lOMoARcPSD|421 417 54

ABSTRACT

The Book based e-commerce website project is a complete platform for readers and book lovers to explore, discover, and connect with their favorite authors and books. It was created using the MERN stack. With a contemporary, responsive design and simple user interface, this online application offers an enjoyable user experience.

The MERN stack, which consists of MongoDB, Express.js, React, and Node.js, is used to build the project. The application's backend is created with Node.js and Express.js, which offer a powerful API for dealing with user requests and managing the database. The database used to hold all the data about users, books, authors, and reviews is called MongoDB.

The application's frontend was developed using React, which offers a fluid and dynamic user experience. The website is responsive and mobile-friendly for use on all platforms, including PCs, tablets, and smartphones.

Users get access to a sizable selection of books and authors to search through and explore, add titles to reading lists, provide reviews and ratings for books, and interact with other users who have similar interests. Authors are able to set up online accounts, advertise their publications, and communicate with readers.

Overall, the Book Website Project, created with the MERN stack, is a great tool for book enthusiasts to find new books, interact with authors, and meet others who share their interests.

CHAPTER – 1

INTRODUCTION

1.1 Introduction:

The Book App, which makes use of the MERN stack, is a cutting-edge web application created to meet the demands of book fans. The Node.js, Express.js, MongoDB, and React.js components of the MERN stack are used to build the project. Because of its adaptability, scalability, and simplicity of usage, this stack is frequently used in web development.

The Book App offers a user interface that is easy to use, contemporary, responsive, and intended to deliver an engaging user experience. Users may visit the website at any time, and any devices, including computers, tablets, and smartphones, can use it.

The Book App allows users to browse a vast library of books, experiment with various genres, and discover fresh writers. With the help of the app, users can easily find new books, read reviews and ratings, and interact with other readers who have similar interests.

The Book App also has features for writers, who may use them to build profiles, advertise their work, and engage with readers. Writers may increase their readership, visibility, and access to constructive criticism with the aid of modern technology.

Through the MERN stack-based Book App project, users and writers have access to a wide-ranging platform for communication, book discovery, and book-related trade. It is an excellent resource for readers and a successful strategy for writers to market their writing.

As a result, the MERN stack-based Book App is a unique and practical tool for readers and writers. It offers a comprehensive platform for discovering new publications, communicating with readers, and marketing and spreading content. Due to its extensive features and user-friendly UI, The Book App is a vital tool for both book lovers and authors.

1.2 Problem Statement:

Due to the growing popularity of online book sales, the usual brick-and-mortar retailers have encountered several difficulties. More individuals are now buying their books from online stores since online book sales have increased significantly recently. The bulk of online book vendors, however, don't offer the same level of interaction and customer care as traditional brick-and-mortar bookstores.

The issue is that there isn't an online bookshop that can provide clients the specialized care and in-person interaction of a brick-and-mortar store while still offering the ease of an online purchase. Most online book retailers do not provide the elements that customers need, such as the ability to explore and discover new books, communicate with other readers, and get individualized suggestions.

Making a book-based e-commerce website that makes use of the MERN stack might solve this issue. This internet application may offer readers a platform for finding new books, interacting with one another, and buying online for their preferred books. The MERN stack can provide an e-commerce website the flexibility, scalability, and user-friendliness needed.

The website has a reader community where users may interact and recommend books, as well as a straightforward user interface and tailored recommendations. The website may employ a range of social media and marketing strategies, including email marketing and social media advertising, to raise its profile and draw in more visitors.

By providing a platform that offers a tailored user experience, book recommendations, and a reader community while also providing the convenience of online shopping, a book-based e-commerce website built on the MERN stack aims to bridge the gap between traditional brick-and-mortar bookstores and online bookstores.

1.3 Objectives:

- To provide book enthusiasts a convenient online book purchasing and browsing platform: With the help of a user-friendly website, customers will be able to explore and discover books, read book reviews, and buy books online.

- To create a community where readers can communicate, recommend books to one another, and discuss books they have read. The effort aims to create a social network where readers can communicate, recommend books to one another, and discuss books they have read.

- To provide a user-friendly, interactive environment: The project aims to provide a user-friendly, interactive setting that motivates people to browse and purchase books online.

- In order to provide writers a platform where they may advertise their material, interact with readers, and get book reviews, the initiative was created.

- The project tries to enhance the website's search engine visibility in order to increase its prominence in search results and attract more visitors.

- The project aims to integrate a wide range of security measures to safeguard the integrity and safety of user data in order to guarantee its security and privacy.

1.4 Methodology:

- **Planning and Analysis**: The first phase of development consists of analyzing and determining the project's requirements, goals, and scope. The functions, personalities, and user flow of the website must all be determined. Defining the aims and purposes of your website is the first stage. Determine the functionality and services you want to offer, the audience you want to connect with, and the website's objective. To understand your target audience's needs, interests, and behaviors, we must first define your target audience and develop user personas. We could use this data to develop a website that appeals to your target audience.

  Researching your competition will enable you to recognize them and discover their strengths and weaknesses. This data may be used to set your website apart from those of your rivals and pinpoint market niches. Define the website's components and features you wish to include, such as user profiles, social sharing, book reviews, and recommendation functions. Create a sitemap and user flow to help you envision how people will explore your website and interact with its features. Plan the architecture and user experience of your website using wireframes and mockups. By doing this, you'll be able to see any possible design flaws and guarantee a consistent user experience throughout the whole website. [8]

Establish a project plan and timeline to list important project milestones and deadlines. The resources your project will need, such as staff, gear, and software, should be determined. You might compute the project budget and identify potential dangers using this information. Building an excellent book-based website often requires planning and study. To ensure the project's success, it is necessary to identify the target audience, specify the features and objectives of the website, and create a thorough project plan.

- **Design:** Throughout the design phase, wireframes, mockups, and prototypes are created for the user interface, navigation, and layout of the website.

  For designing and components we have used **React-Bootstrap:**
  Responsive web apps are created using the well-known UI framework React-Bootstrap. The Bootstrap CSS framework was used to generate a collection of pre-built, editable components.

  There are several UI components offered by React-Bootstrap, including navigation bars, forms, buttons, modals, and more. These React-built parts are incredibly flexible and easy to customize.

  Maintaining and upgrading your application is made simpler by React-Bootstrap's uniform appearance and feel. This is among its key advantages. The components' flexible design allows them to adapt to various screen sizes and devices.

  For styling we have used **SCSS (Sassy CSS):**

Sassy CSS, a preprocessor for CSS, adds additional functionality to the CSS syntax. It is an improved version of CSS that enables the development of more sophisticated styles with less code. Given that SCSS is a superset of CSS, every valid CSS code also qualifies as a valid SCSS code. However, SCSS also adds new syntax, such as functions, mixins, variables, and nesting.

SCSS has many features, including:

Variables: In SCSS, variables may be declared and given values to be used throughout your stylesheet. As a result, it is easier to maintain and update your code since you can modify a variable only once and have it affected every instance of your stylesheet.

Nesting: You may organize and make your code more readable by nesting selections inside of other selectors using SCSS. Because of this, you might have to write less repetitive code.

Mixins: By using reusable code blocks in your CSS, mixins make it simpler to apply sophisticated styles to your HTML components.

Functions: Mixins and functions are similar, but functions are more flexible and powerful since they accept parameters and return values.

Overall, SCSS may make the process of creating CSS easier and assist you in writing more effective and manageable code. The preprocessor required to transform the SCSS code into standard CSS, however, can make your development process more challenging.

- **Front-end development:** React is used in the front-end development stage of a website to create the user interface and client-side functionality. Making the website's pages, elements, and interactive features falls under this category.

User interfaces (UIs) for web apps are made using the popular JavaScript library React. It was developed by Facebook and made publically available in 2013. React is now supported by Facebook and a substantial development community.

Reusable UI components may be made by developers using React and used in several places of an application. Simply creating complex user interfaces is possible with these JavaScript-built components. A virtual DOM (Document Object Model) is used by React to manage the state of the user interface and render updates quickly.

React's essential characteristics include things like:

- **Component-based architecture**: With React, you can create reusable user interface (UI) components that can be swiftly combined to create intricate UIs.
- **Declarative programming**: You declare your goals and React takes care of the implementation details in this declarative programming style.
- **Virtual DOM**: Performance is improved by React since it controls the user interface's state and renders changes rapidly.
- **JSX**: By allowing you to write HTML-like code in JavaScript files, the JSX syntactic extension for JavaScript makes it easier to write and understand code.

React supports a wide range of libraries and frameworks, including Redux, React Router, and Material UI. Additionally, it is quite expandable.

React is a robust and flexible framework that can be used to build user interfaces for internet applications. Developers like it because of its component-based architecture, declarative programming, and virtual DOM.

- **Back-end development**: For backend development, Node.js is used to write clean JS code for server development and also Express.js is used to build decent routes and easy to use controllers, services, and databases using Mongoose. In order to do this, new features must be developed, including social sharing, book search, user authentication, and recommendation engines.

Node.js has a number of important features, including:

- **Event Driven I/O**: Node.js uses a non-blocking, event-driven I/O model that enables it to manage several connections simultaneously without slowing down the program's performance.
- **NPM**: Node.js has an integrated package management called NPM (Node Package management) that enables programmers to quickly install and maintain third-party libraries and modules.
- **Scalability**: Because Node.js is built to be extremely scalable, it is a popular option for developing apps that must manage a lot of traffic.
- **Cross-platform**: Node.js can work on any system (Linux, Mac Windows).

APIs (Application Programming Interfaces) are regularly developed with Node.js, as are web apps and real-time applications. Its interoperability with a variety of web frameworks, such as Express.js, etc. allows developers a range of choices.

Express JS:

An open-source web application framework for Node.js called Express.js is popular. It offers a complete set of capabilities that make it easy and quick to create web apps and application programming interfaces (APIs).

Due to the fact that Express.js, which is based on Node.js, allows JavaScript to be used in both the front end and the back end of a web application. It offers a simple framework for creating web applications, allowing programmers to concentrate on the essential components of their project without being constrained by needlessly complex framework features.

