Section 1: Introduction

HootAndTrade

1.1 Purpose

HootAndTrade is a web-based application where users may pool, store, and retrieve books in our database. These activities will be facilitated through our search, shopping cart, and comment/message writing features. Through these features, our web application strives to make textbooks more accessible to students, and the pursuit of knowledge more accessible.

1.2 Scope

This document offers an overview of the HootAndTrade project's development, along with recommended resources for those who will maintain the website. This document is meant to archive important information about the application and how to best use it.

1.3 Intended Audience

This document is specifically crafted for the students and professors of the Software Engineering (CS 3337)course at California State University of Los Angeles. It aims to provide a detailed understanding of the design and development aspects of the HootAndTrade project produced by Group 4 in CS 3337 Section 1.

1.4 References

- “Software Requirements Specification for HootAndTrade”

1.6 Definitions, Acronyms, and Abbreviations

- California State University: Cal State LA, CSULA
- Software Engineering: CS 3337

Section 2: System Overview

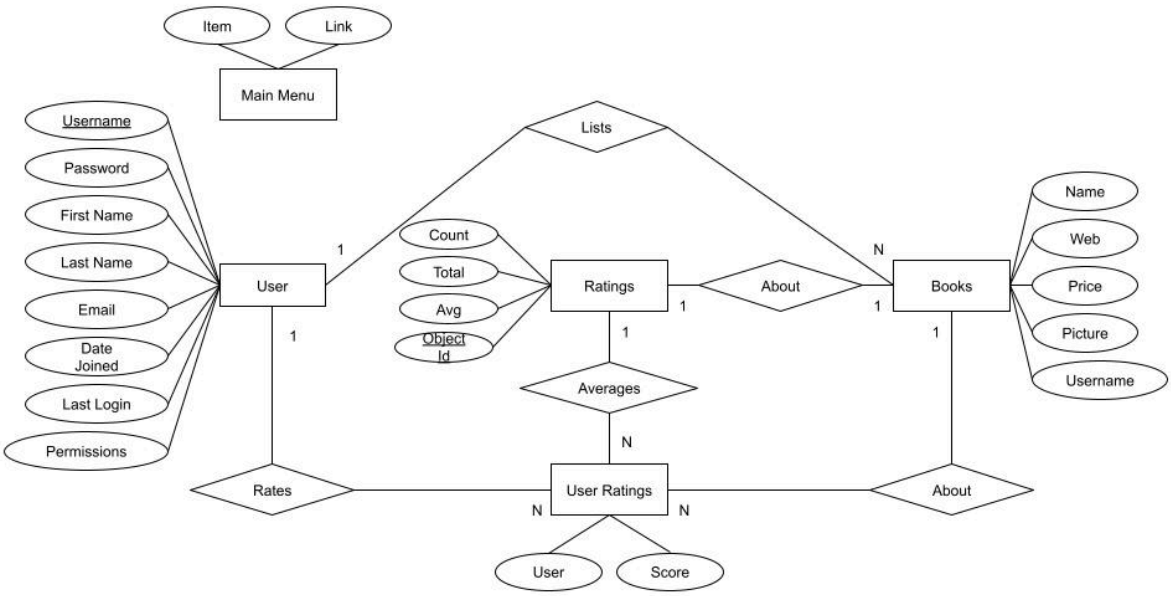
The user interface program, 'HootAndTrade,' aims to enhance the

user-friendliness in buying and selling books. Utilizing the established infrastructure, including the Django framework, 'HootAndTrade' serves as an enterprise software tool compliant with CSULA protocols and industry standards. It offers diverse user interfaces for seamless user experiences, encompassing functions such as user registration, posting, purchasing, reviewing entries, and managing account details.

The advantage of our system is that it allows users to pool, store, and retrieve books in our database, making the source of textbooks available completely scalable to the number of users willing to share their books. This makes textbooks more accessible to students, and the pursuit of knowledge more accessible.

HootAndTrade allows users to:

- Search for books
- Add books to a shopping cart
- Write comments/messages
- Purchase books
- Upload books



Section 3: Design Considerations

3.1 Assumptions and Dependencies

Software used in the project:

- Django (version 3.1.3)
- Python
- HTML5
- PyCharm
- W3.CSS

3.2 General Constraints

To access the services provided by the website, users are required to have an internet connection and create an account. The internet connection enables seamless interaction with the online features, while the account ensures personalized access to functionalities such as posting, purchasing, and managing account details. This account-based system enhances security and tailors the user experience to individual preferences.

3.3 Goals and Guidelines

The primary objectives of the website revolve around fostering the exchange of knowledge through the sharing of books among users. Simultaneously, the platform is committed to ensuring the utmost security and privacy of user information and data. Rigorous measures are in place to prevent the posting of any malicious content, maintaining a safe and trustworthy environment for the seamless exchange of knowledge.

3.4 Development Methods

The website was meticulously developed using a robust combination of HTML and CSS for the front end, ensuring an intuitive and visually appealing user interface. The back-end functionality was implemented using Django, a powerful web framework, which follows the Model-View-Template (MVT) architecture. This architectural approach enhances maintainability and scalability by clearly separating data models, user interfaces, and application logic. The use of Django facilitates rapid development, seamless integration with databases such as SQLite, and robust security features. HTML, CSS, Django, and the MVT architecture collectively contribute to an efficient and responsive platform, offering users a smooth and secure experience while navigating and interacting with the 'HootAndTrade' website.

Section 4: Architectural Strategies

4.1 Use of Particular Products

- HTML: Front-end design
- Python: Back-end scripting
- Django (Model View Template Architecture)

4.2 Reuse of existing software components to implement various parts/features of the system

- Django uses its own templating engine to embed Python code within HTML templates
- Django uses URL patterns to map specific URLs to corresponding views.
- HTML forms can be used to collect user input for users to upload new books.
- Django framework to implement all its components and store
- data in a database.

