**JWT Authentication in Spring Boot – Step-by-Step Implementation**

**Step 1: Create a Spring Boot Project**

Use Spring Initializr or your IDE (like Spring Tool Suite or IntelliJ) to create a Maven-based Spring Boot project with the following dependencies:

* spring-boot-starter-web
* spring-boot-starter-security
* jjwt-api
* jjwt-impl
* jjwt-jackson

**Sample pom.xml Dependencies**

xml

CopyEdit

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

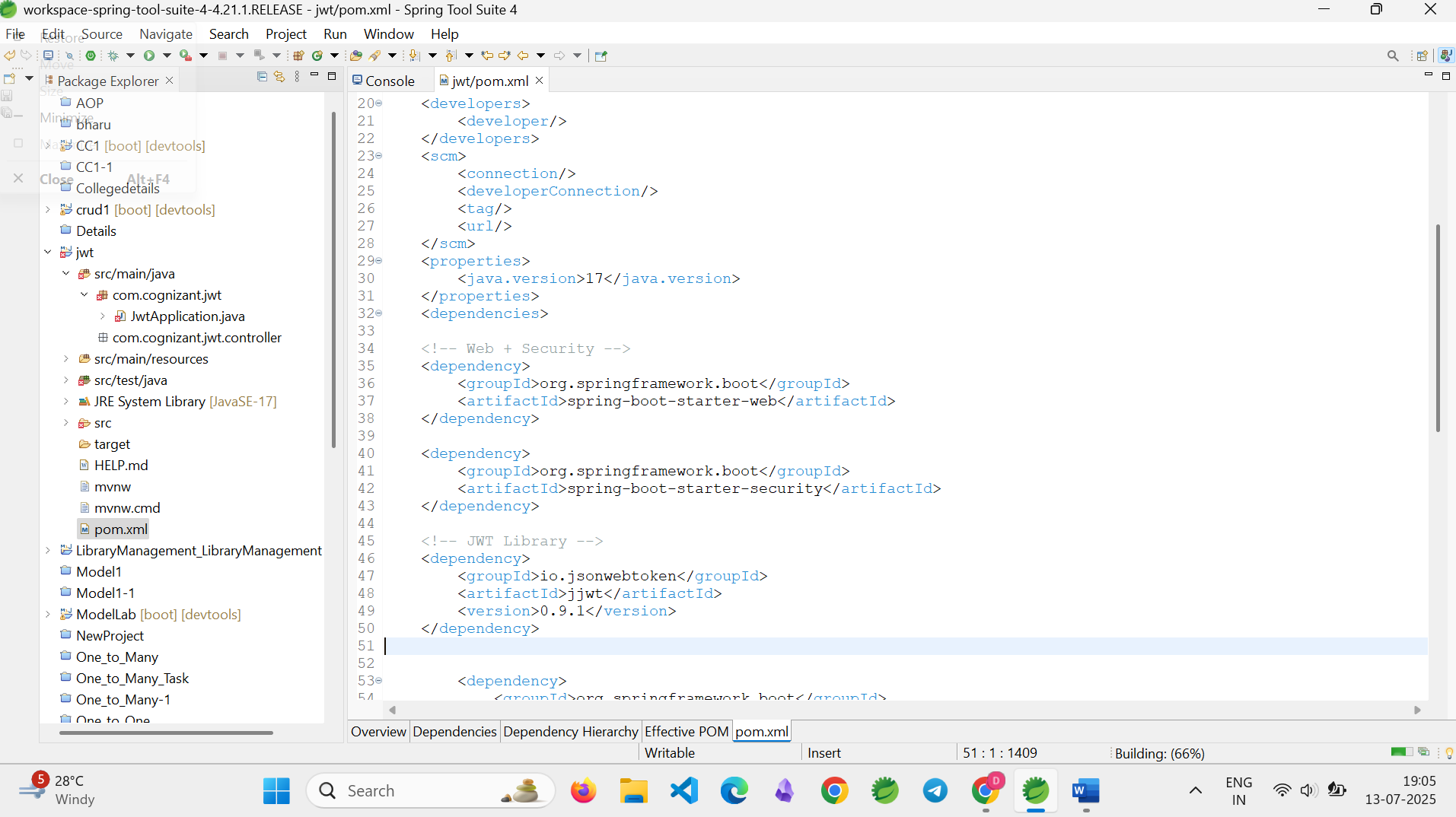
<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

</dependencies>



**Step 2: Configure Application Port**

Open src/main/resources/application.properties and add the following line:

server.port=8091

This sets the application to run on port 8091.

**Step 3: Create the Main Application Class**

Create a main class named JwtApplication.java in the base package:

@SpringBootApplication

public class JwtApplication {

public static void main(String[] args) {

SpringApplication.run(JwtApplication.class, args);

}

}

**Step 4: Create the JWT Service**

Create a class JwtService.java under com.cognizant.jwt.service.

