Catering order management (backend)

Overview

This documentation provides a comprehensive guide to the backend of a Order management MERN (MongoDB, Express.js, React.js, Node.js) stack application. The backend serves as the core of the application, handling data processing, server-side logic, and communication with the database.

Technologies Used

- **MongoDB**: A NoSQL database used to store application data.
- **Express.js**: A web application framework for Node.js used to build the RESTful API.
- **Node.js**: A JavaScript runtime environment used to execute server-side code.
- **Mongoose**: An Object Data Modeling (ODM) library for MongoDB and Node.js, used for schema validation and interaction with MongoDB.

Project Structure

```
/backend
 |-- config
 |   |-- .env                    # Environment Variables
 |   |-- dbConnection.js    # Configuration settings (e.g., database URI)
 |
 |-- controllers
 |   |-- AuthController.js               # Controllers for user-related operations
 |   |-- CustomerController.js        # controllers for customer details
 |   |-- LightController.js              # Controller for light related ordered
 |   |-- TentController.js              # Controller for tent related ordered
 |   |-- CateringController.js        # Controller for catering related ordered
 |   |-- BistarController.js            # Controller for Bistar related ordered
 |   |-- DecorationController.js    # Controller for decoration ordered
 |
 |-- models
 |   |-- authModel.js              # auth model
 |   |-- customerModel.js      # customer model
 |   |-- catringModel.js          # catering model
 |   |-- tentModel.js              # tent model
 |   |-- decorationModel.js    # decoration model
 |   |-- bistarModel.js           # bistar model
 |   |-- lightModel.js             # light model
 |
 |-- routes
 |   |-- authRoutes.js      # API routes for user-related endpoints
 |   |-- customerRoutes.js     # API routes for customer-related endpoints
 |   |-- cateringRoutes.js       # API routes for catering-related endpoints
 |   |-- tentRoutes.js              # API routes for tent-related endpoints
 |   |-- lightRoutes.js            # API routes for light-related endpoints
 |   |-- decorationRoutes.js  # API routes for decoration-related endpoints
 |   |-- bistarRoutes.js          # API routes for bistar-related endpoints
 |
 |-- middleware
 |   |-- auth.js   # Middleware for authentication
 |
 |-- utils
 |   |-- feature.js  # Utility functions
 |
 |-- app.js               # connection and API start point of the backend application
 |-- index.js             # Entry point of the backend application
```

Setup

1. **Install Dependencies**: Run **npm install** to install all required dependencies listed in package.json.
2. **Environment Configuration**: Set up environment variables, such as the MongoDB connection URI, port number, and any other necessary configurations. This can be done in a .env file.
3. **Database Configuration**: Configure MongoDB connection in config.js file located in the config directory.
4. **Run the Server**: Execute npm start to start the backend server. By default, the server will run on the specified port.

API Endpoint

Customer

1. **'POST /api/v1/customer/new': create new entry of customer.**
2. **'GET /api/v1/customer/all: Retrieve all customer details**
3. **PUT/api/v1/customer/:id: Update a customer details**
4. **DELETE /api/v1/users/:id: Delete a customer details**

Catering

1. **'POST /api/v1/catering/new': create new entry of catring.**
2. **'GET /api/v1/catering/all: Retrieve all catering details**
3. **PUT/api/v1/catering/:id: Update a catering ordered details**

Tent

1. **'POST /api/v1/tent/new': create new entry of tent.**
2. **'GET /api/v1/tent/all: Retrieve all tent details**
3. **PUT/api/v1/tent/:id: Update a tent details**

Bister

1. **‘POST /api/v1/bistar/new’: create new entry of bistar order.**
2. **‘GET /api/v1/bistar/all: Retrieve all bistar order details**
3. **PUT/api/v1/bistar/:id: Update a Bistar details**

Light

1. **‘POST /api/v1/light/new’: create new entry of light order.**
2. **‘GET /api/v1/light/all: Retrieve all light order details**
3. **PUT/api/v1/light/:id: Update a light details**

Decoration

1. **‘POST /api/v1/deco/new’: create new entry of decoration order.**
2. **‘GET /api/v1/deco/all: Retrieve all decoration order details**
3. **PUT/api/v1/deco/:id: Update a decoration details**

Authentication

The backend supports authentication using JSON Web Tokens (JWT).

