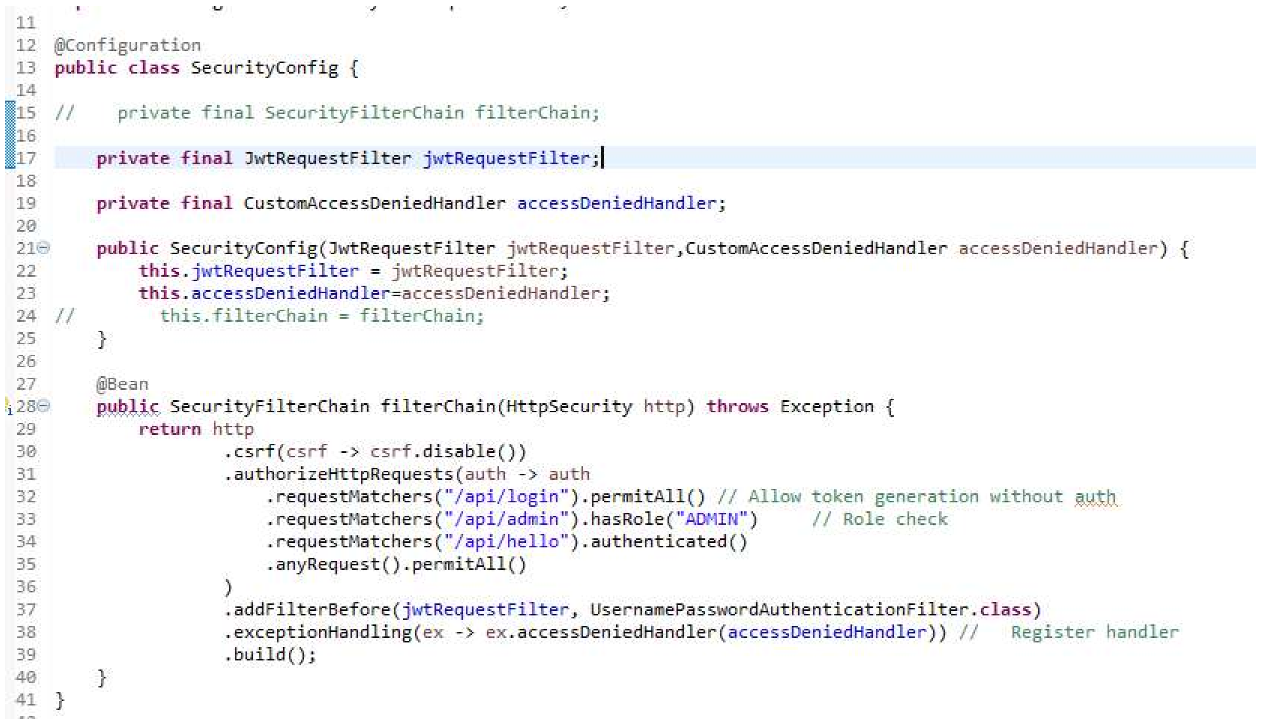
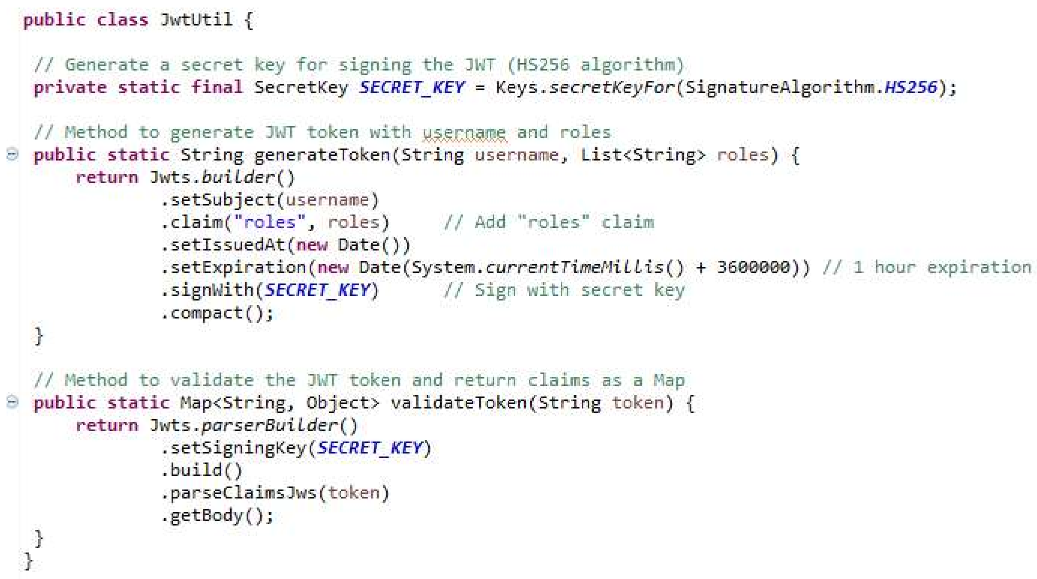
Rudraksh Chourey Assignment 3

