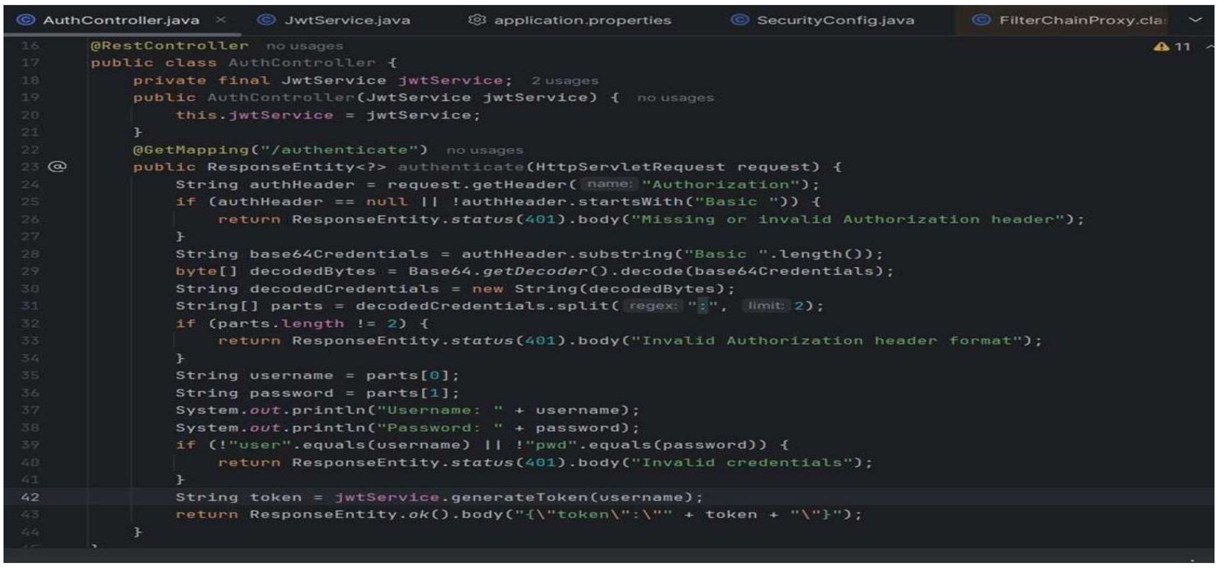
Week 4

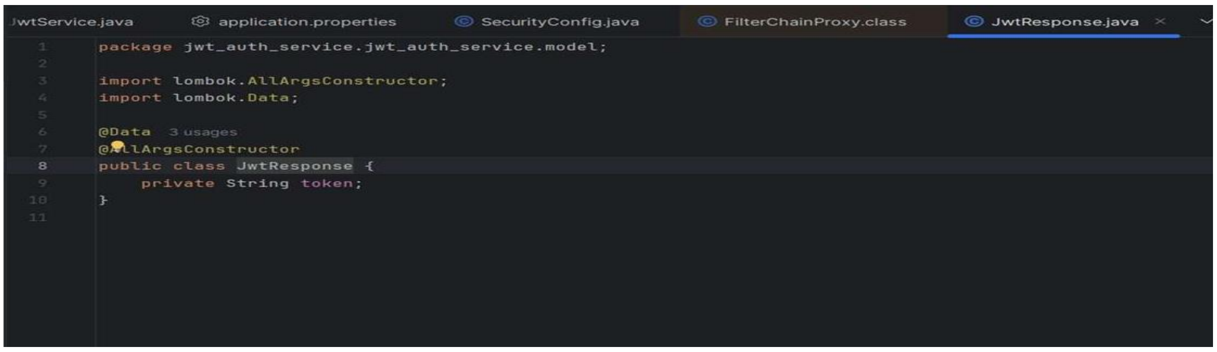
Create authentication service that returns JWT

