Library Management

Signature Assignment Project

Nauka Shah : 24083814
Aditi Parikh : 95991

Problem Statement

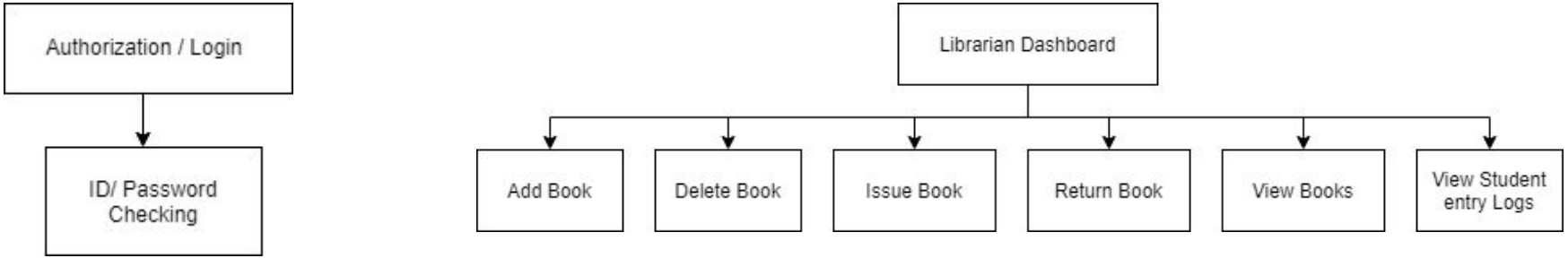
- The Project is for monitoring and controlling the transactions in a library. It is developed in Python and mainly focuses on basic operations

- This project also includes a module of Artificial Intelligence - Students log entries by their Face recognition. It is useful for the librarian or administrator to sometimes check the records of student entry and exit logs.

System Environment

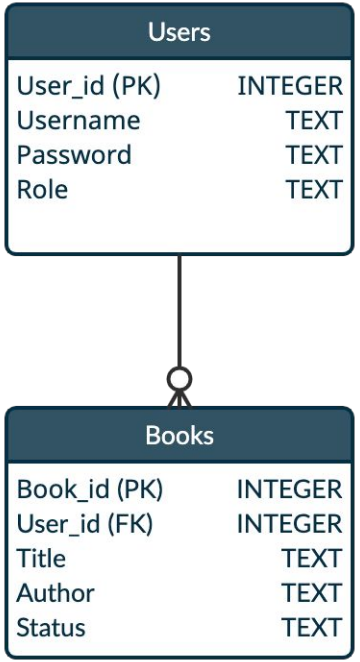
- Language - Python
- UI - Tkinter
- Libraries Used - OS, OpenCV, Dlib, Face_Recognition, NumPy, CSV, DateTime
- Database - Sqlite3

System modules



Database structure

```
conn = None
conn = sqlite3.connect('library_info.db')
conn.execute("PRAGMA foreign_keys = 1")
cur = conn.cursor()
try:
   cur.execute('''CREATE TABLE IF NOT EXISTS Users(User_id INTEGER PRIMARY KEY NOT NULL,
                                        Username TEXT UNIQUE,
                                        Password TEXT,
                                        Role TEXT)''')
   conn.commit()

except sqlite3.Error as err:
   print('Database Error ', err)

print("creating another table..")
cur.execute('''CREATE TABLE IF NOT EXISTS Books(Book_id INTEGER PRIMARY KEY NOT NULL, Title
TEXT,
                                        Author TEXT,
                                        Status TEXT,
                                        User_id INTEGER NOT NULL,
                                        FOREIGN KEY (User_id) REFERENCES Users (User_id)
                                        ON DELETE CASCADE
                                        ON UPDATE CASCADE)''')
```



Book management module - Login

```
def currentLogin():
    name = entry1.get()
    password = entry2.get()
    cur.execute('''SELECT User_id FROM
Users WHERE Users.Username = ? AND
Users.Password = ?''', (name,
password))
    result = cur.fetchall()
    for row in result:
        print(f'from login : id
{row[0]:<3}')
    return result
```