Security config code:



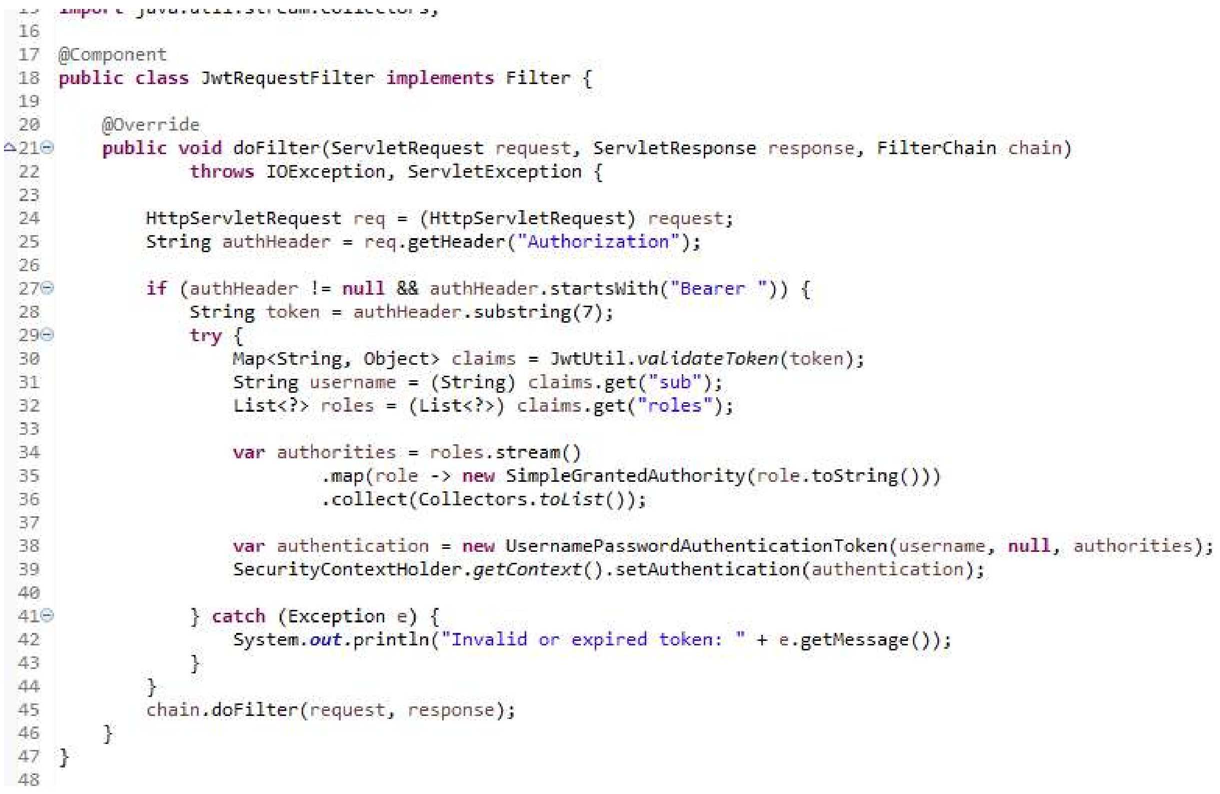
```java
@Configuration
public class SecurityConfig {

//    private final SecurityFilterChain filterChain;

    private final JwtRequestFilter jwtRequestFilter;

    private final CustomAccessDeniedHandler accessDeniedHandler;

    public SecurityConfig(JwtRequestFilter jwtRequestFilter,CustomAccessDeniedHandler accessDeniedHandler) {
        this.jwtRequestFilter = jwtRequestFilter;
        this.accessDeniedHandler=accessDeniedHandler;
//          this.filterChain = filterChain;
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        return http
                .csrf(csrf -> csrf.disable())
                .authorizeHttpRequests(auth -> auth
                    .requestMatchers("/api/login").permitAll() // Allow token generation without auth
                    .requestMatchers("/api/admin").hasRole("ADMIN")     // Role check
                    .requestMatchers("/api/hello").authenticated()
                    .anyRequest().permitAll()
                )
                .addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class)
                .exceptionHandling(ex -> ex.accessDeniedHandler(accessDeniedHandler)) //   Register handler
                .build();
    }
}
```

JWTUtil:



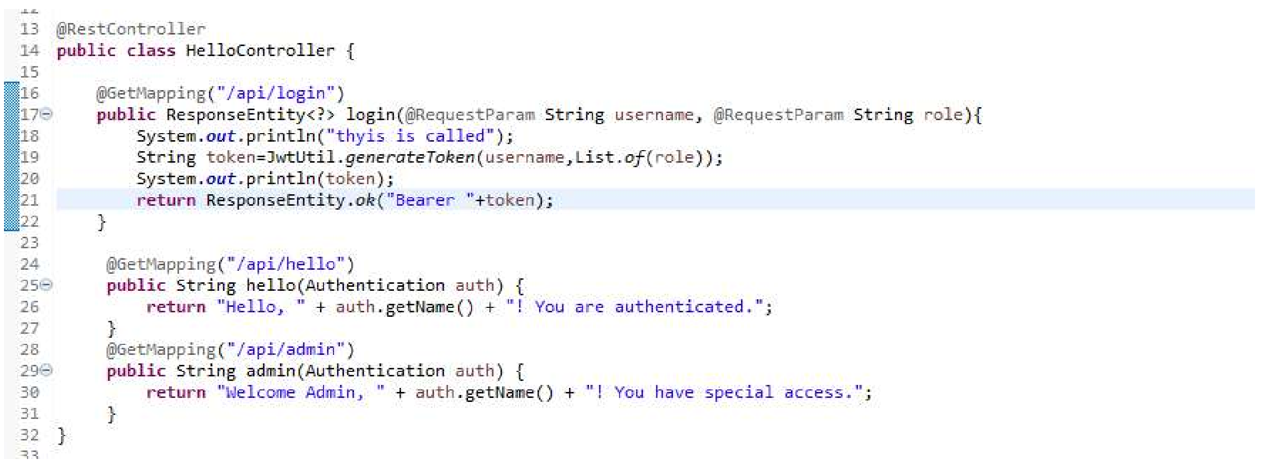
```java
public class JwtUtil {

 // Generate a secret key for signing the JWT (HS256 algorithm)
 private static final SecretKey SECRET_KEY = Keys.secretKeyFor(SignatureAlgorithm.HS256);

 // Method to generate JWT token with username and roles
 public static String generateToken(String username, List<String> roles) {
     return Jwts.builder()
             .setSubject(username)
             .claim("roles", roles)     // Add "roles" claim
             .setIssuedAt(new Date())
             .setExpiration(new Date(System.currentTimeMillis() + 3600000)) // 1 hour expiration
             .signWith(SECRET_KEY)      // Sign with secret key
             .compact();
 }

 // Method to validate the JWT token and return claims as a Map
 public static Map<String, Object> validateToken(String token) {
     return Jwts.parserBuilder()
             .setSigningKey(SECRET_KEY)
             .build()
             .parseClaimsJws(token)
             .getBody();
 }
}
```

Filter:



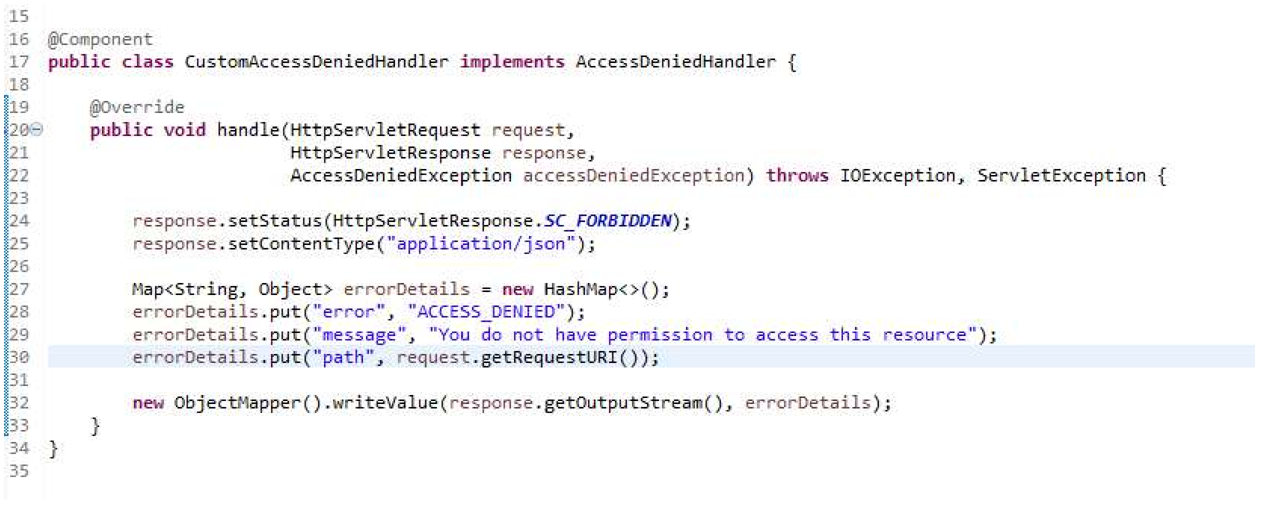
```java

@Component
public class JwtRequestFilter implements Filter {

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {

        HttpServletRequest req = (HttpServletRequest) request;
        String authHeader = req.getHeader("Authorization");

        if (authHeader != null && authHeader.startsWith("Bearer ")) {
            String token = authHeader.substring(7);
            try {
                Map<String, Object> claims = JwtUtil.validateToken(token);
                String username = (String) claims.get("sub");
                List<?> roles = (List<?>) claims.get("roles");

                var authorities = roles.stream()
                        .map(role -> new SimpleGrantedAuthority(role.toString()))
                        .collect(Collectors.toList());

                var authentication = new UsernamePasswordAuthenticationToken(username, null, authorities);
                SecurityContextHolder.getContext().setAuthentication(authentication);

            } catch (Exception e) {
                System.out.println("Invalid or expired token: " + e.getMessage());
            }
        }
        chain.doFilter(request, response);
    }
}