controller



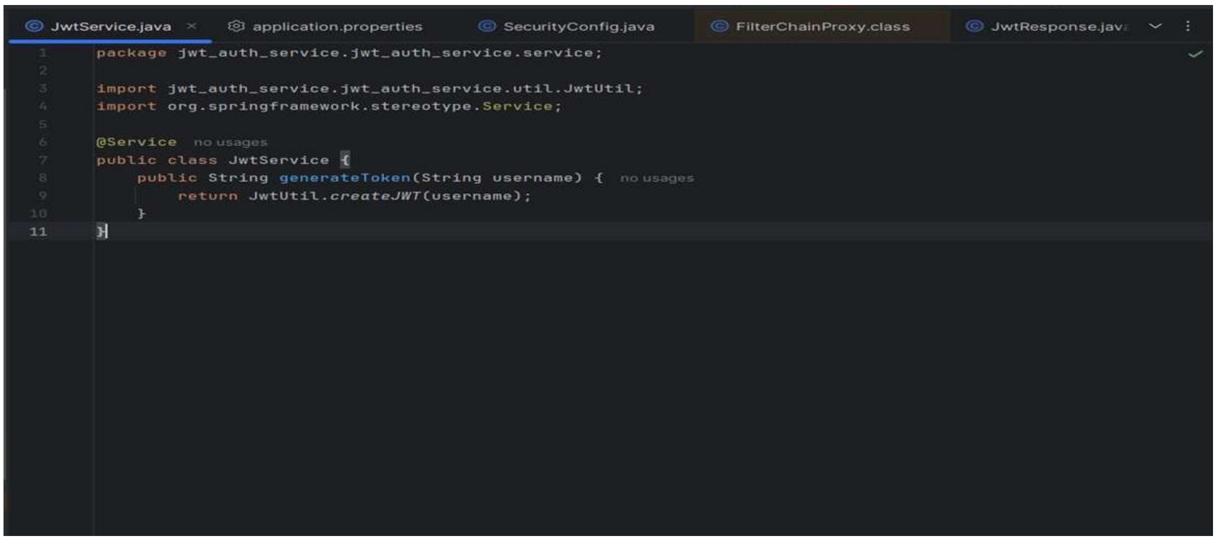
```java
@RestController
public class AuthController {
    private final JwtService jwtService;
    public AuthController(JwtService jwtService) {
        this.jwtService = jwtService;
    }
    @GetMapping("/authenticate")
    public ResponseEntity<?> authenticate(HttpServletRequest request) {
        String authHeader = request.getHeader("Authorization");
        if (authHeader == null || !authHeader.startsWith("Basic ")) {
            return ResponseEntity.status(401).body("Missing or invalid Authorization header");
        }
        String base64Credentials = authHeader.substring("Basic ".length());
        byte[] decodedBytes = Base64.getDecoder().decode(base64Credentials);
        String decodedCredentials = new String(decodedBytes);
        String[] parts = decodedCredentials.split(":", 2);
        if (parts.length != 2) {
            return ResponseEntity.status(401).body("Invalid Authorization header format");
        }
        String username = parts[0];
        String password = parts[1];
        System.out.println("Username: " + username);
        System.out.println("Password: " + password);
        if (!"user".equals(username) || !"pwd".equals(password)) {
            return ResponseEntity.status(401).body("Invalid credentials");
        }
        String token = jwtService.generateToken(username);
        return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");
    }
```

