**EXERCISE 6:** **CREATE AUTHENTICATION SERVICE THAT RETURNS JWT**

**AuthenticationController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.config.JwtTokenUtil;

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

import javax.servlet.http.HttpServletRequest;

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private JwtTokenUtil jwtTokenUtil;

@Autowired

private InMemoryUserDetailsManager inMemoryUserDetailsManager;

@Autowired

private PasswordEncoder passwordEncoder;

@GetMapping("/authenticate")

public ResponseEntity<?> createAuthenticationToken(HttpServletRequest request) {

final String authorization = request.getHeader("Authorization");

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

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

String credentials = new String(Base64.getDecoder().decode(base64Credentials));

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

if (values.length == 2) {

String username = values[0];

String password = values[1];

try {

UserDetails userDetails = inMemoryUserDetailsManager.loadUserByUsername(username);

if (passwordEncoder.matches(password, userDetails.getPassword())) {

String token = jwtTokenUtil.generateToken(username);

return ResponseEntity.ok(new HashMap<>() {{

put("token", token);

}});

}

} catch (Exception e) {

return ResponseEntity.badRequest().body("Invalid credentials");

}

}

}

return ResponseEntity.badRequest().body("Authentication failed");

}

}

**SecurityConfig.java**

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.config.annotation.web.configuration.EnableWebSecurity;

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

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

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

.anyRequest().authenticated();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public InMemoryUserDetailsManager inMemoryUserDetailsManager() {

UserDetails user = User.builder()

.username("user")

.password(passwordEncoder().encode("pwd"))

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

}

**JwtTokenUtil.java**

package com.cognizant.springlearn.config;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@Component

public class JwtTokenUtil {

private static final String SECRET\_KEY = "cognizant-secret-key-123"; // Stronger secret key

private static final long EXPIRATION\_TIME = 3600000; // 1 hour

public String generateToken(String username) {

Map<String, Object> claims = new HashMap<>();

return Jwts.builder()

.setClaims(claims)

.setSubject(username)

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

.setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**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-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>

</dependencies>

**OUTPUT:**