@Service

public class JwtService {

private final String secretKey = "myverysecuresecretkey123456789012";

public String[] extractCredentials(String authHeader) {

if (authHeader != null && authHeader.startsWith("Basic ")) {

String base64Credentials = authHeader.substring("Basic ".length());

byte[] decoded = Base64.getDecoder().decode(base64Credentials);

String credentials = new String(decoded);

return credentials.split(":", 2);

}

throw new RuntimeException("Missing or invalid Authorization header");

}

public String generateToken(String username) {

long now = System.currentTimeMillis();

long expiry = now + (10 \* 60 \* 1000);

Key key = Keys.hmacShaKeyFor(secretKey.getBytes());

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(now))

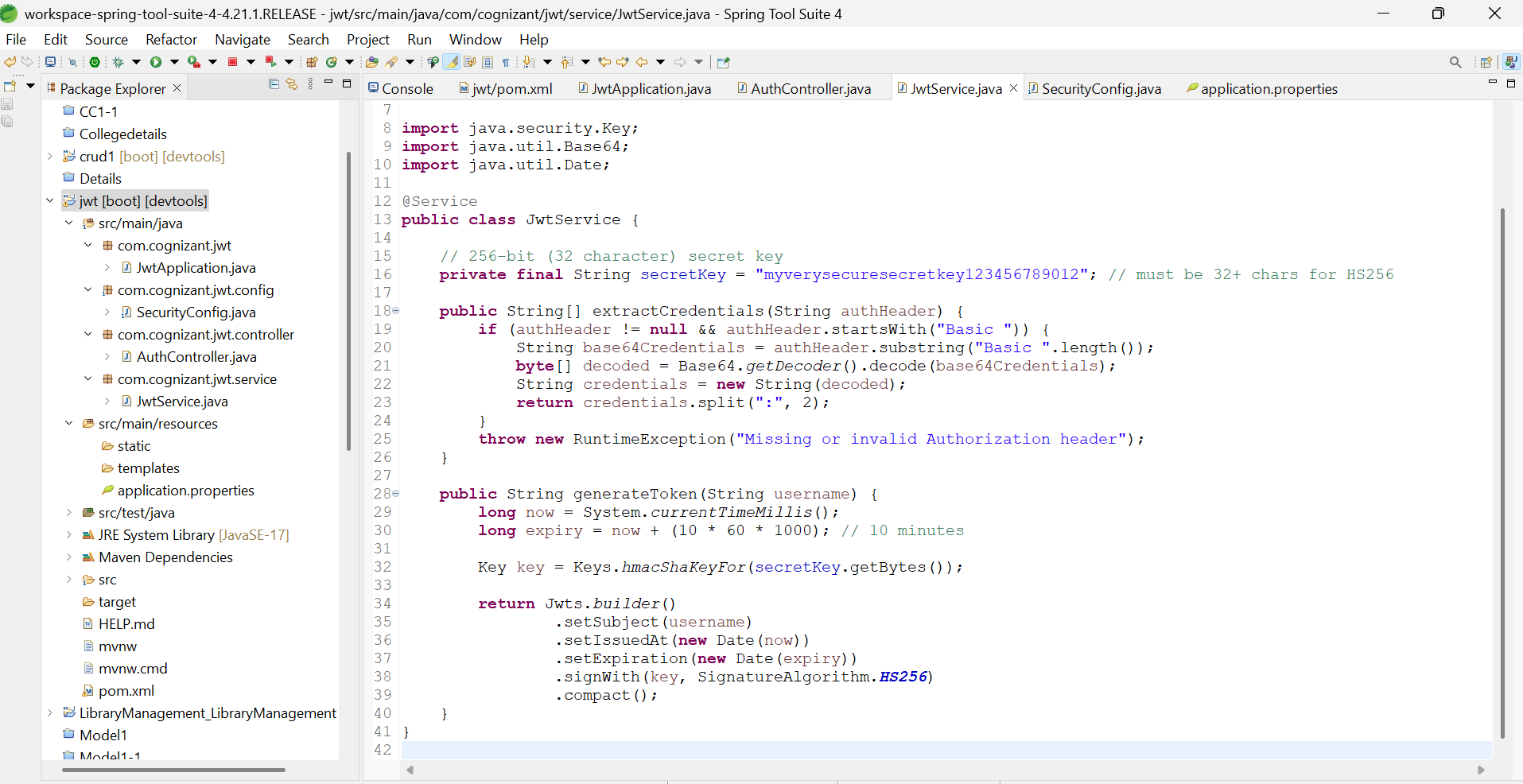
.setExpiration(new Date(expiry))

.signWith(key, SignatureAlgorithm.HS256)

.compact();

}

}



**Step 5: Create the Authentication Controller**

Create a class named AuthController.java under com.cognizant.jwt.controller.

