**Create authentication service that returns JWT**

1. **Create authentication controller and configure it in SecurityConfig**

**Code –**

*AuthController.java –*

package com.cognizant.jwttest.controller;

import org.springframework.web.bind.annotation.\*;

*@RestController*

public class AuthController {

*@PostMapping*("/authenticate")

public String authenticate() {

return "Authentication endpoint hit";

}

}

*SecurityConfig.java –*

package com.cognizant.jwttest.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 securityFilterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/authenticate").permitAll() // Allow this endpoint

.anyRequest().authenticated() // Secure all others

.and()

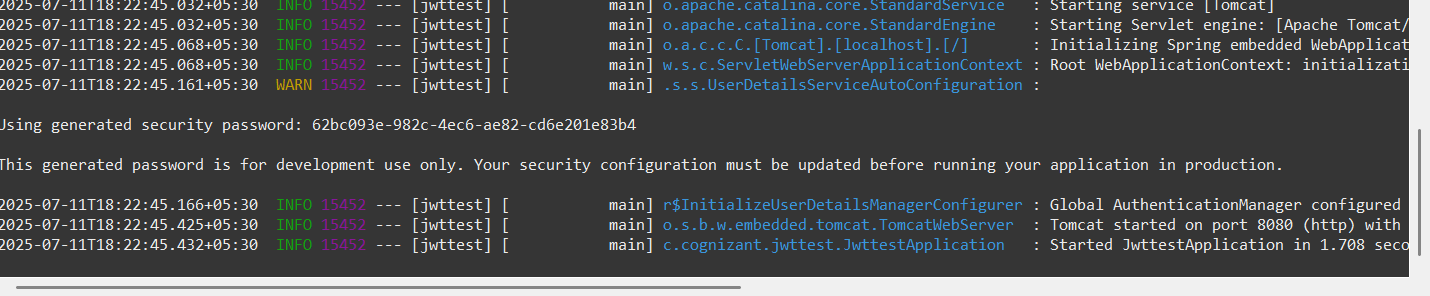
.httpBasic(); // Enable basic auth for username/password

return http.build();

}

}

**Output –**

****

1. **Read Authorization header and decode the username and password**

**Code –**

*AuthController.java –*

package com.cognizant.jwttest.controller;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

*@RestController*

public class AuthController {

*@PostMapping*("/authenticate")

public ResponseEntity<?> authenticate(*@RequestHeader*("Authorization") String authHeader) {

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

return ResponseEntity.*status*(*HttpStatus*.***UNAUTHORIZED***).body("Missing or invalid Authorization header");

}

// Decode Base64 credentials

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

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

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

// Split into username and password

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

if (values.length != 2) {

return ResponseEntity.*status*(*HttpStatus*.***BAD\_REQUEST***).body("Invalid credentials format");

}

String username = values[0];

String password = values[1];

// TEMP: Print to verify

return ResponseEntity.*ok*("Username: " + username + ", Password: " + password);

}

}

*SecurityConfig.java –*

package com.cognizant.jwttest.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

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(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