Express.js's important characteristics include the following:

- **Routing:** Express.js provides a simple and flexible method to build routes for handling HTTP requests and responses.
- **Middleware**: Programmers may integrate functions like logging, authentication, and error handling into their applications using the middleware layer provided by Express.js.
- **Template Engine:** EJS and Pug are just a few of the many templating engines that Express.js supports, making it simple to create dynamic HTML pages.
- **Modularity:** Express.js's modularity feature makes it simple for developers to divide their applications into smaller, reusable components.

Express.js is frequently used to create APIs and online apps. It frequently pairs with other well-known Node.js frameworks and modules, like MongoDB.

- **Database development:** During this stage, MongoDB is used to design and create the queries, data models, and database structure that are necessary for the website.

MongoDB, a well-known database management system for documents, makes use of NoSQL (non-relational) technology. It has grown in prominence as a platform for developing cutting-edge web apps since its release in 2009.

MongoDB stores data as flexible documents with varying forms that mimic JSON. This makes it possible for developers to store and retrieve sophisticated data structures without having to conform to a set model, unlike traditional relational databases.

MongoDB has a number of important characteristics, including:

- **Scalability**: MongoDB can manage massive volumes of data and traffic because of its highly scalable architecture.

- **Flexibility**: Developers may store data in a flexible manner using MongoDB's document-oriented data format, which makes it simple to modify and update data structures.

- **Performance**: MongoDB has built-in caching and automated sharding, making it a very performant database.

- **Aggregation:** MongoDB has strong aggregation features, making it simple to carry out complicated data analysis and aggregation.

  MongoDB is commonly used to create real-time applications and application programming interfaces (APIs). It typically works in conjunction with cutting-edge web development technologies like Node. All things considered, MongoDB is a robust and flexible database management system with a lot to offer creators of modern web applications. Because of its document-oriented data format, scalability, and performance, developers regularly employ it.

- **Integration and testing:** During the integration and testing phase, the front-end, back-end, and database components are integrated and the functionality, performance, and security of the website are tested.

CHAPTER 2

SYSTEM DEVELOPMENT

Why do we need E-Commerce websites?

E-commerce websites are essential for many reasons:

- **Convenience**: E-commerce websites make it simple for people to buy things without having to go to real stores since they let them shop from the comfort of their homes or offices at any time of day or night.

- **Global audience**: E-commerce websites allow companies to contact a worldwide clientele, erasing geographical boundaries and enabling them to grow their clientele beyond their local region.

- **Cost-effectiveness**: E-commerce websites sometimes have lower overhead costs than conventional businesses do, such as rent, utilities, and personnel pay.

- **Sales growth**: E-commerce websites may assist firms in growing their sales by facilitating consumer product discovery and purchases, delivering personalized suggestions, and providing discounts and promotions.

- **Analytics**: E-commerce websites give businesses access to useful data and analytics that can be utilized to streamline processes, enhance customer satisfaction, and boost sales.

  So, we decided to design a website for book sales which is completely authenticated and secure and easy to use for customers.

  Our project can be broken down into the following stages:

- **Planning and Analysis:** During this phase, the project's parameters and the needs for the e-commerce website are established. This includes figuring out the target market, the kinds of books to sell, the payment choices, the shipping procedures, and other aspects that must be put into place. The website's blueprint is created once a thorough examination of the requirements is completed.

- **Design**: The layout, color scheme, typography, and visuals for the website are all produced at this stage. The layout should be simple to use, visually appealing, and navigable.

  An essential component of the overall design and user experience of a MERN (MongoDB, Express, React, Node.js) programme is the UI (User Interface). Here are some pointers for creating a MERN app's user interface:

- **Ensure simplicity**: Users may find it easier to explore your app with a clear and straightforward design. Make good use of the white space and refrain from overcrowding the interface.

- **Use a responsive design**: As more and more people use the internet from mobile devices, it's critical to make sure your MERN app is adaptable and can adjust to various screen sizes.

- **Pick a color palette that is dependable**: Choose a color scheme for your MERN app that represents your brand and stick with it.

- **Use icons and graphics:** to assist users rapidly distinguish between different parts in your MERN app and to improve the app's aesthetic attractiveness.

- **Easy to navigate**: Make sure the MERN app is simple and easy to navigate. Users should be able to navigate the app without getting lost and locate what they're searching for promptly.

- **Provide users’ feedback**: When users engage with various MERN app components, provide them feedback. Give a visual confirmation, for instance, when a button is clicked or a form is submitted.

- **Development**: At this point, the MERN stack is actually used to start building the website. React and React-Bootstrap are used to build the front end, and Node.js, Express.js, and MongoDB are used to build the back end. The website should be enhanced for scalability, security, and performance.

- **Testing**: A website undergoes a rigorous testing procedure once it is developed in order to identify and address any flaws or issues. It is important to do functional, security, performance, and usability tests.

  Testing is a key phase in the development of software since it ensures that the programme works as planned. The three forms of testing that may be done are unit testing, integration testing, and end-to-end testing.
  Unit testing is a method used to test certain elements or functionalities of an application. The MERN stack may be subjected to unit testing using testing frameworks like Mocha or Jest.
  Integration testing examines how various application components interact with one another. Trying things out, like how the frontend and backend interact.
  Frameworks like Cypress or Selenium may be used for integration testing.
  End-to-end testing entails evaluating the entire programme while mimicking user interactions. Frameworks like Puppeteer or Test Case may be used for end-to-end testing.

In addition to these many types of testing, other aspects of testing, such as load testing and security testing, must be considered.

Demand testing is the process of analyzing an application's performance under a heavy demand. For this, you can use programme like Apache JMeter or LoadRunner.

Security testing is the process of putting the application's security measures to the test to make sure it is protected from attacks. Two tools that might be used for this are OWASP ZAP and Burp Suite.

- **Deployment**: Following extensive testing, the website is set up on a web server or cloud hosting platform. To guarantee that the website is constantly current and functioning properly, the deployment process should be automated and incorporate continuous integration and deployment (CI/CD).

Technologies/Libraries used:

1) **React-Bootstrap:** In order to employ helpful React.js components, we choose to use the **React-Bootstrap** framework. Speaking specifically about the library:

   React-Bootstrap, a well-known UI framework, is used to build responsive web applications. It is a set of ready-to-use, customizable components created with the help of the Bootstrap CSS framework.

   React-Bootstrap provides a wide range of UI components, such as navigation bars, forms, buttons, modals, and more. These React-built elements are easily style able and adaptable.

   React-Bootstrap gives a consistent look and feel throughout your application, making maintenance and updating easier. One of its main benefits is this. Because the components are responsive in design, they will adjust to different screen sizes and devices.

2) **React.js:** During the front-end development stage, React is used to create the website's user interface and client-side functionality. Making the pages, elements, and interactive features of the website falls under this category.

   React is a popular JavaScript library that is used to build user interfaces (UIs) for internet applications. It was developed by Facebook and released to the public in 2013. Today, React is supported by both Facebook and a substantial developer community.

   Reusable UI components that may be used in many different parts of an application can be made by developers using React. Simple user interface creation may be done using these JavaScript-built components. React uses a virtual DOM (Document Object Model) to maintain the state of the user interface and swiftly render changes.

   Redux, React Router, and Material UI are just a few of the many libraries and frameworks that React can be used with. It is also quite extensible.

   In general, React is a strong and adaptable toolkit for creating user interfaces in online applications. It is a well-liked option among developers due to its component-based architecture, declarative programming, and virtual DOM.

3) **Node.js:** Node.js is a backend-based framework which is used to develop API which can be used for various purposes. It is an open-source platform based on JavaScript. Previously only feasible with languages like Python, Ruby, and PHP, writing server-side code is now made available with JavaScript.

   Developing server-side apps, online APIs, and microservices with Node.js is common practice in web development. The creation of desktop apps and software for the Internet of Things (IoT) are only a couple of the numerous businesses that employ it.

4) **Express.js:** Express is a Node.js web framework with a lot of features for developing online and mobile apps. It is speedy, flexible, and easy to use. It has developed into one of the web frameworks for building web apps that is most widely used in the Node.js community.

   Express provides a simple and approachable API for controlling HTTP requests and responses. In order to improve the functionality of the web application, middleware, which handles duties like authentication, logging, and error handling, is also included. EJS and Pug are only two of the many templating engines that Express supports and allows you to employ to create dynamic HTML pages.

   Express is open source, and a substantial developer community actively contributes to and supports its development.

5) **MongoDB:** In this project, mongoose was used for MongoDB. Mongoose is an Object Data Modelling (ODM) module for Node.js and MongoDB. The higher level of abstraction it provides over the MongoDB driver makes working with MongoDB databases in Node.js applications easier. Mongoose uses a schema, which is a description of the document's structure, to build your data models. The data may then be subject to limitations like data types, required fields, default values, and validation procedures. Mongoose provides a wide range of options for dealing with data, including querying, updating, deleting, and aggregating data.

6) **Nodemailer:** For sending email to the admin and user after completing the shopping, we have used the Nodemailer module which is built in Node.js. Email messages may be sent via a variety of transport protocols thanks to its simple and flexible interface. Using Nodemailer makes sending emails with HTML content, attachments, and integrated images straightforward.

Nodemailer supports many recipients and may send emails using a variety of SMTP servers or services, such as Gmail. Since it supports a wide range of email authentication techniques, such as OAuth2, it is genuine to communicate with many email providers. [16]

One of the main advantages of Nodemailer is its capacity to handle errors and automatically repeat failed email sending attempts. Additionally, it might be configured to monitor email sending problems using custom error handling algorithms or third-party error monitoring systems.

7) **JWT:** JWT, or JSON Web Tokens, is a sort of authentication technique that lets users securely send data between parties in the form of a JSON object. [14]

   With a special secret key that only the server is aware of, the server produces a token in a JWT authentication system. After a successful login, this token is provided to the client and saved there. The client provides this token in the request header for each future request to the server, enabling the server to validate the token and validate the user.

   Compared to conventional session-based authentication systems, JWT authentication provides a number of benefits. First off, because JWT tokens are complete and capable of holding all relevant user data, they can do away with the necessity for session storage on the server. As a result, JWT authentication can grow considerably more easily since it requires less server storage. [14]

   JWT authentication, however, may potentially have significant drawbacks. An attacker who successfully intercepts a token may be able to access all the secured resources linked to it. A security risk arises if a token is stolen or lost since JWT tokens cannot be invalidated until they expire because they are self-contained.

