JWT Authentication Implementation (File-wise Order)

This document lists the exact files to create in sequence, what to implement inside each file, and why it is required.

File Name	What to Implement	Why We Do It
application.yml	Add port, DB config, eureka config, jwt secret, jwt expiration	JWT needs secret key + DB is needed to store users.
entity/Role.java	Create enum: ADMIN, CUSTOMER	Role is required for authorization.
entity/User.java	Create entity fields: id, name, email, password, role	User table is required for login/register.
repository/UserRepository.java	Add findByEmail(), existsByEmail()	Login/register works using email.
dto/RegisterRequestDto.java	Add fields: name, email, password	Register API should take input in DTO not entity.
dto/LoginRequestDto.java	Add fields: email, password	Login request should be validated and clean.
dto/AuthResponseDto.java	Add fields: token, email	Login response must return JWT token.
payload/ApiResponse.java	Create common response format: success, message, data	All APIs should return standard response.
security/jwt/JwtService.java	Implement generateToken(), extractUsername(), validateToken()	Handles all JWT creation and validation.
security/jwt/CustomUserDetailsService.java	Implement loadUserByUsername(email) from DB	Spring Security needs user details during authentication.
security/jwt/JwtAuthenticationFilter.java	Read Authorization header, extract token, validate, set SecurityContext	Protects every request using JWT.
security/SecurityConfig.java	Configure PasswordEncoder, AuthenticationManager, Stateless session, Permit /api/auth/**, Add JWT filter	Main Spring Security configuration file.
service/services/AuthService.java	Add methods: register(), login()	Service interface keeps code clean and scalable.
service/serviceImpl/AuthServiceImpl.java	Implement register (save encrypted password) and login (authenticate + generate JWT)	Actual business logic should be inside service layer.
controller/AuthController.java	Create POST /register and POST /login endpoints	Entry point for user authentication.
controller/UserController.java	Create GET /api/users/profile (protected API)	Confirms JWT protection is working.

Final Result:

- /api/auth/register works without token
- /api/auth/login returns JWT token
- All protected APIs require Authorization: Bearer <token>