Library Management System

Summer Training Report

Submitted in partial fulfilment of the requirement for the

Degree of

B.Tech

In

Information Technology

BPIT

Under the Supervision of

Mr. Shailendra Gaur

By

Anand Vijay Rajsri(00120807717)

Bhagwan Parshuram Institute of Technology

PSP-4, Sector-17, Rohini, Delhi - 89

June – July 2018

DECLARATION

This is to certify that Report entitled "LIBRARY MANAGEMENT SYSTEM" which is submitted in partial fulfillment of the requirement for the award of degree B.Tech in Information Technology to BPIT, GGSIP University, Dwarka, Delhi comprises only our original work and due acknowledgement has been made in the text to all other material used.

Date: 10/10/2018 **Anand Vijay Rajsri(00120807717)**

Acknowledgement

I have taken a great effort in this project. However, it would not have been possible without the kind support and help of many officers of Neb Sarai, Saket. I would like to extend my sincere thanks to all of them.

I am highly grateful to Inspire Tech Zone Facility for their guidance and supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards member of Inspire Tech Zone Facility & Bhagwan Parshuram Institute of Technology for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to **Mr. G.S Singh (Developer)** for giving me such attention and time.

Lastly, I also express my gratitude to other faculty members of Department in Inspire Tech Zone, for their intellectual support throughout the course of this work. Finally, I am indebted to all whosoever have contributed in this work.

Anand Vijay Rajsri (00120807717)

Training Coordinator Certificate

This is to certify that Report entitled “Library Management System” which is submitted by Anand Vijay Rajsri (00120807717 in partial fulfillment of the requirement for the award of degree B.Tech in Information Technology to BPIT, GGSIP University, Dwarka, Delhi is a record of the candidate own work and the matter embodied in this reports adhered to the given format.

Date: 10/10/2018 **Coordinator**

Mr. Shailendra Gaur

Table of Contents

List of Fig & Tables

Abstract

“Library management system” is a project which aims in developing a computerized system to maintain all the daily work of library .This project has many features which are generally not available in normal library management systems like facility of user login and a facility of teachers login .It also has a facility of admin login through which the admin can monitor the whole system .It also has facility of an online notice board where teachers can student can put up information about workshops or seminars being held in our colleges or nearby colleges and librarian after proper verification from the concerned institution organizing the seminar can add it to the notice board . It has also a facility where student after logging in their accounts can see list of books issued and its issue date and return date and the students can request the librarian to add new books by filling the book request form. The librarian after logging into his account i.e. admin account can generate various reports such as student report, issue report, teacher report and book report.

Overall this project of ours is being developed to help the students as well as staff of library to maintain the library in the best way possible and reduce the human efforts.

CHAPTER 1
INTRODUCTION

This chapter gives an overview about the aim, objectives, background and operation environment of the system.

1.1 PROJECT AIMS AND OBJECTIVES

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter.

The aims and objectives are as follows:

- A search column to search availability of books and quantity of book.
- Student login page where student can find books.
- A staff login page where staff can search availability of books, issue book and return book options.
- An admin login page where admin can add, delete of the book available in the system.
- Admin has the authority to update quantity, add student and staff into database.
- The system will be under sole control of the admin.
- The system can calculate the date of issued and return date of book.

1.2 BACKGROUND OF PROJECT

Library Management System is an application which refers to library systems which are generally small or medium in size. It is used by librarian to manage the library using a computerized system where he/she can record various transactions like issue of books, return of books, addition of new books, addition of new students etc.

Books and student maintenance modules are also included in this system which would keep track of the students using the library and a detailed description about the books a library contains. With this computerized system there will be no loss of book record or member record which generally happens when a non-computerized system is used.

In addition, report module is also included in Library Management System. If user’s position is admin, the user can generate different kinds of reports like lists of students registered, list of books, issue and return reports.

All these modules can help librarian to manage the library with more convenience and in a more efficient way as compared to library systems which are not computerized.

1.3 OPERATION ENVIRONMENT

PROCESSOR	INTEL CORE PROCESSOR OR BETTER PERFORMANCE
OPERATING SYSTEM	WINDOWS VISTA, WINDOWS 7 OR MORE, UBUNTU
MEMORY	1GB RAM OR MORE
HARD DISK SPACE	MINIMUM 3 GB FOR DATABASE USAGE FOR FUTURE
DATABASE	SQLite

Table. 1 OPERATION ENVIRONMENT

CHAPTER 2
SYSTEM ANALYSIS

In this chapter, we will discuss and analyze about the developing process of Library Management System including software requirement specification (SRS) and comparison between existing and proposed system. The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out. Besides that, existing vs proposed provides a view of how the proposed system will be more efficient than the existing one.

SOFTWARE REQUIREMENT SPECIFICATION

2.1 GENERAL DESCRIPTION

# PRODUCT DESCRIPTION:

Library Management System is a computerized system which helps user (librarian) to manage the library daily activity in electronic format. It reduces the risk of paper work such as file lost, file damaged and time consuming.

It can help user to manage the transaction or record more effectively and time saving.

#PROBLEM STATEMENT:

The problem occurred before having computerized system includes:

1. File lost
   When computerized system is not implemented file is always lost because of human environment. Sometimes due to some human error there may be a loss of records.

2. File damaged when a computerized system is not their file is always lost due to some accident like spilling of water by some member on file accidentally. Besides some natural disaster like floods or fires may also damage the files.

3. Difficult to search record
   When there is no computerized system there is always a difficulty in searching of records if the records are large in number.

4. Space consuming
   After the number of records become large the space for physical storage of file and records also increases if no computerized system is implemented.

5. Cost consuming
   As there is no computerized system to add each record paper will be needed which will increase the cost for the management of library.

2. 2 SYSTEM OBJECTIVES

1. Improvement in control and performance
   The system is developed to cope up with the current issues and problems of library. The system can add user, validate user and is also bug free.

2. Save cost
   After computerized system is implemented less human force will be required to maintain the library thus reducing the overall cost.

3. Save time
   Librarian can search record by using few clicks of mouse and few search keywords thus saving his valuable time.

2.3 SYSTEM REQUIREMENTS

2.3.1 NON-FUNCTIONAL REQUIREMENTS

- Product Requirements

EFFICIENCY REQUIREMENT

When a library management system will be implemented librarian and user will easily access library as searching and book transaction will be very faster.

RELIABILITY REQUIREMENT

The system should accurately perform member registration, member validation, book transaction and search.

USABILITY REQUIREMENT

The system is designed for a user-friendly environment so that student and staff of library can perform the various tasks easily and in an effective way.

- Organizational Requirement

IMPLEMENTATION REQUIREMNTS

In implementing whole system, it uses python in front end language which will be used for database connectivity and the backend i.e. the database part is developed using SQLite.

DELIVERY REQUIREMENTS

The whole system is expected to be delivered in six months of time with a weekly evaluation by the project guide.

2.3.2 FUNCTIONAL REQUIREMENTS

1) **REGISTER NEW USER**

1.1 Description of feature

This feature used by the user to login into system. They are required to enter user id and password before they can enter the system. The user id and password will be verified and if invalid id is there user can not enter the system.

1.2 Functional requirements

- User id is provided when they register.
- The system must only allow user with valid id and password to enter the system.
- The system performs authorization process which decides what user level can access to.
- The user must be able to logout/exit after they finished using system.

2) REGISTER NEW BOOK

2.1 Description of feature

This feature allows to add new books to the library.

2.2 Functional requirements

- System must be able to verify information.
- System must be able to enter number of copies into table.
- System must be able to not allow two books having same book id (ISBN).

3) SEARCH BOOK

3.1 Description of feature

This feature is found in book maintenance part. We can search book based on book name.

3.2 Functional requirements

- System must be able to search the book on database.
- System must be able to show quantity of the book.

4) ISSUE BOOKS AND RETURN BOOKS DATE

4.1 Description of feature

This feature allows to issue and return books and view reports of book issued.

4.2 Functional requirements

- System must be able to maintain record of issue book information.
- System must be able to update number of books.
- System must be able to search if book is available or not before issuing books.
- System should be able to enter issue and return date information.

2.4 SOFTWARE AND HARDWARE REQUIREMENTS

This section describes the software and hardware requirements of the system.

2.4.1 SOFTWARE REQUIREMENTS

- Operating system Windows 7 is used as the operating system as it is stable and supports more features and is more user friendly.
- Database "SQLite" is used as database as it easy to maintain and retrieve records by simple queries which are in English language which are easy to understand and easy to write.
- Development tools and Programming language "PYTHON" is used to write the whole code and develop.

2.4.2 HARDWARE REQUIREMENTS

- Intel core i5 2nd generation is used as a processor because it is fast than other processors and provide reliable and stable and we can run our pc for long time. By using this processor, we can keep on developing our project without any worries.
- Ram 1GB is used as it will provide fast reading and writing capabilities and will in turn support in processing.
- Hard disk space minimum 3 GB for database usage for future

CHAPTER 3
DIAGRAMS

3.1 DATA FLOW DIAGRAMS

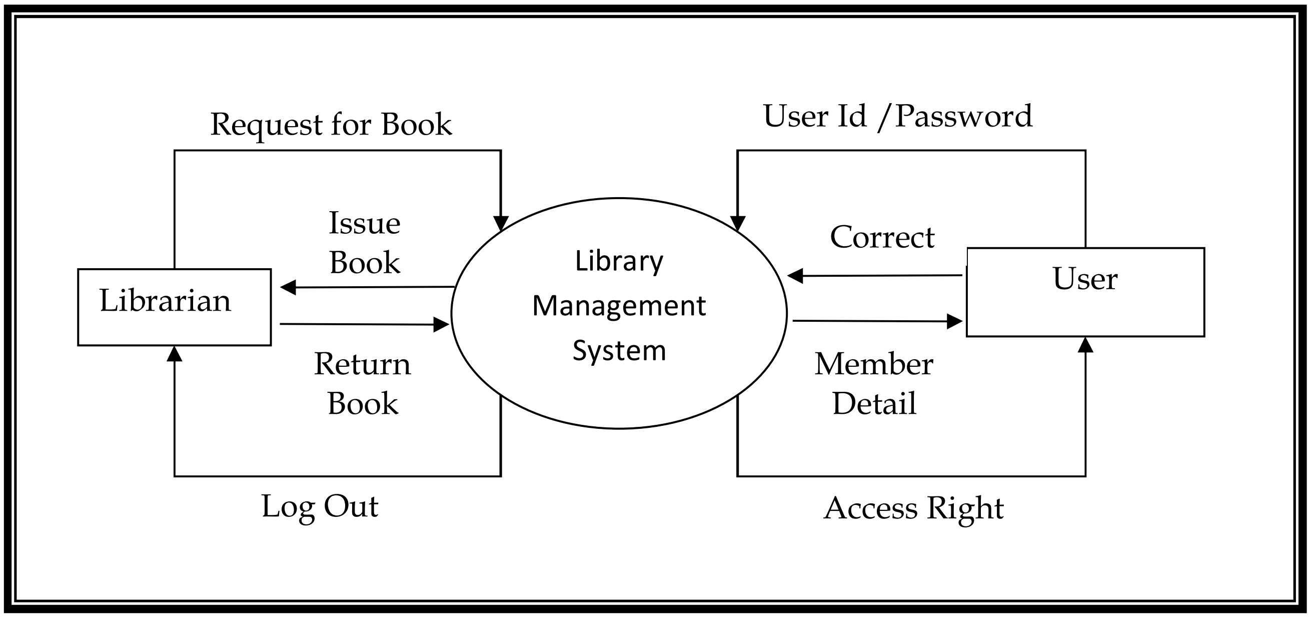


Figure 1 Zero Level DFD

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A level 0 data flow diagram (DFD), also known as a context diagram, shows a data system and emphasizes the way it interacts with external entities. This DFD level 0 example of Library Management System.