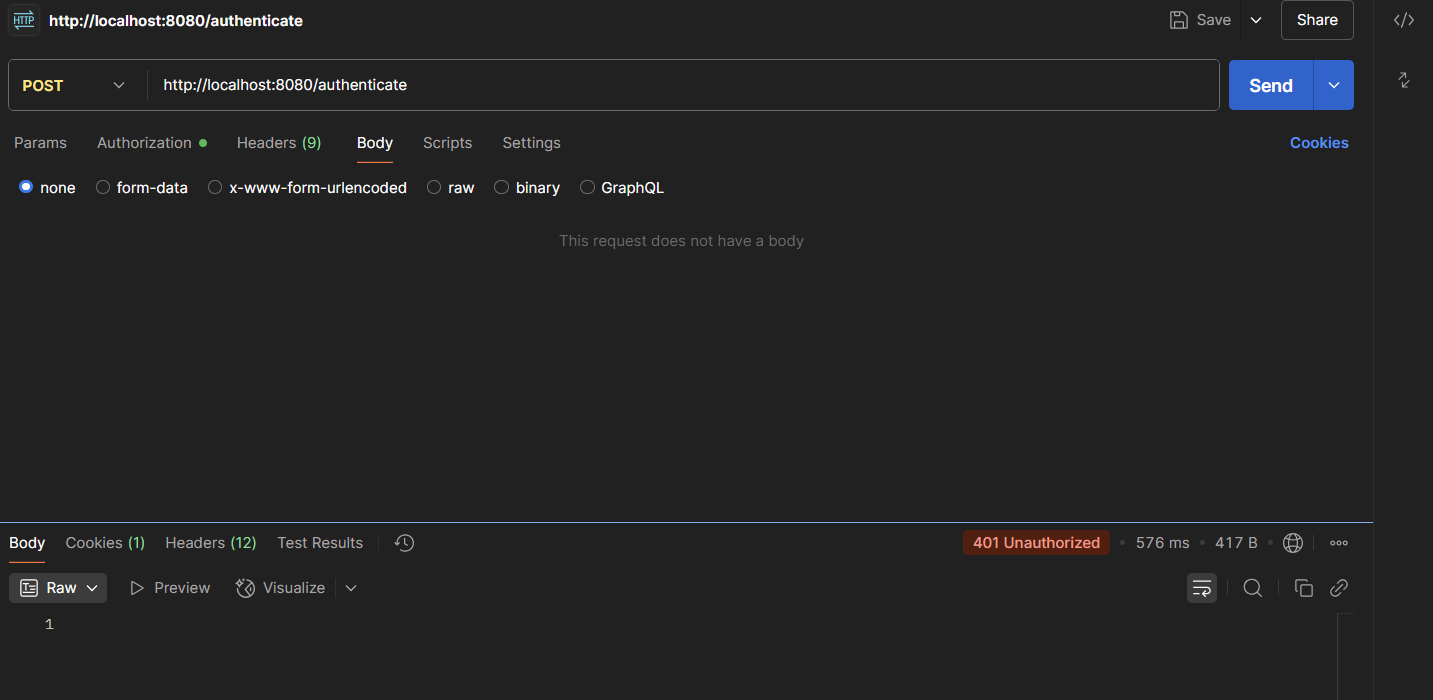
.httpBasic(Customizer.*withDefaults*()); // Enables Basic Auth

return http.build();

}

}

**Output –**

****

1. **Generate token based on the user retrieved in the previous step**

**Code –**

*JwtUtil.java –*

package com.cognizant.jwttest.util;

import java.util.Date;

import org.springframework.stereotype.Component;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

*@Component*

public class JwtUtil {

private final String SECRET\_KEY = "secret123"; // In real apps, use env vars

public String generateToken(String username) {

long nowMillis = System.*currentTimeMillis*();

long expMillis = nowMillis + 1000 \* 60 \* 10; // 10 mins expiry

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date(nowMillis))

.setExpiration(new Date(expMillis))

.signWith(*SignatureAlgorithm*.***HS256***, SECRET\_KEY.getBytes())

.compact();

}

}

*AuthController.java –*

package com.cognizant.jwttest.controller;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import com.cognizant.jwttest.util.JwtUtil;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

*@RestController*

public class AuthController {

*@Autowired*

private JwtUtil jwtUtil;

*@PostMapping*("/authenticate")

public ResponseEntity<?> authenticate(*@RequestHeader*("Authorization") String authHeader) {

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

return ResponseEntity.*status*(*HttpStatus*.***UNAUTHORIZED***).body("Missing or invalid Authorization header");

}

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

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

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

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

if (values.length != 2) {

return ResponseEntity.*status*(*HttpStatus*.***BAD\_REQUEST***).body("Invalid credentials format");

}

String username = values[0];

String password = values[1];

// Hardcoded check — later replace with DB or user service

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

String token = jwtUtil.generateToken(username);

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

response.put("token", token);

return ResponseEntity.*ok*(response);

