**6.Create authentication service that returns JWT**

**SecurityConfig.java**

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER")

.and()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable().httpBasic().and()

.authorizeRequests()

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

.anyRequest().authenticated();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

**AuthenticationController.java**

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

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

String user = getUser(authHeader);

String token = generateJwt(user);

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

map.put("token", token);

return map;

}

private String getUser(String authHeader) {

String encodedCredentials = authHeader.replace("Basic ", "");

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

String decodedString = new String(decodedBytes);

return decodedString.split(":")[0];

}

private String generateJwt(String user) {

JwtBuilder builder = Jwts.builder()

.setSubject(user)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1200000))

.signWith(SignatureAlgorithm.HS256, "secretkey");

return builder.compact();

}

}

**OUTPUT:**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzAyNTg0NjAwLCJleHAiOjE3MDI1ODY0MDB9.AbcDEFg123XYZabcDEFghIjklMNO1234xyz"}