4.3 Future plans for extending or enhancing the software

We have many more features that we can add to the HootAndTrade site, such as:

1. Filtering:
    a. Enhance the search functionality to allow users to filter books based on genres, authors, or user ratings.
    b. Implement an advanced search with options like publication year, language, etc.
2. Wishlists:
    a. Enable users to create and share wishlists of books they are interested in acquiring.
    b. Notify users when a book on their wishlist becomes available for exchange.
3. Book Recommendations:
    a. Implement a recommendation system based on user's preferences, previous exchanges, or favorite genres.

4.4 User interface paradigms (or system input and output models)

- The home page shall serve as the main template for the whole website, with all other pages maintaining a consistent style.
- The heading, navigation bar, will be present on all pages.
- User input may be received through the search bar, comment/messaging sections, and the postbook page.

4.5 Hardware and/or software interface paradigms

- The user must have a device with a screen or monitor to view the pages, as well as a browser to access them. The user must also have a mouse and keyboard if their device is not touch screen.

4.6 Error detection and recovery

- Implement avid use of exception handling.
- Will provide a help page in the future for people to report errors.

4.7 Memory management Policies

- Django will handle memory management

4.8 External databases and/or data storage management and persistence

- Cloud storage can be implemented using AZURE with an SQL database in the future.

4.9 Generalized approaches to control

- Maintainers can log into the django admin page
- Maintainers can also update the source code.

4.10 Concurrency and synchronization

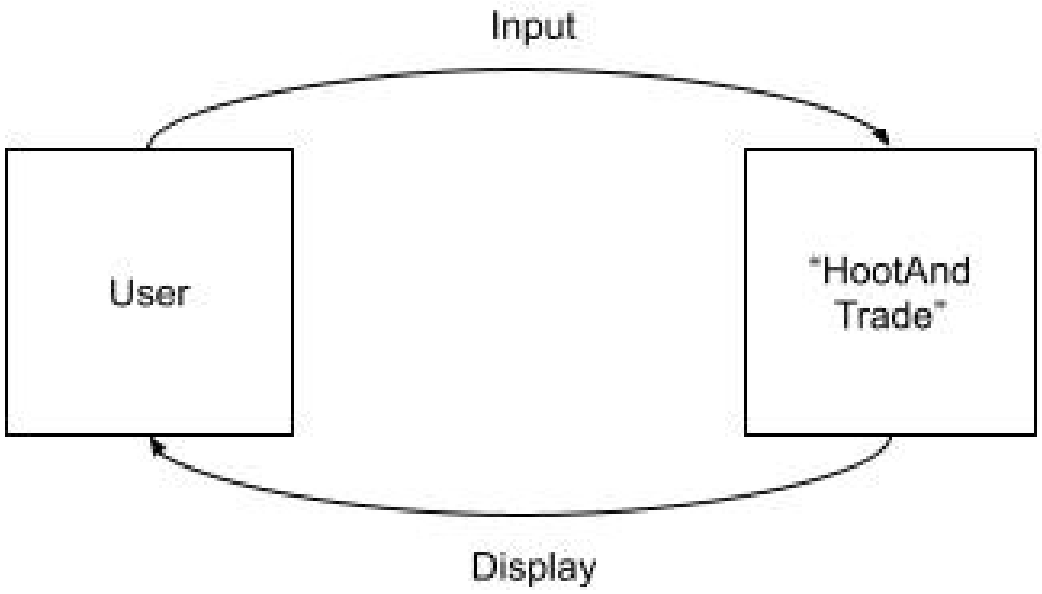
- Concurrency and Synchronization shall be handled by the Django Framework

4.11 Communication mechanisms

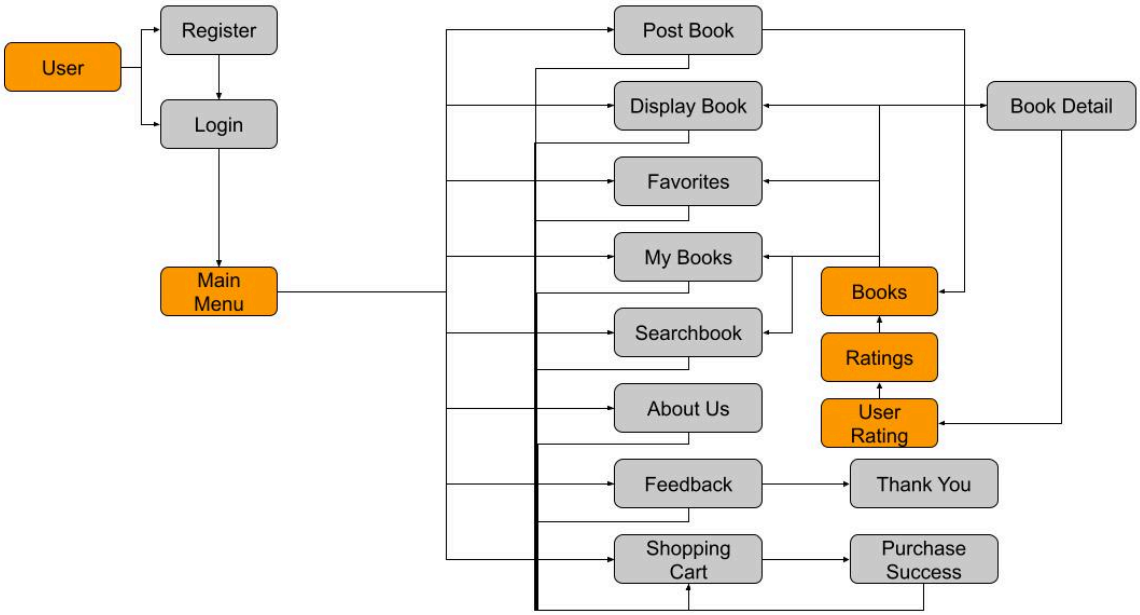
- The user may post comments on books.
- The user may email the owners of the website.

