Shri Vile Parle Kelvani Mandal's

Shri Bhagubhai Mafatlal Polytechnic

Library Management System

Project by:
Sneh Shrimankar 57498210019 (T019)
Atharva Gitaye 57498210027 (T027)
John Bright 57498220061 (T064)

Semester: IV
Department: Information Technology
Course: Programming in Python
Course Code: PRP198918

Acknowledgement

I would like to express and deliver my heartful gratitude and appreciation to all those who have helped and guided me throughout the development of this project and made a meaningful success of this project physically.
It was a great experience during the development of this project due to the hard work and the supportive nature of the faculty which led us towards the great success of this project. Their expertise, experienced and supportive nature guided and shaped our project which played a crucial and valuable role during the making of this project.
I would also like to extend my heartfelt appreciation to my team members. The hard work, dedication, collaboration and collective effort made this project a truly enhancing experience.

Index

1. Problem Statement

To create a Library Management System where Admin can perform various operations like Add Book, Delete Book, Issue book to the customer, Update book details, View books and renter's list and also can print receipt in PDF format.

2. Features

i. Add books to the database
ii. Delete books from the database
iii. Issue book to the customer
iv. Return book from the customer
v. Update book details
vi. View details of all available books
vii. View renter's list with their return status
viii. Print receipt in the PDF format

3. System Requirement Specification

1. Introduction

1.1 Purpose

The purpose of this document is to know how Books from the library are given. Various functionalities of library are mentioned and how the library Management System works. The purpose of this document is to know the basic functionalities of this system and how the system interacts with the admin.

1.2 Document Conventions

This document uses the following conventions:
1. DDB Distributed Database
2. ER Entity Relation

1.3 Indented Audience and Reading Suggestions

This project is a prototype for the Library Management System. This has been implemented under the guidance of a college professor. This project is useful for all owners who have Book Store or a library. This the only prototype model of the system. This prototype model gives us the basic idea of how the bookstore Management system is implemented and gives us the basic theme of the system along with its working and its various functionalities.

1.4 Project Scope

The scope of this project is very high as it reduces the paperwork by a huge amount. As well as it is a user-friendly system. The system is based on a relational database. We will have a database server supporting hundreds of users around the world as well as their activity on this system. Above all, we hope to provide a comfortable user experience along with the best system available. This project is useful for all owners who have Book Store or a library. This the only prototype model of the system.

2. Overall Description

2.1 Product Perspectives

A Library System database stores the following information:
1. Book details
2. Customer details
3. Stock details
4. Rent Book details

5. Update Book details
6. Return Book details
7. Receipt

2.2 User Class and Characteristics
As the project is desktop-based so the main role is of admin instead of a customer. The customer has to just visit the library and rent or buy book. Here the user is admin. The admin has various features that he/she can perform while using this system. The admin has the following features that he can perform:
1. Enter Book details
2. Issue Book to customer
3. Update Book details
4. Delete Book
5. View available books
6. View renter's list
7. Print receipt
8. Return Book

2.3 Design and Implementations Constraints
1. How the response for applications 1 and 2 will be generated. Assuming these are global quire.

2.4 Assumption Dependencies
This is a Library Management System, here the admin will remain same, but customer can also buy book at anytime and anywhere using this project. Here the admin can perform various tasks mentioned in the system.

3. System Features
3.1 Design and Priority
The Library Management System maintains the records of book. Of course, this project has a higher priority as it reduces the paperwork by a huge amount, and as well as there is no fear of losing information or data. This system an also maintains the records of the books given on rent and the details of the customers.

3.2 Stimulus/Response Sequences
1. Enter book details
2. Shows details of book
3. Delete Book
4.Customer rents/buys the book
5. View lists
6. Print Receipt

4. Nonfunctional Requirements
4.1 Performance Requirements
The steps involved to perform the implementation of the Library Management System database are as follows:

4.1.1 ER - Diagram
The E-R Diagram constitutes a technique for representing the logical structure of a database pictorially. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.
1. Entities: Specifies distinct real-world items in an application.
2. Properties / Attributes: Specifies properties of an entity and relationships.
3. Relationships: Connects entities and represents meaningful dependencies between them.

4.1.2 Normalization

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and an increase in the total size of the data stored. If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed, or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. There are three different kinds of modifications of anomalies and formulated the first, second, and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and a complete understanding of its implications.

4.2 Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed-up log, up to the time of failure.

4.3 Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

4.4 Software Quality Attributes

1. Correctness: All the expense and the income amount and date should be registered correctly into the database.
2. Maintainability: The database should be regularly backup through several RAID levels for recovery from failure.
3. Usability: The database should be large enough to store the data of all of its customers who want to use the app.
4. Reliability: The software or the database that we use for storing the data should be reliable in terms of its efficiency, maintainability and should be able to back up the data in case of failure of the system.

