|  |
| --- |
| **Team ID :** LTVIP2025TMID57320 |
| **Project Name :** OrderOnTheGo: Your On-Demand Food Ordering Solution |

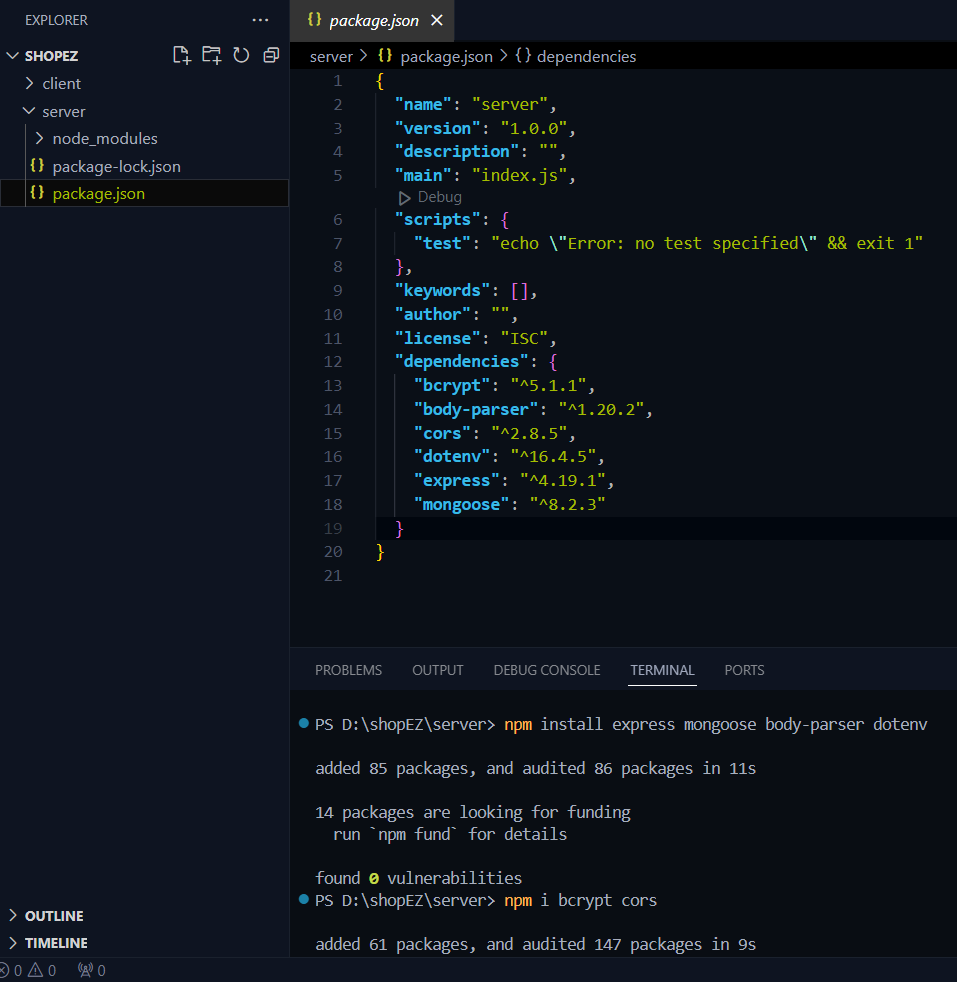
**Backend Development**

**Set Up Project Structure:**

• Create a new directory for your project and set up a package.json file using the npm init command.

• Install necessary dependencies such as Express.js, Mongoose, and other required packages.

