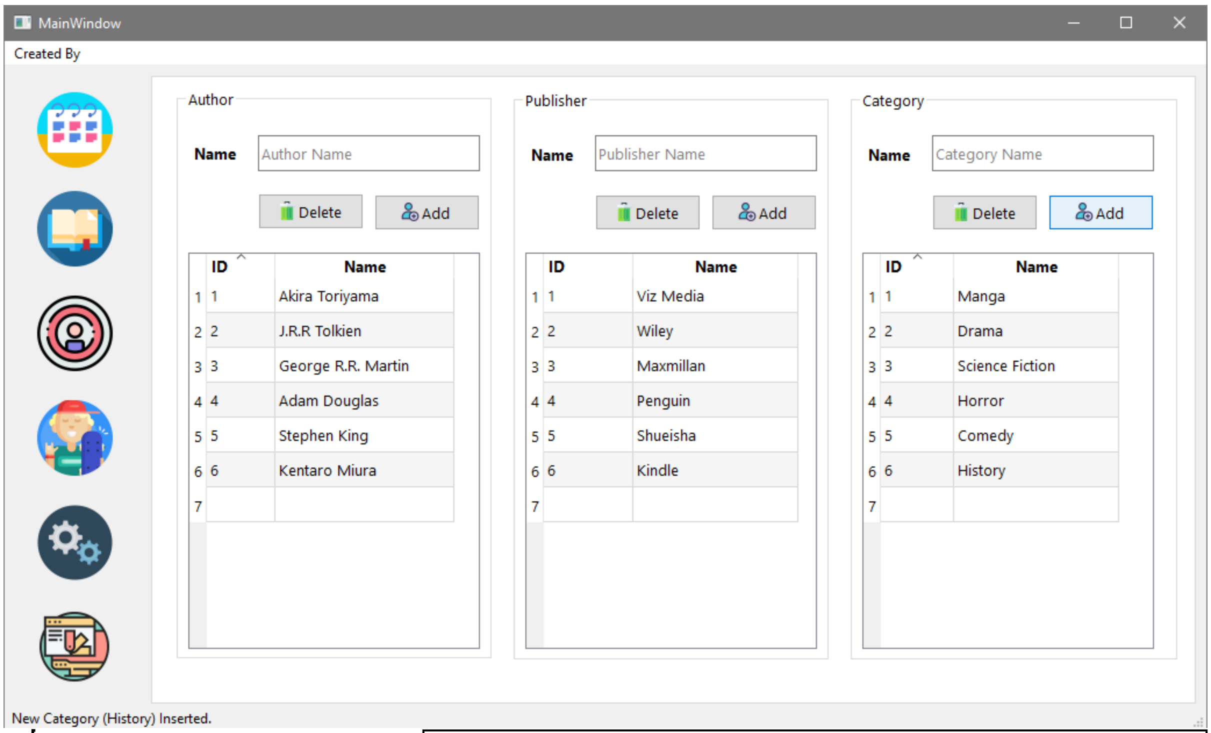
	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6		

New Publisher (Kindle) Inserted.

Confirmation on success

Image: *Add New Publisher in Database.*

Add New Category:



MainWindow

Created By

Author

Name: Author Name

Delete | Add

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6	6	Kentaro Miura
7		

Publisher

Name: Publisher Name

Delete | Add

	ID	Name
1	1	Viz Media
2	2	Wiley
3	3	Maxmillan
4	4	Penguin
5	5	Shueisha
6	6	Kindle
7		

Category

Name: Category Name

Delete | Add

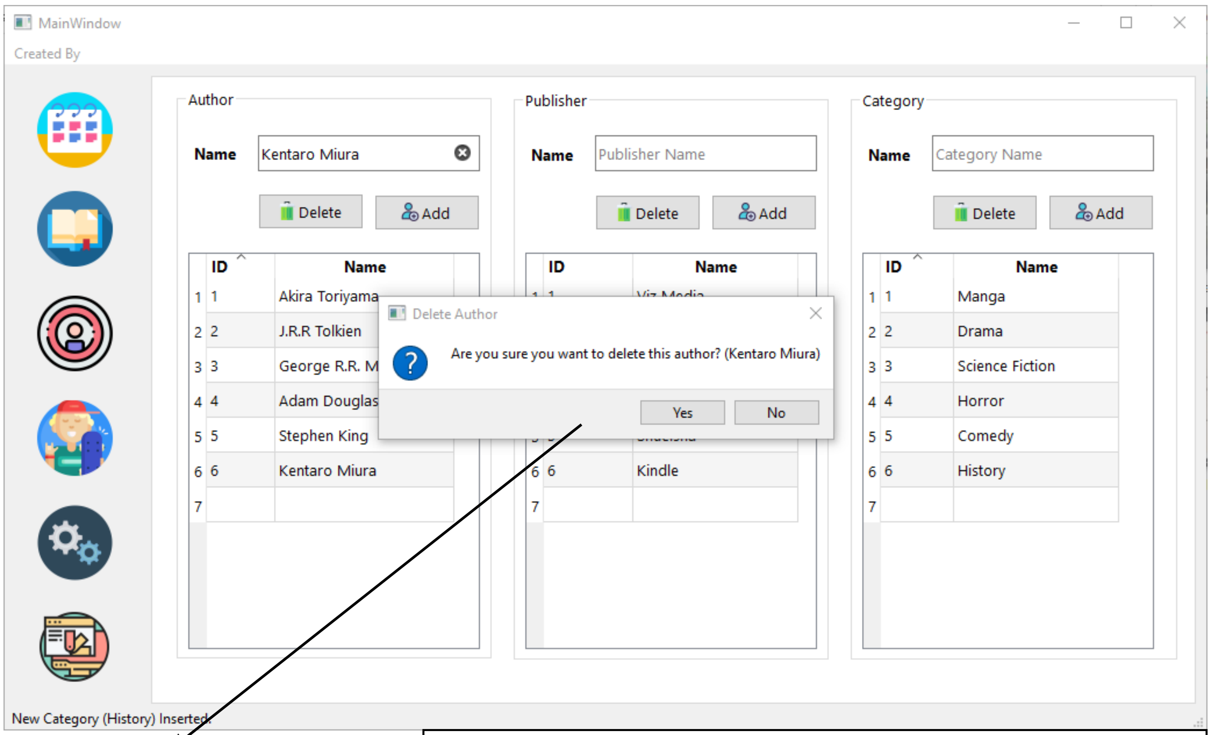
	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

New Category (History) Inserted.

Confirmation on success

Image: *Add New Category in Database.*

Delete Existing Author:



MainWindow

Created By

Author

Name: Kentaro Miura

Delete | Add

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. M
4	4	Adam Douglas
5	5	Stephen King
6	6	Kentaro Miura
7		

Publisher

Name: Publisher Name

Delete | Add

	ID	Name
6	6	Kindle
7		

Category

Name: Category Name

Delete | Add

	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

Delete Author

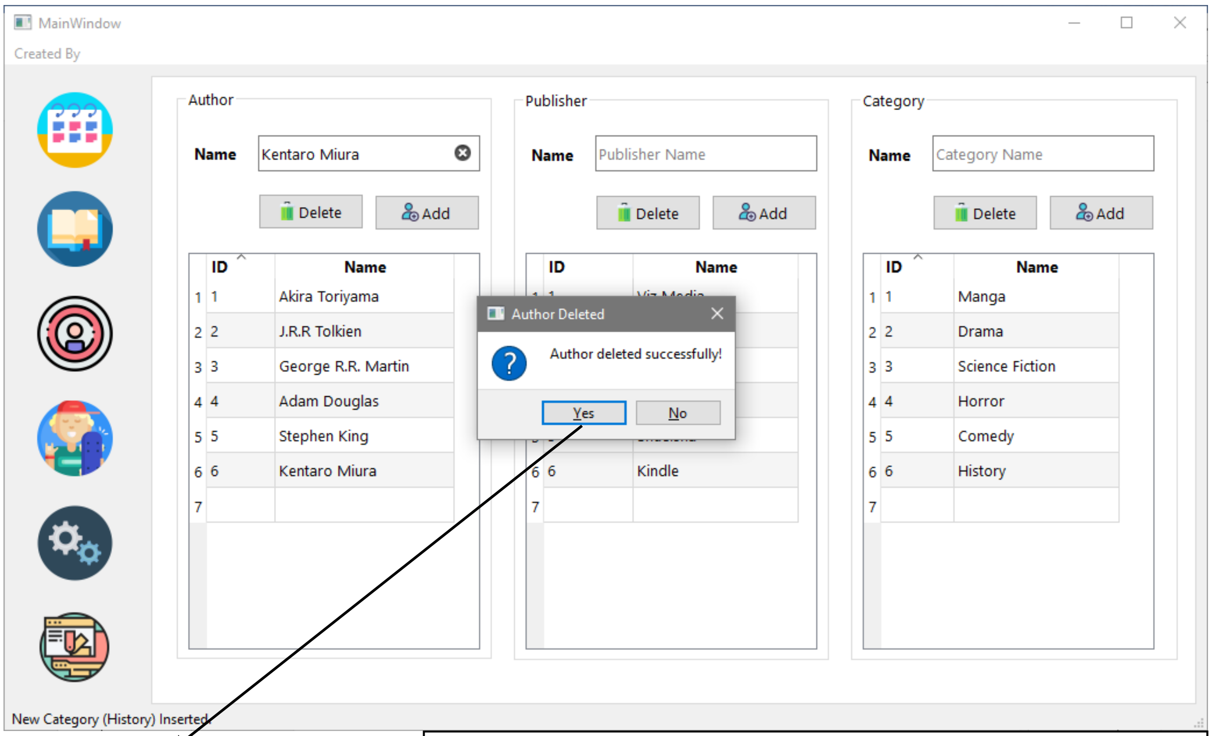
Are you sure you want to delete this author? (Kentaro Miura)

Yes | No

New Category (History) Inserted.

Asks confirmation from user

Image: *Delete Author from database.*



Kentaro Miura

Publisher Name

Category Name

Author deleted successfully!

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6	6	Kentaro Miura
7		

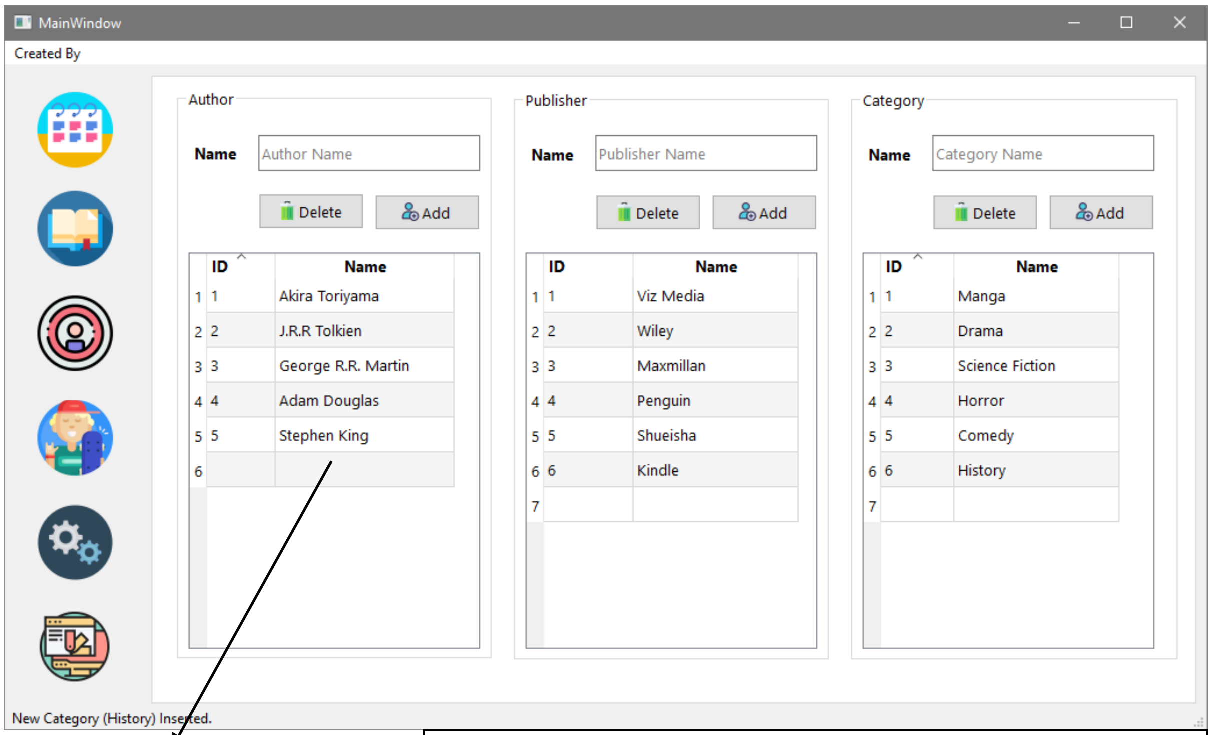
	ID	Name
6	6	Kindle
7		

	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

New Category (History) Inserted.

Successful deletion message

Image: *Deletion is successful.*



Author Name

Publisher Name

Category Name

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6		

	ID	Name
1	1	Viz Media
2	2	Wiley
3	3	Maxmillan
4	4	Penguin
5	5	Shueisha
6	6	Kindle
7		

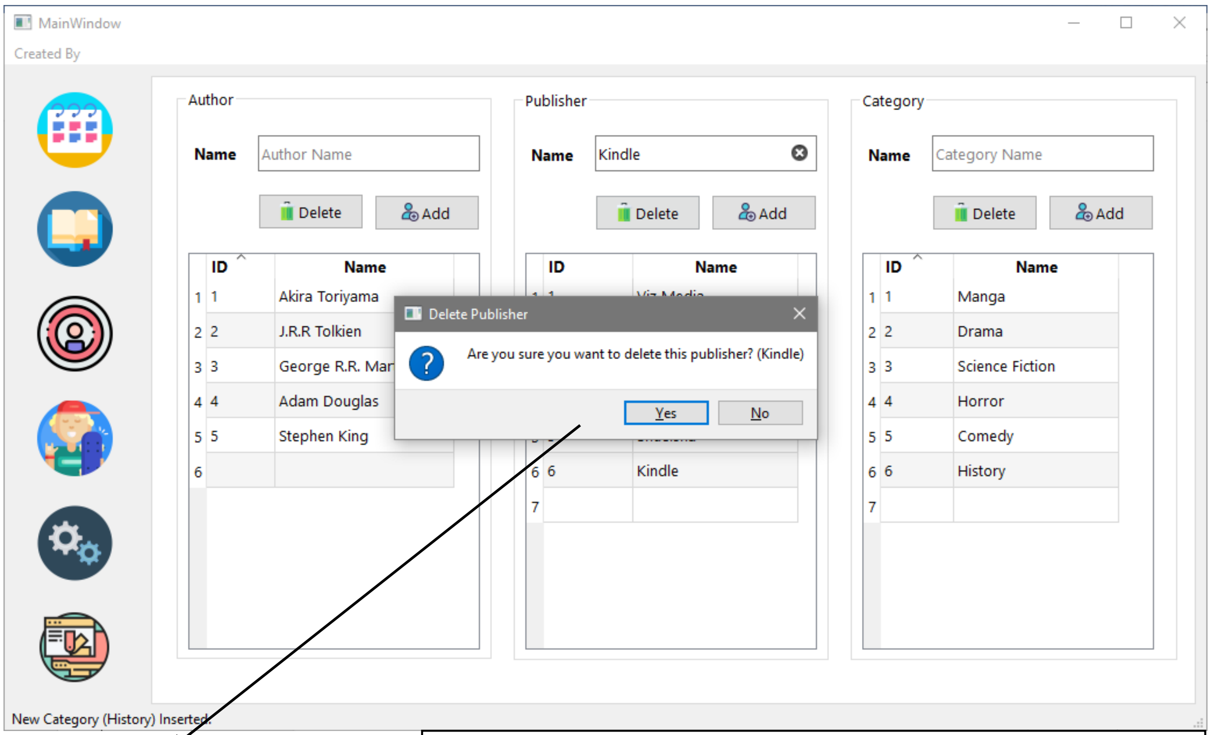
	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

New Category (History) Inserted.

