- **Introduction to assignment**

I've created a Book Catalogue web application where the user can create a new account on it and search for a book based on its ISBN number. Also this application allows the user to save a list of books searched by him and to visualise the book list in a tabular format. The user can delete a book from the table.

- **Requirements specifications of the assignment**

To use this application Python should be installed in the system and the required libraries must be installed. I've put the instructions to run this application in a README.md file inside code folder.

- **Source Code**

Source code is available in the code folder. Main logic is available in the app.py file.

- **Application and Implementation**

Main entry point of the application is app.py file. In that I've imported all the required dependencies. I begin by initialising the Flask application, LoginManager and SQLAlchemy instances.

```
app = Flask(__name__)
app.config.from_object(BaseConfig)
db = SQLAlchemy(app)
migrate = Migrate(app,db)
login = LoginManager(app)
login.login_view = 'login'
```

For this application I am using SQLite Database. I've create two tables – one for users and one for books saved.

Below is the Schema for Books table –

```
    id = db.Column(db.Integer, primary_key=True)
    isbn = db.Column(db.String(128))
    username = db.Column(db.String(64), index=True)
    title = db.Column(db.String(128))
    authors = db.Column(db.String(128))
    pageCount = db.Column(db.Integer)
    rating = db.Column(db.String(128))
```

We are storing ISBN, Username, Title, Authors, Page Count and Rating inside this books tables.

Schema For User Table –

```
id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), index=True, unique=True)
    password_hash = db.Column(db.String(128))
```

We are storing Username, and Password in hashed format. In order to generate password_hash, werkzeug library was used.

Then I create different routes for each page.

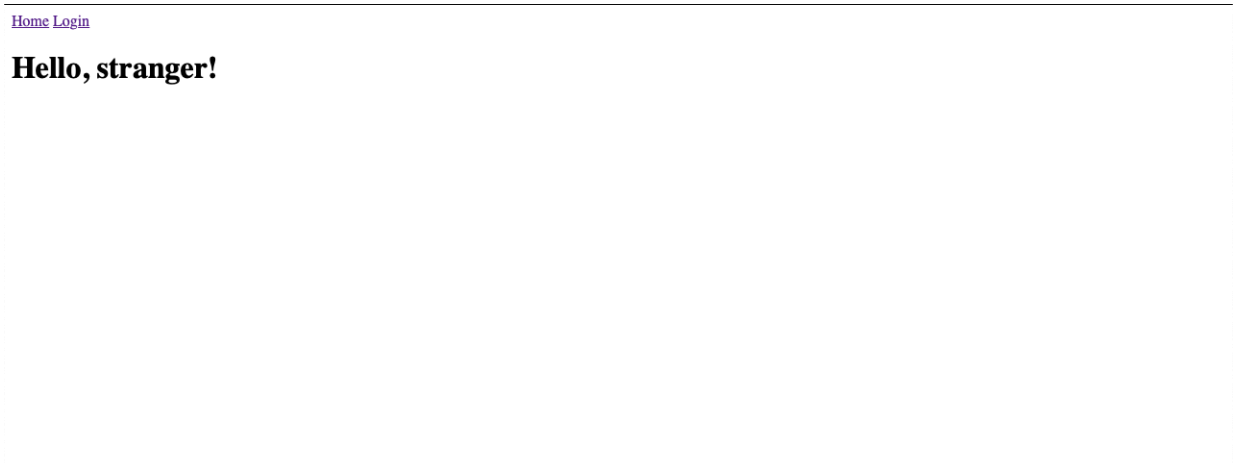
1. Home Page – In this page, if the user is not already logged in then it'll show Login option to login. No book list will be visible. But if the user is already logged in, then it'll fetch the records corresponding to that user and display it in a HTML table.
2. Login Page – It'll display two textboxes for username and password. Username must be unique otherwise a message will be displayed that the user is already existing. If user want to create a new account then Click to register! Can be used. Once the user logged it, app will redirect to home page where the list of books will be shown.
3. Home Page – In the Home Page, there is a search box which takes an ISBN number for a book and returns the matching record. After clicking on submit button, that record gets updated in the books tables shown on the screen.
4. There is a delete button which deletes the record from the database. In the database it searches based on the given username and the ISBN number to find the book.
5. On top there is a logout button, which logs out the user from the application and the session ends.