8) **Passport.js:** A popular middleware for authentication in Node.js apps are Passport.js. It is quite adaptable and works with a variety of authentication methods. Passport.js is renowned for its dependability, simplicity, and ease of usage. It offers a selection of authentication methods, including third-party, local, and social authentication providers.

   For username- and password-based authentication, which is frequently used for online applications, a local authentication technique is utilized. Users are authenticated through social authentication strategies using OAuth providers like Facebook, Google, Twitter, etc. Users can also be authenticated in Passport.js using third-party authentication services.

   By offering a standard interface and combining several authentication techniques, Passport.js streamlines the authentication and authorization process in Node.js apps. It simplifies the implementation of safe and dependable authentication in apps by abstracting the specifics of authentication from developers.

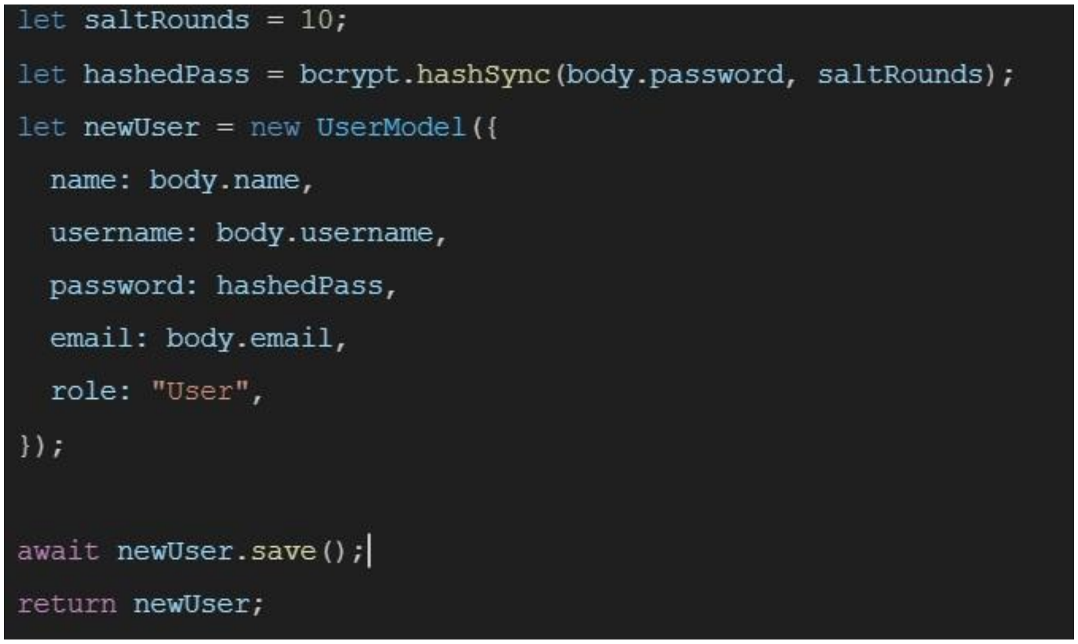
   Passport.js enables developers to design unique authentication methods depending on their particular needs in addition to the pre-built authentication strategies. Because developers may select the authentication strategy that best matches their application, authentication is now more flexible.

   Passport.js is a strong authentication mechanism for Node.js apps overall. It is a popular option for developers wishing to establish safe and dependable user authentication and authorization due to its flexibility, usability, and dependability.

9) **Bcrypt:** We have used the Bcrypt library for encryption and decryption of the user passwords. For password hashing and encryption, Node.js users frequently use the Bcrypt package. By encrypting user passwords

into an unbreakable, irreversible string of characters, it offers a method for securely storing user passwords. Bcrypt creates distinct and unexpected encrypted passwords for each user by combining salting and hashing methods.

For saving the password in encrypted format:



```
let saltRounds = 10;
let hashedPass = bcrypt.hashSync(body.password, saltRounds);
let newUser = new UserModel({
  name: body.name,
  username: body.username,
  password: hashedPass,
  email: body.email,
  role: "User",
});

await newUser.save();
return newUser;
```

For verifying the user:

```
verified=bcrypt.compareSync(enteredPassword, actualPassword);
```

“Verified” is a boolean variable which tells whether the user is verified or not i.e. entered password matches the actual password or not.

Before hashing the user's password, salting includes adding a random string of characters to it. Even if two users share the same password, the resultant hash is unique to each of them as a consequence. The technique of hashing creates an unchangeable string of characters from the initial password and salt. Blowfish, a potent hashing algorithm, is used by Bcrypt to generate the hash.

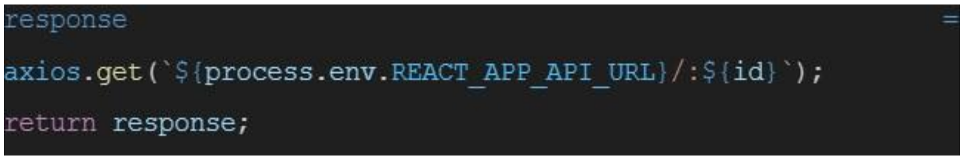
A way to compare a plaintext password with the hashed password is also included in Bcrypt. Bcrypt fetches the previously saved hash from the database and compares it with the hashed version of the user's password when they log in. The user is allowed access if they match.

Passwords may be safely stored in Node.js with the help of Bcrypt, which also shields user accounts from theft and hacking. It is a commonly used library that security professionals advise using to password-protect web applications.

10) **Axios:** A popular JavaScript package for sending HTTP requests to web servers is called Axios. Requests are often sent from the frontend React components to the backend Node.js server in MERN stack apps. [15]
For performing various HTTP requests, including GET, POST, PUT, DELETE, and others, Axios offers a number of alternative ways. Here are a few succinct descriptions of these techniques:

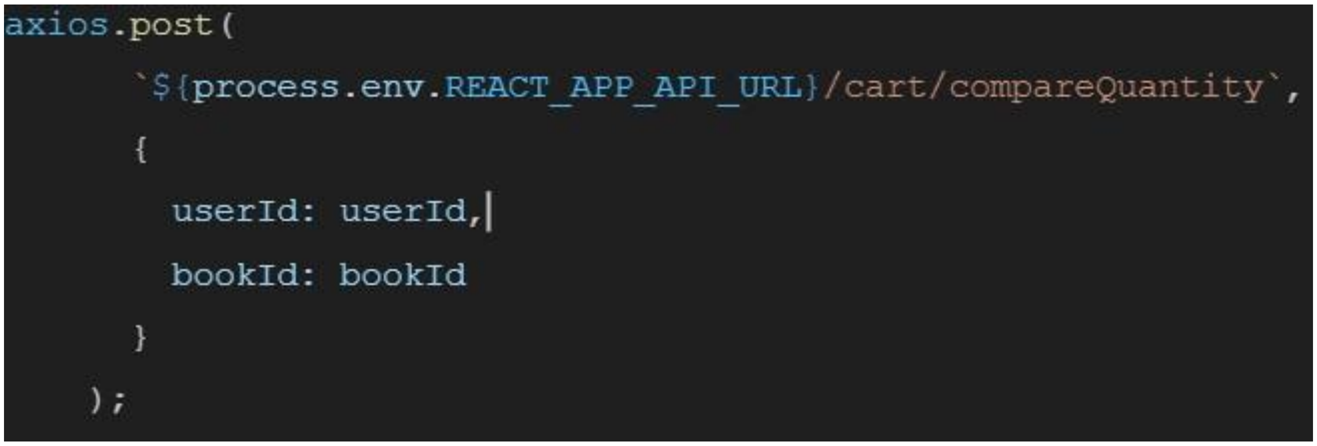
- **GET:** Data is retrieved from the server using the GET technique. To get data from a database or an API, it is frequently used. The axios.get () function in Axios may be used to invoke the GET method, passing the URL to be obtained as an argument.

Example:



```
response                                                          =
axios.get(`${process.env.REACT_APP_API_URL}/:${id}`);
return response;
```

- **POST:** Data is sent to the server via the POST method. A database or an API are frequently updated or added to using this method. The axios.post() function in Axios may be used to call the POST method, and the data to send is supplied as an argument.



```
axios.post(
      `${process.env.REACT_APP_API_URL}/cart/compareQuantity`,
      {
        userId: userId,
        bookId: bookId
      }
    );
```

Example:

- **PUT**: The server's existing data can be updated using this technique. The axios.put () function in Axios may be used to invoke the PUT method, passing the data to be updated as an argument.
Example:

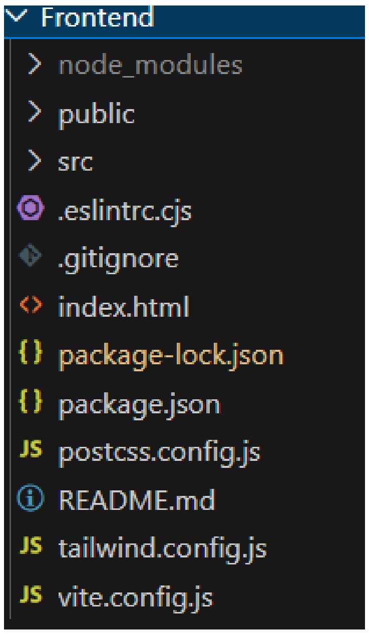
- **DELETE**: The server's data can be deleted using this technique. The axios. delete () function in Axios may be used to invoke the DELETE method, passing the data to be erased as an argument.

Example:

```
axios. delete(`${process.env.REACT_APP_API_URL}/${id}`);
```

Folder structure:

1) **Frontend:** Main App.js file is present in the parent directory of the project. Execution of the app starts from this file. The Src folder contains the components and each component is segregated into different folders with different purposes. We also have a services folder for the API calls and routes folder for protection of routes from unauthenticated users.