Deleted Author cannot be seen

Image: *Author is deleted from the database.*

Delete Existing Publisher:



MainWindow

Created By

Author

Name: Author Name

Delete | Add

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Mar
4	4	Adam Douglas
5	5	Stephen King
6		

Publisher

Name: Kindle

Delete | Add

	ID	Name
1	1	Viz Media
6	6	Kindle
7		

Delete Publisher

Are you sure you want to delete this publisher? (Kindle)

Yes | No

Category

Name: Category Name

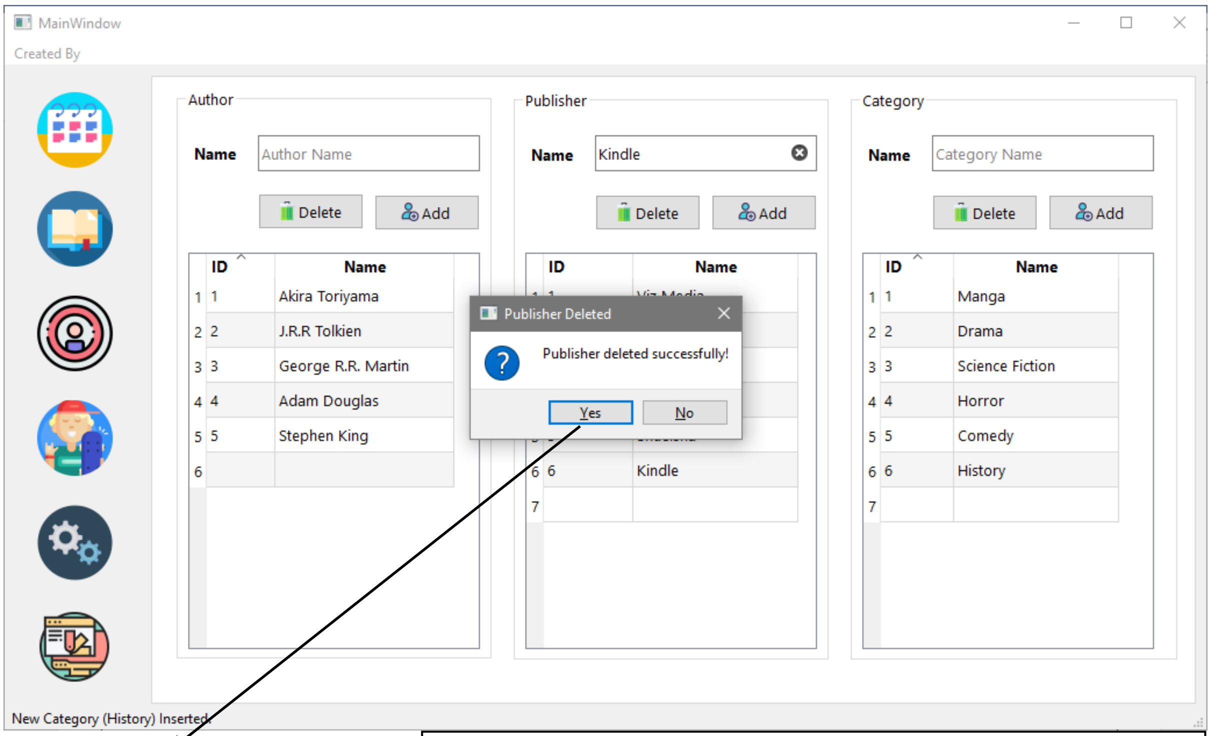
Delete | Add

	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

New Category (History) Inserted.

Asks confirmation from user

Image: *Delete Publisher from database.*



MainWindow

Created By

Author

Name: Author Name

Delete | Add

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6		

Publisher

Name: Kindle

Delete | Add

	ID	Name
1	1	Viz Media
6	6	Kindle
7		

Publisher Deleted

Publisher deleted successfully!

Yes | No

Category

Name: Category Name

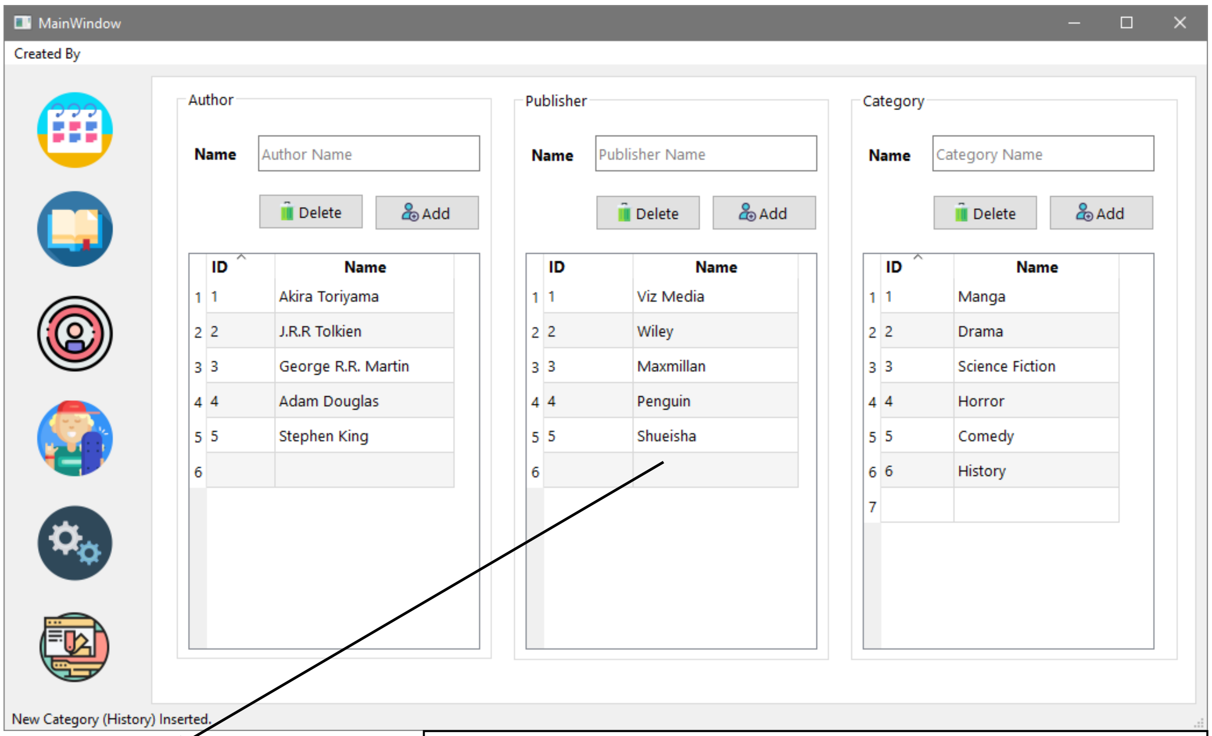
Delete | Add

	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

New Category (History) Inserted.

Successful deletion message

Image: *Deletion is successful.*



	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6		

	ID	Name
1	1	Viz Media
2	2	Wiley
3	3	Maxmillan
4	4	Penguin
5	5	Shueisha
6		

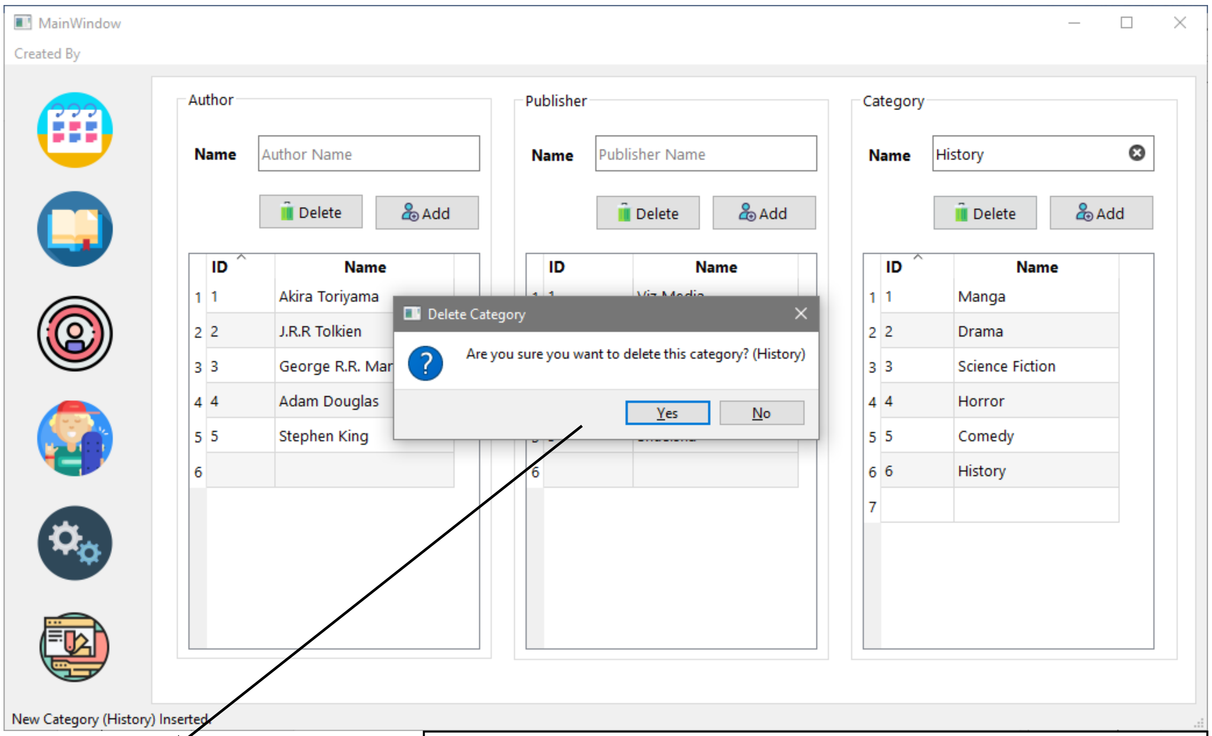
	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

New Category (History) Inserted.

Deleted Publisher cannot be seen

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

Delete Existing Category:



Are you sure you want to delete this category? (History)

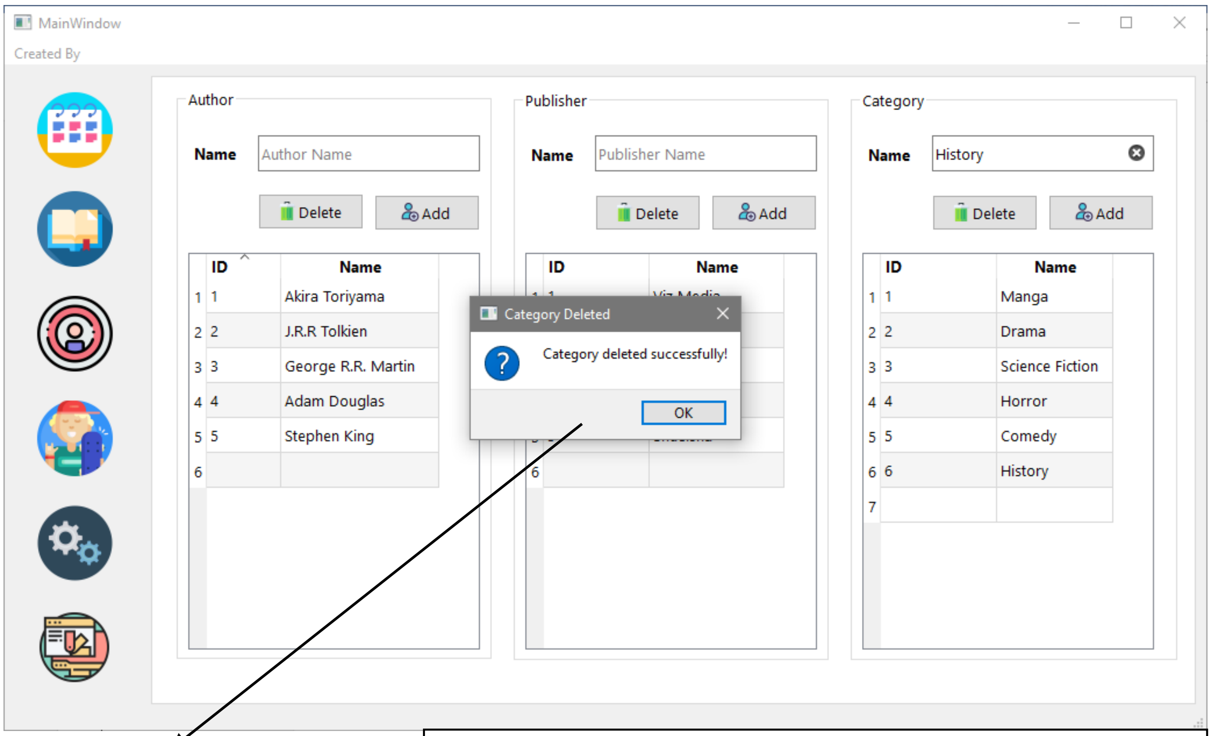
	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Mar
4	4	Adam Douglas
5	5	Stephen King
6		

	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

New Category (History) Inserted.

Asks confirmation from user

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


MainWindow

Created By

Author

Name: Author Name

Delete | Add

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6		

Publisher

Name: Publisher Name

Delete | Add

	ID	Name
6		

Category

Name: History

Delete | Add

	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6	6	History
7		

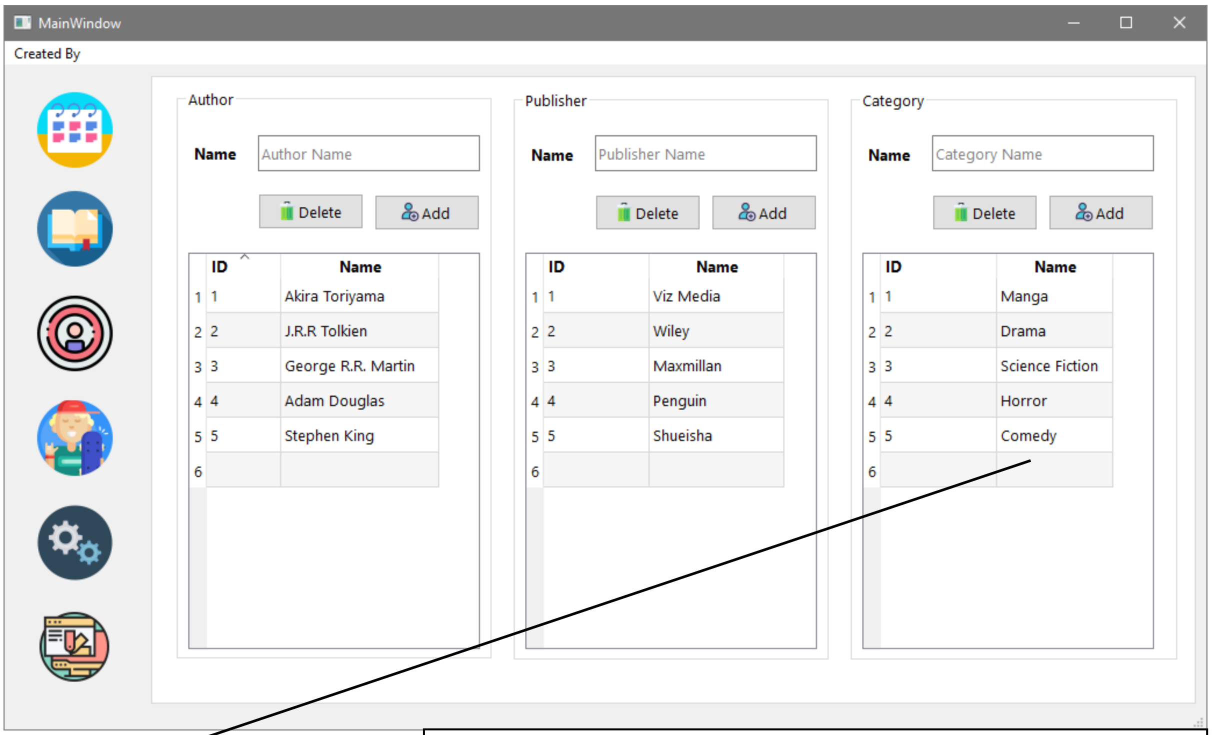
Category Deleted

Category deleted successfully!

