**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JNANA SANGAMA”, BELAGAVI-590018**

****

*A MINI PROJECT REPORT ON*

LIBRARY MANAGEMENT SYSTEM

Submitted in partial fulfilment of the requirement

For the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

By

YASHWANTH K

[1KS18CS121]

Under the guidance of

Mr. Kumar K Dr. Dayananda R B

Asst. Prof, Dept. Of CSE Asst. Prof, Dept. Of CSE



Department of Computer Science & Engineering

K.S.INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

2020-21

K. S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-56010

**Department of Computer Science & Engineering**

****

**CERTIFICATE**

This is to certify that mini project work entitled “LIBRARY MANAGEMENT SYSTEM” carried out by Mr. YASHWANTH K bearing USN 1KS18CS121 bonafide students of K.S. Institute of Technology in the partial fulfilment for the award of the Bachelor of Engineering in Computer Science & Engineering of the Visvesvaraya Technological University, Belagavi, during the year 2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini Project work prescribed for the said degree for the 5th semester

**Dr . Rekha B Venkatapur Dr .K V A Balaji**

Prof & HOD, CS & E Department Principal/CEO, KSIT

**Mr. Kumar K** **Dr. Dayananda R B­­**

Asst. Prof, Dept. Of CSE Asst. Prof, Dept. Of CSE

Name of the Examiners Signature with date

1.

2.

ACKNOWLEDGEMENT

I take this opportunity to thank one and all involved in building this project. Firstly I would like to thank the college for providing us an opportunity to work on the project.

I would also like to thank the management of K. S. Institute of Technology for providing all the resources required for the project.

I wish to acknowledge my sincere gratitude to our beloved Principal, **Dr. K V A BALAJI** for his encouragement and providing all facilities for the accomplishment of this project.

This project would not have been possible without the support of our beloved Prof & HOD, **Dr. Rekha.B.Venkatapur**, Dept. of CSE.

I am also highly grateful to my project guides, **Mr. Kumar K and Dr. Dayananda R B, Dept. of CSE** who have been very generous in assisting and supporting, to do this Project **“LIBRARY MANAGEMENT SYSTEM”**.

I also would like to thank all other teaching and non-teaching staff members who have extended their support and co-operation while bringing up this project.

YASHWANTH K

(1KS18CS121)

ABSTRACT

A library management system keeps track of the books present in the library. It is an important piece of software which is a must at schools and colleges.

CONTENTS

***1.INTRODUCTION 1-3***

1.1 OVERVIEW 1

1.2 PROBLEM STATEMENT 1

1.3 DATABASE MANAGEMENT SYSTEM 1-2

1.4 SQL 2

1.5 PYTHON WITH TKINTER 2-3

1.6 SQLITE3 3

***2. REQUIREMENTS SPECIFICATION 4-5***

2.1 OVER DESCRIPTION 4

2.2 SPECIFIC REQUIREMENTS 4

2.2.1 SOFTWARE REQUIREMENTS 4

2.2.2 HARDWARE REQUIREMENTS 4

2.3 TECHNOLOGY 5

***3. DETAILED DESIGN 6-13***

3.1 SYSTEM DESIGN 6-7

3.2 ENTITY RELATIONSHIP DIAGRAM 8-10

3.3 RELATIONAL SCHEMA 11

3.4 DESCRIPTION OF TABLES 12-13

* 1. OVERVIEW

A library management is a project that manages and stores books information electronically accordingly. It keeps track of the books present in the library. It is an important piece of software which is a must at schools and colleges. It allows to search for the desired book. It becomes necessary to keep a continuous check on the books issued. This task if carried out manually will be tedious and includes chances of mistakes. These errors are avoided by allowing the system to keep track of information such as issue date and book details thus there is no need to keep manual track of this information which thereby avoids chances of mistakes. Thus this system reduces manual work to a great extent allows smooth flow of library activities by removing chances of errors in the details.

1.2 PROBLEM STATEMENT

The main aim of the Library Management Software is to handle the entire activity of a library. The software keeps track of all the information about the books in the Library. The user will find it easy in this automated system rather than using the manual writing system. The system contains a database where all the information will be stored safely. The system is user-friendly and error free. The “Library Management Software” has been developed to override the problems prevailing in the practicing manual system.

* 1. DATABASE MANAGEMENT SYSTEM

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. The DBMS essentially serves as an interface between the database and end users 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 scheme, which defines the database’s logical structure. These three foundational elements help to provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

* 1. SQL

SQL is a standard language for storing, manipulating and retrieving data in databases. SQL stands for Structured Query Language Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

* 1. PYTHON WITH TKINTER

Python offers various utilities to design the GUI wiz Graphical User Interface, and one such utility is Tkinter which is most commonly used. It is indeed one of the fastest and easiest ways to build GUI applications. Moreover, Tkinter is cross-platform, hence the same code works on macOS, Windows, and Linux. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that’s functional and cross-platform quickly.

* 1. SQLITE3

To connect the database with the front end we use sqlite3. Sqlite3 is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages. SQLite is a C-language library that implements a small, fast, selfcontained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world.

1. **REQUIREMENTS SPECIFICATION**

A computerized way of handling information about property and users details

is efficient, organized and time saving, compared to a manual way of doing

so. This is done through a database driven software application whose

requirements are mentioned in this section.

* 1. **OVERALL DESCRIPTION**

A reliable and scalable database driven software application with security features that is easy to use and maintain is the requisite.

* 1. **SPECIFIC REQUIREMENTS**

The specific requirements of the Library Management System are stated as follows:

* + 1. ***SOFTWARE REQUIREMENTS***

|  |  |  |
| --- | --- | --- |
| **Number** | **Description** | ***Type*** |
| ***1*** | Operating System | Windows |
| ***2*** | Language | PYTHON |
| ***3*** | Database | SQLITE3 |
| ***4*** | IDE | cmd or Visual Code |

* + 1. ***HARDWARE REQUIREMENTS***

|  |  |
| --- | --- |
| **Number** | **Description** |
| ***1*** | PC with 5 GB or more Hard disk. |
| ***2*** | PC with 2 GB RAM |
| ***3*** | PC with Pentium 1 and Above |
| ***4*** | Monitor Screen Resolution 1300x700 |
| ***5*** | Keyboard and Mouse |

* + 1. ***TECHNOLOGY***
* Tkinter : Tkinter is one of the Python libraries which contains many functions for the development of graphic user interface pages and windows. Python with tkinter used as Front end for our Library Management System
* SQL: SQL is the language used to manipulate relational databases. It is tied closely with the relational model. It is issued for the purpose of data definition and data manipulation.
* Sqlite3: SQLite is a software library that provides a relational database management system. The lite in SQLite means lightweight in terms of setup, database administration, and required resources.

SQLite has the following noticeable features:

1. self-contained

2. server less

3. zero-configuration

4. transactional

Sqlite3 is used as backend to store the data of our Library Management System

3. DETAILED DETAILS

**3.1 SYSTEM DESIGN**

SQLite is a single file relational database bundled with most standard Python installs. Creating a new SQLite database is as simple as creating a connection using the sqlite3 module in the Python standard library. To establish a connection all you need to do is pass a file path to the connect(...) method in the sqlite3 module, and if the database represented by the file does not exists one will be created at that path.

