Minor Project

Library Management GUI



PROJECT REPORT

GUIDED BY

Mrs. Hema Mishra

SUBMITTED BY

Dev Agnihotri

In partial fulfilment for the award of degree of

Bachelors of Computer Application (B.C.A.)

Academic Year 2023-2024



JAGRAN COLLEGE OF ARTS SCIENCE AND COMMERCE

[Date]

HP

[Company name]

[Company address]

[Document title]

[Document subtitle]

|  |
| --- |
|  |
|  |

## COLLEGE CERTIFICATE

## This is to certify that Dev Agnihotri a student of Bachelors of Computer Application 5th semester of this institute has completed the minor project titled “Library Management GUI” in the partial fulfilment of the Bachelors of computer Application program has to prepared this report under my guidance and supervision as required for the completion of his degree. To the best of my knowledge and belief, the above report has been prepared by the above student.

## He possesses a good moral character and I wish him success in his life.

Date:-

Mrs. Hema Mishra

(BCA)

## Candidate Declaration

I hereby declare that the work which is being presented in the Dissertation, entitles “**Minor Project Report**” in the partial fulfillment for the award of Degree of “**Bachelors of Computer Application**”, Jagran College of Arts, Science and Commerce, Chhatrapati Shahu ji Maharaj University, Kanpur is a record of my own investigations carried under the Guidance of Mrs. Hema Mishra Mam, Department of Computer Application, Jagran College of Arts Science and Commerce.

I have not submitted the matter presented in this Dissertion anywhere for the award of any other Degree.

Dev Agnihotri

B.C.A. 5th Semester

Jagran College of Arts Science and Commerce

## Acknowledgement

At this special moment, I want to thank everyone who helped me finish this project. I couldn't have done it without all the support I received along the way.

I want to start by giving a big thank you to my mentor, Hema Mam. Their advice and encouragement meant a lot to me. Also, a huge shout-out to my college teachers who taught me so much. They really shaped how I think.

My college has been an amazing place for learning, and it's been a huge part of my journey. I also want to thank my friends for all the great talks and encouragement. Their ideas really made this project better.

My family has been my rock. I can't thank them enough for always believing in me. And of course, there's a whole world of information online that helped me understand this project better. I'm grateful for all the articles, forums, and tutorials that expanded my knowledge.

To everyone who played a part, big or small, thank you. Your help has meant everything to me and has made me grow, both academically and personally.

Dev Agnihotri

Roll No- 21015002236

BCA 5th Semester

## Abstract

In a world teeming with information, the need for efficient organization and management cannot be overstated. Enter the "Library Management GUI" project, a digital solution designed to revolutionize the way libraries operate.

This project, constructed using the Python programming language and powered by SQLite, offers a user-friendly Graphical User Interface (GUI) that simplifies the complex web of library activities. Within these digital walls, users can seamlessly navigate through the catalog of books, lending and returning with the click of a button. Authors, genres, and publication details are at their fingertips, making the quest for knowledge an accessible and engaging experience.

This project is a testament to the power of innovation in simplifying the complex. It embodies the essence of user-focused design, transforming intricate data structures into an elegant interface. Through the use of SQLite, it ensures data integrity and efficient retrieval, ensuring that the library's operations run seamlessly. The "Library Management GUI" project is a testament to the convergence of technology and traditional knowledge management. It promises to elevate library operations, providing a seamless user experience while maintaining the integrity of data

## Introduction

This project, the "Library Management GUI," stands as a beacon of innovation, made possible through the fusion of Python, SQLite, and the power of technology.

Python, a versatile and commanding programming language, assumes the role of the project's architect. I orchestrate the creation of a user-friendly Graphical User Interface (GUI), akin to a maestro conducting a symphony. Python's elegance and readability render it an ideal choice, ensuring that both seasoned developers and novices like me can comprehend and contribute to this project with ease.

Furthermore, I leverage Python's libraries, with Tkinter at the forefront for GUI development and SQLite for database management. Tkinter crafts the visual interface through which users interact with our library system, imparting a sense of intuitiveness and accessibility. SQLite, on the other hand, serves as the repository of our library's knowledge, adeptly storing and retrieving data, thus ensuring the seamless execution of library operations.

In essence, the "Library Management GUI" project represents a fusion of tradition and technology, envisioning a future where libraries continue to be reservoirs of knowledge, now seamlessly interfaced with the efficiencies and possibilities of the digital era.

## Objectives

In this section I dive in deeper into the technical aspects and the specific goals I aim to achieve with this project:-

Clear Goals: My first and foremost objective is to set clear and well-defined goals for this project. I need to have a precise understanding of what I want to accomplish. In this context, my goal is to develop a sophisticated yet user-friendly library management system using Python and SQLite.

Purposeful Aim: The primary aim of this project is to revolutionize library management. I intend to create a digital system that empowers librarians to efficiently manage the library's vast collection of books and allow them to easily explore, locate, and borrow books.

SMART Objectives: To ensure the success of this project, I am adopting the SMART criteria for my objectives:

* Specific: My objective is specific in that I will create a Graphical User Interface (GUI) library management system using Python programming language and integrate a SQLite database for data storage and retrieval.
* Measurable: I will measure the project's success by quantifying the efficiency gains achieved through automation. Metrics like the time taken for book checkouts, accuracy in tracking book availability, and user satisfaction ratings will be used for measurement.
* Achievable: I will break down the project into manageable tasks and ensure that each component, from the user registration module to the database management, is feasible within the project's scope and resources.
* Relevant: All project tasks and objectives will directly contribute to the development of the library management system. No effort will be wasted on elements that do not align with the project's purpose.
* Time-bound: I will establish a project timeline with clearly defined milestones and deadlines for each phase of development, ensuring that the project progresses efficiently.

In addition to these objectives, I aim to leverage the power of Python libraries like Tkinter for creating an intuitive GUI and SQLite for efficient data handling. The GUI will allow librarians to manage book records, track borrower information, and generate reports with ease. Readers will experience a seamless interface for searching, checking out, and returning books.

## Project-Category

## ‘*Software Development*’

Now, let's delve into the category of our project, the "Library Management GUI." This project falls under the domain of "Software Development," where I use computer programming skills to create something remarkable. In essence, we're modernizing libraries by harnessing the power of technology.

We're probably familiar with libraries as places filled with books, but they're also hubs of information management. Libraries need to keep track of who borrows books, when they're due back, and so much more. That's where our project steps in. I'm developing a specialized computer program using Python, a programming language, and SQLite, a clever way to store and manage information. This program will give libraries a fresh and contemporary look, akin to the apps you might use on your smartphone. In the world of technology, our project is akin to a digital transformation for libraries.

It's like giving them a digital makeover, making library tasks more efficient and user-friendly. It's about bringing the magic of technology to the world of books and knowledge.

So, in summary, our project category is "Software Development," where I'm using my computer skills to create a digital adventure within the library's traditional setting.

## Project-Scope

In this section I give a closer look at what our project includes and what it doesn't. It is like drawing lines around our project to see where it starts and where it ends.

**WHAT'S INSIDE:**

1. Databases: Our project consists of three essential databases:

* Admin Database: This database stores information about library administrators. It keeps track of who can access and manage the system.
* Book Database: Here, we store details about all the books in the library. This includes information like the book's title, author, ISBN, and availability status.
* Student Database: This database records information about the students who use the library. It includes their names, IDs, and borrowing history.

1. Login Form: We're creating a secure login system. Administrators and students will each have their own login credentials to enter the system. This adds a layer of security and ensures that only authorized users can access the library functions.

1. Book Operations: Our project focuses primarily on book-related operations:

* Add Book: This function allows administrators to add new books to the library database. It includes fields to input book details, such as title, author, and price etc.
* Delete Book: When a book is no longer part of the library's collection, administrators can remove it from the database. This keeps our records accurate.
* Issue Book: This feature records when a student borrows a book. It links the book to the student and keeps track of due dates.
* Return Book: When a student returns a book, this function updates the database to reflect the returned status.
* Search Book: Users can search for books using criteria like title, author, or ISBN. This makes finding books in the library quick and easy.
* Show All Books: We can display the entire library collection. It's like looking at a digital library shelf with all the books neatly organized.

**WHAT'S NOT INSIDE:**

1. .Financial Transactions: Our project does not handle money matters. We won't deal with fines, payments, or financial transactions. Our focus is strictly on managing book-related tasks.(though charges column is made available that only provides information)
2. Advanced Features: We won't include advanced features like recommending books based on user preferences or integrating with external databases. Our primary goal is to streamline basic library management.
3. Student Functionalities: We won’t include Student related operations like add student, delete student, search student etc. as it is something reserved to student details and not library details (student related stuff would have been included if it’s some student management project here we are pre-provided by the student table.)

## Project-Timeline

In this section I have mentioned about our timeline to show how our project will unfold. It is like a roadmap with important stops along the way. The project timeline will consist of 12 weeks in total and is split into two sections one the learning phase (will be done in first 8 weeks) and then the implementation phase (will be done in last 3 weeks).

Learning Phase

* Week 1:- Python Basics, Lists, tuple, dictionaries, sets.
* Week 2:- Conditions and Branching, loops, functions. ⎫
* Week 3:- Objects and Classes, file handling, exception handling. ⎫
* Week 4:- Introduction to Relational Databases and Tables. ⎫
* Week 5:- Basic SQL Commands. ⎫
* Week 6:- SQL Functions, Multiple tables, Sub-Queries. ⎫
* Week 7:- Accessing Databases using Python. ⎫
* Week 8:- Introduction to Tkinter.

Implementation Phase/ Project Making Phase

* Week 9: Planning and Setup
* Days 1-2: Define project objectives and scope.
* Day 3: Gather required resources and materials.
* Day 4-5: Set up the development environment (Python, SQLite).
* Days 6-7: Begin designing the database schemas for admin, book, and student databases
* Week 10: Implementation and Testing
* Days 1-2: Complete database design and create database tables.
* Day 3: Establish connections within the system.
* Days 4-5: Develop the login system and user authentication.
* Day 6-7: Design the main user interface for administrators.
* Week 11: Finalization and Presentation
* Day 1: Implementing "Add Book" feature, allowing administrators to input book details.
* Day 2: Implement "Delete Book" and "Search Book" features to manage the library's collection.
* Day 3: Work on "Issue Book" functionality, allowing the system to record book borrowings.
* Day 4: Implement the "Return Book" feature, updating the database when books are returned.
* Day 5: Conduct comprehensive testing to ensure all functions work smoothly.
* Day 6: Make any necessary adjustments based on testing results.
* Day 7: Prepare and finalize the project.
* Week 12: Reserve Week

## Key Features and Functionalities

Here in this section I have mentioned the exciting features and functionalities that our software will do. These features are the heart and soul of our digital library management system - they're what will make our project reliable.

* User Authentication:
* Reason (Why): This feature is designed to ensure that only authorized librarians can access the system, enhancing security and control.
* Description (What it does): With this feature, each librarian will have their unique login credentials, providing secure access to the system.
* Database Management:
* Reason (Why): This feature is crucial to keep our data well-organized, making it easier for librarians to manage and retrieve information.
* Description (What it does): We'll establish separate databases for books, administrators, and students, streamlining data management and retrieval processes for librarians.
* Add Book and Delete Book:
* Reason (Why): These features empower librarians to efficiently update the library's collection by adding new books and removing those no longer available.
* Description (What it does): Librarians will effortlessly add new books to the library catalog and remove books from the records when necessary
* Search Book and Show All Books:
* Reason (Why): These features enhance librarians' ability to find and manage books within the library.
* Description (What it does): Librarians can search for books using various criteria like title, author, or ISBN, making book management a breeze. They can also access a comprehensive list of all available books.
* Issue Book and Return Book:
* Reason (Why): These features simplify book transactions, allowing librarians to efficiently manage book borrowings and returns.
* Description (What it does): Librarians can easily facilitate book borrowings for students and update records when books are returned.