4. System Requirements

4.1.1 Hardware Requirements

- 20 GB minimum system storage
- 2 GB minimum RAM

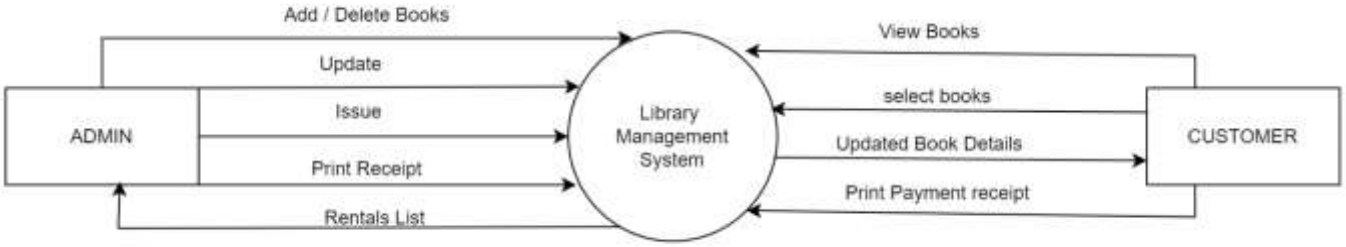
4.1.2 Software Requirements

- Windows 7 or Higher
- Python 3.x installed
- XAMPP Control Panel
- PDF Viewer

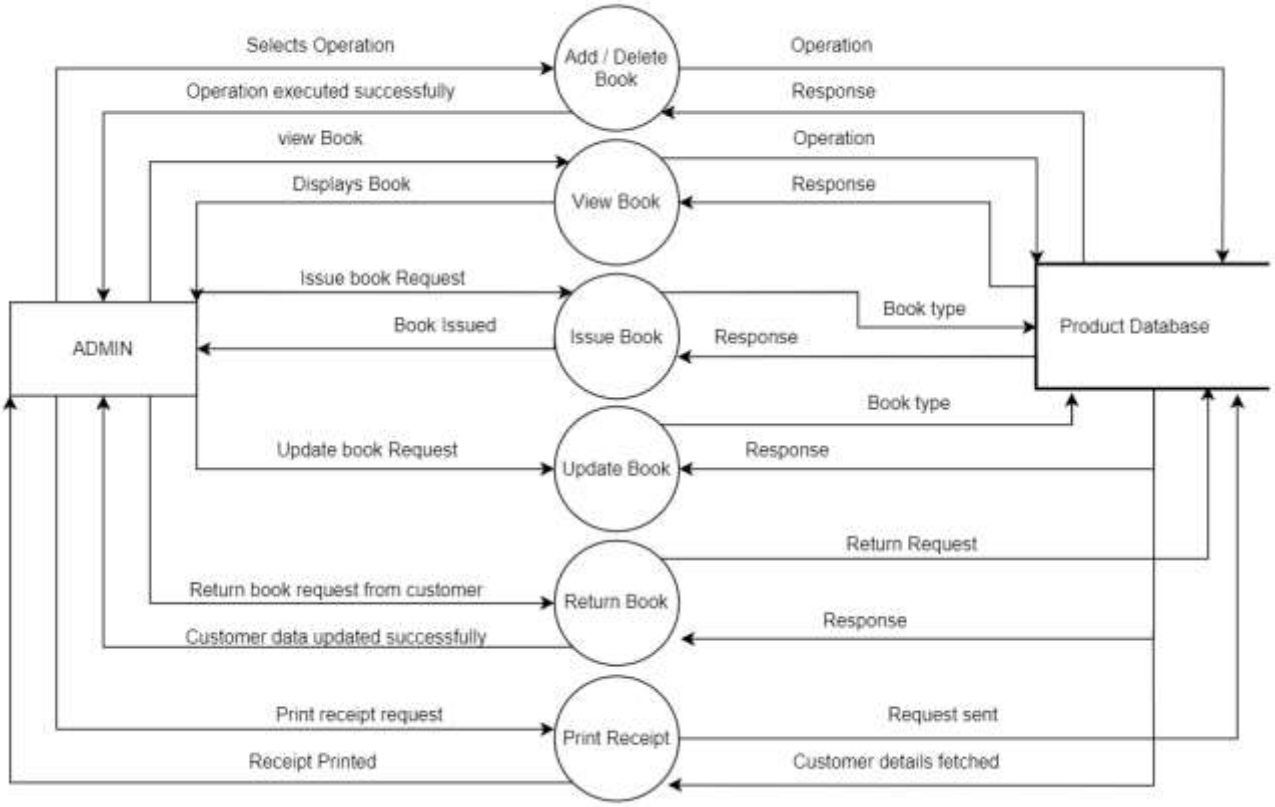
5. System Design

5.1 Data Flow Diagram

5.1.1 DFD Level 0



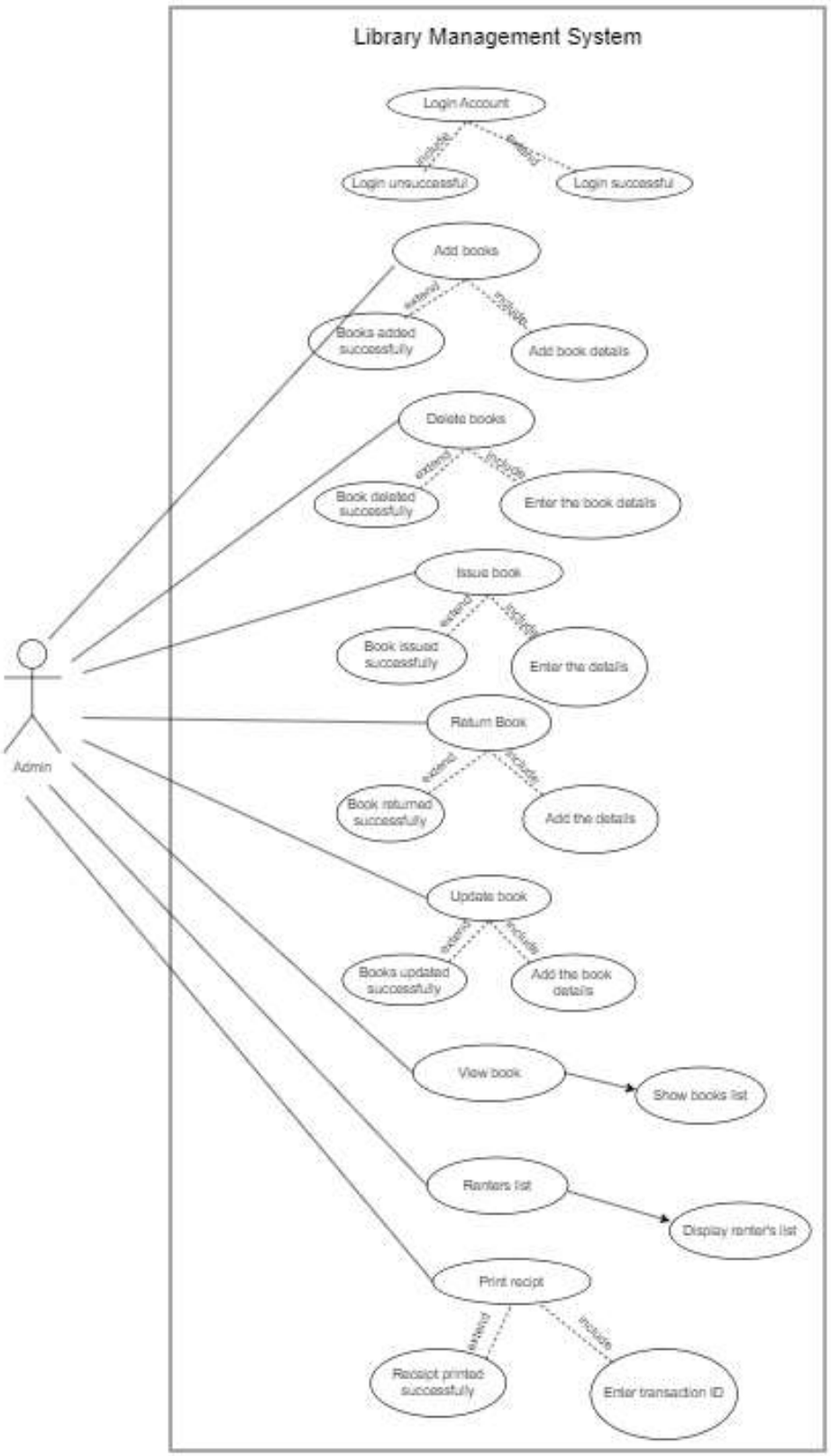
5.1.2 DFD Level 1



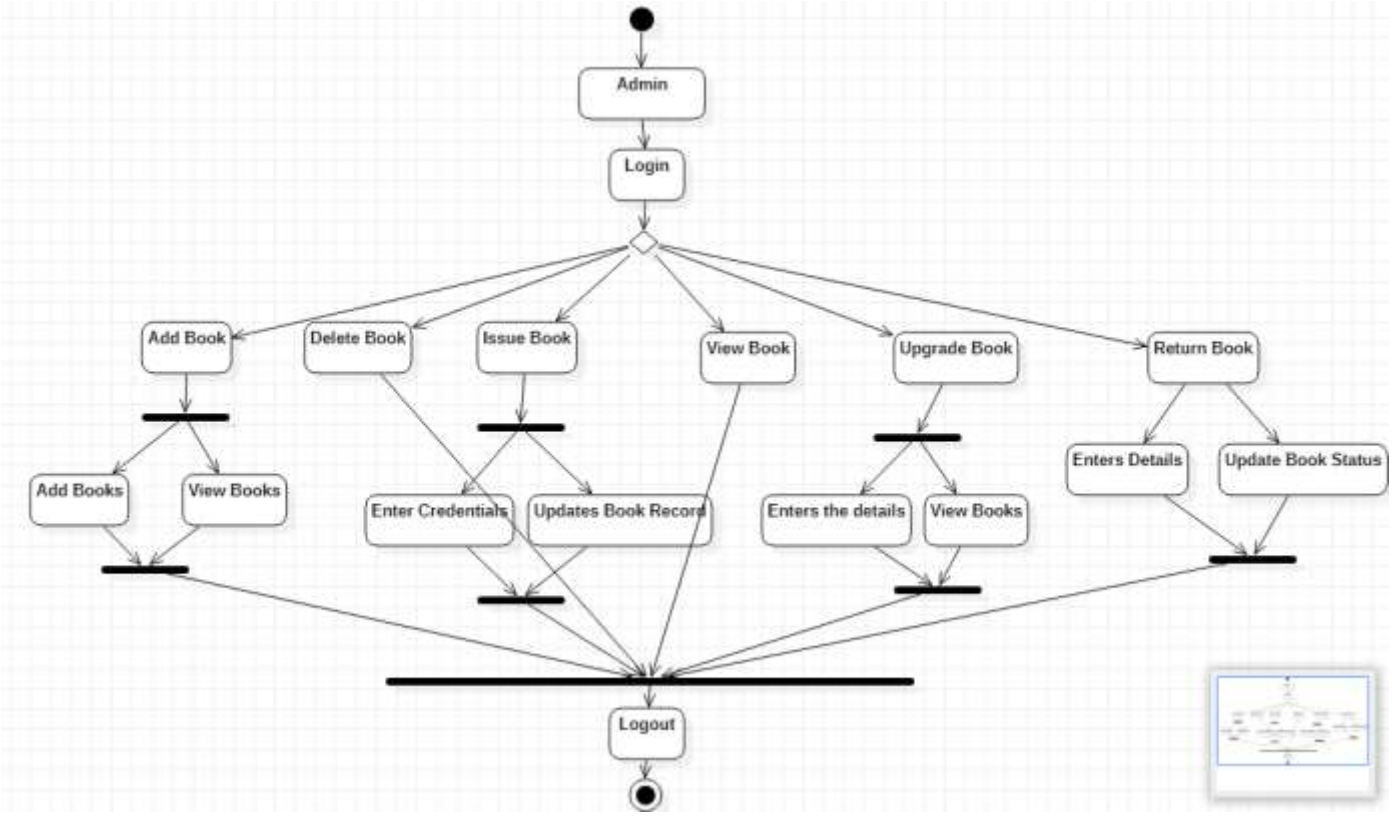
DFD Level 1

5.2 UML Diagrams

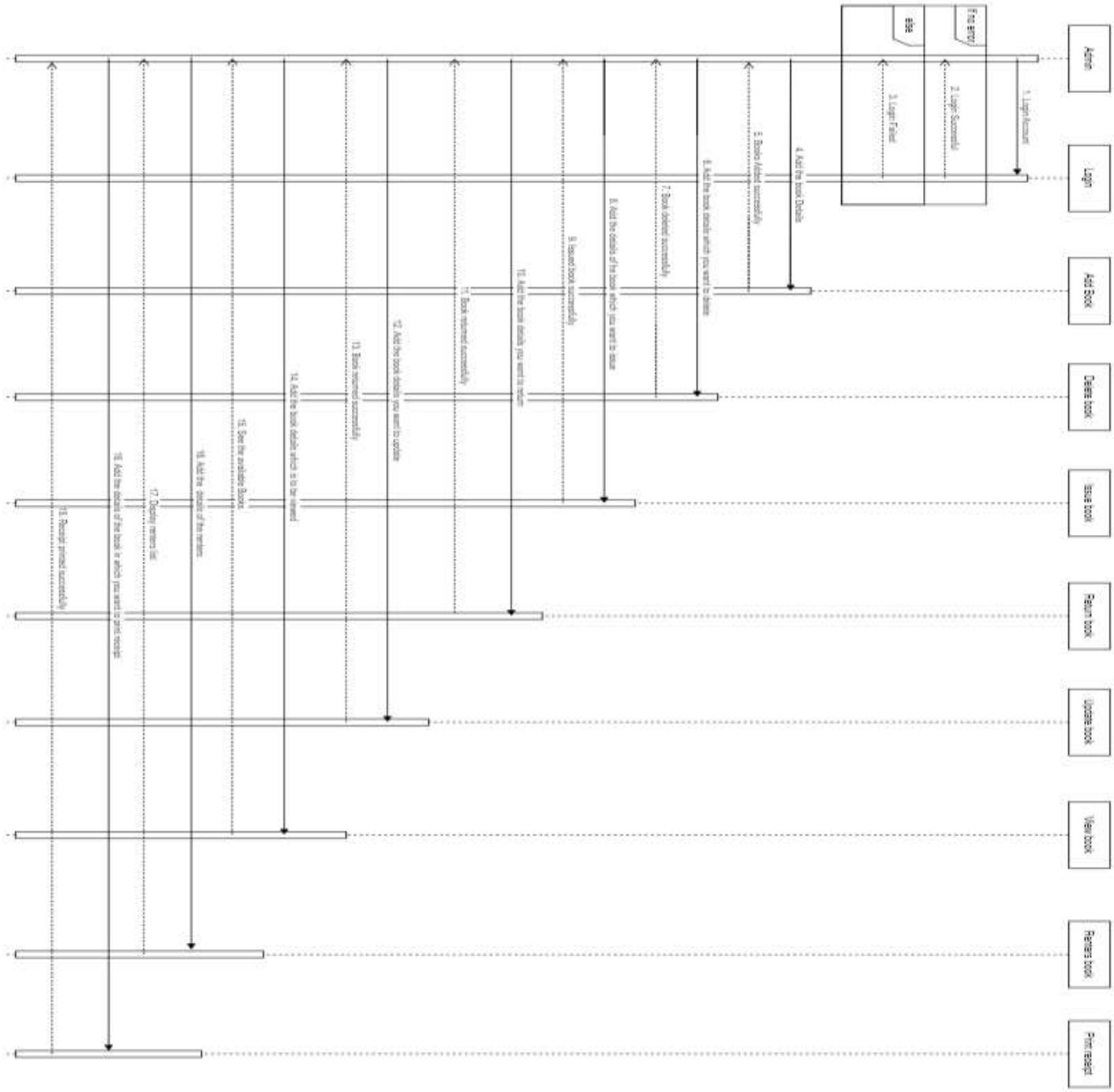
5.2.1 Use Case Diagram



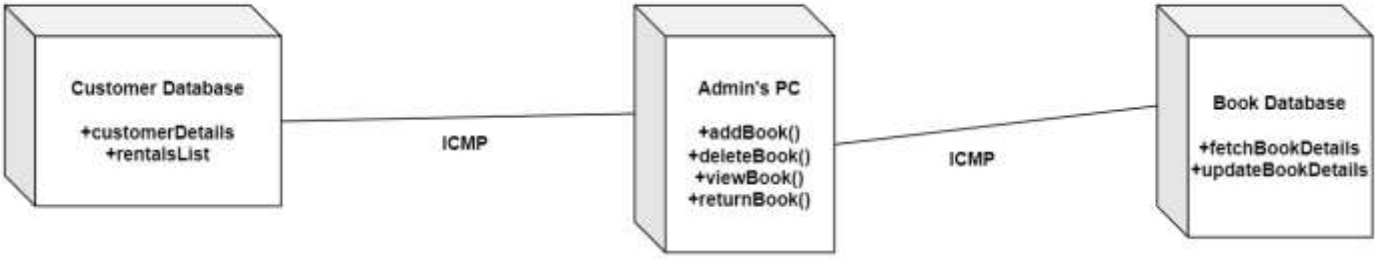
5.2.2 Activity Diagram



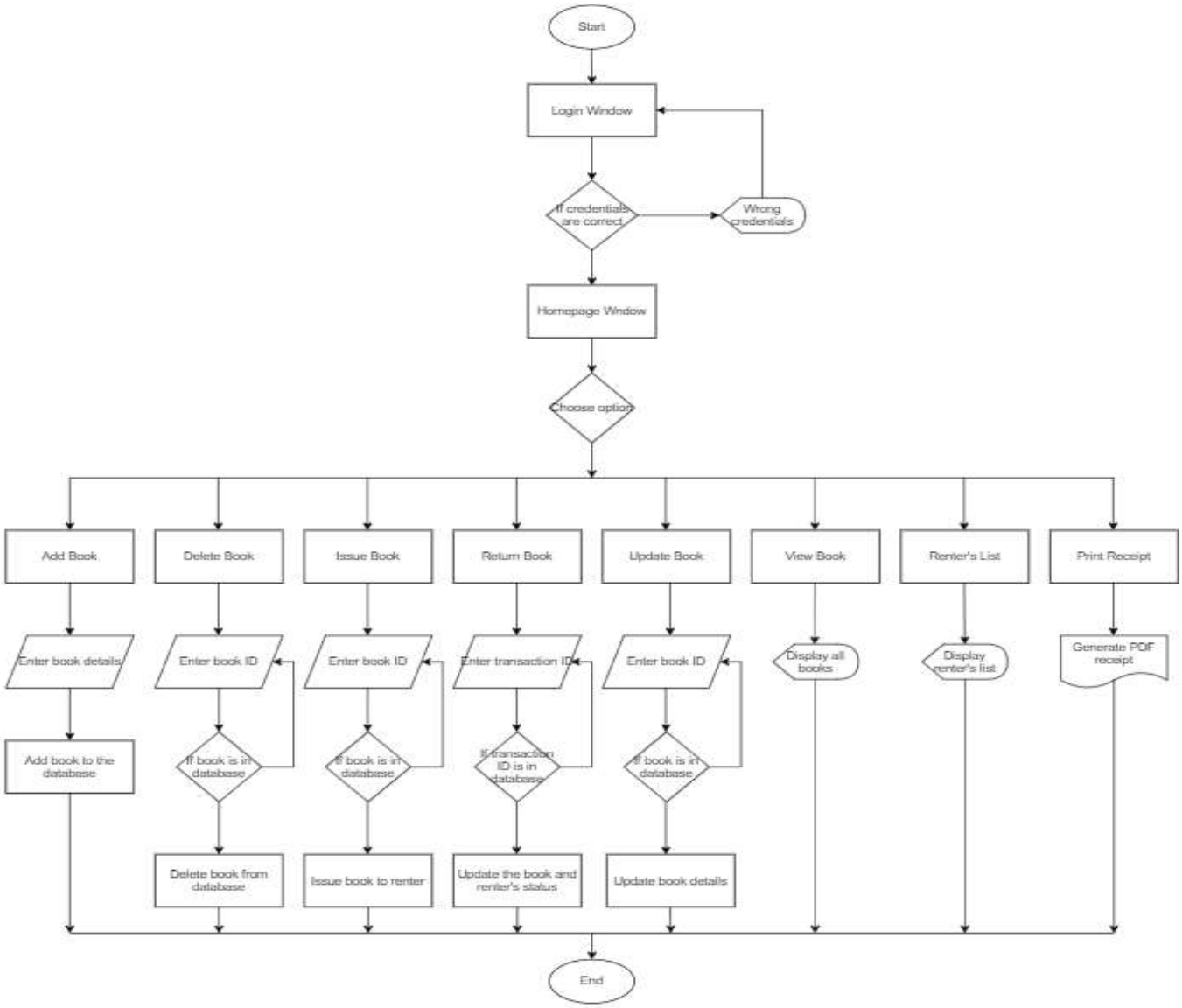
5.2.3 Sequence Diagram



5.2.4 Deployment Diagram



5.3 Flowchart Diagram



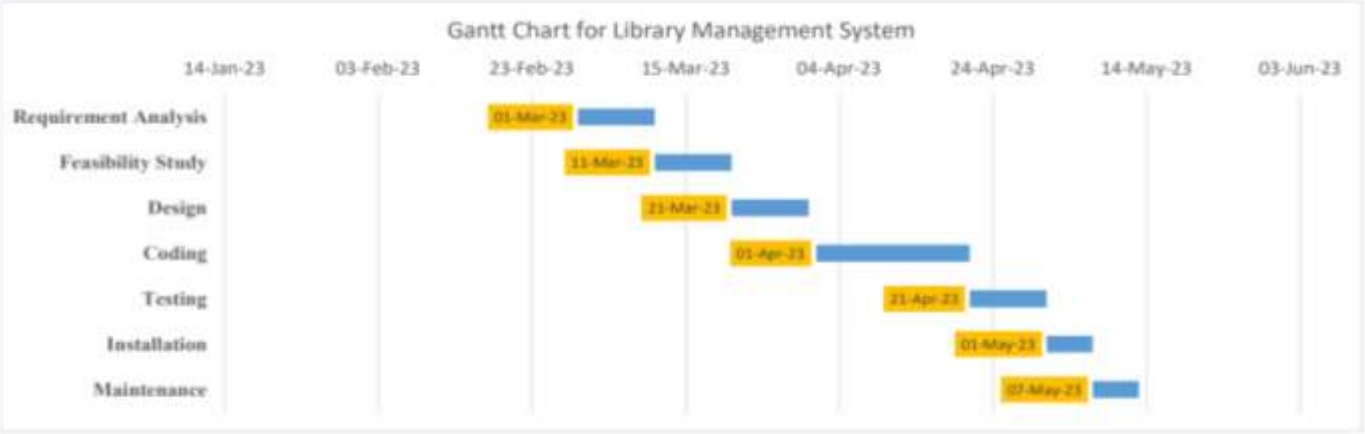
6. Literature Survey

In Library Management System, Literature Survey (new Technologies/Tools learned) you can explore various technologies and tools that can enhance the development and functionality of the system like:

1. GUI (Tkinter)
2. Database Connectivity (MySQL)
3. Various modules
    i. fpdf

7. Gantt Chart

Gantt chart for Library Management System				
SDLC Phase	Project Activity	Start Date	Duration	End Date
Phase 1	Requirement Analysis	01-Mar-23	10	10-Mar-23
Phase 2	Feasibility Study	11-Mar-23	10	20-Mar-23
Phase 3	Design	21-Mar-23	10	31-Mar-23
Phase 4	Coding	01-Apr-23	20	20-Apr-23
Phase 5	Testing	21-Apr-23	10	30-Apr-23
Phase 6	Installation	01-May-23	6	06-May-23
Phase 7	Maintenance	07-May-23	6	12-May-23



---

8. Module Description

Sr. No.	Name	Description	Start Date	End Date
1	Add Book	Adds a book to the database	01 April 2023	03 April 2023
2	Delete Book	Deletes a book from the database	04 April 2023	06 April 2023
3	Issue Book	Issues book to a customer/renter	07 April 2023	09 April 2023
4	Return Book	Allows customer/renter to return book	10 April 2023	12 April 2023
5	Update Book	Updates the book details	13 April 2023	15 April 2023
6	View Books	Displays all the books available in database	16 April 2023	17 April 2023
7	Renter's List	Displays all the renters that have borrowed book	18 April 2023	19 April 2023

8	Print Receipt	Generates a receipt and save it into a PDF format	19 April 2023	20 April 2023

9. System Implementation

main.py

```
from tkinter import *
from tkinter import messagebox
from Functions import *

def verify():
    fnlusr = usrInfo.get()
    fnlpasswd = passwdInfo.get()

    if fnlusr == "Admin" and fnlpasswd == "Admin@123":
        messagebox.showinfo("Successful", "Login Successful")

        lgnwn.destroy()
        hmpage()
    else:
        messagebox.showinfo("Error", "Wrong Credentials")

def hmpage():
    # Creating a window
    win = Tk()
    win.geometry("1280x720")
    win.minsize(width = 1280, height = 720)
    win.title("Library Management System")

    # Giving icon path
    logo = PhotoImage(file = 'Images/Logo.png')

    # Setting icon of master window
    win.iconphoto(False, logo)

    # Creating a title and menu frame
    titleFrame = Frame(win, bg = "#005B96")
    titleFrame.place(relx = 0.0, rely = 0.0, relwidth = 1.0, relheight = 0.15)

    menuFrame = Frame(win, bg = "#6497B1")
    menuFrame.place(relx = 0.0, rely = 0.15, relwidth = 1.0, relheight = 0.85)

    # Creating a title heading
    heading = Label(titleFrame, text = "Welcome to \n Absolute Minds Library", font =
("Barlow", 26, "bold"), bg = "#005B96", fg = "white")
    heading.place(relx = 0.50, rely = 0.50, anchor = CENTER)

    # Creating a menu
    adbk = Button(menuFrame, text = "Add Book", command = AddBook.op, bg = "white", fg =
"blue", font = ("Barlow", 16))
    adbk.place(relx = 0.50, rely = 0.10, relwidth = 0.15, anchor = CENTER)
    dltbk = Button(menuFrame, text = "Delete Book", command = DeleteBook.op, bg = "white",
fg = "blue", font = ("Barlow", 16))
```

```
    dltbk.place(relx = 0.50, rely = 0.20, relwidth = 0.15, anchor = CENTER)
    isbk = Button(menuFrame, text = "Issue Book", command = IssueBook.op, bg = "white", fg
= "blue", font = ("Barlow", 16))
    isbk.place(relx = 0.50, rely = 0.30, relwidth = 0.15, anchor = CENTER)
    rtnbk = Button(menuFrame, text = "Return Book", command = ReturnBook.op, bg = "white",
fg = "blue", font = ("Barlow", 16))
    rtnbk.place(relx = 0.50, rely = 0.40, relwidth = 0.15, anchor = CENTER)
    updtbk = Button(menuFrame, text = "Update Book", command = UpdateBook.op, bg =
"white", fg = "blue", font = ("Barlow", 16))
    updtbk.place(relx = 0.50, rely = 0.50, relwidth = 0.15, anchor = CENTER)
    vwbk = Button(menuFrame, text = "View Books", command = ViewBooks.viewbk, bg =
"white", fg = "blue", font = ("Barlow", 16))
    vwbk.place(relx = 0.50, rely = 0.60, relwidth = 0.15, anchor = CENTER)
    vwbk = Button(menuFrame, text = "Renter's List", command = RenterList.rntslist, bg =
"white", fg = "blue", font = ("Barlow", 16))
    vwbk.place(relx = 0.50, rely = 0.70, relwidth = 0.15, anchor = CENTER)
    prtrcpt = Button(menuFrame, text = "Print Receipt", command = PrintReceipt.op, bg =
