ADV WEB APPLICATION DEVELOPMENT
CS-621
FINAL PROJECT REPORT

FULL NAME :- (BLAZERID)

VENKATARAMA SAI TEJA DASARATHI :- VDASARAT

LIKITH KUMAR TARALA :- LTARALA

SIDDHANTH ERRAMARAJU :- SERRAMAR

SURYA SATWIK VARMA NADIMPALLI :- SNADIMP3

LibraryApp

Overview and Purpose

LibraryApp is a library management application that serves two types of users:

1. **Librarians** that can manage the library's collection and borrowers.
2. **Users** can borrow books from the library, add to cart and their borrowing history with fines if its overdue.

Here, the librarian can manage the collection of books, track borrowing and returns, and manage user accounts while users can view available books, borrow books, and see their borrowing history.

The purpose of the app is to streamline the management of library resources and make it easier for users to access and borrow books.

Technologies Used:-

Dependencies

Backend Technologies: LibraryApp was built with Python's Django framework, which offers sophisticated web development tools. For data storage, we used SQLite, a lightweight, serverless database system. This combination encourages quick development and smooth integration across the application.

Front End Technologies

To design the user interface, we used HTML and CSS to organise and style the program. HTML is the core markup for organising content, whereas CSS enhances visual design and responsiveness. Rather than building Vue.js, we used Django's built-in template system. This choice shortens call times and streamlines data administration.

Reasons for Choosing Technology

We chose technologies based on the need for performance and scalability. We can improve scalability by exploiting Django's built-in features for database navigation. The use of HTML, CSS, and Django templates results in efficient data handling and speedier load times. Together, these technologies provide a solid basis for LibraryApp.

Installation and Setup

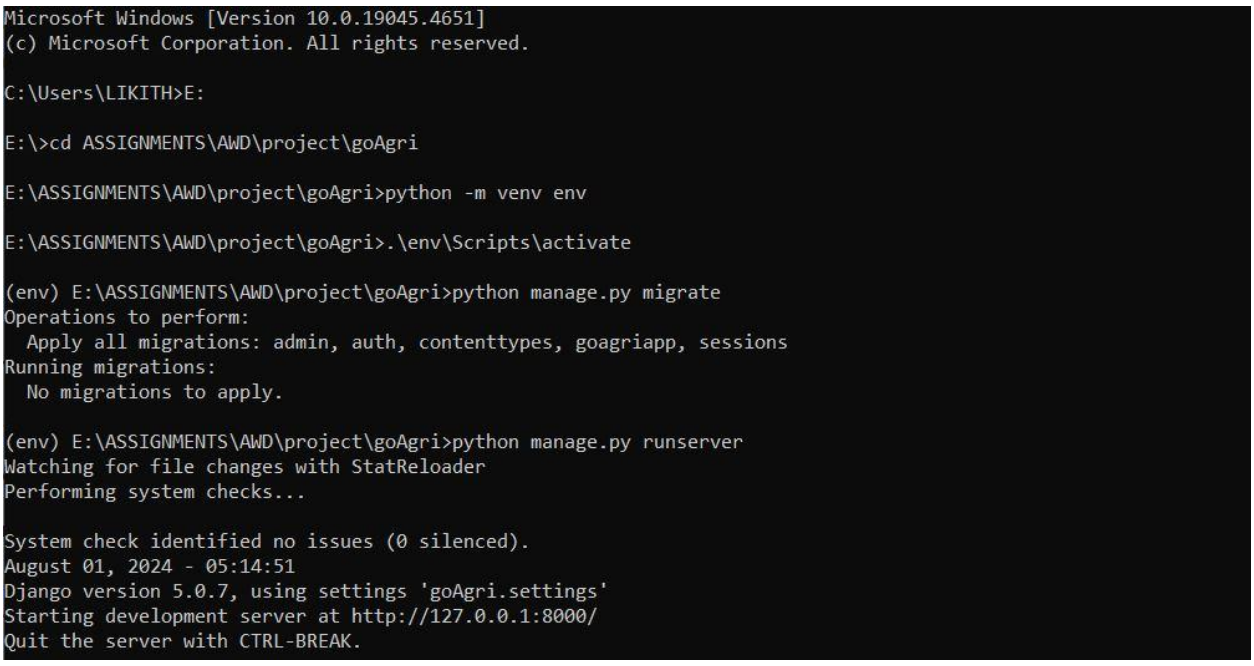
1. Ensure you have Python and Django installed on your machine.
2. Extract the contents of the libraryapp.zip file.
3. Navigate to the project directory where the manage.py file is located.
4. Run the following commands to set up and start the application:
    - python manage.py makemigrations
    - python manage.py migrate
    - python manage.py runserver

This will start the app on the localhost. The URL will be displayed in the terminal.