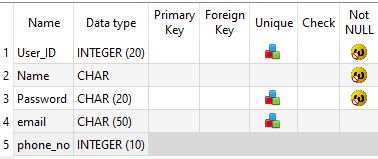
These features represent the backbone of our software project. They have been thoughtfully designed to align perfectly with our project objectives. Together, they will contribute to the creation of a robust, secure, and user-friendly library management system, benefitting both librarians and students alike

.

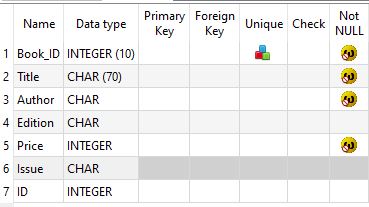
## Data Structure and Table Design

In this section I have mentioned the inner workings of how I have structured and organized our data. This is like the blueprint for the library management system's brain, where we decide how to store and manage information effectively.

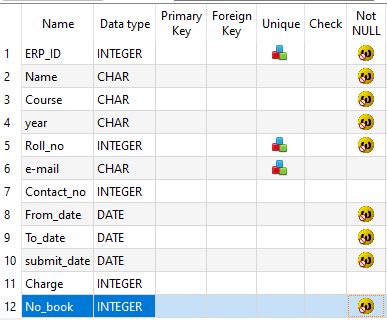
Admin Database:



Book Database:



Student Database:



Relationships:

* Between the Book Database and Student Database: The Book ID in the Student Database will help identify which student has borrowed a specific book.

Data Flow:

* When a librarian adds a new book to the library, it will be entered into the Book Database with a unique Book ID.
* If a student borrows a book, the Student Database will record the Book ID to establish which student has the book.
* Upon returning the book, the Book Database will update the Availability field to indicate that the book is once again available on the shelf.

