**Full Stack Development with MERN**

**Database Design and Development Report**

|  |  |
| --- | --- |
| Date | 10 July 2024 |
| Team ID | SWTID1719939027 |
| Project Name | Project – Book Nest |
| Maximum Marks | 5 |

**Project Title**: Book Nest

**Date**: 10 July 2024

**Prepared by**: Yavish Sahrawat

**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**

* \_id: Automatically generated by MongoDB.
* username: Name of the user.
* email: Email address of the user (unique and required).
* password: Encrypted password for user authentication.
* createdAt: Date when the user account was created (defaulted to current date/time).
* address: Address of the user (required field).

1. **Sellers**

* id: Automatically generated by MongoDB.
* name: Name of the seller.
* email: Email address of the seller (unique and required).
* password: Encrypted password for seller authentication.
* createdAt: Date when the seller account was created (defaulted to current date/time).

1. **Admins**

* id: Automatically generated by MongoDB.
* username: Name of the admin.
* email: Email address of the admin (unique and required).
* password: Encrypted password for admin authentication.
* createdAt: Date when the admin account was created (defaulted to current date/time).

1. **Books**

* id: Automatically generated by MongoDB.
* name: Name of the book.
* author: Author of the book.
* seller: Reference to the Seller who is selling the book (stored as \_id referencing Sellers).
* createdAt: Date when the book entry was created (defaulted to current date/time).
* imgurl: URL for the book's image.
* price: Price of the book.

1. **Orders**

* id: Automatically generated by MongoDB.
* user: Reference to the User who placed the order (stored as \_id referencing Users).
* books: Reference to the Book(s) included in the order (stored as \_id referencing Books).
* seller: Reference to the Seller from whom the order is placed (stored as \_id referencing Sellers).
* bookname: Name of the book included in the order.
* imgurl: URL for the book's image included in the order.
* createdAt: Date when the order was created (defaulted to current date/time).

1. **Wishlist**

* id: Automatically generated by MongoDB.
* user: Reference to the User who added the book to their wishlist (stored as \_id referencing Users).
* bookname: Name of the book added to the wishlist.
* imgurl: URL for the book's image added to the wishlist.
* createdAt: Date when the book was added to the wishlist (defaulted to current date/time).

**Implement the Database using MongoDB**

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

**Database Name: test**

1. **Collection: users**

- Schema:

```

{

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

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

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

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

address:{type:String,required:true}

}

```

1. **Collection: sellers**

- Schema:

```

{

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

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

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

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

}

```

1. **Collection: admins**

- Schema:

```

{

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

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

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

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

}

```

1. **Collection: books**

- Schema:

```

{

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

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

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

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

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

price:{type:Number,required:true}

}

```

1. **Collection: orders**

- Schema:

```

{

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

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

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

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

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

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

}

```

1. **Collection: wishlists**

- Schema:

```

{

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

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

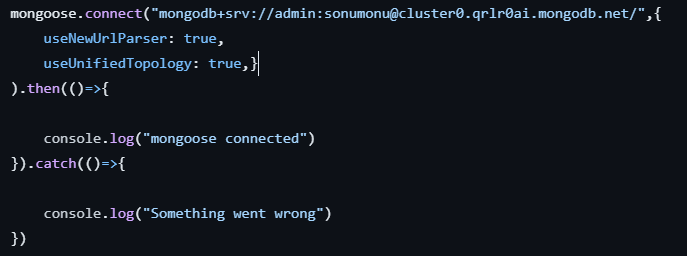
imgurl:{ type:String, required:true},

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

}

```

**Integration with Backend**



The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:

* 1. User Management: CRUD operations for users.
* **Create**: Register a new user.
* **Read**: Retrieve user details
* **Update**: Modify user details (e.g., password).
* **Delete**: Remove a user.
  1. Book Management: CRUD operations for books, with seller authentication.
* **Create**: Add a new book.
* **Read**: Retrieve books.
* **Update:** Modify book details.
* **Delete**: Remove a book.
  1. Admin Management: CRUD operations for admins.
* **Create**: Register a new admin.
* **Read**: Retrieve admin details.
* **Update**: Modify admin details (e.g., password).
* **Delete**: Remove an admin.
  1. Order Management: CRUD operations for orders.
* **Create**: Add a new order.
* **Read**: Retrieve orders.
* **Update** Modify order details.
* **Delete**: Remove an order
  1. Wishlist Management: CRUD operations for wishlist.
* **Create**: Add a new item to the wishlist.
* **Read**: Retrieve wishlist items.
* **Update**: Modify wishlist details.
* **Delete**: Remove an item from the wishlist.