Library Management System V2

Problem definition

Modern Application Development - II

Last Updated: 24-01-07

Frameworks to be used

- Flask for API
- VueJS for UI
- VueJS Advanced with CLI (only if required, not necessary)
- Jinja2 templates if required
    - Not to be used for UI
- Bootstrap, if required (No other CSS framework is allowed)
- SQLite for database (No other database is allowed)
- Redis for caching
- Redis and Celery for batch jobs
- It should be possible to run all the demos on the student's computer, which should either be a Linux based system or should be able to simulate the same. You can use WSL for Windows OS.

Library Management System

- It is a multi-user app (one required librarian and other general users/students)
- Used for issuing e-books to users
- User can request, read, return e-books
- Librarian can add new sections/e-books, issue/revoke access for a book.
- Each Section may have
  - ID
  - Name
  - Date created
  - Description
- Each book will have
  - ID
  - Name
  - Content
  - Author(s)
  - Date issued
  - Return date.
- Every section can have a number of books
- System will automatically show recently added sections/books or based on certain rating

Terminology

- Sections
- User, Librarian, Admin
- e-Book, content
- Policies*

Similar Products in the Market:

1. Amazon - e-books
    - Web, IOS and Android
2. Book Depository
    - Web, IOS and Android

- These are meant for exploring the idea and inspiration
- Don’t copy, get inspired

Refer to the wireframe given below;
Library Management

Core Functionality

- This will be graded
- Base requirements:
  - Librarian Dashboard
  - General User signup and login (using RBAC)
  - Mandatory Librarian Login (using RBAC)
  - General User Profile
  - Section Management
  - e-book Management
  - Search functionality for sections/e-books
- Backend Jobs
  - Export Jobs
  - Reporting Jobs
  - Alert Jobs
- Backend Performance

Core - Librarian and User Login

- User and admin should be authenticated using username or email and password
- Use Flask Security or JWT based Token Based Authentication only
- Registration form for General User is required.
- New librarian cannot be created.
- Suitable model for general user (model that stores all the type of users and differentiates them correctly based on their role)
- The application should have only one librarian.

Core - General User functionalities

- Login/Register
- View all the existing Sections/e-books
- Request/Return Books (content)
- A user can request for a maximum of 5 e-books
- A user can access a book for a specific period of time (say N hours/days/weeks).
    - For e.g. if N = 7 days, user can return a book before 7 days. If he/she fails to do so, the access for that will be automatically revoked after 7 days.
- User can give feedback for an e-book

Core - Librarian functionalities

- Issue one or multiple e-book(s) to a user
- Revoke access for one or multiple e-book(s) from a user
  - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
- Edit an existing section/e-book
  - Change content, author name, no. of pages/volume etc.
- Remove an existing section/e-book
- Assign a book to a particular section
- A librarian can monitor current status of each e-book and the user it is issued to
- Available e-books in the Library

Core - Search for sections/e-books

- Ability to search for a particular section.
- Ability to search e-books based on section, author etc.

Core - Daily Reminder Jobs

- Scheduled Job - Daily reminders on Google Chat using webhook or SMS or Email
  - In the evening, every day (you can choose time of your choice)
  - Check if the user has visited the app.
  - If no, then send the alert asking them to visit

Core - Scheduled Job - Monthly Activity Report

- Scheduled Job - Monthly Activity Report
  - Come Up with a monthly progress report in HTML or PDF (email)
  - The monthly activity report is meant for a user
  - The activity report can consist of sections & e-books issued, ratings received on sections/e-books etc.
  - On the first day of the month
    - Start a job
    - Create a report
    - Send it as email

Recommended (graded)

- Download e-books as PDF for a price
- APIs for interaction with sections and books
  - CRUD on e-books
  - Additional APIs for getting the creating graphs for librarian dashboard
- Validation
  - All form inputs fields - text, numbers, dates etc. with suitable messages
  - Backend validation before storing / selecting from database

Optional

- Styling and Aesthetics
- Proper login system
- Subscriptions or paid versions of the app, become author etc
- Ability of app to read books for a user (text-to-speech)

Evaluation

- Report (not more than 2 pages) describing models and overall system design
  - Include as PDF inside submission folder
- All code to be submitted on portal
- A brief (2-3 minute) video explaining how you approached the problem, what you have implemented, and any extra features
  - This will be viewed during or before the viva, so should be a clear explanation of your work
- Viva: after the video explanation, you are required to give a demo of your work, and answer any questions
  - This includes making changes as requested and running the code for a live demo
  - Other questions that may be unrelated to the project itself but are relevant for the course

Instructions

- This is a live document and will be updated with more details and FAQs (possibly including suggested wireframes, but not specific implementation details) as we proceed.
- We will freeze the problem statement on or before 8th January, beyond which any modifications to the statement will be communicated via proper announcements.
- The project has to be submitted as a single zip file.