**Spring Boot JWT Auth Backend: Explanation of Each File and Purpose**

**1. Main Application Class**

**File:** JwtauthNagaApplication.java  
**Purpose:** Entry point for the Spring Boot application. Automatically configures beans and runs the application.

**2. Controller Layer**

**File:** AuthController.java  
**Purpose:**

* Handles HTTP POST requests for /signup and /login
* Accepts request body data (JSON), validates, and sends to the service layer
* Returns JWT token or error response

**3. DTOs (Data Transfer Objects)**

**Files:** SignupRequest.java, LoginRequest.java  
**Purpose:**

* Represent incoming request payloads for signup and login
* Used to decouple the internal model (User) from external input

**4. Model**

**File:** User.java  
**Purpose:**

* Represents a user in the database
* Mapped to a table using JPA annotations

**5. Repository**

**File:** UserRepository.java  
**Purpose:**

* Interface extending JpaRepository
* Provides CRUD operations on the User entity
* Allows querying users by email

**6. Service Layer**

**File:** JwtService.java  
**Purpose:**

* Responsible for generating, parsing, and validating JWT tokens
* Encapsulates JWT logic (e.g., signing key, claims, expiration)

**7. Security Configuration**

**Files:**

* SecurityConfig.java
* JwtFilter.java
* AuthProvider.java

**Purpose:**

* **SecurityConfig:** Configures Spring Security (paths, CORS, CSRF, filters)
* **JwtFilter:** Intercepts requests, checks Authorization header, validates token
* **AuthProvider:** Handles authentication based on the token and user details

**8. Exception Handling (Optional)**

**File:** CustomExceptionHandler.java (if used)  
**Purpose:**

* Centralized place to catch and handle exceptions thrown by the API

**9. Configuration File**

**File:** application.properties or application.yml  
**Purpose:**

* Configure database connection, server port, JWT expiration, etc.

**10. Build Tool Configuration**

**File:** pom.xml (for Maven) or build.gradle (for Gradle)  
**Purpose:**

* Manage dependencies like Spring Boot Starter Web, Security, JPA, jjwt, MySQL driver

**Summary**

Each file in the backend has a clearly defined role:

* **Controller**: Handles requests
* **DTOs**: Receive data
* **Model/Entity**: Represents database
* **Repository**: Database operations
* **Service**: Business logic (e.g., JWT)
* **Security**: Auth and filtering
* **Config**: App-wide settings
* **Build Tool**: Manage project dependencies

**Authentication & Authorization Explained:**

* **Authentication** happens in:
  + AuthController.login(...) — verifies email/password and issues a JWT if credentials are valid.
  + JwtService.generateToken(...) — creates a signed JWT token using user info.
* **Authorization** happens in:
  + JwtFilter.doFilterInternal(...) — intercepts requests, validates JWT, sets authentication in Spring Security context.
  + SecurityConfig — restricts access to protected endpoints based on roles and authenticated users.

**Project Structure and File Purpose:**

**JwtauthNagaApplication.java**

* **Purpose:** Entry point of the Spring Boot application.
* **Why Needed:** Bootstraps the app using @SpringBootApplication.

**controller/AuthController.java**

* **Purpose:** Handles /signup and /login API endpoints.
* **Why Needed:** Exposes auth endpoints for user registration and login.

**dto/SignupRequest.java & LoginRequest.java**

* **Purpose:** DTO classes to encapsulate user input data for signup and login.
* **Why Needed:** Cleanly separates request body mapping from domain model (User).

**model/User.java**

* **Purpose:** Entity class mapped to the user table in the database.
* **Why Needed:** Represents authenticated users, used in security and DB persistence.

**repository/UserRepository.java**

* **Purpose:** Extends JpaRepository to allow CRUD operations on User entity.
* **Why Needed:** To fetch/save user data from/to database.

**service/JwtService.java**

* **Purpose:** Generates, parses, and validates JWT tokens.
* **Why Needed:** Core utility for JWT-based authentication.

**service/UserService.java (if present)**

* **Purpose:** Implements UserDetailsService for Spring Security.
* **Why Needed:** Provides user detail fetching logic for authentication.

**filter/JwtFilter.java**

* **Purpose:** Intercepts HTTP requests, extracts JWT, validates it, and sets authentication.
* **Why Needed:** Ensures only authenticated users can access secured endpoints.

**config/SecurityConfig.java**

* **Purpose:** Defines Spring Security rules (which endpoints are secured, JWT filter setup, etc).
* **Why Needed:** Customizes and secures the web application endpoints.

**application.properties**

* **Purpose:** Holds configuration such as DB credentials and server port.
* **Why Needed:** Allows externalizing config values for different environments.