**Create authentication service that returns JWT**

**Theory & Steps:**

1. What is JWT?
   * JSON Web Token is a compact, URL-safe token used for authentication and authorization.
2. How Basic Auth is handled?
   * Credentials are sent as Authorization: Basic <base64(username:password)>.
   * Server decodes and validates them.
3. JWT Generation:
   * Uses io.jsonwebtoken.Jwts to create a token with subject, issuedAt, and expiration.
   * Signed using a secret key with HS256 algorithm.
4. Security Config:
   * Disabled default form login.
   * Allowed unauthenticated access to /authenticate.
5. How to test:
   * Via browser (with a plugin) or curl.
   * Example: curl -u user:pwd <http://localhost:8090/authenticate>.

**STEP 1: Project Setup**

1. Create a Spring Boot Maven project.
2. Add the required dependencies in pom.xml:

<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</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

**STEP 2: Create AuthenticationController.java**

package com.example.jwt.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

import java.util.Date;

@RestController

public class AuthenticationController {

private static final String SECRET\_KEY = "mySecretKey";

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

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

// Decode base64 username:password

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

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

String credentials = new String(decodedBytes);

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

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

// Create JWT Token

String token = Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

return ResponseEntity.ok("{\"token\":\"" + token + "\"}");

} else {

return ResponseEntity.status(401).body("Invalid Credentials");

}

} else {

return ResponseEntity.status(400).body("Missing Authorization Header");

}

}

}

**STEP 3: Configure Security (Disable default login prompt)**

Create SecurityConfig.java:

package com.example.jwt.config;

import org.springframework.context.annotation.Bean;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests()

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

.anyRequest().authenticated()

.and()

.httpBasic(); // Enables Basic Auth for /authenticate

return http.build();

}

}

**STEP 4: Run and Test**

**Test using CURL:**

curl -s -u user:pwd <http://localhost:8090/authenticate>

**Expected Output:**

{"token":"<JWT\_TOKEN>"}