* Server-side database

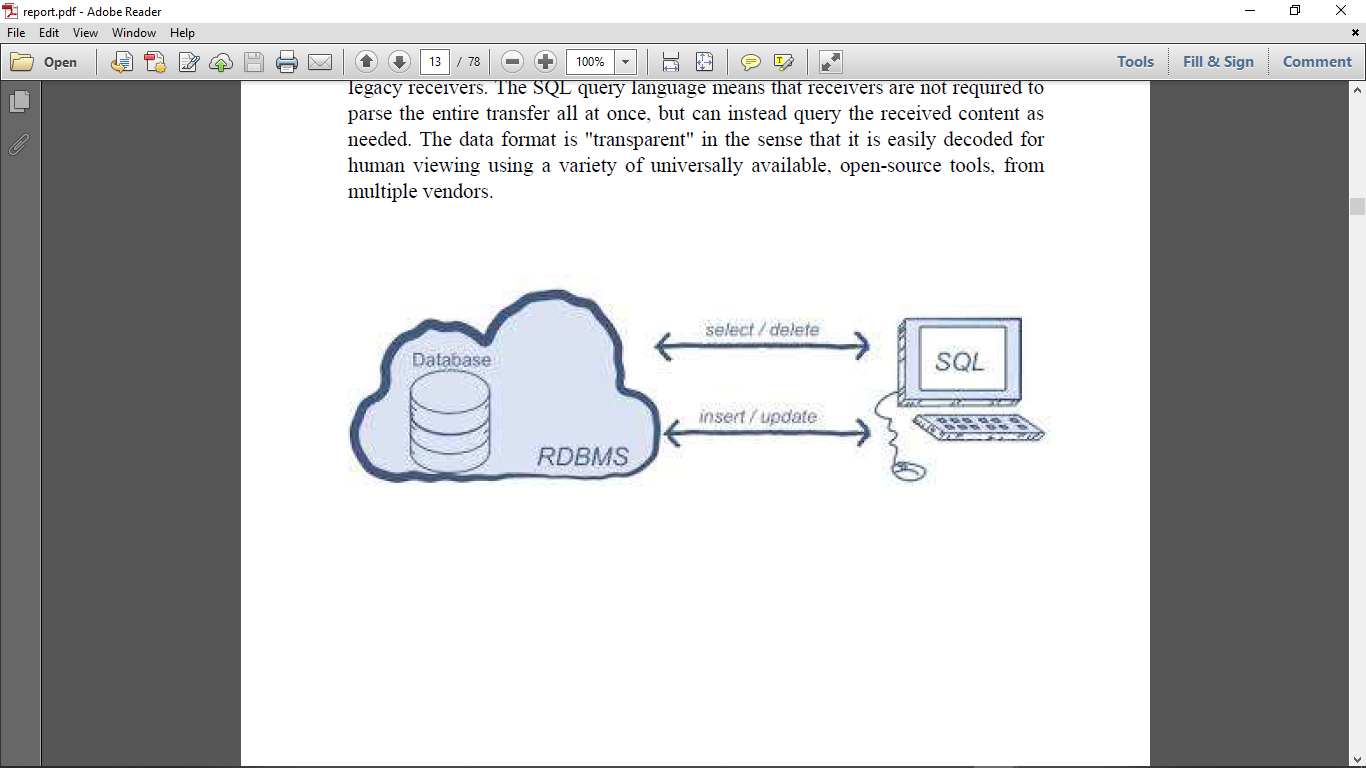
With this pattern, the overall system is still client/server: clients send requests to the server and get back replies over the network. But instead of sending generic SQL and getting back raw table content, the client requests and server responses are highlevel and application-specific. The server translates requests into multiple SQL queries, gathers the results, does post-processing, filtering, and analysis, then constructs a high-level reply containing only the essential information. SQLite is often faster than a client/server SQL database engine in this scenario. Database requests are serialized by the server, so concurrency is not an issue. Concurrency is also improved by "database sharing": using separate database files for different subdomains. For example, the server might have a separate SQLite database for each user, so that the server can handle hundreds or thousands of simultaneous connections, but each SQLite database is only used by one connection.

* Data transfer format

Because an SQLite database is a single compact file in a well-defined cross-platform format, it is often used as a container for transferring content from one system to another. The sender gathers content into an SQLite database file, transfers that one file to the receiver, then the receiver uses SQL to extract the content as needed. An SQLite database facilitates data transfer between systems even when the endpoints have different word sizes and/or byte orders. The data can be a complex mix of large binary blobs, text, and small numeric or boolean values. The data format can be easily extended by adding new tables and/or columns, without breaking legacy receivers. The SQL query language means that receivers are not required to parse the entire transfer all at once, but can instead query the received content as needed. The data format is "transparent" in the sense that it is easily decoded for

human viewing using a variety of universally available, open-source tools,

from multiple vendors



**3.2 ENTITY RELATIONSHIP DIAGRAM**

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business.

An E-R model does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities.

Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity-relationship models.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type.

In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

There is a tradition for ER/data models to be built at two or three levels of

abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering. While useful for organizing data that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semistructured or unstructured data, and an ER Diagram is unlikely to be helpful on its own in integrating data into a pre-existing information system.

Cardinality notations define the attributes of the relationship between the entities. Cardinalities can denote that an entity is optional.

**ENTITY RELATIONSHIP DIAGRAM :**

|  |  |
| --- | --- |
|  | **BOOK\_ISSUED** |
| ***FK*** | BID |
|  | ISSUE\_ID |

|  |  |
| --- | --- |
|  | **BOOKS** |
| ***PK*** | BID |
|  | TITLE |
|  | AUTHOR |
|  | STATUS |

Fig : Enhanced ER diagram of Library Management System

**ADMIN**

**ISSUE TO**

**BOOKS**

**BOOK\_ISSUED**

**3.3 RELATIONAL SCHEMA**

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. A relational schema shows references among fields in the database. When a primary key is referenced in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing at the referenced key attribute. A schema diagram helps organize values in the database. The following diagram shows the schema diagram for the database.

BOOKS :

|  |  |  |  |
| --- | --- | --- | --- |
| **BID** | **TITLE** | **AUTHOR** | **STATUS** |

BOOK\_ISSUED :

|  |  |
| --- | --- |
| **BID** | **ISSUE\_ID** |

**3.4 DESCRIPTION OF TABLES**

The database consists of 2 tables:

**1.BOOK: It stores book details**

* *BID: Unique ID of a Book*
* *TITLE: Title of a Book*
* *AUTHOR: The name of a person who written the Book*
* *STATUS: It shows that book is available or issued*