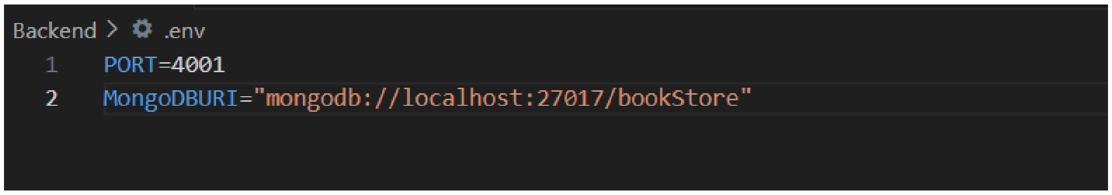


```
Frontend
  > node_modules
  > public
  > src
  .eslintrc.cjs
  .gitignore
  index.html
  package-lock.json
  package.json
  postcss.config.js
  README.md
  tailwind.config.js
  vite.config.js
```

Fig: Folder structure for Frontend

2) **Backend:** Execution of the backend folder starts from the server.js file which is present in the parent directory of the folder and different folders are present in the project:

Following line starts the execution of the backend code:



Backend > .env

```
PORT=4001
MongoDBURI="mongodb://localhost:27017/bookStore"
```

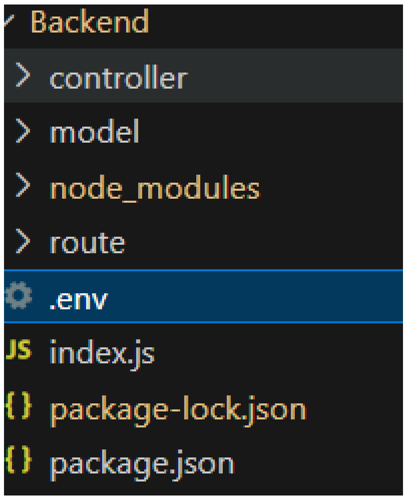
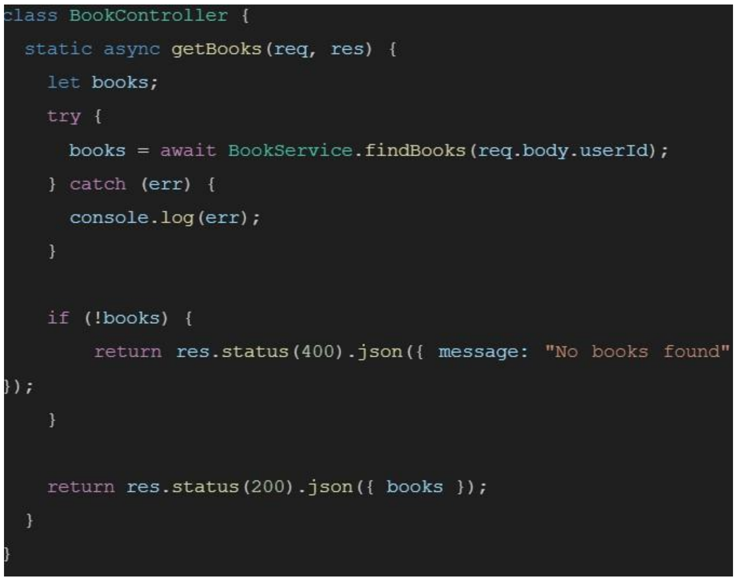


Fig : Folder structure for Backend

Controller: Contains files for functions that are to be executed for the routes hit from the frontend. Every controller has a class and they have static methods enclosed inside them.

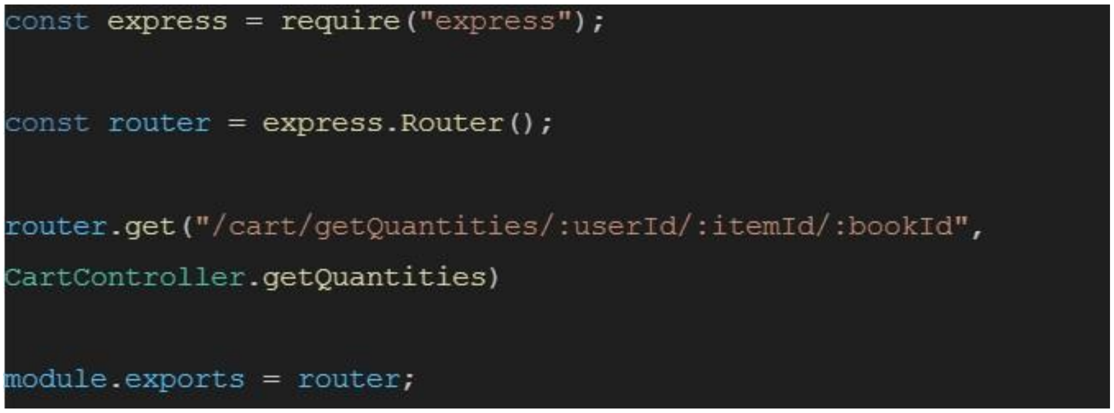
Example:



```
class BookController {
  static async getBooks(req, res) {
    let books;
    try {
      books = await BookService.findBooks(req.body.userId);
    } catch (err) {
      console.log(err);
    }

    if (!books) {
        return res.status(400).json({ message: "No books found"
});
    }

    return res.status(200).json({ books });
  }
}
```

Routes: Contains different routes and their description. Every router file contains a “router” instance of the express module.

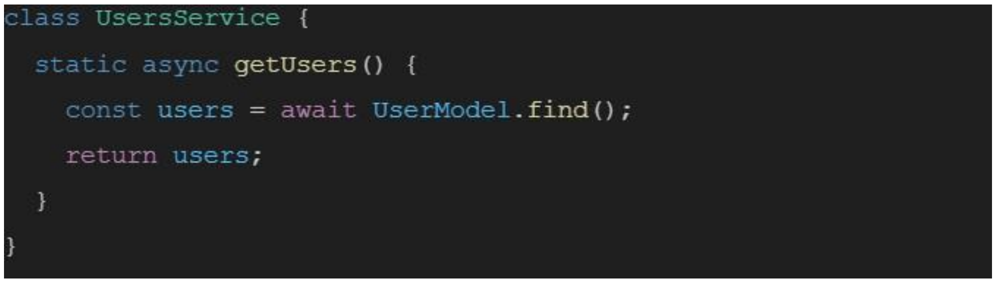
Example:



```
const express = require("express");

const router = express.Router();

router.get("/cart/getQuantities/:userId/:itemId/:bookId",
CartController.getQuantities)

module.exports = router;
```

Services: Contains mongoose calls to the database to fetch and post data for different controllers.Every service file has its own class and it has many static methods inside them.

Example:

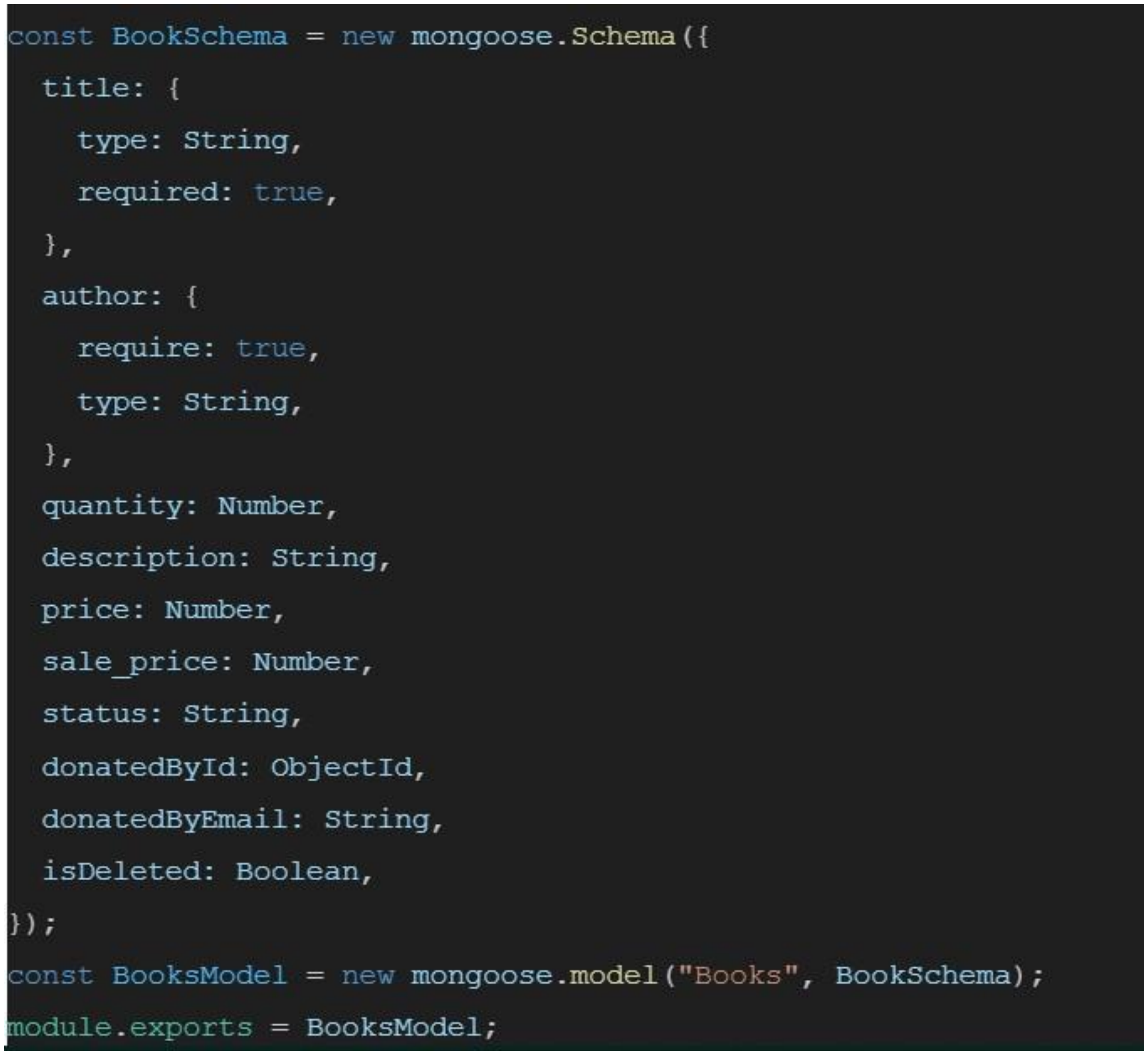


```
class UsersService {
  static async getUsers() {
    const users = await UserModel.find();
    return users;
  }
}
```

Database: Contains schema of different models.