```

Controller:



```java
@RestController
public class HelloController {

    @GetMapping("/api/login")
    public ResponseEntity<?> login(@RequestParam String username, @RequestParam String role){
        System.out.println("thyis is called");
        String token=JwtUtil.generateToken(username,List.of(role));
        System.out.println(token);
        return ResponseEntity.ok("Bearer "+token);
    }

     @GetMapping("/api/hello")
     public String hello(Authentication auth) {
        return "Hello, " + auth.getName() + "! You are authenticated.";
     }
     @GetMapping("/api/admin")
     public String admin(Authentication auth) {
        return "Welcome Admin, " + auth.getName() + "! You have special access.";
     }
}
```

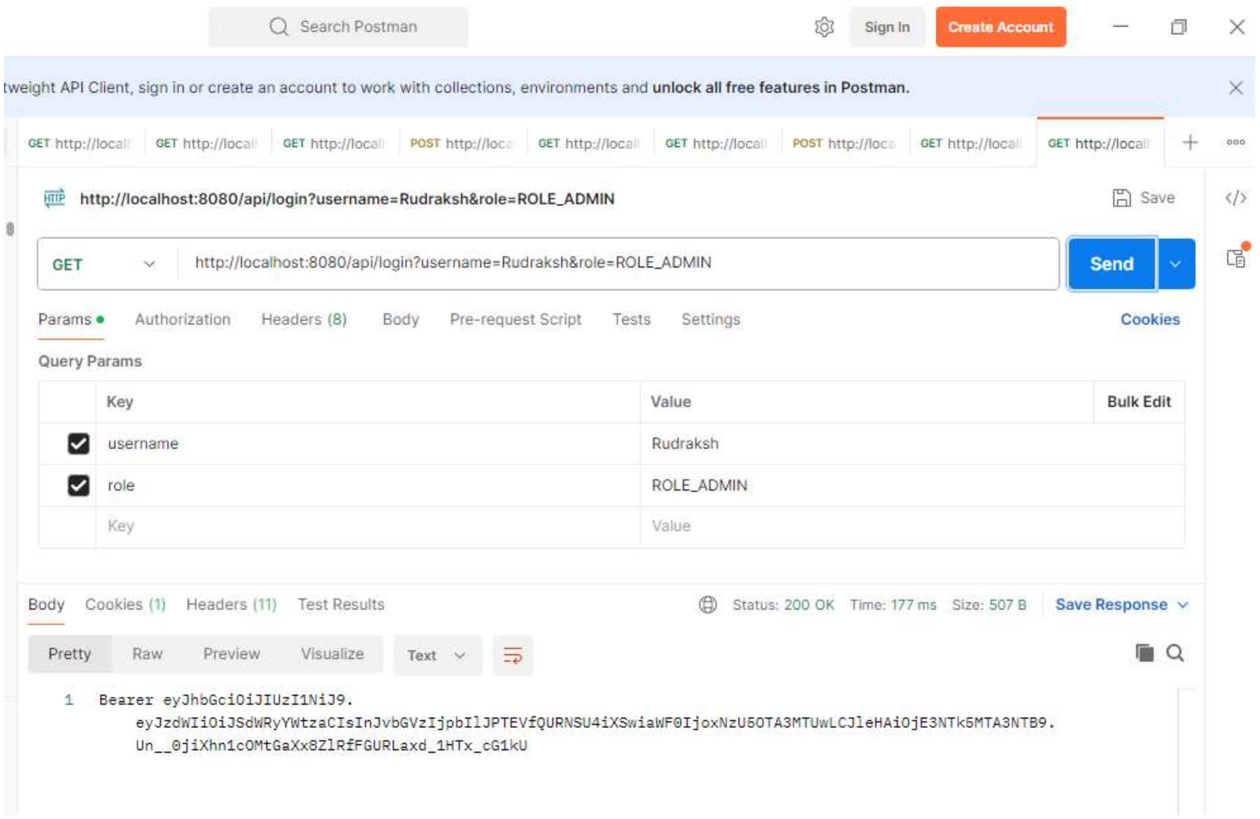
Custom Access Denied Handler:



```java

@Component
public class CustomAccessDeniedHandler implements AccessDeniedHandler {

    @Override
    public void handle(HttpServletRequest request,
                       HttpServletResponse response,
                       AccessDeniedException accessDeniedException) throws IOException, ServletException {

        response.setStatus(HttpServletResponse.SC_FORBIDDEN);
        response.setContentType("application/json");

        Map<String, Object> errorDetails = new HashMap<>();
        errorDetails.put("error", "ACCESS_DENIED");
        errorDetails.put("message", "You do not have permission to access this resource");
        errorDetails.put("path", request.getRequestURI());

        new ObjectMapper().writeValue(response.getOutputStream(), errorDetails);
    }
}