Model



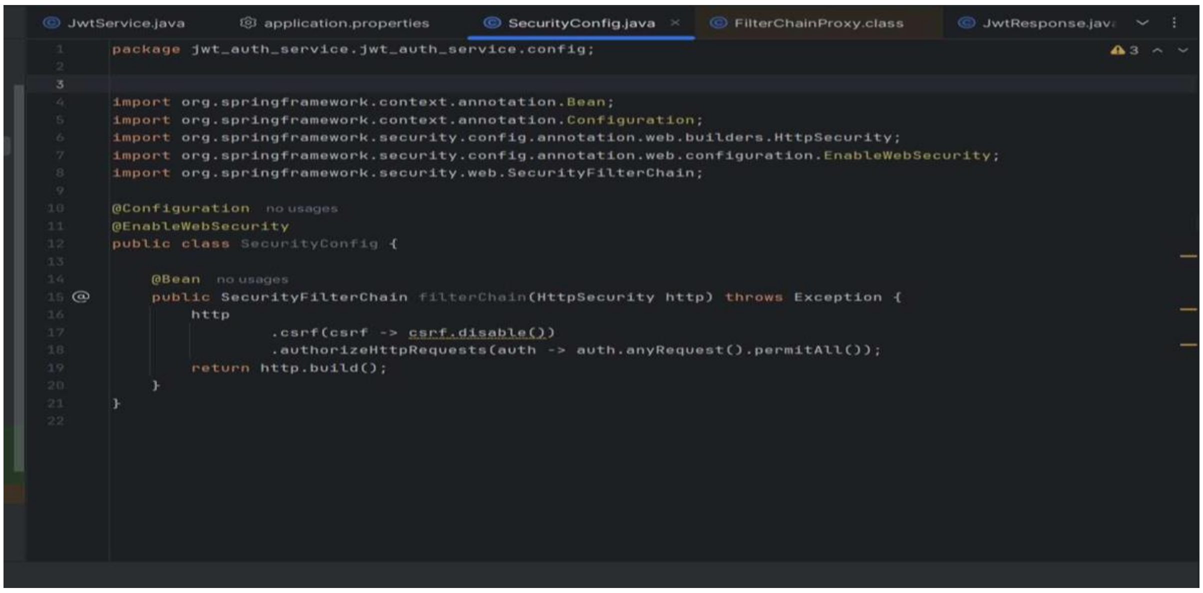
```java
package jwt_auth_service.jwt_auth_service.model;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class JwtResponse {
    private String token;
}
```

Service



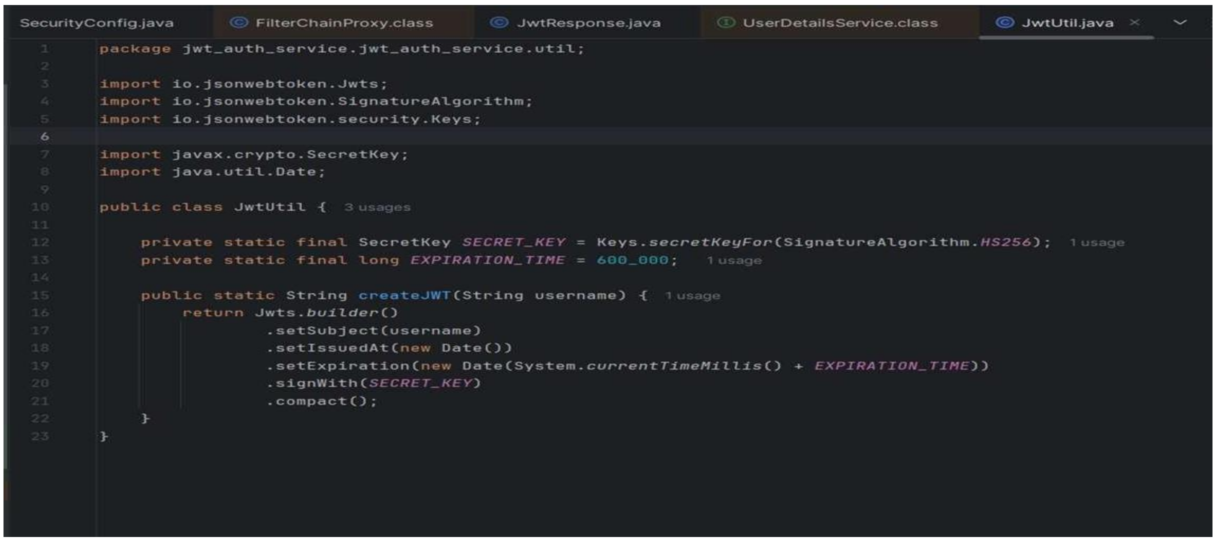
```java
package jwt_auth_service.jwt_auth_service.service;

import jwt_auth_service.jwt_auth_service.util.JwtUtil;
import org.springframework.stereotype.Service;

@Service
public class JwtService {
    public String generateToken(String username) {
        return JwtUtil.createJWT(username);
    }
}
```