OK

Successful deletion message

Image: *Delete Publisher from database.*


MainWindow

Created By

Author

Name: Author Name

Delete | Add

	ID	Name
1	1	Akira Toriyama
2	2	J.R.R Tolkien
3	3	George R.R. Martin
4	4	Adam Douglas
5	5	Stephen King
6		

Publisher

Name: Publisher Name

Delete | Add

	ID	Name
1	1	Viz Media
2	2	Wiley
3	3	Maxmillan
4	4	Penguin
5	5	Shueisha
6		

Category

Name: Category Name

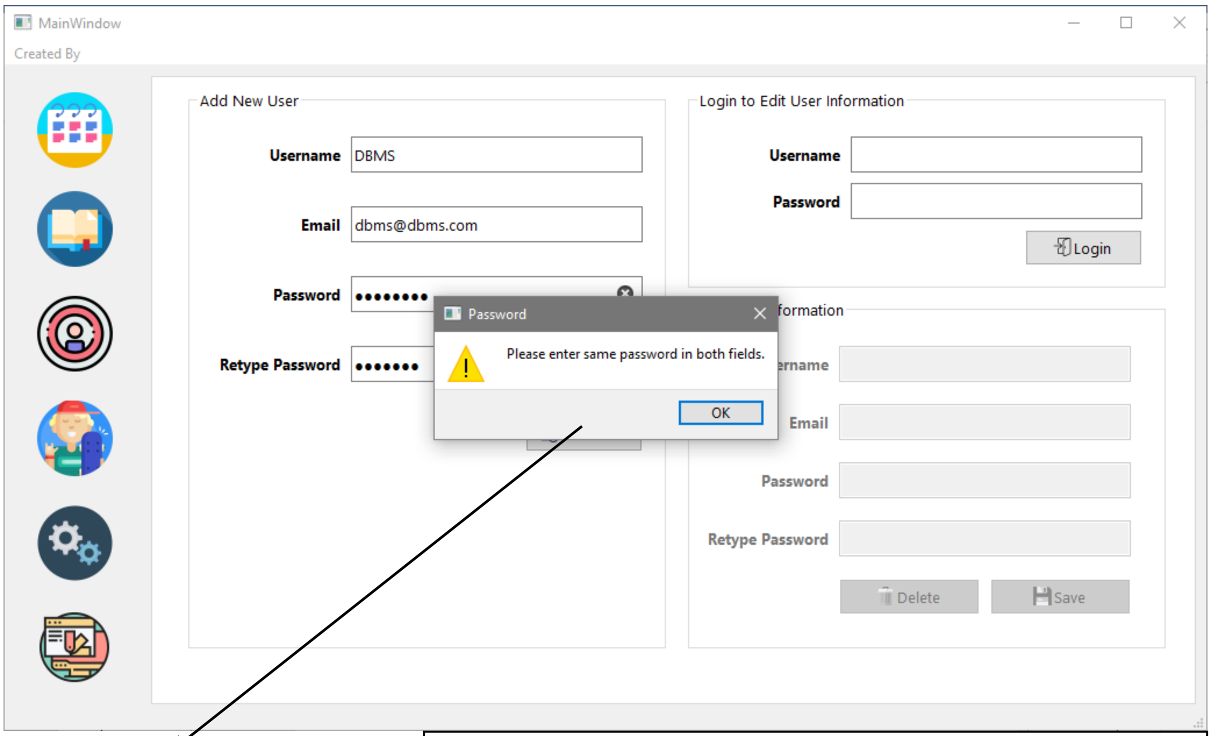
Delete | Add

	ID	Name
1	1	Manga
2	2	Drama
3	3	Science Fiction
4	4	Horror
5	5	Comedy
6		

Deleted Category cannot be seen

Image: *Category is deleted from the database.*

Creating New User:


MainWindow

Created By

Add New User

Username: DBMS

Email: dbms@dbms.com

Password: ••••••••

Retype Password: •••••••

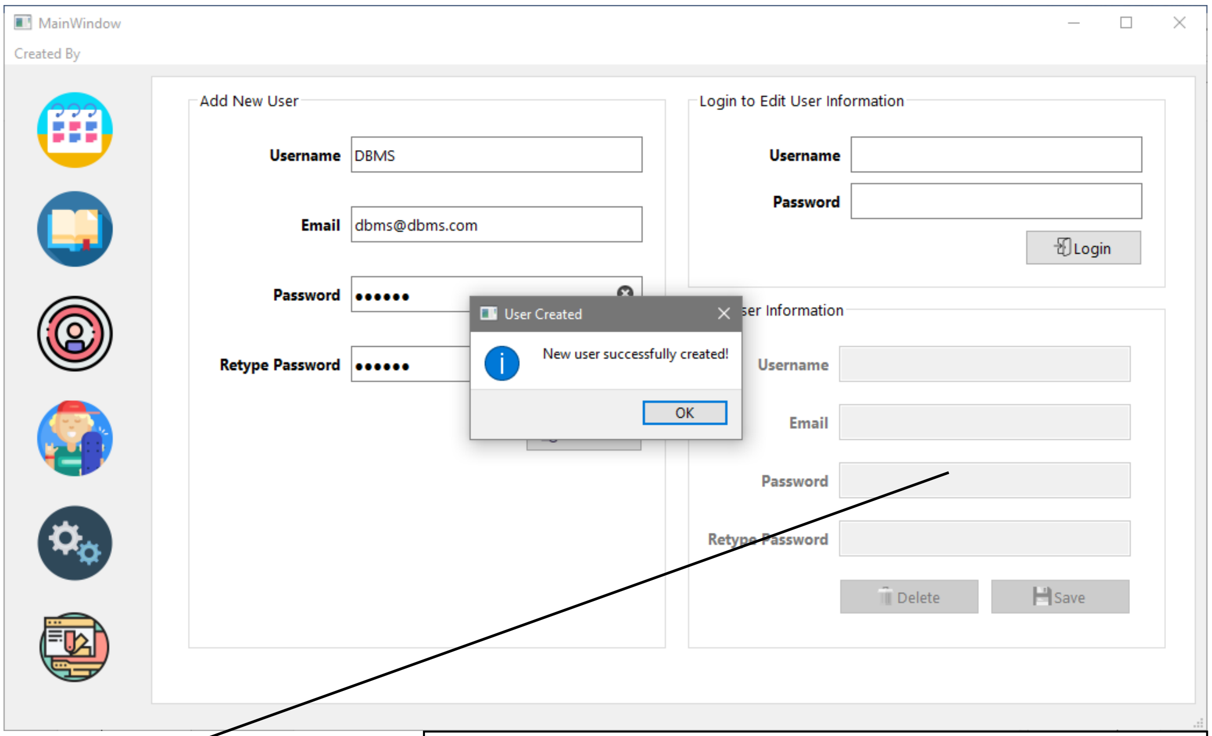
Password

Please enter same password in both fields.

OK

Error Message

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


MainWindow

Created By

Add New User

Username: DBMS

Email: dbms@dbms.com

Password: ••••••

Retype Password: ••••••

User Created

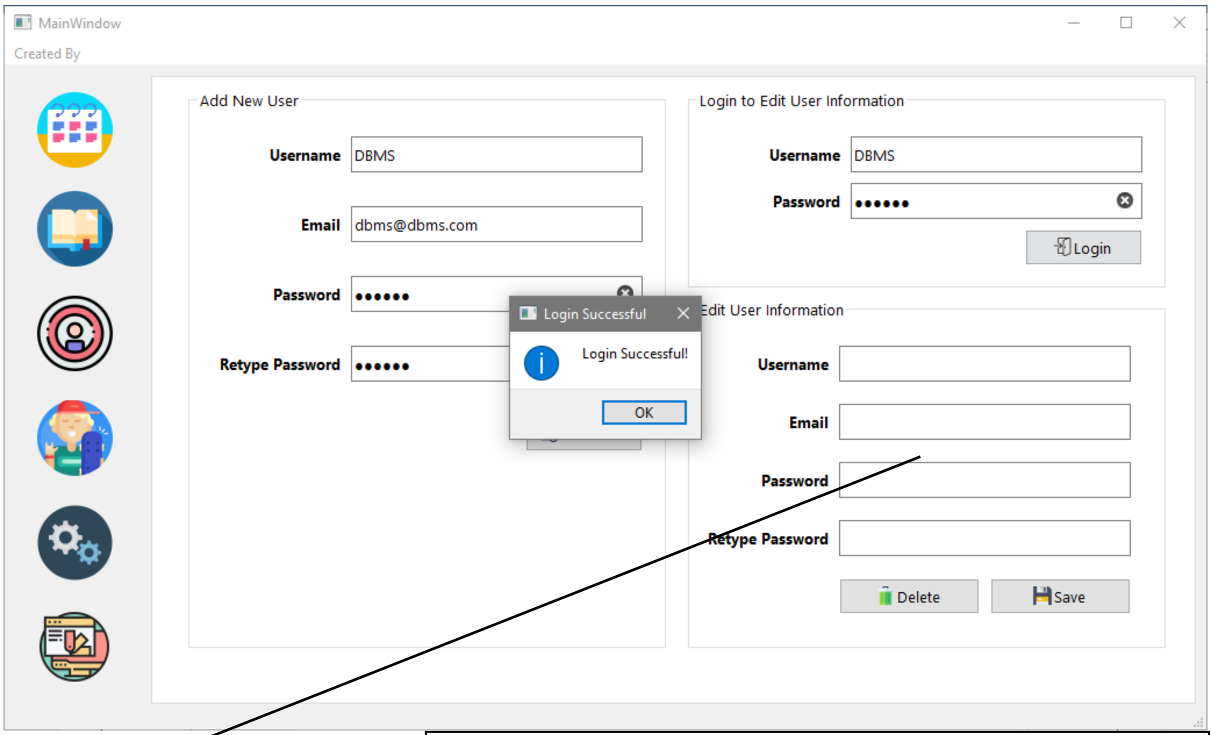
New user successfully created!

OK

Group box is initially disabled

Image: *New User has been created.*

Login to Edit Information:



MainWindow

Created By

Add New User

Username	DBMS
Email	dbms@dbms.com
Password	••••••
Retype Password	••••••

Login to Edit User Information

Username	DBMS
Password	••••••

Login Successful

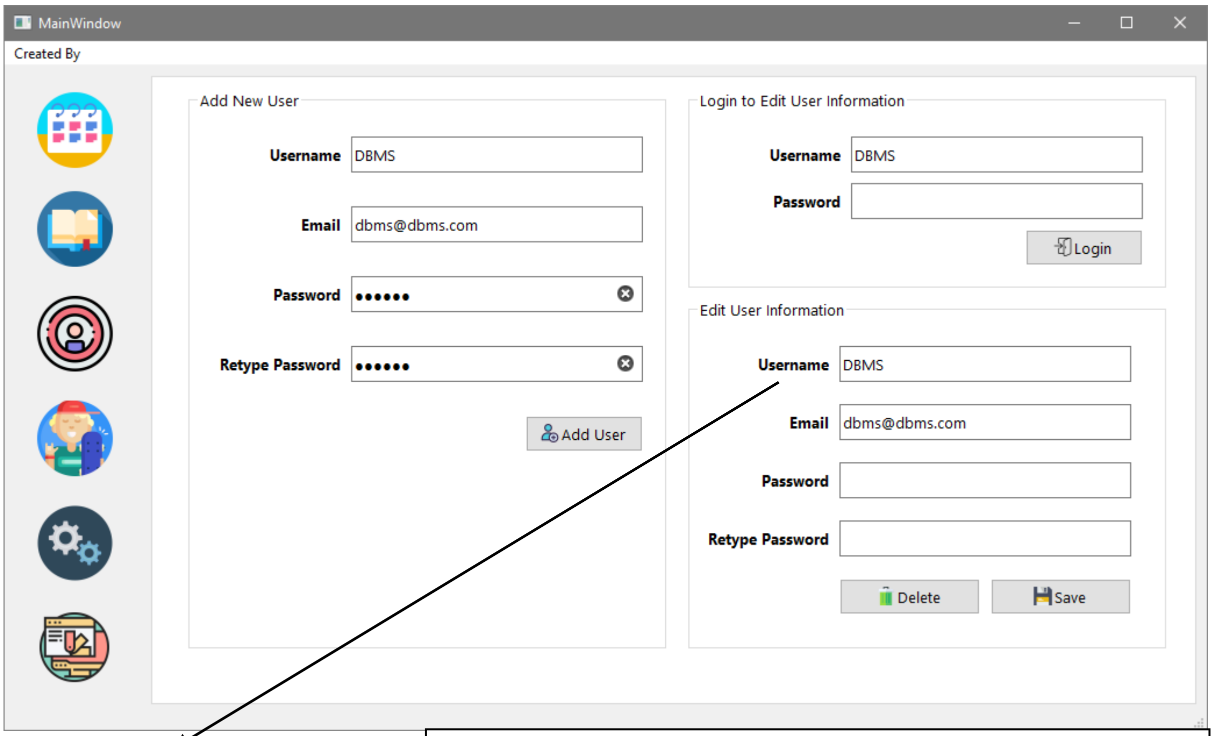
Login Successful!

Edit User Information

Username	
Email	
Password	
Retype Password	

Group box enables upon successful login

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



MainWindow

Created By

Add New User

Username	DBMS
Email	dbms@dbms.com
Password	••••••
Retype Password	••••••

Login to Edit User Information

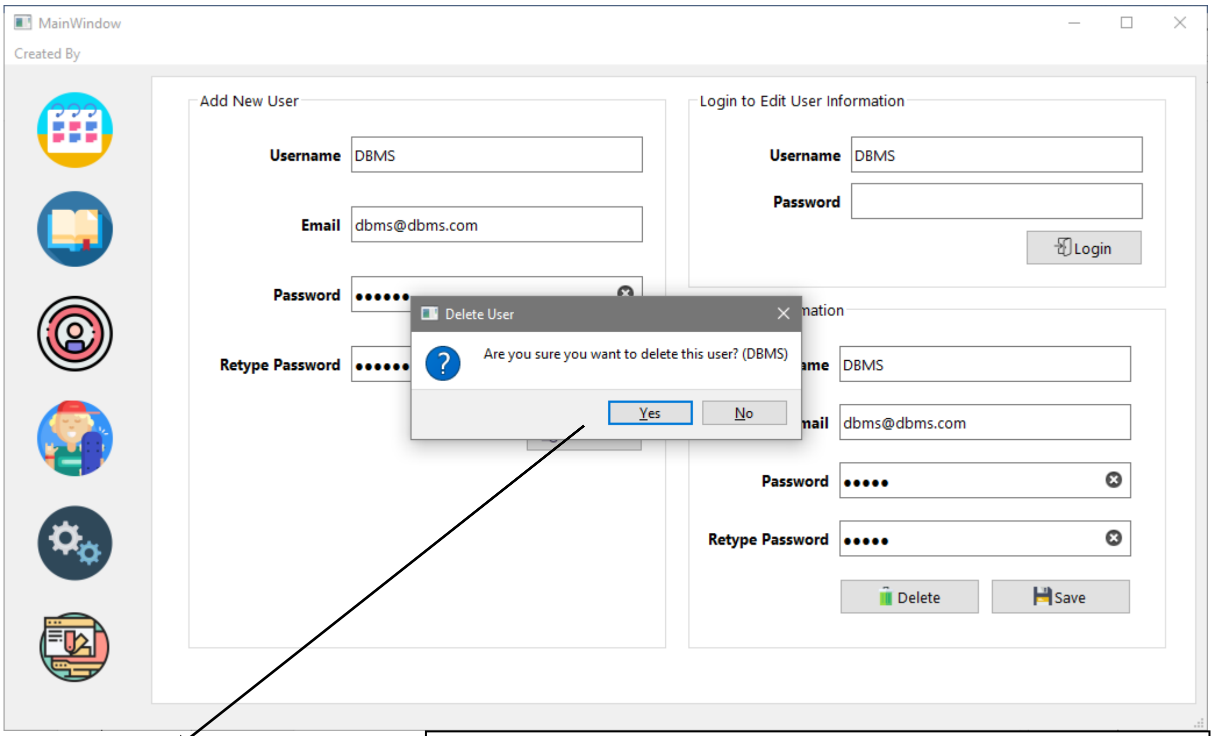
Username	DBMS
Password	

Edit User Information

Username	DBMS
Email	dbms@dbms.com
Password	
Retype Password	

Username cannot be edited

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



MainWindow

Created By

Add New User

Username: DBMS

Email: dbms@dbms.com

Password: ●●●●●●

Retype Password: ●●●●●●

Login to Edit User Information

Username: DBMS

Password:

Delete User

Are you sure you want to delete this user? (DBMS)

Username: DBMS

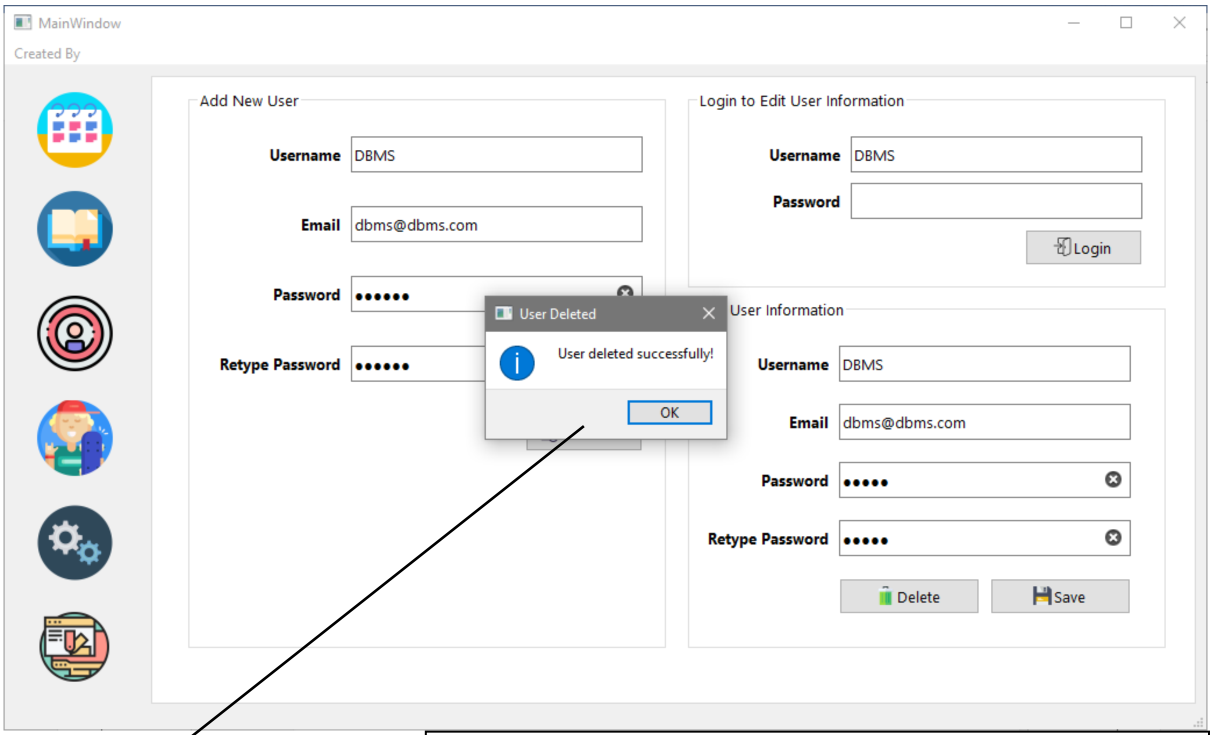
Email: dbms@dbms.com

Password: ●●●●●

Retype Password: ●●●●●

Asks confirmation from user

Image: *Delete User from database.*



MainWindow

Created By

Add New User

Username: DBMS

Email: dbms@dbms.com

Password: ●●●●●●

Retype Password: ●●●●●●

Login to Edit User Information

Username: DBMS

Password:

User Deleted

User deleted successfully!

User Information

Username: DBMS

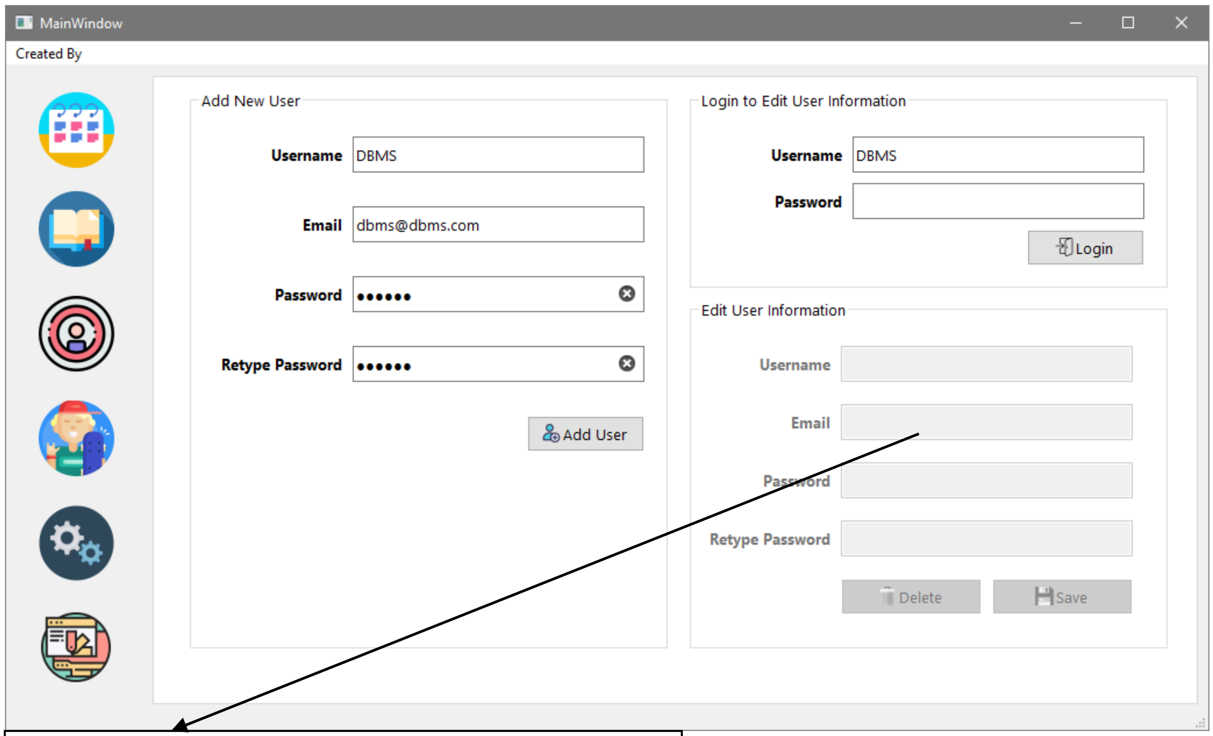
Email: dbms@dbms.com

Password: ●●●●●

Retype Password: ●●●●●

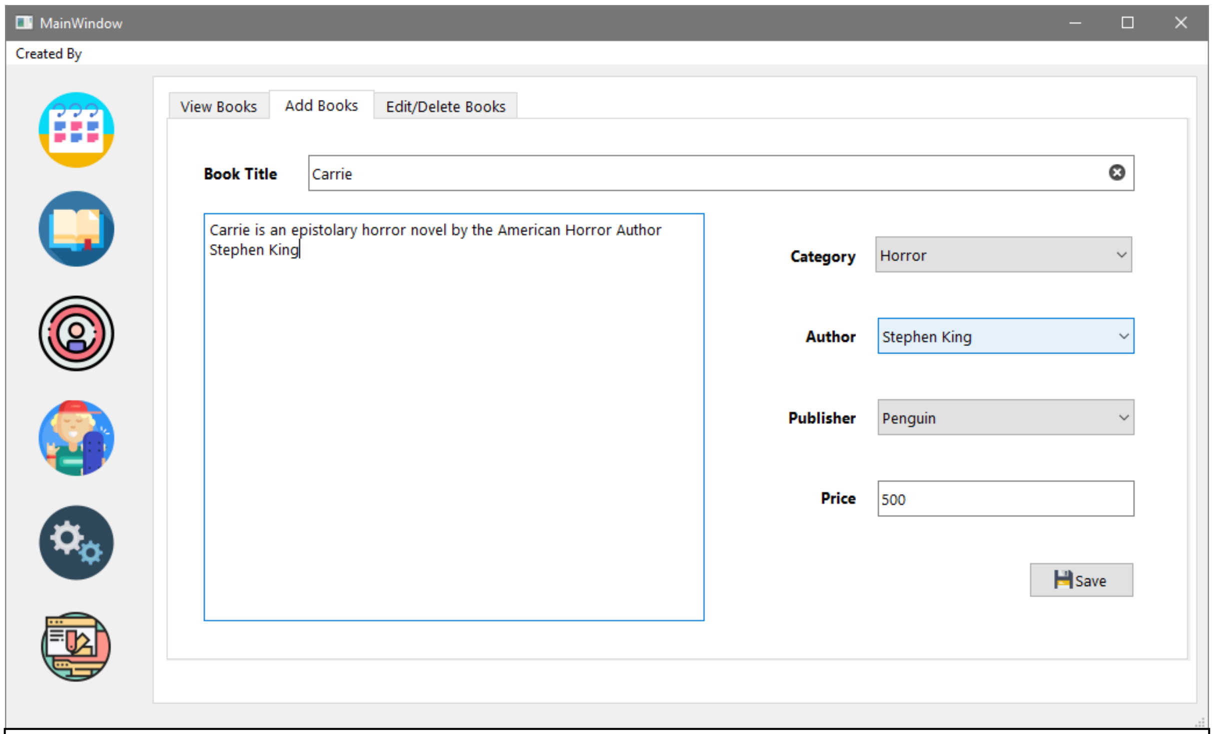
Successful deletion message

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



Group Box disables upon successful Edit/Delete operation

Add New Book:



Book Title: Carrie

Carrie is an epistolary horror novel by the American Horror Author Stephen King

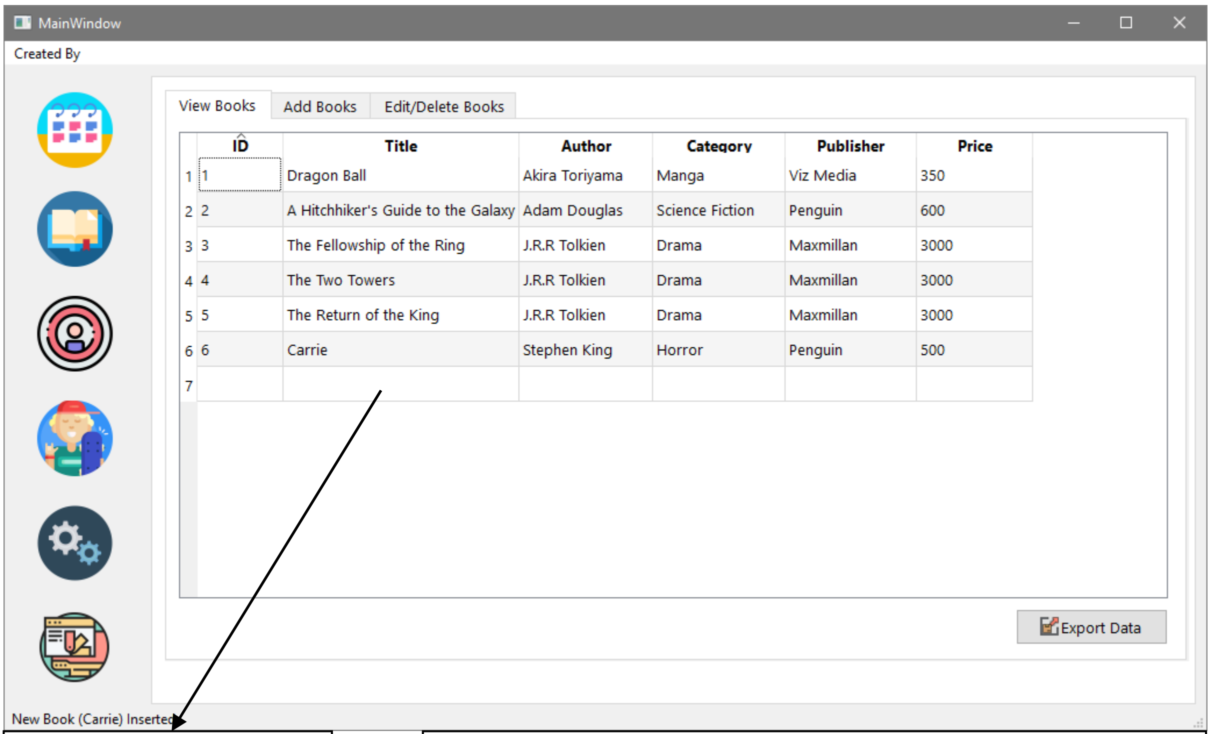
Category: Horror

Author: Stephen King

Publisher: Penguin

Price: 500

Image: *Adding new book in the library.*



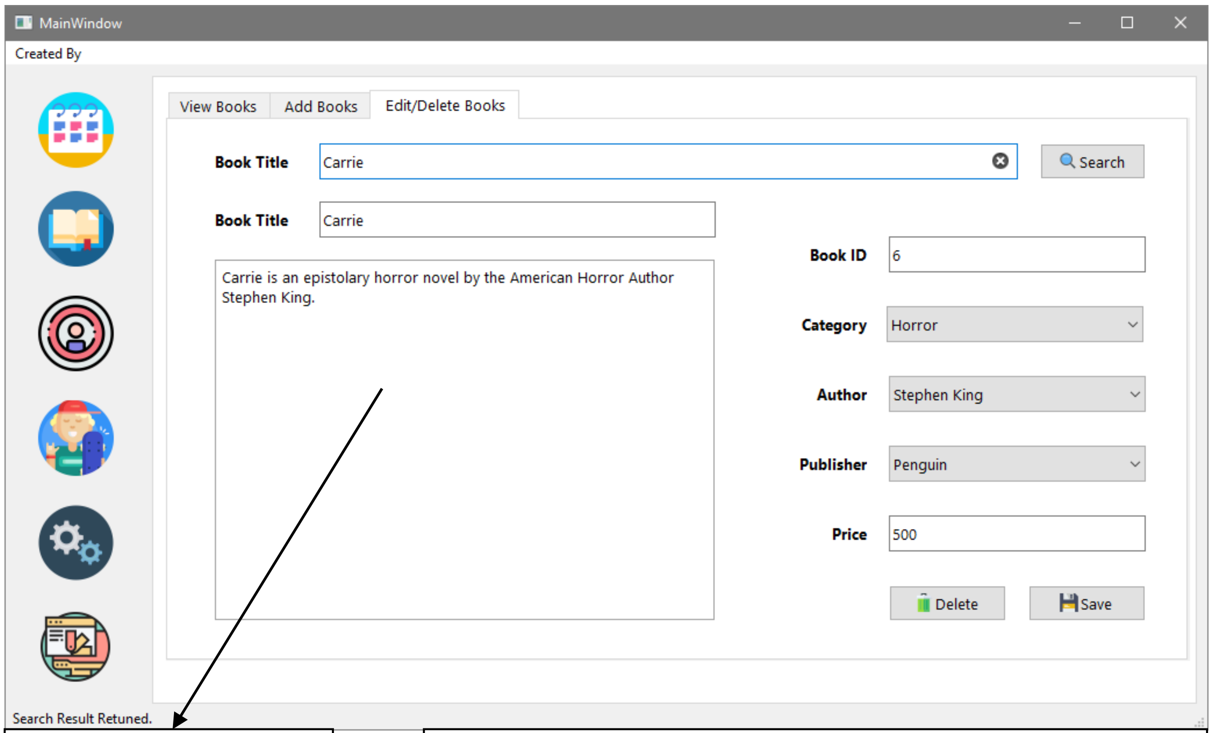
	ID	Title	Author	Category	Publisher	Price
1	1	Dragon Ball	Akira Toriyama	Manga	Viz Media	350
2	2	A Hitchhiker's Guide to the Galaxy	Adam Douglas	Science Fiction	Penguin	600
3	3	The Fellowship of the Ring	J.R.R Tolkien	Drama	Maxmillan	3000
4	4	The Two Towers	J.R.R Tolkien	Drama	Maxmillan	3000
5	5	The Return of the King	J.R.R Tolkien	Drama	Maxmillan	3000
6	6	Carrie	Stephen King	Horror	Penguin	500
7						

New Book (Carrie) Inserted

Book can be seen in the list

Image: *Book successfully added.*

Searching Books:



Book Title: Carrie

Book Title: Carrie

Carrie is an epistolary horror novel by the American Horror Author Stephen King.