- On button click **command=lambda: validate()**
- Validate username and password, if matches to the database then perform further
- Also, using **global logged_in_id** we are storing current logged in user's id
- GUI : Frame, label, entry, buttons

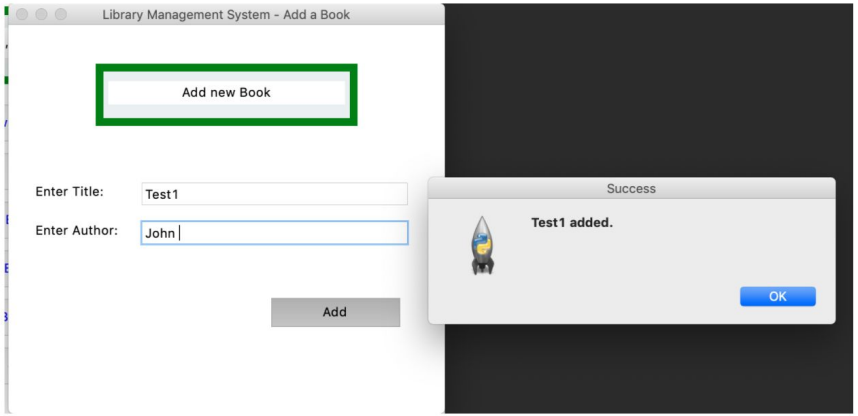
```
def validate():
    print("in validate function 2")
    name = entry1.get()
    password = entry2.get()
    cur.execute('''SELECT * FROM Users WHERE
Users.Username = ? AND Users.Password = ?''', (name,
password))
    results = cur.fetchall()
    if len(results) == 1:
        print("validation matches and perform book
operations")
        cur.execute('''SELECT User_id FROM Users WHERE
Users.Username = ? AND Users.Password = ?''',
                    (name, password))
        result = cur.fetchall()
        global logged_in_id
        issueBook.logged_in_id = result
        returnBook.logged_in_id = result
        manageBookOperations()
    else:
        messagebox.showerror("Error", "Invalid
Credentials,Please try again!")
```

Book management module - Book home

```
addbtn = Button(window, text="Add New Book", command=addBooks, bg="#455A64", fg="blue")
addbtn.place(relx=0.35, rely=0.30, relwidth=0.30, relheight=0.08)

deletebtn = Button(window, text="Delete a Book", command=deleteBooks, bg="#455A64",
fg="blue")
deletebtn.place(relx=0.35, rely=0.40, relwidth=0.30, relheight=0.08)

issuebtn = Button(window, text="Issue a Book", command=issueBooks, bg="#455A64",
fg="blue")
issuebtn.place(relx=0.35, rely=0.50, relwidth=0.30, relheight=0.08)

returnbtn = Button(window, text="Return Book", command=returnBooks, bg="#455A64",
fg="blue")
returnbtn.place(relx=0.35, rely=0.60, relwidth=0.30, relheight=0.08)

viewbtn = Button(window, text="List of Books", command=viewBooks, bg="#455A64", fg="blue")
viewbtn.place(relx=0.35, rely=0.70, relwidth=0.30, relheight=0.08)

viewbtn = Button(window, text="View Student entry Logs", command=read_csv, bg="#455A64",
fg="blue")
viewbtn.place(relx=0.35, rely=0.80, relwidth=0.30, relheight=0.08)
```

