

Full Stack Development with MERN

Database Design and Development Report

Date	16th July 2024
Team ID	SWTID1720076571
Project Name	Project - Food Ordering System
Maximum Marks	

Project Title: Food Ordering System

Date: 16th July 2024

Prepared by: Sankalp Sharma

Objective

The objective of this report is to outline the database design and implementation details for the [Your Project Title] 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: `_id`, `email`, `password`, `username`, `userType`

2. Food Items

- Attributes: `_id`, `itemName`, `itemCategory`, `itemDescription`, `itemPrice`, `itemPhoto`

3. Carts

- Attributes: **_id**, **userId** (references **User**), **items** (array of **foodItem** (references **FoodItem**), **quantity**)

Orders

- Attributes: **_id**, **userId** (references **User**), **items** (array of **foodItem** (references **FoodItem**), **quantity**), **paymentStatus**, **createdAt**, **updatedAt**

Restaurants

- Attributes: **_id**, **restaurantName**, **ownerName**, **email**, **phone**, **address**, **description**, **restaurantPicture**, **approved**, **rejected**, **submissionTimestamp**

Implement the Database using MongoDB

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

Database Name: Food Delivery

1. Collection: users

-{ **_id**: ObjectId, **email**: String, **password**: String, **username**: String, **userType**: String, **createdAt**: Date, **updatedAt**: Date }

2. Collection Food Items

{ **_id**: ObjectId, **itemName**: String, **itemCategory**: String, **itemDescription**: String, **itemPrice**: Number, **itemPhoto**: String, **createdAt**: Date, **updatedAt**: Date }

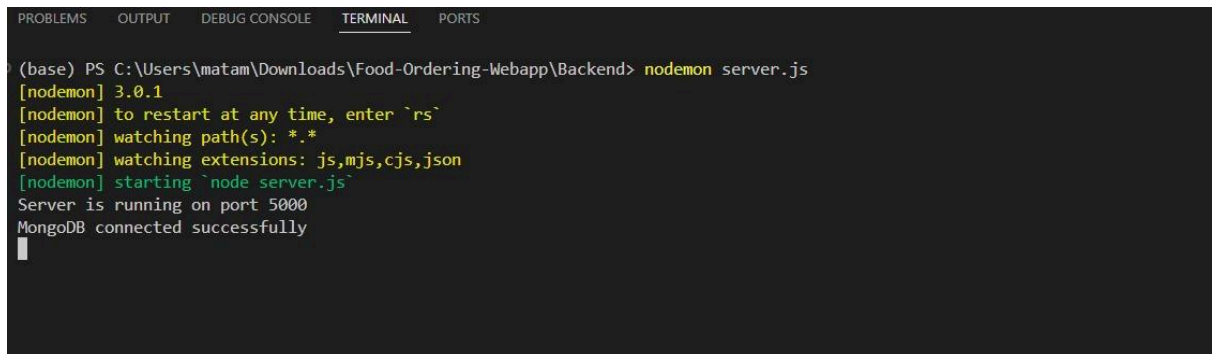
3. Collection Carts

{ **_id**: ObjectId, **userId**: ObjectId (references users), **items**: [{ **foodItem**: ObjectId (references foodItems), **quantity**: Number }], **createdAt**: Date, **updatedAt**: Date }

4. { **_id**: ObjectId, **userId**: ObjectId (references users), **items**: [{ **foodItem**: ObjectId (references foodItems), **quantity**: Number }], **paymentStatus**: Boolean, **createdAt**: Date, **updatedAt**: Date }

5. { _id: ObjectId, restaurantName: String, ownerName: String, email: String, phone: String, address: String, description: String, restaurantPicture: String, approved: Boolean, rejected: Boolean, submissionTimestamp: Date, createdAt: Date, updatedAt: Date }

Integration with Backend



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(base) PS C:\Users\matam\Downloads\Food-Ordering-Webapp\Backend> nodemon server.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on port 5000
MongoDB connected successfully
```

- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
 - **User Management:** CRUD operations for users.
 - **Food Item Management:** CRUD operations for food items.
 - **Cart Management:** CRUD operations for carts, linked to user and food items.
 - **Order Management:** CRUD operations for orders, with user authentication.
 - **Restaurant Management:** CRUD operations for restaurant profiles and approval/rejection status.