3.2 Use Case

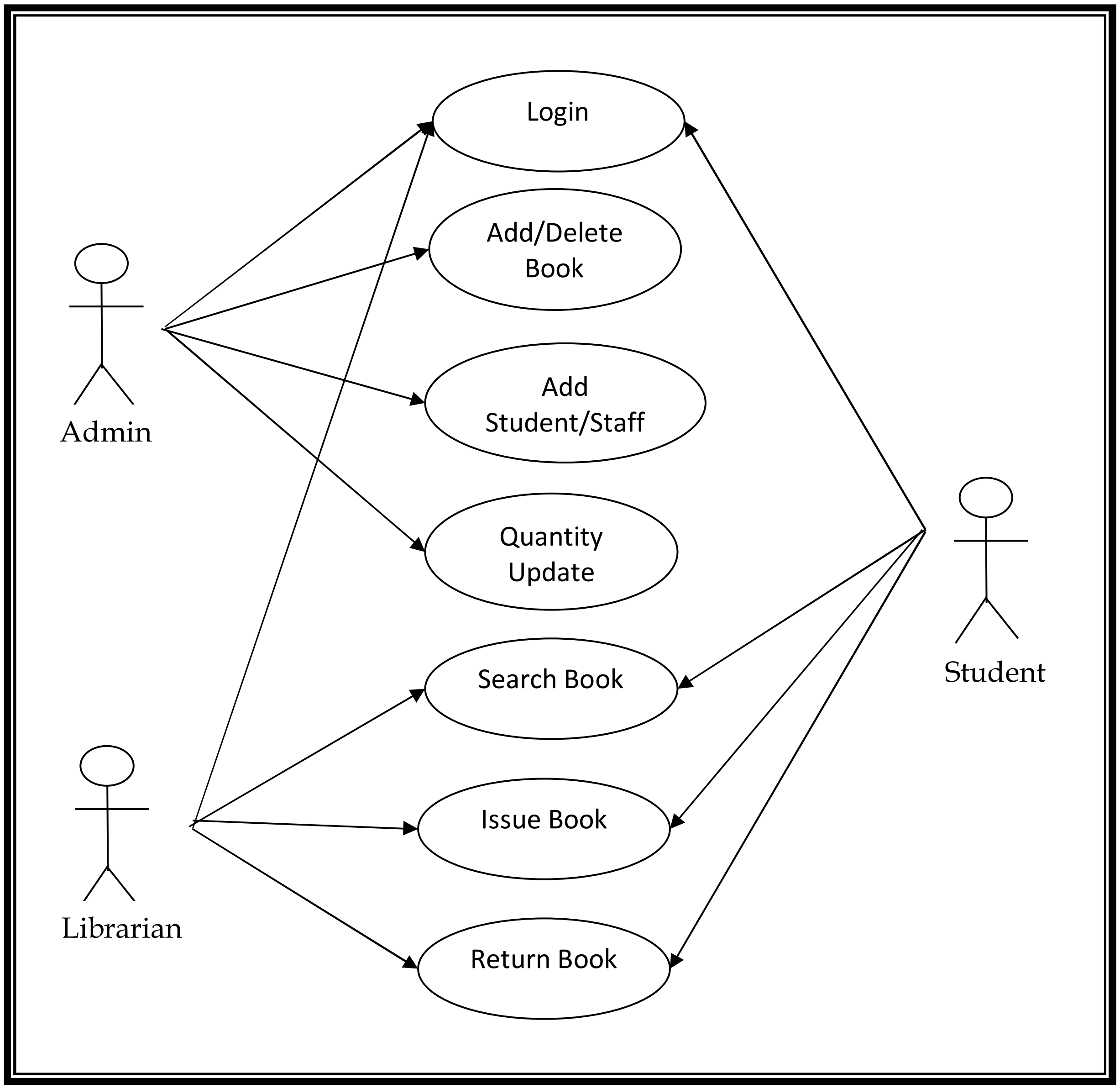


Figure 2 Use Case Diagram

The use case diagram is usually referred to as behavior diagram used to describe the actions of all user in a system. All user describe in use case are actors and the functionality as action of system.

3.3 ER DIAGRAM

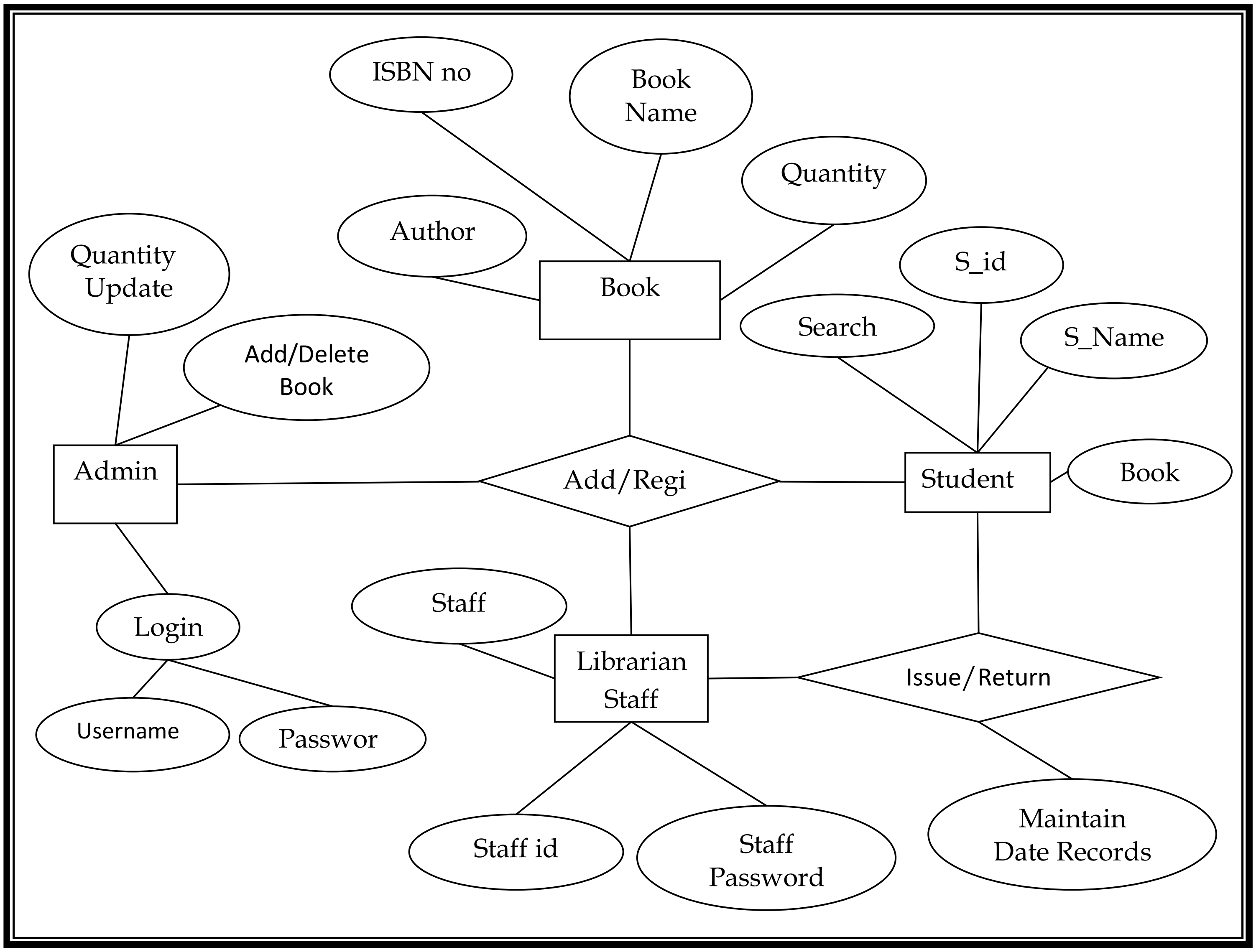
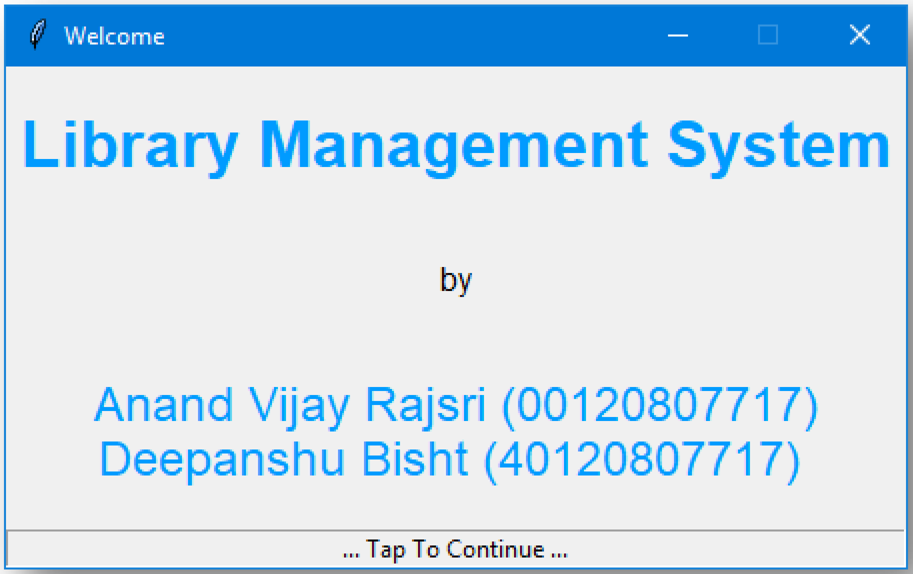


Figure 3 ER Diagram

An **entity–relationship model** (**ER model** for short) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

CHAPTER 4
CODE

1. Welcome Window



Library Management System

by

Anand Vijay Rajsri (00120807717)
Deepanshu Bisht (40120807717)

... Tap To Continue ...

Fig.1 Screenshot for Welcome Window

```
from tkinter import *
from time import *
from tkinter import messagebox
import sqlite3
import datetime

# ----------Main Win-----------------------
root = Tk()
root.geometry("450x250")
root.resizable(width=False, height=False)
root.title("Welcome")


# -----------Show win in center--------------
def center(win):
    win.update_idletasks()
    width = win.winfo_width()
    height = win.winfo_height()
    x = (win.winfo_screenwidth() // 2) - (width // 2)
    y = (win.winfo_screenheight() // 2) - (height // 2)
    win.geometry('{}x{}+{}+{}'.format(width, height, x, y))
center(root)
root.bind("<Button-1>", fun)
root.bind("<Return>", fun)
root.bind("<Tab>", fun)
root.bind("<space>", fun)

l1 = Label(root, text="Library Management System",
font=('arial 24 bold'), fg="#009bff")
l1.pack(fill="both", expand=True)
root.grid_rowconfigure(1, weight=1)
root.grid_columnconfigure(1, weight=1)
l2 = Label(root, text="by", font=('arial 12'))
l2.pack(fill="both", expand=True)
l3 = Label(root, text="Anand Vijay Rajsri
(00120807717)\nDeepanshu Bisht (40120807717) ",
font=('arial 18'),
           fg="#009bff")
l3.pack(fill="both", expand=True)
l4 = Label(root, text="... Tap To Continue ...",
relief=SUNKEN, bd=1)
l4.pack(side=BOTTOM, fill=X)
root.mainloop()
```