Book ID: 6

Category: Horror

Author: Stephen King

Publisher: Penguin

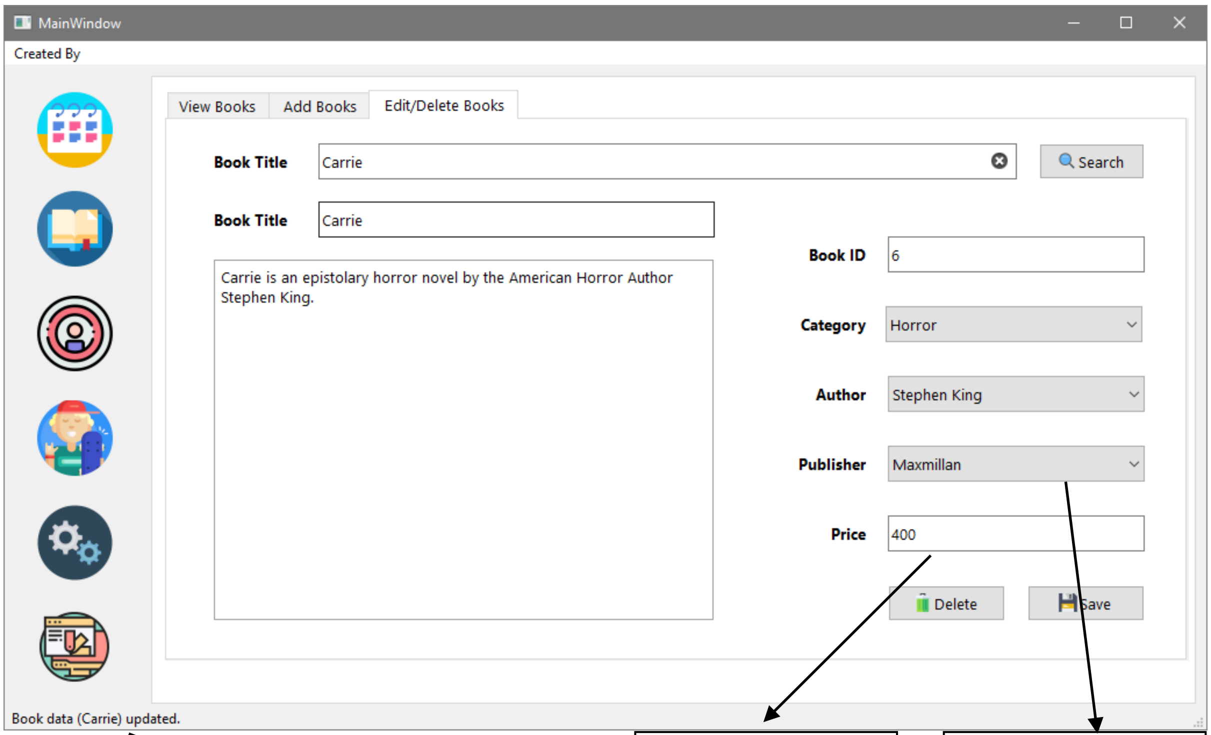
Price: 500

Search Result Retuned.

Book data returned

Image: *Book data successfully fetched.*

Editing Book Data:


MainWindow

Created By

View Books | Add Books | Edit/Delete Books

Book Title Carrie

Book Title Carrie

Carrie is an epistolary horror novel by the American Horror Author Stephen King.

Book ID 6

Category Horror

Author Stephen King

Publisher Maxmillan

Price 400

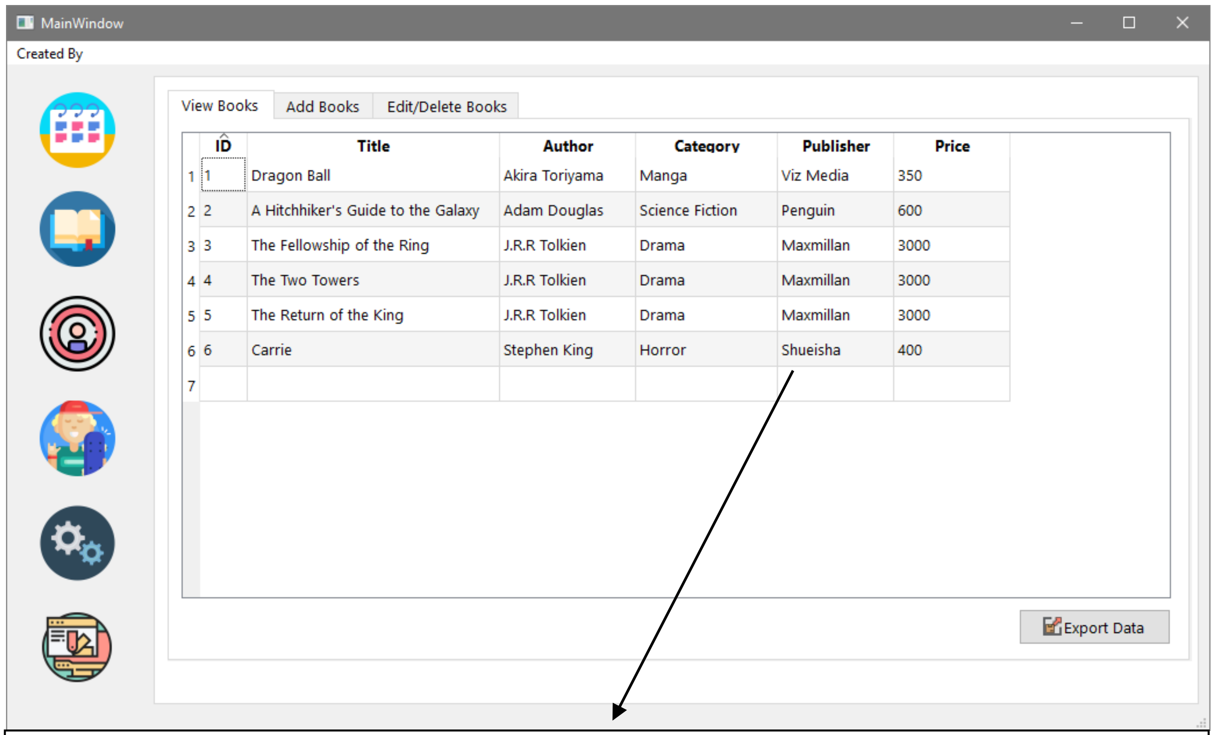
Delete | Save

Book data (Carrie) updated.

Changing the price

Changing the publisher

Updation Success


MainWindow

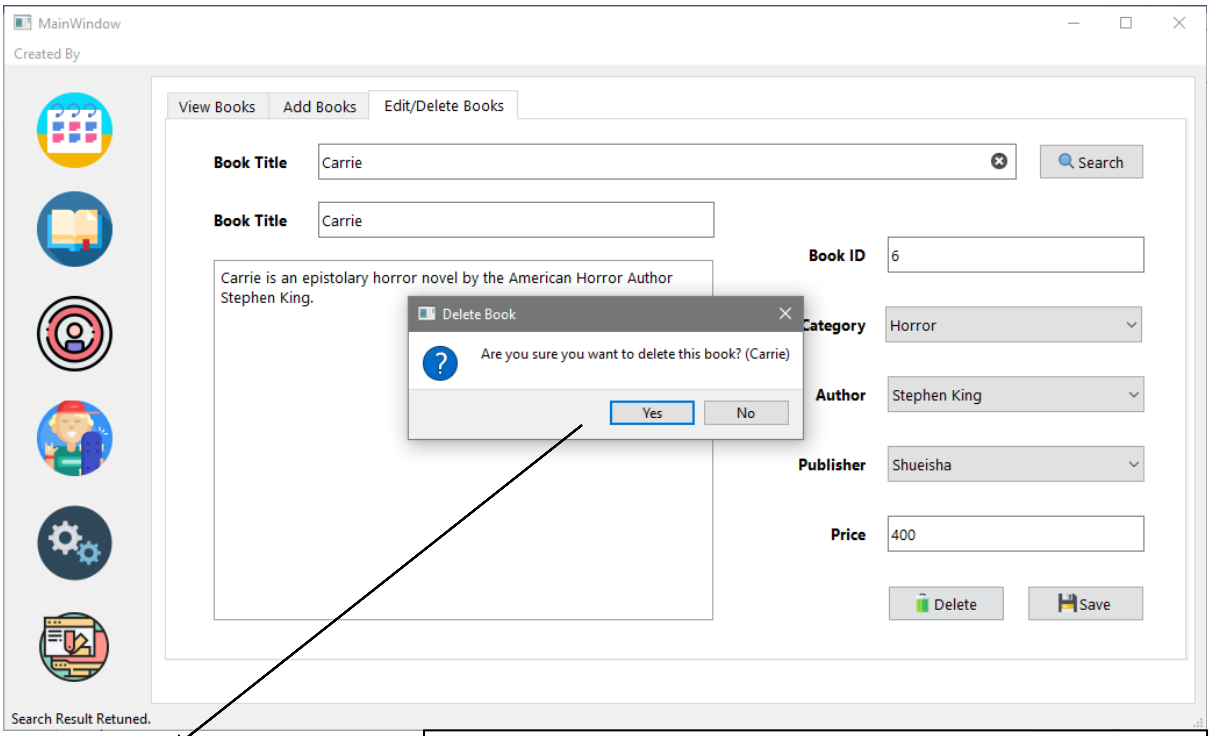
Created By

View Books | Add Books | Edit/Delete Books

	ID	Title	Author	Category	Publisher	Price
1	1	Dragon Ball	Akira Toriyama	Manga	Viz Media	350
2	2	A Hitchhiker's Guide to the Galaxy	Adam Douglas	Science Fiction	Penguin	600
3	3	The Fellowship of the Ring	J.R.R Tolkien	Drama	Maxmillan	3000
4	4	The Two Towers	J.R.R Tolkien	Drama	Maxmillan	3000
5	5	The Return of the King	J.R.R Tolkien	Drama	Maxmillan	3000
6	6	Carrie	Stephen King	Horror	Shueisha	400
7						

Export Data

Updated information can be seen in View Books tab



Book Title: Carrie

Book Title: Carrie

Carrie is an epistolary horror novel by the American Horror Author Stephen King.

Book ID: 6

Category: Horror

Author: Stephen King

Publisher: Shueisha

Price: 400

Delete Book

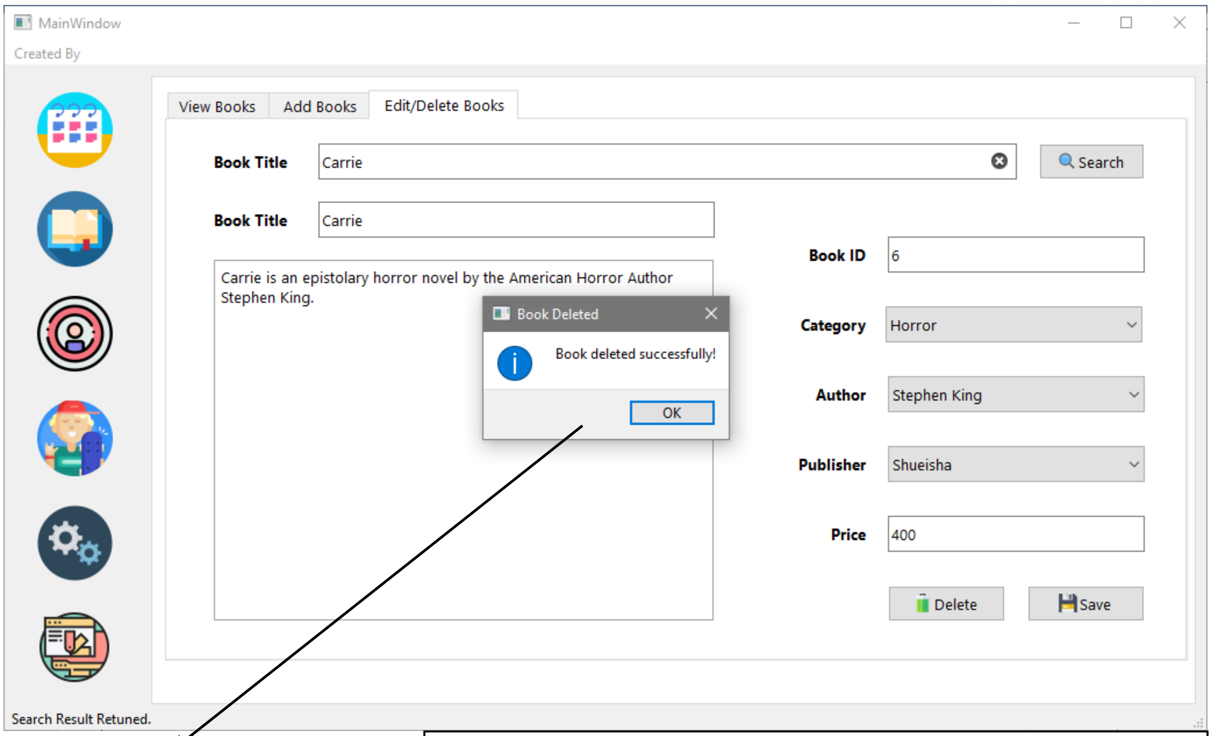
Are you sure you want to delete this book? (Carrie)

Search Result Retuned.

Asks confirmation from user

Image: *Delete Book from database.*

Deleting a Book:



Book Title: Carrie

Book Title: Carrie

Carrie is an epistolary horror novel by the American Horror Author Stephen King.

Book ID: 6

Category: Horror

Author: Stephen King

Publisher: Shueisha

Price: 400

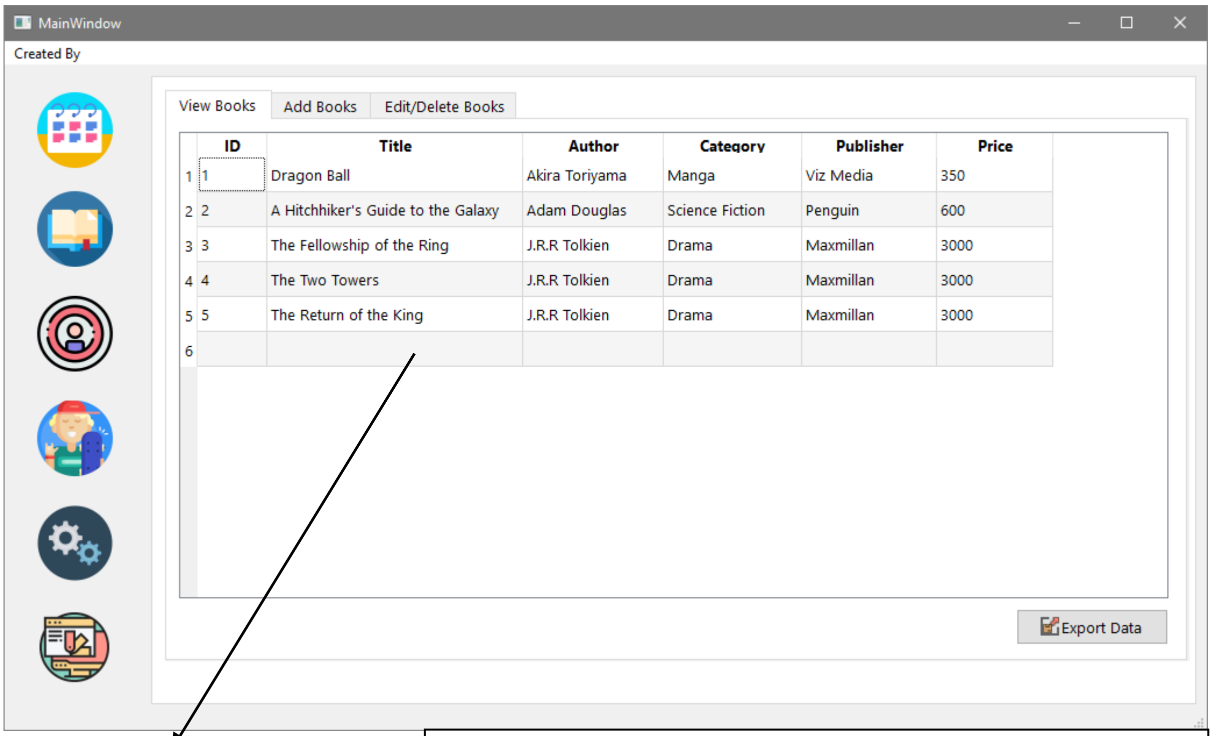
Book Deleted

Book deleted successfully!