"white", fg = "blue", font = ("Barlow", 16))
    prtrcpt.place(relx = 0.50, rely = 0.80, relwidth = 0.15, anchor = CENTER)

    win.mainloop()

# Login Page

global lgnwn, usrInfo, passwdInfo

# Creating a login window
lgnwn = Tk()
lgnwn.geometry("480x300")
lgnwn.title("Absolue Minds Library Login")
lgnwn.resizable(0, 0)

# Giving icon path
logolg = PhotoImage(file = 'Images/Logo.png')

# Setting icon of master window
lgnwn.iconphoto(False, logolg)

# Creating a login page and form frame
ttlFrame = Frame(lgnwn, bg = "#005B96")
ttlFrame.place(relx = 0.0, rely = 0.0, relwidth = 1.0, relheight = 0.15)

mnFrame = Frame(lgnwn, bg = "#6497B1")
mnFrame.place(relx = 0.0, rely = 0.15, relwidth = 1.0, relheight = 0.85)

# Creating a title heading
heading = Label(ttlFrame, text = "Login", font = ("Barlow", 20, "bold"), bg = "#005B96",
fg = "white")
heading.place(relx = 0.50, rely = 0.50, anchor = CENTER)

# Creating a form
usr = Label(mnFrame, text = "Username: ", bg = "#6497B1", fg = "white", font = ("Barlow",
10))
usr.place(relx = 0.40, rely = 0.10, anchor = CENTER)
```

```
usrInfo = Entry(mnFrame)
usrInfo.place(relx = 0.60, rely = 0.10, anchor = CENTER)

passwd = Label(mnFrame, text = "Password: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
passwd.place(relx = 0.40, rely = 0.25, anchor = CENTER)
passwdInfo = Entry(mnFrame, show = "*")
passwdInfo.place(relx = 0.60, rely = 0.25, anchor = CENTER)

lgBtn = Button(mnFrame, text = "Login", font = ("Barlow", 10), command = verify)
lgBtn.place(relx = 0.50, rely = 0.40, anchor = CENTER)

lgnwn.mainloop()
```

AddBook.py

```
from tkinter import *
from tkinter import messagebox
from PIL import *
from mysql.connector import *

def adbkdb():
    bname = nameInfo.get()
    authname = authInfo.get()
    pubname = pubInfo.get()
    isbnname = isbnInfo.get()
    rntprcdb = rntprcInfo.get()
    ttlqtydb = ttlqtyInfo.get()

    insertBook = f"INSERT INTO `book_list_master` (`book_id`, `book_name`, `book_author`,
`book_pub`, `book_isbn`, `book_rent_price`, `book_total_qty`, `book_avlb_qty`,
`book_status`) VALUES ('', '{bname}', '{authname}', '{pubname}', '{isbnname}',
'{rntprcdb}', '{ttlqtydb}', '{ttlqtydb}', '1');"

    try:
        cur.execute(insertBook)
        con.commit()
        messagebox.showinfo("Successful", "Book added successfully")
    except:
        messagebox.showinfo("Error", "Can't add book")

    win.destroy()

def op():
    global win, con, cur, nameInfo, authInfo, pubInfo, isbnInfo, rntprcInfo, ttlqtyInfo

    # Adding database details
    con = connect(host = "localhost", user = "root", password = "", database =
"lib_mgmt_sys")
    cur = con.cursor()
```

```
    # Creating a window
    win = Tk()
    win.title("Library Management System - Add Book")
    win.geometry("800x480")
    win.minsize(width = 800, height = 480)

    # Creating a title and menu frame
    titleFrame = Frame(win, bg = "#005B96")
    titleFrame.place(relx = 0.0, rely = 0.0, relwidth = 1.0, relheight = 0.15)

    menuFrame = Frame(win, bg = "#6497B1")
    menuFrame.place(relx = 0.0, rely = 0.15, relwidth = 1.0, relheight = 0.85)

    # Creating a title heading
    heading = Label(titleFrame, text = "Add Book", font = ("Barlow", 20, "bold"), bg =
"#005B96", fg = "white")
    heading.place(relx = 0.50, rely = 0.50, anchor = CENTER)

    # Creating a form
    name = Label(menuFrame, text = "Name: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    name.place(relx = 0.40, rely = 0.10, anchor = CENTER)
    nameInfo = Entry(menuFrame)
    nameInfo.place(relx = 0.60, rely = 0.10, anchor = CENTER)

    auth = Label(menuFrame, text = "Author: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    auth.place(relx = 0.40, rely = 0.20, anchor = CENTER)
    authInfo = Entry(menuFrame)
    authInfo.place(relx = 0.60, rely = 0.20, anchor = CENTER)

    pub = Label(menuFrame, text = "Publisher: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    pub.place(relx = 0.40, rely = 0.30, anchor = CENTER)
    pubInfo = Entry(menuFrame)
    pubInfo.place(relx = 0.60, rely = 0.30, anchor = CENTER)

    isbn = Label(menuFrame, text = "ISBN: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    isbn.place(relx = 0.40, rely = 0.40, anchor = CENTER)
    isbnInfo = Entry(menuFrame)
    isbnInfo.place(relx = 0.60, rely = 0.40, anchor = CENTER)

    rntprc = Label(menuFrame, text = "Rent Price: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    rntprc.place(relx = 0.40, rely = 0.50, anchor = CENTER)
    rntprcInfo = Entry(menuFrame)
    rntprcInfo.place(relx = 0.60, rely = 0.50, anchor = CENTER)
```

```
    ttlqty = Label(menuFrame, text = "Total Quantity: ", bg = "#6497B1", fg = "white",
font = ("Barlow", 10))
    ttlqty.place(relx = 0.40, rely = 0.60, anchor = CENTER)
    ttlqtyInfo = Entry(menuFrame)
    ttlqtyInfo.place(relx = 0.60, rely = 0.60, anchor = CENTER)

    SubmitBtn = Button(menuFrame, text = "Add Book", font = ("Barlow", 10), command =
adbkdb)
    SubmitBtn.place(relx = 0.50, rely = 0.75, anchor = CENTER)

    win.mainloop()
```