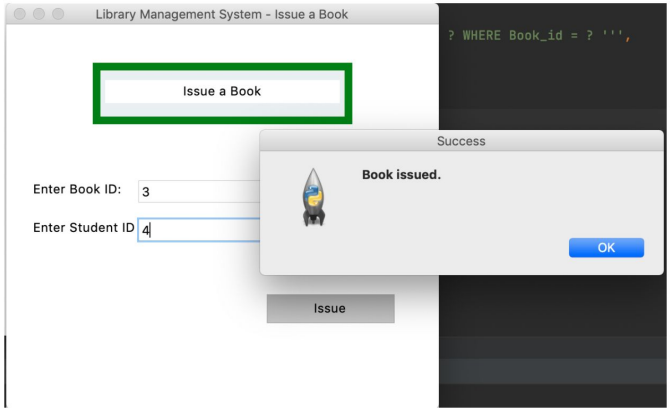
Book management module - Add Book

```
try:
    if btitle == '' or bauthor == '':
        messagebox.showinfo('Warning', "Input required")
    else:
        cursor.execute('''INSERT INTO Books(Title, Author,
Status, User_id )
                          VALUES(?, ?, ?, ?)''',
(btitle, bauthor, "available", 1))

        conn.commit()
        messagebox.showinfo('Success', btitle+" added.")
        window.destroy()

except:
    messagebox.showerror("Error", "Could not add given
book data into Database!")
```



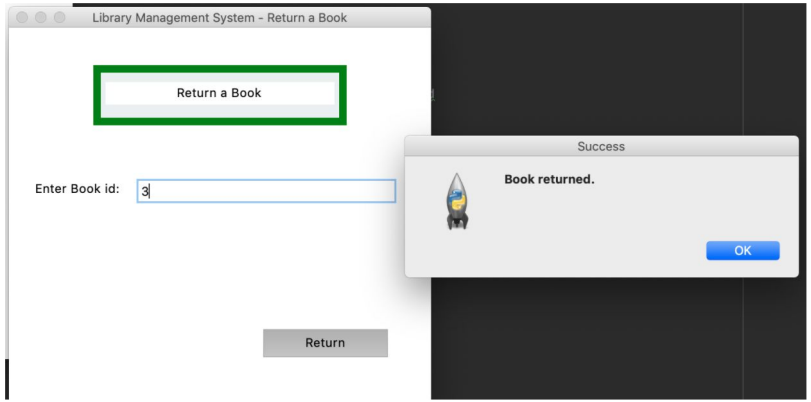
Book management module – Issue Book

```
try:
   cursor.execute('''SELECT Book_id from Books where Status =
'Available' or Status = 'available' ''')
   result = cursor.fetchall()
   if bid == '':
       messagebox.showinfo('Warning', "Input required!")

   for i in result:
       if int(i[0]) == int(bid):
           print("to change value of flag check each i", i[0], bid)
           cursor.execute('''UPDATE Books SET Status = ?, User_id = ?
WHERE Book_id = ? ''',
                          ("no", int(bStudentId), int(bid)))
           conn.commit()
           messagebox.showinfo('Success', "Book issued.")
           window.destroy()
           break
       else:
           print("Error", "Required Book is not available!")
except:
   messagebox.showerror("Error", "Cannot issue given book!")
```



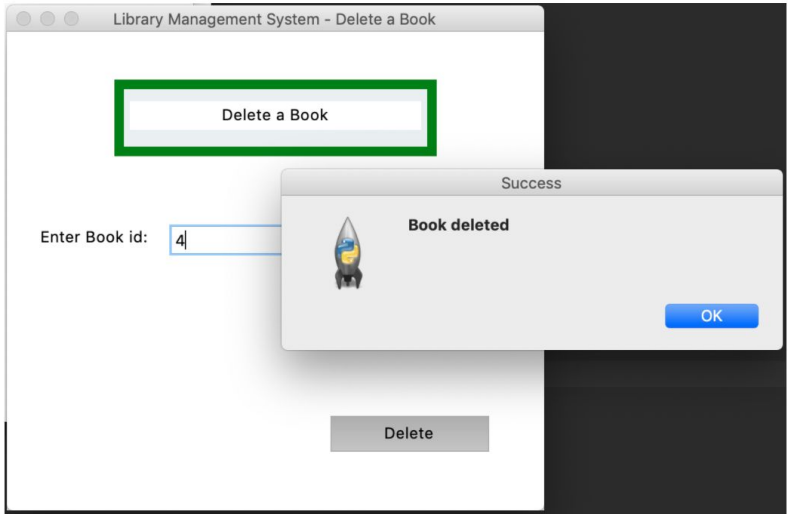
Book management module – Return Book

```
try:
    # update entered book id's status as available and set
user id = tempid
    for i in logged_in_id:
        temp_id = i[0]
    cursor.execute('''UPDATE Books SET Status = ?, User_id =
? WHERE Book_id = ? and Status = ? ''',
                   ("available", int(temp_id), int(bid),
"no"))
    conn.commit()
    messagebox.showinfo('Success', "Book returned.")
    window.destroy()

except:
    messagebox.showerror("Error", "Cannot return given
book!")
```