```

Output:

Generate token with role:



tweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

http://localhost:8080/api/login?username=Rudraksh&role=ROLE_ADMIN

GET http://localhost:8080/api/login?username=Rudraksh&role=ROLE_ADMIN

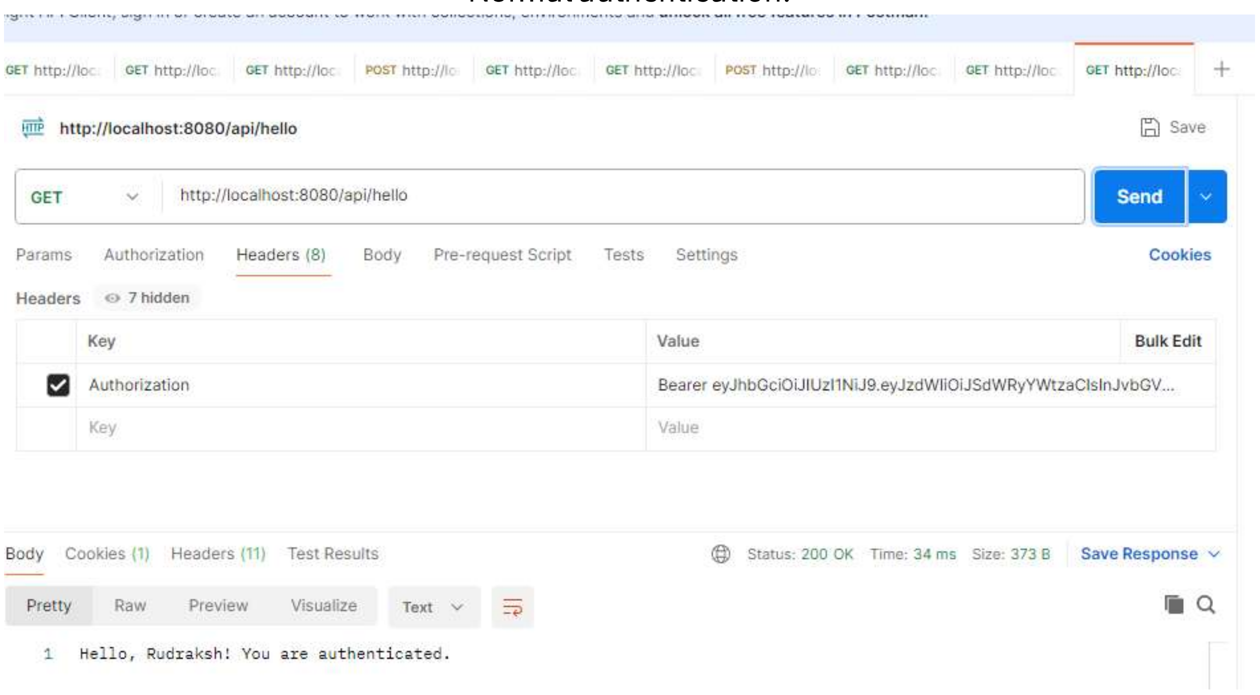
Query Params

	Key	Value
✓	username	Rudraksh
✓	role	ROLE_ADMIN

Body Cookies (1) Headers (11) Test Results — Status: 200 OK Time: 177 ms Size: 507 B

```
Bearer eyJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJSdWRyYWtzaCIsInJvbGVzIjpbIlJPTEVfQURNSU4iXSwiaWF0IjoxNzU5OTA3MTUwLCJleHAiOjE3NTk5MTA3NTB9.
    Un__0jiXhn1cOMtGaXx8ZlRfFGURLaxd_1HTx_cG1kU
```

