🖥️ ShopSmart – Server (Backend) Explanation

The **server** of ShopSmart is built using **Node.js** and **Express.js**. It handles the business logic, connects to the database (MongoDB), processes API requests, manages users/products/orders, and ensures security through authentication and error handling.

⚙️ Main Responsibilities of the Server

1. **Handle HTTP Requests**
    - The Express server receives client requests and routes them to the correct controller functions (e.g., for products, users, orders).
2. **Business Logic**
    - The logic for creating accounts, logging in, adding items, updating inventory, and processing orders is written in controller files.
3. **Database Interaction**
    - The server uses **Mongoose** to interact with MongoDB. It performs CRUD operations (Create, Read, Update, Delete) on collections like users, products, and orders.
4. **Authentication & Authorization**
    - Uses **JWT (JSON Web Tokens)** to protect routes (e.g., only logged-in users can place orders, only admins can delete products).
5. **Error Handling**
    - A centralized error middleware captures and handles all errors gracefully.

□ Important Components

Folder/File	Purpose
/server.js	Entry point – sets up Express app, connects to MongoDB, starts server
/config/db.js	MongoDB connection file
/models/	Mongoose models for Products, Users, Orders
/routes/	Defines API endpoints like /api/users, /api/products
/controllers/	Functions that implement the logic behind each route
/middleware/	Authentication and error handling logic
.env	Stores sensitive variables like database URI and JWT secret

---

🔐 Security Features

- **JWT Authentication**: Secures routes and validates users
- **Bcrypt**: Hashes user passwords before storing
- **Helmet & CORS**: Protects server from common attacks and allows safe cross-origin requests

---

🔁 Request Flow Example

text

CopyEdit

Client → /api/products → productRoutes → productController → MongoDB → Response