LIBRARY MANAGEMENT SYSTEM

[DBMS MINI-PROJECT REPORT]

Submitted by

I. Mohana Chaitanya Reddy (RA1911028010141)
and
Pratiksha Ghosh (RA1911028010142)

in partial fulfillment for the award of the degree of
B. TECH
in
COMPUTER SCIENCE and ENGINEERING
IN
CLOUD COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR

[April, 2022]

SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the Mini project of DBMS report titled 'LIBRARY MANAGEMENT SYSTEM' is the bonafide work of **"**I. MOHANA CHAITANYA REDDY **[RA1911028010141]**" and "PRATIKSHA GHOSH **[RA1911028010142]",** submitted for the course- **18CSC303J** name of DBMS. This report is a record of successful completion of the specified course evaluated based on literature reviews and the supervisor. No part of the DBMS has been submitted for any degree, diploma, title, or recognition before.

SIGNATURE
DR.S. SURESH
Professor
Dept. of Computer Science &Engineering

ACKNOWLEDGEMENT

We would like to convey our heartfelt gratitude to Dr. S. Suresh for providing us with this wonderful opportunity to work on a project with the topic Library Management System. This project would not have been accomplished without his tremendous direction and assistance.

We would also like to thank our parents for their constant support and encouragement.

TABLE OF CONTENTS

SERIAL NO.	TITLE	PAGE NO.
1	INTRODUCTION	5
2	ABSTRACT	5
3	SOFTWARE REQUIREMENTS	6
4	ENTITIES AND RELATIONSHIPS	7
5	E-R DIAGRAM	8
6	SQL QUERIES	9
7	CODE: GITHUB LINK	11
8	OUTPUT SCREENSHOTS	11
9	CONCLUSION AND FUTURE WORK	14
10	REFERENCES	14

INTRODUCTION

Database is an organized collection of data. The data is typically organized to model aspects of reality in a way that supports processes requiring information. A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible. The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data.

ABSTRACT

Library Management System allows the user to store the book details and the customer details. The system is designed to withstand monthly operations under conditions where the database is maintained and cleared over a certain time span. The implementation of this system will reduce data entry time and efforts, and also provide readily calculated reports.

It keeps track of all the information about the books in the library, their cost, status and total number of books available. User will find it easy as it is automated system rather than using the manual writing system. This database system contains all the information safely.

SOFTWARE REQUIREMENTS

- TKINTER LIBRARY: Tkinter is a standard library in Python which is used for GUI application. Tkinter has various controls which are used to build a GUI-based application.
  Our project frontend has been created with the use of the Tkinter libraries.
  Python with Tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using Tkinter is an easy task.

- SQLITE 3 LIBRARY: SQLite in general is a server-less database that you can use within almost all programming languages including Python. Server-less means there is no need to install a separate server to work with SQLite so you can connect directly with the database.
  We have used SQLite3 in our project, in order to connect the databases to the python code.

- VISUAL STUDIO CODE: Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control.

ENTITIES AND RELATIONSHIPS

ENTITIES:

BOOKS
PUBLISHER
STATUS
MEMBER

ATTRIBUTES:

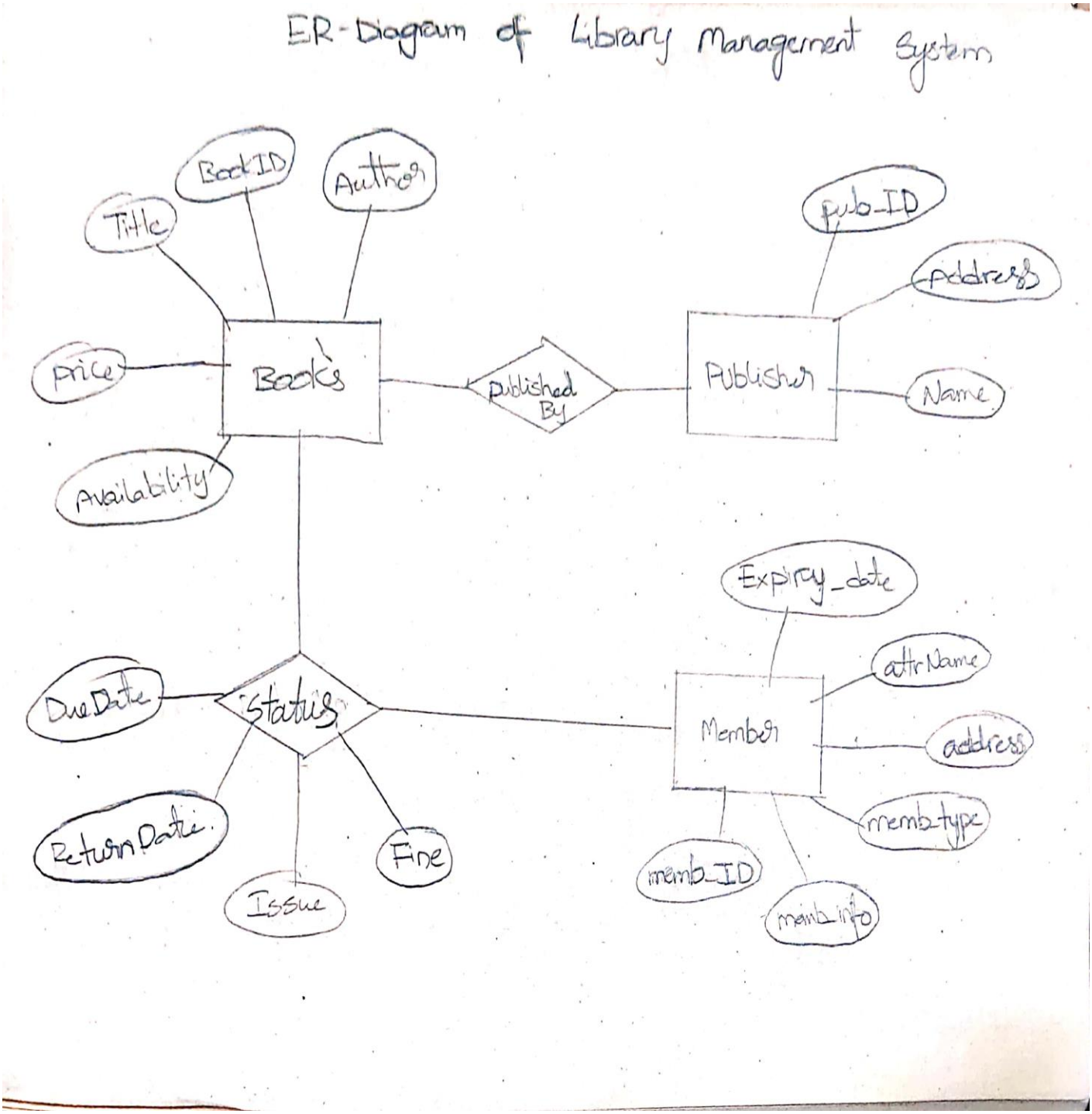
BOOKS:
- -BOOK ID
- -TITLE
- -AUTHOR
- -GENRE
- -PRICE

PUBLISHER:
- -PUBLISHER_ID
- -PUBLISHER_NAME
- -ADDRESS

STATUS:
- -ISSUE DATE
- -DUE DATE
- -RETURN DATE
- -FINE AMT

MEMBER:
- -MEM_ID
- -MEM_NAME
- -ADDRESS
- -PHONE NO.
- -EMAIL ID

ENTITY RELATIONSHIP (E-R) DIAGRAM:



SQL QUERIES

CREATING TABLES:

```
CREATE TABLE BOOK(
book_id INT NOT NULL PRIMARY KEY,
title VARCHAR(200),
author VARCHAR(200),
genre VARCHAR(200),
price FLOAT,
availability INT,
publisher_id INT,
FOREIGN KEY(publisher_id) REFERENCES PUBLISHER(publisher_id) ON DELETE CASCADE
);
```

INPUT VALUES INTO TABLES:

```
c.execute("INSERT INTO BOOK VALUES (:book_id, :title, :author, :genre, :price, :availability,
:publisher_id)",
                {
                   'book_id' : book_id.get(),
                   'title' : title.get(),
                   'author' : author.get(),
                   'genre' : genre.get(),
                   'price' : price.get(),
                   'availability' : availability.get(),
                   'publisher_id' : publisher_id.get(),
                })
```

