****

**CA for B'9IS109**

**Web Development for Information Systems**

**Presented by**

**Lecturer:**

# 

Contents

[Introduction 3](#_Toc185193033)

[What Research we have done before? 3](#_Toc185193034)

[Choice of Framework and Technologies 3](#_Toc185193035)

[Frontend Technologies 3](#_Toc185193036)

[Authentication and Security 5](#_Toc185193037)

[Design Patterns 6](#_Toc185193038)

[UX Design 7](#_Toc185193039)

[Design Philosophy 7](#_Toc185193040)

[Wireframes and Prototypes 7](#_Toc185193041)

[Web Service 9](#_Toc185193042)

[API Design 9](#_Toc185193043)

[5. Security Threats and Measures 13](#_Toc185193044)

[Potential Threats 13](#_Toc185193045)

[Security Measures 13](#_Toc185193046)

[GitHub 14](#_Toc185193047)

[GitHub URL: https://github.com/Vamshireddy2001/BookSeller 14](#_Toc185193048)

[Deployment 14](#_Toc185193049)

[Render URL: https://frontend-zwrc.onrender.com/ 14](#_Toc185193050)

[Conclusion 14](#_Toc185193051)

# Introduction

**Super Seller Book Store** is a web application which is innovative and highly extensive for user who are book enthusiasts. It gives perfect experience for those who love to purchase books because this web application goes beyond the normal functionality like they can get recommendation of books, adding notes and purchase their books based on their genre, plus page count. User can experience simple and easy interface to interact with our website. This website is not limited to this but this gives a user, safe and secure platform to work as all their details kept secure and encrypted. This website is created by taking consideration of security that their account cannot be hacked. For more features and detail about our website we have discussed the features, web services plus security, and deployment.

# What Research we have done before?

We have researched several of websites existing who use to sell their books on their platform but with limited features like having cart, adding favourites plus some sort of security and also their website is very simple and designs of their pages are not much attractive. Due to this we have created a web application which not only working those features but more addons and attractive design that will give user a good experience as they can view books with excitement plus, they can use more features like they can get book recommendation based on their author, genre, and page counts. They can add reminder for future purpose as they can add their favourite book for future purchase as they can get reminded that they must purchase in case they forget to buy.

# Choice of Framework and Technologies

## Frontend Technologies

For frontend design we chose ReactJS due to its modular and component-based structure as developers find efficiency and reusability of code inside ReactJS framework.

1. **React.js:**
   * **Why React:**  
     We chose react due to component-based architecture which benefits developer to create components and can reuse them again. Code becomes easy and more readable and saves time to write code each time. We created features like book listing through Json object and google API, create user dashboard, and recommendation system inside react.
   * **Advantages:**
     + It fastens up the rendering of virtual DOM inside javascript.
     + It is easy to integrate library and tools in code.
2. **Styling and UI Frameworks:**
   * **CSS:** Cascading style sheet has been used to stylize the webpages and create more attractive design. It has many features as in react we can use CSS internally and externally. So, we created our pages using both techniques.

In our React application, we use several key features and plugins to make the website dynamic, efficient, and user-friendly:

**useState**:useState is used to manage state inside a component. This allows us to monitor changes, such as user input or make an element on the page more and less visible.

**useEffect**: It is used for handling side effects in components: when the component loads fetching data, or when the component is removed cleaning up resources.

**localStorage**:It is a mechanism for storing data on the user’s device using the browser features local Storage. It allows you to save user settings, or even caching data so loading times are faster.

React features and plugins help keep things simple by offering a one-stop solution for state management, performance boosting, and a pleasant user experience across the entire application.

**Backend Technologies**

1. **Node.js with Express:**
   * **Why Node.js:**  
     Nodejs has been javascript based framework which integrates some of the libraries like express, mongodb to create server in we use endpoints to communicate and exchange information through API.
   * **Advantages:**
     + Code is easy to write and MVC can be followed up to create modular code.
     + It is easy to create as it is based on JavaScript.
2. **Database:**
   * **MongoDB:**  
     We used MongoDB as a database which has NO SQL in it meaning data can be unstructured, structured, or semi structured as it can store most of the data easily. It follows JSON to create schema for tables.
   * **Key Features:**

It can be scaled up horizontally as it can handled large dataset.