@RestController

public class AuthController {

@Autowired

private JwtService jwtService;

@GetMapping("/authenticate")

public ResponseEntity<Map<String, String>> authenticate(@RequestHeader("Authorization") String authHeader) {

try {

String[] creds = jwtService.extractCredentials(authHeader);

String username = creds[0];

String password = creds[1];

if ("user".equals(username) && "pwd".equals(password)) {

String token = jwtService.generateToken(username);

return ResponseEntity.ok(Collections.singletonMap("token", token));

} else {

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();

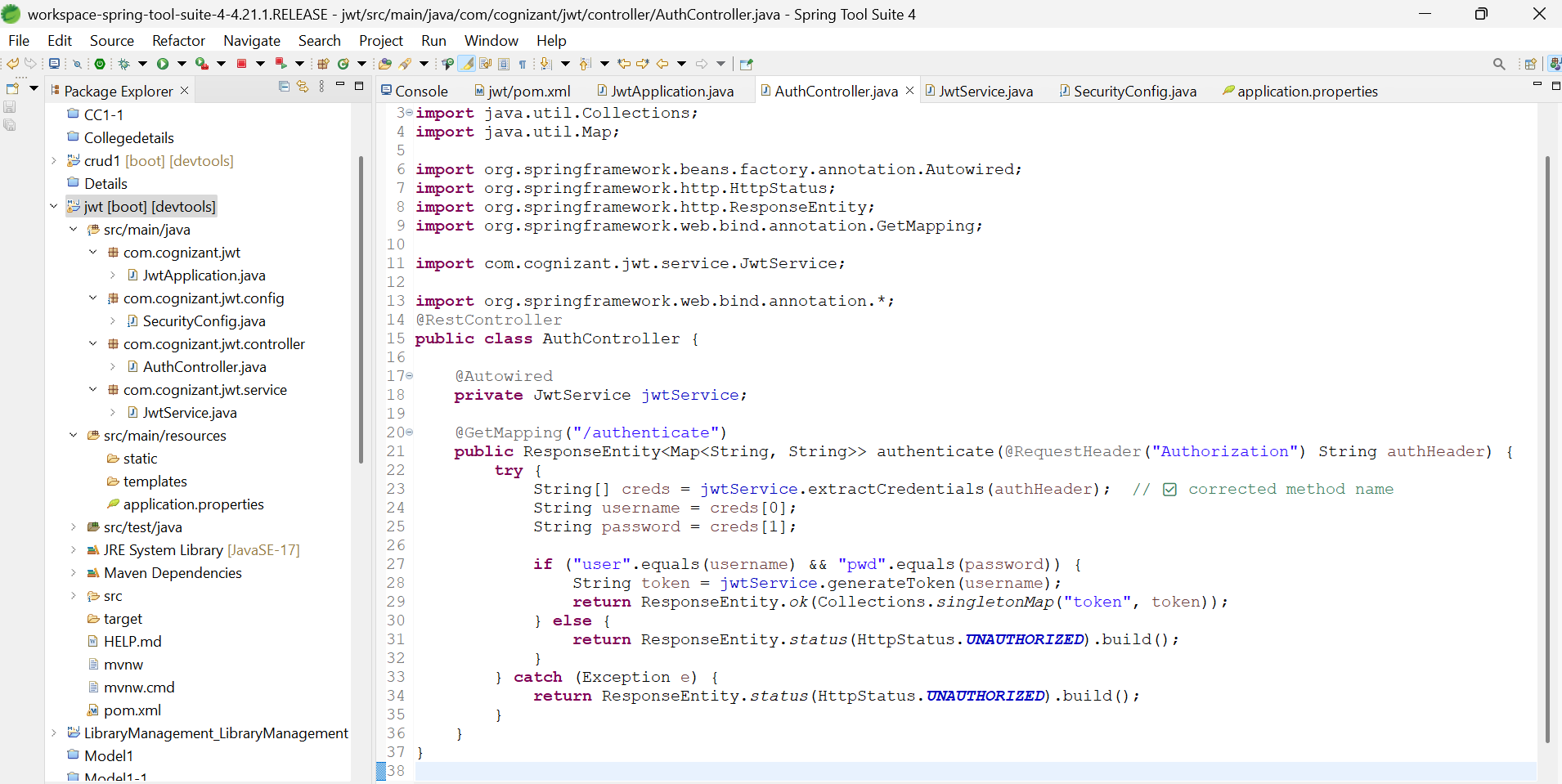
}

} catch (Exception e) {

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();

}

}

}

**Step 6: Configure Spring Security**

Create a configuration class SecurityConfig.java under com.cognizant.jwt.config.