For the HTML pages, I used Jinja2 templates where I passed the values from flask backend and displayed on the UI. From the UI, I used html forms to pass the value to flask routes.
For each route, I put @login_required decorator to make sure that only authorised user can access the application and make changes to DB.

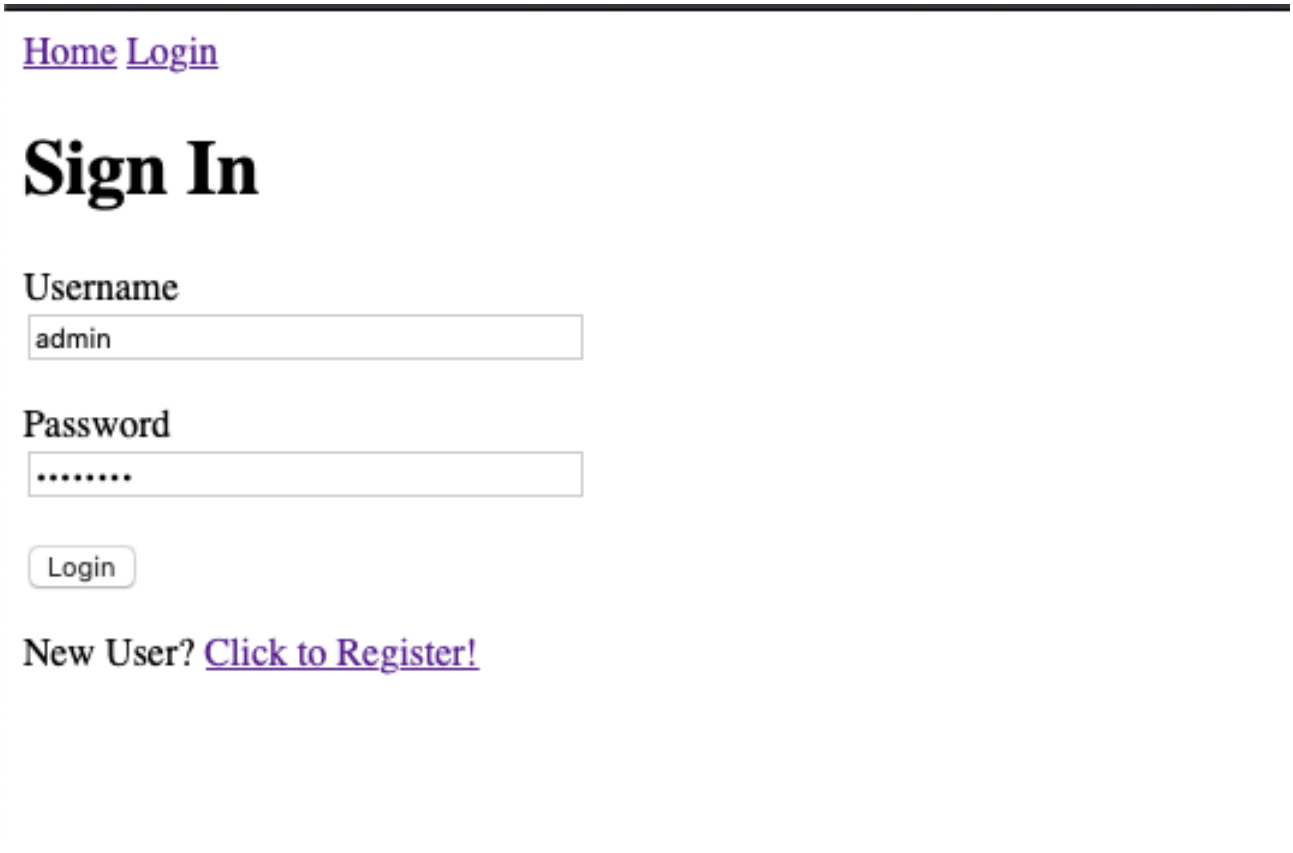
In the book_api.py file, I've put a method to fetch the Google Books API responses and extract the required details to be shown in the HTML table.

- **ScreenShots –**

Page – 1



Page – 2



Page – 3

Home Login

Register

Username

Password

Register

Page – 4

Home Logout

Hi, admin!

ISBN:
9781449372620
Submit

Title	Authors	Page Count	Rating	Action
Flask Web Development	Miguel Grinberg	237	NOT_MATURE	Delete
Learning Java	Patrick Niemeyer, Daniel Leuck	1010	NOT_MATURE	Delete