QUERYING THE TABLE:

```
my_tree = ttk.Treeview(QUERY_BOOK)

        c.execute("SELECT * FROM BOOK")
        records = c.fetchall()
        # print(records)

        my_tree['columns'] = ('book_id', 'title', 'author', 'genre', 'price', 'availability', 'publisher_id')

        my_tree.column("#0", width=0, stretch=NO)
        my_tree.column("book_id", width=80, anchor=CENTER)
        my_tree.column("title", width=200, anchor=W)
        my_tree.column("author", width=200, anchor=W)
```

```
        my_tree.column("genre", width=140, anchor=W)
        my_tree.column("price", width=100, anchor=W)
        my_tree.column("availability", width=100, anchor=W)
        my_tree.column("publisher_id", width=80, anchor=W)

        my_tree.heading("#0", text="", anchor=W)
        my_tree.heading("book_id", text="book_id", anchor=CENTER)
        my_tree.heading("title", text="title", anchor=W)
        my_tree.heading("author", text="author", anchor=W)
        my_tree.heading("genre", text="genre", anchor=W)
        my_tree.heading("price", text="price", anchor=W)
        my_tree.heading("availability", text="availability", anchor=W)
        my_tree.heading("publisher_id", text="publisher_id", anchor=CENTER)

        count = 0
        for rec in records:
            my_tree.insert(parent='', index='end', iid=count, text="",
values=(rec[0],rec[1],rec[2],rec[3],rec[4],rec[5],rec[6]))
            count+=1
```

DELETING VALUES:

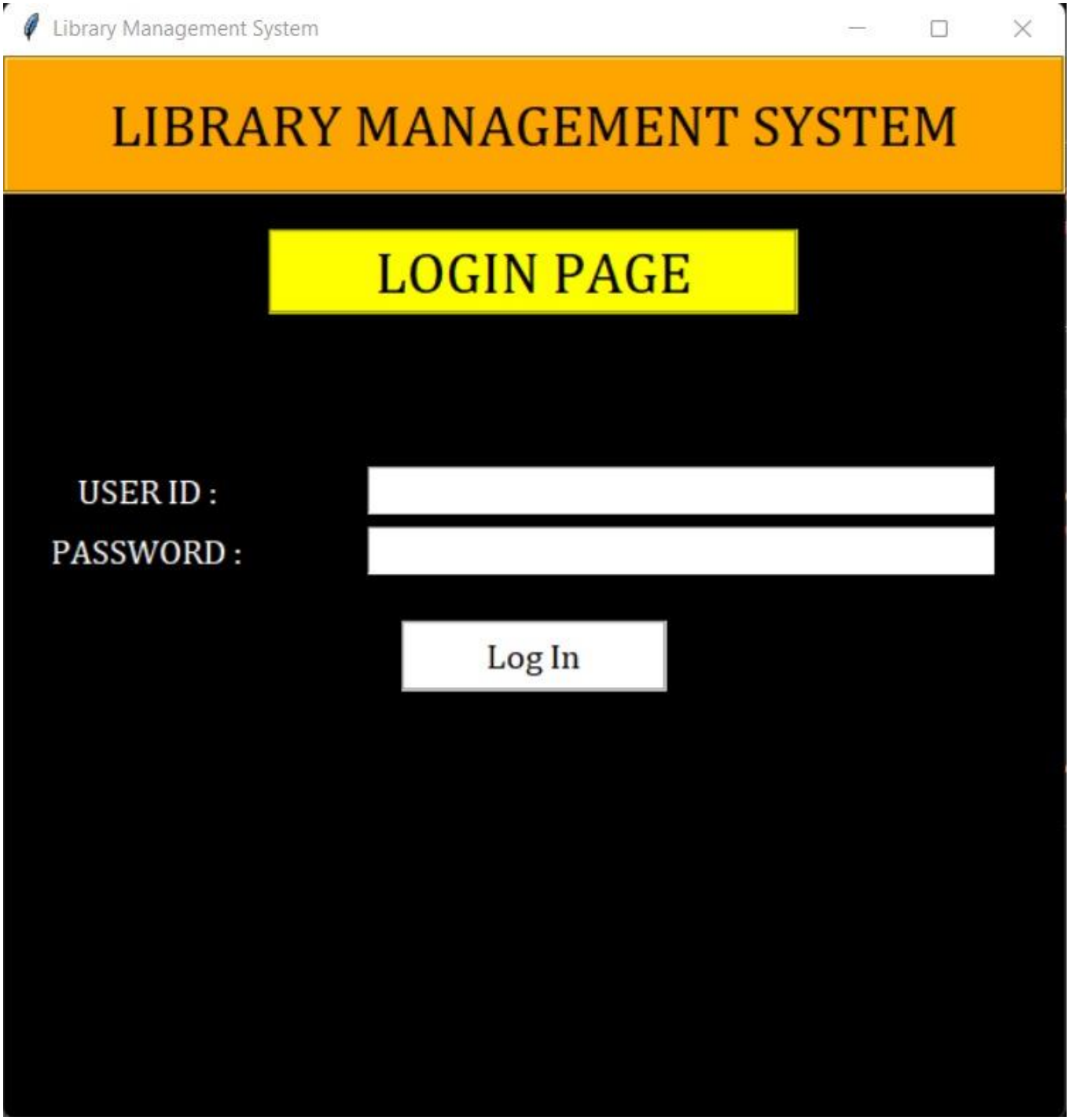
```
c.execute("DELETE FROM BOOK WHERE book_id = " + delete_book_id.get())
```