2. Login Window

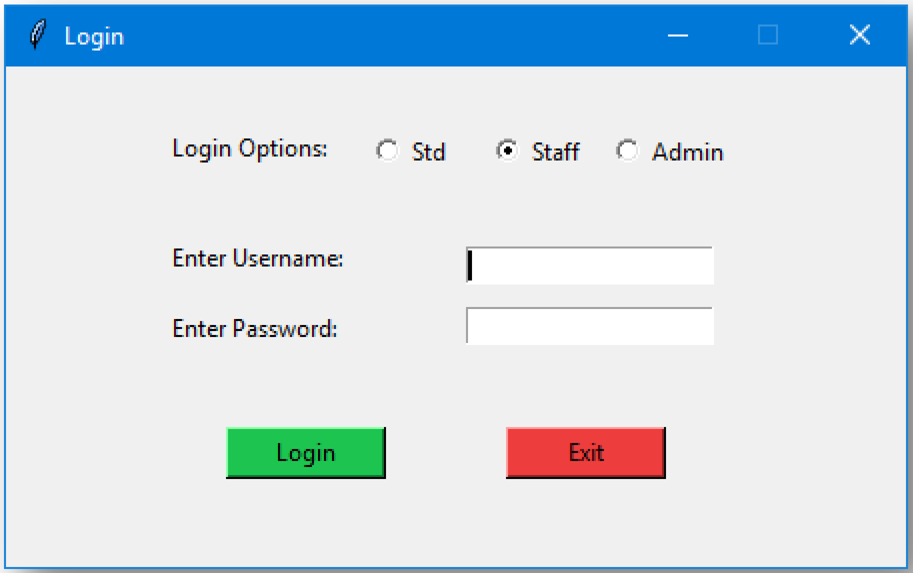
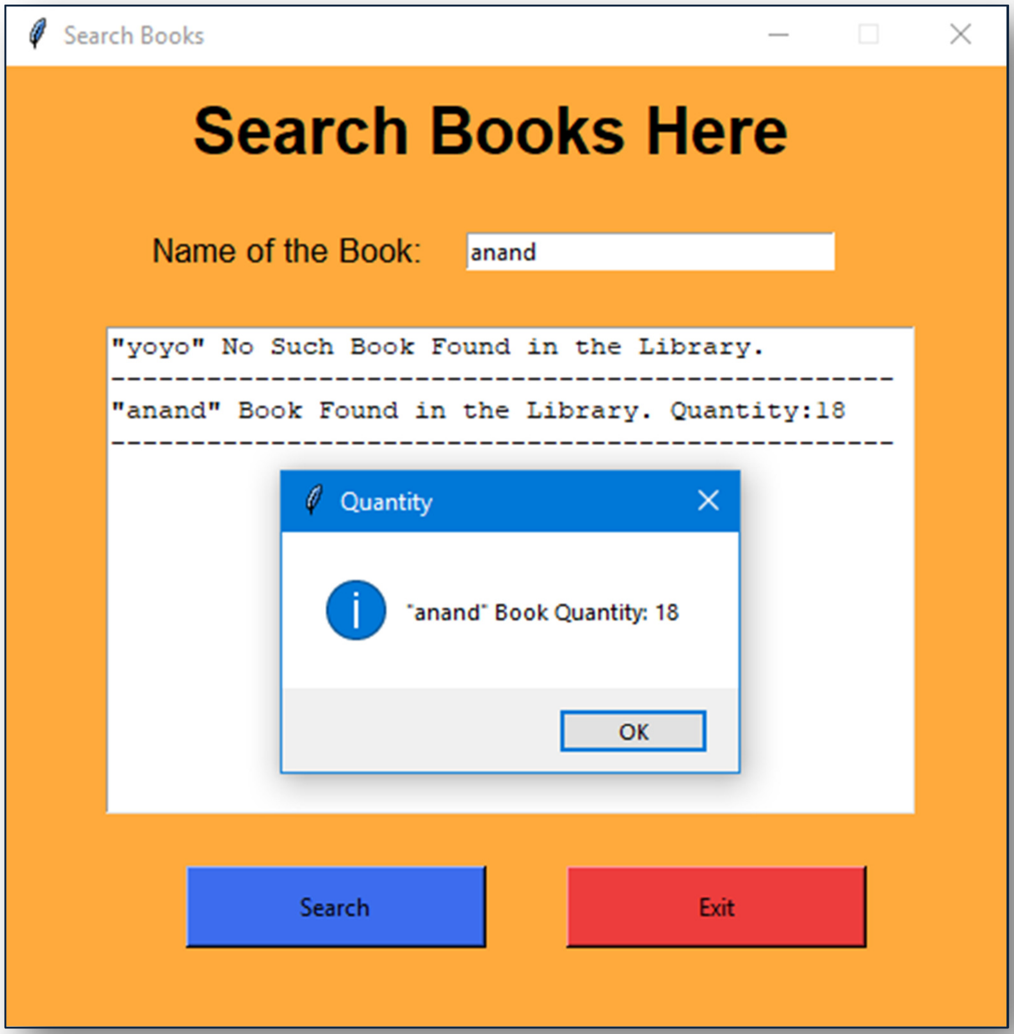


Fig. 2 Screenshot for Login Window

```
def fun(event):
    root.destroy()
    sleep(1)
    root1 = Tk()
    root1.geometry("450x250")
    root1.resizable(width=False, height=False)
    root1.title("Login")
    root1.focus_set()
    center(root1)


# -------- see Radiobutton--------------------------------
def login():
    ch = selected.get()
    # ----------login win---------------------------------
    if (ch == 1):  # ---------------------std log in
        try:
            u = e1.get()
            p = int(e2.get())
            std_call(u, p)

        except Exception as e:
            messagebox.showwarning("Value Error", str(e))

    elif (ch == 2):  # -----------------------staff log in
        try:
            u = e1.get()
            p = int(e2.get())
            staff_call(u, p)

        except Exception as e:
            messagebox.showwarning("Value Error", str(e))

    else:  # ---------------------------------admin log in
        au = "qwerty"
        ap = 123
        try:
            u = e1.get()
            p = int(e2.get())

            if (u == '' or p == ''):
                messagebox.showwarning("Error", "Please
Fill The Missing Details")

            else:
                if (au == u):
                    if (ap == p):
                        admin_call_class()

                    else:
                        messagebox.showwarning("Failed",
"Please Check The \"Password\"")
                else:
                    messagebox.showwarning("Failed",
"Please Check Username.")

        except Exception as e:
            messagebox.showwarning("Value Error", str(e))

# --------Radiobutton----------------------------------------
------
selected = IntVar()
rad1 = Radiobutton(root1, text='Std', value=1,
variable=selected)
rad2 = Radiobutton(root1, text='Staff', value=2,
variable=selected)
rad3 = Radiobutton(root1, text='Admin', value=3,
variable=selected)
selected.set(2)
rad1.place(x=180, y=30)
rad2.place(x=240, y=30)
rad3.place(x=300, y=30)

l = Label(root1, text="Login Options: ")
l.place(x=80, y=30)

l1 = Label(root1, text="Enter Username: ")
l2 = Label(root1, text="Enter Password: ")
l1.place(x=80, y=85)
l2.place(x=80, y=120)

e1 = Entry(root1)
e2 = Entry(root1, show="*")
e1.place(x=230, y=90)
e1.focus_set()
e2.place(x=230, y=120)

b1 = Button(root1, text="Login", width=10, height=1,
bg='#1DC550', command=login)
b1.place(x=110, y=180)
b2 = Button(root1, text="Exit", width=10, height=1,
bg='#EE3D3D', command=quit)
b2.place(x=250, y=180)
root1.mainloop()
```

```
#--------Staff Call Login-----------------------------------

def staff_call(u, p):
    conn = sqlite3.connect('lib.db')
    c = conn.cursor()
    if (u == '' or p == ''):
        messagebox.showwarning("Error", "Please Fill The 
Missing Details")

    else:
        sname = u
        sid = int(p)
        c.execute("SELECT * FROM staff_info WHERE stf_name 
= ?", [sname])
        data = c.fetchall()

        if data:
            c.execute("SELECT * FROM staff_info WHERE 
stf_pass = ?", [sid])
            data = c.fetchall()
            if data:
                stf_call_class()

            else:
                messagebox.showwarning("Invalid Pass", 
"Staff Pass is Invalid.")

        else:
            messagebox.showwarning("Invalid Name", "Staff 
Name Not Found.")
```

3. Student Login Into System



```
"yoyo" No Such Book Found in the Library.
-----------------------------------------------
"anand" Book Found in the Library. Quantity:18
-----------------------------------------------
```

Fig. 3 Screenshot for Student Search Window

```
def std_call(u, p):
    conn = sqlite3.connect('lib.db')
    c = conn.cursor()
    if (u == '' or p == ''):
        messagebox.showwarning("Error", "Please Fill The
Missing Details")

    else:
        sname = u
        sid = int(p)
        c.execute("SELECT * FROM std_info WHERE std_name =
?", [sname])
        data = c.fetchall()

        if data:
            c.execute("SELECT * FROM std_info WHERE std_id
= ?", [sid])
            data = c.fetchall()
            if data:
                root1.destroy()
                call_search_class()

            else:
                messagebox.showwarning("Invalid ID",
"Student ID is Invalid.")

        else:
            messagebox.showwarning("Invalid Name", "Student
Name Not Found.")

def call_search_class():
    root_s = Tk()
    root_s.geometry('500x480')
    root_s.resizable(False, False)
    root_s.config(background='#ffaa3c')
    root_s.title("Search Books")
    center(root_s)
    obj = Search(root_s)
    root_s.mainloop()
    pass

class Search(Text):
    def __init__(self, faster):
        Text.__init__(self, faster)

        self.heading = Label(faster, text="Search Books
Here", font=('arial 25 bold'), bg='#ffaa3c')
        self.heading.place(x=90, y=10)

        self.name = Label(faster, text="Name of the Book:
", font=('arial 13'), bg='#ffaa3c')
        self.name.place(x=70, y=80)

        self.ent = Entry(faster, width=30)
        self.ent.place(x=230, y=83)
        self.ent.focus_set()

        self.sbox = Text(faster, height=15, width=50,
bg="white")
        self.sbox.place(x=50, y=130)
        self.sbox.configure(state='disabled')

        self.bt = Button(faster, text="Search",
command=self.get_it, width=20, height=2, bg='#3D6DEE')
        self.bt.place(x=90, y=400)

        self.qt = Button(faster, text="Exit",
command=faster.destroy, width=20, height=2,
bg='#EE3D3D').place(x=280,

y=400)

    def get_it(self):

        conn = sqlite3.connect('lib.db')
        c = conn.cursor()
        c.execute(
            "CREATE TABLE IF NOT EXISTS lib_book(datestamp
TEXT, book_name TEXT, author_name TEXT, isbn_no INTEGER
PRIMARY KEY, quantity INTEGER)")
        bk = self.ent.get()

        if len(bk) == 0:
            messagebox.showwarning("Failed", "Please,
Enter Book Name.")

        else:
            c.execute("SELECT * FROM lib_book WHERE
book_name = ?", [bk])
            data = c.fetchall()
            if data:
                z = c.execute("SELECT * FROM lib_book WHERE
book_name = ?", [bk])
                for q in z:
                    if (q[4] > 0):
                        messagebox.showinfo("Quantity",
"\"" + bk + "\" Book Quantity: " + str(q[4]))
                    else:
                        messagebox.showwarning("Quantity",
"Sorry, ""\"" + bk + "\" Book Out of Stock..!")

                self.sbox.configure(state='normal')
                self.sbox.insert(END, "\"" + bk + "\" Book
Found in the Library. Quantity:" + str(q[4]) + "\n")
```

```
                self.sbox.insert(END, "--------------------
-------------------------------\n")
                self.sbox.configure(state='disabled')


            else:
                self.sbox.configure(state='normal')
                self.sbox.insert(END, "\"" + bk + "\" No
Such Book Found in the Library.\n")

                self.sbox.insert(END, "--------------------
-------------------------------\n")
                self.sbox.configure(state='disabled')
                messagebox.showwarning("Not Found", "Sorry,
Book Not Found.")

        c.close()
        conn.close()
```