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated());

return http.build();

}

@Bean

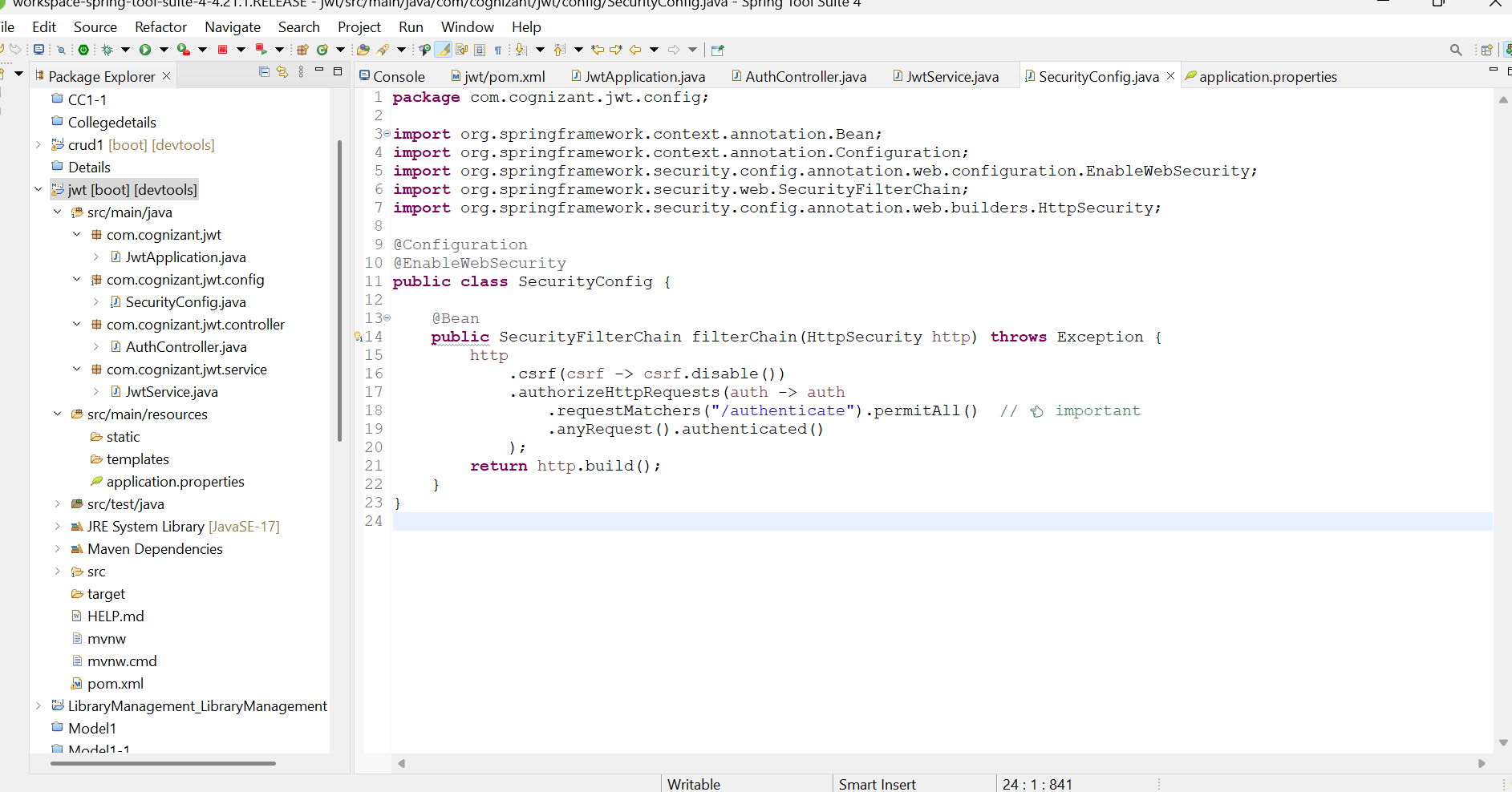
public UserDetailsService userDetailsService() {

return username -> null;

}

}

This disables CSRF, allows access to the /authenticate endpoint without authentication, and disables Spring Boot’s default user authentication.



**Step 7: Test the JWT Authentication Endpoint**

Use curl or Postman to test the /authenticate endpoint.

curl -u user:pwd http://localhost:8091/authenticate

Expected Output:

{

"token": "eyJhbGciOiJIUzI1NiJ9..."

}

Postman:

* Method: GET
* URL: <http://localhost:8091/authenticate>
* Authorization tab: Basic Auth (username = user, password = pwd)

If password wrong

401 error Unauthorized

