**PROGRAM:**

**SECURITYCONFIG.JAVA:**

package com.example.springrestapi;  
  
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.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configurers.HttpBasicConfigurer;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.stereotype.Component;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(authz -> authz  
 .requestMatchers("/authenticate").authenticated()  
 .anyRequest().permitAll()  
 )  
 .httpBasic(Customizer.*withDefaults*());  
  
 return http.build();  
 }  
 @Bean  
 public UserDetailsService users() {  
 UserDetails user = User  
 .*withUsername*("user")  
 .password("{noop}pwd") // NoOp means no encoding, plain text  
 .roles("USER")  
 .build();  
 return new InMemoryUserDetailsManager(user);  
 }  
}

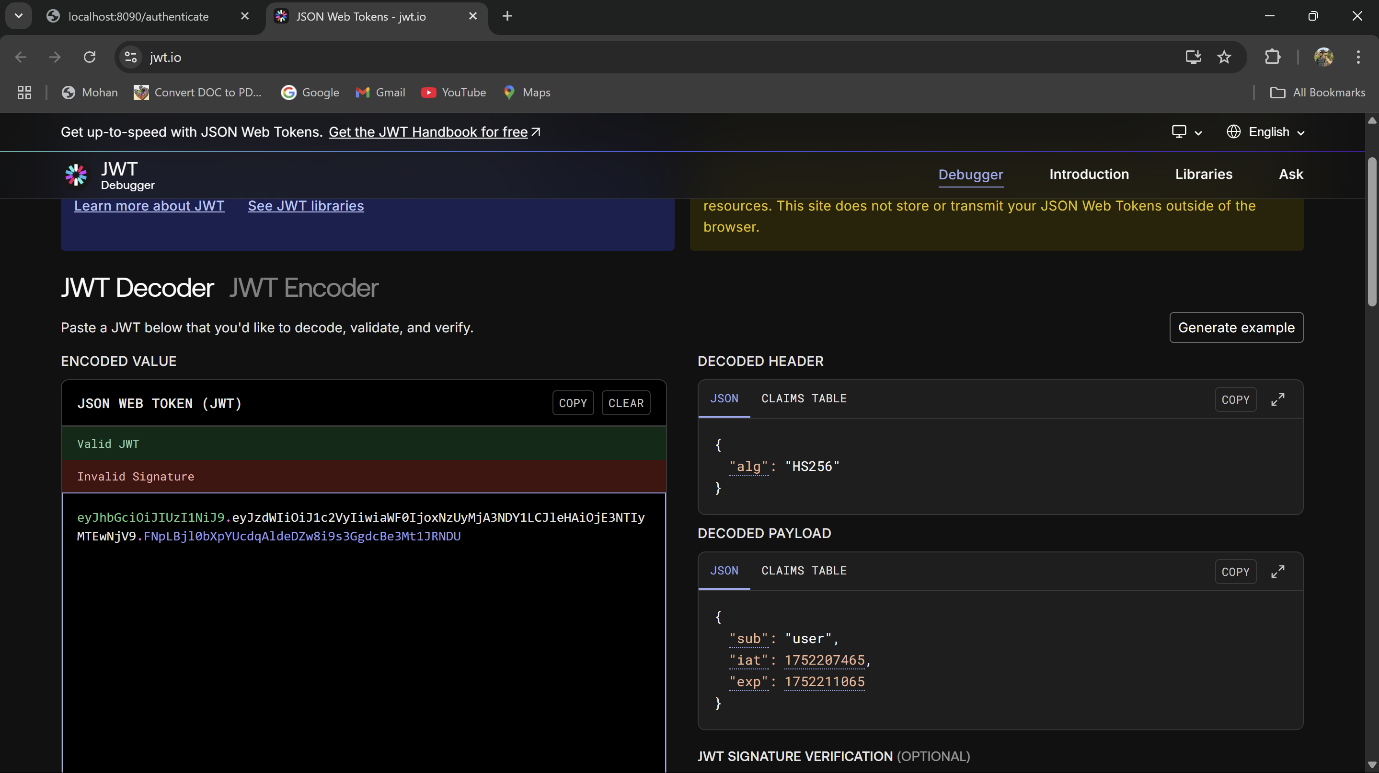
**JWTUNIT.JAVA:**

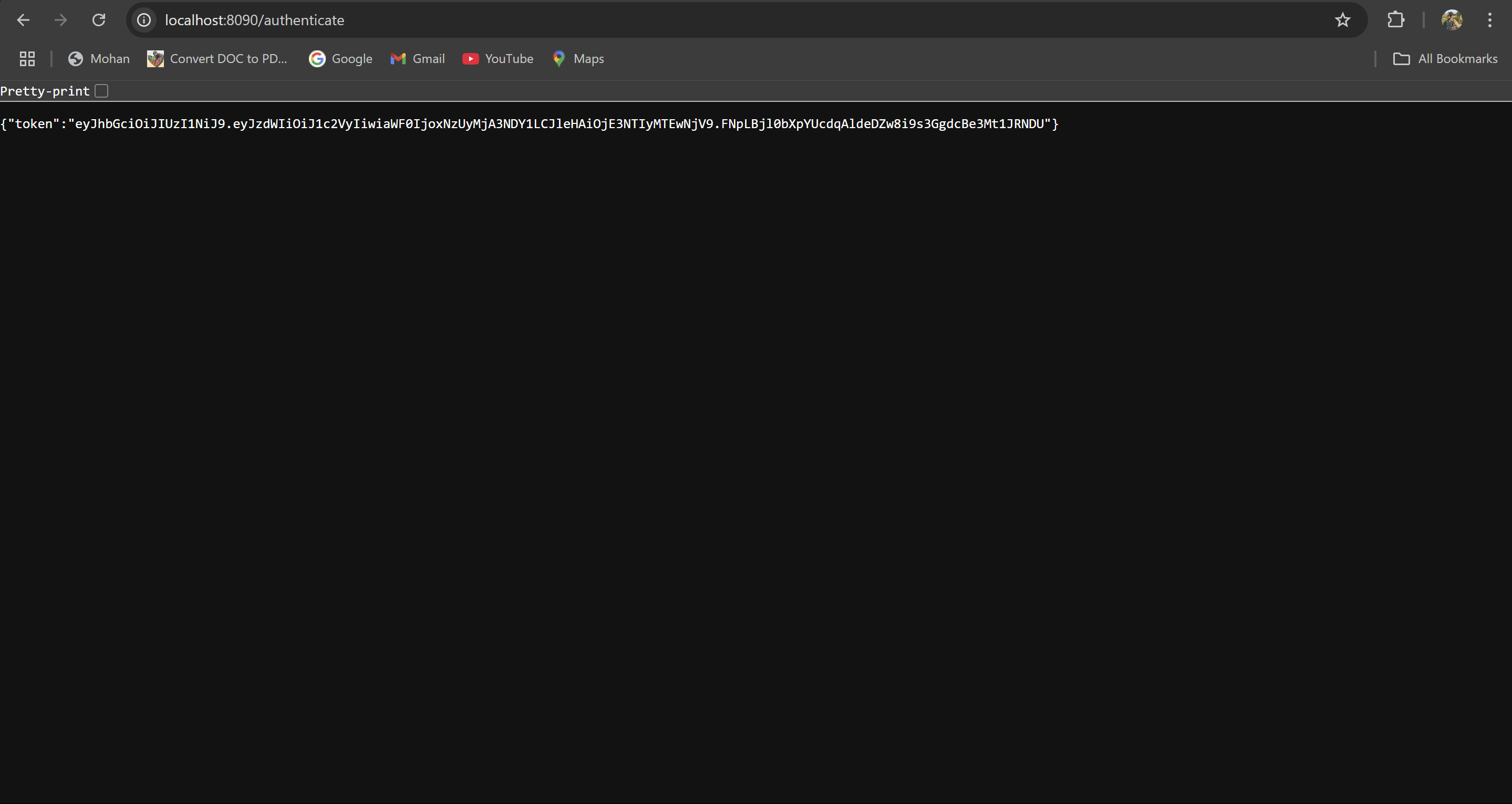
package com.example.springrestapi;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import io.jsonwebtoken.security.Keys;  
import org.springframework.stereotype.Component;  
  
import java.security.Key;  
import java.util.Date;  
  
  
@Component  
public class JwtUtil {  
 private static final String *secret* = "mysupersecretkeymysupersecretkey";  
 private static final Key *key* = Keys.*hmacShaKeyFor*(*secret*.getBytes());  
 public String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*()+3600000))  
 .signWith(SignatureAlgorithm.*HS256*, *key*)  
 .compact();  
 }  
}

**AUTHCONTROLLER:**

package com.example.springrestapi;  
  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestHeader;  
import org.springframework.web.bind.annotation.RestController;  
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;  
  
 @GetMapping("/authenticate")  
 public ResponseEntity<Map<String, String>>  
 authenticate(@RequestHeader("Authorization") String authHeader) {  
 if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 return ResponseEntity.*status*(HttpStatus.*BAD\_REQUEST*).build();  
 }  
 try {  
 String base64Credentials = authHeader.substring(6);  
 byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);  
 String credentials = new String(credDecoded, StandardCharsets.*UTF\_8*);  
 String[] values = credentials.split(":", 2);  
 String username = values[0];  
 String password = values[1];  
 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*).build();  
 }  
 } catch (Exception e){  
 return ResponseEntity.*badRequest*().build();  
 }  
 }  
}

**OUTPUT:**

****

****