                    Reference Image:



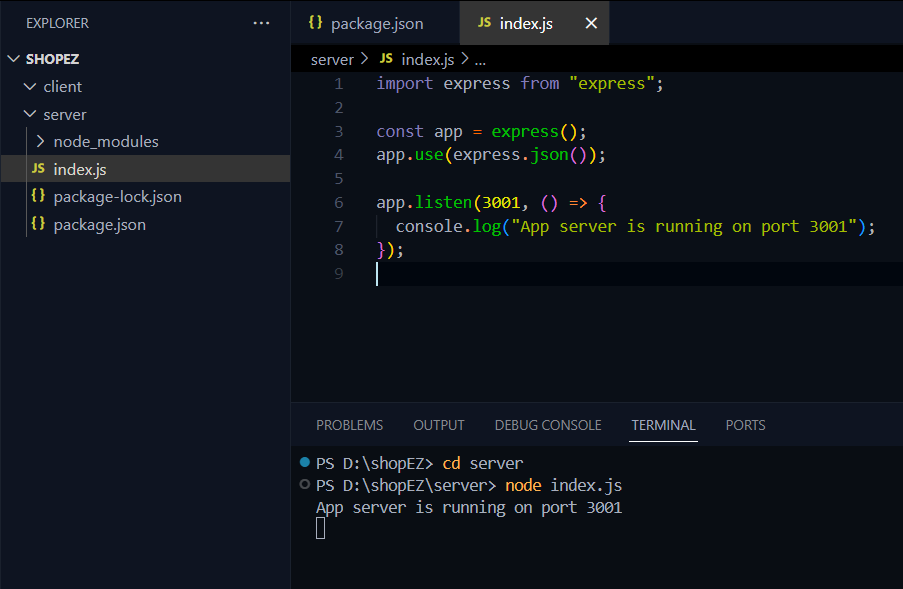
**1. Setup express server:**

• Create index.js file.

• Create an express server on your desired port number.

• Define API’s

Reference Image:



**2. Database Configuration:**

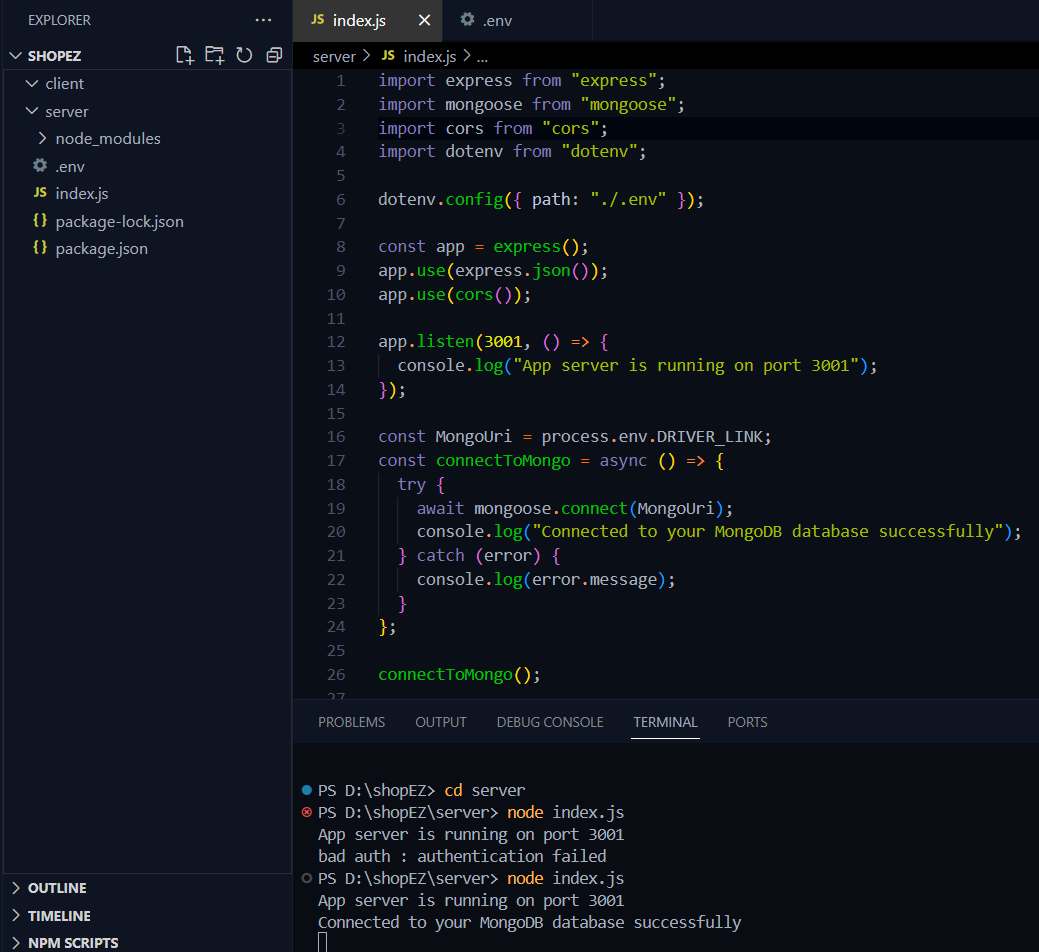
• Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas or use locally with MongoDB compass.