You can then access the app by opening the URL in a web browser.

You will see the homepage.

*****A detailed instruction of the installation and execution was clearly described in the readme file of the project folder please walkthrough the text file for clear understanding***or else you can follow the below screenshot for a reference!**



Models

We created several models for our app:

CustomUser

This extends the base class of AbstractUser. It defines all the users in our app. It consists of the following additional fields:

- email: the email address of the user.
- first_name: first name of the user.
- last_name: last name of the user.
- phone: phone number of the user.
- address: address of the user.
- is_librarian: a boolean field that defines if the user is a librarian.

Book

These are the books available in the library. It extends the models.Model class and has the following fields:

- title: the title of the book.
- author: the author of the book.
- isbn: the ISBN of the book.
- description: a brief description of the book.
- quantity: the number of copies available in the library.
- quantity_borrowed: the number of copies currently borrowed.

Borrow

This model tracks the borrowing details. It includes:

- user: a reference to the CustomUser who borrowed the book.
- book: a reference to the Book being borrowed.
- borrow_date: the date when the book was borrowed.
- due_date: the due date for returning the book.
- return_date: the date when the book was returned (if applicable).

Endpoints

We have the following endpoints in our app:

- python Copy code
- path('login/', views.login_view, name='login'),
- path('sign-up/', views.sign_up_view, name='sign_up'),

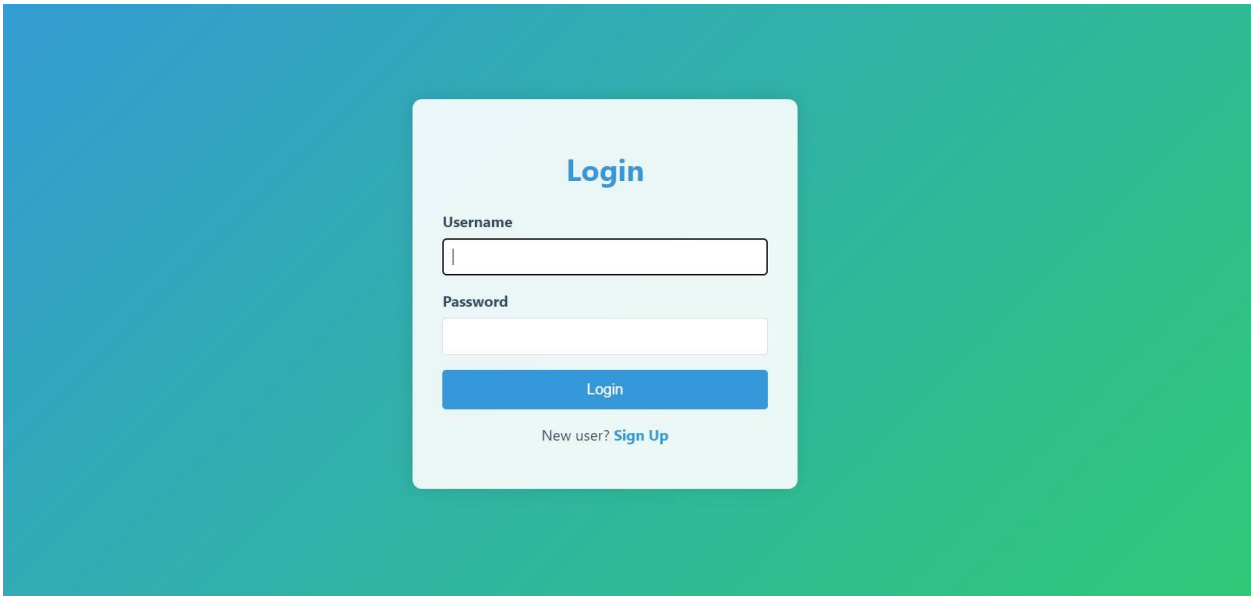
- path('logout/', views.logout_view, name='logout'),
- path('home/', views.home_view, name='home'),
- path('borrow/<int:book_id>/', views.borrow_book, name='borrow_book'),
- path('return/<int:borrow_id>/', views.return_book, name='return_book'),
- path('books/', views.books_view, name='books'),
- path('book/<int:pk>/', views.book_detail, name='book_detail'),
- path('my-borrows/', views.my_borrows_view, name='my_borrows'),
- path('', views.home_view, name='home'),

