**Create authentication service that returns JWT**

**Dependencies (pom.xml)**

Ensure you have:

xml

CopyEdit

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<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>

**Create JwtUtil.java**

Utility to generate JWT tokens.

**Path:** src/main/java/com/cognizant/springlearn/util/JwtUtil.java

package com.cognizant.springlearn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secretKey = "secret"; // Use secure key in production

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60)) // 1 hour validity

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

}

**Create AuthenticationController.java**

**Path:** src/main/java/com/cognizant/springlearn/controller/AuthenticationController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.util.JwtUtil;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpHeaders;

import org.springframework.http.ResponseEntity;

import org.springframework.util.StringUtils;

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

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

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@RequestMapping("/authenticate")

public ResponseEntity<Map<String, String>> authenticate(org.springframework.http.HttpEntity<String> httpEntity) {

String authHeader = httpEntity.getHeaders().getFirst(HttpHeaders.AUTHORIZATION);

Map<String, String> response = new HashMap<>();

if (StringUtils.hasText(authHeader) && authHeader.startsWith("Basic ")) {

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

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

String decodedCredentials = new String(decodedBytes, StandardCharsets.UTF\_8);

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

String username = credentials[0];

String password = credentials[1];

// Basic hardcoded validation; replace with DB/user service check in production

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

String token = jwtUtil.generateToken(username);

response.put("token", token);

return ResponseEntity.ok(response);

} else {

response.put("error", "Invalid credentials");

return ResponseEntity.status(401).body(response);

}

} else {

response.put("error", "Authorization header missing or invalid");

return ResponseEntity.status(400).body(response);

}

}

}

**Configure SecurityConfig.java**

**Path:** src/main/java/com/cognizant/springlearn/config/SecurityConfig.java

java

CopyEdit

package com.cognizant.springlearn.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.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

.and()

.httpBasic(); // For future secured endpoints if needed

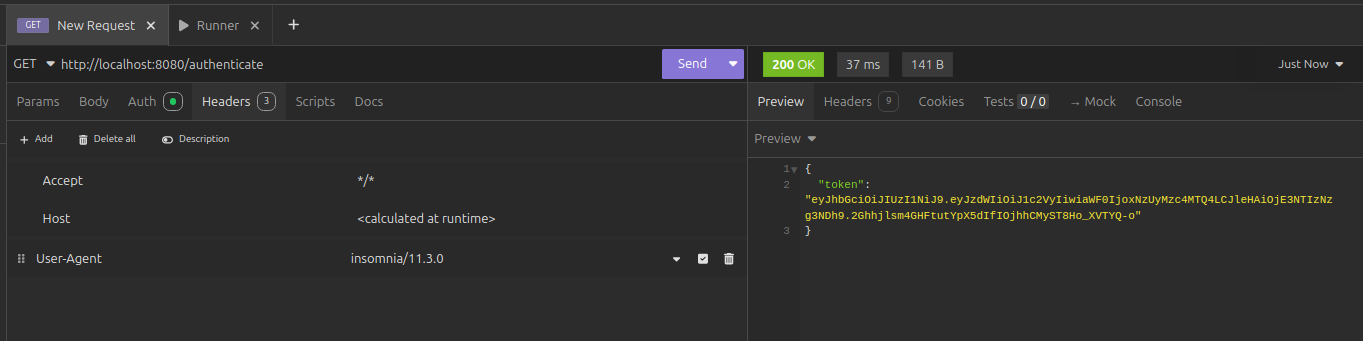
return http.build();

}

}

**Test with curl:** curl -s -u user:pwd http://localhost:8090/authenticate

**OUTPUT:**

****

Top of Form

Bottom of Form