NestJS Project Full Flow Documentation

1. Bootstrapping & Global Setup

main.ts

- Creates Nest app (from AppModule).
- Adds Helmet, CORS, and ValidationPipe globally.
- Starts server.

app.module.ts

- Imports ConfigModule (.env), Mongoose connection, Throttler (rate limit).
- Imports AuthModule and BookModule.

2. Auth Feature (JWT + Roles)

auth.module.ts

- Configures Passport (JWT), JwtModule (with async secret).
- Registers User schema (Mongoose).

user.schema.ts

- Fields: name, email, password, role[] (default User).

DTOs

- SignupDto, LoginDto with class-validator.

auth.controller.ts

- POST /auth/signup → AuthService.signUp()
- GET /auth/login → AuthService.login()

auth.service.ts

- signUp: hash password, save, return token.
- login: check email/password, return token.

jwt.strategy.ts

- Reads Bearer token, validates with secret.
- Finds user by id, attaches to req.user.

Roles system

- @Roles decorator, RolesGuard checks req.user.role.

3. Book Feature

book.schema.ts

- Fields: title, author, description, price, category (enum), user (ref to User).

DTOs

- CreateBookDto (all required, user forbidden).
- updateBookDto (all optional, user forbidden).

book.service.ts

- findAll: pagination + search.
- create: assign req.user._id.
- findById: validate id, return or throw.
- updateById / deleteById.

book.controller.ts

- GET /books → AuthGuard + RolesGuard (Admin/Moderator only).
- POST /books/new → AuthGuard (any user).
- GET /books/:id, PUT, DELETE → public in current code.

4. Request Flow Diagram

Client → Middleware (helmet, cors, pipes) → Guards (AuthGuard, RolesGuard) → Controller → Service → Mongoose → MongoDB → Response

5. Sequence Diagrams

Signup

Client → AuthController → AuthService → bcrypt → UserModel → JwtService → Client (token).

Login

Client → AuthController → AuthService → UserModel → bcrypt → JwtService → Client (token).

Create Book

Client (with Bearer token) → AuthGuard (jwt) → JwtStrategy → UserModel → Controller → Service → BookModel → MongoDB → Response.