**2.ISSUEBOOK: It stores the details of book**

**issued**

* *BID: Issued Book ID*
* *ISSUED\_ID: Borrower Library Member ID*

**4.IMPLEMENTATION**

**4.1 MODULES AND THEIR ROLES**

*Before seeing the modules and their roles we have to install few modules*

**1. tkinter 2. sqlite3**

*To install these use following commands in terminal(cmd)*

* pip install sqlite3
* pip install tkinter

**4.1.1 MAIN PAGE :**

**from tkinter import \***

**from PIL import ImageTk,Image**

**import sqlite3**

**from tkinter import messagebox**

**from AddBook import \***

**from DeleteBook import \***

**from ViewBooks import \***

**from IssueBook import \***

**from ReturnBook import \***

**from ViewIssueBook import \***

**con = sqlite3.connect("librarydb.sqlite")**

**cur = con.cursor()**

**root = Tk()**

**root.title("Library Management System (Developed by Yashwanth and Prateek)")**

**root.maxsize(width=800,height=700)**

**root.geometry("600x500")**

**# Take n greater than 0.25 and less than 5**

**same=True**

**n=2**

**# Adding a background image**

**background\_image =Image.open("lib.jpg")**

**[imageSizeWidth, imageSizeHeight] = background\_image.size**

**newImageSizeWidth = int(imageSizeWidth\*n)**

**if same:**

**newImageSizeHeight = int(imageSizeHeight\*n)**

**else:**

**newImageSizeHeight = int(imageSizeHeight/n)**

**background\_image = background\_image.resize((newImageSizeWidth,newImageSizeHeight),Image.ANTIALIAS)**

**img = ImageTk.PhotoImage(background\_image)**

**Canvas1 = Canvas(root)**

**Canvas1.create\_image(300,340,image = img)**

**Canvas1.config(bg="white",width = newImageSizeWidth, height = newImageSizeHeight)**

**Canvas1.pack(expand=True,fill=BOTH)**

**headingFrame1 = Frame(root,bg="#FFBB00",bd=5)**

**headingFrame1.place(relx=0.2,rely=0.1,relwidth=0.6,relheight=0.16)**

**headingLabel = Label(headingFrame1, text="Welcome to KSIT \n Library", bg='YELLOW', fg='RED', font=('Courier',15))**

**headingLabel.place(relx=0,rely=0, relwidth=1, relheight=1)**

**btn1 = Button(root,text="Add Book Details",bg='black', fg='white', command=addBook)**

**btn1.place(relx=0.28,rely=0.4, relwidth=0.45,relheight=0.1)**

**btn2 = Button(root,text="Delete Book",bg='black', fg='white', command=delete)**

**btn2.place(relx=0.28,rely=0.5, relwidth=0.45,relheight=0.1)**

**btn3 = Button(root,text="View Book List",bg='black', fg='white', command=View)**

**btn3.place(relx=0.28,rely=0.6, relwidth=0.45,relheight=0.1)**

**btn4 = Button(root,text="Issue Book to Student",bg='black', fg='white', command = issueBook)**

**btn4.place(relx=0.28,rely=0.7, relwidth=0.45,relheight=0.1)**

**btn5 = Button(root,text="Return Book",bg='black', fg='white', command = returnBook)**

**btn5.place(relx=0.28,rely=0.8, relwidth=0.45,relheight=0.1)**

**btn6 = Button(root,text="View Issued Book Details",bg='black', fg='white', command = issueView)**

**btn6.place(relx=0.28,rely=0.9, relwidth=0.45,relheight=0.1)**

**root.mainloop()**

**4.1.2 ADD BOOK DETAILS :**

**from tkinter import \***

**from PIL import ImageTk,Image**

**from tkinter import messagebox**

**import sqlite3**

**def bookRegister():**

**bid = bookInfo1.get()**

**title = bookInfo2.get()**

**author = bookInfo3.get()**

**status = bookInfo4.get()**

**status = status.lower()**

**insertBooks = "insert into "+bookTable+" values('"+bid+"','"+title+"','"+author+"','"+status+"')"**

**try:**

**cur.execute(insertBooks)**

**con.commit()**

**messagebox.showinfo('Success',"Book added successfully")**

**except:**

**messagebox.showinfo("Error","Can't add data into Database")**

**root.destroy()**

**def addBook():**

**global bookInfo1,bookInfo2,bookInfo3,bookInfo4,Canvas1,con,cur,bookTable,root**

**root = Tk()**

**root.title("Library Management System (Developed by Yashwanth and Prateek)")**

**root.minsize(width=400,height=400)**

**root.geometry("600x500")**

**con = sqlite3.connect("librarydb.sqlite")**

**cur = con.cursor()**

**cur.execute('''create table if not exists books(bid varchar(20) primary key, title varchar(30), author varchar(30), status varchar(30));''')**