SecurityConfig



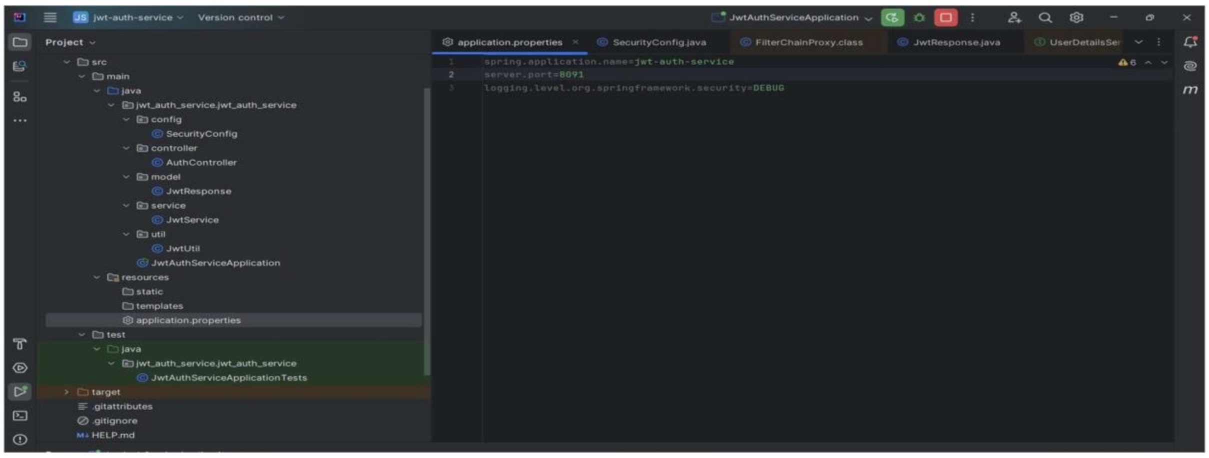
```java
package jwt_auth_service.jwt_auth_service.config;


import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
                .csrf(csrf -> csrf.disable())
                .authorizeHttpRequests(auth -> auth.anyRequest().permitAll());
        return http.build();
    }
}
```

JwtUtils



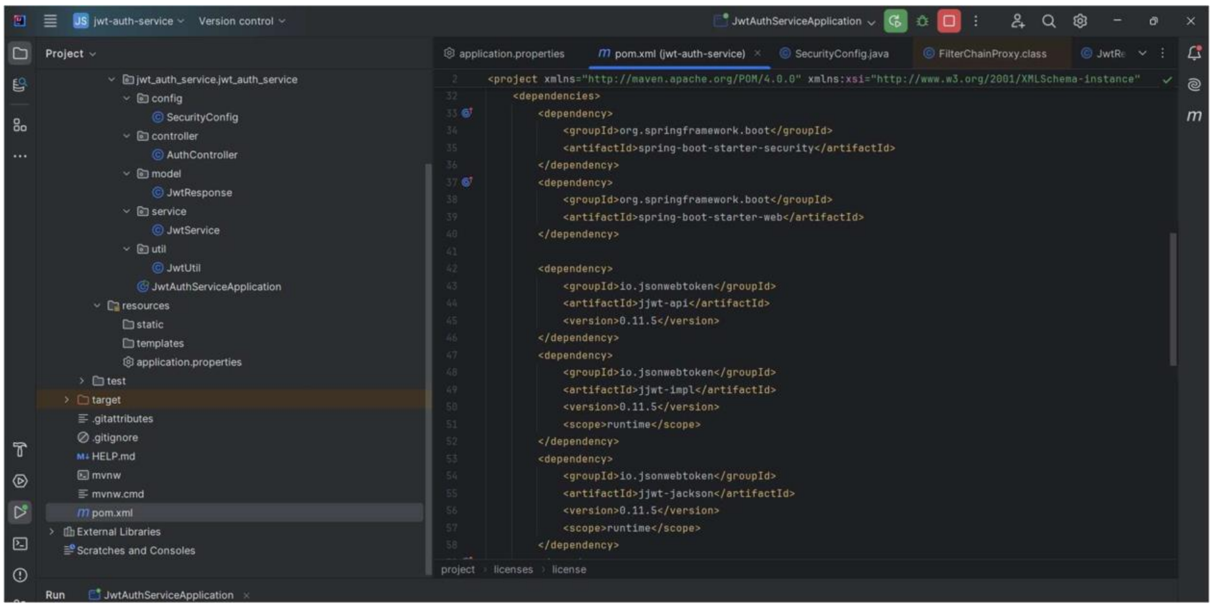
```java
package jwt_auth_service.jwt_auth_service.util;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;

import javax.crypto.SecretKey;
import java.util.Date;

public class JwtUtil {

    private static final SecretKey SECRET_KEY = Keys.secretKeyFor(SignatureAlgorithm.HS256);
    private static final long EXPIRATION_TIME = 600_000;

    public static String createJWT(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + EXPIRATION_TIME))
                .signWith(SECRET_KEY)
                .compact();
    }
}
```

