**Hands On 4: Create authentication service that returns JWT** 

**AuthController.java**

package com.example.demo.controller;  
  
import com.example.demo.security.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Base64;  
import java.util.Map;  
  
import jakarta.servlet.http.HttpServletRequest;  
  
@RestController  
public class AuthController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @GetMapping("/authenticate")  
 public Map<String, String> authenticate(HttpServletRequest request) {  
 String authHeader = request.getHeader("Authorization"); if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 throw new RuntimeException("Missing or invalid Authorization header.");  
 }  
  
 String base64Credentials = authHeader.substring("Basic ".length()).trim();  
 byte[] decoded = Base64.*getDecoder*().decode(base64Credentials);  
 String[] credentials = new String(decoded).split(":", 2);  
 String username = credentials[0];  
 String password = credentials[1];  
String token = jwtUtil.generateToken(username);  
 return Map.*of*("token", token);  
 }  
}

**SecurityConfig.java**

package com.example.demo.security;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 @Bean  
 public InMemoryUserDetailsManager userDetailsService() {  
 UserDetails user = User.*withUsername*("user")  
 .password("pwd")  
 .roles("USER")  
 .build();  
 return new InMemoryUserDetailsManager(user);  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return NoOpPasswordEncoder.*getInstance*();}  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 )  
 .httpBasic(customizer -> {}); return http.build();  
 }  
}

**JwtUtil.java**

package com.example.demo.security;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
  
import java.util.Date;  
  
import org.springframework.stereotype.Component;  
  
@Component  
public class JwtUtil {  
  
 private final String secretKey = "ThisIsASecretKeyUsedToSignAndVerifyJWTsWithHmacSha256ItMustBeLongEnoughToProvideSecurityAndAvoidWarningsOrExceptionsRelatedToKeyLengthEspeciallyWhenUsingStrongEncryptionAlgorithmsInProduction";  
  
 public String generateToken(String username) {  
 long currentTimeMillis = System.*currentTimeMillis*();  
 long expirationTime = currentTimeMillis + 10 \* 60 \* 1000;  
  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date(currentTimeMillis))  
 .setExpiration(new Date(expirationTime))  
 .signWith(SignatureAlgorithm.*HS256*, secretKey)  
 .compact();  
 }  
}

**DemoApplication.java**

package com.example.demo;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class DemoApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(DemoApplication.class, args);  
 }  
  
}

**Output**