DeleteBook.py

```
from tkinter import *
from tkinter import messagebox
from mysql.connector import *
from Functions.ViewBooks import *

def dltbkdb():
    bid = idInfo.get()

    deleteBook = f"DELETE FROM book_list_master WHERE book_id = {bid};"

    try:
        cur.execute(deleteBook)
        con.commit()
        messagebox.showinfo("Successful", "Book deleted successfully")
    except:
        messagebox.showinfo("Error", "Some error has been occured \nCan't delete book")

    win.destroy()

def op():
    global win, con, cur, idInfo

    # Adding database details
    con = connect(host = "localhost", user = "root", password = "", database =
"lib_mgmt_sys")
    cur = con.cursor()

    # Creating a window
    win = Tk()
    win.title("Library Management System - Delete Book")
    win.geometry("800x480")
    win.minsize(width = 800, height = 480)

    # Creating a title and menu frame
    titleFrame = Frame(win, bg = "#005B96")
    titleFrame.place(relx = 0.0, rely = 0.0, relwidth = 1.0, relheight = 0.15)
```

```
    menuFrame = Frame(win, bg = "#6497B1")
    menuFrame.place(relx = 0.0, rely = 0.15, relwidth = 1.0, relheight = 0.85)

    # Creating a title heading
    heading = Label(titleFrame, text = "Delete Book", font = ("Barlow", 20, "bold"), bg =
"#005B96", fg = "white")
    heading.place(relx = 0.50, rely = 0.50, anchor = CENTER)

    # Creating a form
    id = Label(menuFrame, text = "ID: ", bg = "#6497B1", fg = "white", font = ("Barlow",
10))
    id.place(relx = 0.40, rely = 0.10, anchor = CENTER)
    idInfo = Entry(menuFrame)
    idInfo.place(relx = 0.60, rely = 0.10, anchor = CENTER)

    ViewBooks = Button(menuFrame, text = "View Books", font = ("Barlow", 10), command =
viewbk)
    ViewBooks.place(relx = 0.40, rely = 0.25, anchor = CENTER)

    SubmitBtn = Button(menuFrame, text = "Delete Book", font = ("Barlow", 10), command =
dltbkdb)
    SubmitBtn.place(relx = 0.60, rely = 0.25, anchor = CENTER)

    win.mainloop()
```

IssueBook.py

```
from tkinter import *
from tkinter import messagebox
from mysql.connector import *
from Functions.ViewBooks import *

def isbkdb():
    # List to store books id
    bkid = []
    bkid2 = []
    global bkqtycheck, bkqty
    bkqtycheck = []
    bkqty = 0

    rname = nameInfo.get()
    aInfo = ageInfo.get()
    rtn = rtndateInfo.get()
    addr1 = add1Info.get("1.0", END)
    addr2 = add2Info.get("1.0", END)
    pincode = pncdInfo.get()
    contact = contInfo.get()
    emInfo = emailInfo.get()
    bkInfo = int(bkidInfo.get())

    extractbk = "SELECT book_id FROM book_list_master;"
    try:
```

```
        cur.execute(extractbk)
        result = cur.fetchall()

        for i in result:
            bkid.append(i[0])

        if bkInfo in bkid:
            checkAvail = f"SELECT book_status FROM book_list_master WHERE book_id =
{bkInfo};"
            cur.execute(checkAvail)
            result = cur.fetchall()

            for i in result:
                bkid2.append(i[0])

            if bkid2[0] == 1:
                status = True
                bkqty = f"SELECT book_avlb_qty FROM book_list_master where book_id =
{bkInfo};"
                cur.execute(bkqty)
                result = cur.fetchall()

                for i in result:
                    bkqtycheck.append(i[0])

                bkqty = bkqtycheck[0]
                bkqty -= 1
            else:
                status = False
        else:
            messagebox.showinfo("Error", "Book ID not present")
    except:
        messagebox.showinfo("Error", "Can't fetch Book IDs")

    issue = f"INSERT INTO `book_rent_master` (`transac_num`, `renter_name`, `renter_age`,
`renting_date`, `prm_return_date`, `return_date`, `rent_status`, `address_line1`,
`address_line2`, `pincode`, `renter_cont`, `renter_email`, `book_id`) VALUES ('',
'{rname}', '{aInfo}', current_timestamp(), '{rtn} 00:00:00', '', '1', '{addr1}',
'{addr2}', '{pincode}', '{contact}', '{emInfo}', '{bkInfo}');"
    update = f"UPDATE book_list_master SET book_avlb_qty = {bkqty} where book_id =
{bkInfo};"

    try:
        if bkInfo in bkid and status == True:
            cur.execute(issue)
            con.commit()
            cur.execute(update)
            con.commit()
            messagebox.showinfo('Success', "Book Issued Successfully")
        else:
            bkid.clear()
            bkid2.clear()
            bkqtycheck.clear()
            messagebox.showinfo('Message', "Book Already Issued")
```

```
    except:
        messagebox.showinfo("Search Error", "The value entered is wrong, Try again")
    finally:
        bkid.clear()
        bkid2.clear()
        bkqtycheck.clear()

    win.destroy()

def op():
    global win, con, cur, nameInfo, ageInfo, rtndateInfo, add1Info, add2Info, pncdInfo,
contInfo, emailInfo, bkidInfo

    # Adding database details
    con = connect(host = "localhost", user = "root", password = "", database =
"lib_mgmt_sys")
    cur = con.cursor()

    # Creating a window
    win = Tk()
    win.title("Library Management System - Issue Book")
    win.geometry("800x480")
    win.minsize(width = 800, height = 480)

    # Creating a title and menu frame
    titleFrame = Frame(win, bg = "#005B96")
    titleFrame.place(relx = 0.0, rely = 0.0, relwidth = 1.0, relheight = 0.15)

    menuFrame = Frame(win, bg = "#6497B1")
    menuFrame.place(relx = 0.0, rely = 0.15, relwidth = 1.0, relheight = 0.85)

    # Creating a title heading
    heading = Label(titleFrame, text = "Issue Book", font = ("Barlow", 20, "bold"), bg =
"#005B96", fg = "white")
    heading.place(relx = 0.50, rely = 0.50, anchor = CENTER)

    # Creating a form
    name = Label(menuFrame, text = "Name: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    name.place(relx = 0.40, rely = 0.09, anchor = CENTER)
    nameInfo = Entry(menuFrame)
    nameInfo.place(relx = 0.60, rely = 0.09, anchor = CENTER)

    age = Label(menuFrame, text = "Age: ", bg = "#6497B1", fg = "white", font = ("Barlow",
10))
    age.place(relx = 0.40, rely = 0.18, anchor = CENTER)
    ageInfo = Entry(menuFrame)
    ageInfo.place(relx = 0.60, rely = 0.18, anchor = CENTER)

    rtndate = Label(menuFrame, text = "Promised Return date: \n(yyyy-mm-dd)", bg =
"#6497B1", fg = "white", font = ("Barlow", 10))
    rtndate.place(relx = 0.40, rely = 0.27, anchor = CENTER)
    rtndateInfo = Entry(menuFrame)
    rtndateInfo.place(relx = 0.60, rely = 0.27, anchor = CENTER)
```

```
    add1 = Label(menuFrame, text = "Address Line 1: ", bg = "#6497B1", fg = "white", font
= ("Barlow", 10))
    add1.place(relx = 0.40, rely = 0.36, anchor = CENTER)
    add1Info = Text(menuFrame, width = 20, height = 2)
    add1Info.place(relx = 0.60, rely = 0.36, anchor = CENTER)

    add2 = Label(menuFrame, text = "Address Line 2: ", bg = "#6497B1", fg = "white", font
= ("Barlow", 10))
    add2.place(relx = 0.40, rely = 0.45, anchor = CENTER)
    add2Info = Text(menuFrame, width = 20, height = 2)
    add2Info.place(relx = 0.60, rely = 0.45, anchor = CENTER)

    pncd = Label(menuFrame, text = "Pincode: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    pncd.place(relx = 0.40, rely = 0.54, anchor = CENTER)
    pncdInfo = Entry(menuFrame)
    pncdInfo.place(relx = 0.60, rely = 0.54, anchor = CENTER)

    cont = Label(menuFrame, text = "Contact No.: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    cont.place(relx = 0.40, rely = 0.63, anchor = CENTER)
    contInfo = Entry(menuFrame)
    contInfo.place(relx = 0.60, rely = 0.63, anchor = CENTER)

    email = Label(menuFrame, text = "Email: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    email.place(relx = 0.40, rely = 0.72, anchor = CENTER)
    emailInfo = Entry(menuFrame)
    emailInfo.place(relx = 0.60, rely = 0.72, anchor = CENTER)

    bkid = Label(menuFrame, text = "Book ID: ", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    bkid.place(relx = 0.40, rely = 0.81, anchor = CENTER)
    bkidInfo = Entry(menuFrame)
    bkidInfo.place(relx = 0.60, rely = 0.81, anchor = CENTER)

    ViewBooks = Button(menuFrame, text = "View Books", font = ("Barlow", 10), command =
viewbk)
    ViewBooks.place(relx = 0.40, rely = 0.95, anchor = CENTER)

    SubmitBtn = Button(menuFrame, text = "Issue Book", font = ("Barlow", 10), command =
isbkdb)
    SubmitBtn.place(relx = 0.60, rely = 0.95, anchor = CENTER)

    win.mainloop()
```

ReturnBook.py

```
from tkinter import *
from tkinter import messagebox
from mysql.connector import *
from Functions.RenterList import *
```

```
def rtnbkdb():

    transID = int(rtnbkInfo.get())

    allBk = []
    qrbkid = []
    qrresult = []
    qravlbqty = []

    extractbk = "SELECT transac_num FROM book_rent_master;"
    try:
        cur.execute(extractbk)
        result = cur.fetchall()

        for i in result:
            allBk.append(i[0])

        if transID in allBk:
            checkQuery = f"SELECT rent_status FROM book_rent_master WHERE transac_num =
{transID};"
            cur.execute(checkQuery)
            result = cur.fetchall()
            for i in result:
                qrresult.append(i[0])

            getbkID = f"SELECT book_id FROM book_rent_master where transac_num =
{transID};"
            cur.execute(getbkID)
            bkid = cur.fetchall()
            for i in bkid:
                qrbkid.append(i[0])

            getavlbqty = f"SELECT book_avlb_qty FROM book_list_master WHERE book_id =
{qrbkid[0]};"
            cur.execute(getavlbqty)
            avlbqty = cur.fetchall()
            for i in avlbqty:
                qravlbqty.append(i[0])

            qty = qravlbqty[0]
            qty +=1

            if qrresult[0] == 1:
                finalQuery = f"UPDATE book_rent_master SET rent_status = 0, return_date =
current_timestamp() WHERE transac_num = {transID};"
                updtval = f"UPDATE book_list_master SET book_avlb_qty = {qty} WHERE
book_id = {qrbkid[0]};"
                cur.execute(finalQuery)
                con.commit()
                cur.execute(updtval)
                con.commit()

                messagebox.showinfo("Successful", "Book returned succesfully")
            else:
```

```
                messagebox.showinfo("Already returned", "Book already returned")
    except:
        messagebox.showinfo("Error", "Some error has been occurred")
    finally:
        allBk.clear()
        qrbkid.clear()
        qrresult.clear()
        qravlbqty.clear()

    win.destroy()

def op():
    global win, con, cur, rtnbkInfo

    # Adding database details
    con = connect(host = "localhost", user = "root", password = "", database =
"lib_mgmt_sys")
    cur = con.cursor()

    # Creating a window
    win = Tk()
    win.title("Library Management System - Return Book")
    win.geometry("800x480")
    win.minsize(width = 800, height = 480)

    # Creating a title and menu frame
    titleFrame = Frame(win, bg = "#005B96")
    titleFrame.place(relx = 0.0, rely = 0.0, relwidth = 1.0, relheight = 0.15)

    menuFrame = Frame(win, bg = "#6497B1")
    menuFrame.place(relx = 0.0, rely = 0.15, relwidth = 1.0, relheight = 0.85)

    # Creating a title heading
    heading = Label(titleFrame, text = "Return Book", font = ("Barlow", 20, "bold"), bg =
"#005B96", fg = "white")
    heading.place(relx = 0.50, rely = 0.50, anchor = CENTER)

    # Creating a form
    rtnbk = Label(menuFrame, text = "Transaction ID: ", bg = "#6497B1", fg = "white", font
= ("Barlow", 10))
    rtnbk.place(relx = 0.40, rely = 0.10, anchor = CENTER)
    rtnbkInfo = Entry(menuFrame)
    rtnbkInfo.place(relx = 0.60, rely = 0.10, anchor = CENTER)

    RenterList = Button(menuFrame, text = "Renter's List", font = ("Barlow", 10), command
= rntslist)
    RenterList.place(relx = 0.40, rely = 0.25, anchor = CENTER)

    SubmitBtn = Button(menuFrame, text = "Return Book", font = ("Barlow", 10), command =
rtnbkdb)
    SubmitBtn.place(relx = 0.60, rely = 0.25, anchor = CENTER)

    win.mainloop()
```