**# Enter Table Names here**

**bookTable = "books" # Book Table**

**Canvas1 = Canvas(root)**

**Canvas1.config(bg="#ff6e40")**

**Canvas1.pack(expand=True,fill=BOTH)**

**headingFrame1 = Frame(root,bg="#FFBB00",bd=5)**

**headingFrame1.place(relx=0.25,rely=0.1,relwidth=0.5,relheight=0.13)**

**headingLabel = Label(headingFrame1, text="Add Books", bg='black', fg='white', font=('Courier',15))**

**headingLabel.place(relx=0,rely=0, relwidth=1, relheight=1)**

**labelFrame = Frame(root,bg='black')**

**labelFrame.place(relx=0.1,rely=0.4,relwidth=0.8,relheight=0.4)**

**# Book ID**

**lb1 = Label(labelFrame,text="Book ID : ", bg='black', fg='white')**

**lb1.place(relx=0.05,rely=0.2, relheight=0.08)**

**bookInfo1 = Entry(labelFrame)**

**bookInfo1.place(relx=0.3,rely=0.2, relwidth=0.62, relheight=0.08)**

**# Title**

**lb2 = Label(labelFrame,text="Title : ", bg='black', fg='white')**

**lb2.place(relx=0.05,rely=0.35, relheight=0.08)**

**bookInfo2 = Entry(labelFrame)**

**bookInfo2.place(relx=0.3,rely=0.35, relwidth=0.62, relheight=0.08)**

**# Book Author**

**lb3 = Label(labelFrame,text="Author : ", bg='black', fg='white')**

**lb3.place(relx=0.05,rely=0.50, relheight=0.08)**

**bookInfo3 = Entry(labelFrame)**

**bookInfo3.place(relx=0.3,rely=0.50, relwidth=0.62, relheight=0.08)**

**# Book Status**

**lb4 = Label(labelFrame,text="Status(avail/issued) : ", bg='black', fg='white')**

**lb4.place(relx=0.05,rely=0.65, relheight=0.08)**

**bookInfo4 = Entry(labelFrame)**

**bookInfo4.place(relx=0.3,rely=0.65, relwidth=0.62, relheight=0.08)**

**#Submit Button**

**SubmitBtn = Button(root,text="SUBMIT",bg='#d1ccc0', fg='black',command=bookRegister)**

**SubmitBtn.place(relx=0.28,rely=0.9, relwidth=0.18,relheight=0.08)**

**quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)**

**quitBtn.place(relx=0.53,rely=0.9, relwidth=0.18,relheight=0.08)**

**root.mainloop()**

**4.1.3 DELETE BOOK**

**from tkinter import \***

**from PIL import ImageTk,Image**

**from tkinter import messagebox**

**import pymysql**

**import sqlite3**

**con = sqlite3.connect("librarydb.sqlite")**

**cur = con.cursor()**

**# Enter Table Names here**

**issueTable = "books\_issued"**

**bookTable = "books" #Book Table**

**def deleteBook():**

**bid = bookInfo1.get()**

**deleteSql = "delete from "+bookTable+" where bid = '"+bid+"'"**

**deleteIssue = "delete from "+issueTable+" where bid = '"+bid+"'"**

**try:**

**cur.execute(deleteSql)**

**con.commit()**

**cur.execute(deleteIssue)**

**con.commit()**

**messagebox.showinfo('Success',"Book Record Deleted Successfully")**

**except:**

**messagebox.showinfo("Please check Book ID")**

**print(bid)**

**bookInfo1.delete(0, END)**

**root.destroy()**

**def delete():**

**global bookInfo1,bookInfo2,bookInfo3,bookInfo4,Canvas1,con,cur,bookTable,root**

**root = Tk()**

**root.title("Library Management System (Developed by Yashwanth and Prateek)")**

**root.minsize(width=400,height=400)**

**root.geometry("600x500")**

**Canvas1 = Canvas(root)**

**Canvas1.config(bg="#006B38")**

**Canvas1.pack(expand=True,fill=BOTH)**

**headingFrame1 = Frame(root,bg="#FFBB00",bd=5)**

**headingFrame1.place(relx=0.25,rely=0.1,relwidth=0.5,relheight=0.13)**

**headingLabel = Label(headingFrame1, text="Delete Book", bg='black', fg='white', font=('Courier',15))**

**headingLabel.place(relx=0,rely=0, relwidth=1, relheight=1)**

**labelFrame = Frame(root,bg='black')**

**labelFrame.place(relx=0.1,rely=0.3,relwidth=0.8,relheight=0.5)**

**# Book ID to Delete**

**lb2 = Label(labelFrame,text="Book ID : ", bg='black', fg='white')**

**lb2.place(relx=0.05,rely=0.5)**

**bookInfo1 = Entry(labelFrame)**

**bookInfo1.place(relx=0.3,rely=0.5, relwidth=0.62)**

**#Submit Button**

**SubmitBtn = Button(root,text="SUBMIT",bg='#d1ccc0', fg='black',command=deleteBook)**

**SubmitBtn.place(relx=0.28,rely=0.9, relwidth=0.18,relheight=0.08)**

**quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)**

**quitBtn.place(relx=0.53,rely=0.9, relwidth=0.18,relheight=0.08)**

**root.mainloop()**

**4.1.4 VIEW BOOKS LIST**

**from tkinter import \***

**from PIL import ImageTk,Image**

**from tkinter import messagebox**

**import sqlite3**

**con = sqlite3.connect("librarydb.sqlite")**

**cur = con.cursor()**

**# Enter Table Names here**

**bookTable = "books"**

**def View():**

**root = Tk()**

**root.title("Library Management System (Developed by Yashwanth and Prateek)")**

**root.minsize(width=400,height=400)**

**root.geometry("600x500")**

**Canvas1 = Canvas(root)**

**Canvas1.config(bg="#12a4d9")**

**Canvas1.pack(expand=True,fill=BOTH)**

**headingFrame1 = Frame(root,bg="#FFBB00",bd=5)**

**headingFrame1.place(relx=0.25,rely=0.1,relwidth=0.5,relheight=0.13)**

**headingLabel = Label(headingFrame1, text="View Books", bg='black', fg='white', font=('Courier',15))**

**headingLabel.place(relx=0,rely=0, relwidth=1, relheight=1)**

**labelFrame = Frame(root,bg='black')**

**labelFrame.place(relx=0.1,rely=0.3,relwidth=0.8,relheight=0.5)**

**y = 0.25**

**Label(labelFrame, text="%-10s%-40s%-30s%-20s"%('BID','Title','Author','Status'),bg='black',fg='white').place(relx=0.07,rely=0.1)**

**Label(labelFrame, text="----------------------------------------------------------------------------",bg='black',fg='white').place(relx=0.05,rely=0.2)**