4. Staff Login Into System

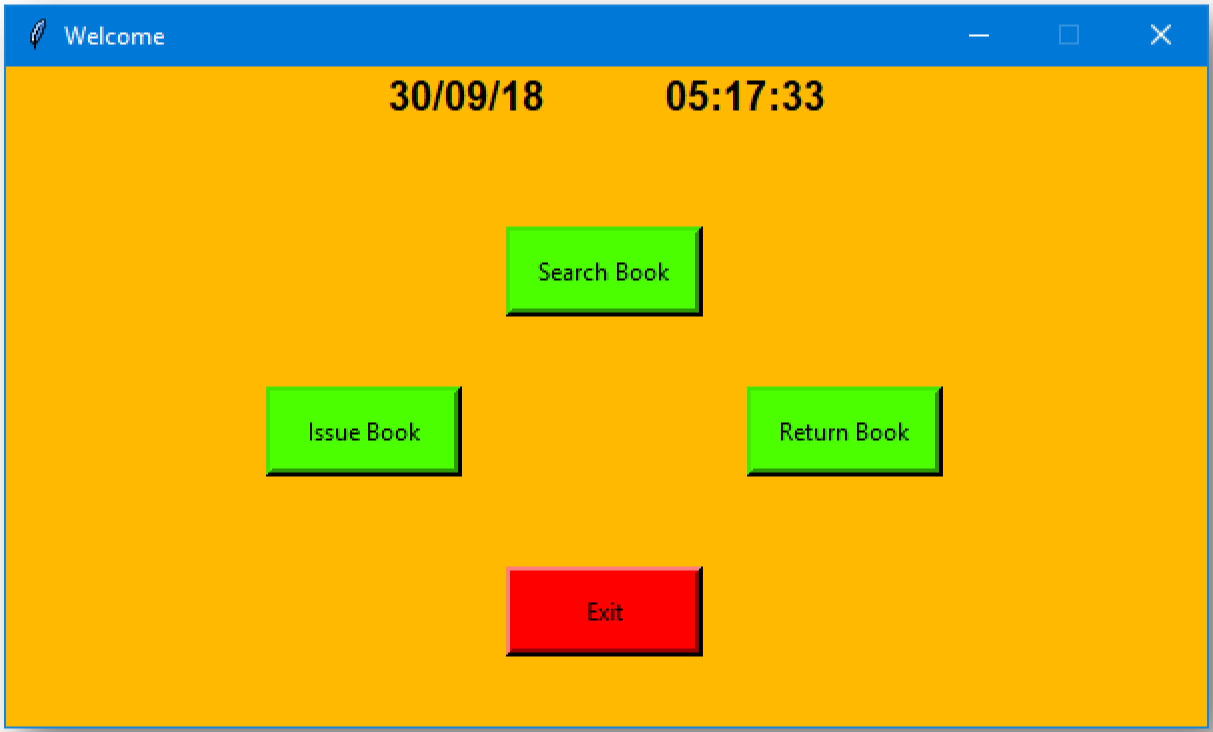
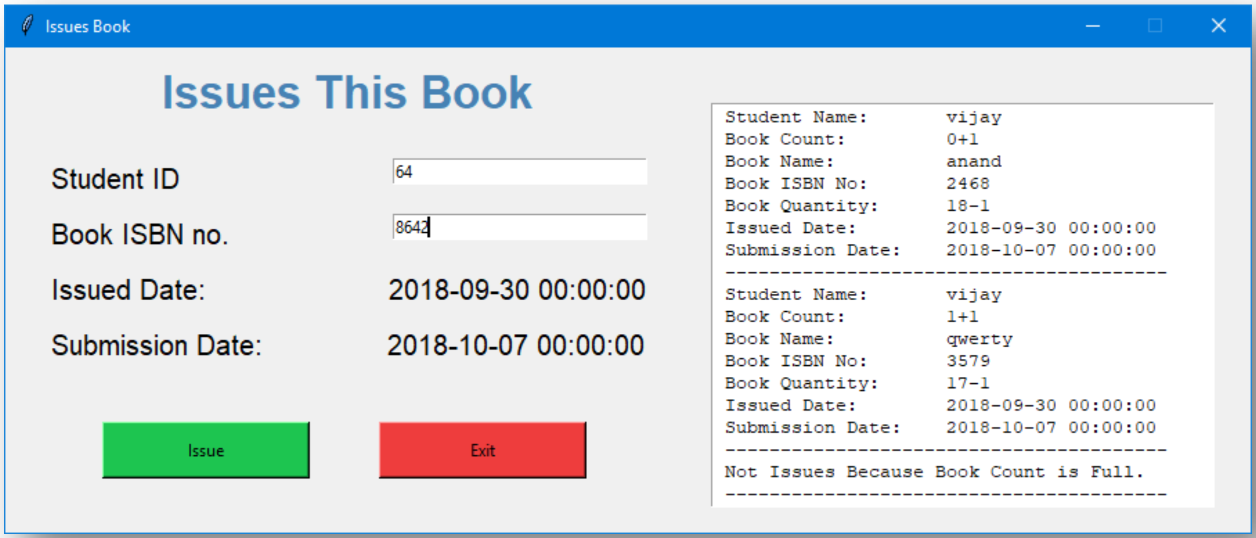


Fig. 4 Screenshot for Staff Login Window

```
def stf_call_class():
    root1.destroy()
    root2 = Tk()
    root2.geometry('600x330')
    root2.config(background='#ffb900')
    root2.resizable(False, False)
    root2.title("Welcome")
    center(root2)

    def tick(time1=''):
        time2 = strftime('%I:%M:%S')
        if time2 != time1:
            l1.config(text=time2)
        l1.after(200, tick)
    f1 = Frame(root2)
    f1.pack(side=TOP)
    l1 = Label(f1, font=('arial 15 bold'),
background='#ffb900')
    l1.pack(side=RIGHT)
    y = strftime("%d/%m/%y         ")
    Label(f1, text=y, font=('arial 15 bold'),
background='#ffb900').pack()
    tick()

    bg = "#4cff00"
    b1 = Button(root2, text="Search Book", bg=bg,
command=call_search_class, borderwidth=4, relief=RAISED,
width=12,
                height=2)
    b1.place(x=250, y=80)
    b2 = Button(root2, text="Issue Book", bg=bg,
command=call_issue_class, borderwidth=4, relief=RAISED,
width=12,
                height=2)
    b2.place(x=130, y=160)
    b3 = Button(root2, text="Return Book", bg=bg,
command=call_return_class, borderwidth=4, relief=RAISED,
width=12,
                height=2)
    b3.place(x=370, y=160)
    b4 = Button(root2, text="Exit", bg="#ff0000",
command=quit, borderwidth=4, relief=RAISED, width=12,
height=2)
    b4.place(x=250, y=250)

    root2.mainloop()
    pass
```

5. Issue Book



```
Student Name:       vijay
Book Count:         0+1
Book Name:          anand
Book ISBN No:       2468
Book Quantity:      18-1
Issued Date:        2018-09-30 00:00:00
Submission Date:    2018-10-07 00:00:00
----------------------------------------
Student Name:       vijay
Book Count:         1+1
Book Name:          qwerty
Book ISBN No:       3579
Book Quantity:      17-1
Issued Date:        2018-09-30 00:00:00
Submission Date:    2018-10-07 00:00:00
----------------------------------------
Not Issues Because Book Count is Full.
----------------------------------------
```

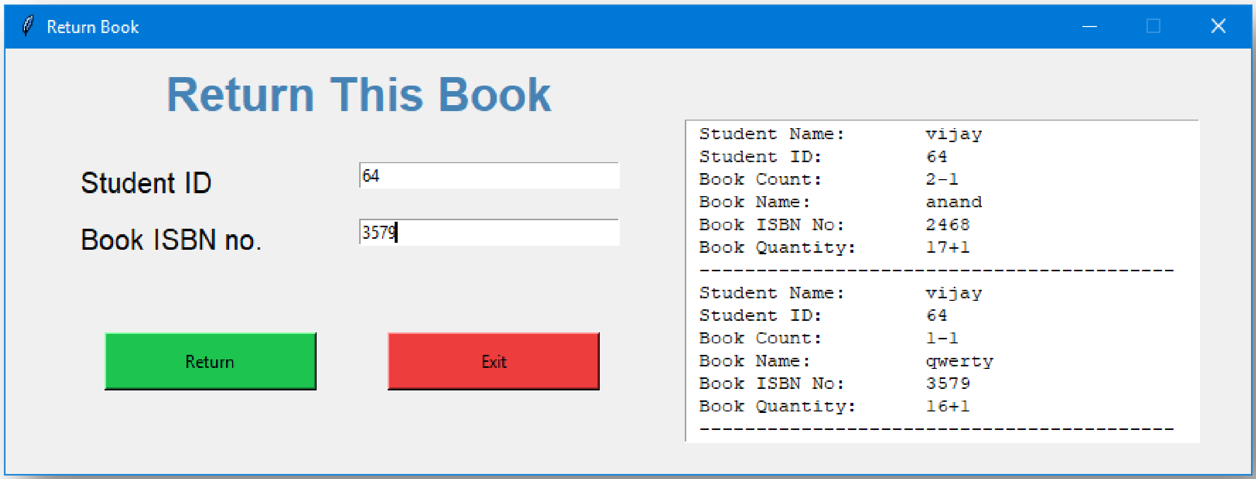
Fig. 5 Screenshot for Issue Book Window

```
def call_issue_class():
    root_i = Tk()
    root_i.geometry('900x350')
    root_i.resizable(False, False)
    root_i.title("Issues Book")
    center(root_i)
    obj = Issues(root_i)
    root_i.mainloop()
    pass

class Issues(Text):
    def __init__(self, fram):
        Text.__init__(self, fram)
        self.hd = Label(fram, text="Issues This Book",
font=('arial 25 bold'), fg='steelblue')
        self.hd.place(x=110, y=10)

        # info of the book taker
        self.id = Label(fram, text="Student ID",
font=('arial 15'))
        self.id.place(x=30, y=80)
        self.book = Label(fram, text="Book ISBN no.",
font=('arial 15'))
        self.book.place(x=30, y=120)

        self.issue = strftime("%x")
        self.date_1 =
datetime.datetime.strptime(self.issue, "%m/%d/%y")
        global Issued
        Issued = self.date_1
        self.end_date = self.date_1 +
datetime.timedelta(days=7)
        global Submission
        Submission = self.end_date

        self.dla = Label(fram, text=("Issued Date:  " + "
" + str(self.date_1)),
                                 font=('arial 15'))
        self.dla.place(x=30, y=160)

        self.end = Label(fram, text=("Submission Date:  " +
"              " + str(self.end_date)),
                                 font=('arial 15'))
        self.end.place(x=30, y=200)

        self.ide = Entry(fram, width=30)
        self.ide.place(x=280, y=80)
        self.ide.focus_set()
        self.booke = Entry(fram, width=30)
        self.booke.place(x=280, y=120)

        self.iss = Button(fram, text="Issue", width=20,
height=2, command=self.issues, bg='#1DC550').place(x=70,

y=270)
        self.xt = Button(fram, text="Exit", width=20,
height=2, command=fram.destroy, bg='#EE3D3D').place(x=270,

y=270)

        self.mty = Text(fram, height=18, width=45)
        self.mty.configure(state='disabled')
        self.mty.place(x=510, y=40)

    def issues(self):
        conn = sqlite3.connect('lib.db')
        c = conn.cursor()
        self.s_id = self.ide.get()
        self.b_id = self.booke.get()

        if self.s_id == '' or self.b_id == '':
            messagebox.showwarning("Error", "Please Fill
The Missing Details")
```

```
        else:
            try:
                sid = int(self.s_id)
                bid = int(self.b_id)
                c.execute("SELECT * FROM std_info WHERE
std_id = ?", [sid])
                data = c.fetchall()
                if data:

                    c.execute("SELECT * FROM lib_book WHERE
isbn_no = ?", [bid])
                    data = c.fetchall()
                    p = c.execute("SELECT * FROM std_info
WHERE std_id = ?", [sid])
                    for r in p:
                        pass

                    if data:
                        z = c.execute("SELECT * FROM
lib_book WHERE isbn_no = ?", [bid])
                        for q in z:
                            if (q[4] > 0):

                                t = c.execute("SELECT *
FROM std_info WHERE std_id = ?", [sid])
                                for n in t:
                                    pass
                                if (n[5] < 2):
                                    content = (" Student
Name:        " + r[2] +
                                               "\n Book
Count:          " + str(n[5]) + "+1" +
                                               "\n Book
Name:           " + q[1] +
                                               "\n Book
ISBN No:        " + self.b_id +
                                               "\n Book
Quantity:       " + str(q[4]) + "-1" +
                                               "\n Issued
Date:         " + str(Issued) +
                                               "\n
Submission Date:     " +

str(Submission) +
                                               "\n --------
--------------------------------\n")
```

```
self.mty.configure(state='normal')
                                    self.mty.insert(END,
content)

self.mty.configure(state='disabled')

messagebox.showinfo("Issued", "Successfully Issued the
Book.")

                                    j = q[4] - 1
                                    k = self.b_id
                                    c.execute("update
lib_book set quantity = ? where isbn_no= ?", (j, k))
                                    conn.commit()

                                    j = r[5] + 1
                                    k = self.s_id
                                    c.execute("update
std_info set book_count = ? where std_id= ?", (j, k))
                                    conn.commit()

                                    file = open(

"/Users/rajsri/PycharmProjects/lib_pro/Book_Issued/" +
self.b_id + ".txt",
                                        "a")
                                    file.write(content)
                                    file.close()

                                else:

self.mty.configure(state='normal')
                                    self.mty.insert(END, "
Not Issues Because Book Count is Full." +
                                                "\n ---
-------------------------------------\n")

self.mty.configure(state='disabled')

messagebox.showwarning("Not Issues", "Sorry, Book Count is
Full..!")

                            else:

messagebox.showwarning("Quantity", "Sorry, Book Out of
Stock..!")

self.mty.configure(state='normal')
                                self.mty.insert(END, " Book
Found in the Library. Quantity:" + str(q[4]) +
                                                "\n -------
---------------------------------\n")

self.mty.configure(state='disabled')

                    else:
                        self.mty.configure(state='normal')
                        self.mty.insert(END,
                                        " No Such Book
Found in the Library.\n -----------------------------------
-----\n")

self.mty.configure(state='disabled')
                        messagebox.showwarning("Not Found",
"Sorry, Book Not Found.")

                else:
                    self.mty.configure(state='normal')
                    self.mty.insert(END,
                                    " \"" + str(
                                        sid) + "\" Invalid
ID.\n ----------------------------------------\n")
                    self.mty.configure(state='disabled')
                    messagebox.showwarning("Not Found",
"Student ID Not Found.")

            except Exception as e:
                messagebox.showwarning("Value Error",
str(e))
```