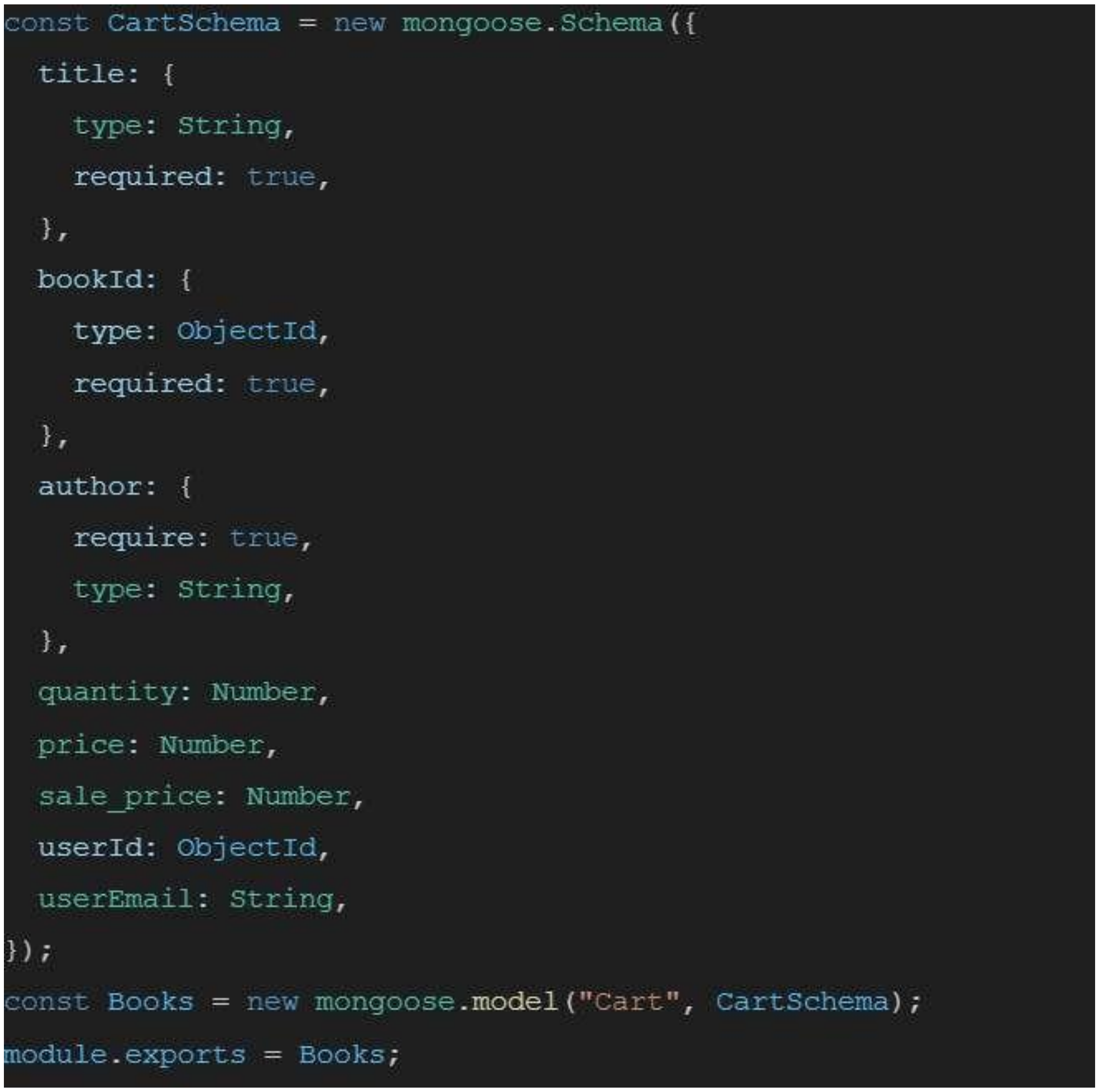
Schema of the databases:

1) **Books Schema:** Every book is donated by field which is the unique ObjectID of the user that donated the book.



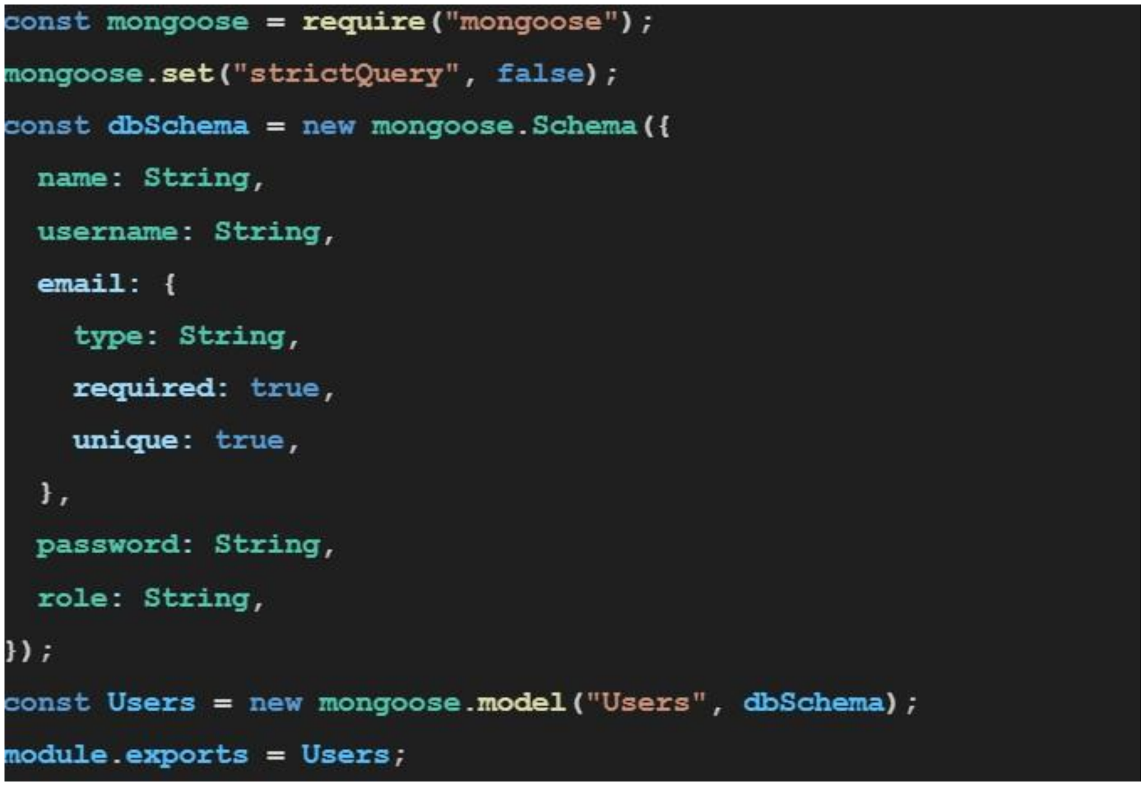
```
const BookSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  author: {
    require: true,
    type: String,
  },
  quantity: Number,
  description: String,
  price: Number,
  sale_price: Number,
  status: String,
  donatedById: ObjectId,
  donatedByEmail: String,
  isDeleted: Boolean,
});
const BooksModel = new mongoose.model("Books", BookSchema);
module.exports = BooksModel;
```

2) **Cart Schema:** It contains a collection of the cart where each cart item has a userId field which contains the unique ObjectID of the user in the user's collection. Thus, a user is mapped with the cart item.



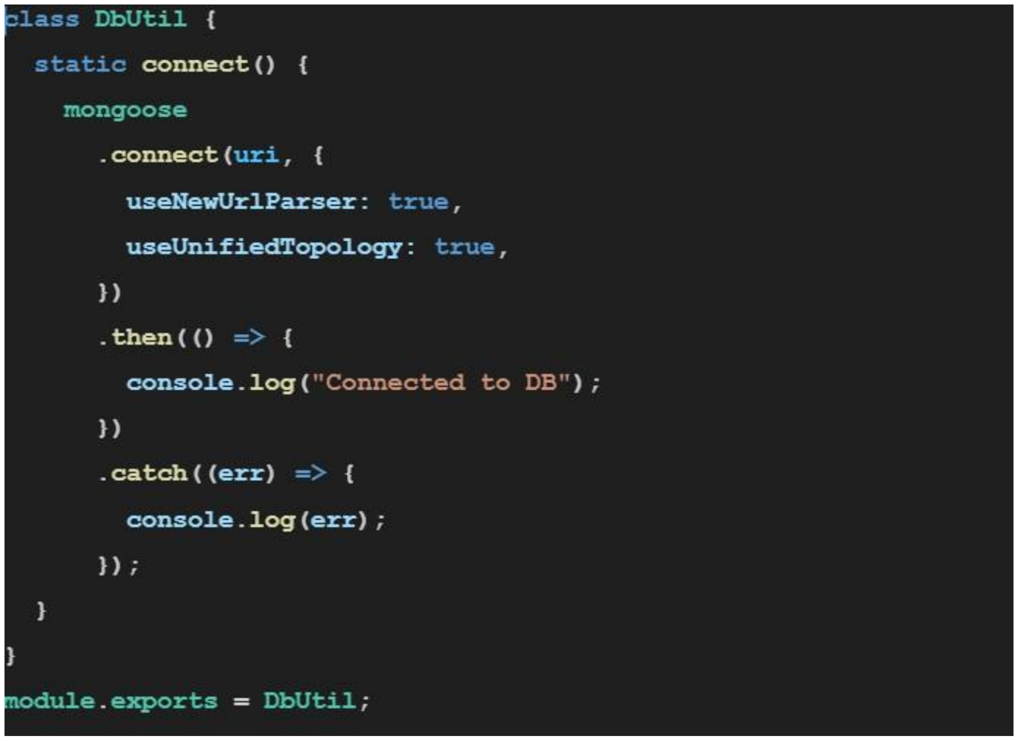
```
const CartSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  bookId: {
    type: ObjectId,
    required: true,
  },
  author: {
    require: true,
    type: String,
  },
  quantity: Number,
  price: Number,
  sale_price: Number,
  userId: ObjectId,
  userEmail: String,
});
const Books = new mongoose.model("Cart", CartSchema);
module.exports = Books;
```

3) **User Schema:** The user schema is a crucial component of web development because it enables programmers to specify the layout of the user object and guarantee that all necessary data is appropriately gathered and saved. The user's information must be accessed in order to authenticate their identity and establish their degree of access to the features and data of the application. This schema is also essential for authentication and authorization procedures.