**getBooks = "select \* from "+bookTable**

**try:**

**cur.execute(getBooks)**

**con.commit()**

**for i in cur:**

**Label(labelFrame, text="%-10s%-30s%-30s%-20s"%(i[0],i[1],i[2],i[3]),bg='black',fg='white').place(relx=0.07,rely=y)**

**y += 0.1**

**except:**

**messagebox.showinfo("Failed to fetch files from database")**

**quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)**

**quitBtn.place(relx=0.4,rely=0.9, relwidth=0.18,relheight=0.08)**

**root.mainloop()**

**4.1.5 ISSUE BOOK TO STUDENT**

**from tkinter import \***

**from PIL import ImageTk,Image**

**from tkinter import messagebox**

**import sqlite3**

**con = sqlite3.connect("librarydb.sqlite")**

**cur = con.cursor()**

**cur.execute('''create table if not exists books\_issued(bid varchar(20) primary key, issuedto varchar(30))''')**

**# Enter Table Names here**

**issueTable = "books\_issued"**

**bookTable = "books"**

**#List To store all Book IDs**

**allBid = []**

**def issue():**

**global issueBtn,labelFrame,lb1,inf1,inf2,quitBtn,root,Canvas1,status**

**bid = inf1.get()**

**issueto = inf2.get()**

**issueBtn.destroy()**

**labelFrame.destroy()**

**lb1.destroy()**

**inf1.destroy()**

**inf2.destroy()**

**extractBid = "select bid from "+bookTable**

**try:**

**cur.execute(extractBid)**

**con.commit()**

**for i in cur:**

**allBid.append(i[0])**

**if bid in allBid:**

**checkAvail = "select status from "+bookTable+" where bid = '"+bid+"'"**

**cur.execute(checkAvail)**

**con.commit()**

**for i in cur:**

**check = i[0]**

**if check == 'avail':**

**status = True**

**else:**

**status = False**

**else:**

**messagebox.showinfo("Error","Book ID not present")**

**except:**

**messagebox.showinfo("Error","Can't fetch Book IDs")**

**issueSql = "insert into "+issueTable+" values ('"+bid+"','"+issueto+"')"**

**show = "select \* from "+issueTable**

**updateStatus = "update "+bookTable+" set status = 'issued' where bid = '"+bid+"'"**

**try:**

**if bid in allBid and status == True:**

**cur.execute(issueSql)**

**con.commit()**

**cur.execute(updateStatus)**

**con.commit()**

**messagebox.showinfo('Success',"Book Issued Successfully")**

**root.destroy()**

**else:**

**allBid.clear()**

**messagebox.showinfo('Message',"Book Already Issued")**

**root.destroy()**

**return**

**except:**

**messagebox.showinfo("Search Error","The value entered is wrong, Try again")**

**allBid.clear()**

**def issueBook():**

**global issueBtn,labelFrame,lb1,inf1,inf2,quitBtn,root,Canvas1,status**

**root = Tk()**

**root.title("Library Management System (Developed by Yashwanth and Prateek)")**

**root.minsize(width=400,height=400)**

**root.geometry("600x500")**

**Canvas1 = Canvas(root)**

**Canvas1.config(bg="#D6ED17")**

**Canvas1.pack(expand=True,fill=BOTH)**

**headingFrame1 = Frame(root,bg="#FFBB00",bd=5)**

**headingFrame1.place(relx=0.25,rely=0.1,relwidth=0.5,relheight=0.13)**

**headingLabel = Label(headingFrame1, text="Issue Book", bg='black', fg='white', font=('Courier',15))**

**headingLabel.place(relx=0,rely=0, relwidth=1, relheight=1)**

**labelFrame = Frame(root,bg='black')**

**labelFrame.place(relx=0.1,rely=0.3,relwidth=0.8,relheight=0.5)**

**# Book ID**

**lb1 = Label(labelFrame,text="Book ID : ", bg='black', fg='white')**

**lb1.place(relx=0.05,rely=0.2)**

**inf1 = Entry(labelFrame)**

**inf1.place(relx=0.3,rely=0.2, relwidth=0.62)**

**# Issued To Student name**

**lb2 = Label(labelFrame,text="Issued To : ", bg='black', fg='white')**

**lb2.place(relx=0.05,rely=0.4)**

**inf2 = Entry(labelFrame)**

**inf2.place(relx=0.3,rely=0.4, relwidth=0.62)**

**#Issue Button**

**issueBtn = Button(root,text="Issue",bg='#d1ccc0', fg='black',command=issue)**

**issueBtn.place(relx=0.28,rely=0.9, relwidth=0.18,relheight=0.08)**

**quitBtn = Button(root,text="Quit",bg='#aaa69d', fg='black', command=root.destroy)**

**quitBtn.place(relx=0.53,rely=0.9, relwidth=0.18,relheight=0.08)**

**root.mainloop()**

**4.1.6 RETURN BOOK TO LIBRARAY**

**from tkinter import \***

**from PIL import ImageTk,Image**

**from tkinter import messagebox**

**import sqlite3**

**con = sqlite3.connect("librarydb.sqlite")**

**cur = con.cursor()**

**# Enter Table Names here**

**issueTable = "books\_issued" #Issue Table**

**bookTable = "books" #Book Table**

**allBid = [] #List To store all Book IDs**

**def returnn():**

**global SubmitBtn,labelFrame,lb1,bookInfo1,quitBtn,root,Canvas1,status**

**bid = bookInfo1.get()**

**extractBid = "select bid from "+issueTable**

**try:**

**cur.execute(extractBid)**

**con.commit()**

**for i in cur:**

**allBid.append(i[0])**

**if bid in allBid:**

**checkAvail = "select status from "+bookTable+" where bid = '"+bid+"'"**

**cur.execute(checkAvail)**

**con.commit()**

**for i in cur:**

**check = i[0]**

**if check == 'issued':**

**status = True**

**else:**

**status = False**

**else:**

**messagebox.showinfo("Error","Book ID not present")**

**except:**

**messagebox.showinfo("Error","Can't fetch Book IDs")**

**issueSql = "delete from "+issueTable+" where bid = '"+bid+"'"**

**print(bid in allBid)**

**print(status)**

**updateStatus = "update "+bookTable+" set status = 'avail' where bid = '"+bid+"'"**

**try:**

**if bid in allBid and status == True:**

**cur.execute(issueSql)**

**con.commit()**

**cur.execute(updateStatus)**

**con.commit()**

**messagebox.showinfo('Success',"Book Returned Successfully")**

**else:**

**allBid.clear()**

**messagebox.showinfo('Message',"Please check the book ID")**

**root.destroy()**

**return**

**except:**

**messagebox.showinfo("Search Error","The value entered is wrong, Try again")**

**allBid.clear()**

**root.destroy()**

**def returnBook():**

**global bookInfo1,SubmitBtn,quitBtn,Canvas1,con,cur,root,labelFrame, lb1**

**root = Tk()**

**root.title("Library Management System (Developed by Yashwanth and Prateek)")**

**root.minsize(width=400,height=400)**

**root.geometry("600x500")**

**Canvas1 = Canvas(root)**

**Canvas1.config(bg="#006B38")**

**Canvas1.pack(expand=True,fill=BOTH)**

**headingFrame1 = Frame(root,bg="#FFBB00",bd=5)**

**headingFrame1.place(relx=0.25,rely=0.1,relwidth=0.5,relheight=0.13)**

**headingLabel = Label(headingFrame1, text="Return Book", bg='black', fg='white', font=('Courier',15))**

**headingLabel.place(relx=0,rely=0, relwidth=1, relheight=1)**

**labelFrame = Frame(root,bg='black')**

**labelFrame.place(relx=0.1,rely=0.3,relwidth=0.8,relheight=0.5)**

**# Book ID to Delete**

**lb1 = Label(labelFrame,text="Book ID : ", bg='black', fg='white')**

**lb1.place(relx=0.05,rely=0.5)**

**bookInfo1 = Entry(labelFrame)**

**bookInfo1.place(relx=0.3,rely=0.5, relwidth=0.62)**

**#Submit Button**

**SubmitBtn = Button(root,text="Return",bg='#d1ccc0', fg='black',command=returnn)**

**SubmitBtn.place(relx=0.28,rely=0.9, relwidth=0.18,relheight=0.08)**

**quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)**

**quitBtn.place(relx=0.53,rely=0.9, relwidth=0.18,relheight=0.08)**

**root.mainloop()**

**4.1.7 VIEW ISSUED BOOK DETAILS**

**from tkinter import \***

**from PIL import ImageTk,Image**

**from tkinter import messagebox**

**import sqlite3**

**con = sqlite3.connect("librarydb.sqlite")**

**cur = con.cursor()**

**# Enter Table Names here**

**bookTable = "books\_issued"**

**def issueView():**

**root = Tk()**

**root.title("Library Management System (Developed by Yashwanth and Prateek)")**

**root.minsize(width=400,height=400)**

**root.geometry("600x500")**

**Canvas1 = Canvas(root)**

**Canvas1.config(bg="#897756")**

**Canvas1.pack(expand=True,fill=BOTH)**

**headingFrame1 = Frame(root,bg="#FFBB00",bd=5)**

**headingFrame1.place(relx=0.25,rely=0.1,relwidth=0.5,relheight=0.13)**

**headingLabel = Label(headingFrame1, text="View Issued Books", bg='black', fg='white', font=('Courier',15))**

**headingLabel.place(relx=0,rely=0, relwidth=1, relheight=1)**

**labelFrame = Frame(root,bg='black')**

**labelFrame.place(relx=0.1,rely=0.3,relwidth=0.8,relheight=0.5)**

**y = 0.25**

**Label(labelFrame, text="%-10s%-40s"%('BID','Issued\_To'),bg='black',fg='white').place(relx=0.07,rely=0.1)**

**Label(labelFrame, text="----------------------------------------------------------------------------",bg='black',fg='white').place(relx=0.05,rely=0.2)**

**getBooks = "select \* from "+bookTable**

**try:**

**cur.execute(getBooks)**

**con.commit()**

**for i in cur:**

**Label(labelFrame, text="%-10s%-30s"%(i[0],i[1]),bg='black',fg='white').place(relx=0.07,rely=y)**

**y += 0.1**

**except:**

**messagebox.showinfo("Failed to fetch files from database")**

**quitBtn = Button(root,text="Quit",bg='#f7f1e3', fg='black', command=root.destroy)**

**quitBtn.place(relx=0.4,rely=0.9, relwidth=0.18,relheight=0.08)**

**root.mainloop()**

**4.2 RESULT**

*The resulting system is able to:*

* *Allows to Add book details*
* *Allows to Delete book details*
* *Allows to View book List*
* *Allows to issue book to library student*
* *Allows to View issue book details*
* *Allows to return book back*

**5 TESTING**

**5.1 SOFTWARE TESTING**

Testing is the process used to help identify correctness, completeness, security and quality of developed software. This includes executing a program with the intent of finding errors. It is important to distinguish between faults and failures. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. It can be conducted as soon as executable software (even if partially complete) exists. Most testing occurs after system requirements have been defined and then implemented in testable programs.