6. Return Book


Return Book

Return This Book

Student ID 64

Book ISBN no. 3579

```
Student Name:       vijay
Student ID:         64
Book Count:         2-1
Book Name:          anand
Book ISBN No:       2468
Book Quantity:      17+1
------------------------------------------
Student Name:       vijay
Student ID:         64
Book Count:         1-1
Book Name:          qwerty
Book ISBN No:       3579
Book Quantity:      16+1
------------------------------------------
```

Fig. 6 Screenshot for Return Book Window

```
def call_return_class():
    root_r = Tk()
    root_r.geometry('880x300')
    root_r.resizable(False, False)
    root_r.title("Return Book")
    center(root_r)
    obj = Return(root_r)
    root_r.mainloop()
    pass

class Return(Text):
    def __init__(self, fram):
        Text.__init__(self, fram)
        self.hd = Label(fram, text="Return This Book",
font=('arial 25 bold'), fg='steelblue')
        self.hd.place(x=110, y=10)

        # info of the book taker
        self.id = Label(fram, text="Student ID",
font=('arial 15'))
        self.id.place(x=50, y=80)
        self.book = Label(fram, text="Book ISBN no.",
font=('arial 15'))
        self.book.place(x=50, y=120)

        # entries for the infos
        self.ide = Entry(fram, width=30)
        self.ide.focus_set()
        self.ide.place(x=250, y=80)
        self.booke = Entry(fram, width=30)
        self.booke.place(x=250, y=120)

        self.iss = Button(fram, text="Return", width=20,
height=2, command=self.return_book, bg='#1DC550').place(
            x=70, y=200)
        self.xt = Button(fram, text="Exit", width=20,
height=2, command=fram.destroy, bg='#EE3D3D').place(x=270,

y=200)

        self.mty = Text(fram, height=14, width=45)
        self.mty.configure(state='disabled')
        self.mty.place(x=480, y=50)

    def return_book(self):
        conn = sqlite3.connect('lib.db')
        c = conn.cursor()
        self.s_id = self.ide.get()
        self.b_id = self.booke.get()

        if self.s_id == '' or self.b_id == '':
            messagebox.showwarning("Error", "Please Fill
The Missing Boxes")

        else:
            try:
                sid = int(self.s_id)
                bid = int(self.b_id)
                c.execute("SELECT * FROM std_info WHERE
std_id = ?", [sid])
                data = c.fetchall()
                if data:

                    c.execute("SELECT * FROM lib_book WHERE
isbn_no = ?", [bid])
                    data = c.fetchall()

                    if data:
                        p = c.execute("SELECT * FROM
std_info WHERE std_id = ?", [sid])
                        for r in p:
                            if (r[5] > 0):
                                z = c.execute("SELECT *
FROM lib_book WHERE isbn_no = ?", [bid])
```

```
                                for k in z:
                                    pass

                                content = (" Student Name:
" + r[2] +
                                           "\n Student ID:
" + self.s_id +
                                           "\n Book Count:
" + str(r[5]) + "-1" +
                                           "\n Book Name:
" + k[1] +
                                           "\n Book ISBN
No:        " + self.b_id +
                                           "\n Book
Quantity:       " + str(k[4]) + "+1" +
                                           "\n ------------
------------------------------\n")

self.mty.configure(state='normal')
                                self.mty.insert(END,
content)

self.mty.configure(state='disabled')

messagebox.showinfo("Return", "Successfully Return the
Book.")

                                j = k[4] + 1
                                k = self.b_id
                                c.execute("update lib_book
set quantity = ? where isbn_no= ?", (j, k))
                                conn.commit()

                                j = r[5] - 1
                                k = self.s_id
                                c.execute("update std_info
set book_count = ? where std_id= ?", (j, k))
                                conn.commit()

                            else:

messagebox.showwarning("Book Count Zero", "No Book
Issue..!")

self.mty.configure(state='normal')
                                self.mty.insert(END, "
Student Name:   " + r[2] +
                                                  "\n Book
Count:     " + str(r[5]) +
                                                  "\n -------
-----------------------------------\n")

self.mty.configure(state='disabled')

                   else:
                       self.mty.configure(state='normal')
                       self.mty.insert(END,
                                          " No Such Book
Found in the Library.\n -----------------------------------
-------\n")

self.mty.configure(state='disabled')
                       messagebox.showwarning("Not Found",
"Sorry, Book Not Found.")

               else:
                   self.mty.configure(state='normal')
                   self.mty.insert(END,
                                  " \"" + str(
                                      sid) + "\" Invalid
ID.\n ------------------------------------------\n")
                   self.mty.configure(state='disabled')
                   messagebox.showwarning("Not Found",
"Student ID Not Found.")

           except Exception as e:
               messagebox.showwarning("Value Error",
str(e))
```

7. Admin login Into System

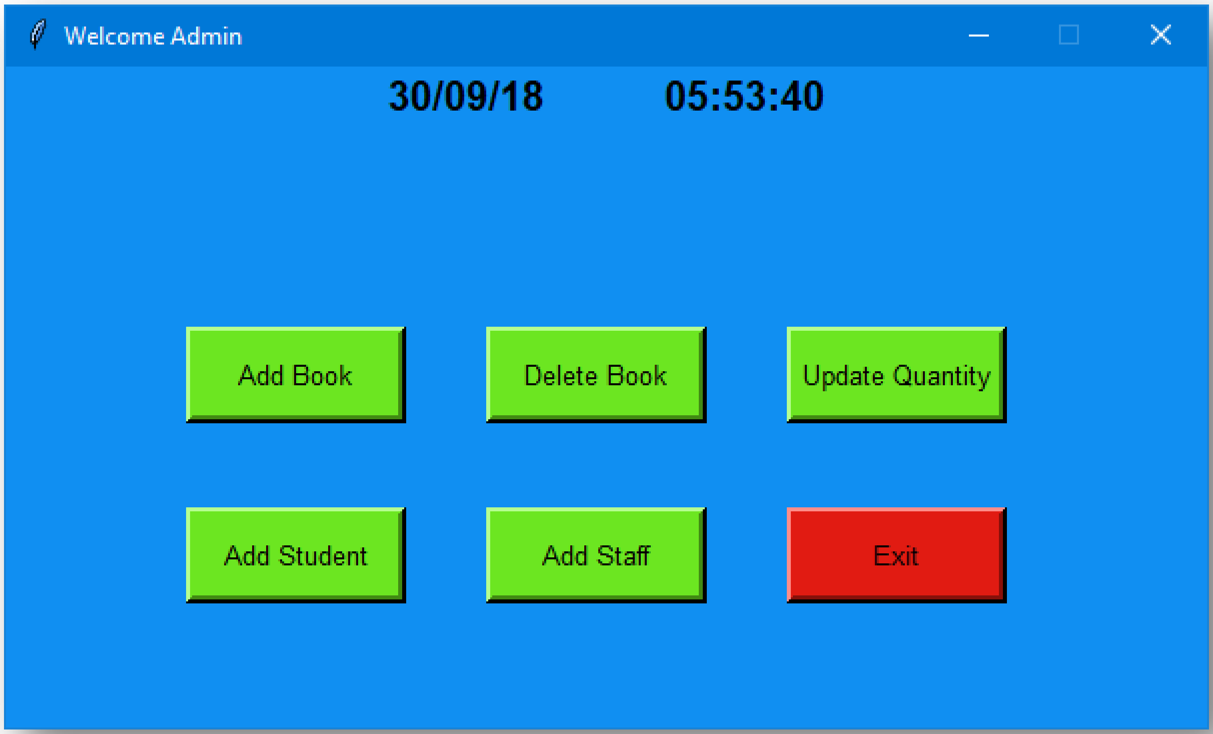
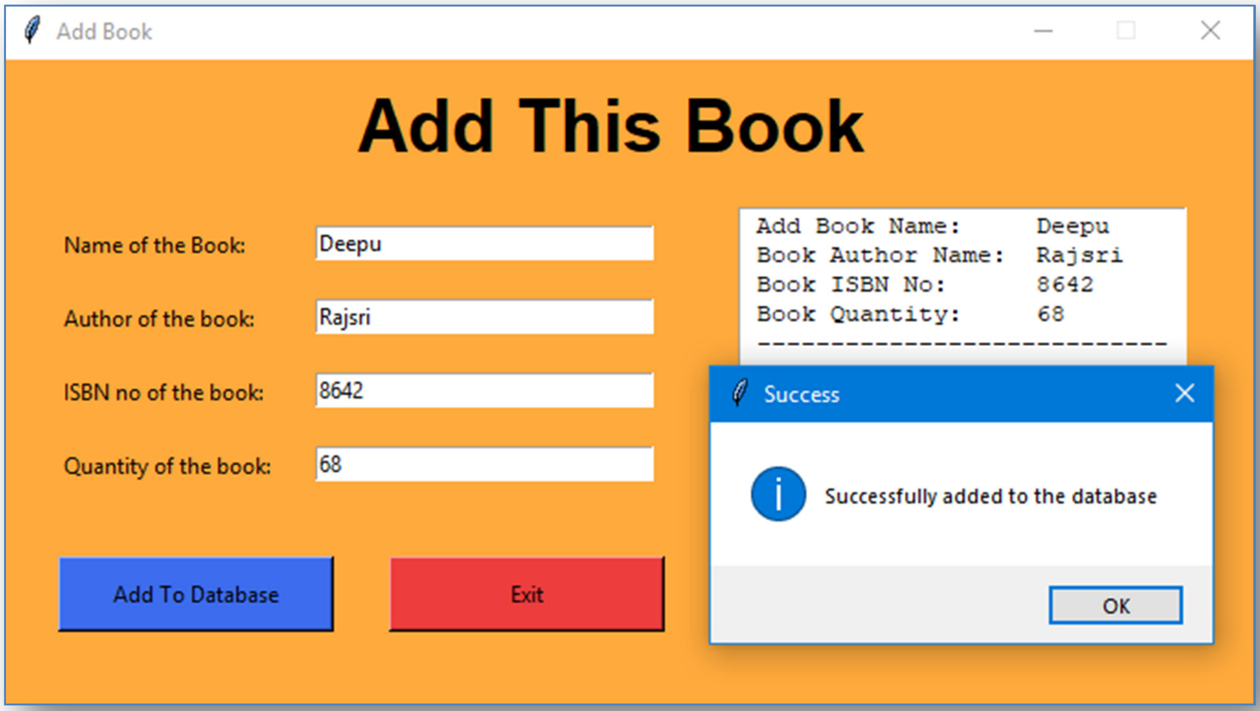