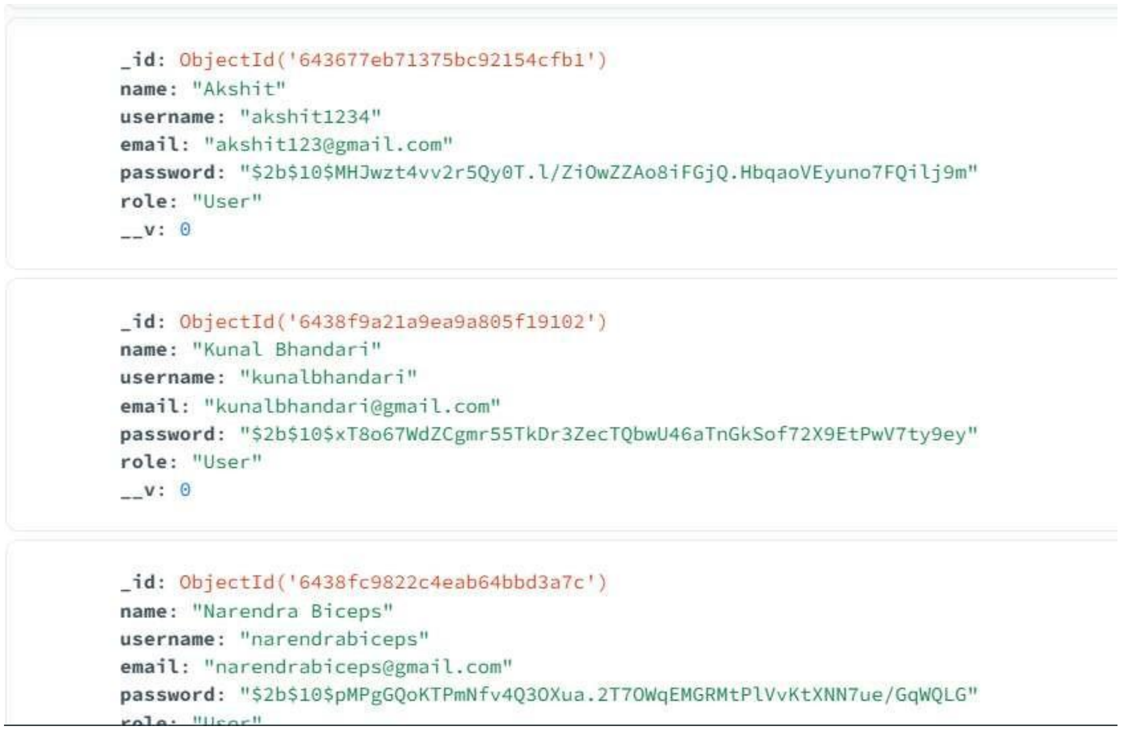
```
const mongoose = require("mongoose");
mongoose.set("strictQuery", false);
const dbSchema = new mongoose.Schema({
  name: String,
  username: String,
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: String,
  role: String,
});
const Users = new mongoose.model("Users", dbSchema);
module.exports = Users;
```

Connection file of the database:



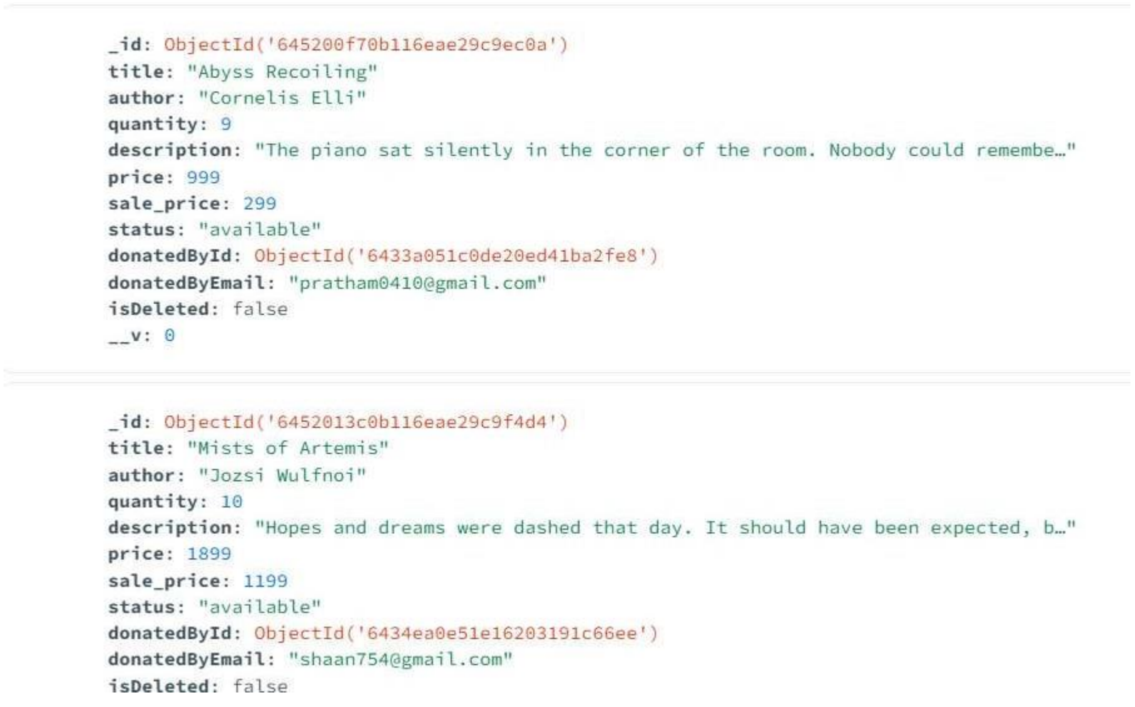
```
class DbUtil {
  static connect() {
    mongoose
      .connect(uri, {
        useNewUrlParser: true,
        useUnifiedTopology: true,
      })
      .then(() => {
        console.log("Connected to DB");
      })
      .catch((err) => {
        console.log(err);
      });
  }
}
module.exports = DbUtil;
```

Database:



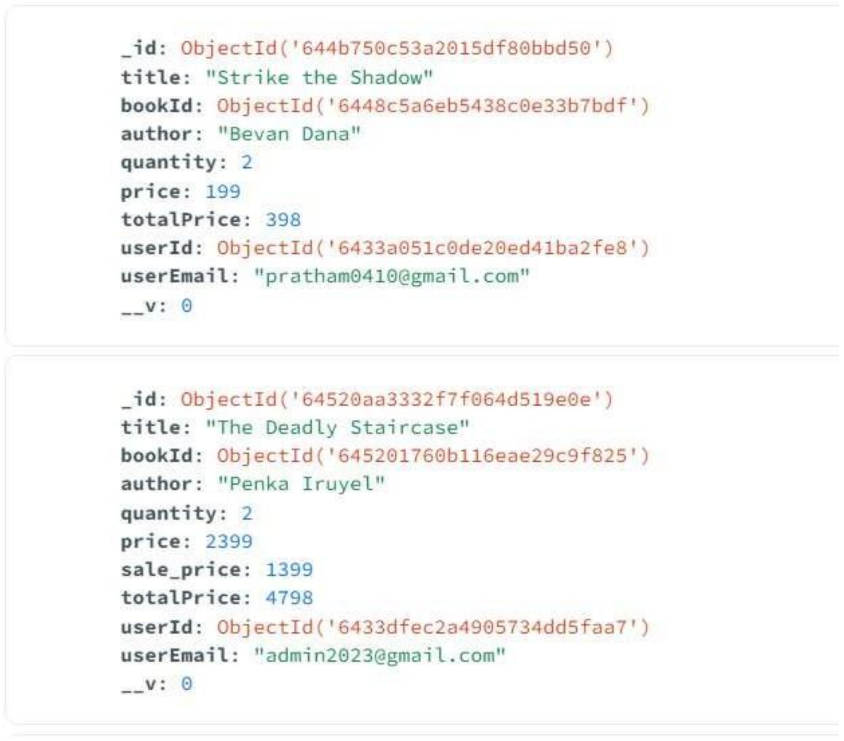
```
_id: ObjectId('643677eb71375bc92154cfb1')
name: "Akshit"
username: "akshit1234"
email: "akshit123@gmail.com"
password: "$2b$10$MHJwzt4vv2r5Qy0T.l/ZiOwZZAo8iFGjQ.HbqaoVEyuno7FQilj9m"
role: "User"
__v: 0

_id: ObjectId('6438f9a21a9ea9a805f19102')
name: "Kunal Bhandari"
username: "kunalbhandari"
email: "kunalbhandari@gmail.com"
password: "$2b$10$xT8o67WdZCgmr55TkDr3ZecTQbwU46aTnGkSof72X9EtPwV7ty9ey"
role: "User"
__v: 0

_id: ObjectId('6438fc9822c4eab64bbd3a7c')
name: "Narendra Biceps"
username: "narendrabiceps"
email: "narendrabiceps@gmail.com"
password: "$2b$10$pMPgGQoKTPmNfv4Q3OXua.2T7OWqEMGRMtPlVvKtXNN7ue/GqWQLG"
role: "User"
```

Fig: Users Collection



```
_id: ObjectId('645200f70b116eae29c9ec0a')
title: "Abyss Recoiling"
author: "Cornelis Elli"
quantity: 9
description: "The piano sat silently in the corner of the room. Nobody could remembe…"
price: 999
sale_price: 299
status: "available"
donatedById: ObjectId('6433a051c0de20ed41ba2fe8')
donatedByEmail: "pratham0410@gmail.com"
isDeleted: false
__v: 0

_id: ObjectId('6452013c0b116eae29c9f4d4')
title: "Mists of Artemis"
author: "Jozsi Wulfnoi"
quantity: 10
description: "Hopes and dreams were dashed that day. It should have been expected, b…"
price: 1899
sale_price: 1199
status: "available"
donatedById: ObjectId('6434ea0e51e16203191c66ee')
donatedByEmail: "shaan754@gmail.com"
isDeleted: false
```

