**Full Stack Development with MERN**

**Database Design and Development Report**

|  |  |
| --- | --- |
| Date | 15/7/24 |
| Team ID | SWTID1719938064 |
| Project Name | ShopEZ: E-Commerce Application |
| Maximum Marks | 5 |

**Project Title**: ShopEZ: E-Commerce Application

**Date**: 15/7/24

**Prepared by**: Mokshagna & TEAM

**Objective**

The objective of this report is to outline the database design and implementation details for the ShopEZ: E-Commerce Application 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**

- Attributes: [username, password, email, usertype]

**2. Admin**

- Attributes: [banner,categories]

**3. products**

- Attributes: [title, description, mainImg, carousel, sizes, category, gender, price, discount]

**4. orders**

- Attributes: [userId, name, email, mobile, address, pincode, title, description, mainImg, size, quantity, price, discount, paymentMethod, orderDate, deliveryDate, orderStatus]

**5. cart**

- Attributes: [userId, title, description, mainImg, size, quantity, price, discount]

**Implement the Database using MongoDB**

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

Database Name: ShopEZ

1. Collection: **users**

- Schema:

```

const userSchema = new mongoose.Schema({

username: {type: String},

password: {type: String},

email: {type: String},

usertype: {type: String}

```

2. Collection: **admin**

- Schema:

```

const adminSchema = new mongoose.Schema({

banner: {type: String},

categories: {type: Array}

});

```

3. Collection: **products**

- Schema:

```

const productSchema = new mongoose.Schema({

title: {type: String},

description: {type: String},

mainImg: {type: String},

carousel: {type: Array},

sizes: {type: Array},

category: {type: String},

gender: {type: String},

price: {type: Number},

discount: {type: Number}

})

```

4. Collection: **orders**

- Schema:

```

const orderSchema = new mongoose.Schema({

userId: {type: String},

name: {type: String},

email: {type: String},

mobile: {type: String},

address: {type: String},

pincode: {type: String},

title: {type: String},

description: {type: String},

mainImg: {type: String},

size: {type: String},

quantity: {type: Number},

price: {type: Number},

discount: {type: Number},

paymentMethod: {type: String},

orderDate: {type: String},

deliveryDate: {type: String},

orderStatus: {type: String, default: 'order placed'}

})

```

5. Collection: **cart**

- Schema:

```

const cartSchema = new mongoose.Schema({

userId: {type: String},

title: {type: String},

description: {type: String},

mainImg: {type: String},

size: {type: String},

quantity: {type: String},

price: {type: Number},

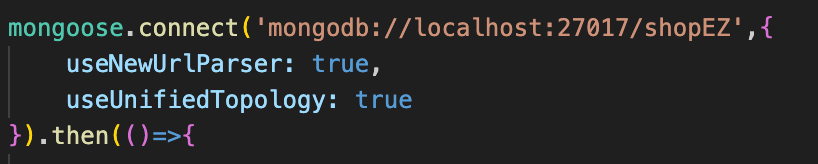
discount: {type: Number}

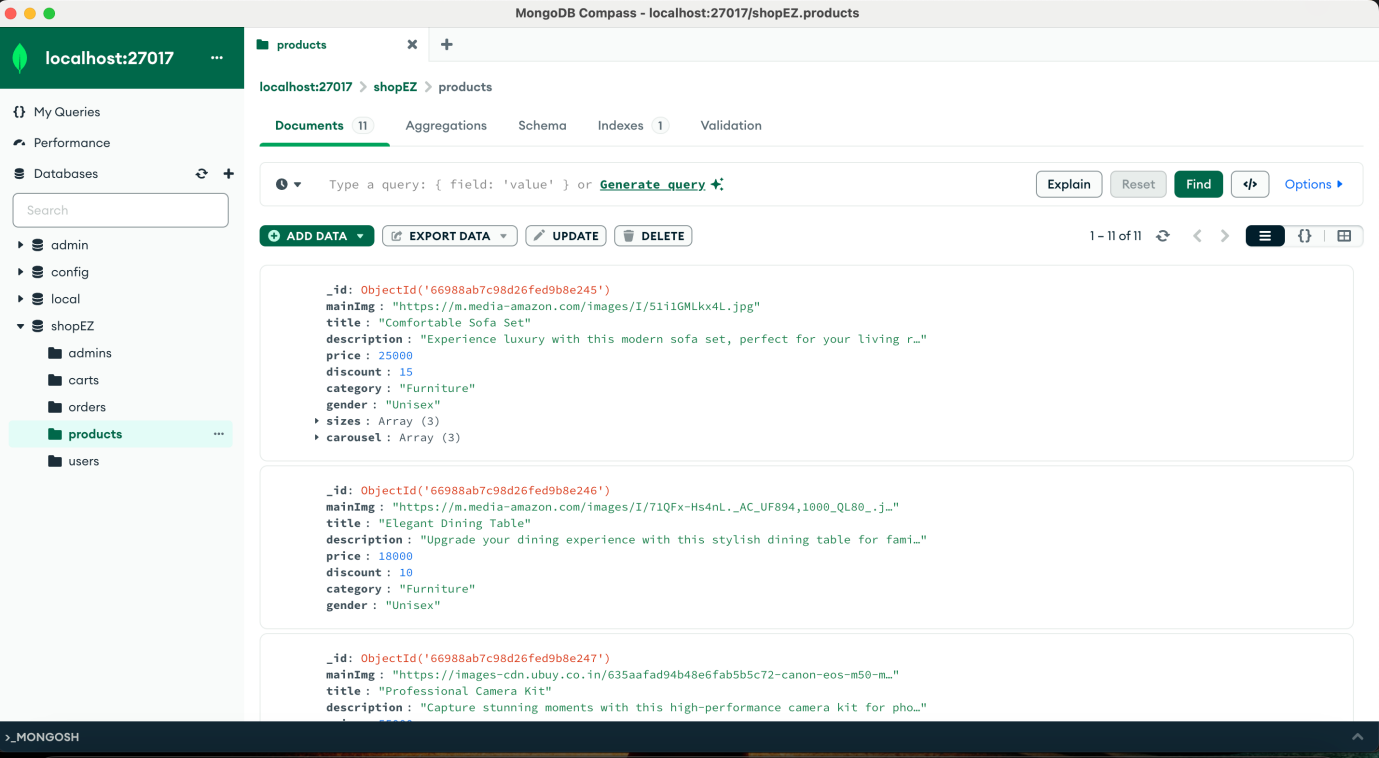
})

```

**Integration with Backend**

* Database connection: Screenshot of Database connection





* The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
  + User Management: CRUD operations for users.
  + Post Management: CRUD operations for posts, with user authentication.
  + Comment Management: CRUD operations for comments associated with posts.