Search Result Retuned.

Successful deletion message

Image: *Delete User from database.*



MainWindow

Created By

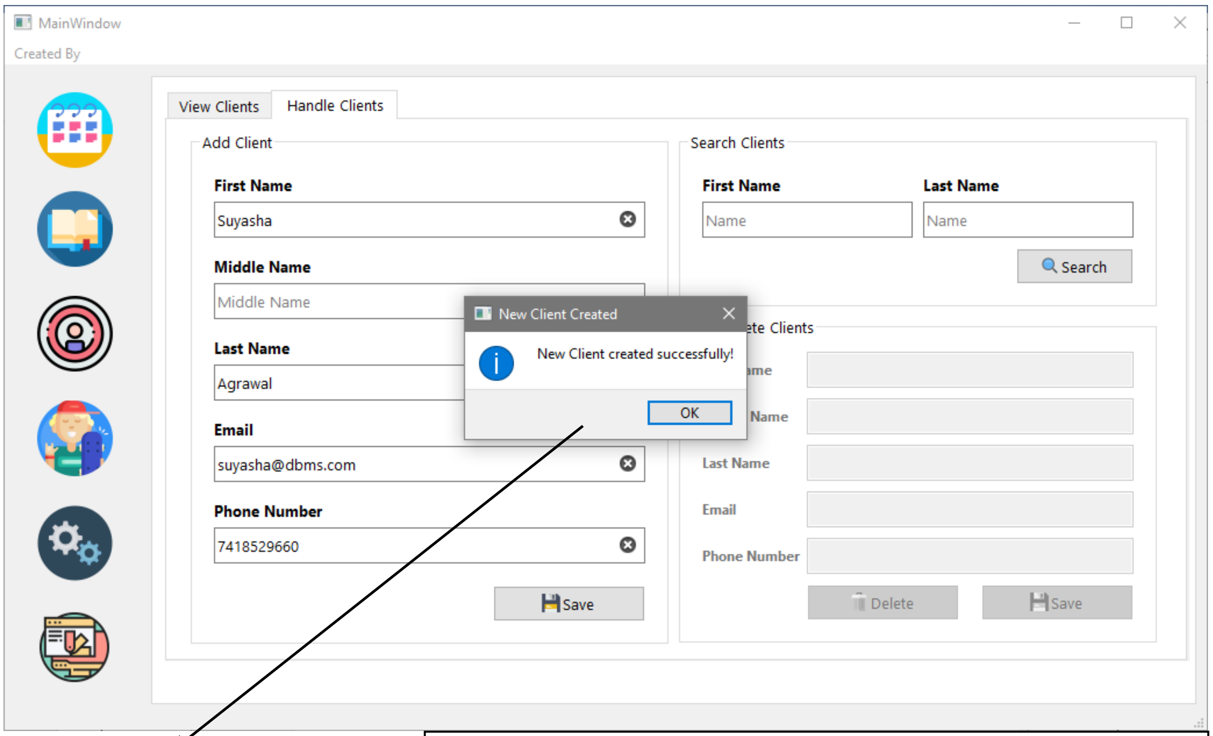
View Books | Add Books | Edit/Delete Books

	ID	Title	Author	Category	Publisher	Price
1	1	Dragon Ball	Akira Toriyama	Manga	Viz Media	350
2	2	A Hitchhiker's Guide to the Galaxy	Adam Douglas	Science Fiction	Penguin	600
3	3	The Fellowship of the Ring	J.R.R Tolkien	Drama	Maxmillan	3000
4	4	The Two Towers	J.R.R Tolkien	Drama	Maxmillan	3000
5	5	The Return of the King	J.R.R Tolkien	Drama	Maxmillan	3000
6						

Deleted book cannot be viewed

Image: *View books in the database.*

Adding a New Client:



MainWindow

Created By

View Clients | Handle Clients

Add Client

First Name: Suyasha

Middle Name: Middle Name

Last Name: Agrawal

Email: suyasha@dbms.com

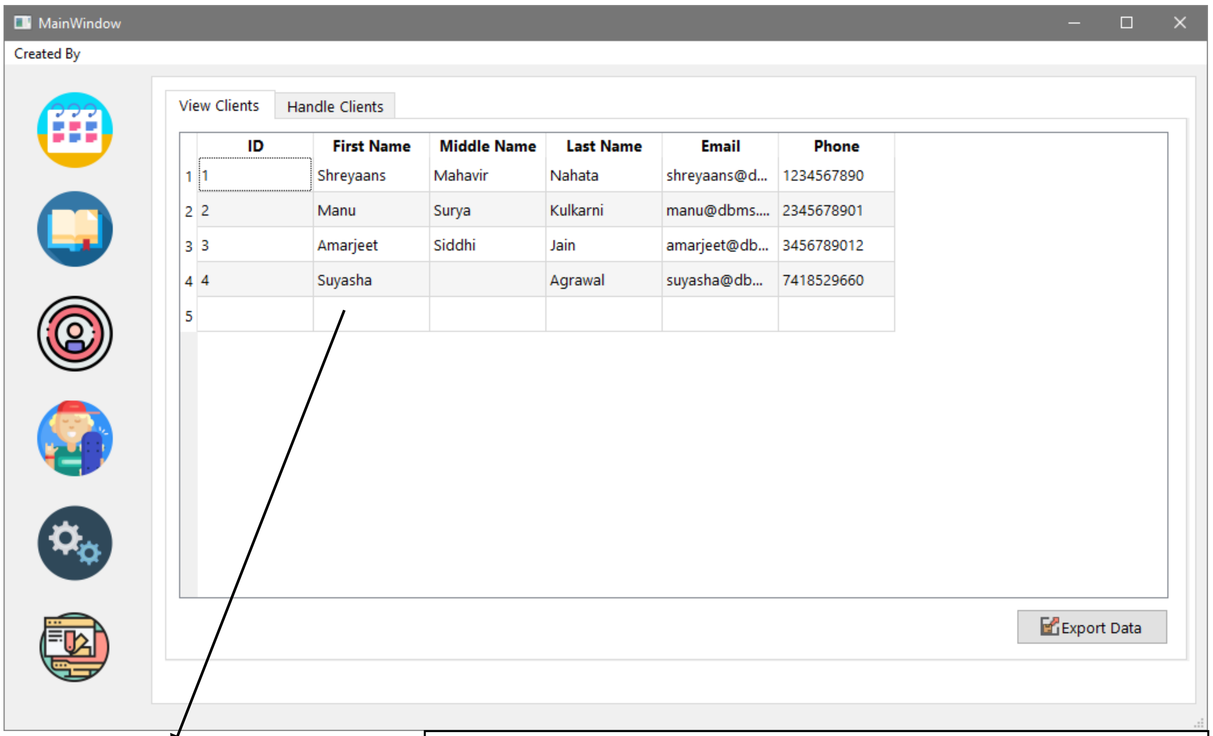
Phone Number: 7418529660

New Client Created

New Client created successfully!

New Client successfully created

Image: *Add new client to database.*

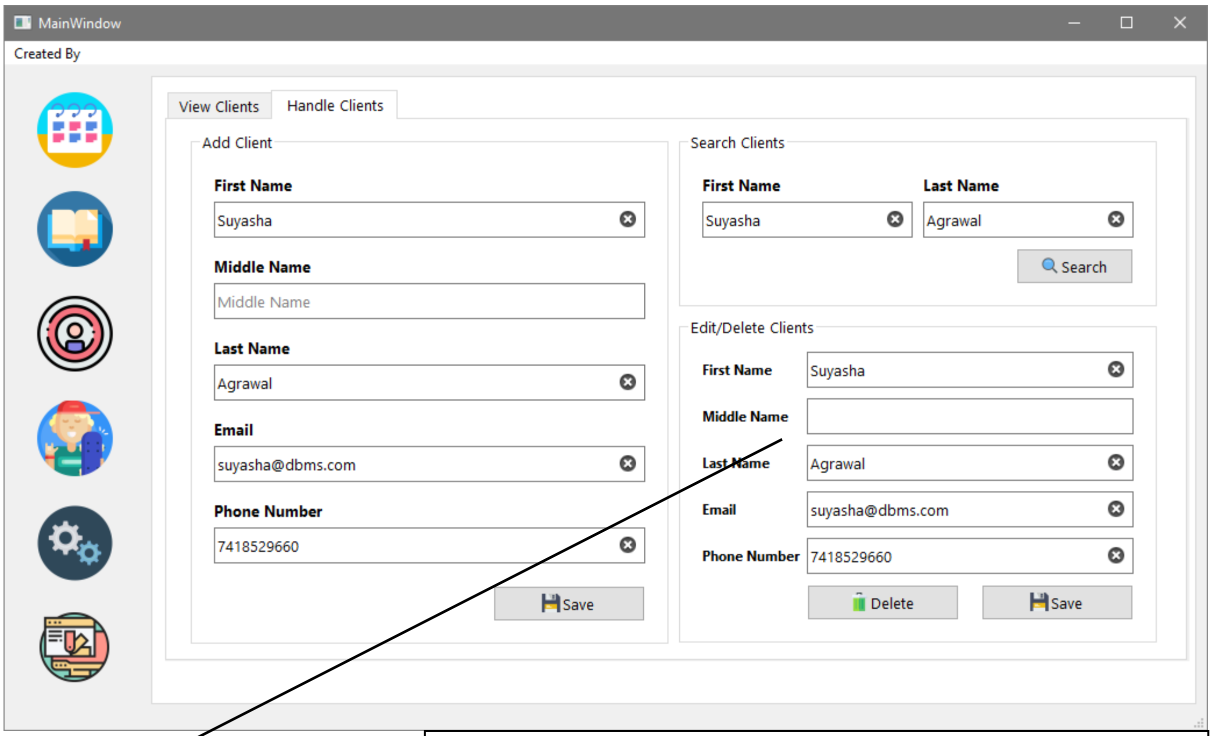


	ID	First Name	Middle Name	Last Name	Email	Phone
1	1	Shreyaans	Mahavir	Nahata	shreyaans@d...	1234567890
2	2	Manu	Surya	Kulkarni	manu@dbms....	2345678901
3	3	Amarjeet	Siddhi	Jain	amarjeet@db...	3456789012
4	4	Suyasha		Agrawal	suyasha@db...	7418529660
5						

New Client can be seen

Image: *View Clients in the database.*

Searching a Client:



Add Client

First Name	Suyasha
Middle Name	Middle Name
Last Name	Agrawal
Email	suyasha@dbms.com
Phone Number	7418529660

Search Clients

First Name	Last Name
Suyasha	Agrawal

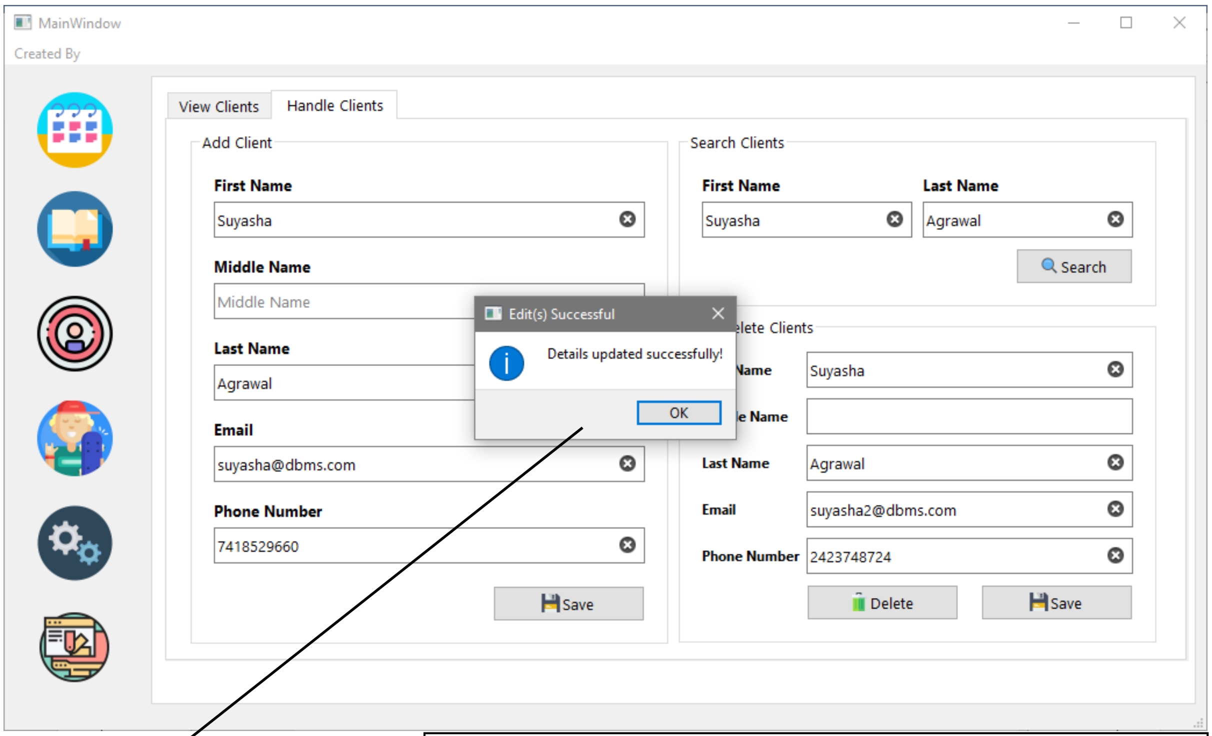
Edit/Delete Clients

First Name	Suyasha
Middle Name	
Last Name	Agrawal
Email	suyasha@dbms.com
Phone Number	7418529660

Client Details Fetched

Image: *Handle Clients tab: Search clients.*

Editing Client Details:



First Name: Suyasha
Middle Name: Middle Name
Last Name: Agrawal
Email: suyasha@dbms.com
Phone Number: 7418529660

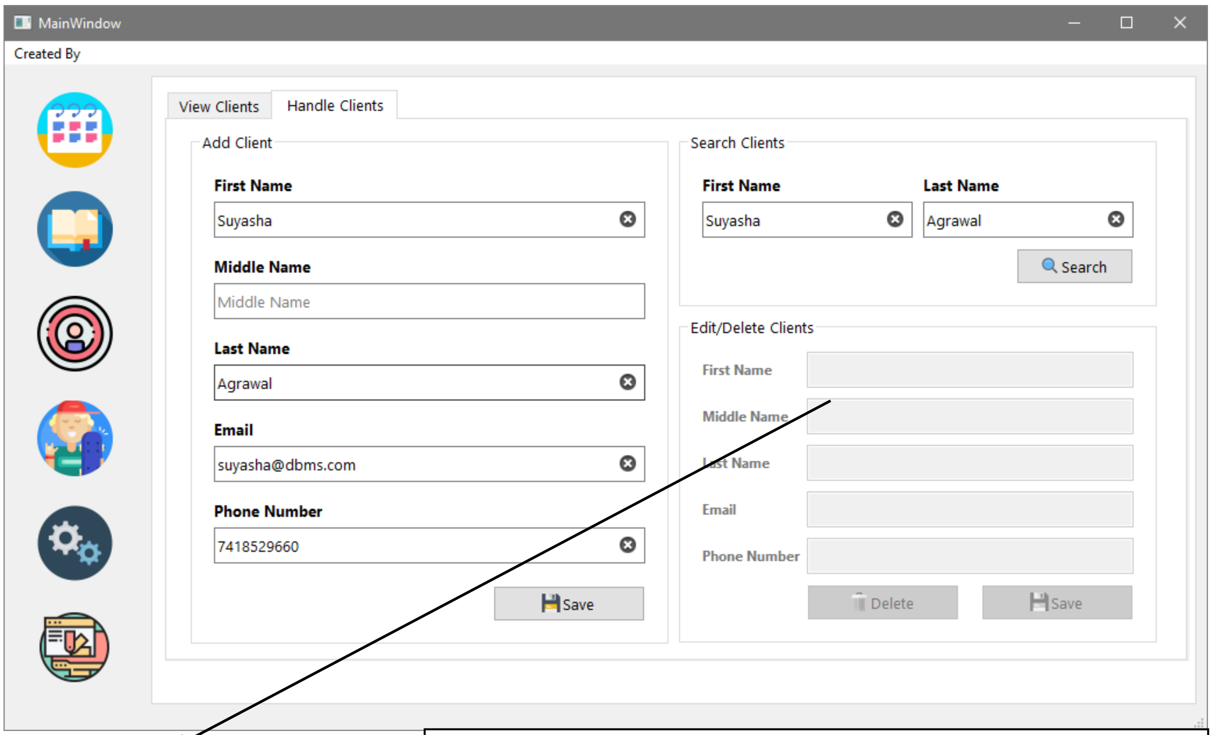
Search Clients — First Name: Suyasha; Last Name: Agrawal