Fig: Books Collection



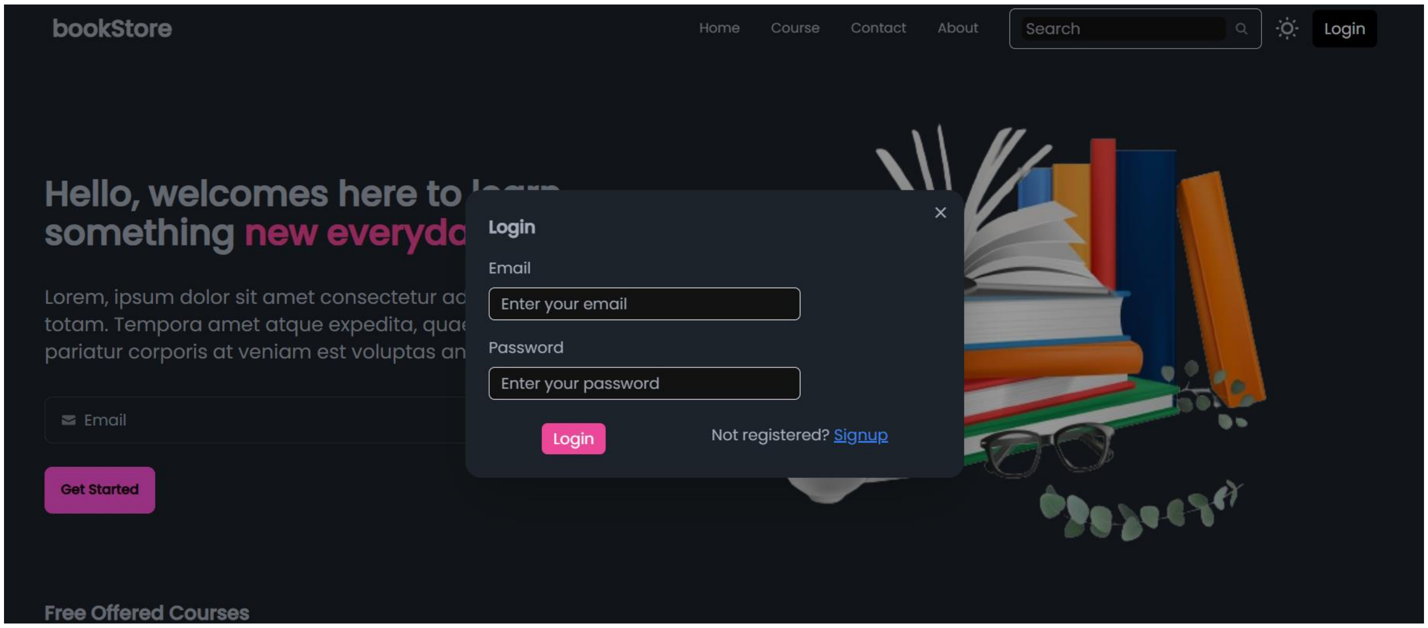
```
_id: ObjectId('644b750c53a2015df80bbd50')
title: "Strike the Shadow"
bookId: ObjectId('6448c5a6eb5438c0e33b7bdf')
author: "Bevan Dana"
quantity: 2
price: 199
totalPrice: 398
userId: ObjectId('6433a051c0de20ed41ba2fe8')
userEmail: "pratham0410@gmail.com"
__v: 0
```

```
_id: ObjectId('64520aa3332f7f064d519e0e')
title: "The Deadly Staircase"
bookId: ObjectId('645201760b116eae29c9f825')
author: "Penka Iruyel"
quantity: 2
price: 2399
sale_price: 1399
totalPrice: 4798
userId: ObjectId('6433dfec2a4905734dd5faa7')
userEmail: "admin2023@gmail.com"
__v: 0
```

Fig: Cart Collection

Flow of the App:

1) **Authentication:** First the user will have to login with his credentials. If he does not have the credentials then he can register with a new account. We have used Localstorage to save the data of logged in users and protect the routes.
   Authentication is a crucial security mechanism that aids in guarding against unauthorized access and safeguarding sensitive data. A website and its data might theoretically be accessed by anybody without authentication, which could result in data breaches, identity theft, and other security problems.



Hello, welcomes here to learn
something new everyda

Lorem, ipsum dolor sit amet consectetur ad
totam. Tempora amet atque expedita, quae
pariatur corporis at veniam est voluptas an

Login

Email

Enter your email

Password

Enter your password

Login

Not registered? Signup

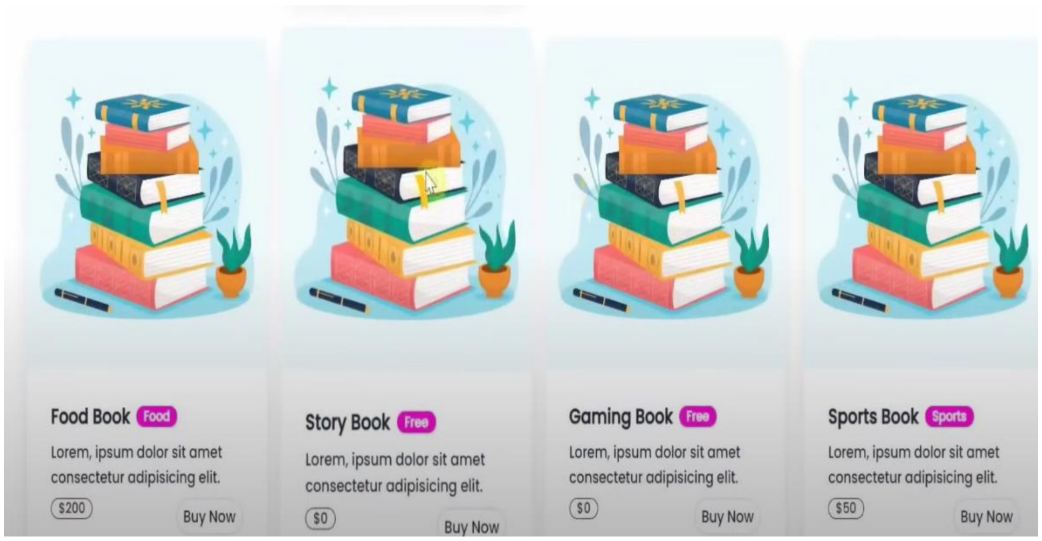
Fig: Login Page

2) **Books List:** Books list will be displayed to the user when he is logged in. We have tried to implement the features of a data table into the books table without using any library.

Data tables: Intuitive, feature-rich, and configurable HTML tables with sophisticated capabilities like pagination, sorting, filtering, searching, and more may be made with the help of the robust Data Tables jQuery plugin. It offers a straightforward and user-friendly interface for visualizing and modifying massive volumes of data from several sources, including JSON, XML, arrays, and server-side processing.

The MERN stack is only one of the many web technologies that Data Tables may be linked with. Data from MongoDB collections maybe shown using it, and users can interact with the data by using capabilities like sorting, searching, and pagination. Overall, Data Tables offers a complete solution for quickly and easily creating sophisticated and interactive HTML tables.

Developers may construct dynamic, responsive tables using Data Tables that are optimized for speed and performance. Additionally, it enables modification of the table's behavior and look using a number of choices and APIs. Additionally, it offers server-side processing, allowing the table to effectively manage big datasets.


Food Book Food
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
$200 Buy Now

Story Book Free
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
$0 Buy Now

Gaming Book Free
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
$0 Buy Now

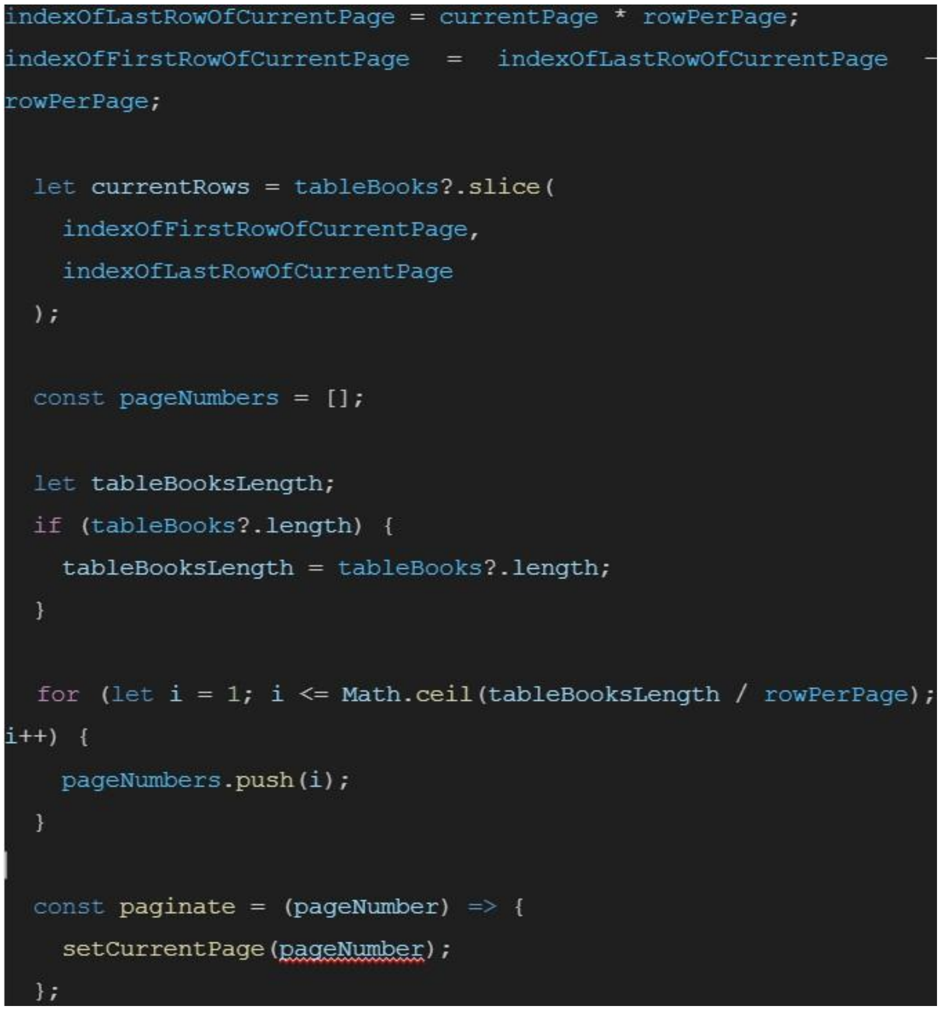
Sports Book Sports
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
$50 Buy Now

Fig : Books Table

Features of the booklist are:

- **Pagination:** The entire data is paginated so that when the number of books increases, then data can be displayed without scrolling the page.