CODE: GITHUB LINK

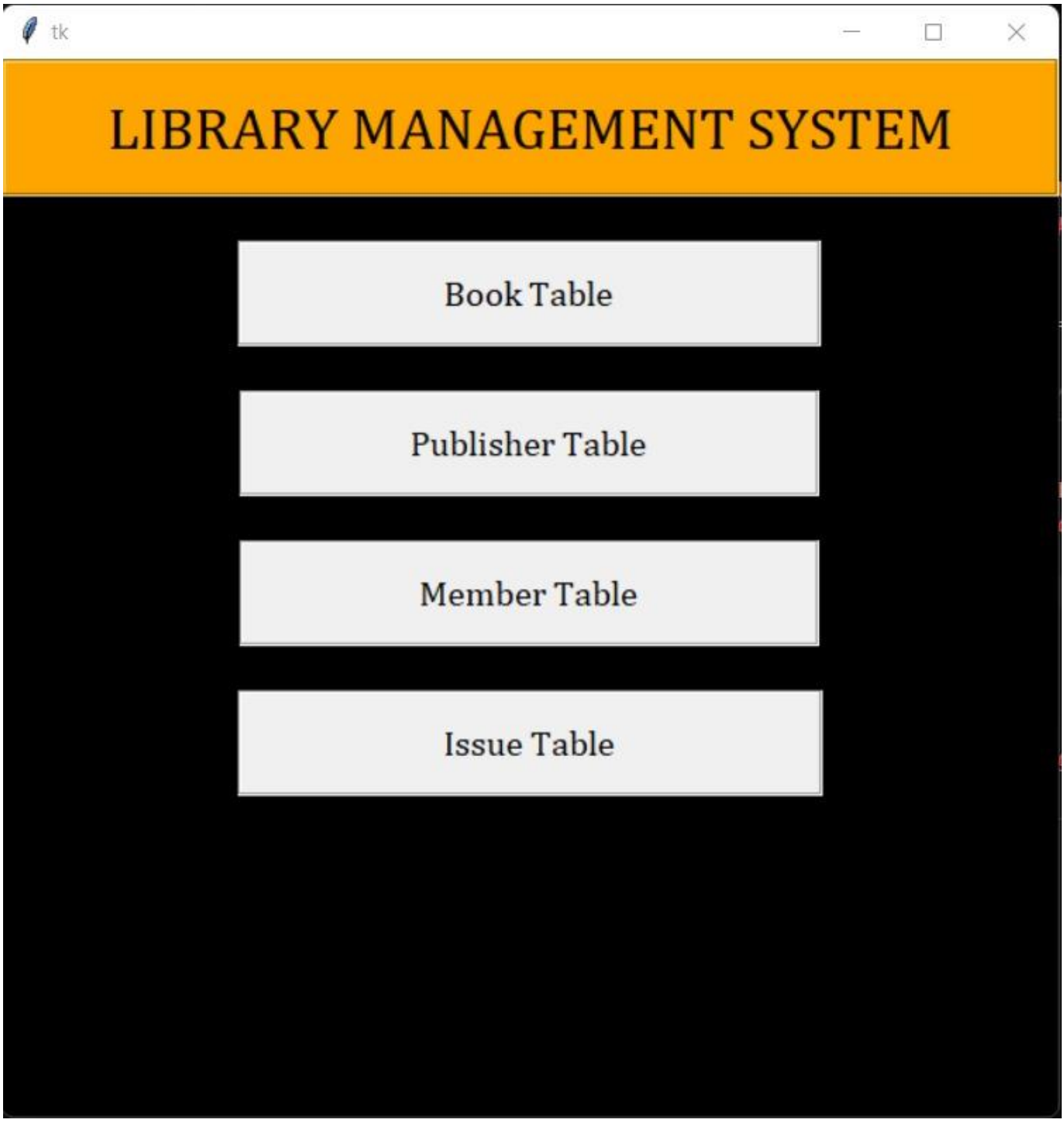
https://github.com/PratikshaGhosh/Library-Management-System

