Full Stack Development with MERN

Database Design and Development Report

Date	5 July 2024
Team ID	SWTID1720171853
Project Name	Project - Food Ordering System
Maximum Marks	

Project Title: - Food Ordering System

Date: 5 July 2024
Prepared by: Kishore Babu U

Objective

The objective of this report is to outline the database design and implementation details for the Food Ordering System 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, categories, promotedRestaurents, createdAt, updatedAt]

3. Restaurant

- Attributes: [list attributes like _id, ownerId, title, address, mainImg, menu(refrences Fooditem), createdAt, updatedAt]

4.FoodItem

- Attributes: [list attributes like   _id, title, descripition, itemIg, category, MenuCategory, restaurantId, price, discount, rating , createdAt, updatedAt]

5.Orders

- Attributes: [list attributes like _id, userId, name, email, mobile, address, pincode, restaurantId, restaurantName, foodItemId, foodItemName, FoodItemImg, quantity, price,discount, paymentMethod, orderdate, orderStatus, createdAt, updatedAt]

6.Cart

- Attributes: [list attributes like _id, userId, restaurentId, restaurantName, foodItemName, foodItemName, quantity, price, discount, createdAt, updatedAt]

Implement the Database using MongoDB

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

Database Name: Foodordering

1. Collection: users

  - Schema:

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

2. Collection: admin

  - Schema:

```
{
   categories: {type: Array},
   promotedRestaurants: []
}
```

3. Collection: restaurant

  - Schema:

~~~
    ```
{
    ownerId: {type: String},
    title: {type: String},
    address: {type: String},
    mainImg: {type: String},
    menu: {type: Array, default: []}
}
 ```
~~~

4. Collection: fooditem

  - Schema:

~~~
    ```
{
    title: {type: String},
    description: {type: String},
    itemImg: {type: String},
    category: {type: String}, //veg or non-veg or beverage
    menuCategory: {type: String},
    restaurantId: {type: String},
    price: {type: Number},
    discount: {type: Number},
    rating: {type: Number}
}
~~~

5. Collection: order

  - Schema:

~~~
    ```
{
~~~

```
   userId: {type: String},
   name: {type: String},
   email: {type: String},
   mobile: {type: String},
   address: {type: String},
   pincode: {type: String},
   restaurantId: {type: String},
   restaurantName: {type: String},
   foodItemId: {type: String},
   foodItemName: {type: String},
   foodItemImg: {type: String},
   quantity: {type: Number},
   price: {type: Number},
   discount: {type: Number},
   paymentMethod: {type: String},
   orderDate: {type: String},
   orderStatus: {type: String, default: 'order placed'}
}
```

6. Collection: cart

  - Schema:

```
    {
   userId: {type: String},
   restaurantId: {type: String},
   restaurantName: {type: String},
   foodItemId: {type: String},
   foodItemName: {type: String},
   foodItemImg: {type: String},
   quantity: {type: Number},
   price: {type: Number},
```

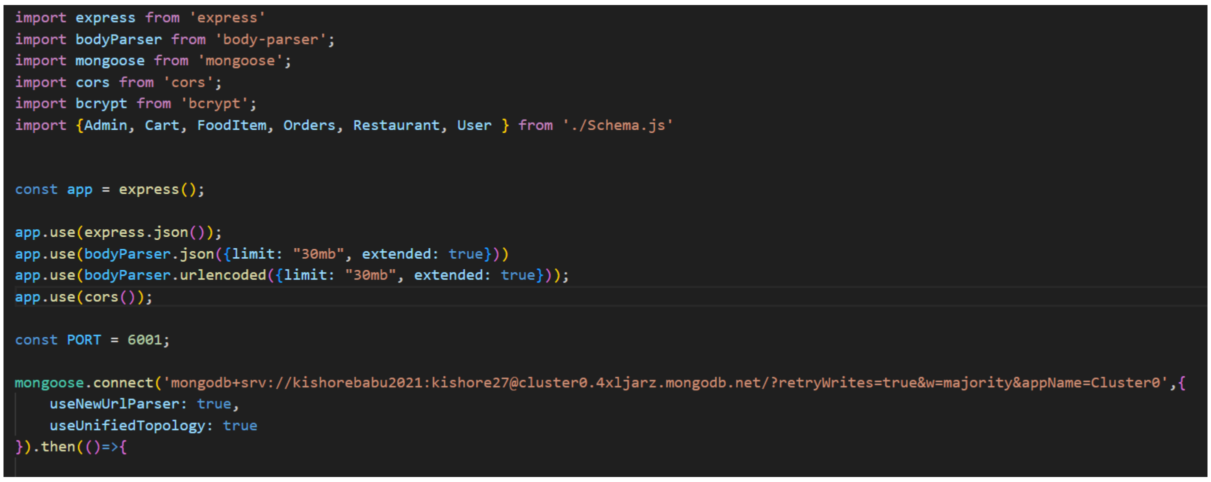
  discount: {type: Number}

}

```

## Integration with Backend

- Database connection: Screenshot of Database connection done using Mongoose



```
import express from 'express'
import bodyParser from 'body-parser';
import mongoose from 'mongoose';
import cors from 'cors';
import bcrypt from 'bcrypt';
import {Admin, Cart, FoodItem, Orders, Restaurant, User } from './Schema.js'

const app = express();

app.use(express.json());
app.use(bodyParser.json({limit: "30mb", extended: true}))
app.use(bodyParser.urlencoded({limit: "30mb", extended: true}));
app.use(cors());

const PORT = 6001;

mongoose.connect('mongodb+srv://kishorebabu2021:kishore27@cluster0.4xljarz.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0',{
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(()=>{
```

- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
  - User Management: CRUD operations for users.
  - Restaurant Management: CRUD operations for restaurants.
  - Food Item Management: CRUD operations for food items.
  - Order Management: CRUD operations for orders.
  - Cart Management: CRUD operations for cart items.
  - Admin Management: Operations for promoting restaurants and managing categories.
```