**Create authentication service that returns JWT**

**AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import javax.crypto.SecretKey;

@RestController

public class AuthenticationController {

private static final Logger *LOGGER* = LoggerFactory.*getLogger*(AuthenticationController.class);

private final SecretKey secretKey = Keys.*secretKeyFor*(SignatureAlgorithm.*HS256*);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

*LOGGER*.info("START - authenticate()");

*LOGGER*.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

*LOGGER*.debug("Extracted User: {}", user);

String token = generateJwt(user);

*LOGGER*.debug("Generated Token: {}", token);

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

map.put("token", token);

*LOGGER*.info("END - authenticate()");

return map;

}

private String getUser(String authHeader) {

*LOGGER*.debug("START - getUser()");

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

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

String credentials = new String(decodedBytes);

*LOGGER*.debug("Decoded credentials: {}", credentials);

String username = credentials.split(":", 2)[0];

*LOGGER*.debug("Extracted username: {}", username);

return username;

}

private String generateJwt(String user) {

JwtBuilder builder = Jwts.*builder*();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date((new Date()).getTime() + 1200000)); // 20 mins

builder.signWith(secretKey);

return builder.compact();

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.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.core.userdetails.User;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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("/countries").hasRole("USER")

.requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

)

.httpBasic();

return http.build();

}

@Bean

public UserDetailsService userDetailsService(PasswordEncoder passwordEncoder) {

return new InMemoryUserDetailsManager(

User.*builder*()

.username("user")

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

.roles("USER")

.build()

);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

**A computer screen with white text

AI-generated content may be incorrect.**

**A computer screen with a black rectangular object with a blue and white button

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**