### General Instructions - DELETE BEFORE CREATING FINAL REPORT

* The report should be not more than 2 pages in length - only brief descriptions are expected. Use standard A4 page size and 10pt font.
* This report, as well as the video are **mandatory** submissions. Your final project will not receive a grade unless you have submitted both.
* The video accompanying the project report should be uploaded under your onlinedegree Google account, and shared with the instructors. Do NOT upload the video as part of your submission - only the link to the video should be uploaded.
* If you use this file as the template - remove all <<tagged>> entries. These are present only as guidelines, and SHOULD NOT BE in the final submission.
* Final submission should have one ZIP file containing the following:
  + This report in PDF form
  + Your code folder with all necessary files needed to run and test. The code should also have a README file that explains how to run the code.
  + YAML file used for API definition

### Author

Ranjeet Sharma

21f2001119

21f2001119@ds.study.iitm.ac.in

Student at IIT-Madras pursuing Bachelor's degree in Data Science and Application, with a fervent interest in development, and programming,

### Description

This project is about an Online Library Management System. There will be one admin and many users. Users can sign-up/register and can start reading books or issue e-books.

Admin can perform CRUD Operations on Section and Books and handle incoming book-requests.

### Technologies used

* ***Flask***: for application code, to handle user requests, manage routing, and creating web pages.
* ***Flask-SQLAlchemy***: for interaction with database.
* ***Flask-Bcrypt***: for hashing password.
* ***Jinja2***: templating engine to generate dynamic HTML content. It allows me to combine python code with HTML templates.
* ***Bootstrap***: for quick css styling and aesthetics.

### DB Schema Design

* **Book Table:** Stores details of the book, having columns as ***isbn***, ***name***, ***content***, ***author*** ***name***, ***section-id***, ***date added***, ***language***, and ***rating***. Primary Key being ***ID*** that stores *unique* id for each book. It also has a column name **enrollments** that is in relationship with the enrollments table.
* **User Table:** Stores details of each user, having columns as **ID**, ***name***, ***email***, and ***password.*** It also has a column name **enrollments** that is in relationship with the enrollments table making a many-to-many relationship.
* **Enrollments Table**: ***Book\_id*** and ***user\_id*** columns are **Foreign Key** to Book and User tables respectively, also these two making a **unique constraint** such that no duplicate records are there.

***Issue\_date*** and ***return\_date*** columns keep track of book issuing and returning date.

**user** and **book** makes many-to-many relationships with **User** and **Book** tables respectively.

* **Sections Table:** Represents different sections in the library. Each section has a unique ID, name, creation date, and description.
* **Book\_req Table:** Tracks requests made by users to issue books.Includes user name and id, book name and id, no. of days requesting for, issue date, and return date.
* RestTablesnamely ***Feedback, Rating, Status, Messages***, stores feedbacks, ratings for each book received. Status to keep track of every user’s completed Book and finally **Messages** to store appropriate notification for user on approval or rejection of requested books.

### Architecture and Features

**app.py** file contains the main code to run the flask application. It has Flask and Flask-sqlalchemy object initialised, it also contains all the necessary imports from controllers **user.py**, **admin.py**, **books.py**, **api,** etc

**controllers** folder contains all the routing done for this project.

**models.py** contains schema for database design using **Flask-SQLALChemy.** It contains

classes that represent tables in the database, including columns and relationships between tables.

**static and templates** folder contains **global.css** along with few images and all the html files are kept in the templates folder.

* **CRUD** Operations for **Sections** and **Books**:
* python files **section.py** and **admin.py** contain routes specifically for admin only and only logged in admin can perform CRUD operations on sections and books.
* For **Securing** ***admin-specific*** routes:
* @app.before\_request() decorator has been used in order to validate each request before executing and only logged in admin in the session can access those route.
* Routes for admins to view all books, sections, book requests, and all registered users.
* Dashboard route to access analytics and insights.
* Routes to accept/deny book requests and revoke book access for users.
* Routes for **User**:
* User actions include requesting for books, downloading, viewing issued books, returning, and managing currently issued books and user profile.
* Message Box feature is provided such that users will be notified about whether the request for a particular book was approved or not.
* **Search Functionality**
* Both admin and regular users can search based on book name, author, sections.
* **Bar Charts** also visible on the admin's dashboard for better track of books and sections.

### API Design

* **Book** Management API with ENDPOINTS **GET**:/api/book/<book\_name>, **POST**:/api/book, **DELETE**:/api/book/<book\_name> and **PUT**:/api/book/<int:book\_id>.

Similarly for **Section** Management API also been implemented, more details in yaml file.

### Video

<https://drive.google.com/file/d/1ssPTRdVz-XE7gYkFoi3Z3nJXKW55J46G/view?usp=sharing>