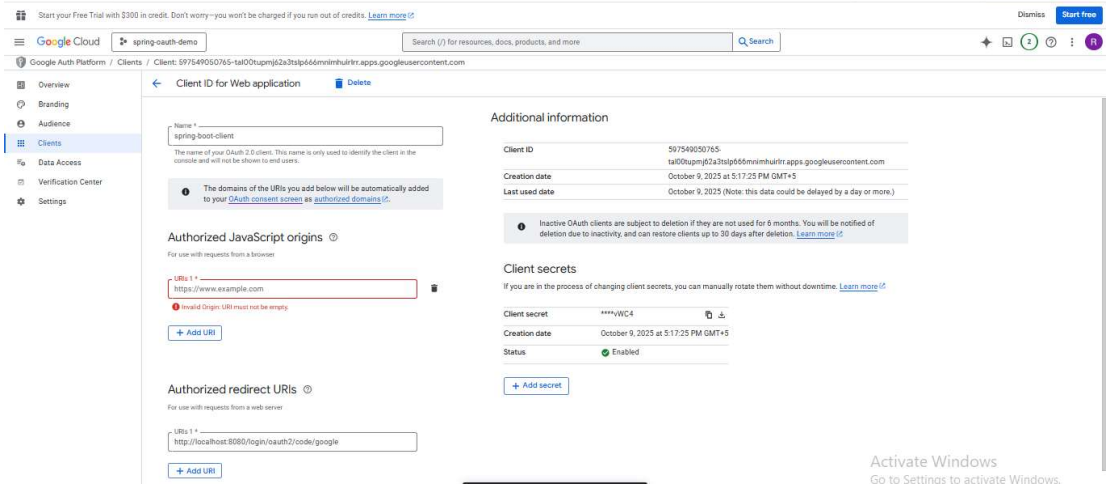
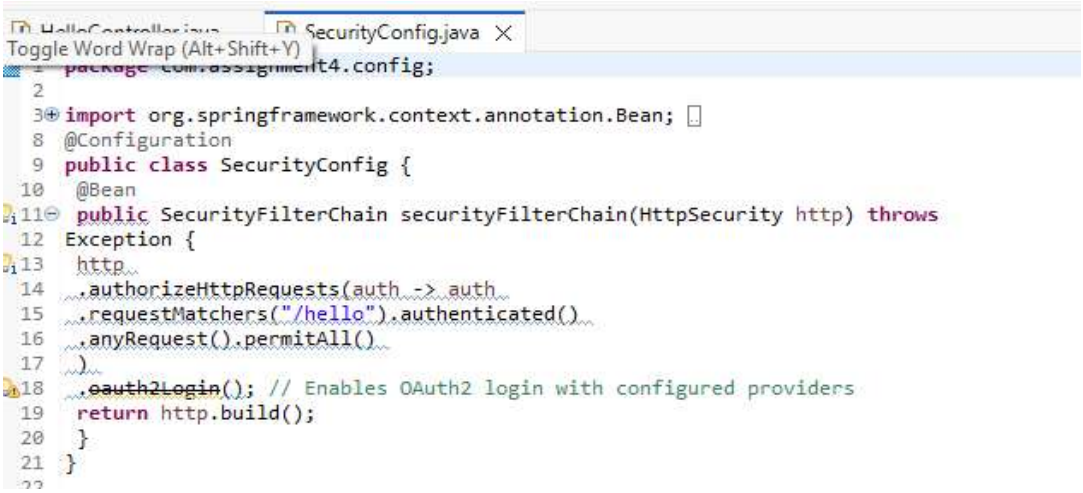
Rudraksh Chourey Assignment 4