• Create a database and define the necessary collections for admin, users,  restaurants, food products, orders,and other relevant data.

Reference Video of connect node with mongoDB database: <https://drive.google.com/file/d/1cTS3_-EOAAvDctkibG5zVikrTdmoY2Ag/view?usp=sharing>

Reference Article: <https://www.mongodb.com/docs/atlas/tutorial/connect-to-your-cluster/>

       Reference Image:



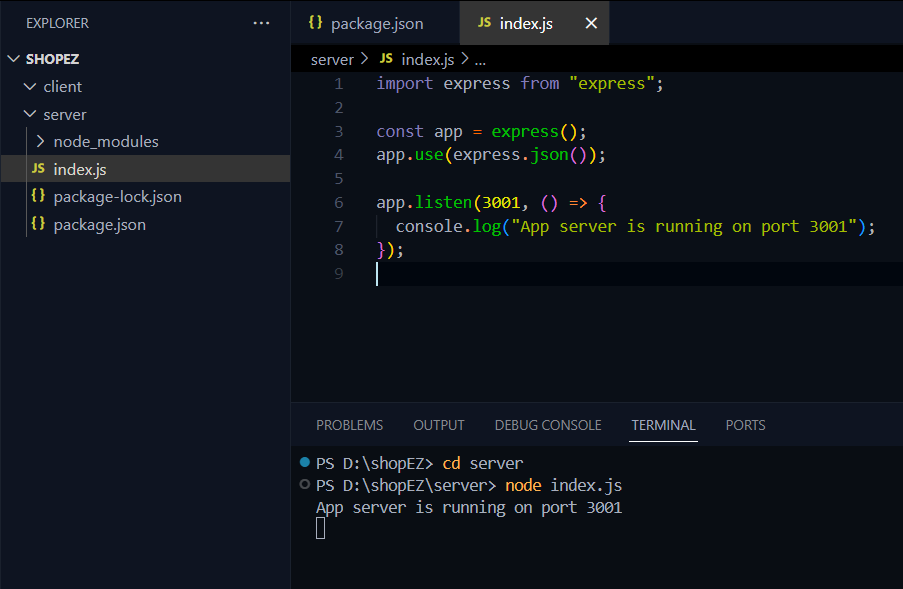
**3. Create Express.js Server:**

• Set up an Express.js server to handle HTTP requests and serve API endpoints.

 • Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

Reference Video: <https://drive.google.com/file/d/1-uKMIcrok_ROHyZl2vRORggrYRio2qXS/view?usp=sharing>

Reference Image:



**4. Define API Routes:**

• Create separate route files for different API functionalities such as users, orders, and authentication.

• Define the necessary routes for listing products, handling user registration and  login,managing orders, etc.

• Implement route handlers using Express.js to handle requests and interact with the database.

**5. Implement Data Models:**

• Define Mongoose schemas for the different data entities like products, users,  and orders.

• Create corresponding Mongoose models to interact with the MongoDB database.

 • Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

**6. User Authentication:**

• Create routes and middleware for user registration, login, and logout.

• Set up authentication middleware to protect routes that require user authentication.

**7. Handle new products and Orders:**

• Create routes and controllers to handle new product listings, including fetching products data from the database and sending it as a response.

• Implement ordering(buy) functionality by creating routes and controllers to  handle order requests, including validation and database updates.

**8. Admin Functionality:**

• Implement routes and controllers specific to admin functionalities such as adding products, managing user orders, etc.

• Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.

**9. Error Handling:**

• Implement error handling middleware to catch and handle any errors that occur during the API requests.

• Return appropriate error responses with relevant error messages and HTTP status codes.