UpdateBook.py

```
from tkinter import *
from tkinter import messagebox
from mysql.connector import *
from Functions.ViewBooks import *

def updtdb():
    clickedInfo = clicked.get()
    update = updt.get()
    bkid = int(bidInfo.get())

    updtDict = {"Name": "book_name", "Author": "book_author", "Publisher": "book_pub",
"ISBN": "book_isbn", "Rent Price": "book_rent_price", "Total Qunatity": "book_total_qty"}
    updtFinal = updtDict[clickedInfo]

    updtQuery = f"UPDATE book_list_master SET {updtFinal} ='{update}' WHERE book_id =
{bkid};"

    try:
        cur.execute(updtQuery)
        con.commit()
        messagebox.showinfo("Success", "Updated successfully")
    except:
        messagebox.showinfo("Error", "Some error has been occurred")

    win.destroy()

def op():
    global win, con, cur, clicked, updt, bidInfo

    # Adding database details
    con = connect(host = "localhost", user = "root", password = "", database =
"lib_mgmt_sys")
    cur = con.cursor()

    # Creating a window
    win = Tk()
    win.title("Library Management System - Update Book")
    win.geometry("800x480")

    # Creating a title and menu frame
    titleFrame = Frame(win, bg = "#005B96")
    titleFrame.place(relx = 0.0, rely = 0.0, relwidth = 1.0, relheight = 0.15)

    menuFrame = Frame(win, bg = "#6497B1")
    menuFrame.place(relx = 0.0, rely = 0.15, relwidth = 1.0, relheight = 0.85)

    # Creating a title heading
    heading = Label(titleFrame, text = "Update Book", font = ("Barlow", 20, "bold"), bg =
"#005B96", fg = "white")
    heading.place(relx = 0.50, rely = 0.50, anchor = CENTER)

    # Adding a menu
```

```
    options = ["Name", "Author", "Publisher", "ISBN", "Rent Price", "Total Quantity"]
    clicked = StringVar()
    clicked.set("Name")
    selectop = Label(menuFrame, text = "Select operation:", bg = "#6497B1", fg = "white",
font = ("Barlow", 10))
    selectop.place(relx = 0.40, rely = 0.10, anchor = CENTER)
    drop = OptionMenu(menuFrame, clicked, *options)
    drop.place(relx = 0.60, rely = 0.10, anchor = CENTER)
    updt = Entry(menuFrame)
    updt.place(relx = 0.50, rely = 0.20, anchor = CENTER)

    bid = Label(menuFrame, text = "Book ID:", bg = "#6497B1", fg = "white", font =
("Barlow", 10))
    bid.place(relx = 0.40, rely = 0.30, anchor = CENTER)
    bidInfo = Entry(menuFrame)
    bidInfo.place(relx = 0.60, rely = 0.30, anchor = CENTER)

    ViewBooks = Button(menuFrame, text = "View Books", command = viewbk)
    ViewBooks.place(relx = 0.40, rely = 0.45, anchor = CENTER)

    updtBt = Button(menuFrame, text = "Update Book", command = updtdb)
    updtBt.place(relx = 0.60, rely = 0.45, anchor = CENTER)

    win.mainloop()
```

ViewBooks.py

```
from tkinter import *
from mysql.connector import *

def viewbk():
    l1 = []
    # Creating a window
    win = Tk()
    win.title("Library Management System - View Books")
    win.geometry("1200x480")
    win.minsize(width = 800, height = 480)

    # Adding database details
    con = connect(host = "localhost", user = "root", password = "", database =
"lib_mgmt_sys")
    cur = con.cursor()

    extractdt = "SELECT * FROM book_list_master;"
    cur.execute(extractdt)
    result = cur.fetchall()

    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 0, row = 0)
    Label(frame, text = "ID", width = 15, font = ("Barlow", 10, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 1, row = 0)
    Label(frame, text = "Name", width = 15, font = ("Barlow", 10, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
```

```
    frame.grid(column = 2, row = 0)
    Label(frame, text = "Author", width = 15, font = ("Barlow", 10, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 3, row = 0)
    Label(frame, text = "Publisher", width = 15, font = ("Barlow", 10, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 4, row = 0)
    Label(frame, text = "ISBN", width = 15, font = ("Barlow", 10, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 5, row = 0)
    Label(frame, text = "Rent Price", width = 15, font = ("Barlow", 10, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 6, row = 0)
    Label(frame, text = "Total Quantity", width = 15, font = ("Barlow", 10,
"bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 7, row = 0)
    Label(frame, text = "Available Quantity", width = 15, font = ("Barlow", 10,
"bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 8, row = 0)
    Label(frame, text = "Available Status", width = 15, font = ("Barlow", 10,
"bold")).pack()

    for i in range(9):
        for j in result:
            l1.append(j[i])

        max = len(l1)

        for k in range(max):
            frame = Frame(win, relief = RAISED, borderwidth = 1)
            frame.grid(column = i, row = k + 1)
            Label(frame, text = l1[k], width = 15, font = ("Barlow", 10)).pack()

        l1.clear()

    win.mainloop()
```

RenterList.py

```
from tkinter import *
from mysql.connector import *

def rntslist():
    l1 = []
    # Creating a window
    win = Tk()
    win.title("Library Management System - View Books")
    win.geometry("1500x480")
    win.minsize(width = 800, height = 480)

    # Adding database details
```

```
    con = connect(host = "localhost", user = "root", password = "", database =
"lib_mgmt_sys")
    cur = con.cursor()

    extractdt = "SELECT * FROM book_rent_master;"
    cur.execute(extractdt)
    result = cur.fetchall()

    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 0, row = 0)
    Label(frame, text = "Transaction ID", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 1, row = 0)
    Label(frame, text = "Name", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 2, row = 0)
    Label(frame, text = "Age", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 3, row = 0)
    Label(frame, text = "Renting Date", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 4, row = 0)
    Label(frame, text = "Promised \nReturn Date", width = 15, font = ("Barlow", 8,
"bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 5, row = 0)
    Label(frame, text = "Return Date", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 6, row = 0)
    Label(frame, text = "Rent Status", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 7, row = 0)
    Label(frame, text = "Address \nLine 1", width = 15, font = ("Barlow", 8,
"bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 8, row = 0)
    Label(frame, text = "Address \nLine 2", width = 15, font = ("Barlow", 8,
"bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 9, row = 0)
    Label(frame, text = "Pincode", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 10, row = 0)
    Label(frame, text = "Contact No", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 11, row = 0)
    Label(frame, text = "Email", width = 15, font = ("Barlow", 8, "bold")).pack()
    frame = Frame(win, relief = RAISED, borderwidth = 1)
    frame.grid(column = 12, row = 0)
    Label(frame, text = "Rented \nBook ID", width = 15, font = ("Barlow", 8,
"bold")).pack()

    for i in range(13):
        for j in result:
```

```
            l1.append(j[i])

        max = len(l1)

        for k in range(max):
            frame = Frame(win, relief = RAISED, borderwidth = 1)
            frame.grid(column = i, row = k + 1)
            Label(frame, text = l1[k], width = 15, font = ("Barlow", 8)).pack()

        l1.clear()

    win.mainloop()
```

PrintReceipt.py

```
from tkinter import *
from tkinter import messagebox
from mysql.connector import *
from fpdf import FPDF
from Functions.RenterList import *

def prtrcpt():

    global success

    qrinfo = []
    rntlst = []
    bknamelt = []

    tid = int(transidInfo.get())

    query = "SELECT transac_num FROM book_rent_master;"

    try:
        cur.execute(query)
        result = cur.fetchall()

        for i in result:
            qrinfo.append(i[0])

        if tid in qrinfo:
            rntqr = f"SELECT * FROM book_rent_master WHERE transac_num = {tid};"

            cur.execute(rntqr)
            result = cur.fetchall()

            for i in result[0]:
                rntlst.append(i)

            bkname = f"SELECT book_name FROM book_list_master WHERE book_id =
{rntlst[12]};"

            cur.execute(bkname)
            result = cur.fetchall()
```

```
            for i in result[0]:
                bknamelt.append(i)

            pdf = FPDF()

            # Add a page
            pdf.add_page()

            # Set the font for the heading
            pdf.set_font("Arial", size = 26, style="B")

            # Write the heading
            pdf.cell(0, 10, "Absolute Minds Library", ln=1, align="C")

            pdf.set_font("Arial", size = 18)
            pdf.cell(0, 10, "Receipt", ln = 1, align = "C")

            # Set the font for the body text
            pdf.set_font("Arial", size=12)

            # Write the body text
            pdf.cell(0, 10, f"Transaction ID: #{rntlst[0]}", ln=1)
            pdf.cell(0, 10, f"Name: {rntlst[1]}", ln=1)
            pdf.cell(0, 10, f"Date and time: {rntlst[3]}", ln=1)
            pdf.cell(0, 10, f"Contact No.: {rntlst[10]}", ln=1)
            pdf.cell(0, 10, f"Email ID: {rntlst[11]}", ln=1)
            pdf.cell(0, 10, f"Book ID: {rntlst[12]}", ln=1)
            pdf.cell(0, 10, f"Book name: {bknamelt[0]}", ln=1)

            # Output the PDF
            pdf.output(f"Receipts/{rntlst[0]} {rntlst[1]}.pdf")

            messagebox.showinfo("Successful", "Receipt generated successfully under
Receipt folder")
        else:
            messagebox.showinfo("Error", "Transaction ID not present")

    except:
        messagebox.showinfo("Error", "Some error has been occurred")

    finally:
        qrinfo.clear()
        rntlst.clear()
        bknamelt.clear()
        win.destroy()

def op():

    global con, cur, win, transidInfo

    # Adding database details
    con = connect(host = "localhost", user = "root", password = "", database =
"lib_mgmt_sys")
```

```
    cur = con.cursor()

    # Creating a window
    win = Tk()
    win.title("Library Management System - Print Receipt")
    win.geometry("800x480")
    win.minsize(width = 800, height = 480)

    # Creating a title and menu frame
    titleFrame = Frame(win, bg = "#005B96")
    titleFrame.place(relx = 0.0, rely = 0.0, relwidth = 1.0, relheight = 0.15)

    menuFrame = Frame(win, bg = "#6497B1")
    menuFrame.place(relx = 0.0, rely = 0.15, relwidth = 1.0, relheight = 0.85)

    # Creating a title heading
    heading = Label(titleFrame, text = "Print Receipt", font = ("Barlow", 20, "bold"), bg
= "#005B96", fg = "white")
    heading.place(relx = 0.50, rely = 0.50, anchor = CENTER)

    # Creating a menu
    transid = Label(menuFrame, text = "Transaction ID: ", bg = "#6497B1", fg = "white",
font = ("Barlow", 10))
    transid.place(relx = 0.40, rely = 0.10, anchor = CENTER)
    transidInfo = Entry(menuFrame)
    transidInfo.place(relx = 0.60, rely = 0.10, anchor = CENTER)

    RenterList = Button(menuFrame, text = "Renter's List", font = ("Barlow", 10), command
= rntslist)
    RenterList.place(relx = 0.40, rely = 0.25, anchor = CENTER)

    SubmitBtn = Button(menuFrame, text = "Print Receipt", font = ("Barlow", 10), command =
prtrcpt)
    SubmitBtn.place(relx = 0.60, rely = 0.25, anchor = CENTER)
```

10. System Testing

Here is a system testing plan for the Library management system in python GUI with MySQL, which includes the following functionality :-

- Admin Login Page
- Adding New Books
- Issue Book and View Book
- Delete Books
- Return Book
- Renters List
- Print Receipt

Test Cases :

Test Case 1 : Admin Login

This case verifies that the login of admin should be successful if correct credentials are given.

