Full Stack Development with MERN

Database Design and Development Report

Date	19 July 2024
Team ID	SWTID1720150432
Project Name	EagerEats-Food Ordering App
Maximum Marks	

Project Title: Eager Eats

Date: 19 July 2024

Prepared by:

- Kishore S
- Sanjay R
- Praveen S
- Rishikeshwaran K R

Objective:

The objective of this report is to outline the database design and implementation details for the Eager Eats 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

Database Name: eagereatsmern

1. Collection: food_category

- **Attributes:**
  - _id: ObjectId
  - CategoryName: String

2. Collection: food_items

- **Attributes:**
  - _id: ObjectId

  - CategoryName: String
  - name: String
  - img: String
  - options: Array
  - description: String

3. Collection: orders

- **Attributes:**
  - _id: ObjectId
  - email: String
  - order_data: Array
    - Order_date: String
    - id: String
    - name: String
    - price: Number
    - qty: Number
    - size: String
    - img: String

4. Collection: users

- **Attributes:**
  - _id: ObjectId
  - name: String
  - location: String
  - email: String
  - password: String
  - date: Date

Implement the Database using MongoDB

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

Database Name: eagereatsmern

1. Collection: food_category

File: `foodCategory.js`

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const FoodCategorySchema = new Schema({
   CategoryName: {
      type: String,
      required: true
   }
});

module.exports = mongoose.model('FoodCategory', FoodCategorySchema);
```

2. Collection: food_items

File: `foodItems.js`

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const FoodItemSchema = new Schema({
   CategoryName: {
      type: String,
      required: true
   },
  name: {
    type: String,
    required: true
  },
  img: {
    type: String,
    required: true
  },
  options: {
    type: Array,
    required: true
  },
  description: {
    type: String,
    required: true
  }
});

module.exports = mongoose.model('FoodItem', FoodItemSchema);
```

3. Collection: orders

File: `order.js`

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const OrderSchema = new Schema({
  email: {
    type: String,
    required: true
```

```
},
order_data: {
    type: Array,
    required: true,
    items: {
        Order_date: {
            type: String,
            required: true
        },
        id: {
            type: String,
            required: true
        },
        name: {
            type: String,
            required: true
        },
        price: {
            type: Number,
            required: true
        },
        qty: {
            type: Number,
            required: true
        },
            size: {
                type: String,
                required: true
            },
            img: {
                type: String,
                required: true
            }
        }
    }
});

module.exports = mongoose.model('Order', OrderSchema);
```

4. Collection: users

File: `user.js`

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const UserSchema = new Schema({
    name: {
        type: String,
        required: true
    },
    location: {
        type: String,

        required: true

    },

    email: {

        type: String,

        required: true,

        unique: true

    },

    password: {

        type: String,

        required: true

    },

    date: {

        type: Date,

        default: Date.now

    }

});


module.exports = mongoose.model('User', UserSchema);
```

Integration with Backend

Database connection:

```
const mongoose = require('mongoose');

// const mongoURI =
'mongodb+srv://eagereats:6600@cluster0.jsndnco.mongodb.net/eagereatsmern?retry
Writes=true&w=majority&appName=Cluster0';
const mongoURI = 'mongodb://eagereats:6600@ac-xqyxgpk-shard-00-
00.jsndnco.mongodb.net:27017,ac-xqyxgpk-shard-00-
01.jsndnco.mongodb.net:27017,ac-xqyxgpk-shard-00-
02.jsndnco.mongodb.net:27017/eagereatsmern?replicaSet=atlas-xbbfoj-shard-
0&ssl=true&authSource=admin&retryWrites=true&w=majority&appName=Cluster0';

const mongoDB = async () => {
    try {
        await mongoose.connect(mongoURI);
        console.log("Connected to MongoDB");

        const fetched_data = await
mongoose.connection.db.collection("food_items");
        let data = await fetched_data.find({}).toArray();
        const foodCategory =await
mongoose.connection.db.collection("foodCategory");
        let catData =await foodCategory.find({}).toArray();

        global.food_items = data;
        global.foodCategory=catData;
    } catch (err) {
        console.log("---" + err);
    }
}

module.exports = mongoDB;
```