Views

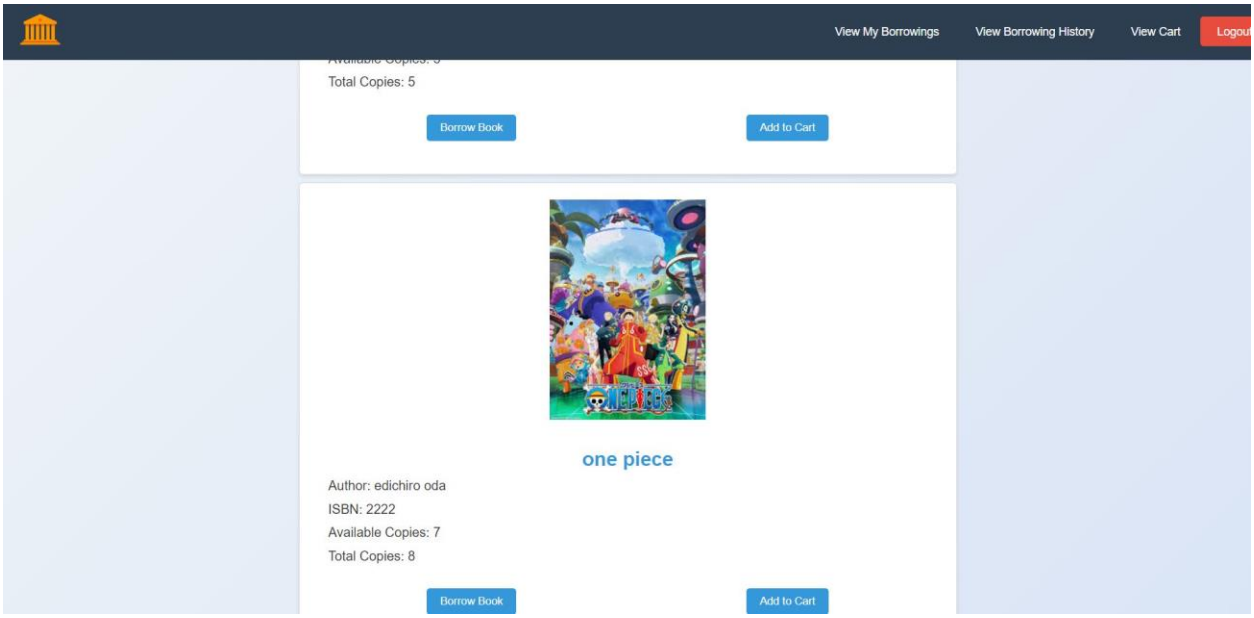
We have created different views for all the endpoints defined above which perform the required actions when the user is using our app.

- The process of signing up, logging in, and logging out via authentication.
- Views for users to borrow and return books.
- Librarians can view the list of all books, manage borrowing records, and perform other administrative tasks.
- Users can view available books, their borrowing history, and details of each book.