Steps:

1) Open the system
2) Enter credentials

3) Click on Login
Expected Results : The admin should be able to login into its particular account and should be able to view various operations of Library management system.

Test Case 2: Adding New Books
This case verifies that the input of adding book details is added to the database.
Steps:
1) Open system and do admin login
2) Click on Add Book Option
3) Fill in the required details
4) Then click on "Add Book"
Expected Results : The book details should be added to the database after clicking on "Add Book".

Test Case 3: Issue Book and View Book
This test case verifies 2 things that after adding book can we view that book in database and after issuing book is the quantity of that book is being reduced in database.
Steps:
1) Open the system and do admin login.
2) Now click on "Issue Book" option.
3) Fill in the required details.
4) Here you can view the books by clicking on "View Books".
Expected Results : After adding book we view that book in database and after issuing book is the quantity of that book is being reduced in database.

Test Case 4: Delete Books
This test case verifies that the book is deleted or not from the database after clicking on "Delete Book " option.
Steps :
1) Open the system and do admin login.
2) Select "Delete Book" option.
3) Enter the "Id".
4) Click "Delete Book" button.
Expected Results : After deleting book we cannot view that book in database.

Test Case 5: Return Book
This test case verifies that the book quantity is updated or not from the customer after clicking on "Return Book " option.
Steps :
1) Open the system and do admin login.
2) Select "Return Book" option.
3) Enter "transaction Id".
4) Click on "Return Book" button.
Expected Result : After clicking on return book the quantity should be updated in database.

Test Case 6: Renters List
This test case verifies after issuing the book renters list is updated or not after clicking on "Return Book " option.
Steps :
1) Open the system and do admin login.
2) Select "Renters List" option.
3) Displays Renters list and return status.

Expected Result : After clicking on renters book it should display the renters list and their return status.

Test Case 7: Print Receipt

This test case verifies that after issuing the book and after clicking the “Print receipt ” option , the pdf of the receipt is generated or not.

Steps :

1) Open the system and do admin login.
2) Issue the book to the customer.
3) Now click on “Print Receipt ” option.

Expected Result : After clicking on “Print Receipt ” option the receipt (PDF) should be generated in the folder path specified in the program.

10.1 Unit Testing

We have done unit testing in this project, while making various modules, we have tested them parallely. We gave one day to each module for testing with the main module of the project.
Modules Testing dates are as follows:

Sr. No.	Module	Testing date
1	Add Book	21 April 2023
2	Delete Book	23 April 2023
3	Issue Book	25 April 2023
4	Return Book	26 April 2023
5	Update Book	27 April 2023
6	View Books	28 April 2023
7	Renter’s List	29 April 2023
8	Print Receipt	30 April 2023

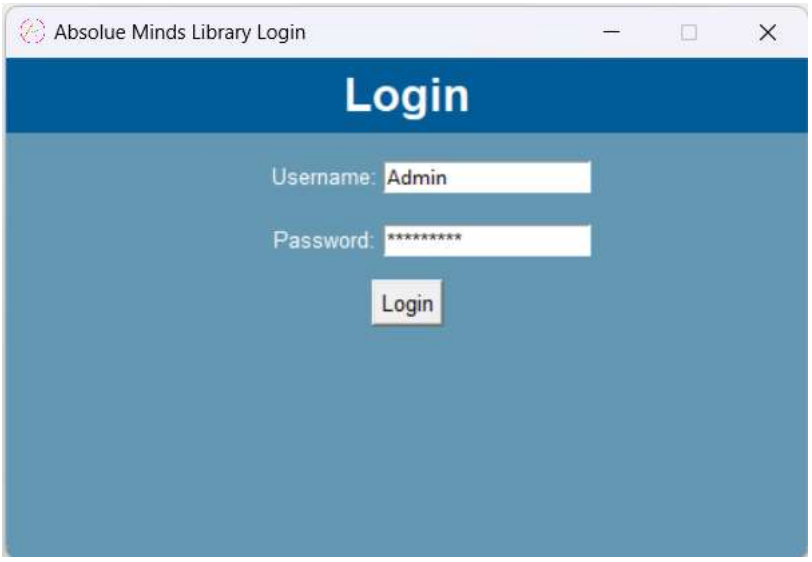
10.2 White Box Testing

We have successfully executed white box testing by seeing the internal code, testing the functionality and observed the correctness of the project. We have tested all modules and observed that the system is working according to the need of the user. The code of the module is tested and the output is according to the requirement of the user. We have observed some errors in code and corrected it instantly. This type of testing helps us to understand the functionality and working of the system.

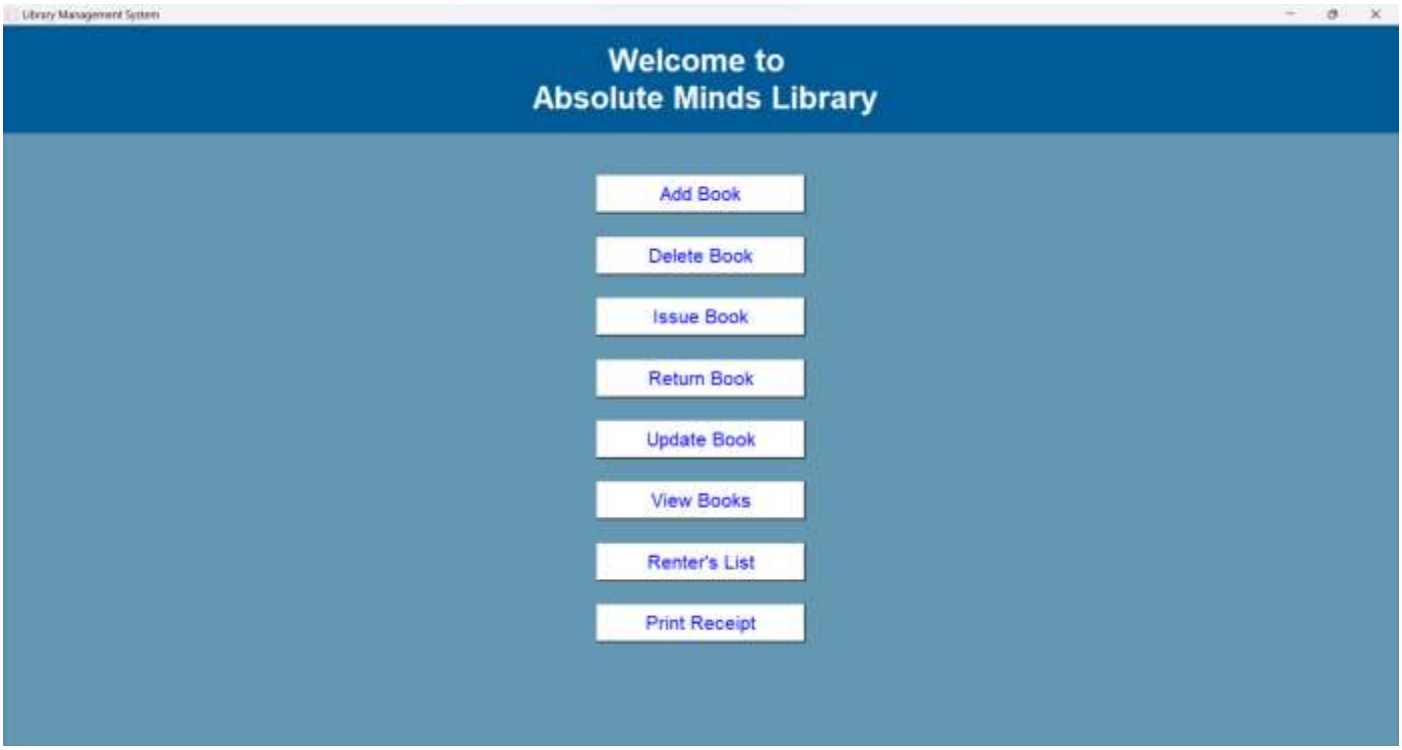
10.3 Black Box Testing

This type of testing is used o test the functionality of the system. The tester in this case was provided with some of the inputs values and respected desired output. On providing input, the output has matched with the desired results and it is tested as ”OK”. We have observed some of the errors and resolved it instantly.