Normal authentication:



http://localhost:8080/api/hello

GET http://localhost:8080/api/hello

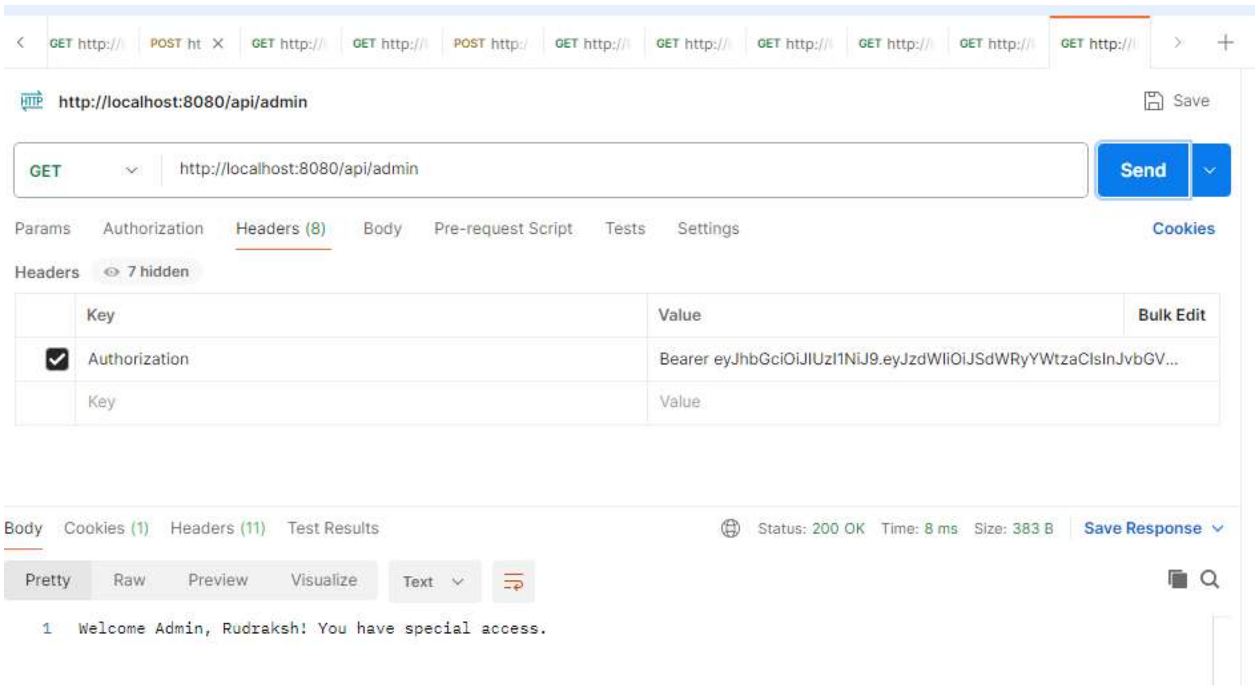
Headers

	Key	Value
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJSdWRyYWtzaCIsInJvbGV...

Body Cookies (1) Headers (11) Test Results — Status: 200 OK Time: 34 ms Size: 373 B

```
Hello, Rudraksh! You are authenticated.
```

Admin Authentication:


http://localhost:8080/api/admin

GET http://localhost:8080/api/admin

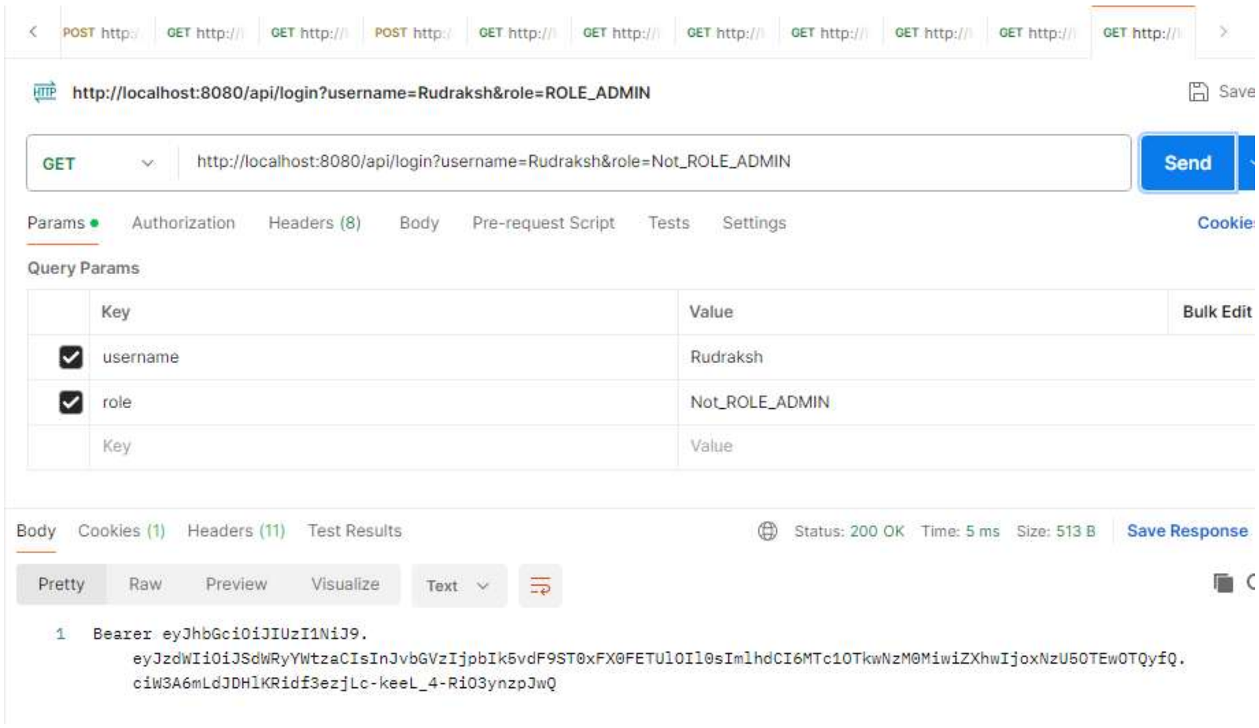
Headers (8)

	Key	Value
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJSdWRyYWtzaClsInJvbGV...

Status: 200 OK Time: 8 ms Size: 383 B

```
Welcome Admin, Rudraksh! You have special access.
```

Generating token with not admin role:


http://localhost:8080/api/login?username=Rudraksh&role=ROLE_ADMIN

GET http://localhost:8080/api/login?username=Rudraksh&role=Not_ROLE_ADMIN

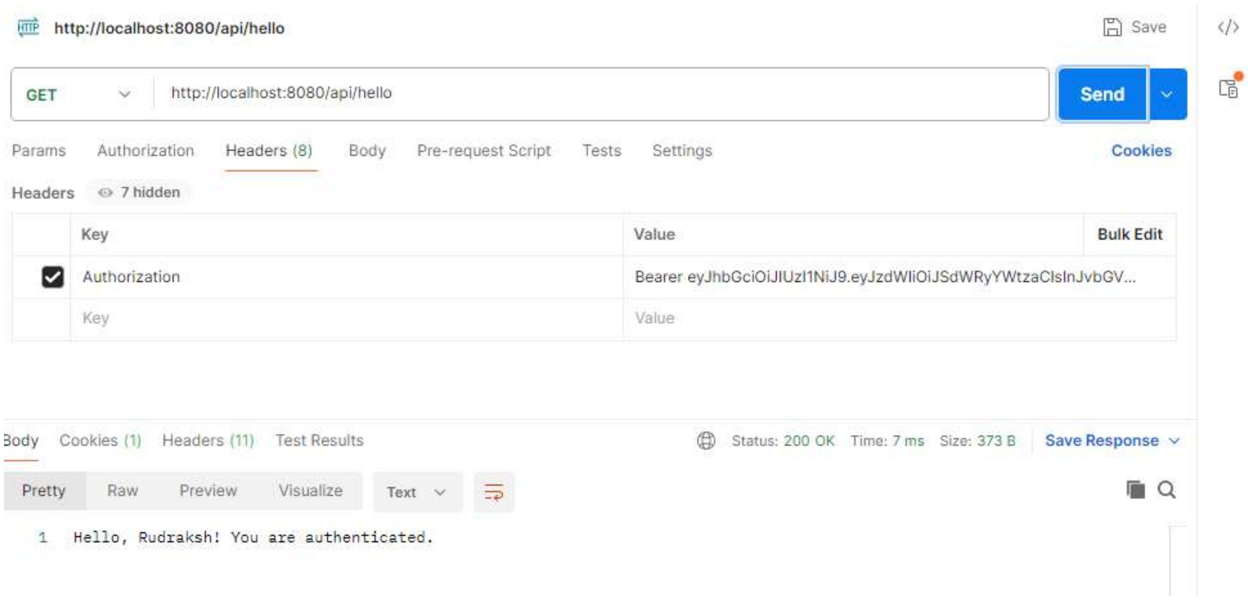
Query Params

	Key	Value
✓	username	Rudraksh
✓	role	Not_ROLE_ADMIN

Status: 200 OK Time: 5 ms Size: 513 B

```
Bearer eyJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJSdWRyYWtzaCIsInJvbGVzIjpbIk5vdF9ST0xFX0FETUlOIl0sImlhdCI6MTc1OTkwNzM0MiwiZXhwIjoxNzU5OTEwOTQyfQ.
    ciW3A6mLdJDHlKRidf3ezjLc-keeL_4-RiO3ynzpJwQ
```