Not required SQL language to create database. It has their own functions which can be used to manipulate database

Some libraries have been added on the Node.js backend of the application to increase the functionality, improve the security, and improve the development process. These key libraries include:

**Express**: This is a powerful library and serves as framework to create web applications and API’s in Nodejs

**Dotenv**: dotenv library handles the environment variables. These secure placeholders for sensitive information include API keys and database credentials. We protect sensitive data by storing this information in a .env file and not expose them directly code.

**Cors (Cross-Origin Resource Sharing):** Cors library helps to communicate securely between frontend and backend when both are hosted on different domains. It enables you to make controlled cross origin requests and let client-side applications like build with react as an example to communicate with the backend without security issues.

**cookie-parser:** It is necessary to manage user sessions and authentication tokens stored in cookies. It parses the cookies stored on browser as without cookie parser, any stored data in cookies cannot be read.

# Authentication and Security

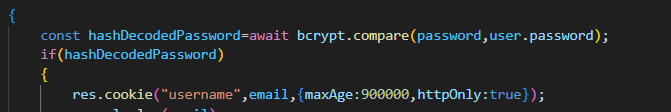
We used some of the security features in our web application to protect sensitive information of user. To protect the user information, we used following approaches below:

1. **JWT (JSON Web Tokens):**

Json web token is a approach to generate a based on user id or any information which can be stores on browser or within application which can be further used by the pages to verify the identification of user whether he is a right user or not

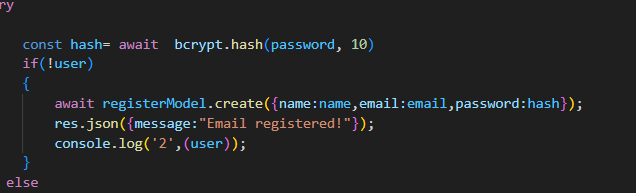


1. **Cookies:**
   * Cookies has been used to store some of the information like user id, token or any user information which can be use within browser and to rotate information within pages like user profile name, or token.



1. **Bcrypt**

Bcrypt has been used to encrypt user password so that any member who is using database cannot manipulate for his/her personal use. Only password can be known by the owner. While login it is used by secret key to check whether encrypted password is correct or not



## Design Patterns

1. **Model (M):**
   * We have used the model folder to create a schema’s for our database like to add reminder, user authentication information, store book collection etc. It stores all the information related to model.
2. **View (V):**
   * Views are something which we show to user or we can call it a frontend and this folder we use to store all web pages.
3. **Controller (C):**
   * Controller is like a middle man between routes and logics integrated in routes. In controller we store all the functions inside routes. This creates proper readability of code and not make code inside same.

With this type of setup, we create a whole backend and in a proper modular approach which makes code more readable and easier to manipulate.

# UX Design

## Design Philosophy

We are focusing on creating a web page which is attractive, simple, and easy to use. Our target is to create a website which can attract more users and make them engage on our website to purchase books and explore features given on website.

1. **Simplicity:** We are focusing on making simple, easy, clean website where user can explore and purchasing books.
2. **Intuitiveness:** Navigation is easy to use, searching and filtering are there to find specific book.
3. **Personalization:** Recommendation system has been created to find a specific book according to the user taste.