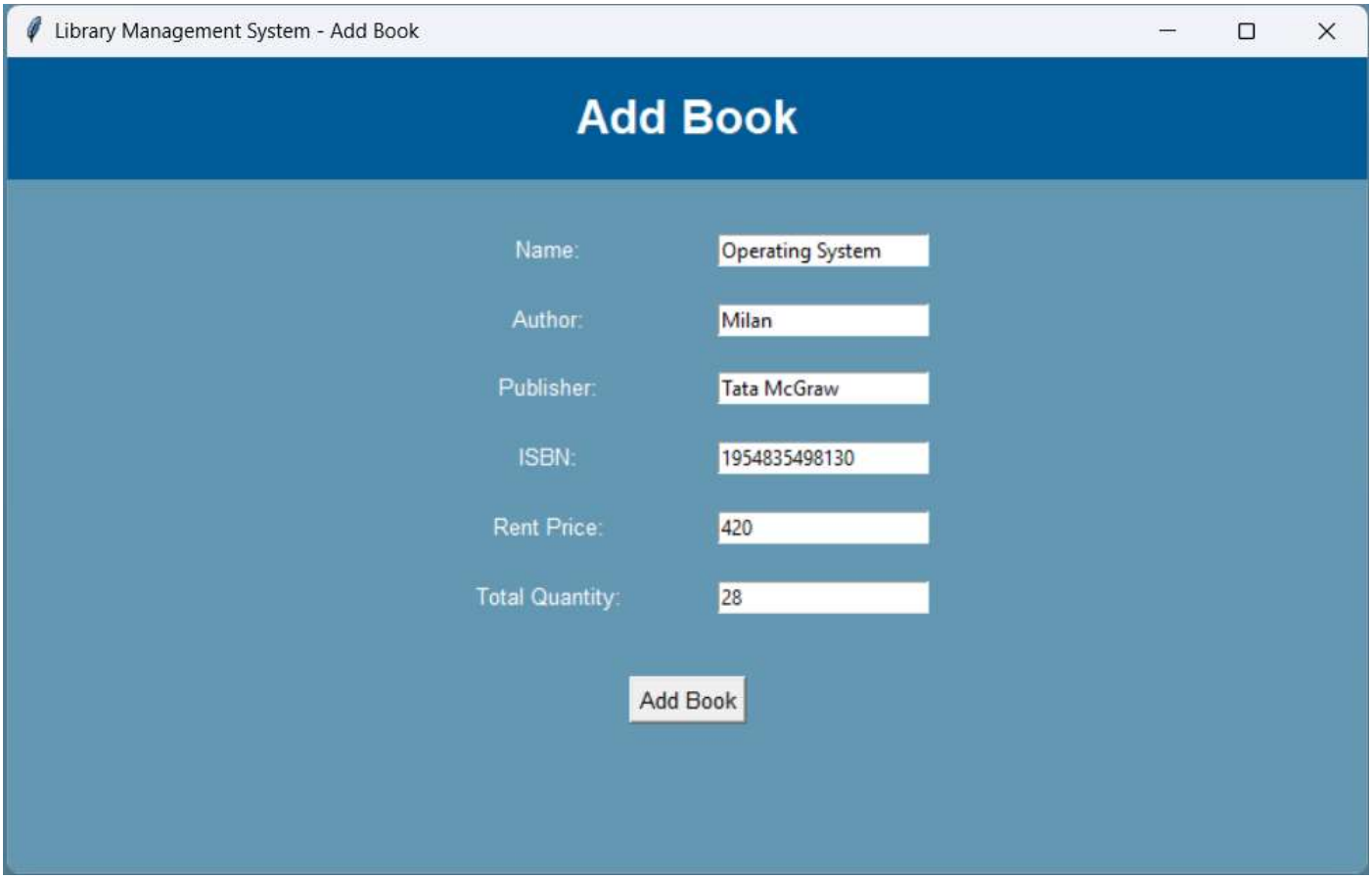
11. Results



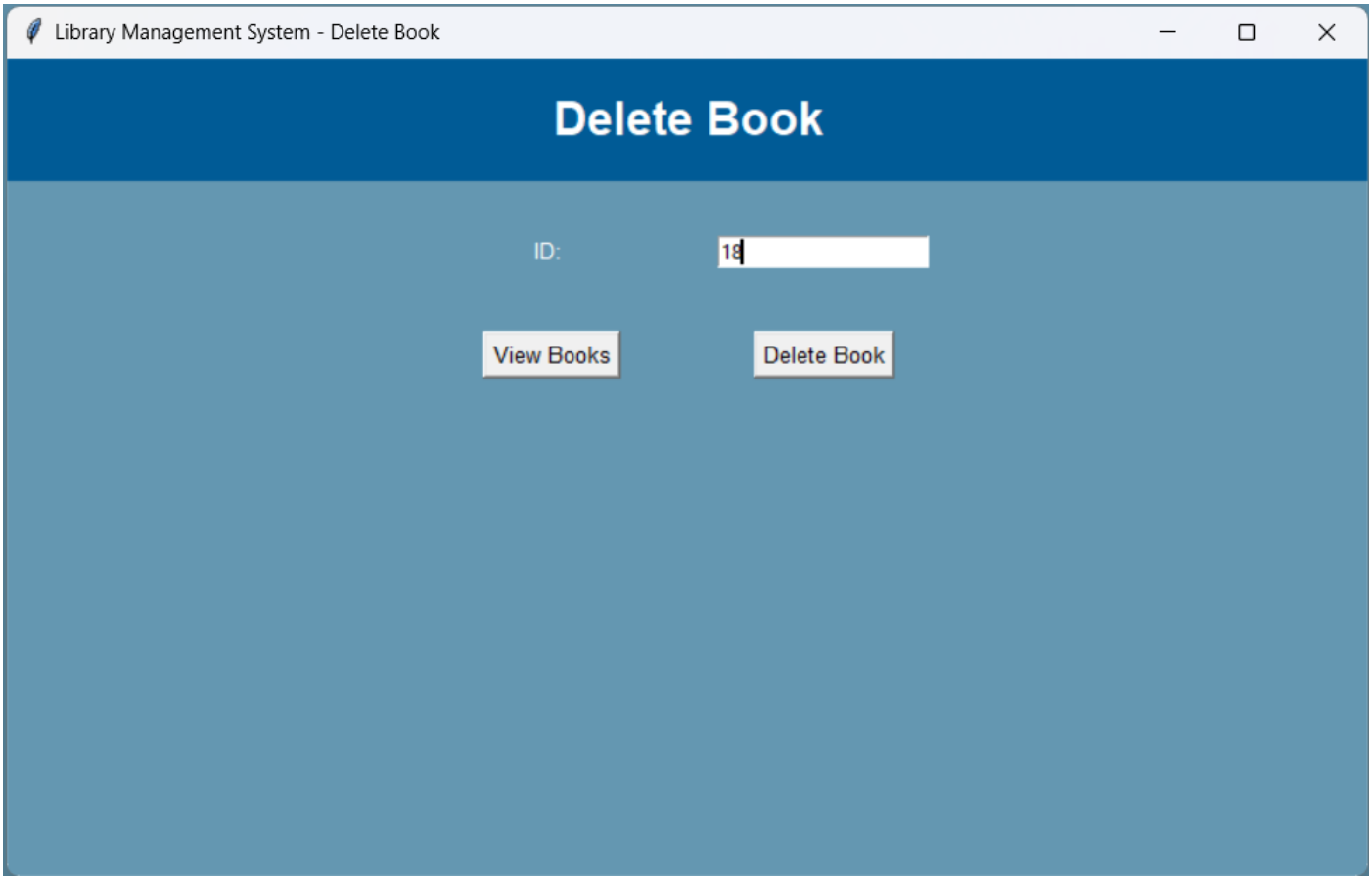
This window is for login of the admin to the system.



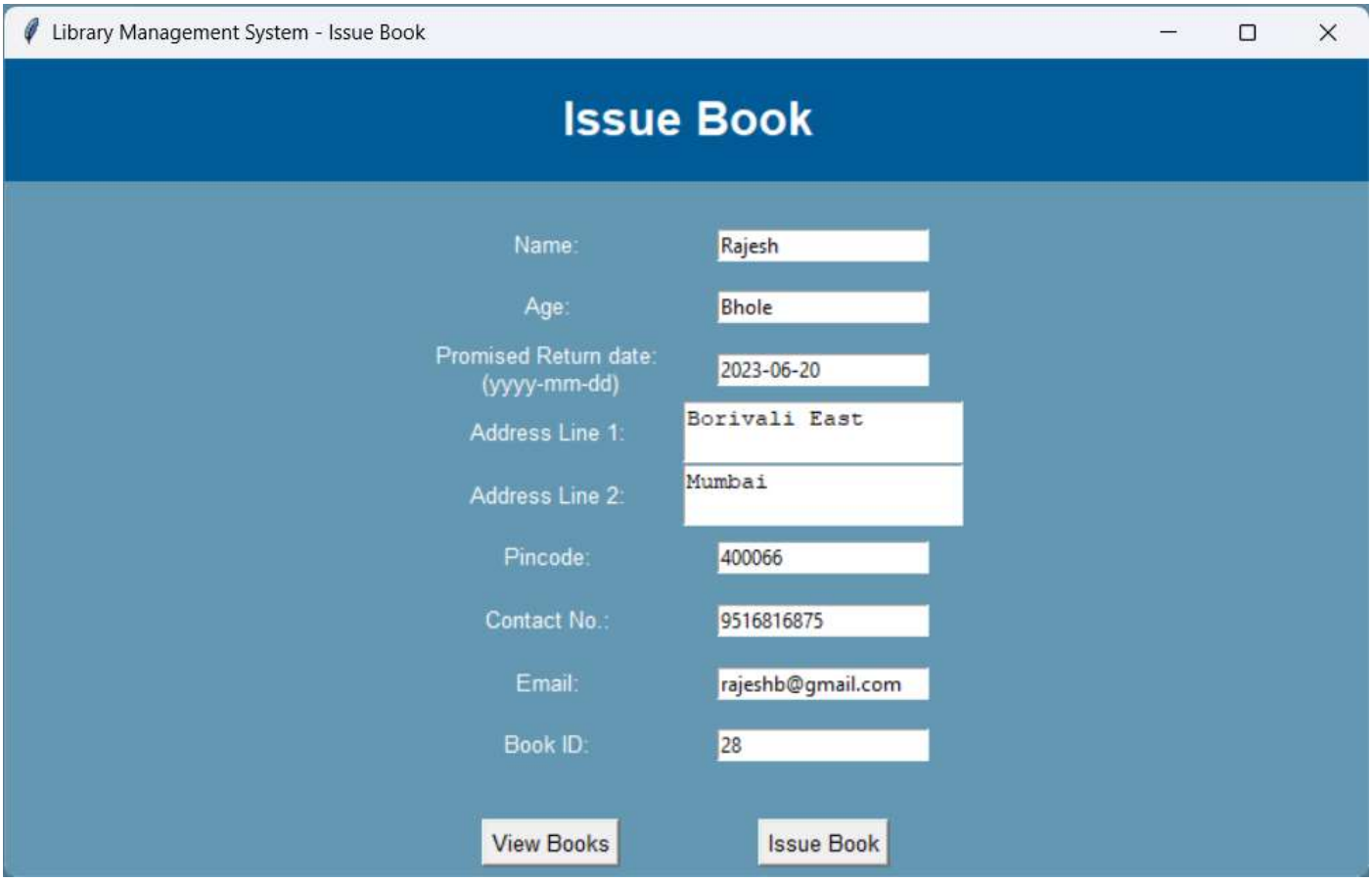
This is the homepage of the Library Management System. Here, we can perform various operations.



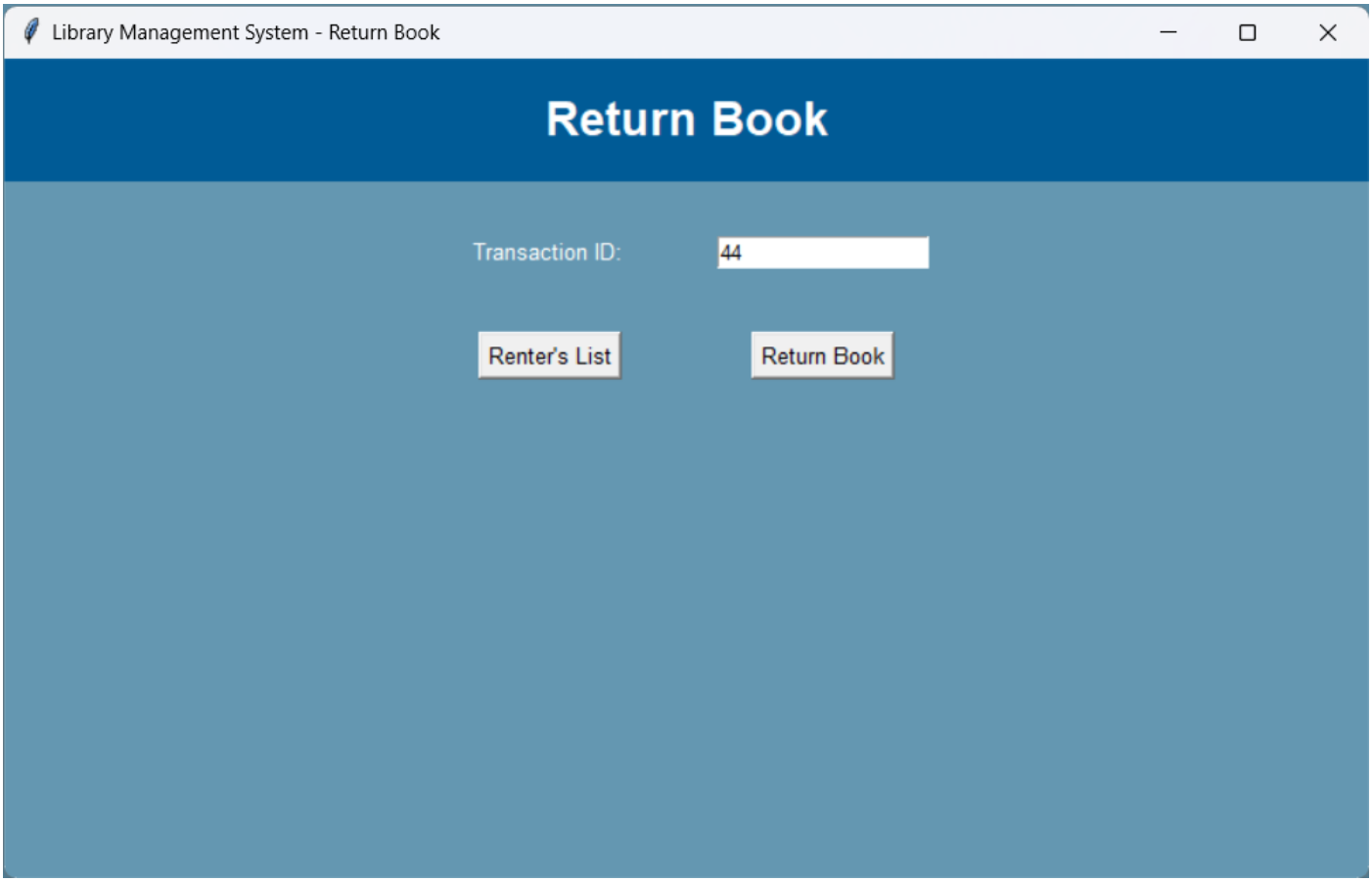
In this window, we can add book to the database.



