

# Full Stack Development with MERN

## Database Design and Development Report

Date	5 July 2024
Team ID	SWTID1720091047
Project Name	Project: E-commerce application
Maximum Marks	

**Project Title:** -

**Date:** 5 July 2024

**Prepared by:** Magesh kumar S

### Objective

The objective of this report is to outline the database design and implementation details for the ShopEZ 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: [list attributes like \_id, username, password, email, usertype, approval, createdAt, updatedAt]

#### 2. Admin

- Attributes: [list attributes like \_id, Banner, categories]

#### 3. Product

- Attributes: [list attributes like \_id, Title, description, mainImg, carousel, sizes, category, gender, price, discount]

#### 4.Orders

- Attributes: [list attributes like  
\_id,userId,name,email,mobile,address,pincode,title,description,mainImg,size,quantity,price,discount,paymentMethod,orderDate,orderStatus]

#### 5.Cart

- Attributes: [list attributes  
like\_id,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: Test

##### 1. Collection: users

- Schema:

```
{
  username: {type: String},
  password: {type: String},
  email: {type: String},
  usertype: {type: String}
}
```

##### 2. Collection: admin

- Schema:

```
{
  banner: {type: String},
  categories: {type: Array}
}
```

##### 3. Collection: Product

- 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: order

- 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'}  
}  
...
```

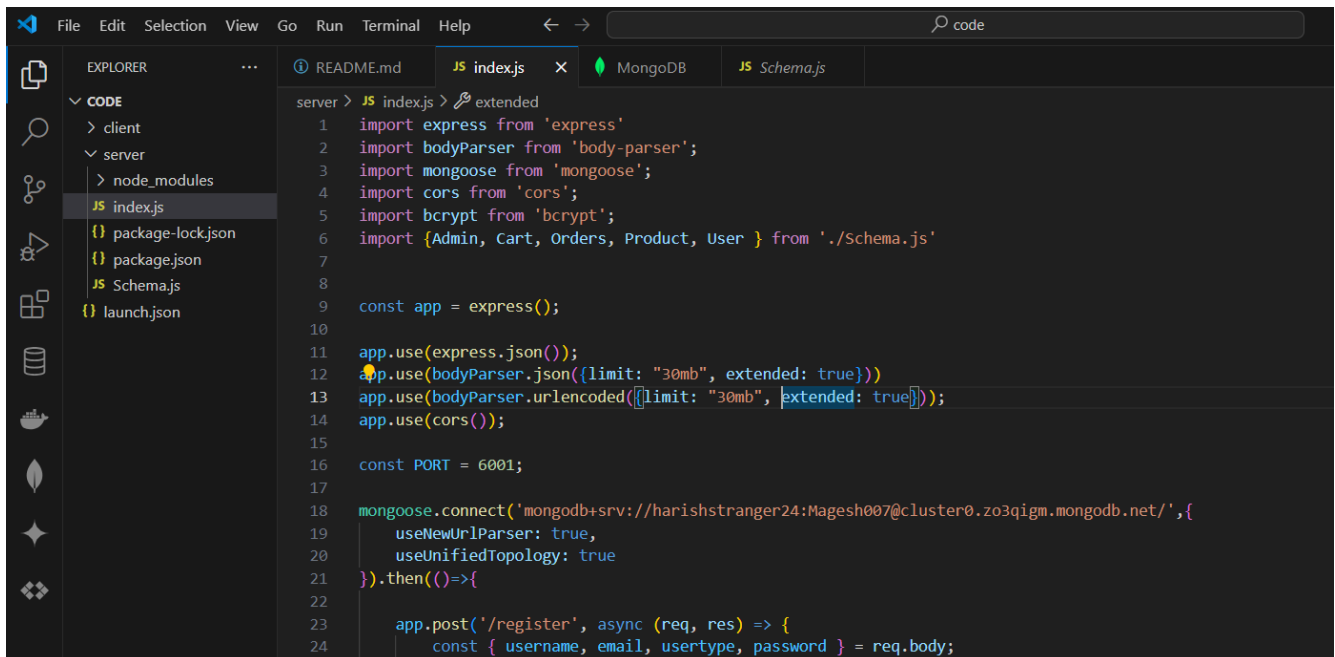
#### 5. Collection: cart

- Schema:

```
...  
  
{  
  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 done using Mongoose



The screenshot shows a VS Code editor window with the following components:

- EXPLORER:** A file tree on the left showing the project structure. The 'server' directory is expanded, showing 'index.js' as the selected file. Other files include 'package-lock.json', 'package.json', 'Schema.js', and 'launch.json'.
- Editor:** The main area displays the content of 'index.js'. The code includes imports for 'express', 'body-parser', 'mongoose', 'cors', and 'bcrypt'. It defines an Express application, sets up middleware for JSON and URL-encoded data, and connects to a MongoDB instance using Mongoose. A POST endpoint for '/register' is also defined.
- Terminal:** A terminal window at the bottom shows the command 'server > JS index.js > extended' and the output of the code execution.

```
server > JS index.js > extended
1  import express from 'express'
2  import bodyParser from 'body-parser';
3  import mongoose from 'mongoose';
4  import cors from 'cors';
5  import bcrypt from 'bcrypt';
6  import {Admin, Cart, Orders, Product, User } from './Schema.js'
7
8
9  const app = express();
10
11  app.use(express.json());
12  app.use(bodyParser.json({limit: "30mb", extended: true}))
13  app.use(bodyParser.urlencoded({limit: "30mb", extended: true}));
14  app.use(cors());
15
16  const PORT = 6001;
17
18  mongoose.connect('mongodb+srv://harishstranger24:Magesh007@cluster0.zo3qigm.mongodb.net/',{
19    useNewUrlParser: true,
20    useUnifiedTopology: true
21  }).then(()=>{
22
23    app.post('/register', async (req, res) => {
24      const { username, email, usertype, password } = req.body;
```

- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
  - User Management: CRUD operations for users.
  - Product Management: CRUD operations for Product.
  - Order Management: CRUD operations for orders.
  - Cart Management: CRUD operations for cart items.
  - Admin Management: Operations for managing categories.