Section 5: System Architecture

5.1 Level 0 DFD



5.2 Entity Relationship



The user initiates interaction by either logging in with existing credentials or creating a new account. Subsequently, entry into the Main Menu is granted, affording access to the website's comprehensive feature set. Navigating to the Display Book page allows users to explore intricate details of a selected book, while the Shopping Cart facilitates book purchase transactions. Additionally, users can engage in communication by leaving messages through the Feedback page, enhancing the interactive and user-driven aspects of the platform.

Section 6: Detailed System Design

6.1 Post Book

- Definition: A page for users to post books for others to download.
- Responsibilities: Receive all details about the book from the user.
- Constraints: The user needs to be logged in to access the page.
- Composition: There are fields for obtaining information such as Title and author.
- Users/Interactions: The user fills out text fields and uploads an image file for the title cover
- Resources: Once the form is filled out, the book will be available in display books
- Processing: The page displays an HTML form that uses the information entered and adds the book to the Django database.
- Interface/Exports:
    - HTML
    - Python
    - Django

6.2 Display Book

- Definition: A page for users to view the books available.

- Responsibilities: Retrieve all the books from our database and display them.
- Constraints: The user must be logged in and there must be books in the database to view.
- Composition: There is a list with many clickable cards containing information on each book.
- Users/Interactions: The user may click on a book to learn more about it.
- Resources: The books displayed depend on the information stored in our Django database.
- Processing: The books are displayed in a list in the order they were uploaded.
- Interface/Exports:
    - HTML
    - Python
    - Django

6.3 Favorites

- Definition: A page where users can view their favorited books.
- Responsibilities: Retrieve the user's purchased books and display only the favorited ones.
- Constraints: The user must be logged in and must have previously purchased and favorited a book.
- Composition: The page displays the books as clickable cards containing information on each book.
- Users/Interactions: The user may click on a book to learn more about it. They can also unfavorite the book.
- Resources: The books displayed depend on the information stored in our Django database, and if they have been favorited.
- Processing: The books are displayed in the order that they were favorited.
- Interface/Exports:
    - HTML
    - Python
    - Django

6.4 My Books

- Definition: The page where users can view the books they have purchased.
- Responsibilities: Display the books that the user has purchased.
- Constraints: The user must be logged in and must have purchased books.
- Composition: The page displays the books as clickable cards containing information on each book.
- Users/Interactions: The user may click on a book to learn more about it. They can also favorite the book.
- Resources: The books displayed depend on the information stored in our Django database, and if they have been purchased.
- Processing: The books are displayed in the order they were purchased.
- Interface/Exports
    - HTML
    - Python

    - Django

6.5 Search Book

- Definition: The feature users can use to search for a particular book in our database.
- Responsibilities: Retrieve whatever book the user searches for.
- Constraints: The user must be logged in and the book the user searches for must exist in our database for it to work properly.
- Composition: There is a text field as a search bar with a button to start the search. Then if the book exists, the page displays the books as clickable cards containing information on each book.
- Users/Interactions: The user can enter their search into a text field and start the search with a button.
- Resources: The books displayed depend on the information stored in our Django database, and if the user has properly searched for it.
- Processing: Our list of books is traversed until something matches the user's search. If something matches then the book will be displayed.
- Interface/Exports
    - HTML
    - Python
    - Django

6.6 About Us

- Definition: The page where users can learn more about the website.
- Responsibilities: Display relevant information.
- Constraints: The user must be logged in and have internet access.
- Composition: Plain text explaining what HootAndTrade is.
- Users/Interactions: The user can only read the information and leave the page when they are done reading.
- Resources: NA
- Processing: The description of HootAndTrade are hard coded and simply displayed.
- Interface/Exports
    - HTML
    - Python
    - Django

6.7 Feedback

- Definition: Allows the user to send an email to the developers and also includes info so they can respond.
- Responsibilities: Allow users to write their feedback on the website and send what they write to the maintainers.
- Constraints: The user must be logged in.
- Composition: There is a text field for users to write their message.
- Users/Interactions: The user can write their feedback in the text field.
- Resources: NA

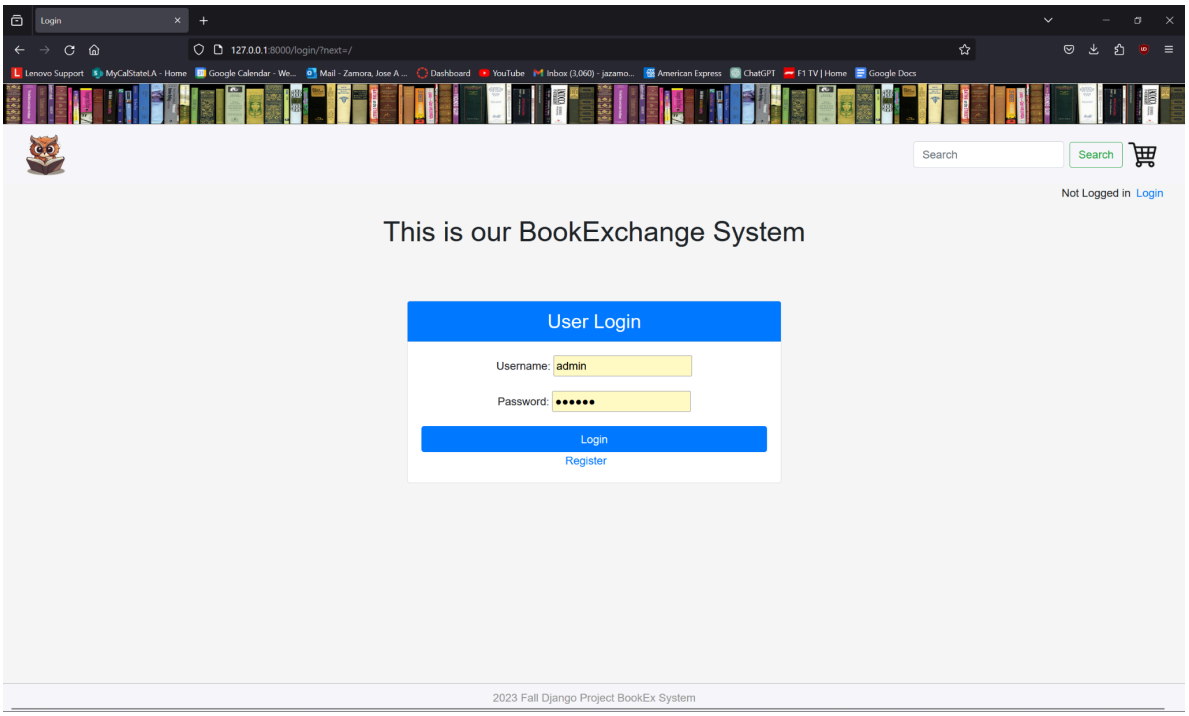
- Processing: The page displays an HTML form that users submit, and their response is sent. Utilized Django built-in functions and Google's SMTP to send the email.
- Interface/Exports
    - Google SMT
    - HTML
    - Python
    - Django

