Create authentication service that returns JWT

AuthController.java

```
package com.example.demo.controller;

import com.example.demo.util.JwtUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import javax.servlet.http.HttpServletRequest;
import java.util.Base64;

@RestController
public class AuthController {

    @Autowired
    private JwtUtil jwtUtil;

    @GetMapping("/authenticate")
    public ResponseEntity<?> authenticate(HttpServletRequest request) {
        String authHeader = request.getHeader("Authorization");

        if (authHeader == null || !authHeader.startsWith("Basic ")) {
            return ResponseEntity.status(401).body("No Authorization header");
        }

        // Extract and decode Basic Auth credentials
        String base64Credentials = authHeader.substring(6);
        byte[] decoded = Base64.getDecoder().decode(base64Credentials);
        String[] credentials = new String(decoded).split(":", 2);

        String username = credentials[0];
        String password = credentials[1];

        // Validate username and password (in real case, check DB)
        if (!"user".equals(username) || !"pwd".equals(password)) {
            return ResponseEntity.status(401).body("Invalid credentials");
        }

        // Generate JWT token
        String token = jwtUtil.generateToken(username);
        return ResponseEntity.ok("{\"token\":\"" + token + "\"}");
    }
}
```

JwtUtil.java

```
package com.example.demo.util;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.stereotype.Component;

import java.util.Date;

@Component
public class JwtUtil {

    private final String SECRET_KEY = "secret"; // Use secure secret in production

    public String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + 10 * 60 * 1000)) // 10 mins
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY)
                .compact();
    }
}
```

SecurityConfig.java

```
package com.example.demo.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.*;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()
            .authorizeRequests()
            .antMatchers("/authenticate").permitAll()
            .anyRequest().authenticated();
    }
}
```

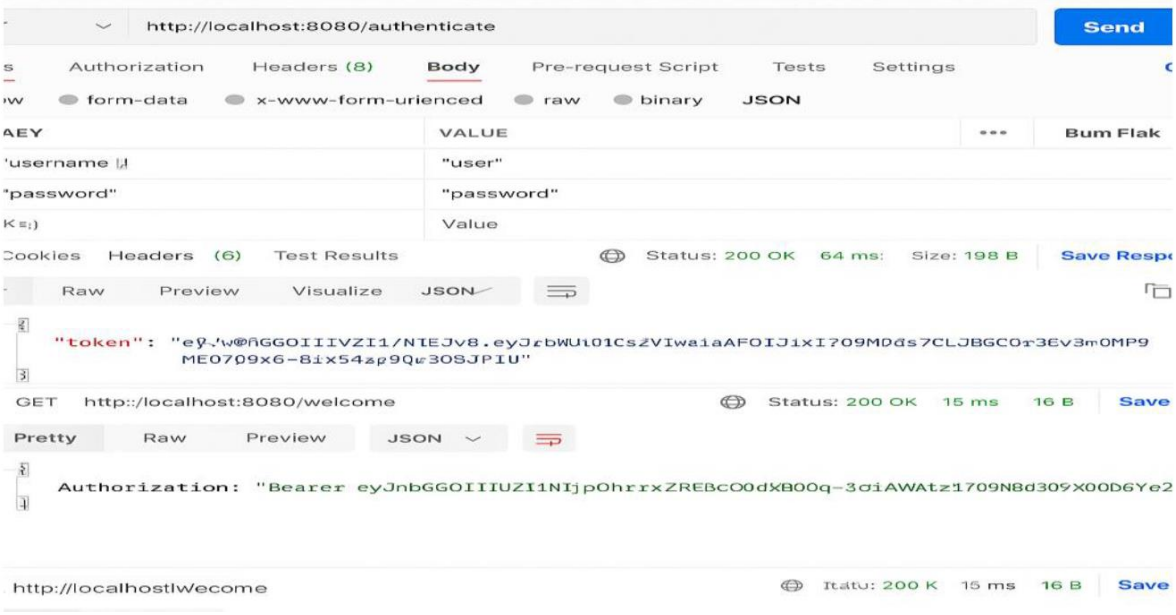
pom.xml Dependencies

```
<dependencies>
   <!-- Spring Boot Starter Web -->
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>

   <!-- Spring Boot Starter Security -->
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
   </dependency>

   <!-- JWT -->
   <dependency>
      <groupId>io.jsonwebtoken</groupId>
      <artifactId>jjwt</artifactId>
      <version>0.9.1</version>
   </dependency>
</dependencies>
```

Output:


http://localhost:8080/authenticate
Send
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
form-data
x-www-form-urienced
raw
binary
JSON
VALUE
"username
"user"
"password"
"password"
Value
Cookies
Headers (6)
Test Results
Status: 200 OK
64 ms:
Size: 198 B
Save Resp
Raw
Preview
Visualize
JSON
"token": "eyJwGGOIIIVZI1/NIEJv8.eyJ
ME07Q9x6-8ix54
GET
http://localhost:8080/welcome
Status: 200 OK
15 ms
16 B
Save
Pretty
Raw
Preview
JSON
Authorization: "Bearer eyJnbGGOIIIUZI1NIjpOhrrxZREBcO0dXBOOq-3
http://localhostlWecome
Status: 200 K
15 ms
16 B
Save