## PROJECT-CODE

## from tkinter import \*

## from datetime import date

## from tkinter import ttk

## from tkinter import messagebox

## from tkcalendar import \*

## import sqlite3

## import time

## import datetime

## from datetime import datetime

## from datetime import timedelta

## import smtplib

## db=sqlite3.connect('admin.db')

## dd=sqlite3.connect('storebook.db')

## dc=sqlite3.connect('students.db')

## root = Tk()

## root.title("Library Management System")

## root.iconbitmap("aa.ico")

## root.geometry("900x500+300+150")

## root.resizable(0, 0)

## class maincode:

## def login(self):

## self.var1 = self.e1.get()

## self.var2 = self.e2.get()

## cursor=db.cursor()

## cursor.execute("SELECT \* FROM adm WHERE User\_ID='"+self.var1+"' and Password='"+self.var2+"'")

## db.commit()

## self.ab = cursor.fetchone()

## if self.ab!=None:

## #messagebox.showinfo('Library System',ab[1])

## self.under\_fm=Frame(root,height=500,width=900,bg='#fff')

## self.under\_fm.place(x=0,y=0)

## self.fm2=Frame(root,bg='#0f624c',height=80,width=900)

## self.fm2.place(x=0,y=0)

## # lgo=Canvas(fm2,bg='#0f624c',height=200,width=100,bd=4,relief='flat')

## # lgo.place(x=0,y=0)

## self.lbb=Label(self.fm2,bg='#0f624c')

## self.lbb.place(x=15,y=5)

## self.ig=PhotoImage(file='library.png')

## self.lbb.config(image=self.ig)

## self.lb3=Label(self.fm2,text='DASHBOARD',fg='White',bg='#0f624c',font=('Arial',30,'bold'))

## self.lb3.place(x=325,y=17)

## #----------------------------name------------------------

## self.name=Label(root,text="Name : ",bg='#fff',fg="black",font=('Arial',10,'bold'))

## self.name.place(x=5,y=83)

## self.name1=Label(root,text=self.ab[1],fg='black',bg='#fff',font=('Arial',10,'bold'))

## self.name1.place(x=60,y=83)

## #------------------------date-------------------------

## self.today=date.today()

## self.dat=Label(root,text='Date : ',bg='#fff',fg='black',font=('Arial', 10, 'bold'))

## self.dat.place(x=740,y=83)

## self.dat2 = Label(root, text=self.today, bg='#fff', fg='black', font=('Arial', 10, 'bold'))

## self.dat2.place(x=790, y=83)

## self.cur()

## else:

## messagebox.showerror('Library System', 'Your ID or Password is not Valid')

## #---------------------------------------------------------

## def cur(self):

## self.fm3=Frame(root,bg='#fff',width=900,height=390)

## self.fm3.place(x=0,y=110)

## #------------------------Clock---------------------------

## def clock():

## h = str(time.strftime("%H"))

## m = str(time.strftime("%M"))

## s = str(time.strftime("%S"))

## if int(h) >=12 and int(m) >=0:

## self.lb7\_hr.config(text="PM")

## #if int(h) > 12:

## #h = str(int(h) // 12)

## self.lb1\_hr.config(text=h)

## self.lb3\_hr.config(text=m)

## self.lb5\_hr.config(text=s)

## self.lb1\_hr.after(200, clock)

## self.lb1\_hr = Label(self.fm3, text='12', font=('times new roman', 20, 'bold'), bg='#fc1c1c', fg='white')

## self.lb1\_hr.place(x=560, y=0, width=60, height=30)

## self.lb3\_hr = Label(self.fm3, text='05', font=('times new roman', 20, 'bold'), bg='#0ee38b', fg='white')

## self.lb3\_hr.place(x=630, y=0, width=60, height=30)

## self.lb5\_hr = Label(self.fm3, text='37', font=('times new roman', 20, 'bold'), bg='#2b1dff', fg='white')

## self.lb5\_hr.place(x=700, y=0, width=60, height=30)

## self.lb7\_hr = Label(self.fm3, text='AM', font=('times new roman', 17, 'bold'), bg='#2b1dff', fg='white')

## self.lb7\_hr.place(x=770, y=0, width=60, height=30)

## clock()

## #-------------------------------clock closed------------------------

## self.canvas8 = Canvas(self.fm3, bg='black', width=400, height=300)

## self.canvas8.place(x=475, y=37)

## self.photo9=PhotoImage(file="bb.png")

## self.canvas8.create\_image(0,0,image=self.photo9,anchor=NW)

## # develop name--------------------

## 

## self.develop=Label(self.fm3,text='Developed By - Dev and Talha',bg='#fff',fg='blue',

## font=('Cursive',12,'italic','bold'))

## self.develop.place(x=600,y=350)

## #-----------------addbutton-----------------

## self.bt1=Button(self.fm3,text=' Add Books',fg='#fff',bg='#ff0076',font=('Arial',15,'bold'),width=170,

## height=0,bd=7,relief='flat',command=self.addbook,cursor='hand2')

## self.bt1.place(x=40,y=40)

## self.logo = PhotoImage(file='bt1.png')

## self.bt1.config(image=self.logo, compound=LEFT)

## self.small\_logo = self.logo.subsample(1,1)

## self.bt1.config(image=self.small\_logo)

## #-------------------------Issuebutton--------------

## self.bt2 = Button(self.fm3, text=' Issue Books', fg='#fff', bg='#ff0076', font=('Arial', 15, 'bold'),

## width=170,height=0, bd=7,relief='flat',command=self.issuebook,cursor='hand2')

## self.bt2.place(x=250, y=40)

## self.log = PhotoImage(file='bt2.png')

## self.bt2.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(1, 1)

## self.bt2.config(image=self.small\_log)

## #---------------------------Editbutton----------------

## self.bt3 = Button(self.fm3, text=' Edit Books', fg='#fff', bg='#ff0076', font=('Arial', 15, 'bold'),

## width=170,height=0,bd=7,relief='flat',cursor='hand2',command=self.edit)

## self.bt3.place(x=40, y=120)

## self.logb = PhotoImage(file='bt3.png')

## self.bt3.config(image=self.logb, compound=LEFT)

## self.small\_logb = self.logb.subsample(1, 1)

## self.bt3.config(image=self.small\_logb)

## #-----------------------------Returnbutton----------------

## self.bt4 = Button(self.fm3, text=' Return Books', fg='#fff', bg='#ff0076', font=('Arial', 15, 'bold'),

## width=170,height=0,bd=7,relief='flat',cursor='hand2',command=self.return\_book)

## self.bt4.place(x=250, y=120)

## self.log4 = PhotoImage(file='bt4.png')

## self.bt4.config(image=self.log4, compound=LEFT)

## self.small\_log4 = self.log4.subsample(1, 1)

## self.bt4.config(image=self.small\_log4)

## #----------------------Deletebutton---------------------

## self.bt5 = Button(self.fm3, text=' Delete Books', fg='#fff', bg='#ff0076', font=('Arial', 15, 'bold'),

## width=170,height=0,bd=7,relief='flat',cursor='hand2',command=self.delete)

## self.bt5.place(x=40, y=200)

## self.log5 = PhotoImage(file='bt5.png')

## self.bt5.config(image=self.log5, compound=LEFT)

## self.small\_log5 = self.log5.subsample(1, 1)

## self.bt5.config(image=self.small\_log5)

## #--------------------Show Button-----------------------------

## self.bt6 = Button(self.fm3, text=' Show Books', fg='#fff', bg='#ff0076', font=('Arial', 15, 'bold'),

## width=170,height=0,bd=7, relief='flat',cursor='hand2',command=self.show)

## self.bt6.place(x=250, y=200)

## self.log6 = PhotoImage(file='bt6.png')

## self.bt6.config(image=self.log6, compound=LEFT)

## self.small\_log6 = self.log6.subsample(1, 1)

## self.bt6.config(image=self.small\_log6)

## #-------------------------Seearch Button------------------

## self.bt7 = Button(self.fm3, text=' Search Books', fg='#fff', bg='#ff0076', font=('Arial', 15, 'bold'),

## width=170,height=0,bd=7, relief='flat',cursor='hand2',command=self.search)

## self.bt7.place(x=40, y=280)

## self.log7 = PhotoImage(file='bt7.png')

## self.bt7.config(image=self.log7, compound=LEFT)

## self.small\_log7 = self.log7.subsample(1, 1)

## self.bt7.config(image=self.small\_log7)

## #---------------------Exit Button-----------------------

## try:

## self.bt8 = Button(self.fm3, text=' log Out', fg='#fff', bg='#ff0076', font=('Arial', 15, 'bold'),

## width=170,

## height=0, bd=7, relief='flat',cursor='hand2',command=self.code)

## self.bt8.place(x=250, y=280)

## self.log8 = PhotoImage(file='bt8.png')

## self.bt8.config(image=self.log8, compound=LEFT)

## self.small\_log8 = self.log8.subsample(1, 1)

## self.bt8.config(image=self.small\_log8)

## except:

## self.bt9 = ttk.Button(self.fm3, text="ram", bg='#11d09a', font=('Arial', 15, 'bold'), width=150,

## height=0)

## self.bt9.place(x=40, y=350)

## self.log9 = PhotoImage(file='bt8.png')

## self.bt9.config(image=self.log9, compound=LEFT)

## self.small\_log9 = self.log9.subsample(3, 3)

## self.bt9.config(image=self.small\_log9)

## def mainclear(self):

## self.e1.delete(0,END)

## self.e2.delete(0,END)

## #-----------------------button add book----------------------

## def addbook(self):

## class temp(maincode):

## def book(self):

## self.fm=Frame(root,bg='#a7ecd9',width=900,height=390)

## self.fm.place(x=0,y=110)

## self.fm1=Frame(self.fm,bg='#fff',width=500,height=360,bd=5,relief='flat')

## self.fm1.place(x=200,y=15)

## self.backbt = Button(self.fm, width=60, bg='#a7ecd9',activebackground='#a7ecd9', bd=0, relief='flat',

## command=self.cur)

## self.backbt.place(x=0, y=0)

## self.log = PhotoImage(file='back.png')

## self.backbt.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(1, 1)

## self.backbt.config(image=self.small\_log)

## #---------------------------Label---------------------------------

## self.f=Frame(self.fm1,bg='#0f624c',width=490,height=35)

## self.f.place(x=0,y=0)

## self.ll=Label(self.f,text='ADD BOOKS',fg='#fff',bg='#0f624c',font=('Arial',12,'bold'))

## self.ll.place(x=200,y=6)

## self.lb=Label(self.fm1,text='ID',fg='black',bg='#fff',font=('Arial',10,'bold'))

## self.lb.place(x=70,y=90)

## self.lb2 = Label(self.fm1, text='Title', fg='black', bg='#fff', font=('Arial', 10, 'bold'))

## self.lb2.place(x=70, y=130)

## self.lb3 = Label(self.fm1, text='Author', fg='black', bg='#fff', font=('Arial', 10, 'bold'))

## self.lb3.place(x=70, y=170)

## self.lb4= Label(self.fm1, text='Edition', fg='black', bg='#fff', font=('Arial', 10, 'bold'))

## self.lb4.place(x=70, y=210)

## self.lb5 = Label(self.fm1, text='Price', fg='black', bg='#fff', font=('Arial', 10, 'bold'))

## self.lb5.place(x=70, y=250)

## #-------------------------------Entry-------------------------------------

## self.ee1=Entry(self.fm1,width=25,bd=4,relief='groove',font=('arial',12,'bold'))

## self.ee1.place(x=180,y=88)

## self.ee2=Entry(self.fm1,width=25,bd=4,relief='groove',font=('arial',12,'bold'))

## self.ee2.place(x=180,y=130)

## self.ee3=Entry(self.fm1,width=25,bd=4,relief='groove',font=('arial',12,'bold'))

## self.ee3.place(x=180,y=170)

## self.ee4=Entry(self.fm1,width=25,bd=4,relief='groove',font=('arial',12,'bold'))

## self.ee4.place(x=180,y=210)

## self.ee5=Entry(self.fm1,width=25,bd=4,relief='groove',font=('arial',12,'bold'))

## self.ee5.place(x=180,y=250)

## self.bt=Button(self.fm1,text='Submit',width=41,bg='red',fg='#fff',font=('Arial',10,'bold'),bd=5,

## relief='flat',command=self.submit1)

## self.bt.place(x=70,y=290)

## #---------------------Back button----------------------------------

## def submit1(self):

## self.id=self.ee1.get()

## self.ttl=self.ee2.get()

## self.aut=self.ee3.get()

## self.edi=self.ee4.get()

## self.pri=self.ee5.get()

## cursor=dd.cursor()

## cursor.execute("INSERT INTO stbook(Book\_ID,Title,Author,Edition,Price) values(?,?,?,?,?)",(self.id,

## self.ttl,self.aut,self.edi,self.pri))

## dd.commit()

## self.clear()

## def clear(self):

## self.ee1.delete(0,END)

## self.ee2.delete(0,END)

## self.ee3.delete(0,END)

## self.ee4.delete(0,END)

## self.ee5.delete(0,END)

## obj=temp()

## obj.book()

## #-----------xxxxxxxxxxxx--------close add book---xxxxxxxxxxxxxxxxxxxx---------------

## #--------------------------------Issue Books---------------------------------

## def issuebook(self):

## class test(maincode):

## max=0

## n = 1

## def issue(self):

## self.f = Frame(root, bg='#a7ecd9', width=900, height=390)

## self.f.place(x=0, y=110)

## self.fmi=Canvas(self.f,bg='#fff',width=900,height=390,bd=0,relief='flat')

## self.fmi.place(x=0,y=0)

## self.fc=Frame(self.fmi,bg='#fff',width=330,height=230,bd=4,relief='flat')

## self.fc.place(x=70,y=20)

## self.ffb=Frame(self.fc,bg='#0f624c',bd=2,relief='flat',width=330,height=35)

## self.ffb.place(x=0,y=0)

## self.lc=Label(self.ffb,text='STUDENT INFORMATION',bg='#0f624c',fg='#fff',font=('Arial',12,'bold'))

## self.lc.place(x=55,y=5)

## self.lb=Label(self.fc,text='Roll-No',bg='#fff',fg='black',font=('Arial',10,'bold'))

## self.lb.place(x=15,y=60)

## self.ob=Label(self.fc,text='or',bg='#fff',fg='black',font=('cursive',12,'bold'))

## self.ob.place(x=180,y=90)

## self.em = Entry(self.fc, width=30, bd=5, relief='ridge', font=('Arial', 8, 'bold'))

## self.em.place(x=105, y=60)

## self.lb = Label(self.fc, text='ERP-ID', bg='#fff', fg='black', font=('Arial', 10, 'bold'))

## self.lb.place(x=15, y=120)

## self.em2 = Entry(self.fc, width=30, bd=5, relief='ridge', font=('Arial', 8, 'bold'))

## self.em2.place(x=105, y=120)

## self.bt = Button(self.fc, text='Submit', width=14, bg='red', fg='#fff', font=('Arial', 10, 'bold'),

## bd=5,relief='flat',command=self.check)

## self.bt.place(x=15,y=180)

## self.bt3=Button(self.fc,text='Clear',width=14,bg='blue',fg='#fff',font=('arial',10,'bold'),bd=5,

## relief='flat',command=self.clr)

## self.bt3.place(x=165,y=180)

## self.backbt = Button(self.fmi,width=60, bg='#fff',activebackground='#fff',bd=0, relief='flat',

## command=self.cur)

## self.backbt.place(x=5, y=5)

## self.log = PhotoImage(file='back.png')

## self.backbt.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(1, 1)

## self.backbt.config(image=self.small\_log)

## def check(self):

## self.ai=self.em.get()

## self.b=self.em2.get()

## cursor=dc.cursor()

## cursor.execute("SELECT \* FROM student WHERE Roll\_no='"+self.ai+"' or ERP\_ID='"+self.b+"'")

## self.var=cursor.fetchone()

## if self.var!=None:

## self.lb1=Label(self.fmi,text='Name :',fg='black',font=('Arial',10,'bold'))

## self.lb1.place(x=60,y=255)

## self.lb2 = Label(self.fmi, text=self.var[1], fg='black', font=('Arial', 10, 'bold'))

## self.lb2.place(x=130, y=255)

## self.lb3 = Label(self.fmi, text='Course :',fg='black', font=('Arial', 10, 'bold'))

## self.lb3.place(x=60, y=275)

## self.lb4 = Label(self.fmi, text=self.var[2],fg='black', font=('Arial', 10, 'bold'))

## self.lb4.place(x=130, y=275)

## self.lb5 = Label(self.fmi, text='Year :', fg='black', font=('Arial', 10, 'bold'))

## self.lb5.place(x=60, y=295)

## self.lb6 = Label(self.fmi, text=self.var[3], fg='black', font=('Arial', 10, 'bold'))

## self.lb6.place(x=130, y=295)

## self.lb7 = Label(self.fmi, text='Contact :', fg='black', font=('Arial', 10, 'bold'))

## self.lb7.place(x=60, y=315)

## self.lb8 = Label(self.fmi, text=self.var[6],fg='black', font=('Arial', 10, 'bold'))

## self.lb8.place(x=130, y=315)

## self.lb9 = Label(self.fmi, text='College :', fg='black', font=('Arial', 10, 'bold'))

## self.lb9.place(x=60, y=335)

## self.lb10 = Label(self.fmi, text=self.var[7],fg='black', font=('Arial', 10, 'bold'))

## self.lb10.place(x=130, y=335)

## self.fr=Frame(self.fmi,bg='#fff',bd=5,relief='flat',width=450,height=320)

## self.fr.place(x=420,y=20)

## self.ff=Frame(self.fr,bg='#0f624c',bd=2,relief='flat',width=450,height=35)

## self.ff.place(x=0,y=0)

## self.lb=Label(self.ff,text='ISSUE BOOK',bg='#0f624c',fg='#fff',font=('Arial',12,'bold'))

## self.lb.place(x=165,y=5)

## self.tt=Label(self.fr,text='Book-ID',bg='#fff',fg='black',font=('arial',10,'bold'))

## self.tt.place(x=50,y=60)

## self.e1 = Entry(self.fr, width=30, bd=5, relief='ridge', font=('Arial', 8, 'bold'))

## self.e1.place(x=160, y=60)

## self.ttp = Label(self.fr, text='Title', bg='#fff', fg='black', font=('arial', 10, 'bold'))

## self.ttp.place(x=50, y=110)

## self.e2 = Entry(self.fr, width=30, bd=5, relief='ridge', font=('Arial', 8, 'bold'))

## self.e2.place(x=160, y=110)

## self.bt1 = Button(self.fr, text='Submit', width=35, bg='#0f624c', fg='#fff', font=('Arial', 10,

## 'bold'),bd=5,relief='flat',command=self.data)

## self.bt1.place(x=60, y=160)

## '''self.bt1 = Button(self.fr, text='Clear', width=13, bg='blue', fg='#fff', font=('Arial', 10,

## 'bold'), bd=5,

## relief='flat', command=self.clr1)

## self.bt1.place(x=215, y=160)'''

## else:

## messagebox.showwarning('Warning','These Student are not Registered !')

## def clr(self):

## self.em.delete(0, END)

## self.em2.delete(0, END)

## '''def clr1(self):

## self.e1.delete(0,END)

## self.e2.delete(0,END)

## self.boot.destroy()

## self.data()'''

## def data(self):

## self.vva=self.e1.get()

## self.vvb=self.e2.get()

## cursor=dd.cursor()

## cursor.execute("SELECT \* FROM stbook WHERE Book\_ID='"+self.vva+"' and Title='"+self.vvb+"'")

## dd.commit()

## self.value=cursor.fetchone()

## if self.value!=None:

## if self.max==0:

## self.boot=Tk()

## self.boot.title("Issue Books")

## self.boot.iconbitmap("aa.ico")

## self.boot.configure(bg='#fff')

## self.boot.geometry("300x680+1202+50")

## self.boot.resizable(0,0)

## self.lb=Label(self.boot,text='Title',bg='#fff',fg='black',font=('Arial',10,'bold'))

## self.lb.place(x=30,y=30)

## self.lbn = Label(self.boot, text=self.value[1], bg='#fff', fg='black', font=('Arial', 10, 'bold'))

## self.lbn.place(x=120,y=30)

## self.lb = Label(self.boot, text='Author', bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lb.place(x=30, y=60)

## self.lbn = Label(self.boot, text=self.value[2], bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbn.place(x=120, y=60)