OUTPUTS

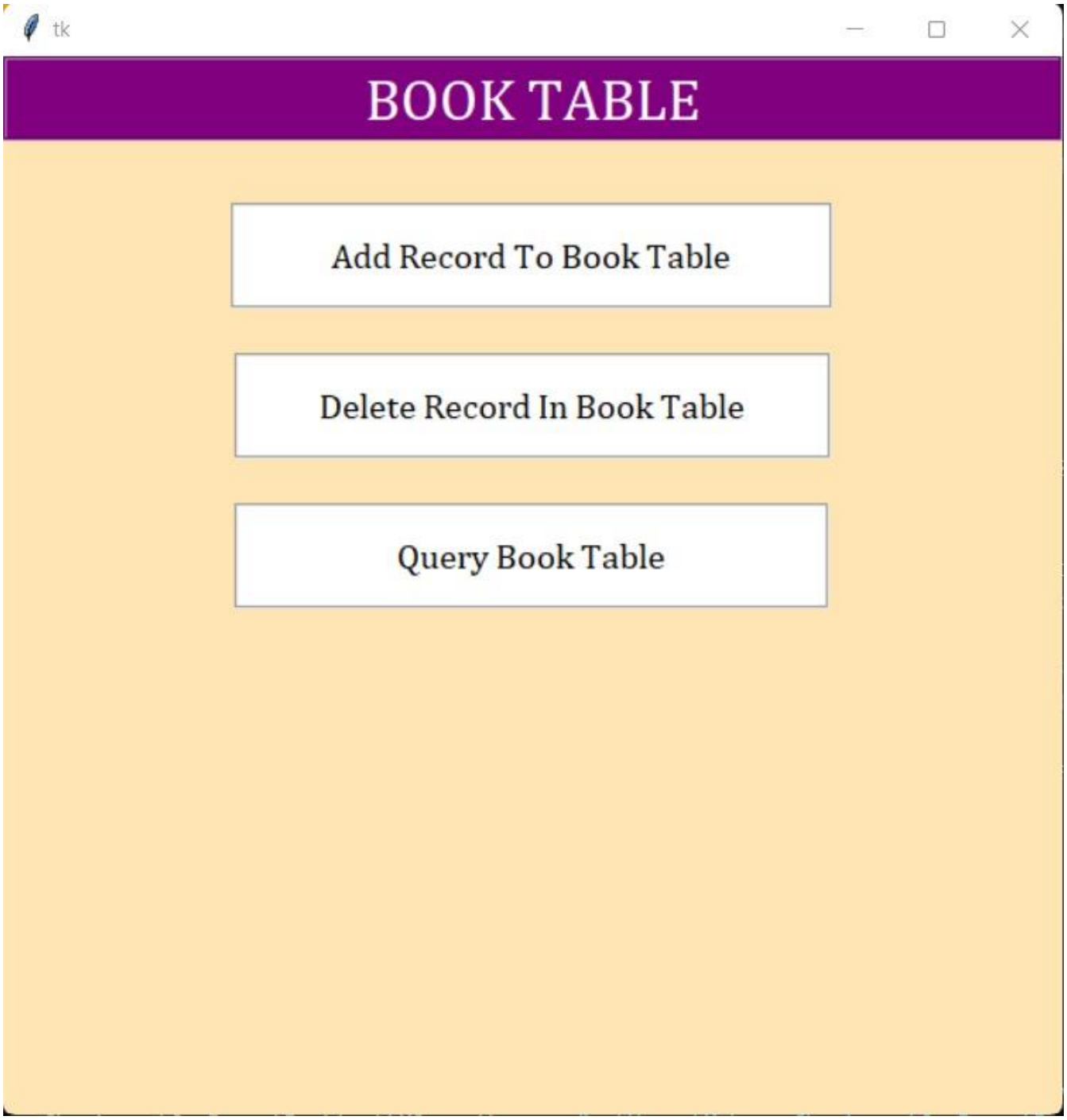
1. LOGIN PAGE:



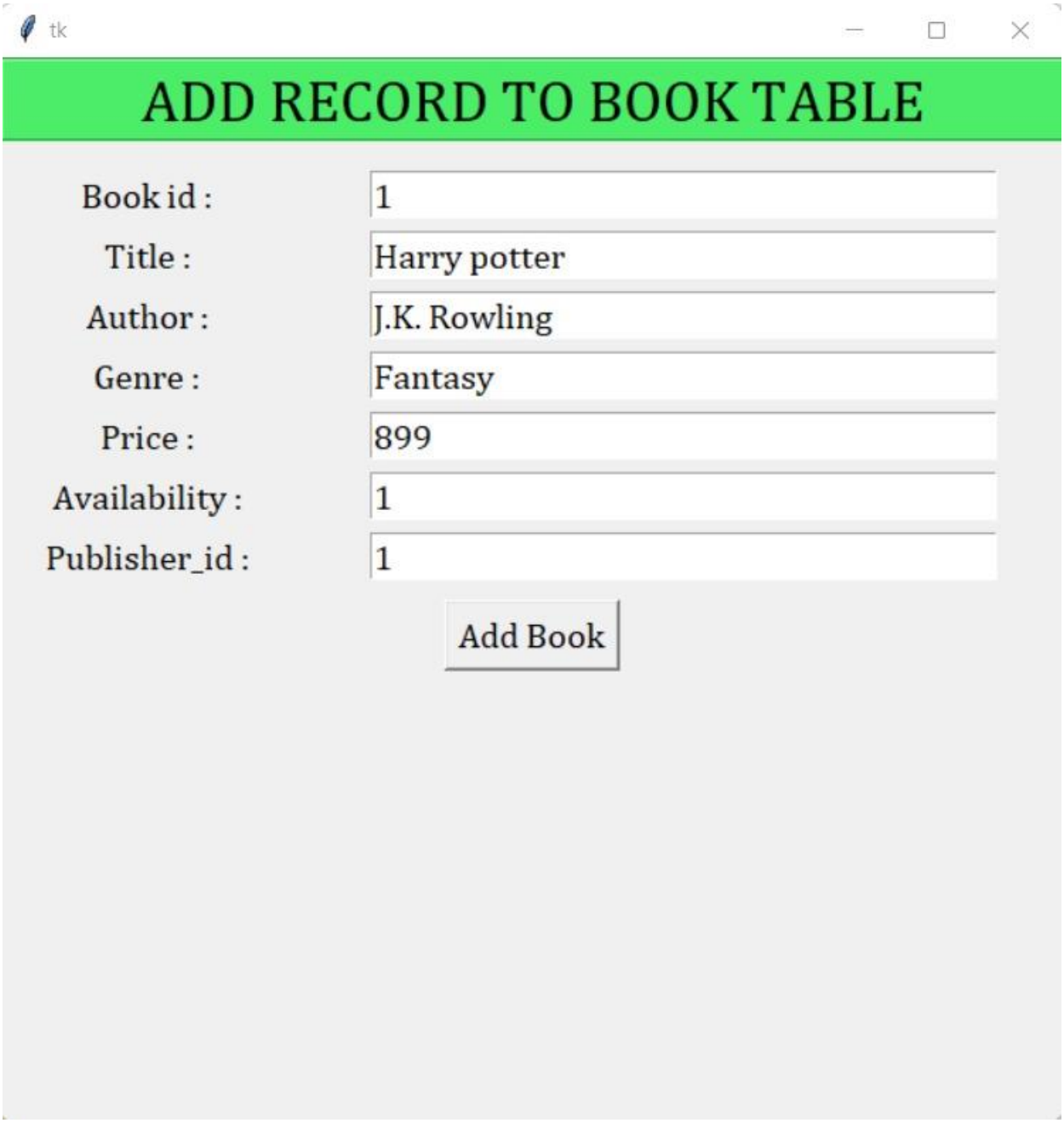
2. WELCOME PAGE:



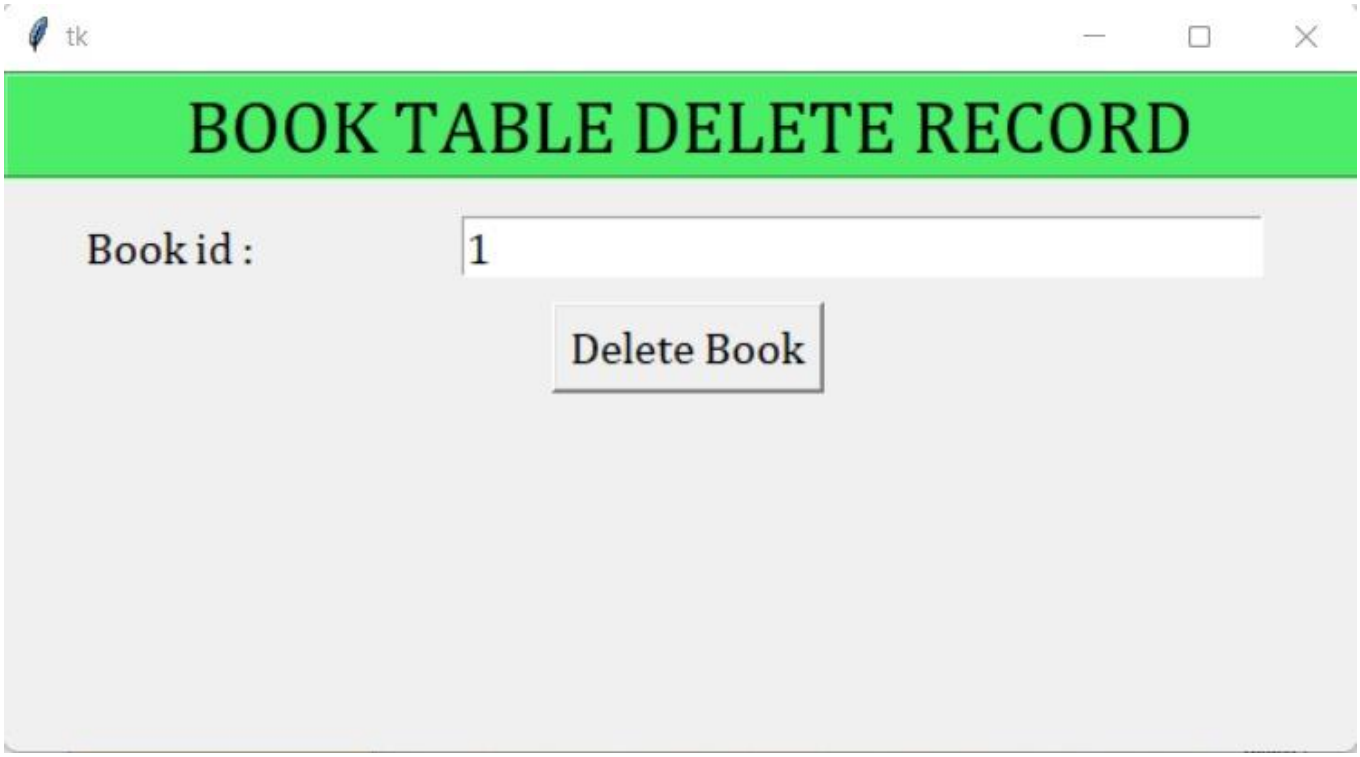
3. TABLE FUNCTIONS:



4. ADDING RECORDS:



5. DELETING RECORDS:



6. QUERYING TABLE:

member_id	name	contact_no	email_id	address
1	Mohana	9876789541	mohana@gmail.com	Guntur
2	Chaitanya	8796728954	chaitanya23@gmail.com	Vizag
3	Pratiksha Ghosh	8567234876	pratiksha791@gmail.cpm	kolkata
4	Gopal	8789098752	gopal428@gmail.com	Hyderabad
5	Harshith	8067854231	harshith356@gmail.com	Banglore

CONCLUSION AND FUTURE WORK:

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient GUI based component. This application is working properly and meeting to all user requirements. This component can be easily plugged in many other systems.

This application can be easily implemented under various situations. We can add new features as and when we require. Reusability is possible as and when require in this application. There is flexibility in all the modules. This software is extendable in ways that its original developers may not expect. The following principles enhances extensibility like hide data structure, avoid traversing multiple links or methods, avoid case statements on object type and distinguish public and private operations. Reusability is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability: Sharing of newly written code within a project and reuse of previously written code on new projects.

REFERENCES:

https://techvidvan.com/tutorials/python-library-management-system/
https://www.geeksforgeeks.org/python-tkinter-tutorial/
https://www.tutorialspoint.com/dbms/dbms_quick_guide.htm