Edit(s) Successful
Details updated successfully!

lete Clients
Name: Suyasha
e Name:
Last Name: Agrawal
Email: suyasha2@dbms.com
Phone Number: 2423748724

Client Details Updated Successfully

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



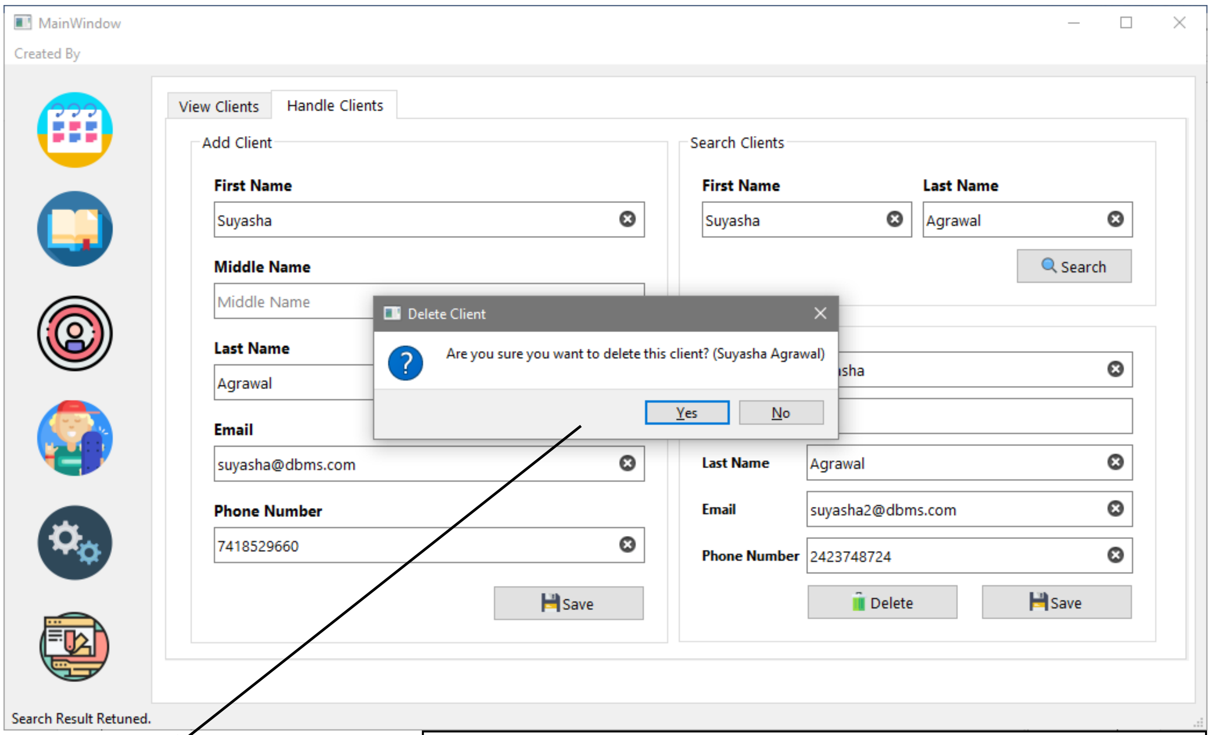
First Name: Suyasha
Middle Name: Middle Name
Last Name: Agrawal
Email: suyasha@dbms.com
Phone Number: 7418529660

Search Clients — First Name: Suyasha; Last Name: Agrawal

Group Box disabled

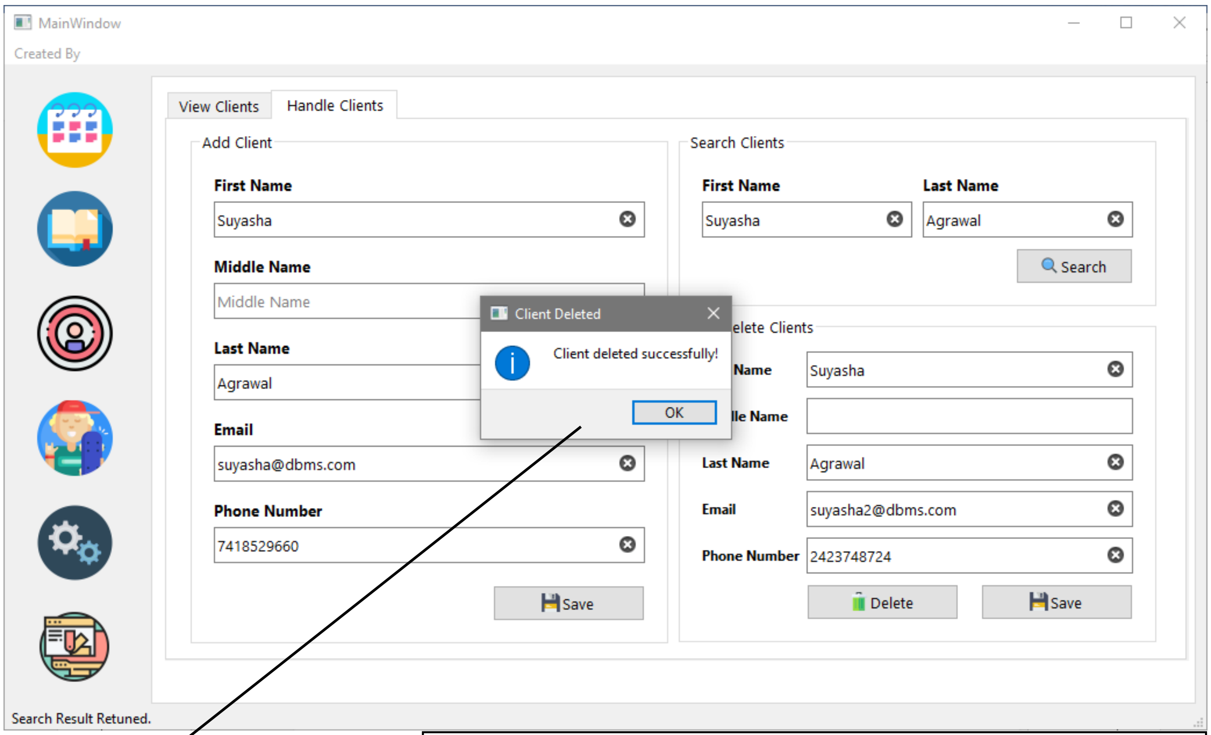
Image: *Handle Clients tab: Post-edit success.*

Deleting a Client:


MainWindow
Created By
View Clients | Handle Clients
Add Client
First Name: Suyasha
Middle Name: Middle Name
Last Name: Agrawal
Email: suyasha@dbms.com
Phone Number: 7418529660
Save
Search Clients
First Name: Suyasha
Last Name: Agrawal
Search
Last Name: Agrawal
Email: suyasha2@dbms.com
Phone Number: 2423748724
Delete
Save
Delete Client
Are you sure you want to delete this client? (Suyasha Agrawal)
Yes | No
Search Result Retuned.

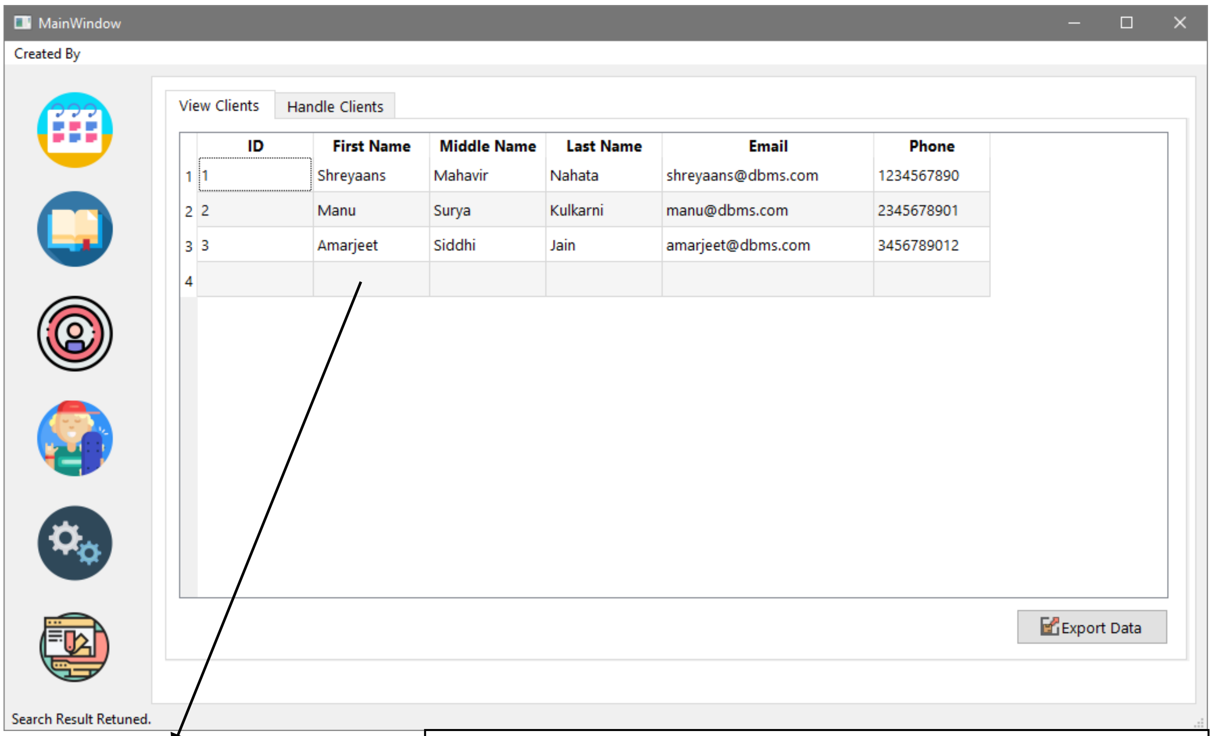
Asking the User for Confirmation

Image: *Delete Client from database.*


MainWindow
Created By
View Clients | Handle Clients
Add Client
First Name: Suyasha
Middle Name: Middle Name
Last Name: Agrawal
Email: suyasha@dbms.com
Phone Number: 7418529660
Save
Search Clients
First Name: Suyasha
Last Name: Agrawal
Search
elete Clients
Name: Suyasha
lle Name:
Last Name: Agrawal
Email: suyasha2@dbms.com
Phone Number: 2423748724
Delete
Save
Client Deleted
Client deleted successfully!
OK
Search Result Retuned.

Successful deletion of client

Image: *Delete Client from database.*


MainWindow

Created By

View Clients | Handle Clients

	ID	First Name	Middle Name	Last Name	Email	Phone
1	1	Shreyaans	Mahavir	Nahata	shreyaans@dbms.com	1234567890
2	2	Manu	Surya	Kulkarni	manu@dbms.com	2345678901
3	3	Amarjeet	Siddhi	Jain	amarjeet@dbms.com	3456789012
4						

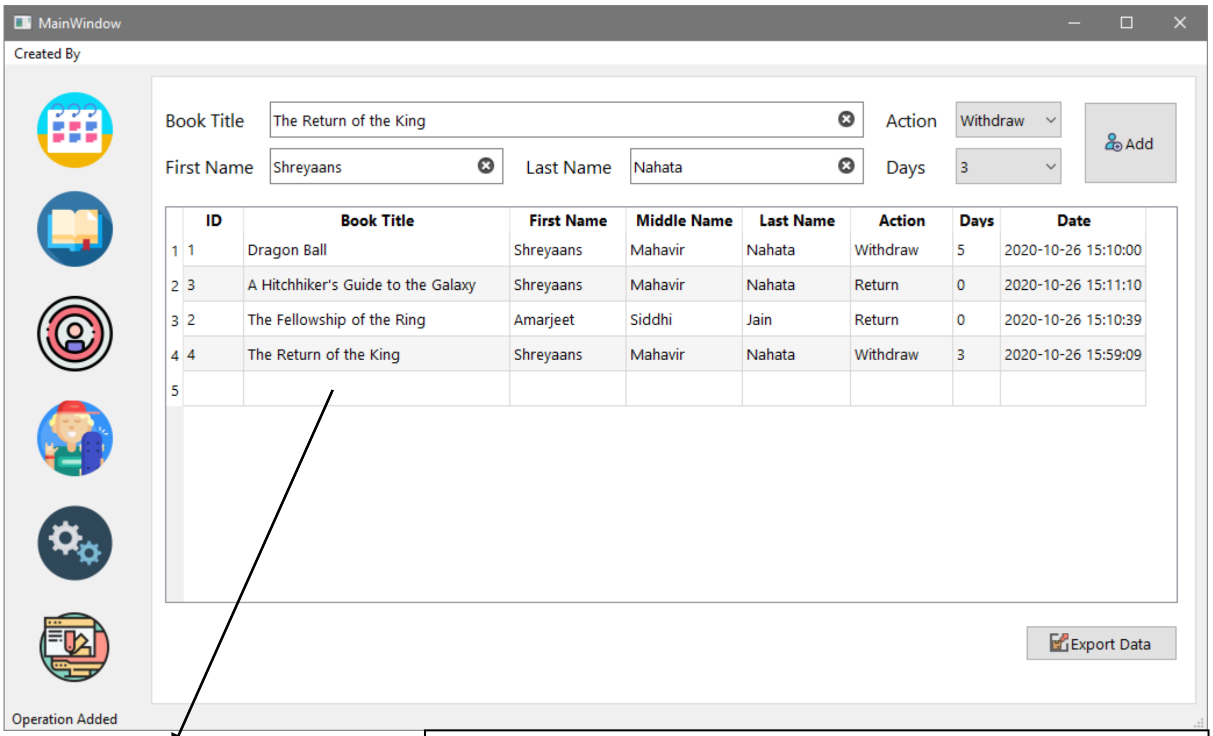
Export Data

Search Result Retuned.