## self.lb = Label(self.boot, text='Edition', bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lb.place(x=30, y=90)

## self.lbn = Label(self.boot, text=self.value[3], bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbn.place(x=120, y=90)

## self.plan = Label(self.boot, text='---------------------------------------------------',

## bg='#fff')

## self.plan.place(x=15, y=120)

## self.planx = Label(self.boot, text='---------------------------------------------------',

## bg='#fff')

## self.planx.place(x=15, y=240)

## self.planx = Label(self.boot, text='---------------------------------------------------',

## bg='#fff')

## self.planx.place(x=15, y=360)

## if self.max==1:

## self.lbt = Label(self.boot, text='Title', bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbt.place(x=30, y=150)

## self.lbnt = Label(self.boot, text=self.value[1], bg='#fff', fg='black', font=('Arial',

## 10, 'bold'))

## self.lbnt.place(x=120, y=150)

## self.lbtd = Label(self.boot, text='Author', bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbtd.place(x=30, y=180)

## self.lbn = Label(self.boot, text=self.value[2], bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbn.place(x=120, y=180)

## self.lbc = Label(self.boot, text='Edition', bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbc.place(x=30, y=210)

## self.lbn = Label(self.boot, text=self.value[3], bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbn.place(x=120, y=210)

## if self.max==2:

## self.lbt = Label(self.boot, text='Title', bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbt.place(x=30, y=270)

## self.lbnt = Label(self.boot, text=self.value[1], bg='#fff', fg='black', font=('Arial',

## 10, 'bold'))

## self.lbnt.place(x=120, y=270)

## self.lbtd = Label(self.boot, text='Author', bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbtd.place(x=30, y=300)

## self.lbn = Label(self.boot, text=self.value[2], bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbn.place(x=120, y=300)

## self.lbc = Label(self.boot, text='Edition', bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbc.place(x=30, y=330)

## self.lbn = Label(self.boot, text=self.value[3], bg='#fff', fg='black', font=('Arial', 10,

## 'bold'))

## self.lbn.place(x=120, y=330)

## if self.max>=3:

## messagebox.showerror('Library System','SIR, MINIMUM 3 BOOKS ARE REQUIRED!')

## self.label = Label(self.fr, text='ADD MORE BOOKS ', bg='#fff', fg='black', font=('arial', 10,

## 'bold'))

## self.label.place(x=60, y=220)

## #---------------------------------Radio Button-------------------------

## self.it1=Radiobutton(self.fr,text='YES',bg='#fff',variable='radio',value=1,command=self.yes)

## self.it1.place(x=210,y=220)

## self.it2 = Radiobutton(self.fr, text='NO',bg='#fff', variable='radio', value=2,command=self.no)

## self.it2.place(x=280, y=220)

## #------------------------ISSUED button-----------------------------

## self.button1 = Button(self.boot, text='Issued', bg='red', fg='#fff', width=30, height=0,

## font=('Arial', 8, 'bold'), command=self.issued)

## self.button1.place(x=30, y=610)

## #-----------------------date module uses-------------------------

## self.x = date.today()

## self.cal = Calendar(self.boot, selectmode="day", bg='black',year=self.x.year ,month=self.x.month ,day=self.x.day)

## self.cal.place(x=20,y=380)

## btn1 = Button(self.boot, text="Confirm Date",command=self.get\_data, bg='#ff0076',

## font=('arial', 10,

## 'bold'),

## fg='#fff', relief='flat')

## btn1.place(x=90,y=575)

## self.boot.mainloop()

## else:

## messagebox.showwarning('Warning','YOUR DATA IS NOT FOUND !')

## def get\_data(self):

## self.datecon=self.cal.selection\_get()

## def yes(self):

## self.n=self.n+1

## self.bt1 = Button(self.fr, text='Submit', width=35, bg='#0f624c', fg='#fff', font=('Arial', 10,

## 'bold'), bd=5,relief='flat',command=self.data, state=ACTIVE)

## self.bt1.place(x=60, y=160)

## self.e1.delete(0, END)

## self.e2.delete(0, END)

## self.max=self.max+1

## def no(self):

## self.bt1 = Button(self.fr, text='Submit', width=35, bg='#0f624c', fg='#fff', font=('Arial', 10,

## 'bold'), bd=5,relief='flat',state=DISABLED)

## self.bt1.place(x=60, y=160)

## def issued(self):

## self.ac=self.e1.get()

## cursor=dd.cursor()

## cursor.execute("UPDATE stbook SET Issue='Issued', ID='"+self.b+"' WHERE "

## "Book\_ID='"+self.ac+"'")

## dd.commit()

## if self.n<=3:

## book=dc.cursor()

## book.execute("UPDATE student SET No\_book='"+str(self.n)+"' WHERE Roll\_no='"+self.ai+"' or "

## "ERP\_ID='"+self.b+"' ")

## dc.commit()

## comm=dc.cursor()

## comm.execute("UPDATE student SET From\_date='"+str(self.x)+"', To\_date='"+str(self.datecon)+"' "

## "WHERE Roll\_no='"+self.ai+"' or ERP\_ID='"+self.b+"'")

## dc.commit()

## messagebox.showinfo('Library System', 'YOUR BOOK ISSUED')

## obk=test()

## obk.issue()

## def edit(self):

## class editing(maincode):

## def edbooks(self):

## self.ffm=Frame(root,bg='#a7ecd9',width=900,height=390)

## self.ffm.place(x=0,y=110)

## self.fm1 = Frame(self.ffm, bg='#fff', width=500, height=200, bd=5, relief='flat')

## self.fm1.place(x=200, y=15)

## self.ed = Frame(self.fm1, bg='#0f624c', bd=0, relief='flat', width=490, height=35)

## self.ed.place(x=0,y=0)

## self.lab = Label(self.ed, text='EDIT BOOKS DETAILS', bg='#0f624c', fg='#fff', font=('Arial', 12,

## 'bold'))

## self.lab.place(x=165, y=5)

## self.label3=Label(self.fm1,text='Book ID',bg='#fff',fg='black',font=('arial',10,'bold'))

## self.label3.place(x=85,y=65)

## self.entry=Entry(self.fm1,width=30,bd=4,relief='groove',font=('arial',8,'bold'))

## self.entry.place(x=188,y=65)

## self.button7 = Button(self.fm1, text='Search', bg='#0f624c', fg='#fff', width=24, height=0,

## font=('Arial', 10, 'bold'),command=self.search)

## self.button7.place(x=140,y=120)

## self.backbt = Button(self.ffm, width=60, bg='#a7ecd9',activebackground='#a7ecd9',

## bd=0, relief='flat', command=self.cur)

## self.backbt.place(x=0, y=0)

## self.log = PhotoImage(file='back.png')

## self.backbt.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(1, 1)

## self.backbt.config(image=self.small\_log)

## #----------------------Database----------------------------------

## def search(self):

## self.datas=self.entry.get()

## cursor=dd.cursor()

## cursor.execute("SELECT \* FROM stbook WHERE Book\_ID='"+self.datas+"'" )

## dd.commit()

## self.val=cursor.fetchone()

## if self.val!=None:

## self.edcat=Tk()

## self.edcat.title("Library System")

## self.edcat.geometry("300x320+590+320")

## self.edcat.configure(bg='#fff')

## self.edcat.iconbitmap("aa.ico")

## self.fc=Frame(self.edcat,bg='#0f624c',width=300,height=30)

## self.fc.place(x=0,y=0)

## self.lab=Label(self.fc,bg='#0f624c',fg='#fff',text='EDIT BOOKS',font=('arial',10,'bold'))

## self.lab.place(x=112,y=5)

## self.labid = Label(self.edcat, bg='#fff', fg='black', text='Book ID', font=('arial', 10,

## 'bold'))

## self.labid.place(x=30, y=45)

## self.labti = Label(self.edcat, bg='#fff', fg='black', text='Title', font=('arial', 10,

## 'bold'))

## self.labti.place(x=30, y=90)

## self.labaut = Label(self.edcat, bg='#fff', fg='black', text='Author', font=('arial', 10,

## 'bold'))

## self.labaut.place(x=30, y=135)

## self.labed = Label(self.edcat, bg='#fff', fg='black', text='Edition', font=('arial', 10,

## 'bold'))

## self.labed.place(x=30, y=180)

## self.labpr = Label(self.edcat, bg='#fff', fg='black', text='Price', font=('arial', 10,

## 'bold'))

## self.labpr.place(x=30, y=225)

## #------------------------------Entry------------------------

## self.en1=Entry(self.edcat,width=25,bd=4,relief='groove',font=('arial',8,'bold'))

## self.en1.place(x=100,y=45)

## self.en2 = Entry(self.edcat, width=25, bd=4, relief='groove',font=('arial',8,'bold'))

## self.en2.place(x=100, y=90)

## self.en3 = Entry(self.edcat, width=25, bd=4, relief='groove',font=('arial',8,'bold'))

## self.en3.place(x=100, y=135)

## self.en4 = Entry(self.edcat, width=25, bd=4, relief='groove',font=('arial',8,'bold'))

## self.en4.place(x=100, y=180)

## self.en5 = Entry(self.edcat, width=25, bd=4, relief='groove',font=('arial',8,'bold'))

## self.en5.place(x=100, y=225)

## self.butt = Button(self.edcat, text='Submit', bg='#0f624c', fg='#fff', width=20, height=0,

## font=('Arial', 10, 'bold'),command=self.savedit)

## self.butt.place(x=67, y=270)

## # -------------------insert value within edcat windows--------------------

## self.en1.insert(0, self.val[0])

## self.en2.insert(0, self.val[1])

## self.en3.insert(0, self.val[2])

## self.en4.insert(0, self.val[3])

## self.en5.insert(0, self.val[4])

## self.edcat.mainloop()

## else:

## messagebox.showerror('Library System','PLEASE! CORRECT BOOK ID')

## #-----------------BOKK is Updated-----------------

## def savedit(self):

## self.id = self.en1.get()

## self.ti = self.en2.get()

## self.au = self.en3.get()

## self.ed = self.en4.get()

## self.pi = self.en5.get()

## cursor= dd.cursor()

## cursor.execute("UPDATE stbook SET Book\_ID='"+self.id+"', Title='"+self.ti+"',Author='"+self.au+"',Edition='"+self.ed+"',Price='"+self.pi+"' WHERE Book\_ID='"+self.datas+"'")