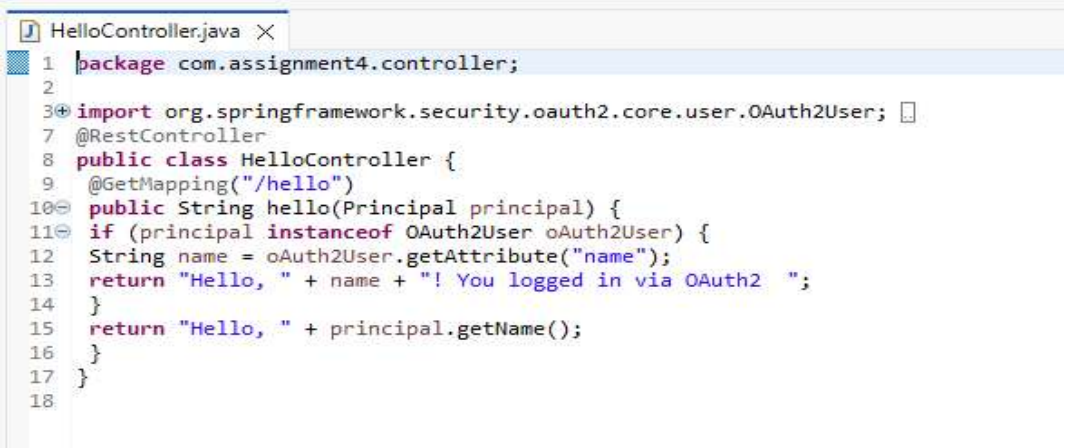
Google oath client creation



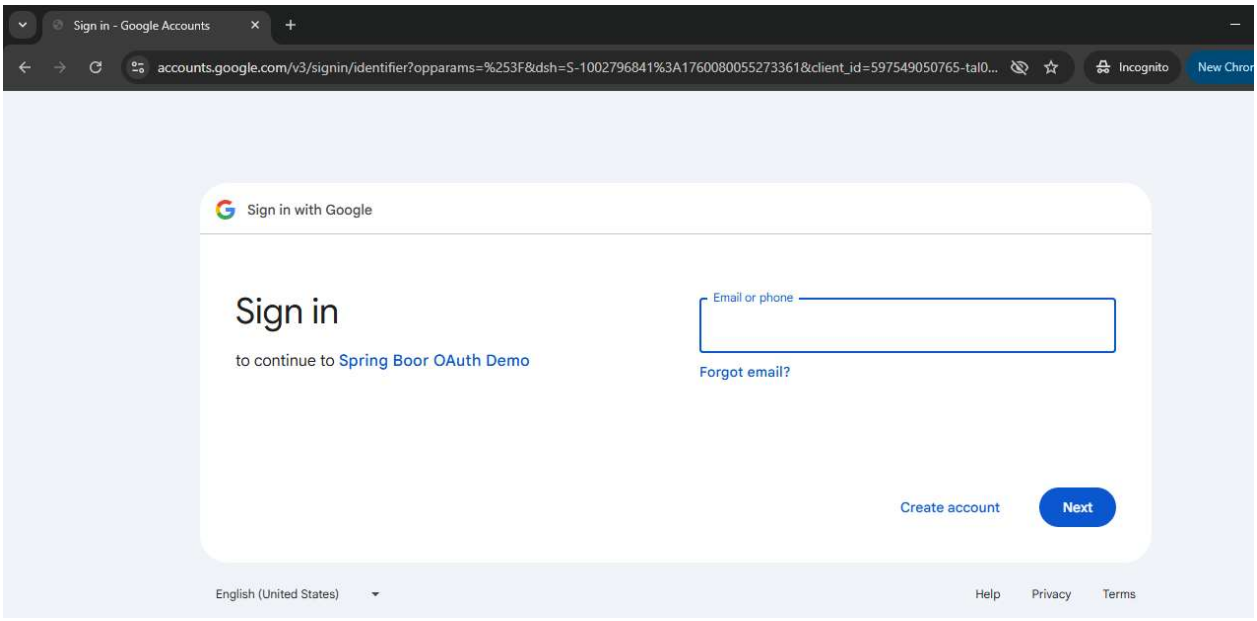
Security Config:



Controller:



Login Page:


Sign in - Google Accounts
accounts.google.com/v3/signin/identifier?opparams=%253F&dsh=S-1002796841%3A1760080055273361&client_id=597549050765-tal0...
Incognito
Sign in with Google
Sign in
to continue to Spring Boor OAuth Demo
Email or phone
Forgot email?
Create account
Next
English (United States)
Help
Privacy
Terms


localhost:8080/hello?continue
localhost:8080/hello?continue
Hello, 110050032130039120461

Swagger First Exercise:

Swagger Config:

HelloController.java | SwaggerConfig.java

```java
package com.assignment4b.config;

import io.swagger.v3.oas.models.OpenAPI;
@Configuration
public class SwaggerConfig {
 @Bean
 public OpenAPI customOpenAPI() {
 return new OpenAPI()
 .info(new Info()
 .title("My School API")
 .version("1.0")
 .description("API documentation for my Spring Boot project"));
 }
}
```