6.8 Shopping Cart

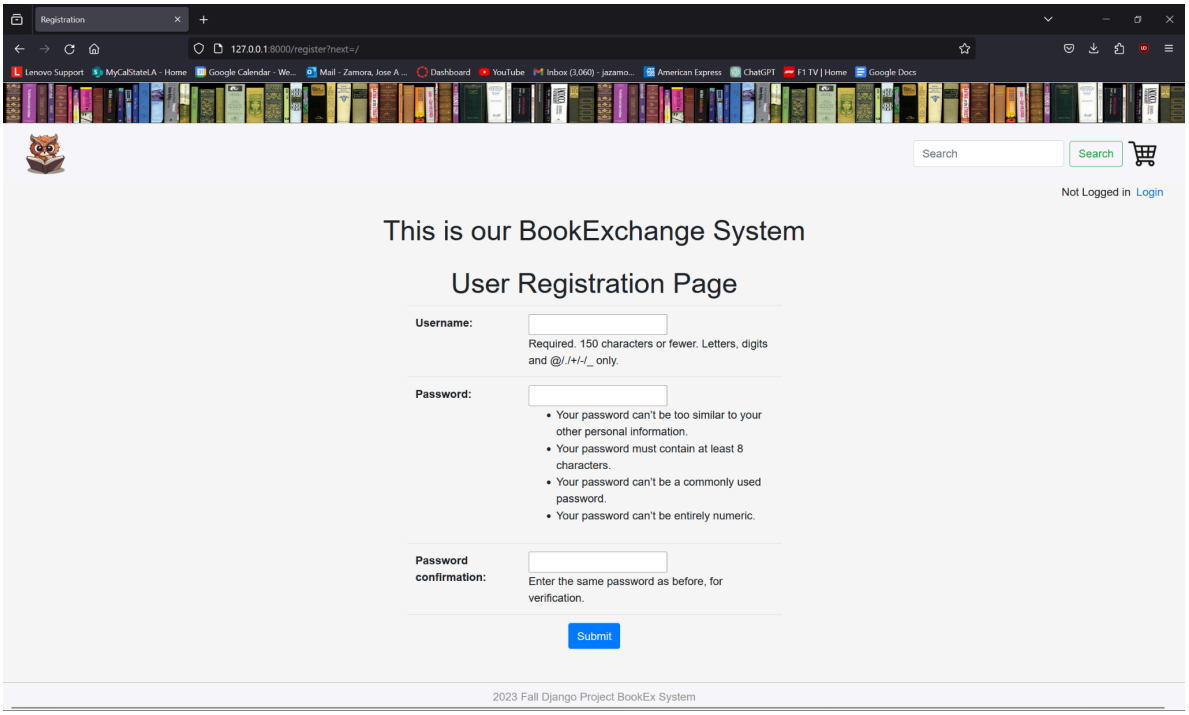
- Definition: The feature where users can specify the books they would like to purchase.
- Responsibilities: Keep track of what books the user would like to purchase.
- Constraints: The user must be logged in and there must be books in the database for the user to save to their shopping cart.
- Composition: Users to check their shopping cart which will display the books in the order that they were saved to the shopping cart.
- Users/Interactions: Users can add and remove books from the shopping cart to purchase all at once when the user is ready.
- Resources: The books displayed depend on the information stored in our Django database.
- Processing: When the user saves a book to the shopping cart, the book is added to a list, and when the user views their shopping cart, all the books in the list are displayed.
- Interface/Exports
    - HTML
    - Python
    - Django