} else {

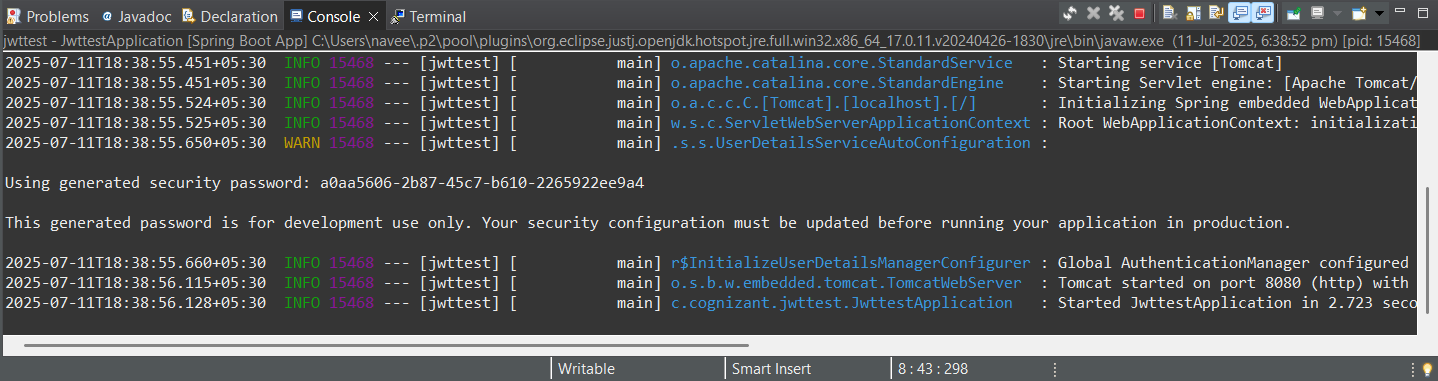
return ResponseEntity.*status*(*HttpStatus*.***UNAUTHORIZED***).body("Invalid username or password");

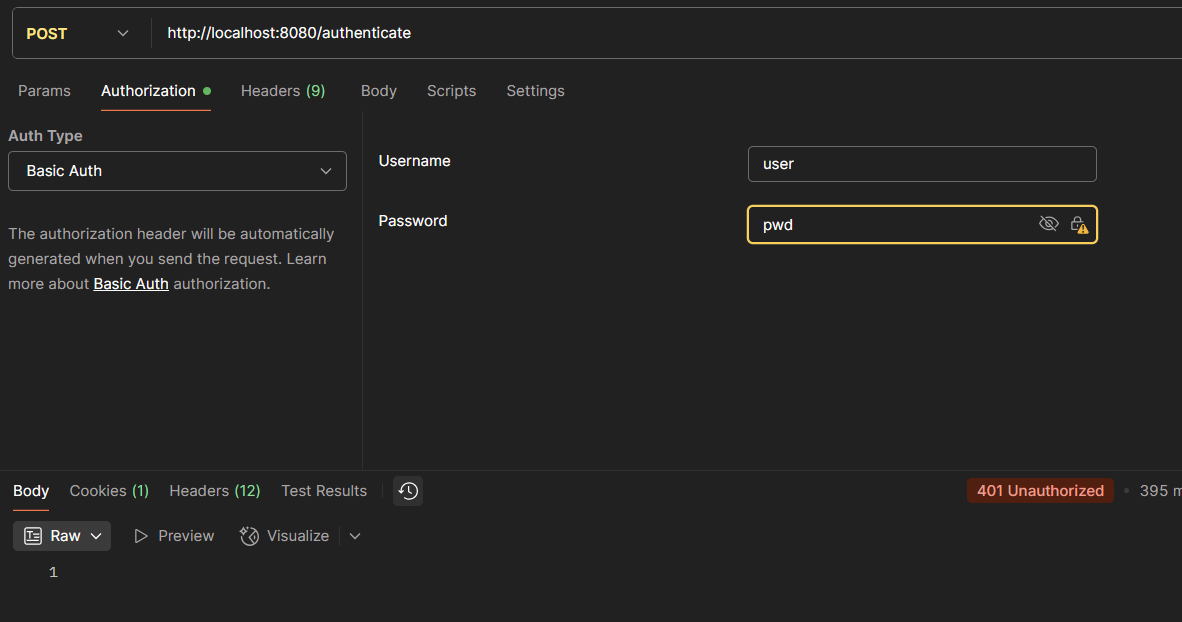
}

}

}

**Output –**

****

****