**Full Stack Development with MERN**

**Database Design and Development Report**

|  |  |
| --- | --- |
| Date | 16-07-2024 |
| Team ID | SWTID1719929609 |
| Project Name | Project – Book Nest |
| Maximum Marks | 5 |

**Project Title**: Book Nest

**Date**: 16-07-2024

**Prepared by**: Team Book Nest

**Objective**

The objective of this report is to outline the database design and implementation details for the Book Nest project, including schema design and database management system (DBMS) integration.

**Technologies Used**

* **Database Management System (DBMS):** MongoDB
* **Object-Document Mapper (ODM):** Mongoose

**Design the Database Schema**

The database schema is designed to accommodate the following entities and relationships:

1. **Users**

**1.\_id :** Automatically generated by MongoDB

**2.name :** Name of the user.

**3.email :** Email address of the user.

**4.password :** Password for user authentication

1. **Sellers**

**1.\_id :** Automatically generated by MongoDB.

**2.name :** Name of the Seller.

**3.email :** Email address of the Seller

**4.password :** Password for Seller authentication

1. **Admins**

**1.\_id:** Automatically generated by MongoDB.

**2.email :** Email address of the admin

**3.password :** password for admin authentication.

1. **Books**

**1.\_id:** Automatically generated by MongoDB.

**2.name:** Name of the Book

**3.author:** Author of the book

**4.seller** : Reference to the seller who is selling the book

**5.genre** : Genre of the book

**6**.**price** : Price of the Book

**7**.**imageurl:**URL for the book’s image

1. **userorders**

**1.\_id:** Automatically generated by MongoDB.

**2.user:** Reference to the user who placed the order

**3.book:** Reference to the book which is being ordered

**4.seller** : Reference to the seller who is selling the book

**5.address** : address of the user

**6**.**City**: city of the user

**7**.**pincode:**pincode of the user’s city

1. **orders**

**1.\_id:** Automatically generated by MongoDB.

**2.user:** Reference to the user who placed the order

**3.book:** Reference to the book which is being ordered

**4.seller** : Reference to the seller who is selling the book

1. **Wishlist**

**1.\_id:** Automatically generated by MongoDB.

**2.user:** Reference to the user who added the book to wishlist

**3.book:** Reference to the book which is being added to the wishlist

**Implement the Database using MongoDB**

The MongoDB database is implemented with the following collections and structures:

Database Name: BookNest

1. Collection: users

- Schema:

```

{ name:{type:String , required:true},

password:{type:String , required:true},

email:{type:String , required:true} }

```

2. Collection: Sellers

- Schema:

```

{

name:{type:String , required:true},

password:{type:String , required:true},

email:{type:String , required:true},

}

```

3. Collection: Admin

- Schema:

```

{

name:{type:String , required:true},

password:{type:String , required:true},

email:{type:String , required:true},

}

```

4. Collection: books

- Schema:

```

{

name:{type:String , required:true},

author: { type: String, required: true },

seller: { type: Schema.Types.ObjectId, ref: 'Seller', required: true },

genre : {type:String , required:true},

price : {type:Number , required:true},

imageurl : {type:String , required:true}

}

```

5. Collection: Orders

- Schema:

```

{

user:{type:Schema.Types.ObjectId , ref:'User',required:true},

book:{type:Schema.Types.ObjectId , ref:'Books',required:true},

seller:{type:Schema.Types.ObjectId , ref:'Seller',required:true},

createdOn:{type:Date , default:Date.now}

}

```

6. Collection:userorders

- Schema:

```

{

user:{type:Schema.Types.ObjectId , ref:'User',required:true},

book:{type:Schema.Types.ObjectId , ref:'Books',required:true},

seller:{type:Schema.Types.ObjectId , ref:'Seller',required:true},

address:{type:String , require:true},

City : {type:String , require:true},

pincode:{type:String , require:true},

createdOn:{type:Date , default:Date.now}

}