Section 7: Graphical User Interface Design

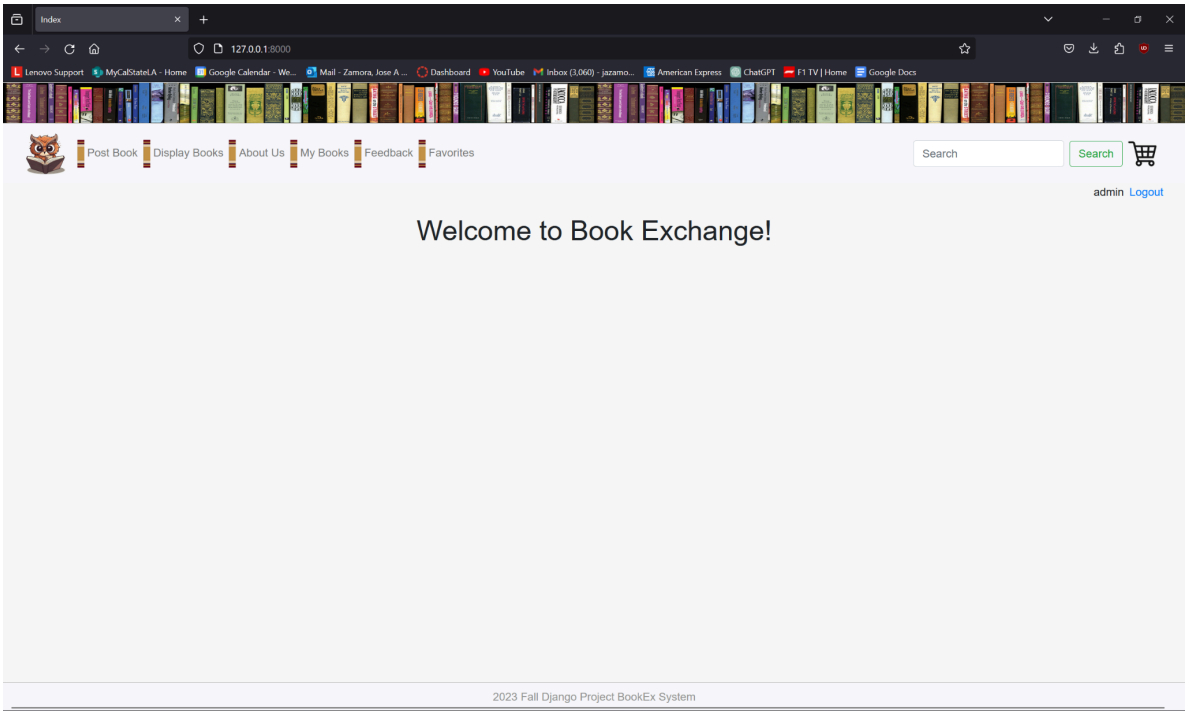
Login



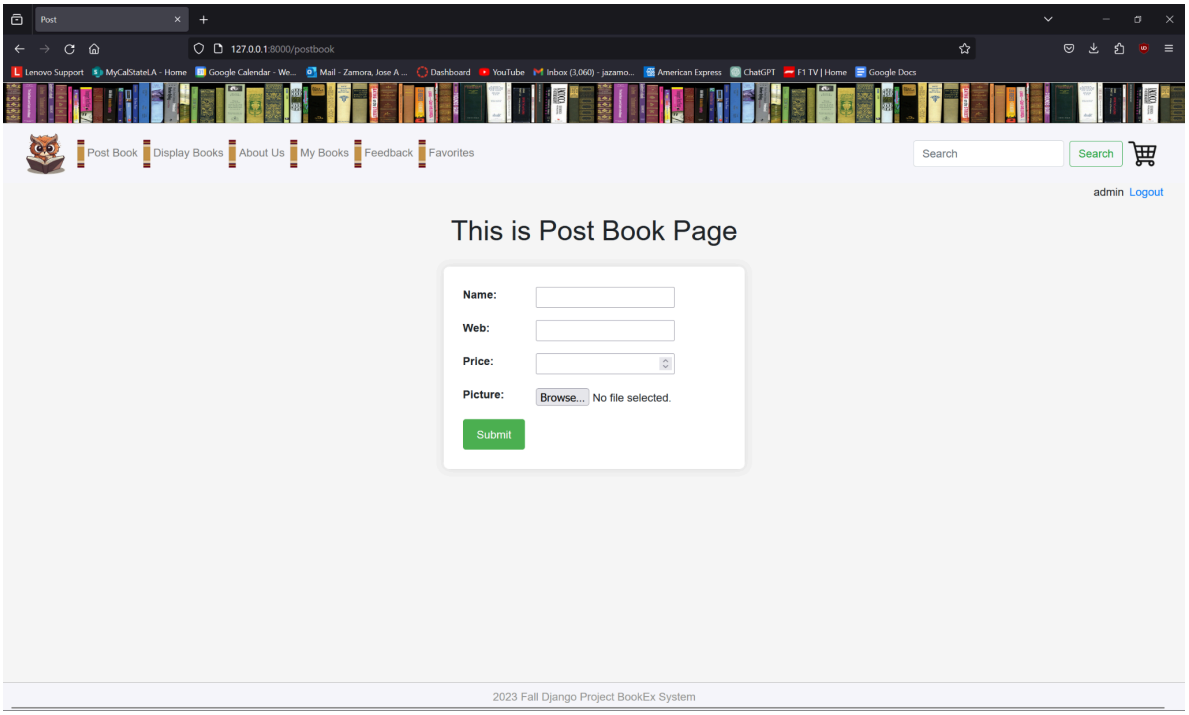
Registration



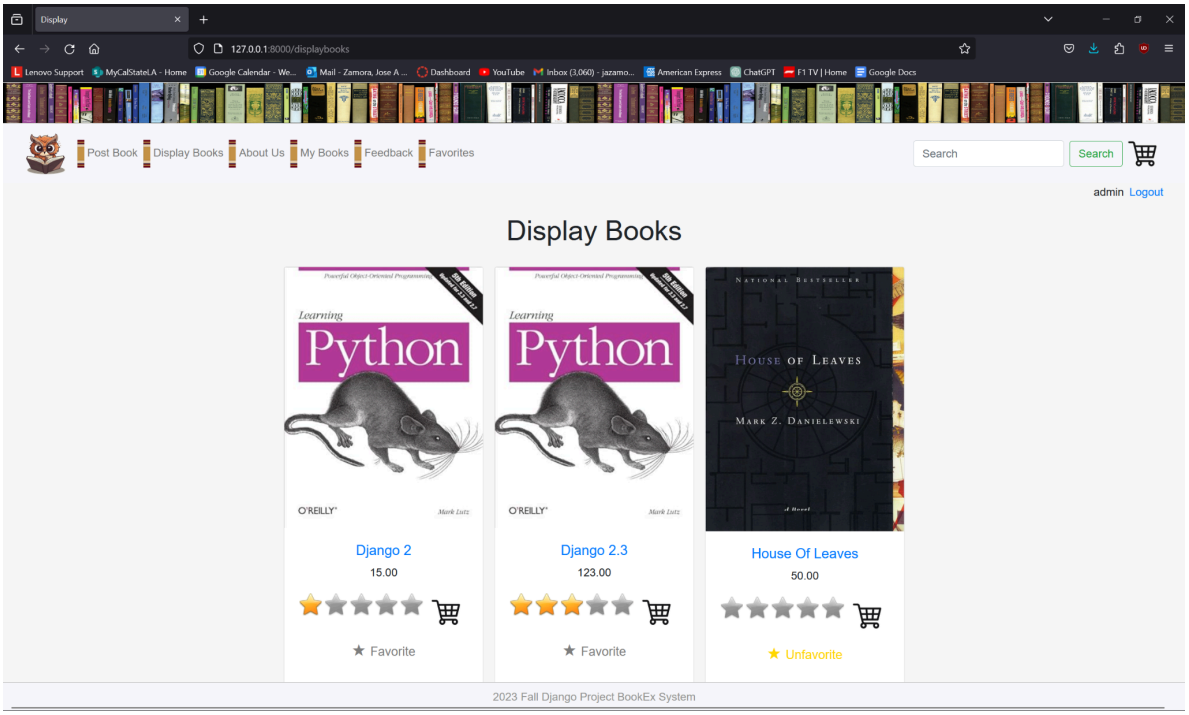
Home Page