Application.property



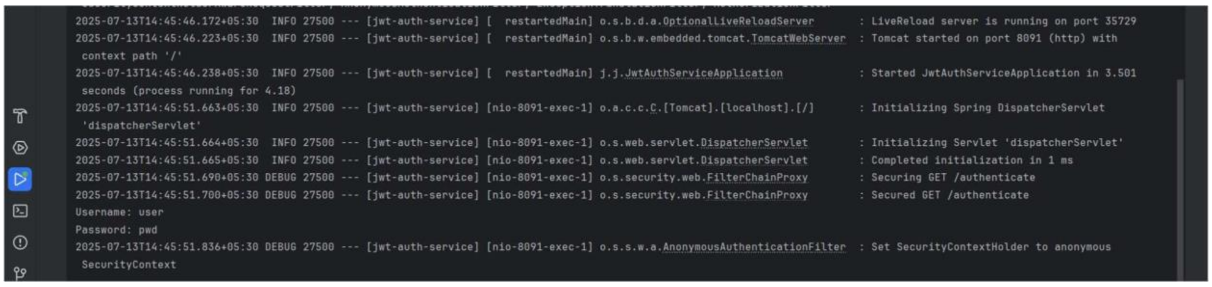
```
spring.application.name=jwt-auth-service
server.port=8091
logging.level.org.springframework.security=DEBUG
```

Pom.xml



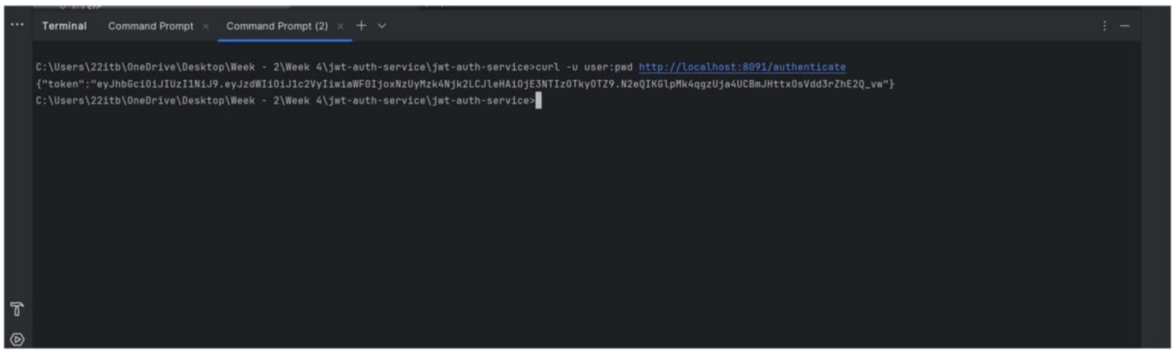
```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</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>
```