## dd.commit()

## messagebox.showinfo('Library System','YOUR DATA IS UPDATED!')

## obj=editing()

## obj.edbooks()

## # -----------------------------------------------------------------------------------------------------

## # ------------------------------Return Book--------------------------------------------------

## def return\_book(self):

## class retu(maincode):

## def \_\_init\_\_(self):

## self.frame=Frame(root,bd=0,relief='flat',bg='#a7ecd9',width=900,height=390)

## self.frame.place(x=0,y=110)

## self.f1 = Frame(self.frame, bg='#fff', width=500, height=200, bd=5, relief='flat')

## self.f1.place(x=200, y=15)

## self.ed = Frame(self.f1, bg='#0f624c', bd=0, relief='flat', width=490, height=35)

## self.ed.place(x=0, y=0)

## self.lac = Label(self.ed, text='RETURN BOOKS ', bg='#0f624c', fg='#fff', font=('Arial', 12, 'bold'))

## self.lac.place(x=175, y=5)

## self.label8 = Label(self.f1, text='ERP ID', bg='#fff', fg='black', font=('arial', 10, 'bold'))

## self.label8.place(x=85, y=65)

## self.entry4 = Entry(self.f1, width=30, bd=4, relief='groove', font=('arial', 8, 'bold'))

## self.entry4.place(x=188, y=65)

## self.button9 = Button(self.f1, text='Return', bg='#0f624c', fg='#fff', width=24, height=0,

## font=('Arial', 10, 'bold'),command=self.retbook)

## self.button9.place(x=140, y=120)

## self.backbt = Button(self.frame, width=60, bg='#a7ecd9', activebackground='#a7ecd9',

## bd=0, relief='flat', command=self.cur)

## self.backbt.place(x=0, y=0)

## self.log = PhotoImage(file='back.png')

## self.backbt.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(1, 1)

## self.backbt.config(image=self.small\_log)

## def retbook(self):

## self.charge=0

## self.entry=self.entry4.get()

## cursor=dc.cursor()

## cursor.execute("SELECT \* FROM student WHERE ERP\_id='"+self.entry+"'")

## dc.commit()

## self.data=cursor.fetchone()

## if self.data!=None:

## self.get\_date = date.today()

## cursor = dc.cursor()

## cursor.execute("UPDATE student SET submit\_date='" + str(

## self.get\_date) + "' WHERE ERP\_ID='" + self.entry + "'")

## dc.commit()

## cursor=dd.cursor()

## cursor.execute("UPDATE stbook SET Issue='', ID='' WHERE ID='"+self.entry+"'")

## dd.commit()

## from datetime import datetime

## self.tom=Tk()

## self.tom.geometry("300x250+590+348")

## self.tom.iconbitmap("aa.ico")

## self.tom.title("Library System")

## self.tom.resizable(0,0)

## self.tom.configure(bg="#fff")

## cursor=dc.cursor()

## cursor.execute("SELECT \* FROM student WHERE ERP\_ID='"+self.entry+"'")

## dc.commit()

## self.var=cursor.fetchone()

## if self.var!=None:

## #-----------------between two date calculate days---------------------

## 

## self.a=self.var[9]

## self.b=self.var[10]

## formatStr='%Y-%m-%d'

## try:

## delta1=datetime.strptime(self.a,formatStr)

## delta2=datetime.strptime(self.b, formatStr)

## delta=delta2-delta1

## chm=delta.days

## except:

## messagebox.showinfo('ERROR','It looks like you have given the wrong date or the student has issued no book.')

## chm=0

## #print(chm)

## #------------------calculate fine charge------------------

## self.lb=Label(self.tom,text="Fine Charge",bg="#fff",fg="black",font=('arial',17,'bold'))

## self.lb.place(x=75,y=60)

## if chm<=20:

## self.lc1 = Label(self.tom, text="0 Rs.", bg="#fff", fg="black", font=('arial', 12,

## 'bold'))

## self.lc1.place(x=120,y=120)

## else:

## self.charge=(1\*chm)\*self.var[12]

## #print(self.charge)

## self.lc2 = Label(self.tom, text=self.charge, bg="#fff", fg="black", font=('arial',12,

## 'bold'))

## self.lc2.place(x=110, y=120)

## self.lc3 = Label(self.tom, text='Rs.', bg="#fff", fg="black",

## font=('arial', 12, 'bold'))

## self.lc3.place(x=130, y=120)

## cursor1 = dc.cursor()

## cursor1.execute("UPDATE student SET From\_date='',To\_date='',submit\_date='',No\_book='',"

## "Charge='"+str(self.charge)+"' WHERE ERP\_ID='"+self.entry+"'")

## dc.commit()

## self.tom.mainloop()

## else:

## messagebox.showwarning("Library System","YOUR ERP\_ID IN NOT FOUND !")

## object=retu()

## #-----------------------------------------------------------------------------------------------

## #-------------------------------------Delete Books---------------------------------------------

## def delete(self):

## class dele(maincode):

## def deleteee(self):

## self.ff = Frame(root, bg='#a7ecd9', width=900, height=390)

## self.ff.place(x=0, y=110)

## self.f1 = Frame(self.ff, bg='#fff', width=500, height=200, bd=5, relief='flat')

## self.f1.place(x=200, y=15)

## self.ed = Frame(self.f1, bg='#0f624c', bd=0, relief='flat', width=490, height=35)

## self.ed.place(x=0, y=0)

## self.lac = Label(self.ed, text='DELETE BOOKS ', bg='#0f624c', fg='#fff', font=('Arial', 12,'bold'))

## self.lac.place(x=175, y=5)

## self.label8 = Label(self.f1, text='Book ID', bg='#fff', fg='black', font=('arial', 10, 'bold'))

## self.label8.place(x=85, y=65)

## self.entry4 = Entry(self.f1, width=30, bd=4, relief='groove', font=('arial', 8, 'bold'))

## self.entry4.place(x=188, y=65)

## self.button9 = Button(self.f1, text='Delete', bg='#0f624c', fg='#fff', width=24, height=0,

## font=('Arial', 10, 'bold'),command=self.deldata)

## self.button9.place(x=140, y=120)

## self.backbt = Button(self.ff,width=60, bg='#a7ecd9',activebackground='#a7ecd9',

## bd=0, relief='flat', command=self.cur)

## self.backbt.place(x=0, y=0)

## self.log = PhotoImage(file='back.png')

## self.backbt.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(1, 1)

## self.backbt.config(image=self.small\_log)

## def deldata(self):

## self.a=self.entry4.get()

## cursor=dd.cursor()

## cursor.execute("DELETE FROM stbook WHERE Book\_ID='"+self.a+"'")

## dd.commit()

## self.da=cursor.fetchone()

## if self.da!=None:

## messagebox.showinfo('Library System','YOUR DATA IS DELETED !')

## else:

## messagebox.showerror('Library System','YOUR DATA IS NOT FOUND !')

## occ=dele()

## occ.deleteee()

## #------------------------------------------------------------------------------------------------

## #---------------------------------------Search Books---------------------------------------------

## def search(self):

## class demt(maincode):

## def delmdata(self):

## self.fc = Frame(root, bg='#a7ecd9', width=900, height=390)

## self.fc.place(x=0, y=110)

## self.fc1 = Frame(self.fc, bg='#fff', width=500, height=200, bd=5, relief='flat')

## self.fc1.place(x=200, y=15)

## self.edm = Frame(self.fc1, bg='#0f624c', bd=0, relief='flat', width=490, height=35)

## self.edm.place(x=0, y=0)

## self.lac = Label(self.edm, text='SEARCH BOOKS ', bg='#0f624c', fg='#fff', font=('Arial', 12, 'bold'))

## self.lac.place(x=175, y=5)

## self.label8 = Label(self.fc1, text='Book ID', bg='#fff', fg='black', font=('arial', 10, 'bold'))

## self.label8.place(x=85, y=65)

## self.entryl= Entry(self.fc1, width=30, bd=4, relief='groove', font=('arial', 8, 'bold'))

## self.entryl.place(x=188, y=65)

## self.butto = Button(self.fc1, text='Search', bg='#0f624c', fg='#fff', width=24, height=0,

## font=('Arial', 10, 'bold'),command=self.srch)

## self.butto.place(x=140, y=120)

## self.backbt = Button(self.fc,width=60, bg='#a7ecd9',activebackground='#a7ecd9',bd=0, relief='flat', command=self.cur)

## self.backbt.place(x=0, y=0)

## self.log = PhotoImage(file='back.png')

## self.backbt.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(1, 1)

## self.backbt.config(image=self.small\_log)

## def srch(self):

## self.emp=self.entryl.get()

## cursor=dd.cursor()

## cursor.execute("SELECT \* FROM stbook WHERE Book\_ID='"+self.emp+"'")

## dd.commit()

## self.srval=cursor.fetchone()

## if self.srval!