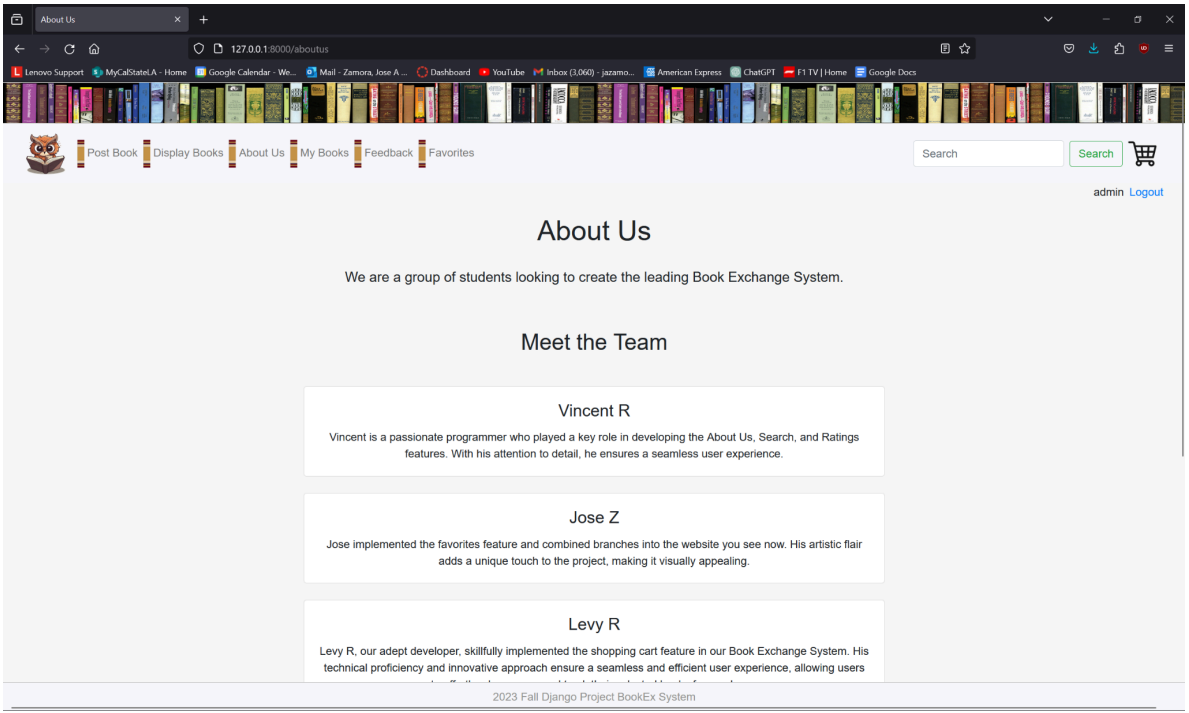
Post Book



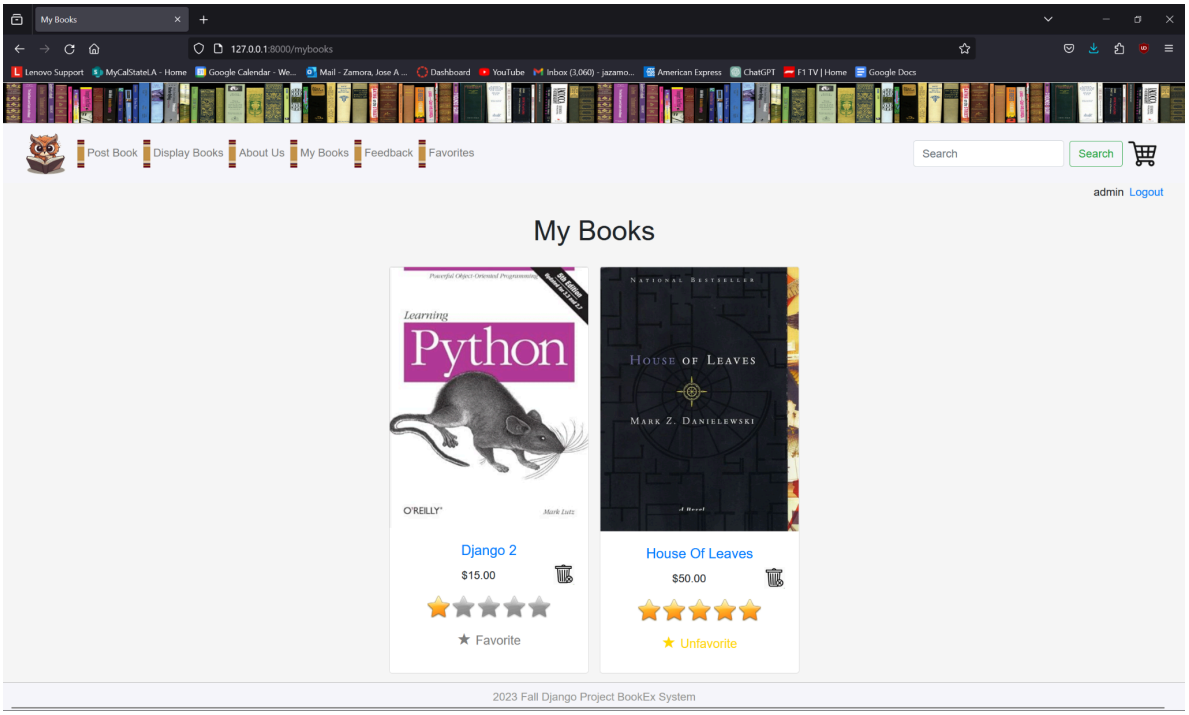
Display Books



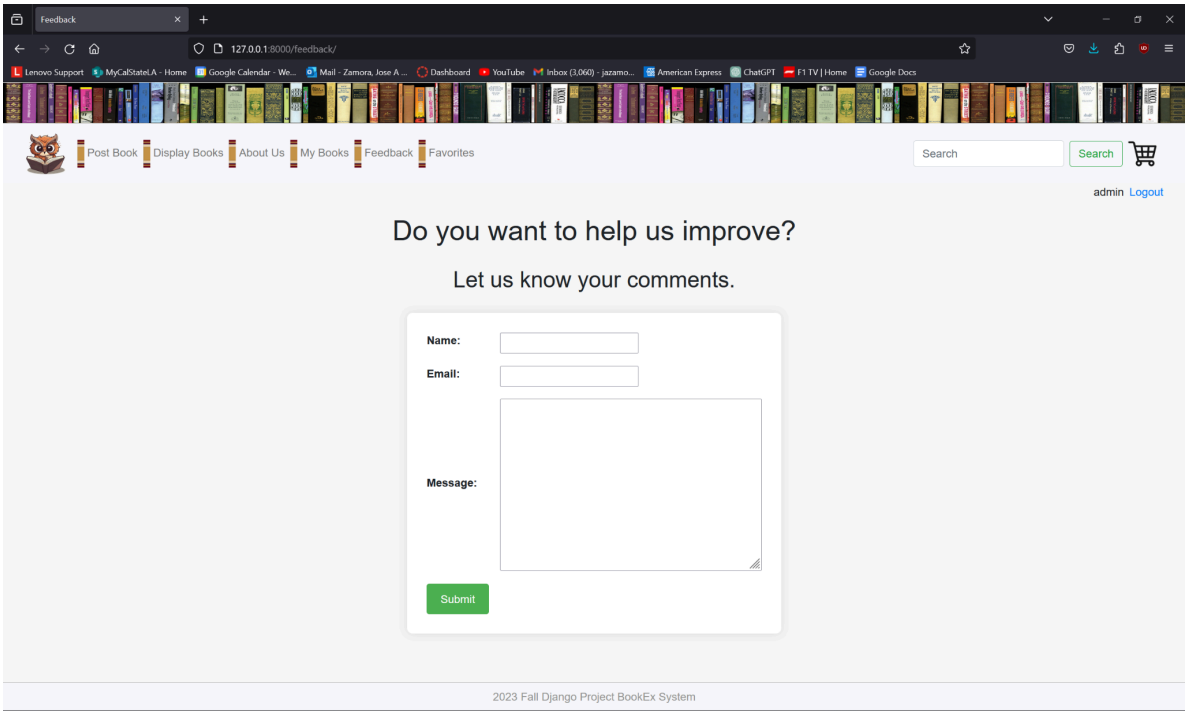
About Us



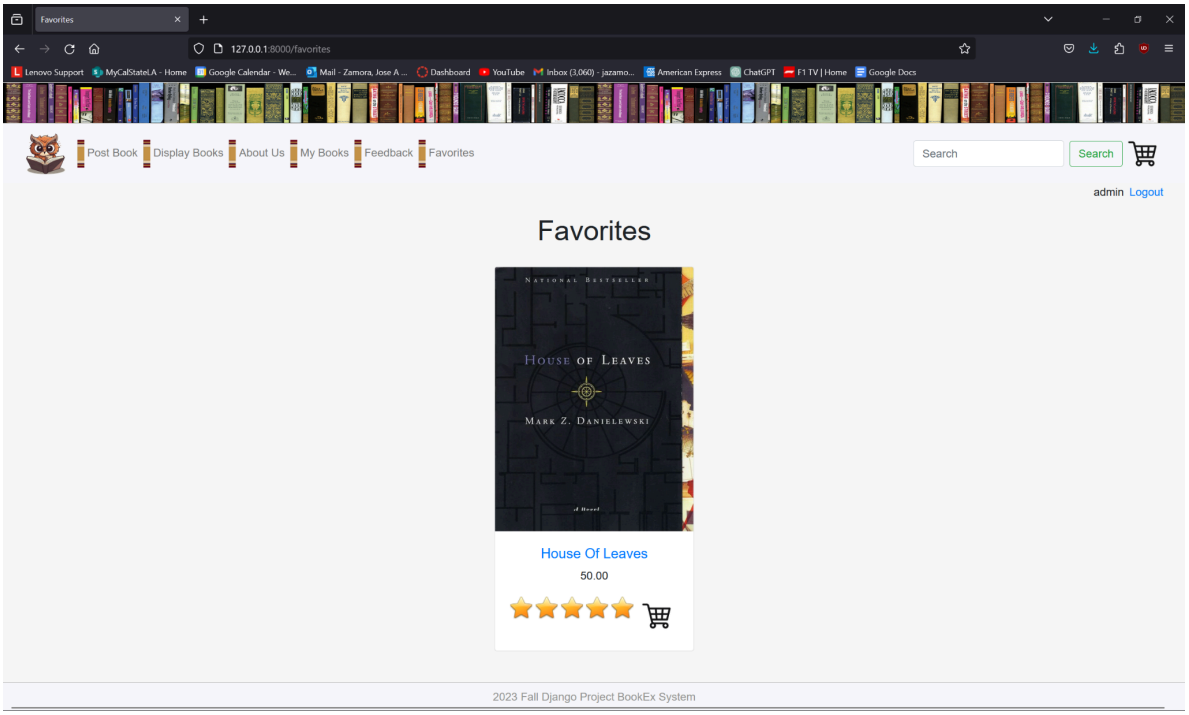
My Books