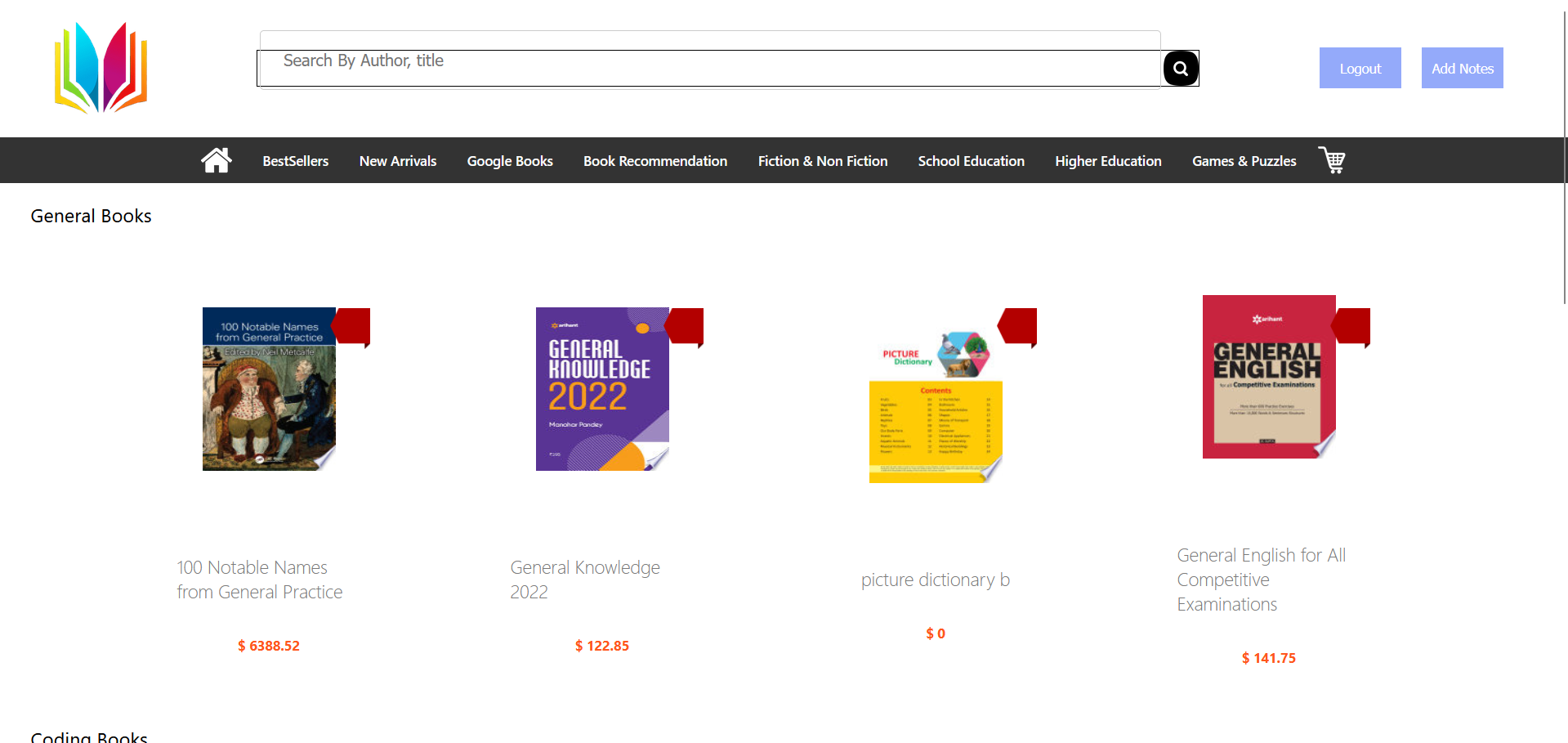
## Wireframes and Prototypes

1. **Homepage:**
   * Displaying of books which are features and popular
   * Proper navigation bar is given to access web pages by just clicking on link also, there is a search option on many links through which user can search for specific book.