=None:

## self.top=Tk()

## self.top.title("Library System")

## self.top.iconbitmap("aa.ico")

## self.top.geometry("300x300+600+300")

## self.top.resizable(0, 0)

## self.top.configure(bg='#fff')

## self.frm=Frame(self.top,bg='#0f624c',width=300,height=35)

## self.frm.place(x=0,y=0)

## self.mnlb=Label(self.frm,bg='#0f624c',fg='#fff',text="Avaliable",font=('arial',11,'bold'))

## self.mnlb.place(x=120,y=5)

## self.lb1 = Label(self.top, text='Title', bg='#fff', fg='black', font=('arial', 12, 'bold'))

## self.lb1.place(x=40,y=80)

## self.lb2=Label(self.top,text=self.srval[1],bg='#fff',fg='black',font=('arial',12,'bold'))

## self.lb2.place(x=120,y=80)

## self.lb3 = Label(self.top, text='Author', bg='#fff', fg='black', font=('arial', 12, 'bold'))

## self.lb3.place(x=40, y=160)

## self.lb4 = Label(self.top, text=self.srval[2], bg='#fff', fg='black', font=('arial', 12, 'bold'))

## self.lb4.place(x=120, y=160)

## self.lb5 = Label(self.top, text='Edition', bg='#fff', fg='black', font=('arial', 12, 'bold'))

## self.lb5.place(x=40, y=240)

## self.lb6 = Label(self.top, text=self.srval[3], bg='#fff', fg='black', font=('arial', 12, 'bold'))

## self.lb6.place(x=120, y=240)

## else:

## messagebox.showwarning('Library System','YOUR DATA IS NOT AVAILABLE !')

## object=demt()

## object.delmdata()

## #-----------------------------------------------------------------------------------------------------

## #-------------------------------------------SHOW BOOKS\_------------------------------------------------

## def show(self):

## class tst(maincode):

## def \_\_init\_\_(self):

## self.fc = Frame(root, bg='#a7ecd9', width=900, height=390)

## self.fc.place(x=0, y=110)

## self.popframe=Frame(self.fc,width=900,height=30,bg='#0f624c')

## self.popframe.place(x=0,y=0)

## self.lbn=Label(self.popframe,bg='#0f624c',text='BOOKS INFORMATION',fg='#fff',font=('arial',10,

## 'bold'))

## self.lbn.place(x=380,y=5)

## self.backbt = Button(self.popframe,width=30, bg='#0f624c',activebackground='#0f624c',

## bd=0, relief='flat', command=self.cur)

## self.backbt.place(x=0, y=0)

## self.log = PhotoImage(file='back.png')

## self.backbt.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(2, 2)

## self.backbt.config(image=self.small\_log)

## self.table\_frame=Frame(self.fc,bg='#fff',bd=1,relief='flat')

## self.table\_frame.place(x=0,y=30,width=900,height=360)

## self.scroll\_x=Scrollbar(self.table\_frame,orient=HORIZONTAL)

## self.scroll\_y=Scrollbar(self.table\_frame,orient=VERTICAL)

## self.book\_table=ttk.Treeview(self.table\_frame,columns=("Book ID","Title","Author","Edition",

## "Price"),

## xscrollcommand=self.scroll\_x.set,yscrollcommand=self.scroll\_y.set)

## self.scroll\_x.pack(side=BOTTOM,fill=X)

## self.scroll\_y.pack(side=RIGHT, fill=Y)

## self.scroll\_x.config(command=self.book\_table.xview)

## self.scroll\_y.config(command=self.book\_table.yview)

## self.book\_table.heading("Book ID",text="Book ID")

## self.book\_table.heading("Title", text="Title")

## self.book\_table.heading("Author", text="Author")

## self.book\_table.heading("Edition", text="Edition")

## self.book\_table.heading("Price", text="Price")

## self.book\_table['show']='headings'

## self.book\_table.column("Book ID",width=200)

## self.book\_table.column("Title", width=200)

## self.book\_table.column("Author", width=200)

## self.book\_table.column("Edition", width=120)

## self.book\_table.column("Price", width=110)

## self.book\_table.pack(fill=BOTH,expand=1)

## self.fetch\_data()

## def fetch\_data(self):

## cursor=dd.cursor()

## cursor.execute("SELECT \* FROM stbook")

## self.rows=cursor.fetchall()

## if len(self.rows)!=0:

## for self.row in self.rows:

## self.book\_table.insert('',END,values=self.row)

## dd.commit()

## oc=tst()

## #-----------------------------------------------------------------------------------------

## #---------------------------------------LOGIN SYSTEM--------------------------------------

## def code(self):

## self.fm=Frame(root,height=500,width=900,bg='white')

## self.fm.place(x=0,y=0)

## self.canvas=Canvas(self.fm,height=500,width=900,bg='#22224b')

## self.canvas.place(x=0,y=0)

## self.photo=PhotoImage(file="images (17).png")

## self.canvas.create\_image(70,45,image=self.photo,anchor=NW)

## self.fm1=Frame(self.canvas,height=260,width=300,bg='white',bd=3,relief='ridge')

## self.fm1.place(x=300,y=170)

## self.photo1=PhotoImage(file="dd.png")

## self.canvas.create\_image(330,5,image=self.photo1,anchor=NW)

## self.b1=Label(self.fm1,text='User ID',bg='white',font=('Arial',10,'bold'))

## self.b1.place(x=20,y=42)

## self.e1=Entry(self.fm1,width=22,font=('arial',9,'bold'),bd=4,relief='groove')

## self.e1.place(x=100,y=40)

## self.lb2=Label(self.fm1,text='Password',bg='white',font=('Arial',10,'bold'))

## self.lb2.place(x=20,y=102)

## self.e2=Entry(self.fm1,width=22,show='\*',font=('arial',9,'bold'),bd=4,relief='groove')

## self.e2.place(x=100,y=100)

## self.btn1=Button(self.fm1,text=' login',fg='white',bg='red',width=100,font=('Arial',11,'bold'),

## activebackground='white',activeforeground='black',command=self.login,bd=3,relief='flat',cursor='hand2')

## self.btn1.place(x=25,y=160)

## self.logo = PhotoImage(file='user.png')

## self.btn1.config(image=self.logo, compound=LEFT)

## self.small\_logo = self.logo.subsample(1, 1)

## self.btn1.config(image=self.small\_logo)

## self.btn2=Button(self.fm1,text=' Clear',fg='white',bg='blue',width=100,font=('Arial',11,'bold'),

## activebackground='white',activeforeground='black',bd=3,relief='flat',cursor='hand2',

## command=self.mainclear)

## self.btn2.place(x=155,y=160)

## self.log = PhotoImage(file='cart.png')

## self.btn2.config(image=self.log, compound=LEFT)

## self.small\_log = self.log.subsample(1, 1)

## self.btn2.config(image=self.small\_log)

## #-----------------------label clicked change password---------------------

## self.forgot=Label(self.fm1,text='forgotten password',fg='red',bg='#fff',activeforeground='black',

## font=('cursive',9,'bold'))

## self.forgot.place(x=80,y=220)

## self.forgot.bind("<Button>",self.mouseClick)

## root.mainloop()

## def mouseClick(self,event):

## self.rog=Tk()

## self.rog.title("Change password")

## self.rog.geometry("400x300+530+280")

## self.rog.iconbitmap("aa.ico")

## self.rog.resizable(0,0)

## self.rog.configure(bg='#fff')

## self.label=Label(self.rog,text="New password",bg='#fff',fg='red',font=("cursive",20,'bold'))

## self.label.place(x=105,y=15)

## self.user=Label(self.rog,text='User ID :',bg='#fff',fg='black',font=("cursive",10,'bold'))

## self.user.place(x=40,y=95)

## self.user = Label(self.rog, text='New password :', bg='#fff', fg='black', font=("cursive", 10, 'bold'))

## self.user.place(x=40, y=170)

## self.e1 = Entry(self.rog, width=24, font=('arial', 9, 'bold'), bd=4, relief='groove')

## self.e1.place(x=170, y=95)

## self.e2 = Entry(self.rog, width=24, font=('arial', 9, 'bold'), bd=4, relief='groove')

## self.e2.place(x=170, y=170)

## self.btn1 = Button(self.rog, text='Submit', fg='white', bg='#5500ff', width=20, font=('Arial', 13, 'bold'),

## activebackground='white', activeforeground='black',bd=3, relief='flat',

## cursor='hand2',command=self.chan\_pas)

## self.btn1.place(x=100, y=240)

## def chan\_pas(self):

## self.a=self.e1.get()

## self.b=self.e2.get()

## import sqlite3

## conn=sqlite3.connect('admin.db')

## cursor=conn.cursor()

## cursor.execute("SELECT \* FROM adm WHERE User\_ID='"+self.a+"'")

## conn.commit()

## self.data=cursor.fetchone()

## if self.data!=None:

## cursor = conn.cursor()

## cursor.execute("UPDATE adm SET Password='" + self.b + "' WHERE User\_ID='" + self.a + "'")

## conn.commit()

## messagebox.showinfo("Library System","Your Password is changed !")

## else:

## self.er = Label(self.rog, text='ID is not required', bg='#fff', fg='red', font=("cursive", 8, 'bold'))

## self.er.place(x=170, y=125)

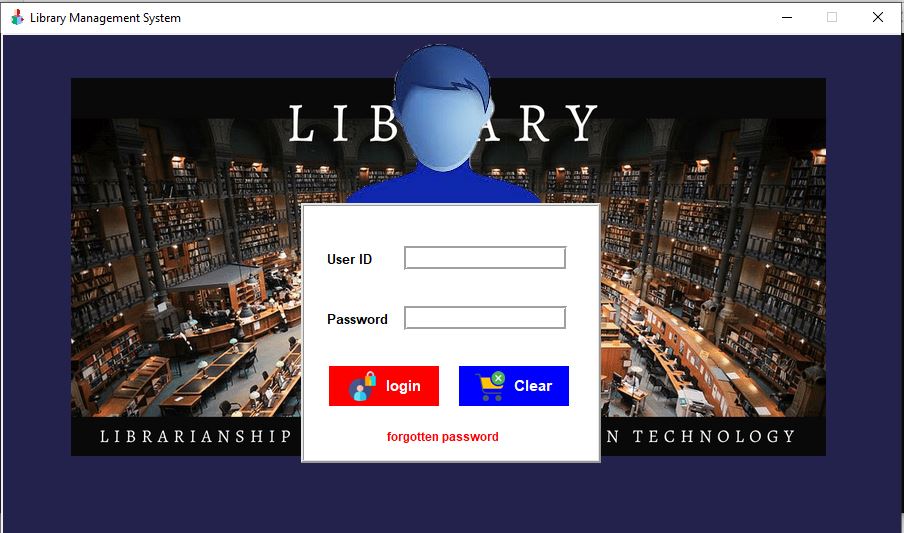
## self.rog.mainloop()

## ob=maincode()

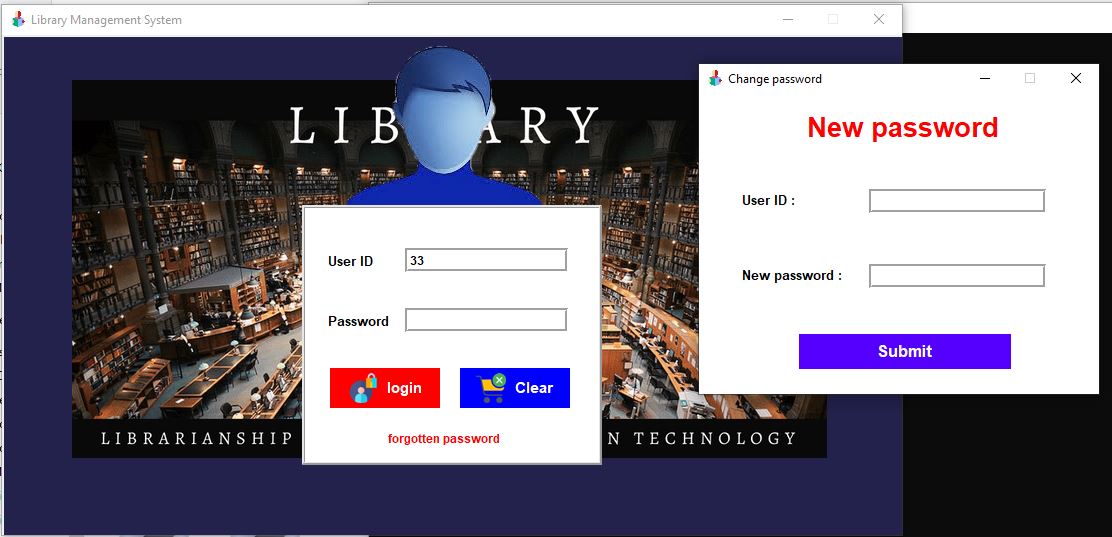
## ob.code()