API Route Implementation:

1.User Management - `createuser.js:`

```
const express = require('express');

const router = express.Router();

const User = require('../models/user');

const { body, validationResult } = require('express-validator');

const jwt = require("jsonwebtoken");

const jwtSecret = "praveenmasterofcoding";

const bcrypt = require('bcryptjs');


router.post("/createuser", [

    body('email').isEmail(),

    body('name').isLength({ min: 5 }),

    body('password', "Incorrect Password").isLength({ min: 5
}).withMessage('Password must be at least 5 characters long')

], async (req, res) => {

    const errors = validationResult(req);

    if (!errors.isEmpty()) {

        return res.status(400).json({ errors: errors.array() });

    }

    const salt = await bcrypt.genSalt(10);

    let secPassword = await bcrypt.hash(req.body.password, salt);


    try {

        const newUser = await User.create({

            name: req.body.name,

            password: secPassword,

            email: req.body.email,

            location: req.body.location

        });

        res.json({ success: true, user: newUser });

    } catch (error) {

        console.error(error.message);

        res.status(500).json({ success: false, message: 'Server Error' });

    }
});


router.post("/loginuser", [
    body('email').isEmail(),
    body('password', "Incorrect Password").isLength({ min: 5
}).withMessage('Password must be at least 5 characters long')
], async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(400).json({ errors: errors.array() });
    }
    let email = req.body.email;
    try {
        let userData = await User.findOne({ email });
        if (!userData) {
            return res.status(400).json({ errors: "Try logging with correct
credentials" });
        }
        const pwdCompare = await bcrypt.compare(req.body.password,
userData.password);
        if (!pwdCompare) {
            return res.status(400).json({ errors: "Try logging with correct
credentials" });
        }
        const data = {
            user: {
                id: userData.id
            }
        };
        const authToken = jwt.sign(data, jwtSecret);
        return res.json({ success: true, authToken: authToken });
    } catch (error) {
        console.error(error.message);
        res.status(500).json({ success: false, message: 'Server Error' });
    }
});
```

```
module.exports = router;
```

2.Display Data - `displaydata.js`

```
const express = require('express');
const router = express.Router();

router.post('/foodData', (req, res) => {
    try {
        console.log(global.food_items);
        res.send({ food_items: global.food_items, foodCategory: 
global.foodCategory });
    } catch (error) {
        console.log(error.message);
        res.send("Server error");
    }
});

module.exports = router;
```

3. **Order Data** - `orderdata.js:`

```
const express = require('express');
const router = express.Router();
const Order = require('../models/Orders');

router.post('/orderData', async (req, res) => {
    let data = req.body.order_data;
    await data.splice(0, 0, { Order_date: req.body.order_date });

    let eId = await Order.findOne({ 'email': req.body.email });
    console.log(eId);

    if (eId === null) {
        try {
            await Order.create({
                email: req.body.email,
                order_data: [data]
            }).then(() => {
                res.json({ success: true });
            });
        } catch (error) {
            console.log(error.message);
            res.status(500).send("Server Error: " + error.message);
        }
    } else {
        try {
            await Order.findOneAndUpdate(
                { email: req.body.email },
                { $push: { order_data: data } }
            ).then(() => {
                res.json({ success: true });
            });
        } catch (error) {
            console.log(error.message);
            res.status(500).send("Server Error: " + error.message);
        }
    }
});

router.post('/myorderData', async (req, res) => {
    try {
        let myData = await Order.findOne({ 'email': req.body.email });
        res.json({ orderData: myData });
    } catch (error) {
        res.send("Server Error: " + error.message);
    }
});
module.exports = router;
```