Fig. 7 Screenshot for Admin login Window

```
def admin_call_class():
    root1.destroy()
    root2 = Tk()
    root2.geometry('600x330')
    root2.config(background='#108ff2')
    root2.resizable(False, False)
    root2.title("Welcome Admin")
    center(root2)

    def tick(time1=''):
        time2 = strftime('%I:%M:%S')
        if time2 != time1:
            l1.config(text=time2)
        l1.after(200, tick)

    f1 = Frame(root2)
    f1.pack(side=TOP)
    l1 = Label(f1, font=('arial 15 bold'),
background='#108ff2')
```

```
    l1.pack(side=RIGHT)
    y = strftime("%d/%m/%y         ")
    Label(f1, text=y, font=('arial 15 bold'),
background='#108ff2').pack()
    tick()

    bg = "#6ce621"
    b1 = Button(root2, text="Add Book", font=('arial 10'),
bg=bg, command=call_add_class, borderwidth=4,
                relief=RAISED, width=12,
                height=2)
    b1.place(x=90, y=130)
    b2 = Button(root2, text="Delete Book", font=('arial
10'), bg=bg, command=call_delete_class, borderwidth=4,
                relief=RAISED, width=12,
                height=2)
    b2.place(x=240, y=130)
    b3 = Button(root2, text="Update Quantity", font=('arial
10'), bg=bg, command=call_update_class, borderwidth=4,
                relief=RAISED,
                width=12, height=2)
    b3.place(x=390, y=130)
    b4 = Button(root2, text="Add Student", font=('arial
10'), bg=bg, command=call_stdadd_class, borderwidth=4,
                relief=RAISED, width=12,
                height=2)
    b4.place(x=90, y=220)
    b4 = Button(root2, text="Add Staff", font=('arial 10'),
bg=bg, command=call_stfadd_class, borderwidth=4,
                relief=RAISED, width=12,
                height=2)
    b4.place(x=240, y=220)
    b6 = Button(root2, text="Exit", font=('arial 10'),
bg="#e11b12", command=quit, borderwidth=4, relief=RAISED,
                width=12, height=2)
    b6.place(x=390, y=220)

    root2.mainloop()
    pass
```

8. Add Book Window



```
Add Book Name:        Deepu
Book Author Name:     Rajsri
Book ISBN No:         8642
Book Quantity:        68
----------------------------
```

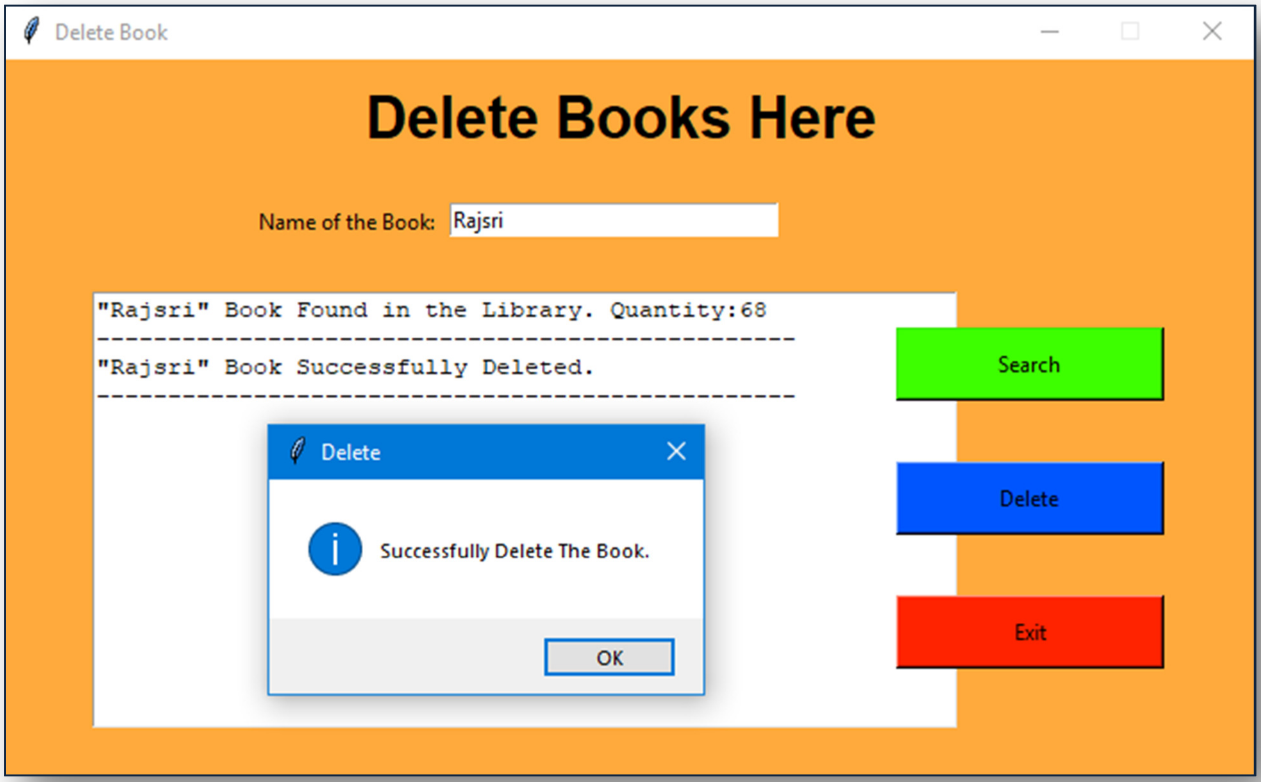
Successfully added to the database

Fig. 8 Screenshot for Add Book Window

```
def call_add_class():
    root_a = Tk()
    root_a.geometry('680x350')
    root_a.resizable(False, False)
    root_a.config(background='#ffaa3c')
    root_a.title("Add Book")
    center(root_a)
    obj = Add(root_a)
    root_a.mainloop()
    pass

class Add(Text):

    def __init__(self, master):
        Text.__init__(self, master)

        # heading for the main window
        self.heading = Label(master, text="Add This Book",
font=('arial 30 bold'), bg='#ffaa3c')
        self.heading.place(x=190, y=10)

        # labels for name date author genre position
        self.name = Label(master, text="Name of the Book: 
", bg='#ffaa3c')
        self.author = Label(master, text="Author of the 
book: ", bg='#ffaa3c')
        self.genre = Label(master, text="ISBN no of the 
book: ", bg='#ffaa3c')
        self.position = Label(master, text="Quantity of the 
book: ", bg='#ffaa3c')

        self.name.place(x=30, y=90)
        self.author.place(x=30, y=130)
        self.genre.place(x=30, y=170)
        self.position.place(x=30, y=210)

        # entries for labels
        self.name_ent = Entry(master, width=30)
        self.name_ent.focus_set()
        self.name_ent.place(x=170, y=90)
        self.author_ent = Entry(master, width=30)
        self.author_ent.place(x=170, y=130)
        self.genre_ent = Entry(master, width=30)
        self.genre_ent.place(x=170, y=170)
        self.position_ent = Entry(master, width=30)
        self.position_ent.place(x=170, y=210)

        # button to perform
        self.submit = Button(master, text="Add To 
Database", command=self.dynamic_data_entry, width=20, 
height=2,
                             bg='#3D6DEE')
        self.submit.place(x=30, y=270)

        self.quit = Button(master, text="Exit", width=20, 
height=2, command=master.destroy, bg='#EE3D3D')
        self.quit.place(x=210, y=270)

        # textbox to display updates
        self.box = Text(master, height=14, width=30)
        self.box.focus_set()
        self.box.configure(state='disabled')
        self.box.place(x=400, y=80)

        # Now adding the user input to database.
        global conn
        global c
        conn = sqlite3.connect('lib.db')
        c = conn.cursor()
```

```
    def dynamic_data_entry(self):
        # creating the database if it doesnot exist
        c.execute(
            "CREATE TABLE IF NOT EXISTS lib_book(datestamp
TEXT, book_name TEXT, author_name TEXT, isbn_no INTEGER
PRIMARY KEY, quantity INTEGER)")

        # adding fields and values to the databse
        unix = time()
        datestamp =
str(datetime.datetime.fromtimestamp(unix).strftime('%Y-%m-
%d %H:%M:%S'))

        try:
            self.book_name = self.name_ent.get()
            self.author_name = self.author_ent.get()
            self.isbn_no = self.genre_ent.get()
            self.quantity = self.position_ent.get()

            if (len(self.book_name) == 0 or
len(self.author_name) == 0 or len(str(self.isbn_no)) == 0
or len(
                    str(self.quantity)) == 0):
                messagebox.showwarning("Failed", "Please
don't leave anything blank.")

            else:
                book_name = self.book_name
                author_name = self.author_name
                isbn_no = int(self.isbn_no)
                quantity = int(self.quantity)

                c.execute(
                    "INSERT INTO lib_book(datestamp,
book_name, author_name, isbn_no, quantity) VALUES (?, ?, ?,
?, ?)",
                    (datestamp, book_name, author_name,
int(isbn_no), int(quantity)))
                conn.commit()

                content = (" Add Book Name:      " +
book_name +
                           "\n Book Author Name:   " +
author_name +
                           "\n Book ISBN No:       " +
str(isbn_no) +
                                          "\n Book Quantity:      " +
str(quantity) +
                                          "\n -----------------------------
\n")

                         self.box.configure(state='normal')
                         self.box.insert(END, content)
                         self.box.configure(state='disabled')

                         messagebox.showinfo("Success",
"Successfully added to the database")

                 except Exception as e:
                     messagebox.showwarning("Value Error", str(e))
```

9. Delete Book Window



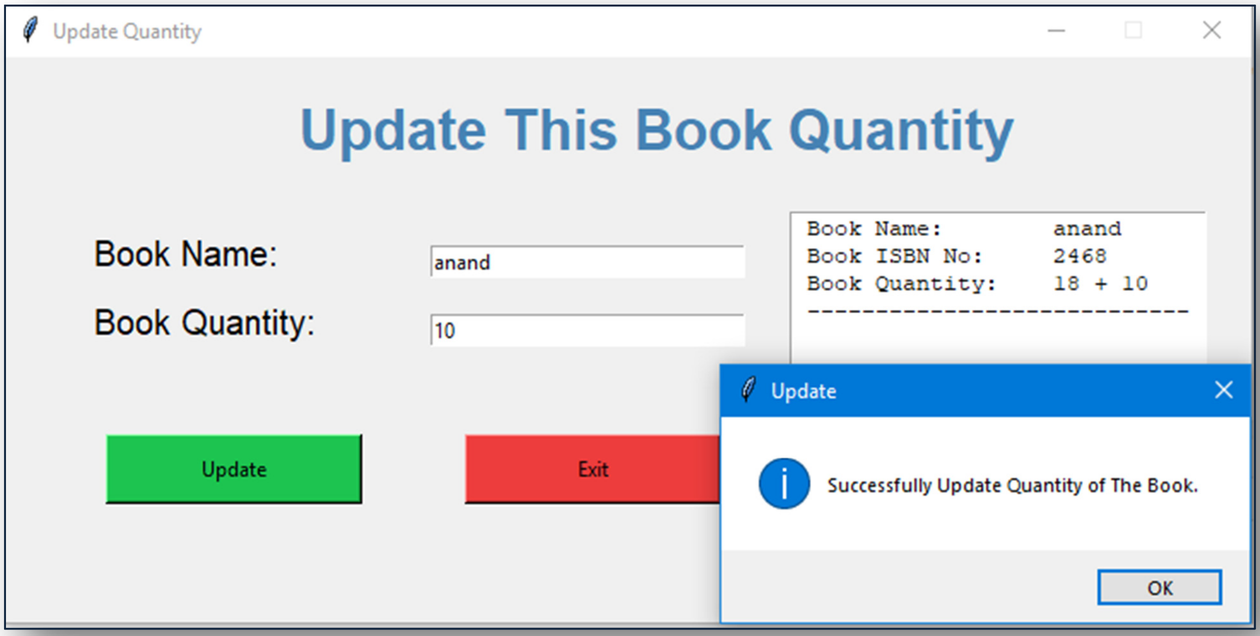
```
"Rajsri" Book Found in the Library. Quantity:68
-------------------------------------------------
"Rajsri" Book Successfully Deleted.
-------------------------------------------------
```

Successfully Delete The Book.