Authentication:



http://localhost:8080/api/hello

GET http://localhost:8080/api/hello

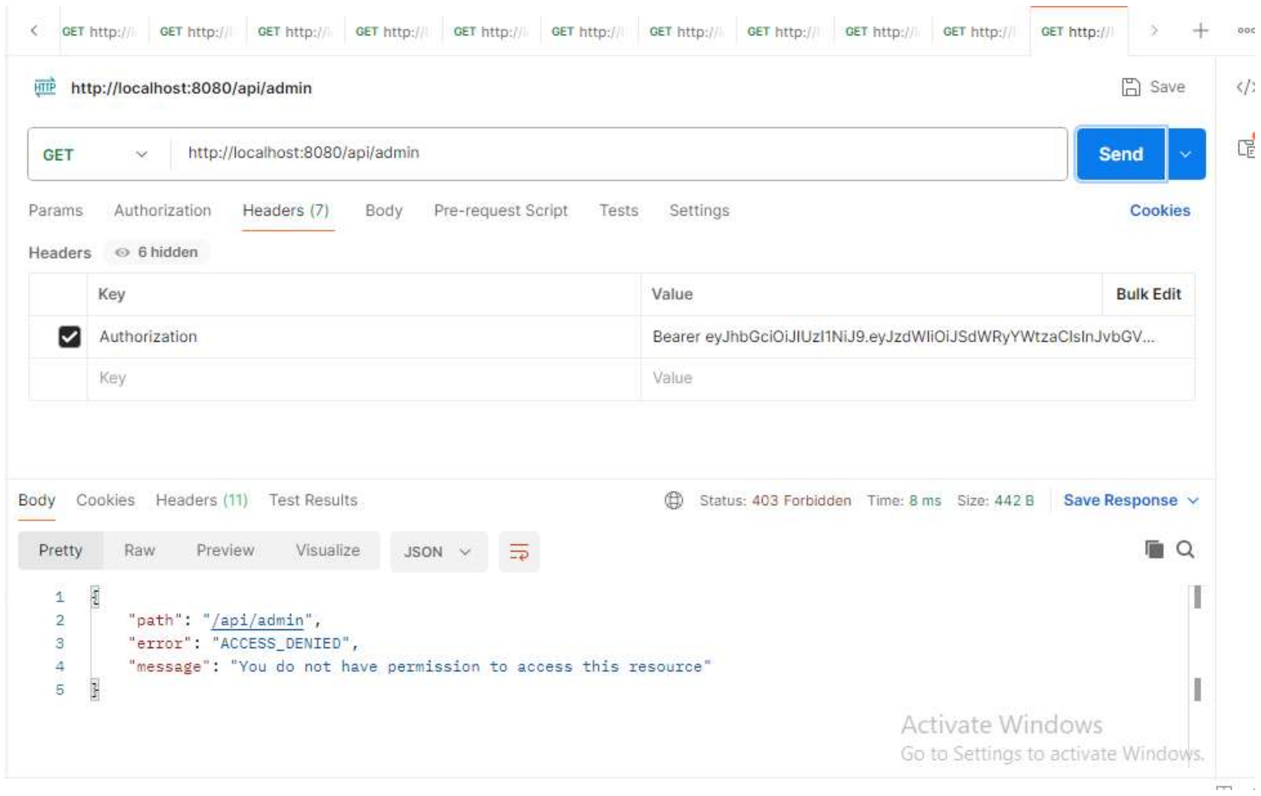
Headers (8)

Key	Value
Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJSdWRyYWtzaCIsInJvbGV...

Status: 200 OK Time: 7 ms Size: 373 B

```
Hello, Rudraksh! You are authenticated.
```

Admin Authentication:



http://localhost:8080/api/admin

GET http://localhost:8080/api/admin

Headers (7)

Key	Value
Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJSdWRyYWtzaCIsInJvbGV...

Status: 403 Forbidden Time: 8 ms Size: 442 B

```
{
    "path": "/api/admin",
    "error": "ACCESS_DENIED",
    "message": "You do not have permission to access this resource"
}
```