**5.2 MODULE TESTING AND INTEGRATION**

Module testing is a process of testing the individual subprograms, subroutines, classes, or procedures in a program. Instead of testing whole software program at once, module testing recommend testing the smaller building blocks of the program. It is largely white box oriented. The objective of doing Module testing is not to demonstrate proper functioning of the module but to demonstrate the presence of an error in the module. Module testing allows implementing of parallelism into the testing process by giving the opportunity to test multiple modules simultaneously.

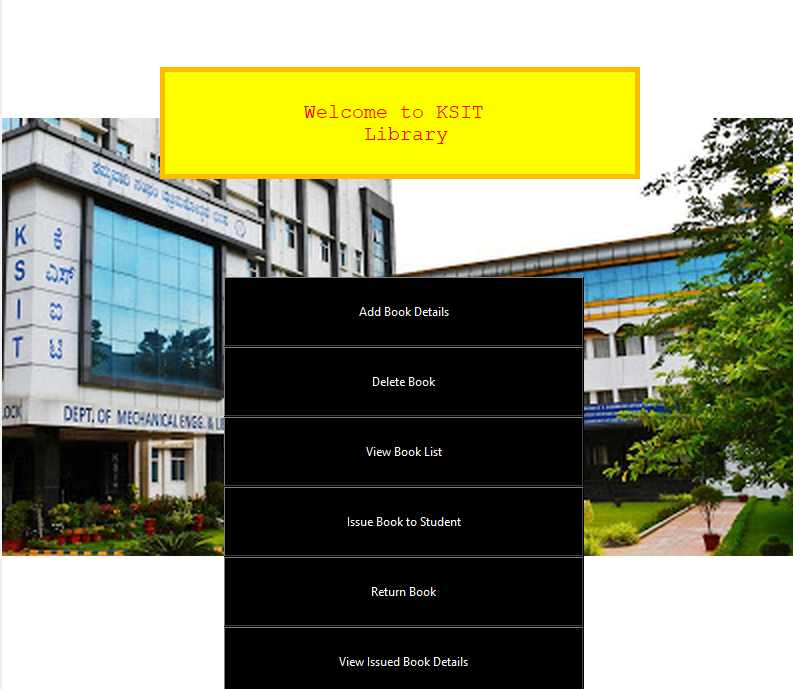
The final integrated system too has been tested for various test cases such as duplicate entries and type mismatch.

**5.3 LIMITATIONS**

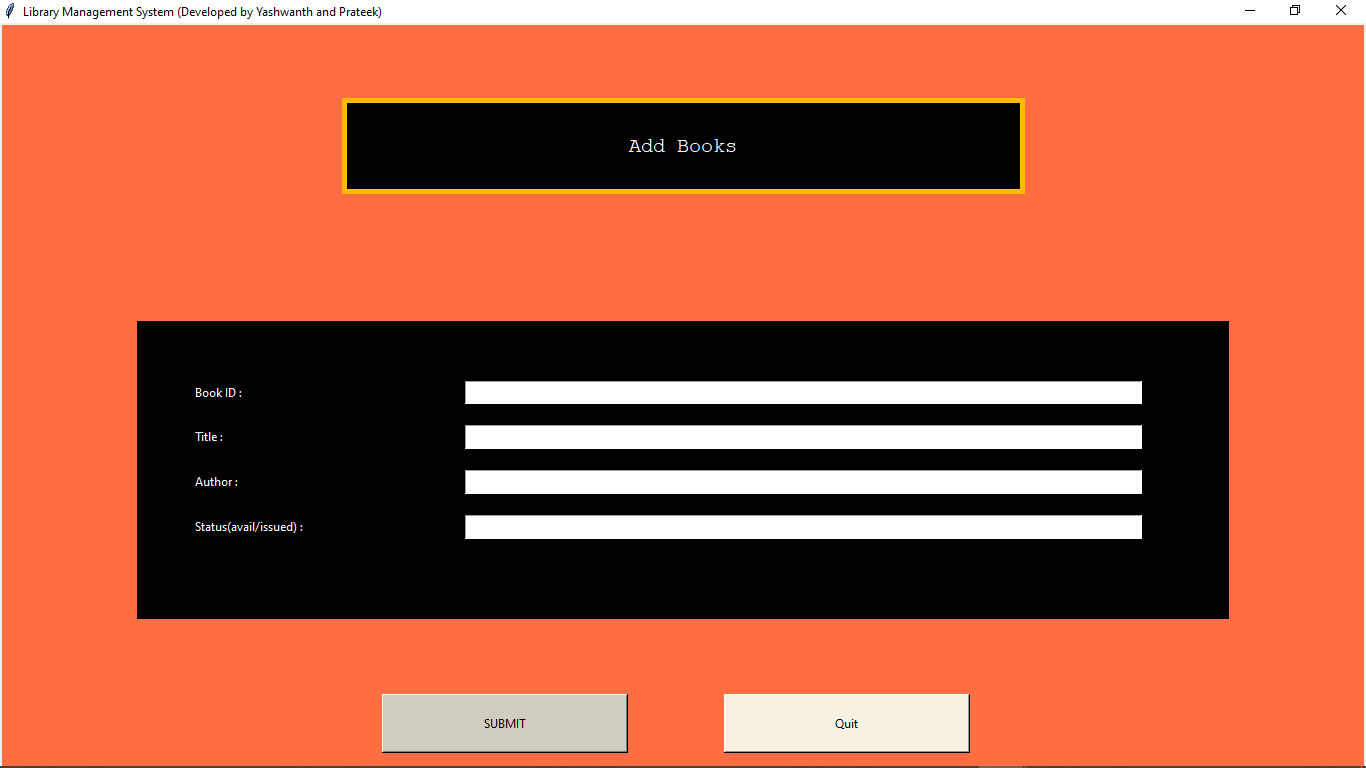
1. Calculation of fine has to be done manually by looking issue date in issue book details
2. Cannot able to sort books based on different domains
3. Risk of computer virus