Fig. 9 Screenshot for Delete Book Window

```
def call_delete_class():
    root_d = Tk()
    root_d.geometry('700x400')
    root_d.resizable(False, False)
    root_d.config(background='#ffaa3c')
    root_d.title("Delete Book")
    center(root_d)
    obj = Delete(root_d)
    root_d.mainloop()
    pass


class Delete(Text):
    def __init__(self, faster):
        Text.__init__(self, faster)

        self.heading = Label(faster, text="Delete Books Here",
font=('arial 25 bold'), bg='#ffaa3c')
        self.heading.place(x=200, y=10)

        self.name = Label(faster, text="Name of the Book: ",
bg='#ffaa3c')
        self.name.place(x=140, y=80)

        self.ent = Entry(faster, width=30)
        self.ent.focus_set()
        self.ent.place(x=250, y=80)

        self.sbox = Text(faster, height=15, width=60,
bg="white")
        self.sbox.place(x=50, y=130)
        self.sbox.configure(state='disabled')

        self.b1 = Button(faster, text="Search",
command=self.get_it, width=20, height=2, bg='#3eff00')
        self.b1.place(x=500, y=150)
        self.b2 = Button(faster, text="Delete",
command=self.del_it, width=20, height=2, bg='#0055ff')
        self.b2.place(x=500, y=225)
        self.b3 = Button(faster, text="Exit",
command=faster.destroy, width=20, height=2, bg='#ff2300')
        self.b3.place(x=500, y=300)

    def get_it(self):

        conn = sqlite3.connect('lib.db')
        c = conn.cursor()
        c.execute(
            "CREATE TABLE IF NOT EXISTS lib_book(datestamp
TEXT, book_name TEXT, author_name TEXT, isbn_no INTEGER
PRIMARY KEY, quantity INTEGER)")
        bk = self.ent.get()

        if len(bk) == 0:
            messagebox.showwarning("Failed", "Please,  Enter
Book Name.")

        else:
            c.execute("SELECT * FROM lib_book WHERE book_name
= ?", [bk])
            data = c.fetchall()
            if data:
                z = c.execute("SELECT * FROM lib_book WHERE
book_name = ?", [bk])
                for q in z:
                    self.sbox.configure(state='normal')
                    self.sbox.insert(END, "\"" + bk + "\" Book
Found in the Library. Quantity:" + str(q[4]) +
                                     "\n------------------------
---------------------------\n")
                    self.sbox.configure(state='disabled')

            else:
                self.sbox.configure(state='normal')
                self.sbox.insert(END, "\"" + bk + "\" No Such
Book Found in the Library."
                                                  "\n---------
----------------------------------------\n")
                self.sbox.configure(state='disabled')
                messagebox.showwarning("Not Found", "Sorry,
Book Not Found.")

        c.close()
        conn.close()

    def del_it(self):

        conn = sqlite3.connect('lib.db')
        c = conn.cursor()
        c.execute(
            "CREATE TABLE IF NOT EXISTS lib_book(datestamp
TEXT, book_name TEXT, author_name TEXT, isbn_no INTEGER
PRIMARY KEY, quantity INTEGER)")
        bk = self.ent.get()
        if len(bk) == 0:
            messagebox.showwarning("Failed", "Please,  Enter
Book Name.")

        else:
            c.execute("SELECT * FROM lib_book WHERE book_name
= ?", [bk])
            data = c.fetchall()
            if data:

                c.execute("delete from lib_book where
book_name= ?", [bk])
                conn.commit()

                self.sbox.configure(state='normal')
                self.sbox.insert(END, "\"" + bk + "\" Book
Successfully Deleted." +
                                 "\n--------------------------
-----------------------\n")
                self.sbox.configure(state='disabled')

                messagebox.showinfo("Delete", "Successfully
Delete The Book.")


            else:
                self.sbox.configure(state='normal')
                self.sbox.insert(END, "\"" + bk + "\" No Such
Book Found in the Library."
                                                  "\n---------
----------------------------------------\n")
                self.sbox.configure(state='disabled')
                messagebox.showwarning("Not Found", "Sorry,
Book Not Found.")

        c.close()
        conn.close()
```

10. Update Quantity Window



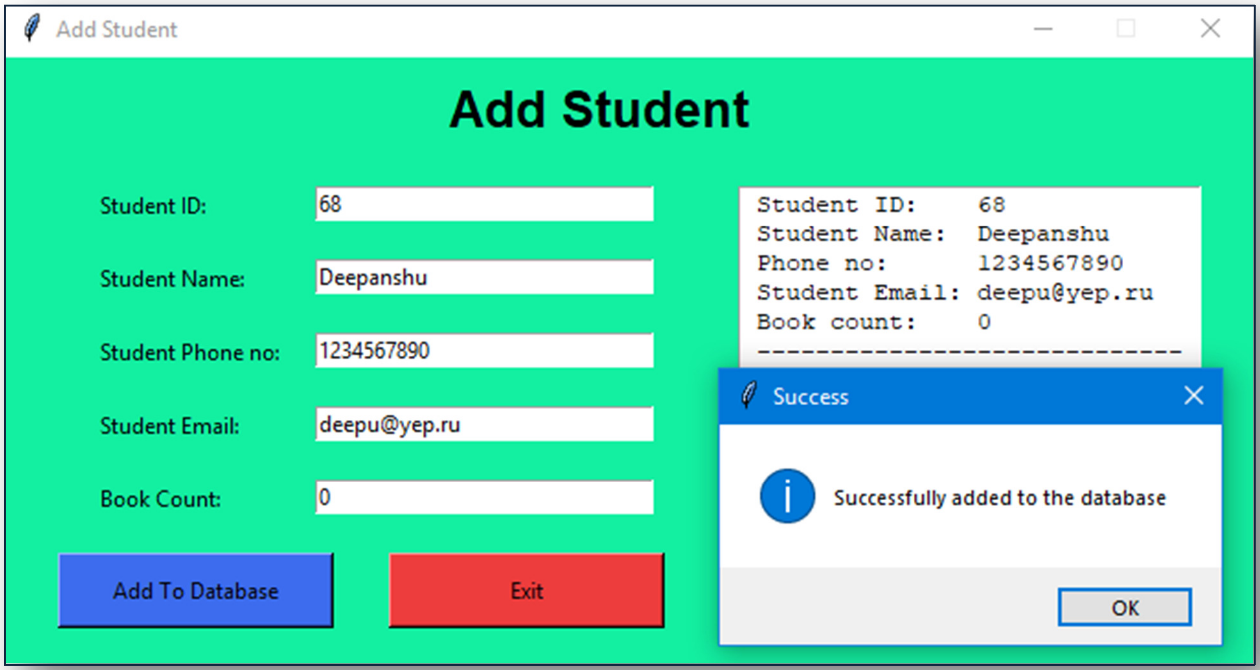
```
Book Name:          anand
Book ISBN No:       2468
Book Quantity:      18 + 10
------------------------------
```

Successfully Update Quantity of The Book.

Fig. 10 Screenshot for Update Quantity Window

```
def call_update_class():
    root_u = Tk()
    root_u.geometry('730x330')
    root_u.resizable(False, False)
    root_u.title("Update Quantity")
    center(root_u)
    obj = Update(root_u)
    root_u.mainloop()
    pass

class Update(Text):
    def __init__(self, fram):
        Text.__init__(self, fram)
        self.hd = Label(fram, text="Update This Book
Quantity", font=('arial 25 bold'), fg='steelblue')
        self.hd.place(x=170, y=20)

        # info of the book taker
        self.id = Label(fram, text="Book Name:",
font=('arial 15'))
        self.id.place(x=50, y=100)
        self.book = Label(fram, text="Book Quantity:",
font=('arial 15'))
        self.book.place(x=50, y=140)

        # entries for the infos
        self.bname = Entry(fram, width=30)
        self.bname.place(x=250, y=110)
        self.bname.focus_set()
        self.quantity = Entry(fram, width=30)
        self.quantity.place(x=250, y=150)

        self.b1 = Button(fram, text="Update", width=20,
height=2, command=self.update_Quantity,
bg='#1DC550').place(
            x=60, y=220)
        self.b2 = Button(fram, text="Exit", width=20,
height=2, command=fram.destroy, bg='#EE3D3D').place(x=270,

y=220)

        self.mty = Text(fram, height=13, width=30)
        self.mty.place(x=460, y=90)
        self.mty.configure(state='disabled')

    def update_Quantity(self):
        conn = sqlite3.connect('lib.db')
        c = conn.cursor()
        self.b_name = self.bname.get()
        self.b_q = self.quantity.get()

        if self.b_name == '' or self.b_q == '':
            messagebox.showwarning("Error", "Please Fill
The Missing Boxes")

        else:
            try:
                bname = self.b_name
                bq = int(self.b_q)
                c.execute("SELECT * FROM lib_book WHERE
book_name = ?", [bname])
                data = c.fetchall()

                if data:
                    z = c.execute("SELECT * FROM lib_book
WHERE book_name = ?", [bname])
                    for p in z:
                        pass
                    content = (" Book Name:        " + p[1]
+
                               "\n Book ISBN No:     " +
str(p[3]) +
                               "\n Book Quantity:    " +
str(p[4]) + " + " + str(bq) +
                               "\n -----------------------
----\n")

                    self.mty.configure(state='normal')
                    self.mty.insert(END, content)
                    self.mty.configure(state='disabled')

                    j = p[4] + bq
                    k = bname
                    c.execute("update lib_book set quantity
= ? where book_name= ?", (j, k))
                    conn.commit()

                    messagebox.showinfo("Update",
"Successfully Update Quantity of The Book.")

                else:
                    self.mty.configure(state='normal')
                    self.mty.insert(END, " \"" + str(bname)
+ "\" Book Not Found.\n ----------------------------\n")
                    self.mty.configure(state='disabled')
                    messagebox.showwarning("Not Found",
"Invalid Book Name.")

            except Exception as e:
                messagebox.showwarning("Value Error",
str(e))
```