Results:-



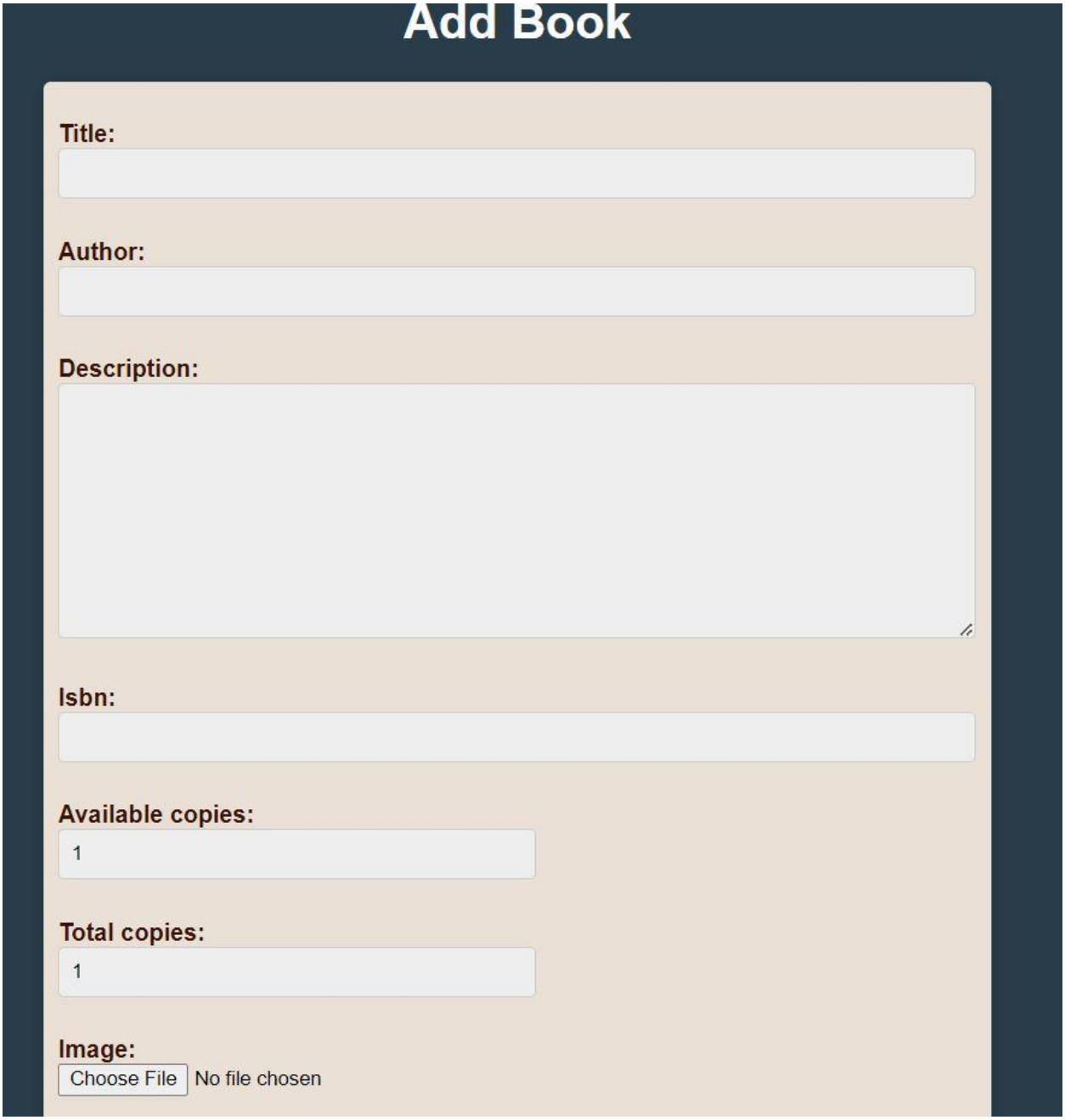
login page for the app



Home page where user can view the books available in library, check his borrowings,and view his cart containing what items he has placed all options are implemented on the menu bar.

Sign in page for the new users that contain fields that are required to create a user or librarian account based on the Boolean field (is librarian)

add book feature available for the librarian containing the the details of the book including the cover image of the book was implemented.



Borrowing history of all the users that have borrowed and returned the book their due dates and fine if overdue only viewable by the librarian.

Discussion and Future work:-

1. Implement sophisticated search feature to filter books based on category and availability.
2. Create a separate mobile app for iOS and Android that provides a native user experience.
3. Create a machine learning-based recommendation system to provide personalised book recommendations.
4. Consider moving to PostgreSQL for better scalability as the user base expands.
5. Create an analytics dashboard to help librarians track borrowing patterns and user interaction.

Conclusion:-

The **LibraryApp** provides a comprehensive solution for **managing** library resources and **borrowing** processes. It simplifies the task of librarians and offers a user-friendly interface for users to access and borrow books.