Project Structure & Microservices Design for Grocery App

🧱 Microservices-Oriented Folder Structure

```
/grocery-app
│
├── api-gateway/          # Central routing and entry point
│   ├── routes/           # Routes for forwarding requests
│   ├── middleware/       # Auth checks, logging, etc.
│   ├── utils/            # Proxy helpers, error handlers, etc.
│   └── server.js         # Bootstraps the API Gateway
│
├── auth-service/         # Handles user registration & login
│   ├── controllers/
│   ├── routes/
│   ├── models/
│   └── server.js
│
├── order-service/        # Handles grocery orders
│   ├── controllers/
│   ├── routes/
│   ├── models/
│   └── server.js
│
├── docker-compose.yml    # (Optional) Compose file to run all services
└── README.md             # Project overview and setup instructions
```

🌐 API Gateway Responsibilities

- Handles **PUT, GET, POST, DELETE** requests from the client
- Uses `routes/` to forward requests to the proper service (auth or order)
- Middleware like authentication, logging, and rate limiting is handled here
- `utils/` can include service URL mapping and centralized error handling
- `server.js` sets up the Express server and mounts everything

✅ Summary

Using microservices allows us to:

- **Decouple** authentication and order logic
- Enable **scalability** and **independent deployments**
- Improve **code maintainability** and testing

Next step: scaffold the `api-gateway/` with base files.