11. Add Student Window



Student ID: 68

Student Name: Deepanshu

Student Phone no: 1234567890

Student Email: deepu@yep.ru

Book Count: 0

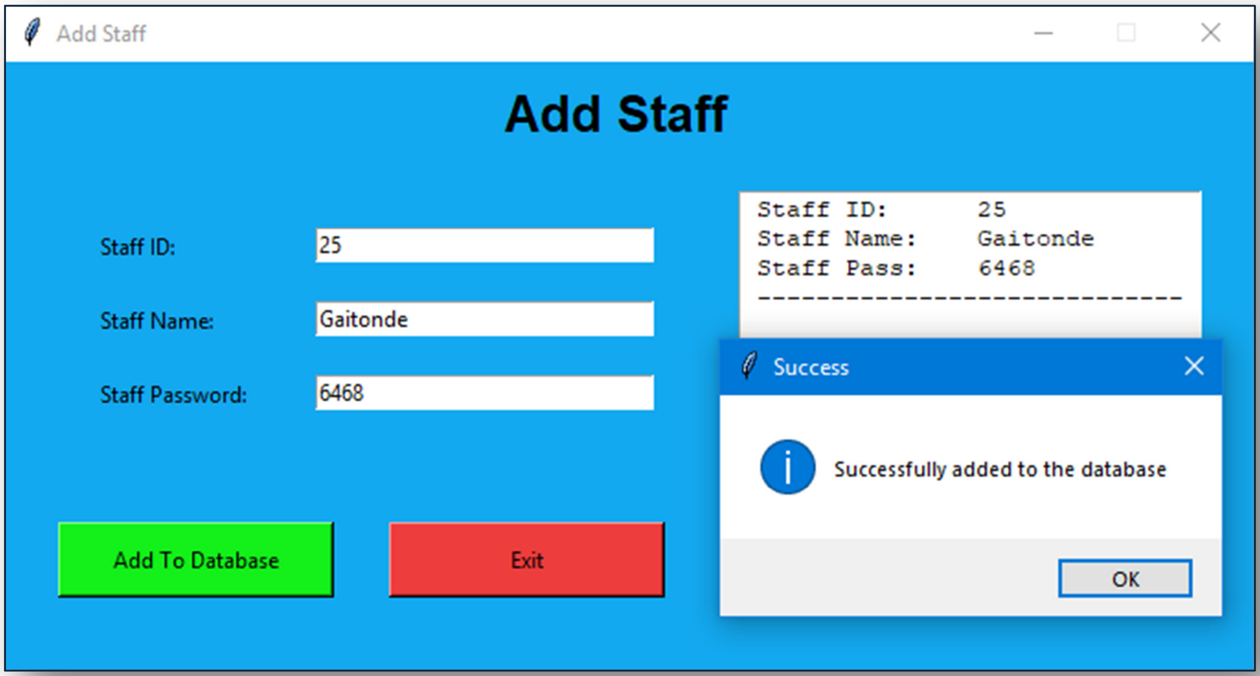
```
Student ID:     68
Student Name:   Deepanshu
Phone no:       1234567890
Student Email:  deepu@yep.ru
Book count:     0
----------------------------
```

Successfully added to the database

Fig. 11 Screenshot for Add Student Window

```
def call_stdadd_class():
    root_sa = Tk()
    root_sa.geometry('680x330')
    root_sa.resizable(False, False)
    root_sa.config(background='#13f0a2')
    root_sa.title("Add Student")
    center(root_sa)
    obj = Add_std(root_sa)
    root_sa.mainloop()
    pass


class Add_std(Text):

    def __init__(self, master):
        Text.__init__(self, master)
        bg = "#13f0a2"
        self.heading = Label(master, text="Add Student",
font=('arial 20 bold'), bg=bg)
        self.heading.place(x=240, y=10)

        # labels for name date author genre position
        self.id = Label(master, text="Student ID: ", bg=bg)
        self.name = Label(master, text="Student Name: ",
bg=bg)
        self.pno = Label(master, text="Student Phone no: ",
bg=bg)
        self.email = Label(master, text="Student Email: ",
bg=bg)
        self.bcount = Label(master, text="Book Count: ",
bg=bg)

        self.id.place(x=50, y=70)
        self.name.place(x=50, y=110)
        self.pno.place(x=50, y=150)
        self.email.place(x=50, y=190)
        self.bcount.place(x=50, y=230)

        # entries for labels
        self.id_ent = Entry(master, width=30)
        self.id_ent.place(x=170, y=70)
        self.id_ent.focus_set()
        self.name_ent = Entry(master, width=30)
        self.name_ent.place(x=170, y=110)
        self.pno_ent = Entry(master, width=30)
        self.pno_ent.place(x=170, y=150)
        self.email_ent = Entry(master, width=30)
        self.email_ent.place(x=170, y=190)
        self.bcount_ent = Entry(master, width=30)
        self.bcount_ent.place(x=170, y=230)

        # button to perform
        self.submit = Button(master, text="Add To
Database", command=self.dynamic_data_entry, width=20,
height=2,
                             bg='#3D6DEE')
        self.submit.place(x=30, y=270)

        self.quit = Button(master, text="Exit", width=20,
height=2, command=master.destroy, bg='#EE3D3D')
        self.quit.place(x=210, y=270)

        # textbox to display updates
        self.box = Text(master, height=14, width=31)
        self.box.focus_set()
        self.box.configure(state='disabled')
        self.box.place(x=400, y=70)

        # Now adding the user input to database.
        global conn
        global c
        conn = sqlite3.connect('lib.db')
        c = conn.cursor()

    def dynamic_data_entry(self):
        # creating the database if it doesnot exist
        c.execute(
            "CREATE TABLE IF NOT EXISTS std_info(datestamp
TEXT, std_id INTEGER PRIMARY KEY, std_name TEXT NOT NULL,
std_pno INTEGER, std_email TEXT, book_count TEXT)")

        # adding fields and values to the databse
        unix = time()
        datestamp =
str(datetime.datetime.fromtimestamp(unix).strftime('%Y-%m-
%d %H:%M:%S'))

        self.id = self.id_ent.get()
        self.name = self.name_ent.get()
        self.pno = self.pno_ent.get()
        self.email = self.email_ent.get()
        self.bcount = self.bcount_ent.get()

        if (len(self.id) == 0 or len(self.name) == 0 or
len(self.pno) == 0 or len(self.email) == 0 or len(
                self.bcount) == 0):
            messagebox.showwarning("Failed", "Please don't
leave anything blank.")

        else:
            try:
                id = int(self.id)
                name = self.name
                pno = int(self.pno)
                email = self.email
                bcount = int(self.bcount)
                c.execute(
                    "INSERT INTO std_info(datestamp,
std_id, std_name, std_pno, std_email, book_count) VALUES
(?, ?, ?, ?, ?, ?)",
                    (datestamp, int(id), name, int(pno),
email, int(bcount)))
                conn.commit()

                content = (" Student ID:     " + str(id) +
                           "\n Student Name:   " + name +
                                    "\n Phone no:       " + str(pno)
+
                                    "\n Student Email: " + email +
                                    "\n Book count:     " +
str(bcount) +
                                    "\n -----------------------------
-\n")

                        self.box.configure(state='normal')
                        self.box.insert(END, content)
                        self.box.configure(state='disabled')

                        messagebox.showinfo("Success",
"Successfully added to the database")

                    except Exception as e:
                        messagebox.showwarning("Value Error",
str(e))
```

12. Add Staff Window



Staff ID: 25
Staff Name: Gaitonde
Staff Password: 6468

```
Staff ID:       25
Staff Name:     Gaitonde
Staff Pass:     6468
-----------------------------
```

Successfully added to the database

Fig. 12 Screenshot for Add Staff Window

```
 def call_stfadd_class():
     root_sfa = Tk()
     root_sfa.geometry('680x330')
     root_sfa.resizable(False, False)
     root_sfa.config(background='#13a9f0')
     root_sfa.title("Add Staff")
     center(root_sfa)
     obj = Add_stf(root_sfa)
     root_sfa.mainloop()
     pass

 class Add_stf(Text):

    def __init__(self, master):
        Text.__init__(self, master)
        bg = "#13a9f0"
        self.heading = Label(master, text="Add Staff",
font=('arial 20 bold'), bg=bg)
        self.heading.place(x=270, y=10)

        # labels for name date author genre position
        self.id = Label(master, text="Staff ID: ", bg=bg)
        self.name = Label(master, text="Staff Name: ",
bg=bg)
        self.pno = Label(master, text="Staff Password: ",
bg=bg)

        self.id.place(x=50, y=90)
        self.name.place(x=50, y=130)
        self.pno.place(x=50, y=170)

        # entries for labels
        self.id_ent = Entry(master, width=30)
        self.id_ent.place(x=170, y=90)
        self.id_ent.focus_set()
        self.name_ent = Entry(master, width=30)
        self.name_ent.place(x=170, y=130)
        self.pass_ent = Entry(master, width=30)
        self.pass_ent.place(x=170, y=170)

        # button to perform
        self.submit = Button(master, text="Add To
Database", command=self.dynamic_data_entry, width=20,
height=2,
                                     bg='#13f01a')
        self.submit.place(x=30, y=250)
        self.quit = Button(master, text="Exit", width=20,
height=2, command=master.destroy, bg='#EE3D3D')
        self.quit.place(x=210, y=250)

        # textbox to display updates
        self.box = Text(master, height=14, width=31)
        self.box.focus_set()
        self.box.configure(state='disabled')
        self.box.place(x=400, y=70)

        # Now adding the user input to database.
        global conn
        global c
        conn = sqlite3.connect('lib.db')
        c = conn.cursor()

    def dynamic_data_entry(self):
        # creating the database if it doesnot exist
        c.execute(
            "CREATE TABLE IF NOT EXISTS
staff_info(datestamp TEXT, stf_id INTEGER PRIMARY KEY,
stf_name TEXT NOT NULL, stf_pass INTEGER)")

        # adding fields and values to the databse
        unix = time()
        datestamp =
str(datetime.datetime.fromtimestamp(unix).strftime('%Y-%m-
%d %H:%M:%S'))

        try:
            id = int(self.id_ent.get())
            name = self.name_ent.get()
            pas = int(self.pass_ent.get())

            if (len(str(id)) == 0 or len(name) == 0 or
len(str(pas)) == 0):
                messagebox.showwarning("Failed", "Please
don't leave anything blank.")

            else:
                c.execute(
                    "INSERT INTO staff_info(datestamp,
stf_id, stf_name, stf_pass) VALUES (?, ?, ?, ?)",
                    (datestamp, int(id), name, int(pas)))
                conn.commit()

                content = (" Staff ID:      " + str(id) +
                                  "\n Staff Name:     " + name +
                                  "\n Staff Pass:     " + str(pas)
+
                                  "\n ----------------------------
-\n")

                    self.box.configure(state='normal')
                    self.box.insert(END, content)
                    self.box.configure(state='disabled')

                    messagebox.showinfo("Success",
"Successfully added to the database")

          except Exception as e:
              messagebox.showwarning("Value Error", str(e))
```

CHAPTER 5
TESTING AND RESULT

Every program or software has errors in it, against the common view that there are no errors in it if the program or software is working. Executing the programs with the intention of finding the errors in it is therefore testing; hence a successful test is one which finds errors. Testing is an activity; however, it is restricted to being performed after the development phase is complete but is carried parallel with all stages of system development, starting with requirement specification.

A test case is a set of the data that a system will process as normal input. The software units developed in the system are modules and routines that are assembled and integrated to perform the required function of the system. Test results once gathered and evaluated, provide a qualitative indication of the software quality and reliability and serve as basis for design modification if required. The testing phase of the implementations works accurately and efficiently before live operation commences.

CHAPTER 6
Conclusions & Future Scope

This software provides a computerized version of library management system which will benefit the students as well as the staff of the library.

In this student can search books, staff can issue and return book. It also has a facility for student login where student can login and can see quantity of book. It has a facility of staff login and admin login where admin can add student and add staff.

There is a future scope of this facility that many more features such as maintain book issue records added to this project thus making it more interactive more user friendly and project which fulfils each user need in the best way possible.

CHAPTER 7
References

Books

- ➢ Software Engineering 3rd Edition (K. K. Aggarwal)
- ➢ Fundamentals of Database System Seventh Edition (Ramez Elmasri, Shamkant B. Navathe)
- ➢ Python the Complete Reference (Martin C. Brown)
- ➢ Using SQLite (Jay A. Kreibich)

Websites

- ✓ https://www.w3schools.com/python/
- ✓ https://www.tutorialspoint.com/sqlite/
- ✓ https://www.tutorialspoint.com/dbms/
- ✓ http://www.w3schools.com/sql/sql_update.asp