## SCREENSHOTS

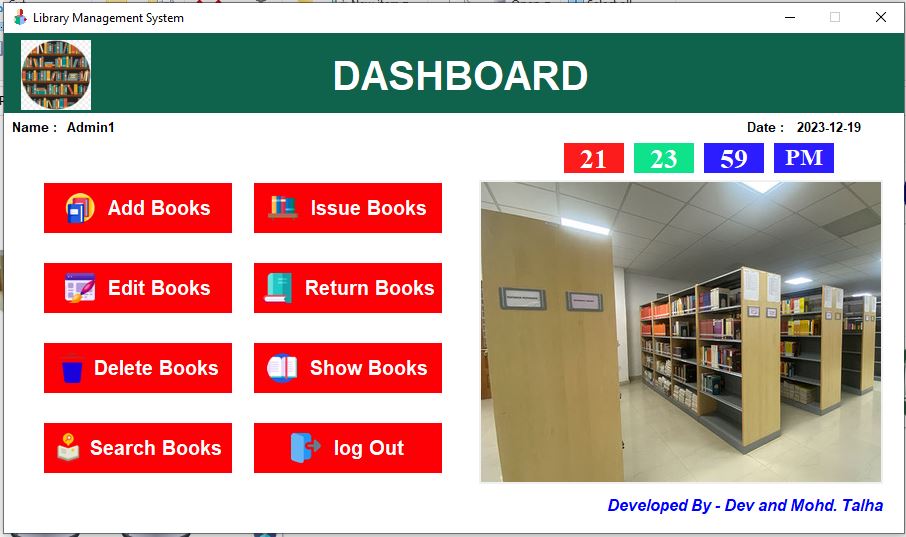
Login Page:-



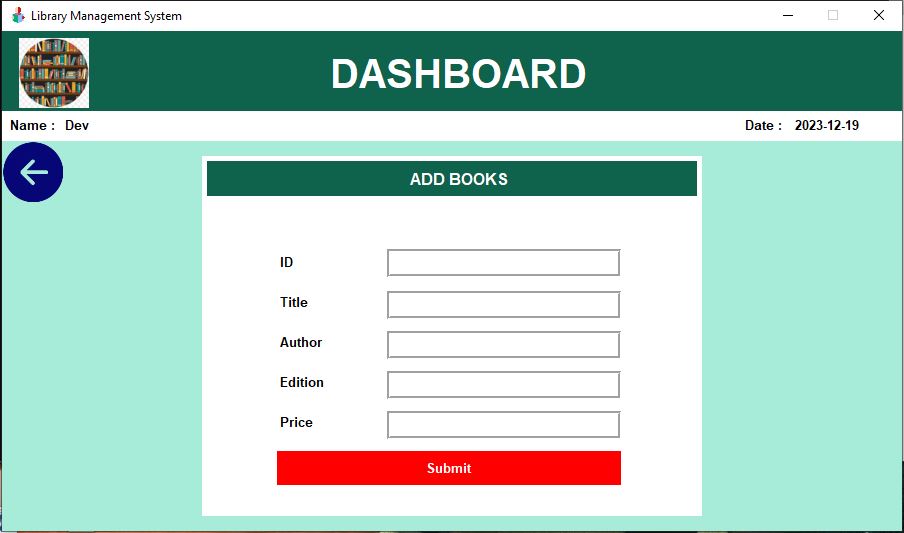
Login Page (Forgotten Password):-



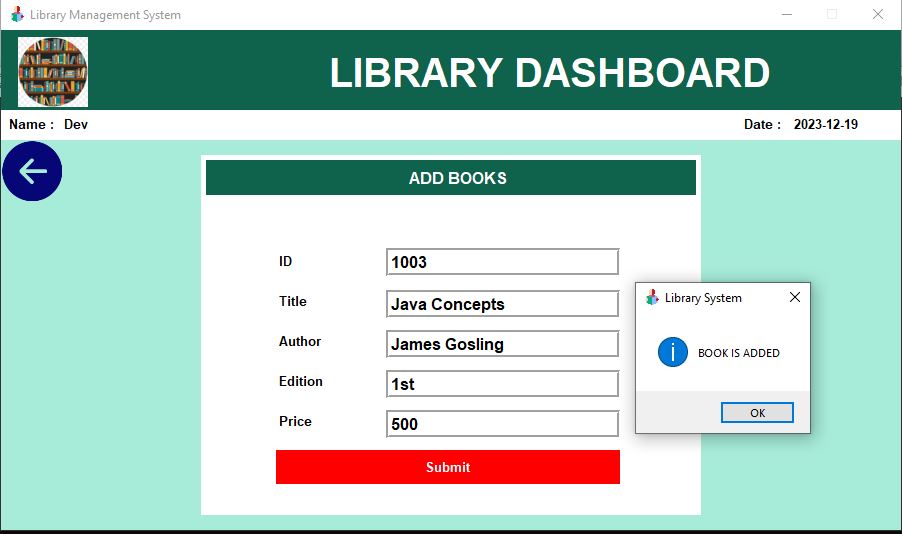
Dashboard:-



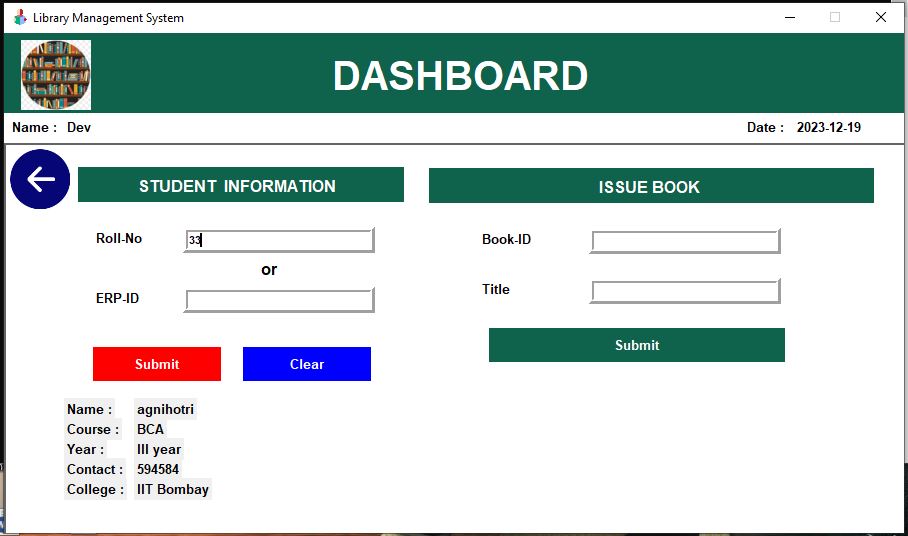
Add Book:-



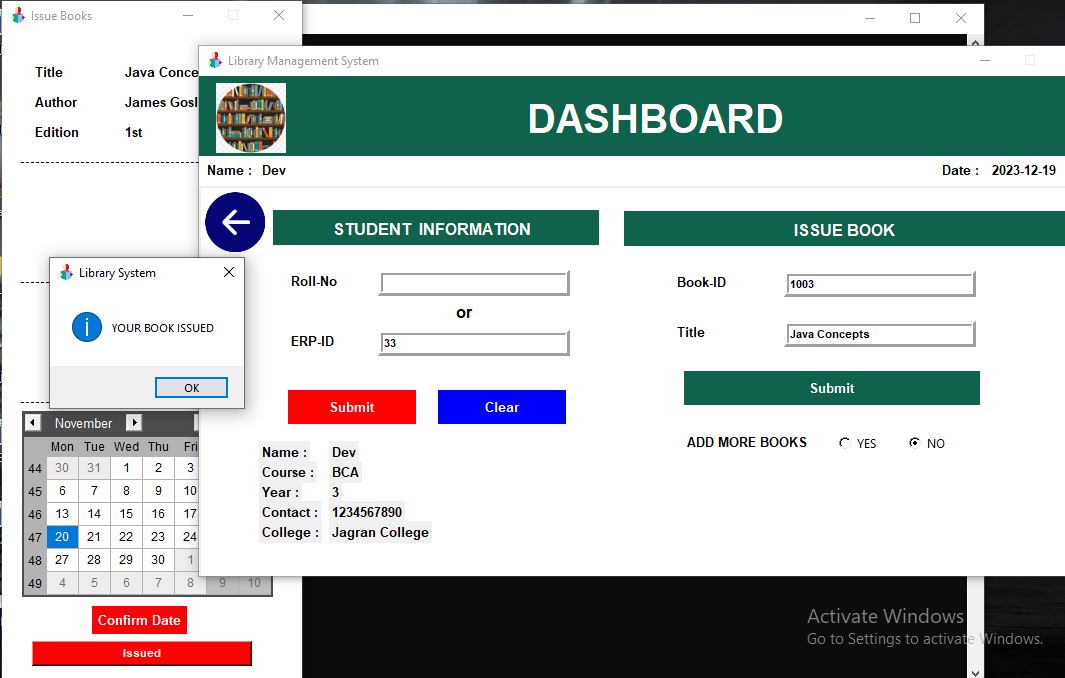
Add Book (after adding book):-



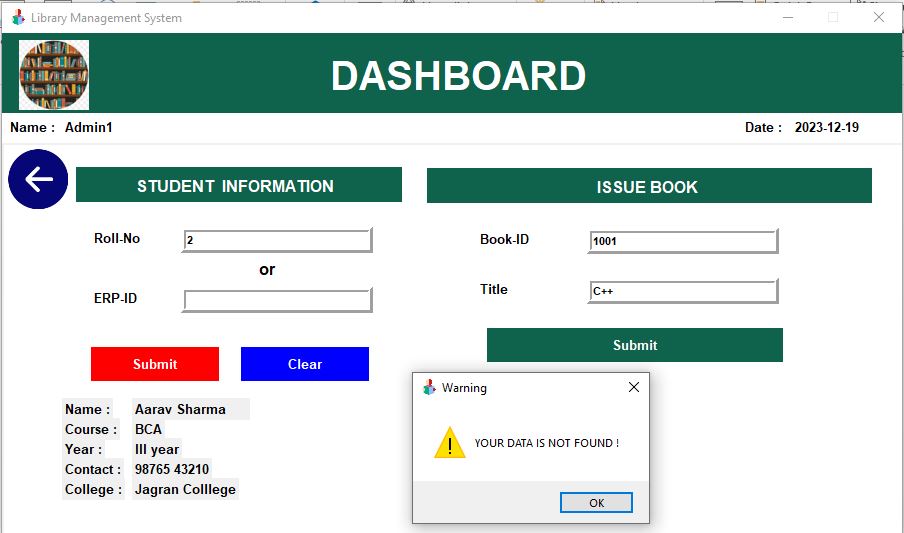
Issue Book:-



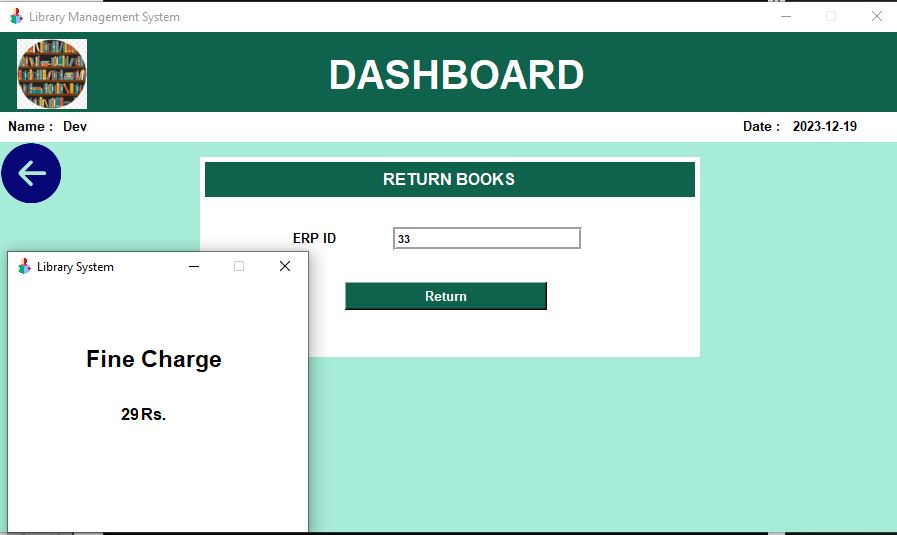
Issue Book (Successfully issuing book):-