```

7. Collection:wishlist

- Schema:

```

{

user:{type:Schema.Types.ObjectId , ref:'Users'},

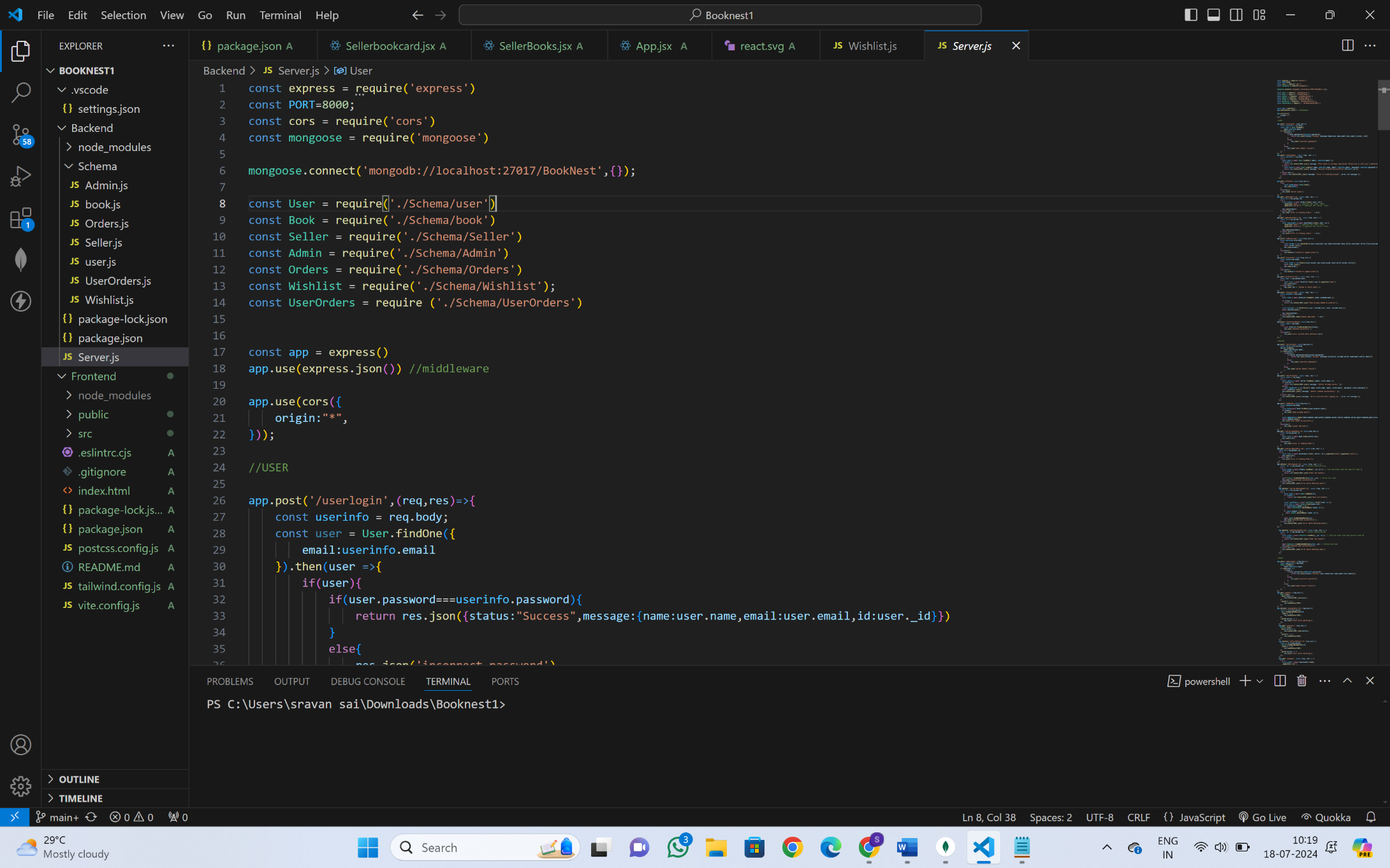
book:{type:Schema.Types.ObjectId , ref:'Books'}

}

```

**Integration with Backend**

* Database connection:



* The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
  + User Management: CRUD operations for users.

1.**Create** : register a new user

2. **Read** : retrieve user details.

3**.Update:**Modify user details

4.**Delete**: Remove a user

* + Seller Management: CRUD operations for users.

1.**Create** : register a new seller

2. **Read** : retrieve seller details.

3**.Update:**Modify seller details

4.**Delete**: Remove a seller

* + Book Management: CRUD operations for books, with seller authentication.

1.**Create** : Adds a new book

2. **Read** : retrieve books

3**.Update:**Modify book details

4.**Delete**: Remove a book

* + Admin Management: CRUD operations for admins

1.**Create** : register a new admin

2. **Read** : retrieve admin details.

3**.Update:**Modify admin details

4.**Delete**: Remove an admin

* + Order Management: CRUD operations for admins

1.**Create** : Adds a new order

2. **Read** : retrieve orders

3**.Update:**Modify order details

4.**Delete**: Remove a order

* + Wishlist Management: CRUD operations for wishlist

1.**Create** : Adds a new item to the wishlist

2. **Read** : retrieve wishlist items

3**.Update:**Modify wishlist details

4.**Delete**: Remove an item from the wishlist