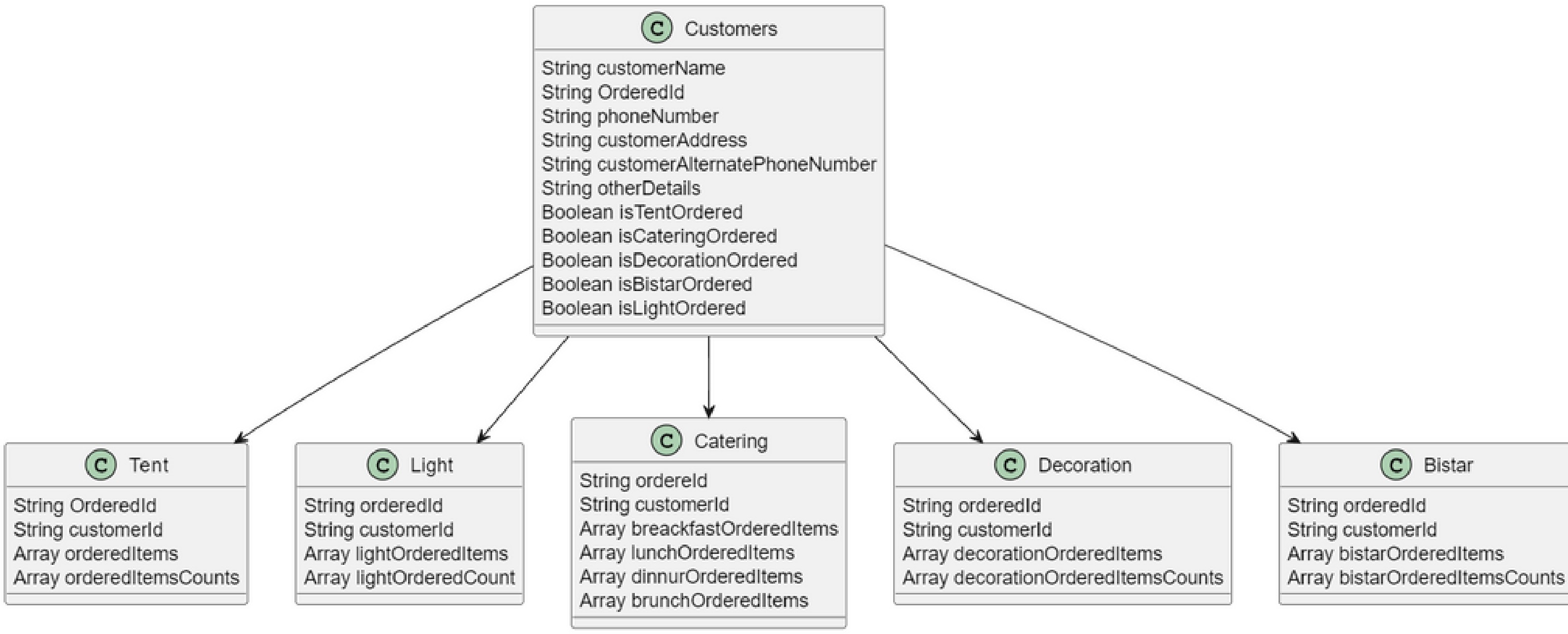
- Signup: Users can sign up by providing required details such as username, email, and password. Passwords are hashed before storing in the database.
- Login: Users can log in using their credentials. Upon successful authentication, a JWT token is generated and returned to the client.
- Authorization: Protected routes are guarded by middleware that verifies the JWT token. Users must include the token in the request header to access protected routes.

Middleware

Custom middleware functions are used for various purposes, such as authentication, error handling, and request validation.

- Authentication Middleware: Verifies JWT tokens and authenticates users.

Data Model Relationship



Conclusion

This documentation provides a thorough understanding of the backend architecture, setup, API endpoints, authentication mechanism, middleware, and data models used in the MERN stack application. Developers can refer to this documentation for development, maintenance, and troubleshooting purposes.