Main code for pagination:



```
indexOfLastRowOfCurrentPage = currentPage * rowPerPage;
indexOfFirstRowOfCurrentPage  =  indexOfLastRowOfCurrentPage  -
rowPerPage;

  let currentRows = tableBooks?.slice(
    indexOfFirstRowOfCurrentPage,
    indexOfLastRowOfCurrentPage
  );

  const pageNumbers = [];

  let tableBooksLength;
  if (tableBooks?.length) {
    tableBooksLength = tableBooks?.length;
  }

  for (let i = 1; i <= Math.ceil(tableBooksLength / rowPerPage);
i++) {
    pageNumbers.push(i);
  }

  const paginate = (pageNumber) => {
    setCurrentPage(pageNumber);
  };
```

- **Tooltips:** When a user hovers on the status of the book, then he can see the quantity of book in the tooltip. Similarly, he can see the email ID of the user on hovering the donated column.

- **Search Box:** Users can search a particular book with the author name or book name.

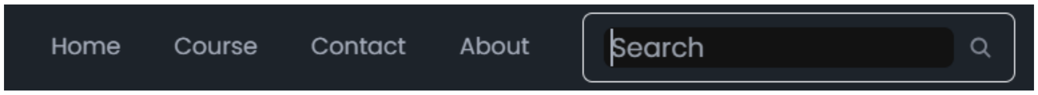
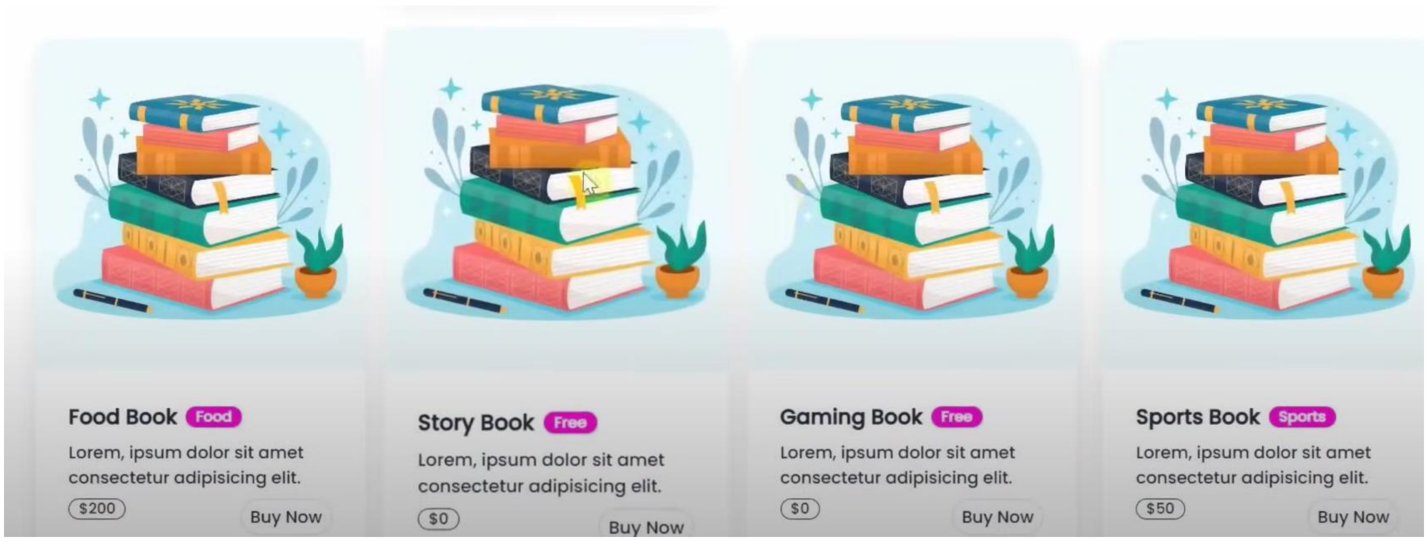
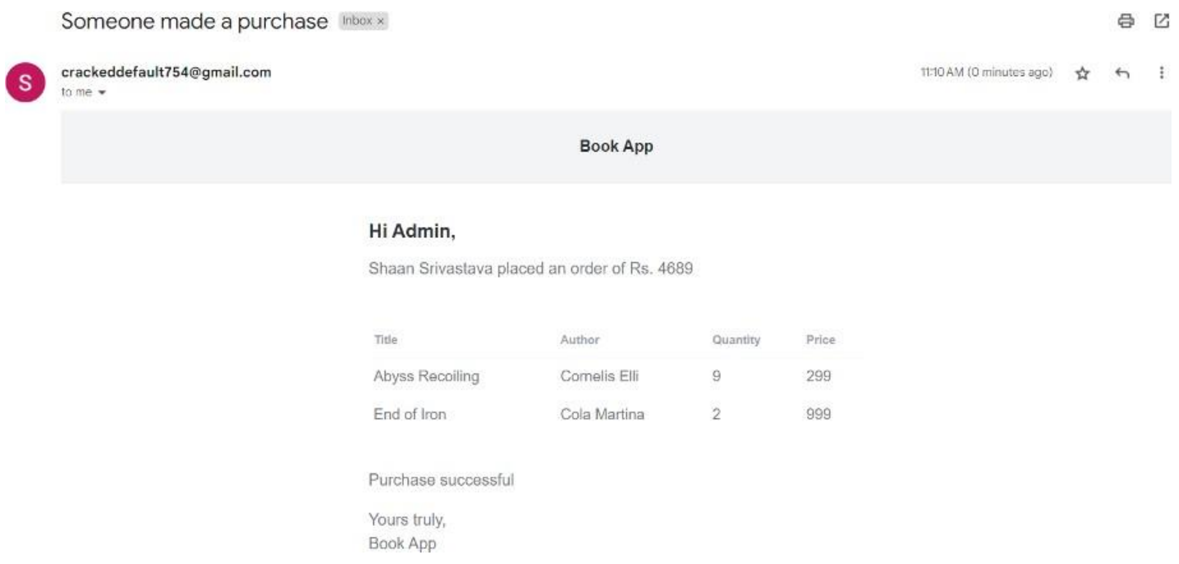


Fig: Search Functionality

3) **Book Details Page:** When a user clicks on the row of a book then he will get the details of the book. After going on the details page, the admin can edit the book details. And also, a normal user can add the book to the cart.


Food Book
Food
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
$200
Buy Now
Story Book
Free
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
$0
Buy Now
Gaming Book
Free
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
$0
Buy Now
Sports Book
Sports
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
$50
Buy Now

4) Cart Page: After clicking on add to cart the user can manage the quantities of books on the cart page. He can proceed to checkout after deciding which books to purchase. He and the admin both will receive an email with the description of the shopping cart.



Someone made a purchase Inbox ×

crackeddefault754@gmail.com
to me

11:10 AM (0 minutes ago)

Book App

Hi Admin,

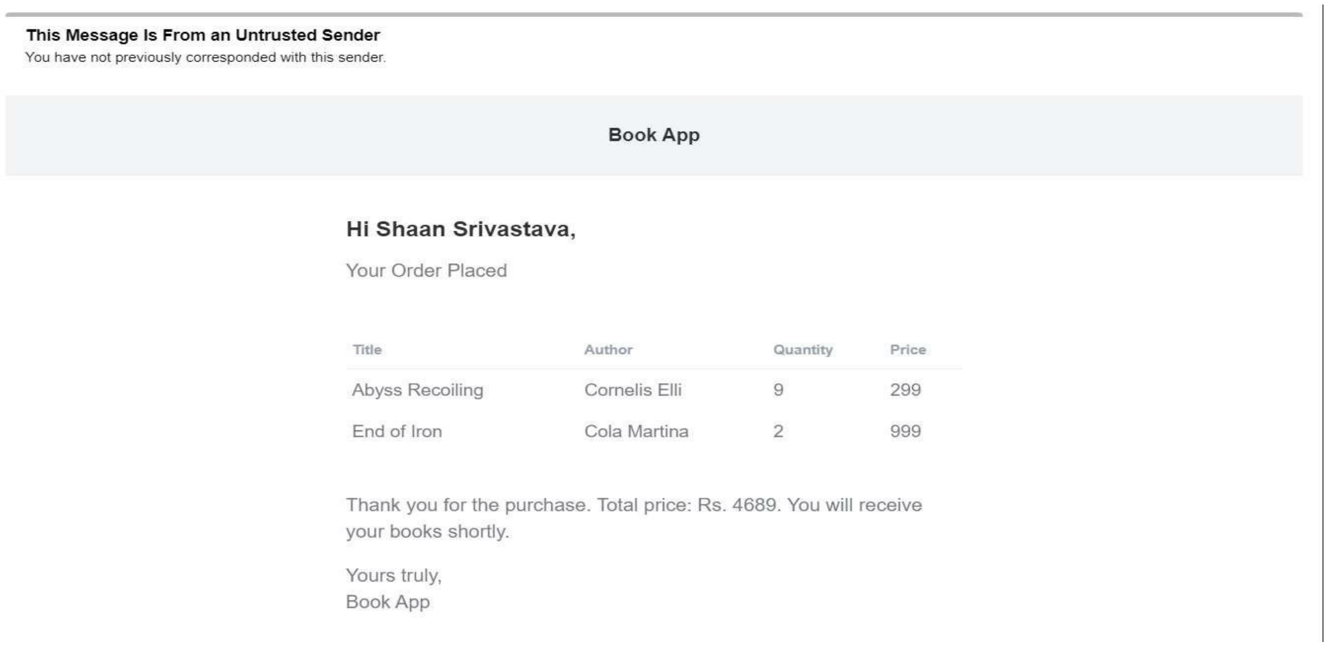
Shaan Srivastava placed an order of Rs. 4689

Title	Author	Quantity	Price
Abyss Recoiling	Cornelis Elli	9	299
End of Iron	Cola Martina	2	999

Purchase successful

Yours truly,
Book App

Fig: Email sent to the admin



This Message Is From an Untrusted Sender
You have not previously corresponded with this sender.

Book App

Hi Shaan Srivastava,

Your Order Placed

Title	Author	Quantity	Price
Abyss Recoiling	Cornelis Elli	9	299
End of Iron	Cola Martina	2	999

Thank you for the purchase. Total price: Rs. 4689. You will receive your books shortly.

Yours truly,
Book App

Fig: Email sent to the user