Console Output



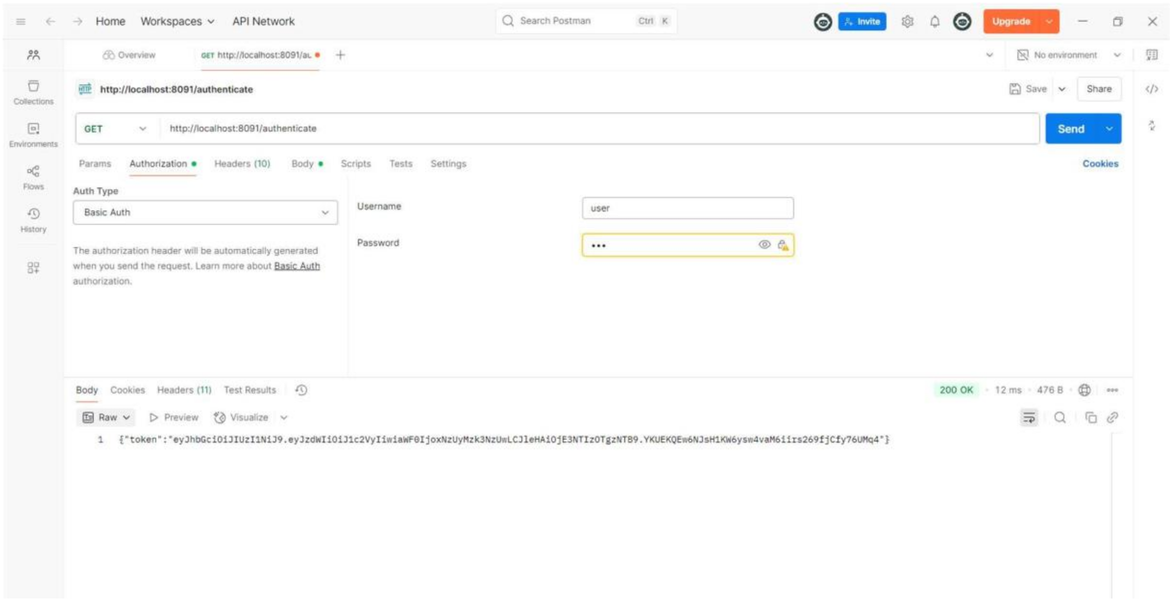
```
2025-07-13T14:45:46.172+05:30  INFO 27500 --- [jwt-auth-service] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2025-07-13T14:45:46.223+05:30  INFO 27500 --- [jwt-auth-service] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8091 (http) with
 context path '/'
2025-07-13T14:45:46.238+05:30  INFO 27500 --- [jwt-auth-service] [  restartedMain] j.j.JwtAuthServiceApplication            : Started JwtAuthServiceApplication in 3.501
 seconds (process running for 4.18)
2025-07-13T14:45:51.663+05:30  INFO 27500 --- [jwt-auth-service] [nio-8091-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet
 'dispatcherServlet'
2025-07-13T14:45:51.664+05:30  INFO 27500 --- [jwt-auth-service] [nio-8091-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-13T14:45:51.665+05:30  INFO 27500 --- [jwt-auth-service] [nio-8091-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2025-07-13T14:45:51.690+05:30 DEBUG 27500 --- [jwt-auth-service] [nio-8091-exec-1] o.s.security.web.FilterChainProxy        : Securing GET /authenticate
2025-07-13T14:45:51.700+05:30 DEBUG 27500 --- [jwt-auth-service] [nio-8091-exec-1] o.s.security.web.FilterChainProxy        : Secured GET /authenticate
Username: user
Password: pwd
2025-07-13T14:45:51.836+05:30 DEBUG 27500 --- [jwt-auth-service] [nio-8091-exec-1] o.s.s.w.a.AnonymousAuthenticationFilter  : Set SecurityContextHolder to anonymous
 SecurityContext
```

Token Output



```
C:\Users\22itb\OneDrive\Desktop\Week - 2\Week 4\jwt-auth-service\jwt-auth-service>curl -u user:pwd http://localhost:8091/authenticate
{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzUyMzk4Njk2LCJleHAiOjE3NTIzOTkyOTZ9.N2eQIKGlpMk4qgzUja4UCBmJHttxOsVdd3rZhE2Q_vw"}
C:\Users\22itb\OneDrive\Desktop\Week - 2\Week 4\jwt-auth-service\jwt-auth-service>
```

Postman output



http://localhost:8091/authenticate

GET http://localhost:8091/authenticate

Auth Type: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about Basic Auth authorization.

Username: user

Password: •••

Body | Cookies | Headers (11) | Test Results — 200 OK · 12 ms · 476 B

Raw

```
{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzUyMzk3NzUwLCJleHAiOjE3NTIzOTgzNTB9.YKUEKQEw6NJsH1KW6ysw4vaM6iirs269fjCfy76UMq4"}
```