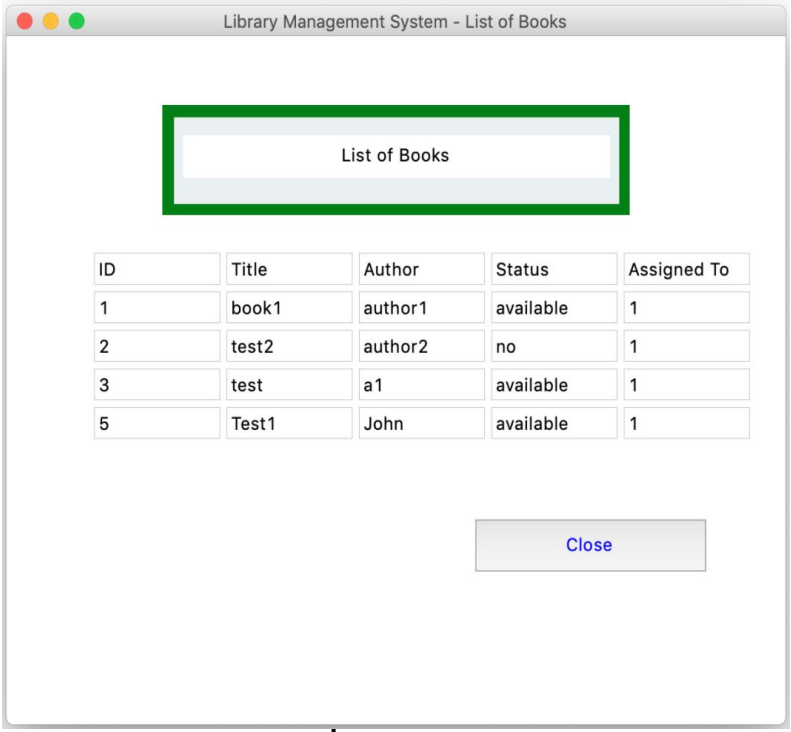
Book management module – Delete Book

```
try:
    print(bid)
    if bid == '':
        messagebox.showinfo('Input required', "please enter
book id to delete")
    else:
        cursor.execute('''DELETE FROM Books WHERE Book_id =
?''', (bid,))
        conn.commit()
        messagebox.showinfo('Success', "Book deleted")
        window.destroy()

except:
    messagebox.showerror("Error", "Book with given id does
not exist!")
```