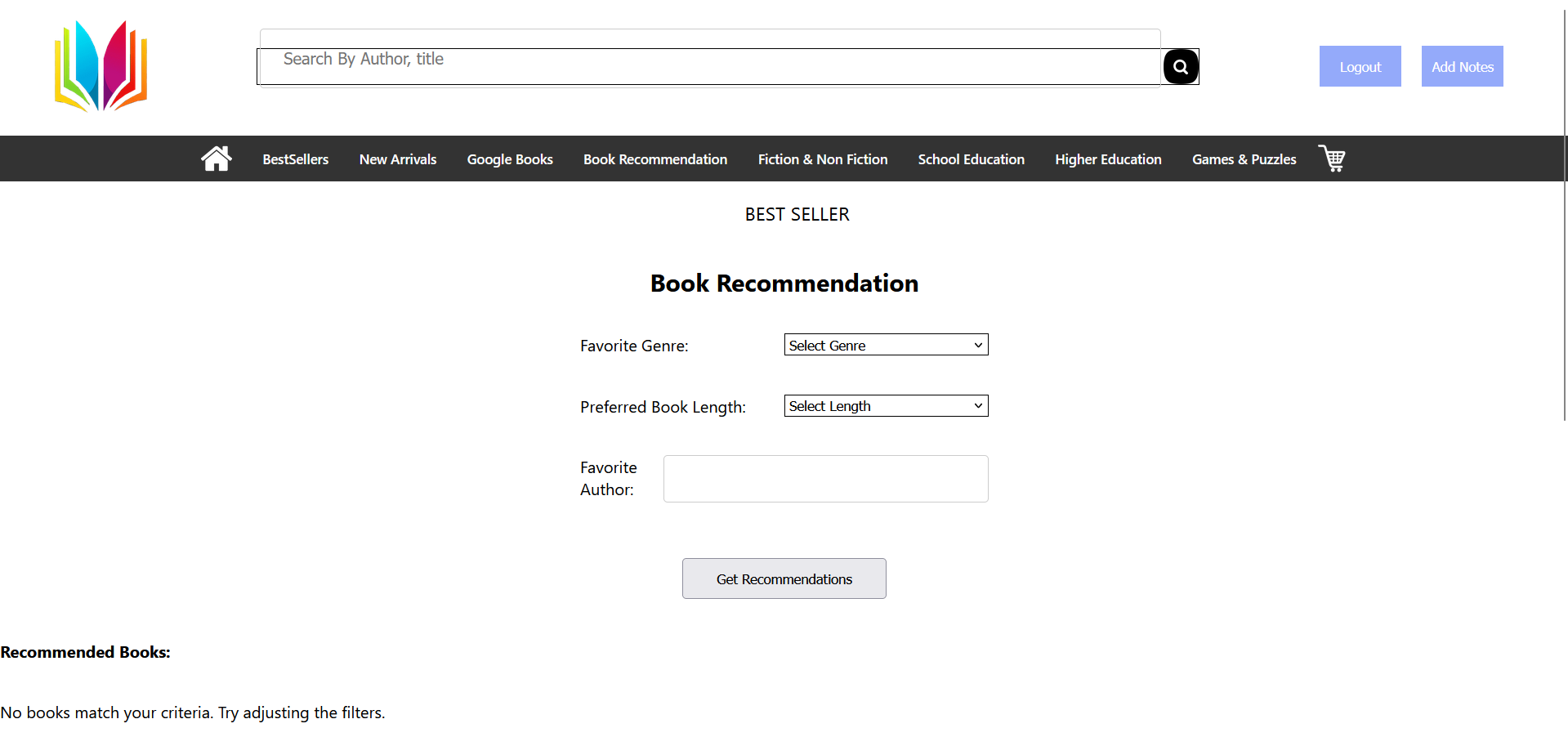
**Home Page**

1. **Book Details Page:**
   * There is detailed information about each book with their name, title, and their price.
   * Add to cart option is given where user can click on and to checkout and purchase



1. **User Dashboard:**

Personal space for user to get recommendation of book on the basis of author, genre, and name



# Web Service

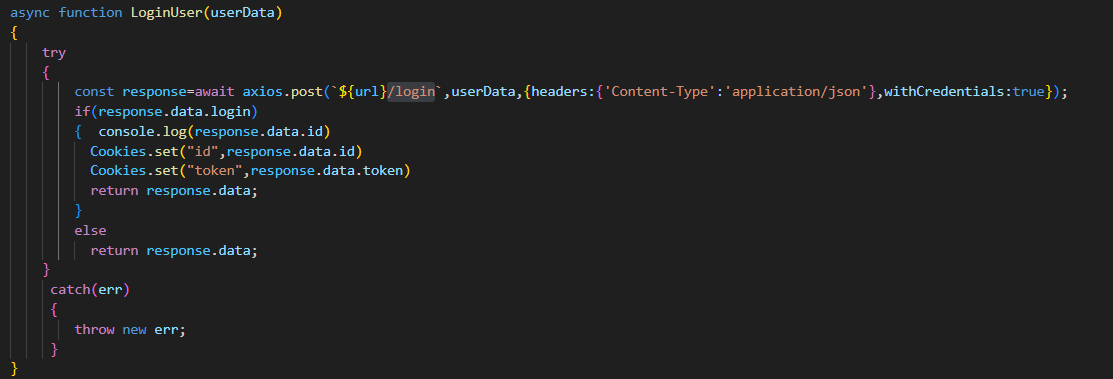
## API Design

For this web application we use restful API system to communicate and exchange information between frontend and a backend. In this we create some of the endpoints in server part so that frontend can fetch and post information using endpoints and display on frontend. We also used external API to fetch book details which is named as Google API which is free and open source.