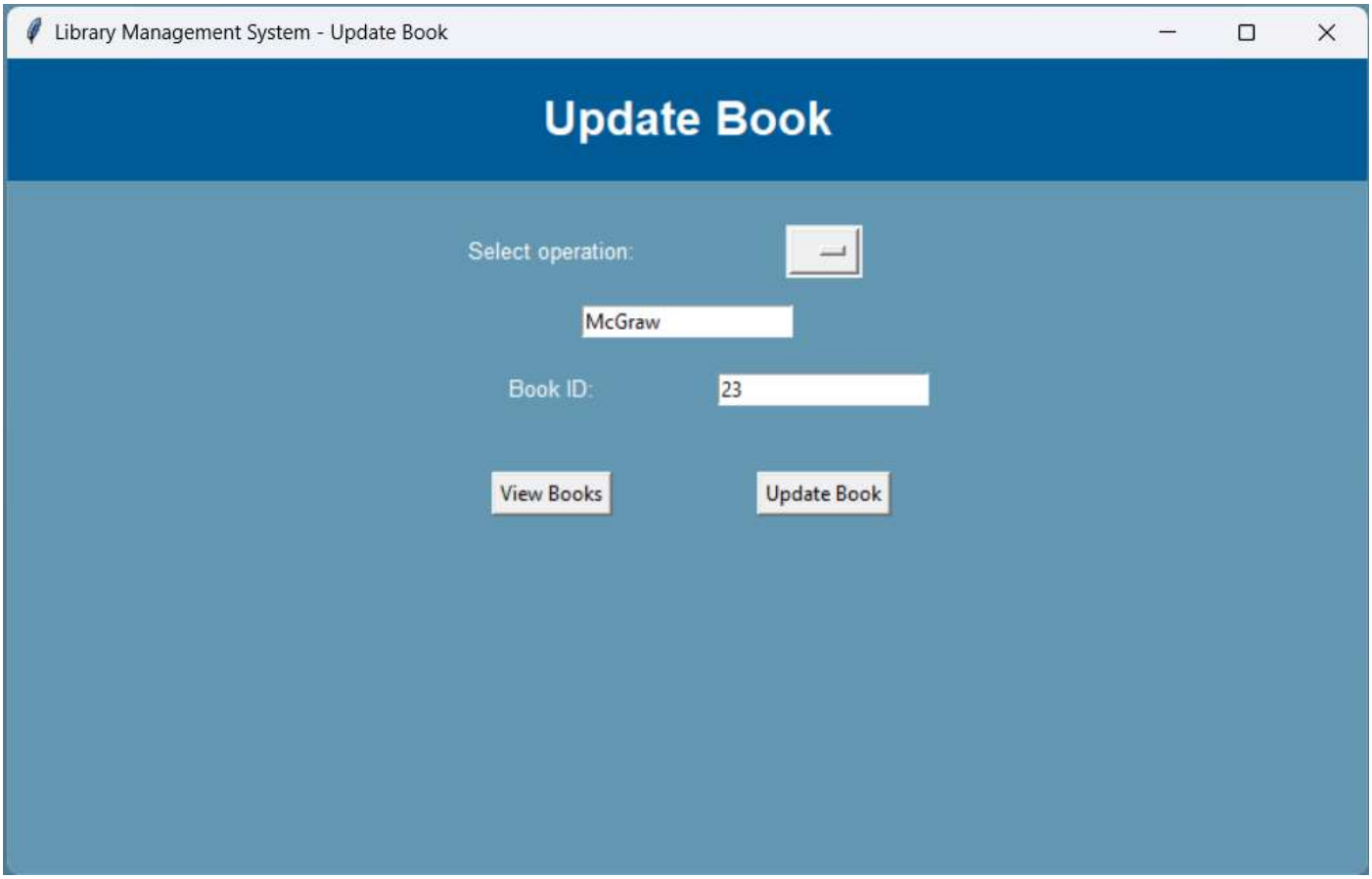
In this window, we can delete book from database.



In this window, we can issue book to the customer.



In this window, we can take back the book means customer can return book.

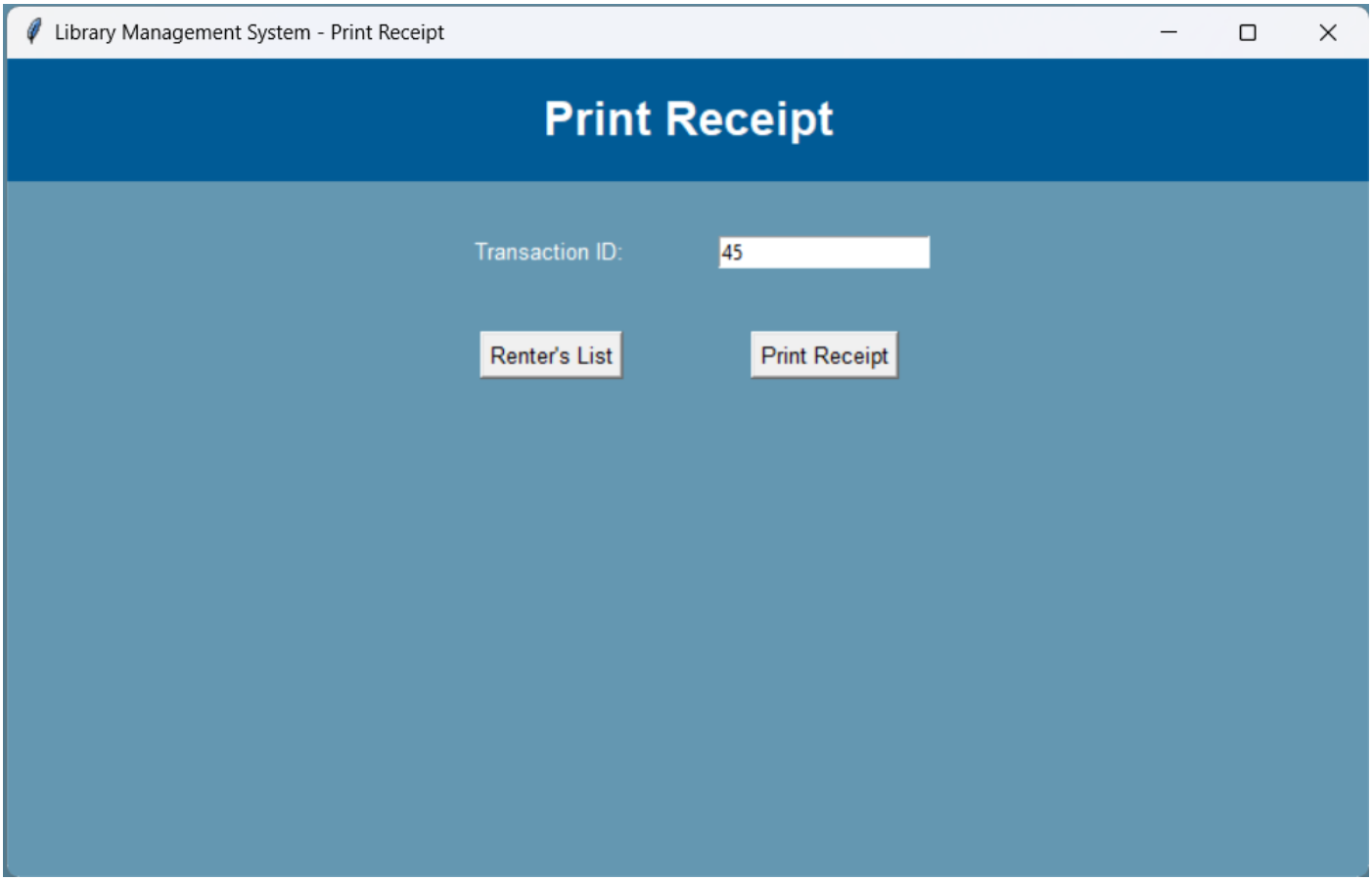


In this window, we can update the book details like Name, Author, Publisher, ISBN, Rent Price, Total Quantity.

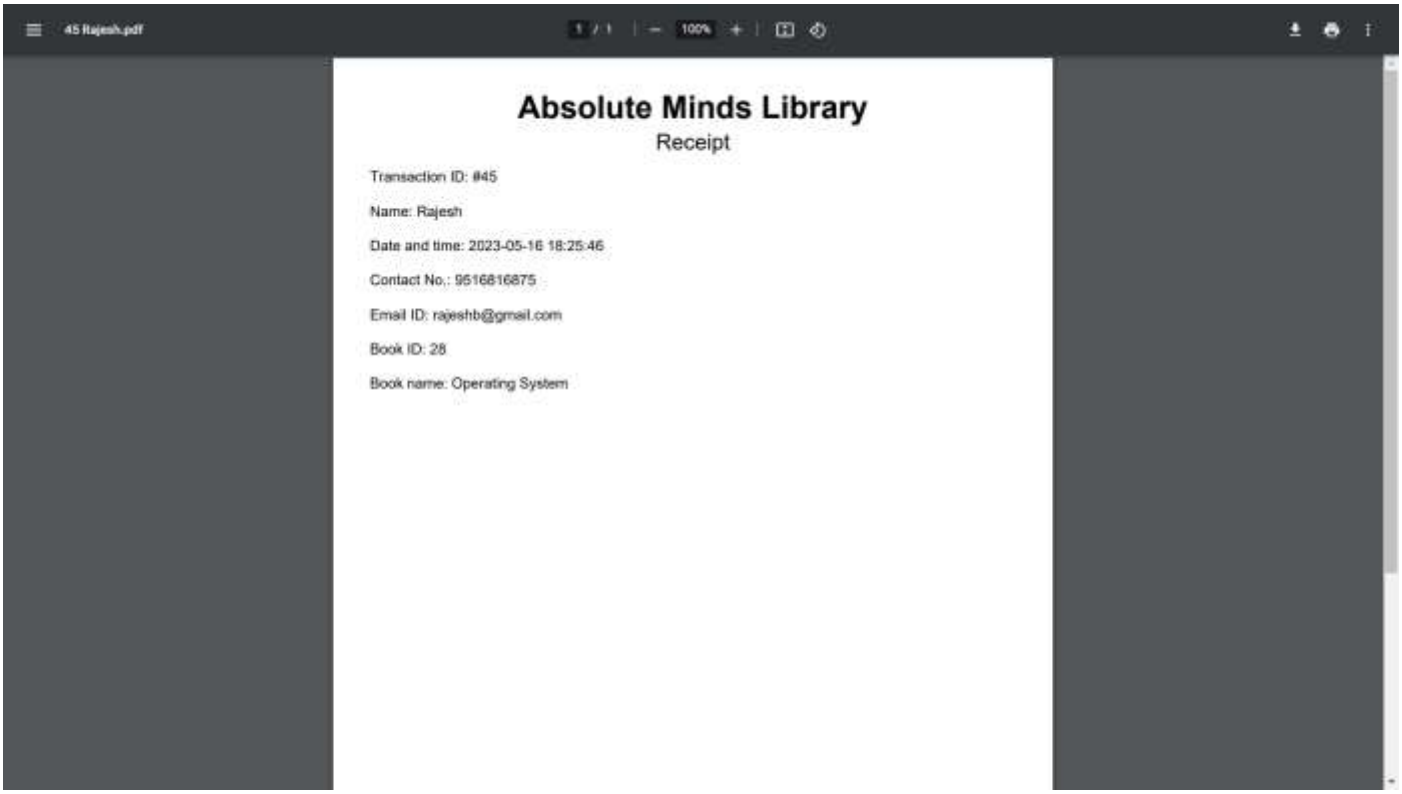
ID	Name	Author	Publisher	ISBN	Rent Price	Total Quantity	Available Quantity	Available Status
18	Past and Future	Hannah Arendt	Pearson	2678434598134	250	30	30	1
20	One Piece	Eiichiro Oda	Informa PLC	510151611611	190	25	24	1
21	Astro Boy	Osamu Tezuka	Simson & Schuster	4318954348726	130	10	10	1
22	The Goon	Eric Powell	McGraw	2832434867878	190	25	24	1
23	Giant Days	John Allison	McGraw	7736818287971	300	18	17	1
24	Naruto	Jiraya	Kishomoto	1551984183651	720	21	20	1
26	The Famous Five	Enid Blyton	Wiley	3027959149016	150	20	20	1
28	Operating System	Milan	Tata McGraw	1954835498130	420	28	27	1

This window shows the available books in the database.

This window shows the renter's list and their return status.



Using functionality of this window, we can print receipt.



This is the receipt for the customer.

12. Future Scope

In future, this system will have connctivity with internet. Any customer can register itself and can buy books from his/her side.
Instead of entering details again and again while issuing book, customer only have to give data once while registering.
UI/UX of the system will imrove.

13. Conclusion

From this mini project, we have learnt about various Python concepts like Exception Handling, Packages, GUI (Tkinter), Database Connectivity with MySQL. Also learnt about various module for PDF operation like “fpdf”. This helped me to strengthen the core Python concepts.

14. References:

https://www.data-flair.training/blogs/library-management-system-python-project
https://www.pyfpdf.readthedocs.io
https://www.pypi.org/project/fpdf
Learning Python (Reference Book)