Deleted Client cannot be seen

Image: *View Clients Tab.*

Add a New Operation:


MainWindow

Created By

Book Title: The Return of the King
Action: Withdraw
First Name: Shreyaans
Last Name: Nahata
Days: 3

Add

	ID	Book Title	First Name	Middle Name	Last Name	Action	Days	Date
1	1	Dragon Ball	Shreyaans	Mahavir	Nahata	Withdraw	5	2020-10-26 15:10:00
2	3	A Hitchhiker's Guide to the Galaxy	Shreyaans	Mahavir	Nahata	Return	0	2020-10-26 15:11:10
3	2	The Fellowship of the Ring	Amarjeet	Siddhi	Jain	Return	0	2020-10-26 15:10:39
4	4	The Return of the King	Shreyaans	Mahavir	Nahata	Withdraw	3	2020-10-26 15:59:09
5								

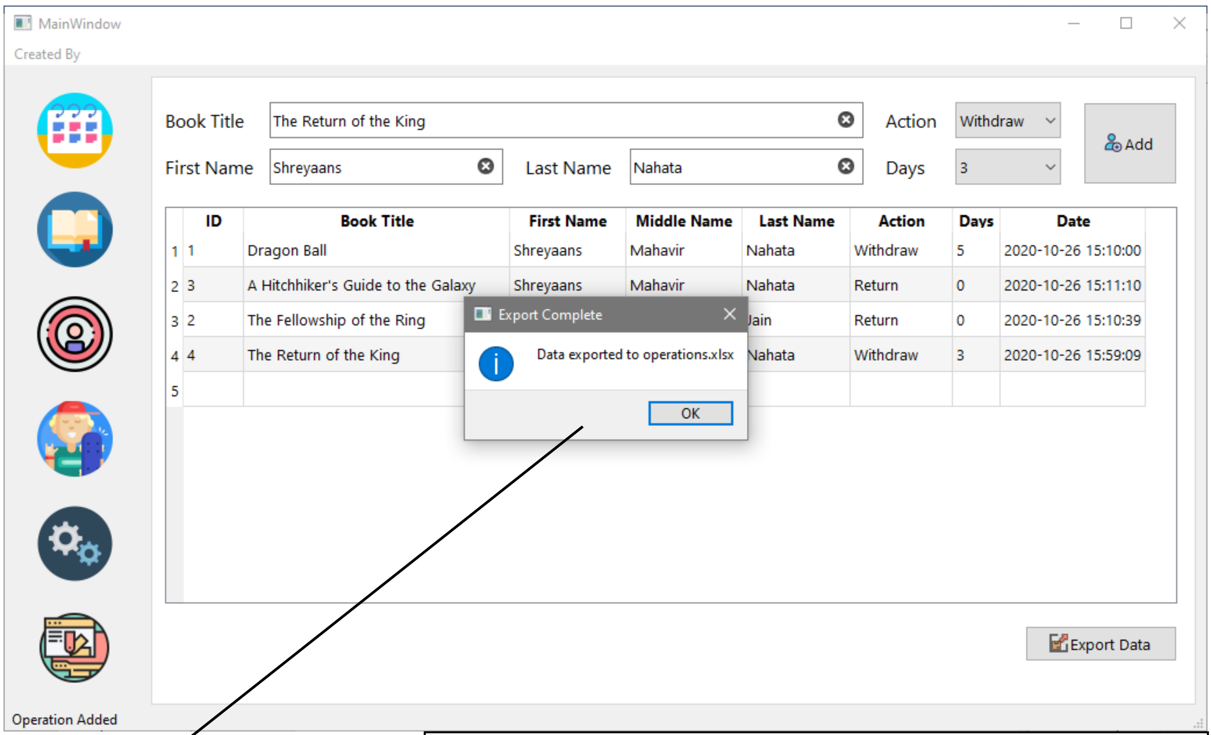
Export Data

Operation Added

New Operation Added

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

Exporting Data to .xlsx format:



MainWindow

Created By

Book Title: The Return of the King

Action: Withdraw

First Name: Shreyaans

Last Name: Nahata

Days: 3

	ID	Book Title	First Name	Middle Name	Last Name	Action	Days	Date
1	1	Dragon Ball	Shreyaans	Mahavir	Nahata	Withdraw	5	2020-10-26 15:10:00
2	3	A Hitchhiker's Guide to the Galaxy	Shreyaans	Mahavir	Nahata	Return	0	2020-10-26 15:11:10
3	2	The Fellowship of the Ring			Jain	Return	0	2020-10-26 15:10:39
4	4	The Return of the King			Nahata	Withdraw	3	2020-10-26 15:59:09
5								

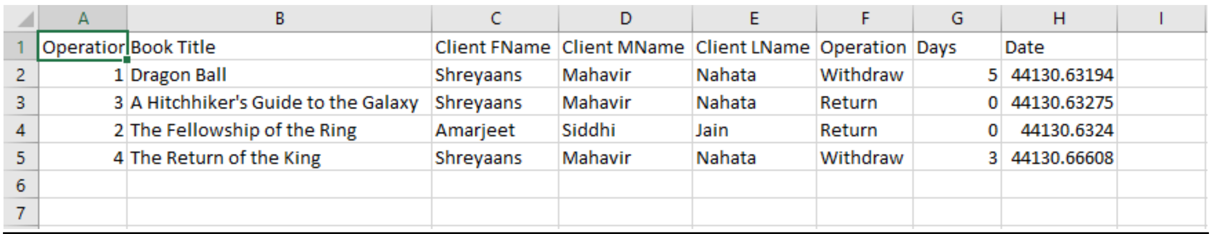
Export Complete

Data exported to operations.xlsx

Operation Added

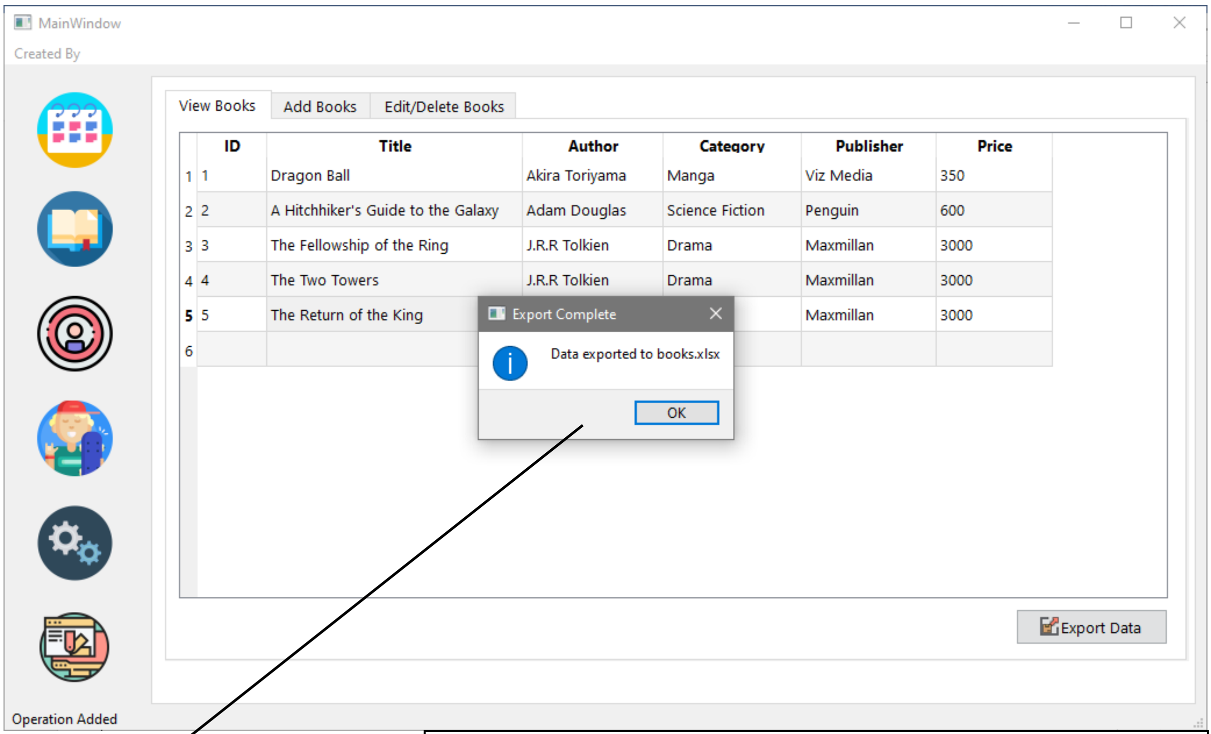
Export completion message

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



	A	B	C	D	E	F	G	H	I
1	Operatior	Book Title	Client FName	Client MName	Client LName	Operation	Days	Date	
2	1	Dragon Ball	Shreyaans	Mahavir	Nahata	Withdraw	5	44130.63194	
3	3	A Hitchhiker's Guide to the Galaxy	Shreyaans	Mahavir	Nahata	Return	0	44130.63275	
4	2	The Fellowship of the Ring	Amarjeet	Siddhi	Jain	Return	0	44130.6324	
5	4	The Return of the King	Shreyaans	Mahavir	Nahata	Withdraw	3	44130.66608	
6									
7									

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



MainWindow

Created By

	ID	Title	Author	Category	Publisher	Price
1	1	Dragon Ball	Akira Toriyama	Manga	Viz Media	350
2	2	A Hitchhiker's Guide to the Galaxy	Adam Douglas	Science Fiction	Penguin	600
3	3	The Fellowship of the Ring	J.R.R Tolkien	Drama	Maxmillan	3000
4	4	The Two Towers	J.R.R Tolkien	Drama	Maxmillan	3000
5	5	The Return of the King			Maxmillan	3000
6						

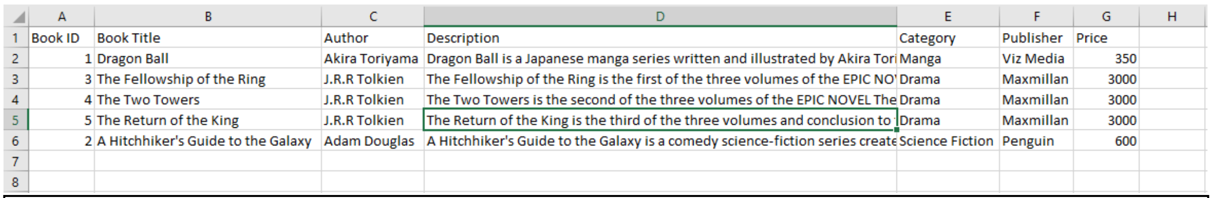
Export Complete

Data exported to books.xlsx

Operation Added

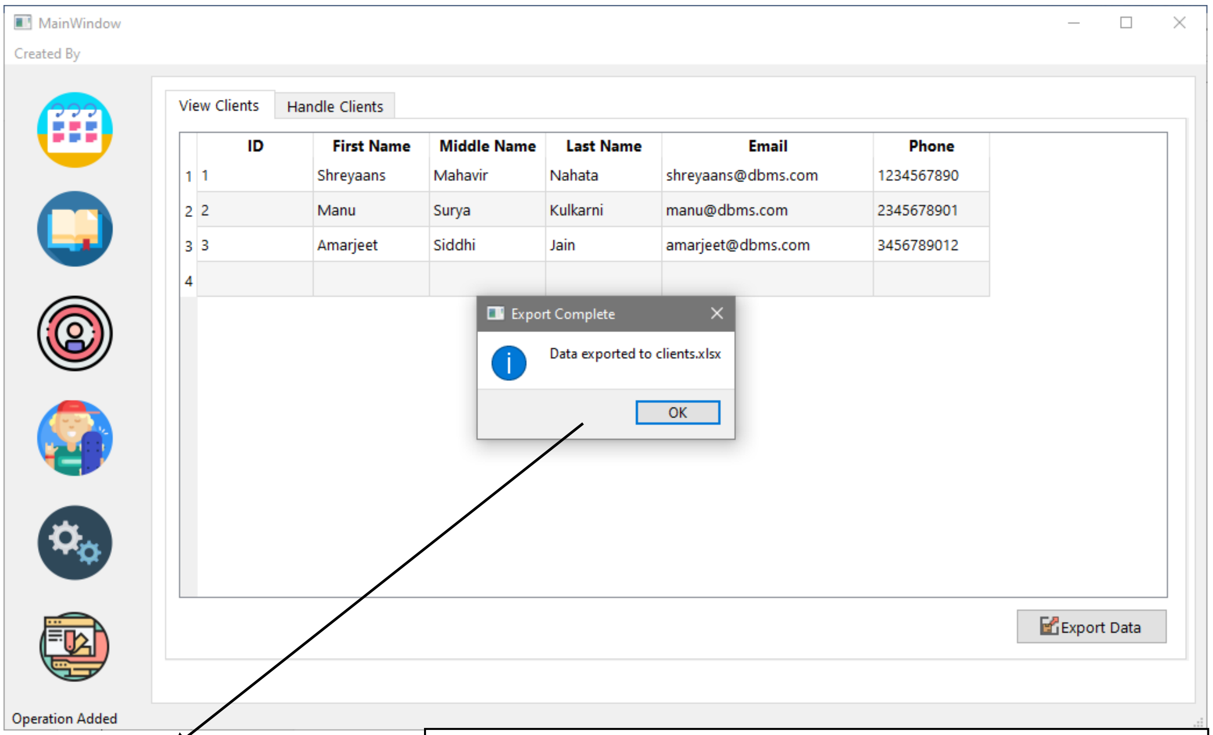
Export completion message

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



Book ID	Book Title	Author	Description	Category	Publisher	Price
1	Dragon Ball	Akira Toriyama	Dragon Ball is a Japanese manga series written and illustrated by Akira Tori	Manga	Viz Media	350
3	The Fellowship of the Ring	J.R.R Tolkien	The Fellowship of the Ring is the first of the three volumes of the EPIC NO	Drama	Maxmillan	3000
4	The Two Towers	J.R.R Tolkien	The Two Towers is the second of the three volumes of the EPIC NOVEL The	Drama	Maxmillan	3000
5	The Return of the King	J.R.R Tolkien	The Return of the King is the third of the three volumes and conclusion to	Drama	Maxmillan	3000
2	A Hitchhiker's Guide to the Galaxy	Adam Douglas	A Hitchhiker's Guide to the Galaxy is a comedy science-fiction series create	Science Fiction	Penguin	600

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

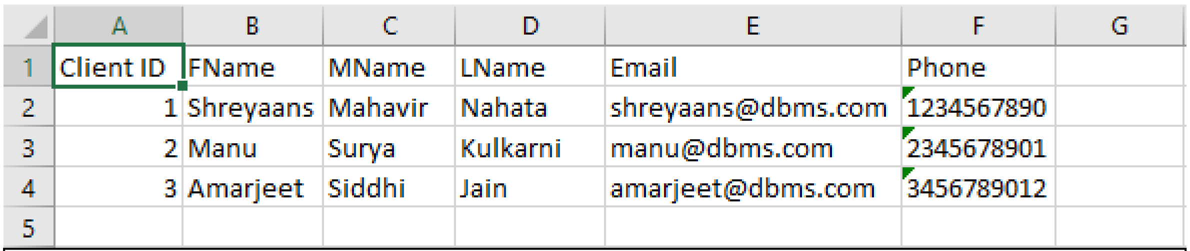


	ID	First Name	Middle Name	Last Name	Email	Phone
1	1	Shreyaans	Mahavir	Nahata	shreyaans@dbms.com	1234567890
2	2	Manu	Surya	Kulkarni	manu@dbms.com	2345678901
3	3	Amarjeet	Siddhi	Jain	amarjeet@dbms.com	3456789012
4						

Data exported to clients.xlsx

Export completion message

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



Client ID	FName	MName	LName	Email	Phone
1	Shreyaans	Mahavir	Nahata	shreyaans@dbms.com	1234567890
2	Manu	Surya	Kulkarni	manu@dbms.com	2345678901
3	Amarjeet	Siddhi	Jain	amarjeet@dbms.com	3456789012

***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='libra
rysys')

        # 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!', QM
essageBox.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 Successf
ul!', 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 t
he 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='libra
rysys')
        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='libra
rysys')
```

```
        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='libra
rysys')
        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(boo
k_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='libra
rysys')
        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!', QMes
sageBox.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='libra
rysys')
        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) V
ALUES (%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='libra
rysys')
        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='libra
rysys')
        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_em
ail = %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='libra
rysys')
        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 dele
te 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='libra
rysys')
        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 b
oth 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='libra
rysys')
        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!', Q
MessageBox.Ok)

                    self.lineEdit_17.setText(row[1])
                    self.lineEdit_15.setText(row[2])

                else:
                    warning = QMessageBox.warning(self, 'Incorrect Details','Please enter t
he 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='libra
rysys')
        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 successfull
y!', 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 b
oth 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='libra
rysys')
        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 de
lete 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 b
oth 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='libra
rysys')
        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='libra
rysys')
        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='libra
rysys')
        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='libra
rysys')
        self.cur = self.db.cursor()
        auth_name = self.lineEdit_19.text()

        warning = QMessageBox.question(self, 'Delete Author','Are you sure you want to dele
te 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!', QMe
ssageBox.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='libra
rysys')
        self.cur = self.db.cursor()

        pub_name = self.lineEdit_22.text()

        warning = QMessageBox.question(self, 'Delete Publisher','Are you sure you want to d
elete 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='libra
rysys')
        self.cur = self.db.cursor()
        cat_name = self.lineEdit_24.text()

        warning = QMessageBox.question(self, 'Delete Category','Are you sure you want to de
lete 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='libra
rysys')
        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, da
te)
            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='libra
rysys')
        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='libra
rysys')
        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='libra
rysys')
        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='libra
rysys')
        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='libra
rysys')
        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='libra
rysys')
        self.cur = self.db.cursor()

        self.cur.execute('''SELECT op_id, book.book_name, clients.first_name, clients.middl
e_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.clie
nt_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='libra
rysys')
        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='libra
rysys')
        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='libra
rysys')
        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='libra
rysys')
        self.cur = self.db.cursor()

        self.cur.execute('''SELECT op_id, book.book_name, clients.first_name, clients.middl
e_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.clie
nt_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='libra
rysys')
        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='libra
rysys')
        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.