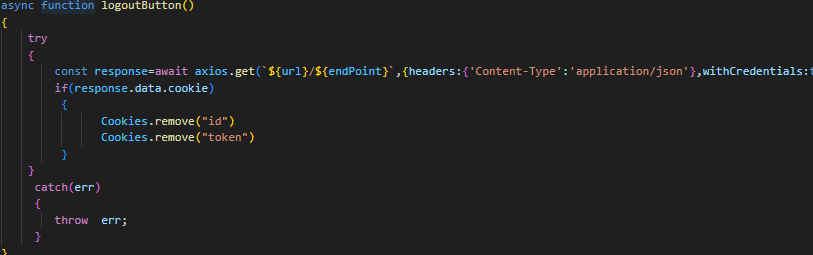
1. **Book Management:**
   * GET: [https://www.googleapis.com/books/v1/volumes?q=${bookName}](https://www.googleapis.com/books/v1/volumes?q=$%7bbookName%7d) – Fetch a list of available books.
   * GET: https://www.googleapis.com/books/v1/volumes?q=subject:${genre}+inauthor:${author} – To fetch a book based on genre and author



1. **User Authentication:**
   * POST /login – Authenticate a user using email and password and return JWTtoken
   * POST /register – Register a new user using details such as name, email, password and confirm password.
   * GET /logout- To expire the token and delete cookies

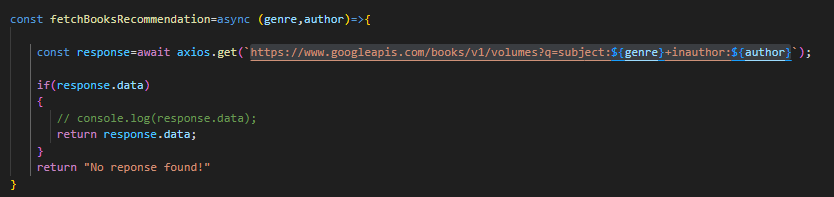


Login



Logout

1. **Recommendations:**
   * GET https://www.googleapis.com/books/v1/volumes?q=subject:${genre}+inauthor:${author} – Provides personalized book recommendations based on user preferences.



1. **Notes:**
   * POST cart/addreminder – This endpoints we use add reminder to a database for specific user ID
   * GET cart/fetchreminder – This endpoint is use to fetch all the added reminders for specific user.



**Database Integration**

* **MongoDB Collections:**
  + **Cart Collection:** In this collection we are storing information for cart in which user is adding items for particular id.

**Books Collection:** This is a collection to create a reminder based on bookname, author, genre, purchase, and date while adding the notes



**Authentication Collection:** This collection is created for users who are creating their account on our web application



# 5. Security Threats and Measures

## Potential Threats

* **Cross-Site Scripting (XSS):** Malicious scripts are used by attackers to steal user data.
* **Data Breaches:** Sensitive user information that they are not authorized to access.
* **SQL Injection**: Netcat can used to change these data in the database through malicious SQL queries.

## Security Measures

* Input Validation and Sensitization’s protect User from the possible XSS attacks all inputs are validated and sanitized.
* **JWT Authentication:** It guarantees the access of routes for pages containing Data like posts, users, pages etc... on which authorized users can view and edit, and only those.
* **Secure Cookies:** HTTP only cookies prevent client-side access to sensitive session information.
* **Password Hashing:** The user passwords are hashed using industry standard technique to protect them from leakage.
* **HTTPS**: All communication between our website and your browser is done over HTTPS to maintain security.

# GitHub

GitHub, we have use to create a repository where we have uploaded all our backend and frontend using git bash commands, and this not only gives benefits to upload content but also track every time we push commits on GitHub and have many more features which we will cover in future. After completing repository then we deploy further on render platform to make our website live.

## GitHub URL: <https://github.com/Vamshireddy2001/BookSeller>

# Deployment

For frontend and backend as the deploy platform, Render was chosen because it is easy to deploy and supports continuous deployments. To deploy, we push changes to GitHub and that will automatically start a new build and push the result to our live site.

## Render URL: <https://frontend-zwrc.onrender.com/>

# Conclusion

Super Seller Book Store web app is an incredible and complete platform that enables you to easy in to the happiness of downloading books without the need of a Account balance. The platform has used modern technologies such as ReactJS (for the frontend) and NodeJS with Express (for the backend) to make it run effectively, be re-used and scalable. The user experience is improved further with key features, including personalized book recommendations, reminders and secure authentication which are beyond traditional e-commerce platforms. Since users’ information is secured using measures such as JWT, bcrypt, and secure cookies, MongoDB for flexible data storage was used. The project is deployed over Render, which ensures continuous deployment on backend and frontend to allow for smooth update and scale. This comprehensive overview tackles the Security, Personalization and Ease of Use needs of book lovers; and hence makes Super Seller Book Store a trusted and engaging platform for book lovers.