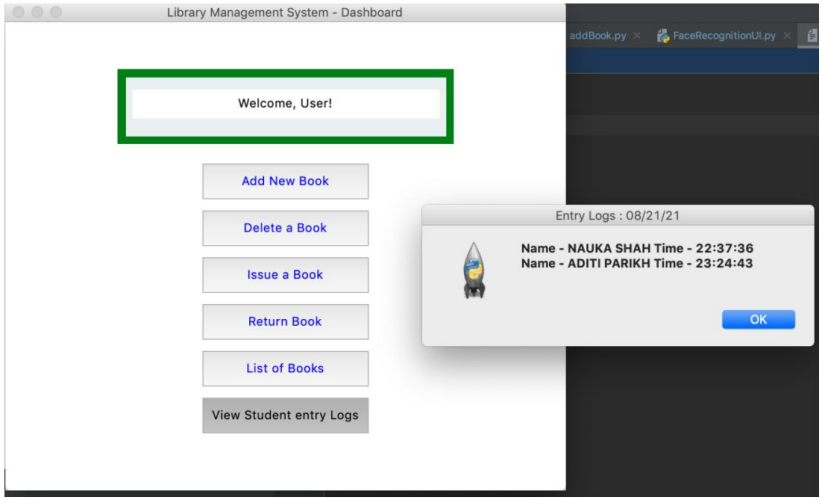
Book management module – View All Books

```
try:
    conn = None
    conn = sqlite3.connect('library_info.db')
    cur = conn.cursor()
    cur.execute('SELECT * FROM Books')
    results = cur.fetchall()
    total_rows = len(results)
    total_columns = len(results[0])

    lst = [('ID', 'Title', 'Author', 'Status', 'Assigned To')]
    for k in range(0, 5):
        e = Entry(tableFrame1, width=10, fg='black')
        e.grid(row=0, column=k)
        e.insert(END, lst[0][k])

    for i in range(total_rows):
        for j in range(total_columns):
            e = Entry(tableFrame1, width=10, fg='black')

            e.grid(row=i + 1, column=j)
            e.insert(END, results[i][j])

except:
    messagebox.showerror("Error", "Cannot Fetch data.")
```



Book management module – View All Logs

```
def read_csv():
        try:
            with open(attendance_csv, 'r') as file:
                reader = csv.reader(file)
                entry_logs = ''
                for row in reader:
                    print(row)
                    if(len(row) == 2):
                        entry_logs = entry_logs +"Name - "+
row[0]+" Time - "+row[1] + "\n"
                print(entry_logs)
                messagebox.showinfo("Entry Logs :
"+today.strftime("%m/%d/%y"),entry_logs)
        except Exception as e :
            messagebox.showerror('Error', e)
```



Face Recognition

```
def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList
```

The Function - **findEncodings** will find the main 128 features of the Human face. The response of this function will be an array of 128 numeric values. Features of face includes - eyes, nose, mouth etc. The numeric values represent the distance between those main features of Human face and they will be unique for each face. In general the human face contains Thousands of these feature numeric values. But this algorithm focuses on the main 128 which will be sufficient to distinguish the faces.

Face Recognition

```
            matches = face_recognition.compare_faces(encodeListKnown, encodeFace)

            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)

            matchIndex = np.argmin(faceDis)

            if matches[matchIndex]:

                name = classNames[matchIndex].upper()

                # print(name)

                y1, x2, y2, x1 = faceLoc

                markAttendance(name)
```

Here, We are using face_recognition lib's method compare_faces to recognize a Face. Input face encodings are matched with the known faces' (stored in database) encodings. If a match found, The face is recognized and named same as the name of the stored person's face.

Face Recognition

```
def markAttendance(name):
    with open(attendance_csv, access) as f:
        myDataList = f.readlines()
        nameList = []
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
            now = datetime.now()
            dtString = now.strftime('%H:%M:%S')
            f.writelines(f'\n{name},{dtString}')
```

The function markAttendance then write this name along with the time the face is recognized to the CSV file. From where we can read the data into any other module.

Conclusion

We have developed this system and tried to accommodate all the major concepts like Tkinter UI, Database functions, Global variables and their usage, Validations of fields and catching exceptions. Apart from these basic concepts, we researched for an advanced concept of AI - Face Recognition and integrated this module in our project. For this we have used OpenCV library and Face-Recognition library.

- More detailed knowledge about the concepts
- Ability to solve the technical issues
- Learned to work in a team
- Project management and Collaboration

Future Enhancement

- Attendance With Face recognition module can be expanded in future.

- We can create a whole new table of attendance which will keep track of attendance of students. In our project, we have considered only entry logs of students entering the library, but in future, similarly We can also implement the feature of capturing the students who are exiting the library in the module

- We are thinking of extending the system and have Role based login. We will expand this system’s scope to have two roles - Librarian and Student.

- All features available in the dashboard can be based on the role.

References

1. https://realpython.com/face-recognition-with-python/
2. https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1
3. https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e
4. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Thank You!

Please find the code for this project from our repository:
https://github.com/itu-edu/library-management