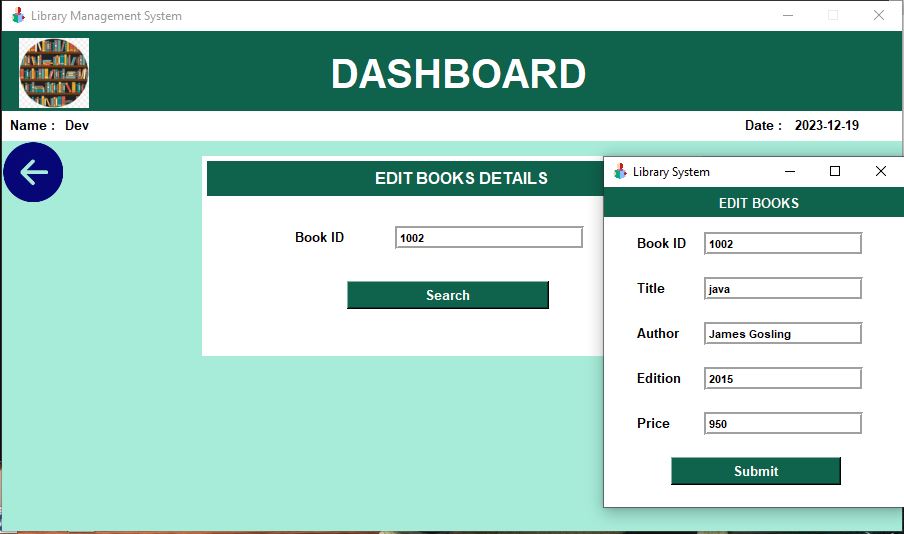
Issue Book (Error):-



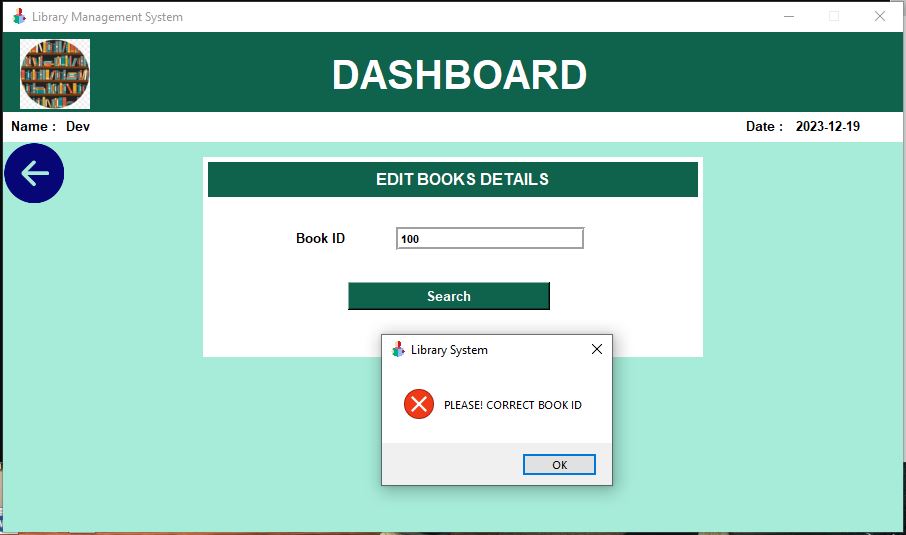
Return Book:-



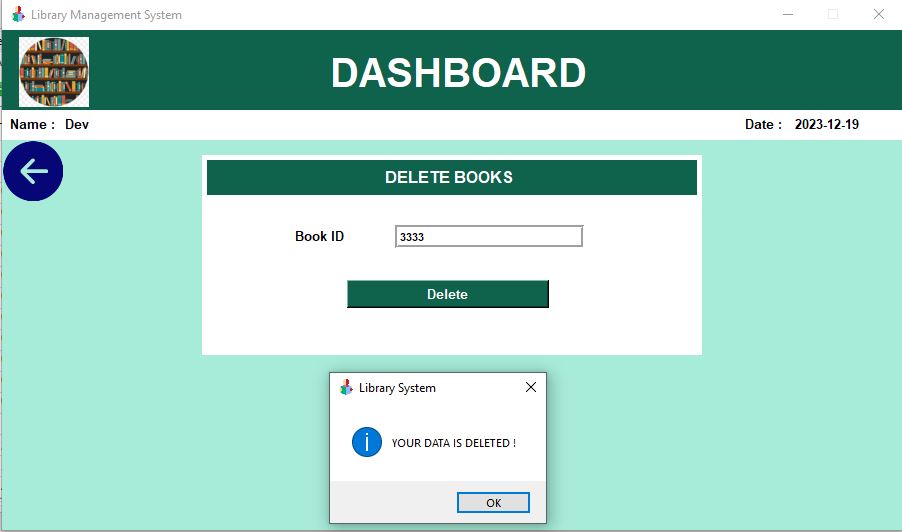
Edit Book (Correct Book ID):-



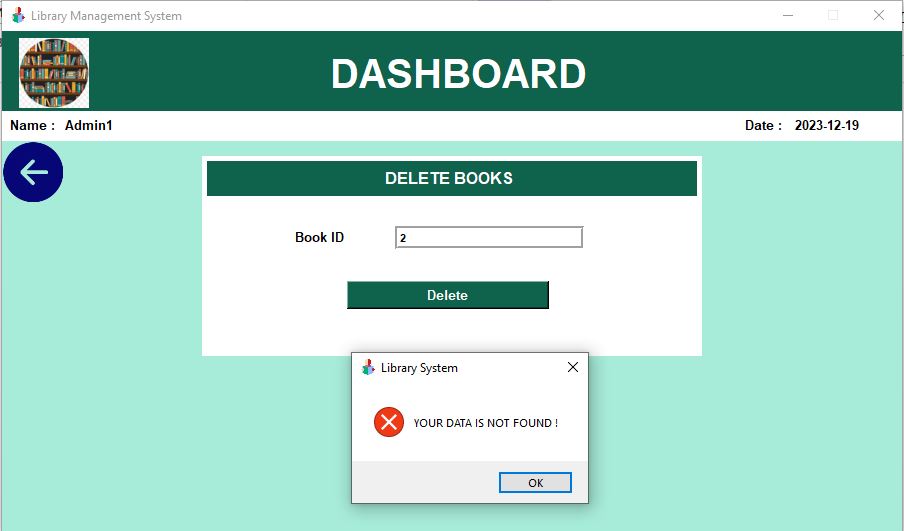
Edit Book (Incorrect Book ID):-



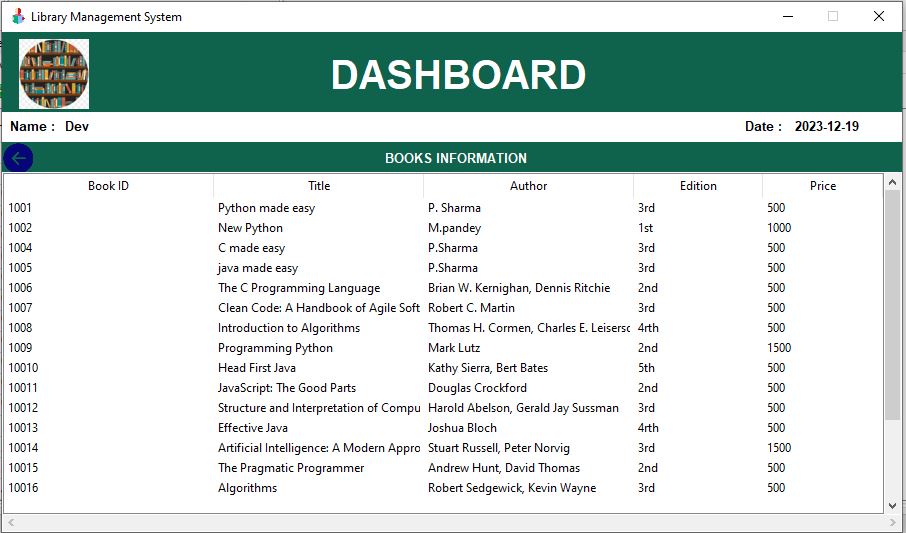
Delete Book (Correct Book ID):-



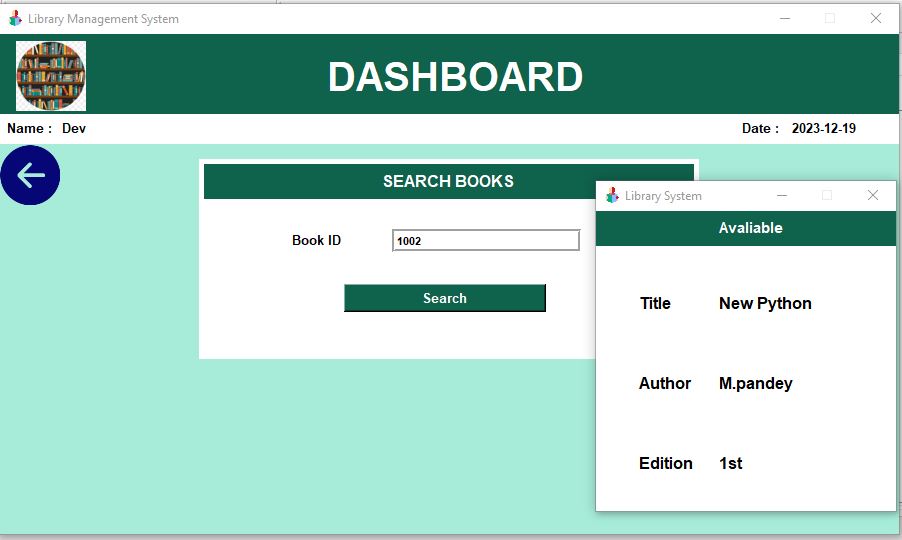
Delete Book (Wrong Book ID):-



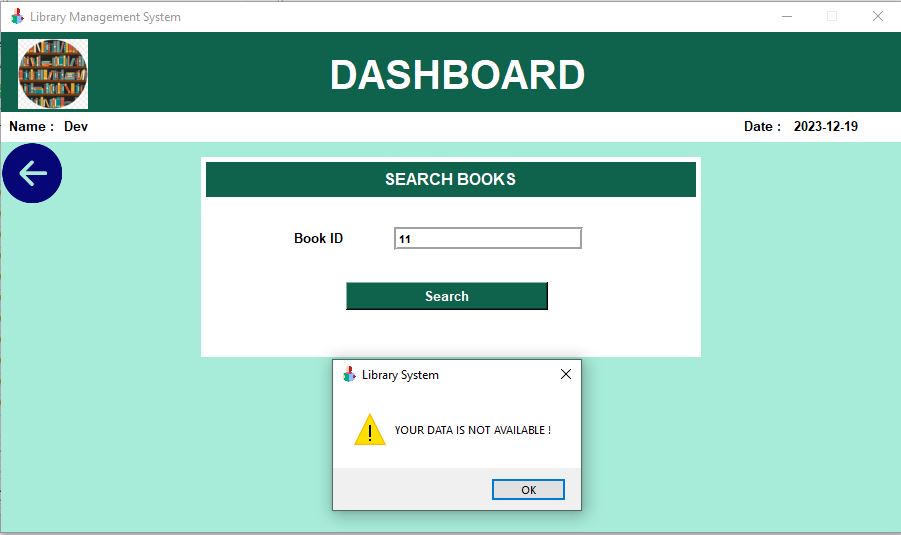
Show Books:-



Search Book (Correct Book ID):-



Search Book (Incorrect Book ID):-



## Software and Hardware Requirements

Below requirements outline what's needed to successfully develop and run the "Lib Management GUI" project.

#### Software Requirements:-

|  |  |
| --- | --- |
| **Requirement** | **Description on** |
| Operating System | Windows 8/10/10.1/11 |
| User Interface | Tkinter, a Python library for creating graphical interfaces. |
| Programming Language | Python version 3.8 or above |
| Database management | SQLite version 3 or above |
| IDE used | Visual Studio Code and SQLite Studio |

#### Hardware Requirements:-

|  |  |
| --- | --- |
| **Requirement** | **Description on** |
| Processor | Pentium 4 or higher |
| Hard Disk | 10 Mb of space required |
| Memory(RAM) | 2 GB ram or above |

## Bibliography

1. Python Software Foundation. (2023). Python 3 Documentation.

(https://docs.python.org/3/)

2. Tkinter Documentation (<https://docs.python.org/3/library/tkinter.html>)

3. SQLite Documentation (<https://www.sqlite.org/docs.html>)

4. GeeksforGeeks {Python Programming Language.} (https://www.geeksforgeeks.org/python-programming-language)

5. Real Python

(https://realpython.com)

6. Stack Overflow

(<https://stackoverflow.com>)

## Team Members

## Dev Agnihotri

## Mohd. Talha