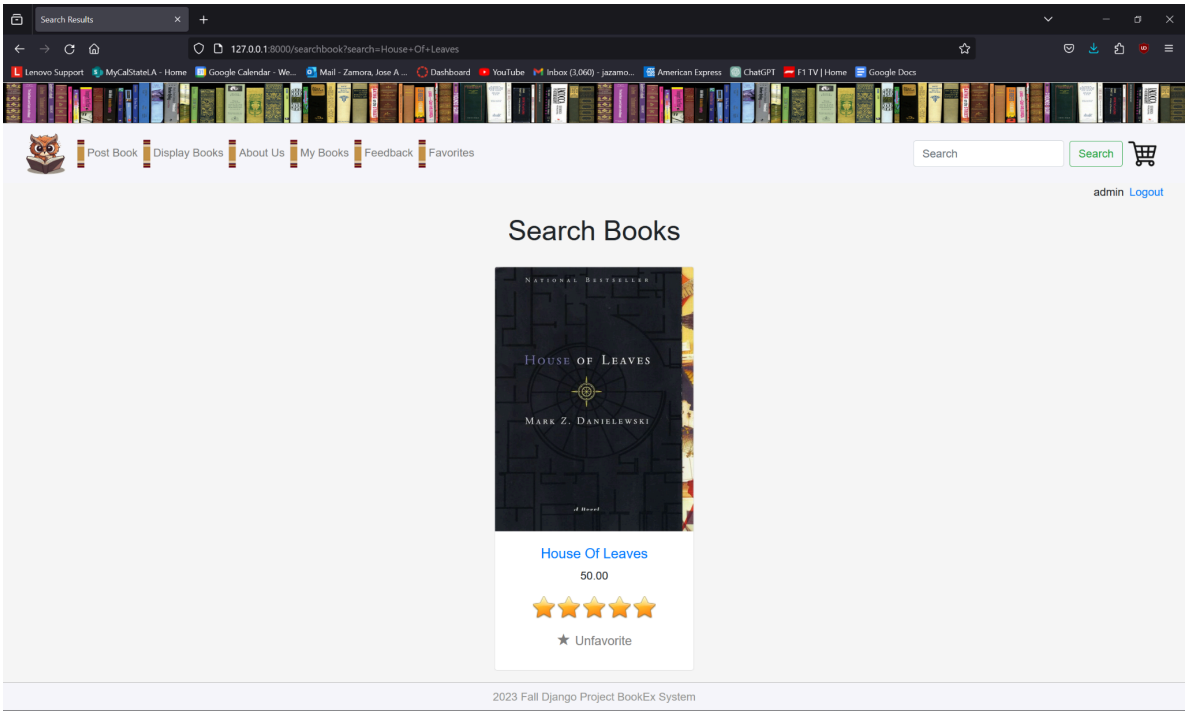
Feedback



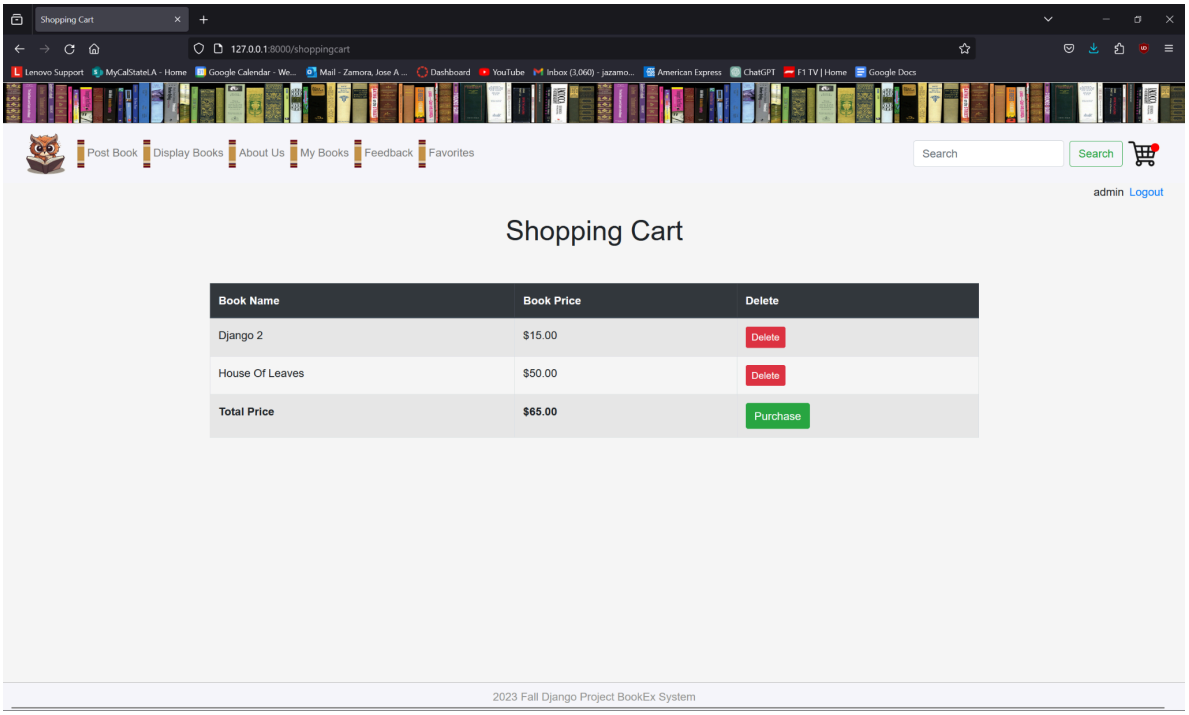
Favorites



Search



Shopping Cart



Section 8: Glossary

- Django: a high-level, open-source web framework for building web applications using the Python programming language.
- MTV Architecture: Model-View-Template is the architectural pattern used by the Django web framework.
- CSS: Cascading Style Sheets
- HTML: Hyper Text Markup Language