**6. Snapshots**

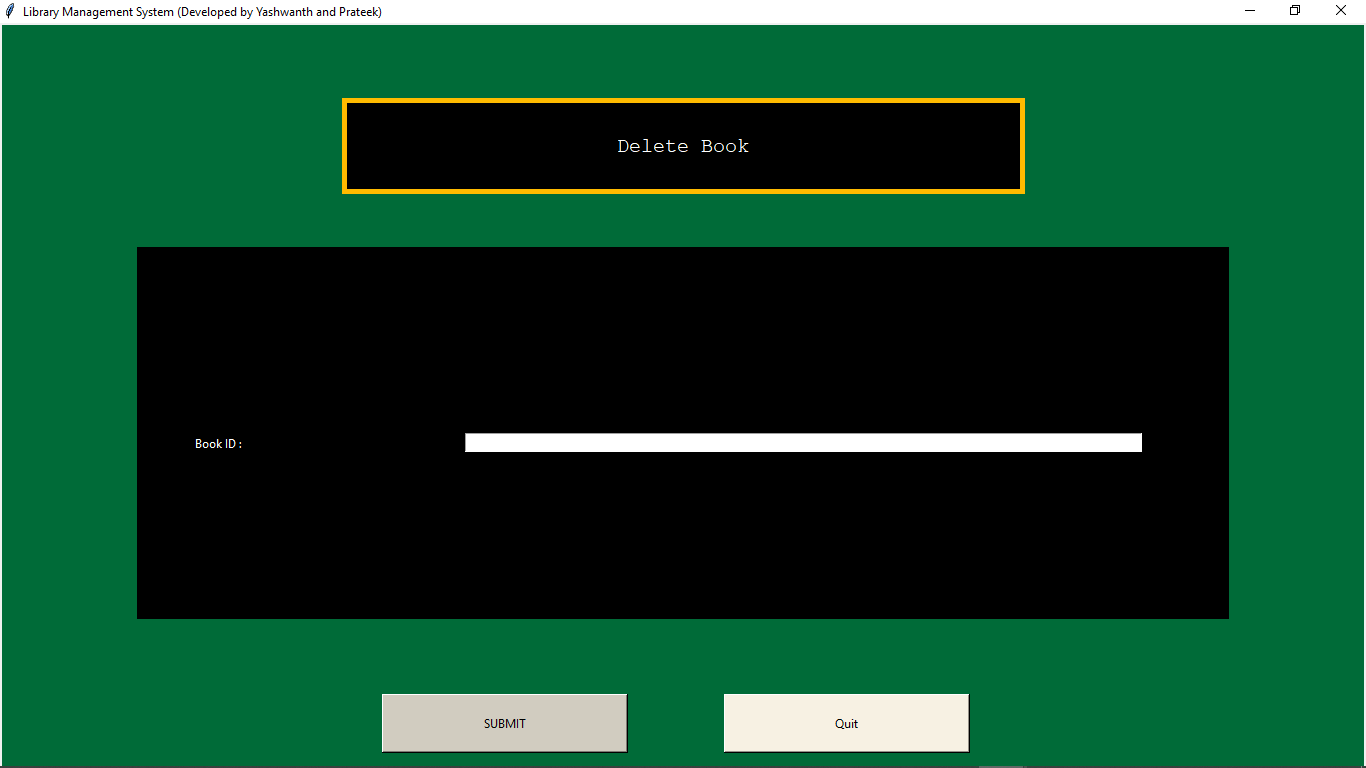
***6.1 MAIN PAGE***

******

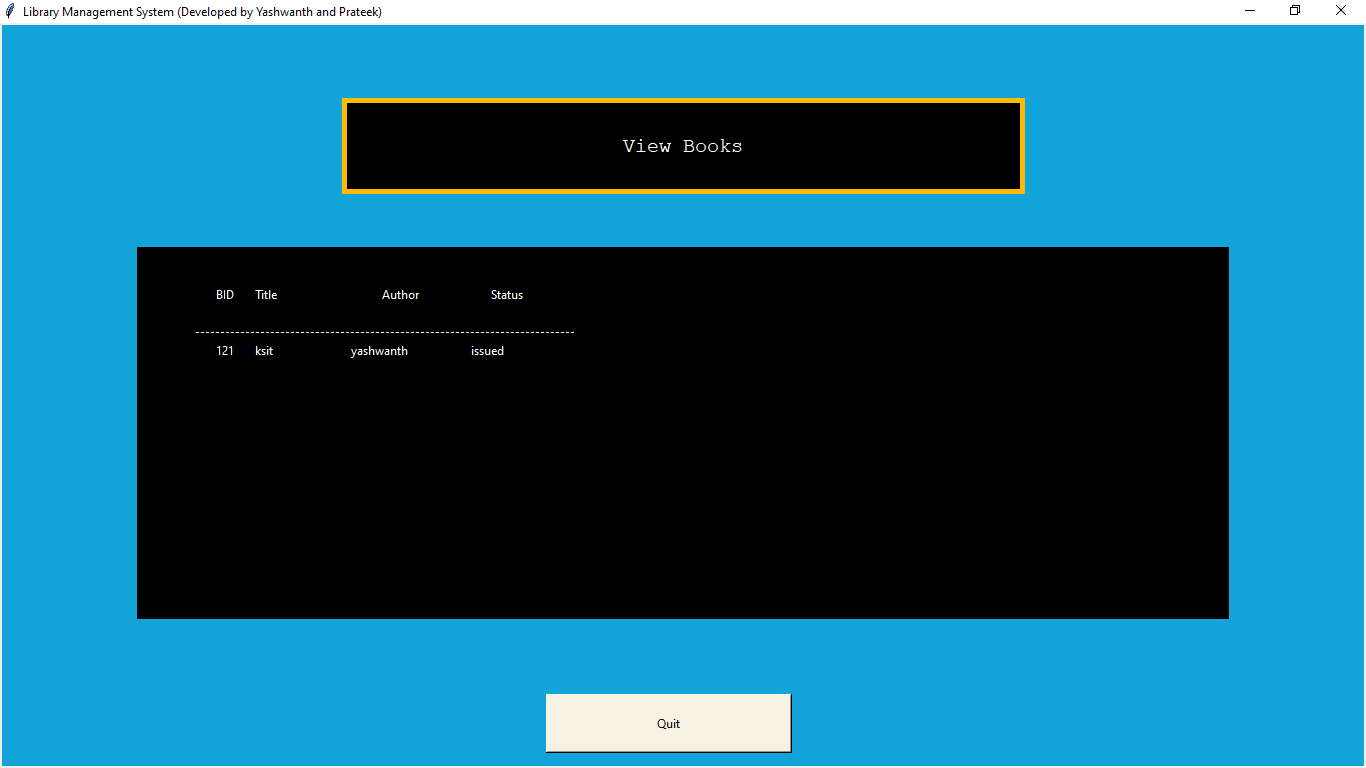
***6.2 DELETE BOOK PAGE***

******

**6.3 DELETE BOOK PAGE**

****

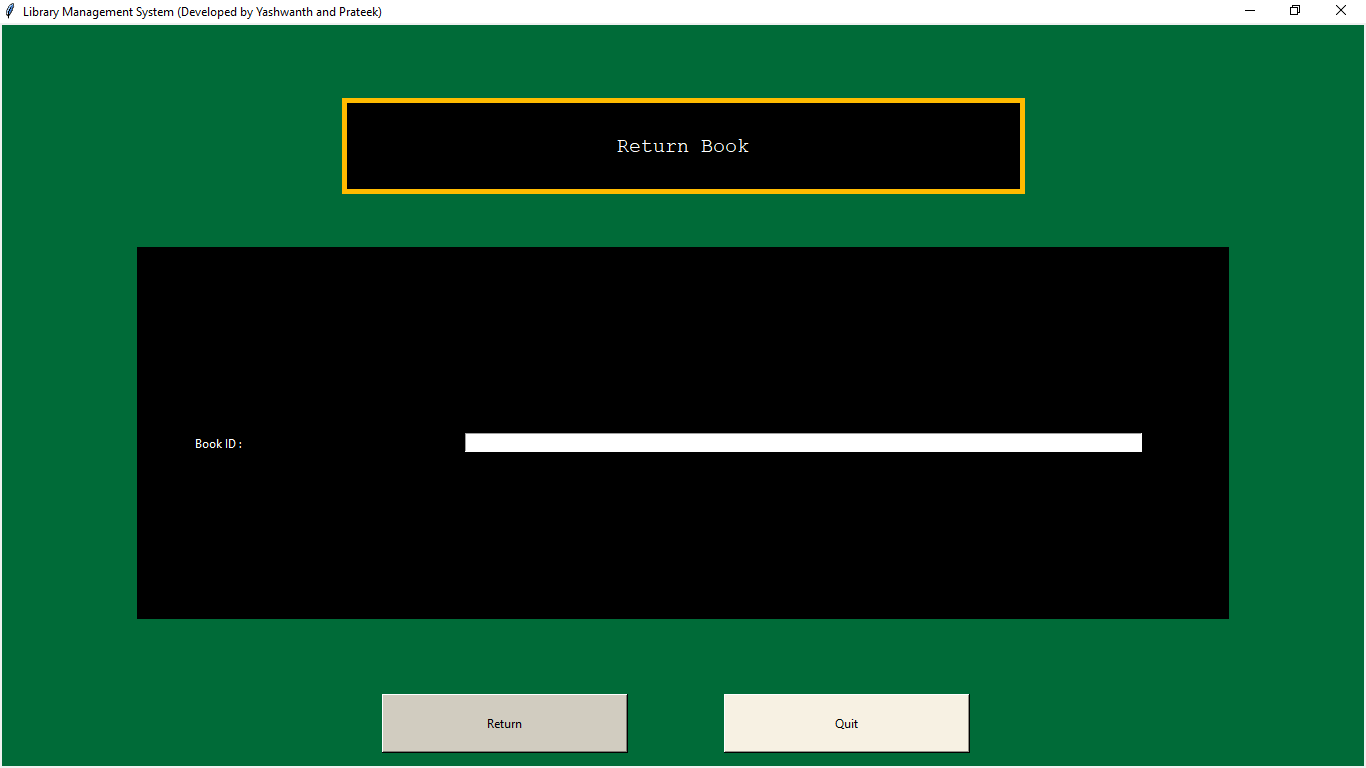
**6.4 VIEW BOOK LIST PAGE**

****

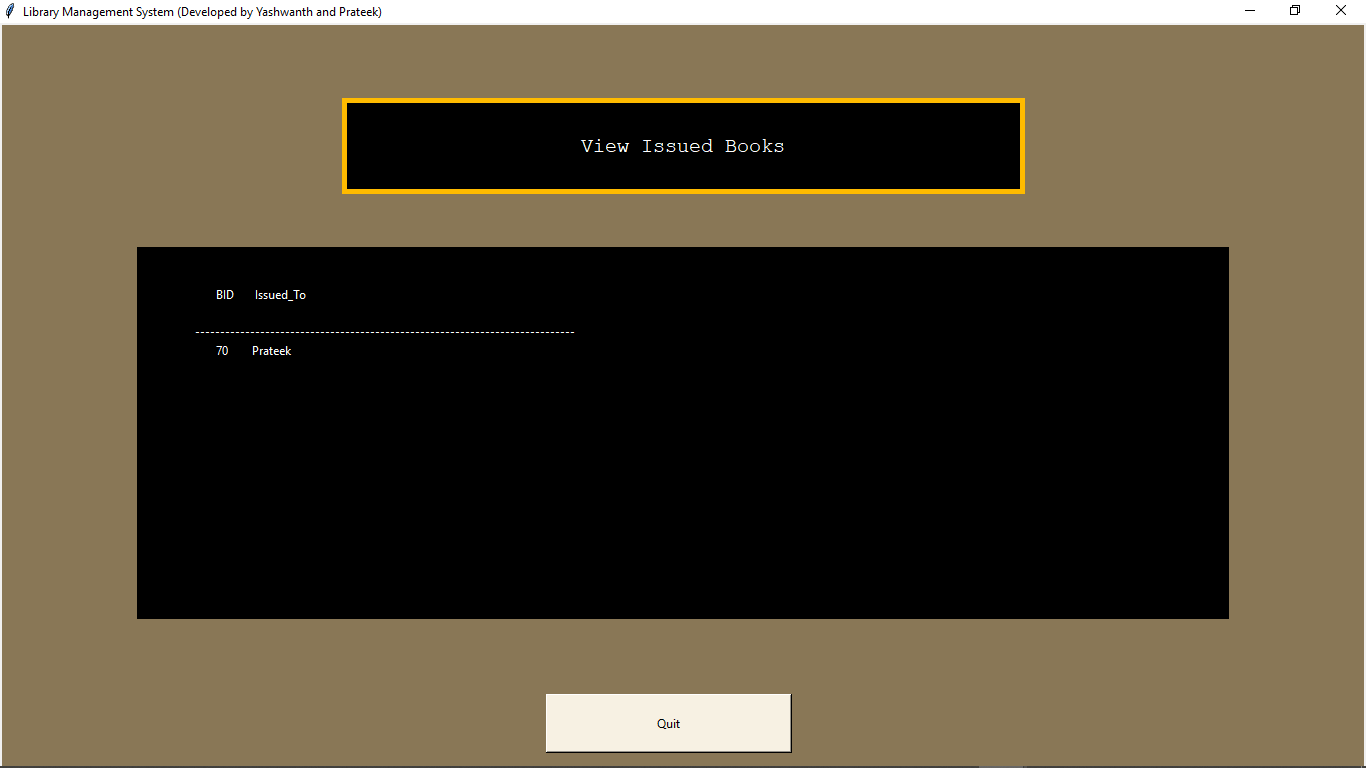
**6.5 ISSUE BOOK TO STUDENT PAGE**

****

**6.6 RETURN BOOK PAGE**

****

**6.7 VIEW ISSUED BOOK DETAILS PAGE**

****

**CONCLUSION**

The Library Management System provides easier maintenance of various books, issue books information. It allows simplified operation and is a time saving platform.

The application has been completed successfully and tested with suitable test cases. It is user friendly. This is developed using Python with tkinter and SQL.

The **goals** achieved by this project are:

* Centralized database
* Easier Maintainance of Records/Details
* User friendly environment
* Efficient management of Library
* Keep track of number of book issued to Students
* Cannot issue same book to person who already borrowed that book

**FUTURE ENHANCEMENTS**

Future **upgrades** to this project will implement:

* Automatically sort the books based on domains or categories
* Calculation of Fines
* login facility
* Setting book limit
* Request to add new book
* Better Front end look

**REFRENCES**

* <https://www.w3schools.com/>
* <https://www.youtube.com/user/edurekaIN>
* <https://www.youtube.com/c/Telusko/playlists>