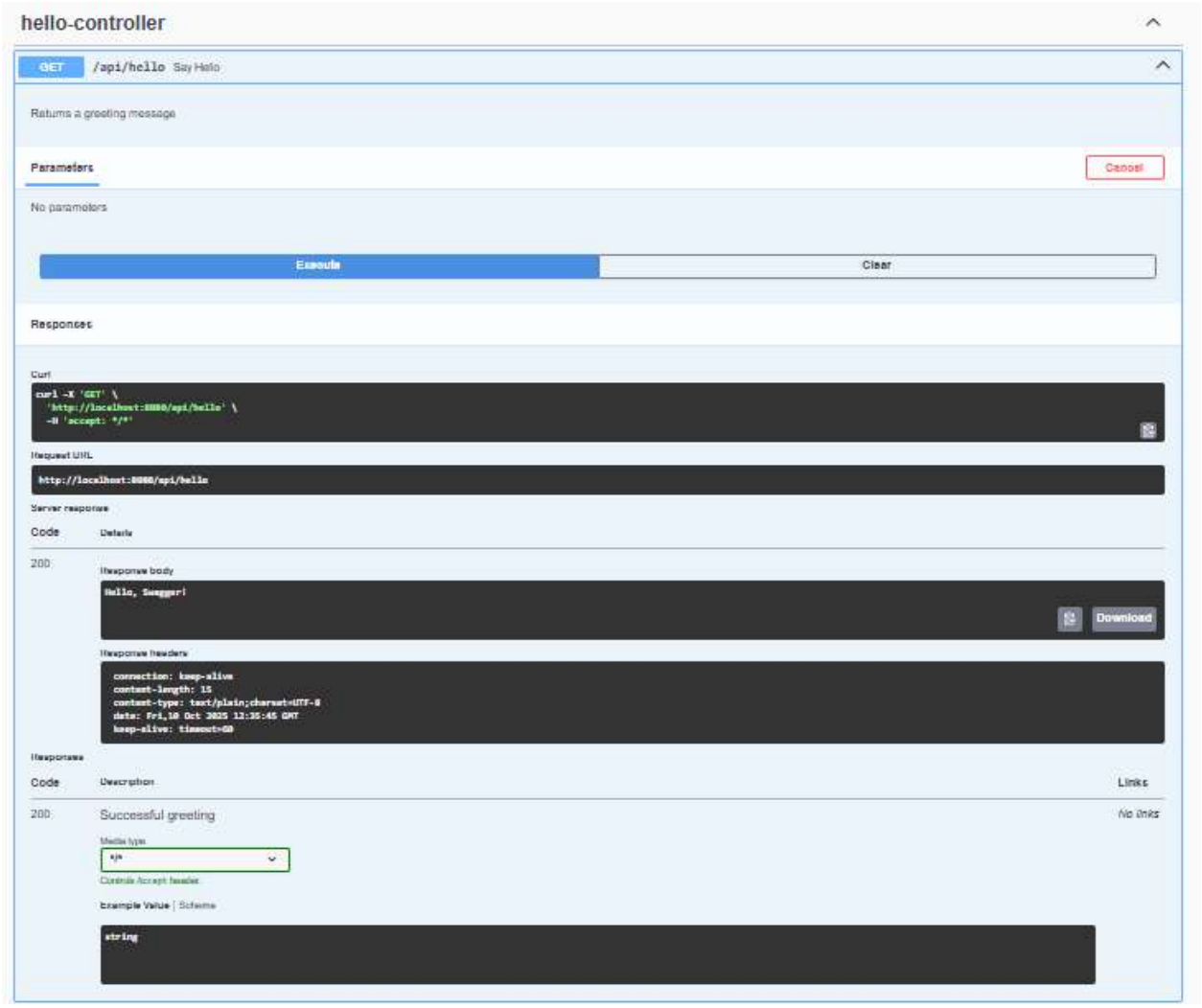
Controller:

HelloController.java | SwaggerConfig.java

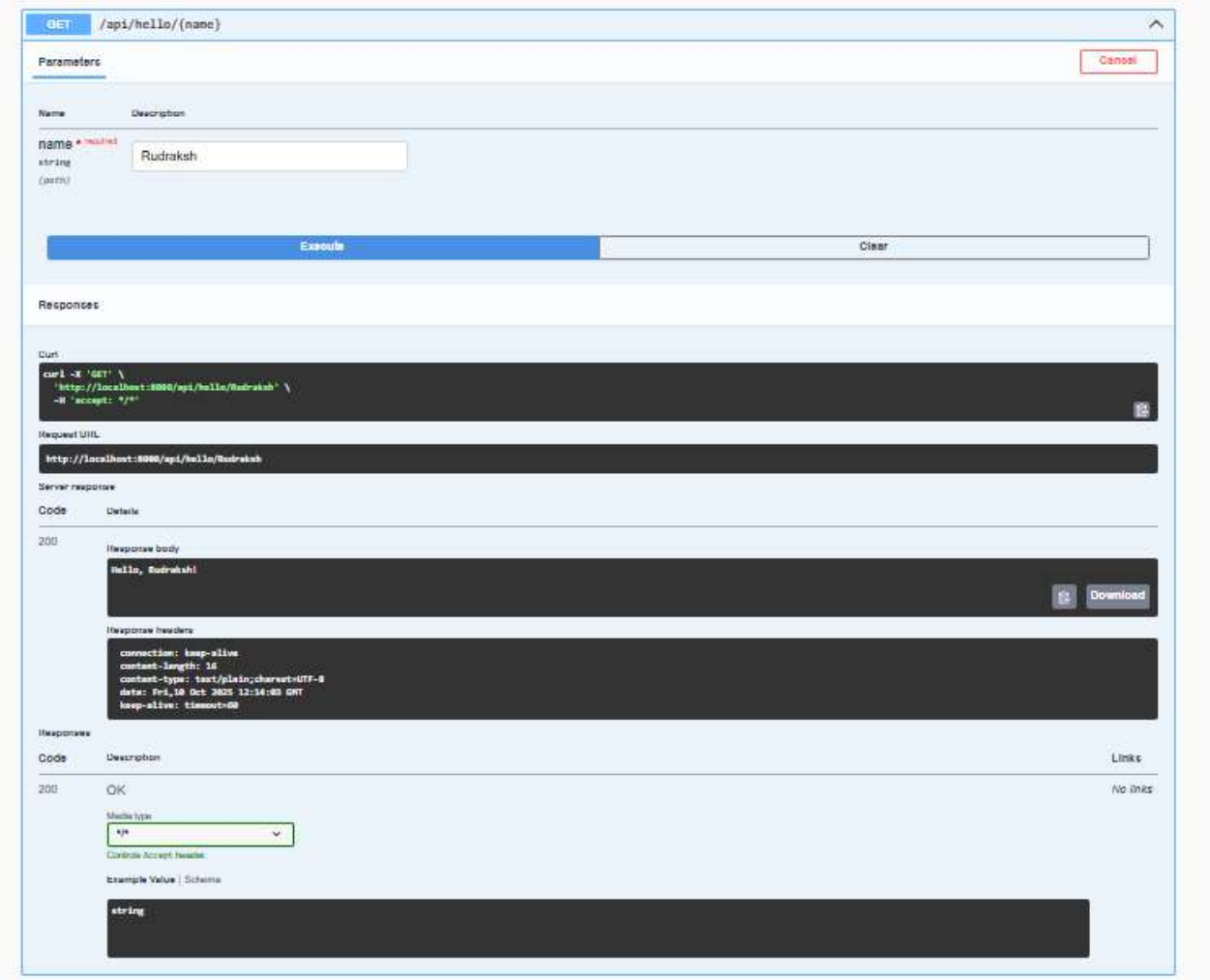
```java
package com.assignment4b.config;

import org.springframework.web.bind.annotation.*;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
@RestController
@RequestMapping("/api/hello")
public class HelloController {
 @GetMapping
 @Operation(summary = "Say Hello", description = "Returns a greeting message")
  @ApiResponse(responseCode = "200", description = "Successful greeting")
 public String sayHello() {
 return "Hello, Swagger!";
 }
 @GetMapping("/{name}")
 public String greetByName(@PathVariable String name) {
 return "Hello, " + name + "!";
 }
}
```

Swagger first ex 1:

First mapping:



Second Mapping:



OpenAPI JSON:


```
{
  "openapi": "3.0.1",
  "info": {
    "title": "My School API",
    "description": "API documentation for my Spring Boot project",
    "version": "1.0"
  },
  "servers": [
    {
      "url": "http://localhost:8080",
      "description": "Generated server url"
    }
  ],
  "paths": {
    "/api/hello": {
      "get": {
        "tags": [
          "hello-controller"
        ],
        "summary": "Say Hello",
        "description": "Returns a greeting message",
        "operationId": "sayHello",
        "responses": {
          "200": {
            "description": "Successful greeting",
            "content": {
              "*/*": {
                "schema": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    },
    "/api/hello/{name}": {
      "get": {
        "tags": [
          "hello-controller"
        ],
        "operationId": "greetByName",
        "parameters": [
          {
            "name": "name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "content": {
              "*/*": {
                "schema": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  },
  "components": {

  }
}
```

Security Config Code:

SecurityConfig.java

```java
package com.assignment4b.config;

import org.springframework.context.annotation.Bean;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    private final AuthController
    authController;

    public SecurityConfig(AuthController authController) {
        this.authController = authController;
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.csrf(csrf -> csrf.disable()) // Disable CSRF for Stateless API (JWT)

            // 1. Authorization Rules (Order Matters: PermitAll first!)
            .authorizeHttpRequests(auth -> auth
                // Allow public access for Swagger UI, API docs, and JWT login/token generation
                .requestMatchers("/swagger-ui/**", "/v3/api-docs/**", "/api/auth/**").permitAll()

                // Require authentication for /hello to trigger the OAuth2 login flow
                .requestMatchers("/hello").authenticated()

                // Require ADMIN role for admin endpoints (JWT-protected)
                .requestMatchers("/api/admin/**").hasRole("ADMIN")

                // All other requests require authentication (will be caught by either OAuth2 Login or JWT)
                .anyRequest().authenticated()
            )

            // 2. Enable Session-based Login (Google OAuth2 Client)
            // This is primarily used for browser access to endpoints like /hello
            .oauth2Login();



            // 3. Enable Token-based Authentication (JWT Resource Server)
            // This is primarily used for API calls to endpoints like /api/secure
            http.oauth2ResourceServer(oauth2 -> oauth2.jwt(jwt ->
                jwt.jwtAuthenticationConverter(this::convertJwt)));

        return http.build();
    }

    /**
     * Configures the JwtDecoder to use the secret key generated by the AuthController
     */
    @Bean
    public JwtDecoder jwtDecoder() {
        return NimbusJwtDecoder.withSecretKey((SecretKey) authController.getKey()).build();
    }

    /**
     * Customizes the JWT authentication process to extract the 'roles' claim
     * and map them to Spring Security authorities.
     */
    private AbstractAuthenticationToken convertJwt(Jwt jwt) {
        // Extracts 'roles' claim (which is a List<String>) and maps them to SimpleGrantedAuthority
        Collection<GrantedAuthority> authorities = ((List<String>)
                jwt.getClaims().get("roles")).stream()
                .map(SimpleGrantedAuthority::new)
                .collect(Collectors.toList());

        return new JwtAuthenticationToken(jwt, authorities);
    }
}
```

Auth Controller:

SecurityConfig.java | AuthController.java

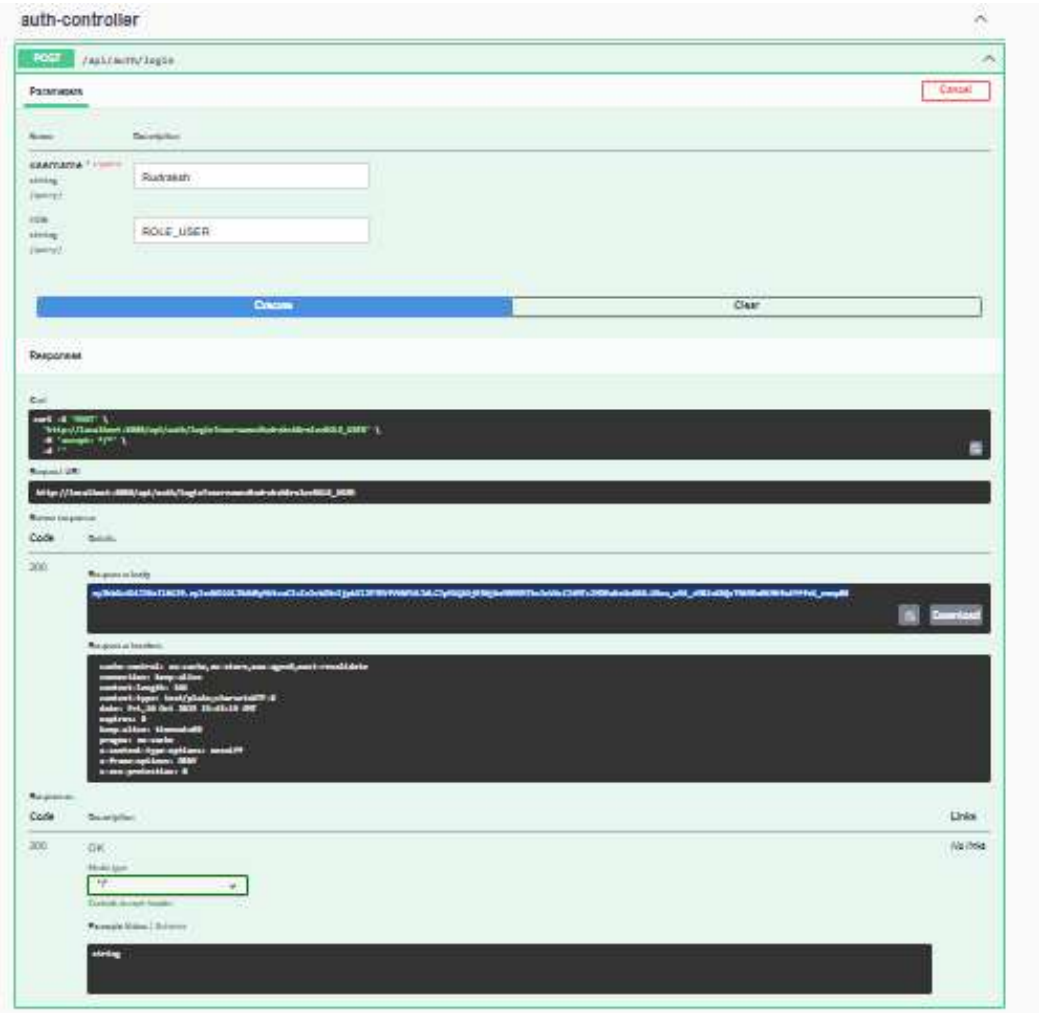
```java
package com.assignment4b.controller;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;
import org.springframework.web.bind.annotation.*;
import java.security.Key;
import java.util.Date;
import java.util.List;

@RestController
@RequestMapping("/api/auth")
public class AuthController {

    private final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

    @PostMapping("/login")
    public String login(@RequestParam String username,
                        @RequestParam(defaultValue = "ROLE_USER") String role) {
        return Jwts.builder()
                .setSubject(username)
                .claim("roles", List.of(role)) // embedding roles into token
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + 3600000)) // 1 hour expiry
                .signWith(key)
                .compact();
    }

    public Key getKey() {
        return key;
    }
}
```

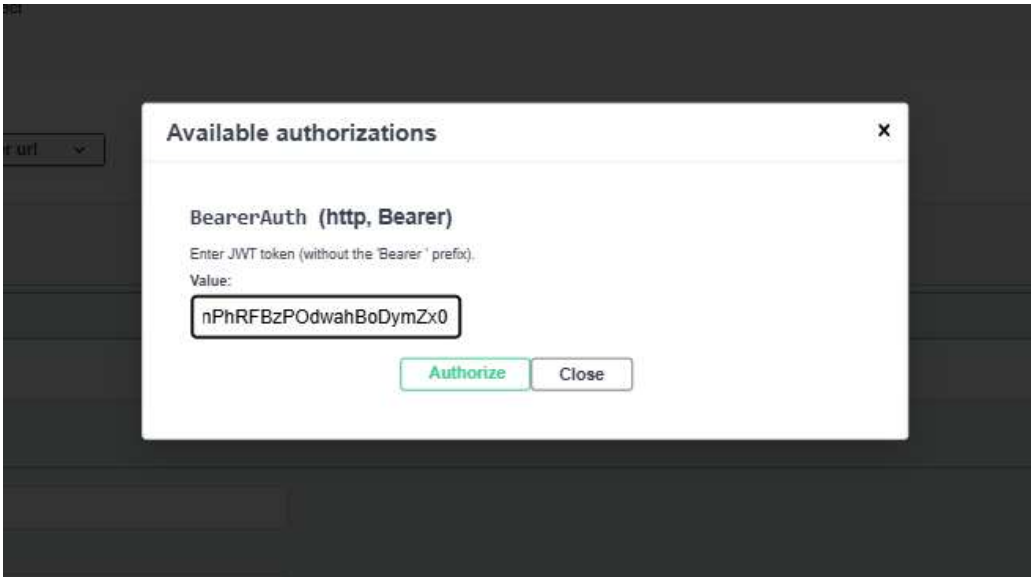
Secure Controller:

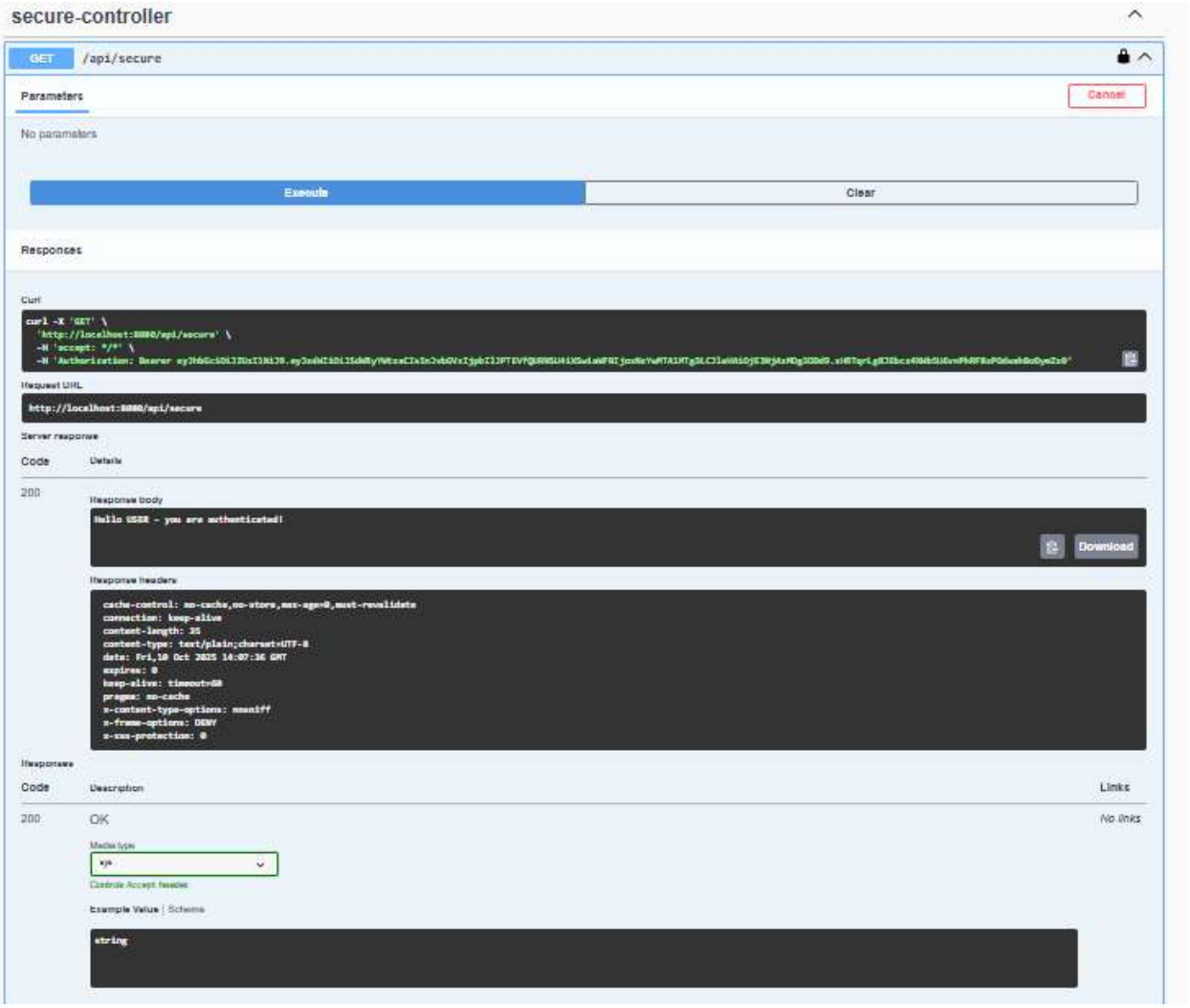
SecurityConfig.java | AuthController.java | SecureController.java

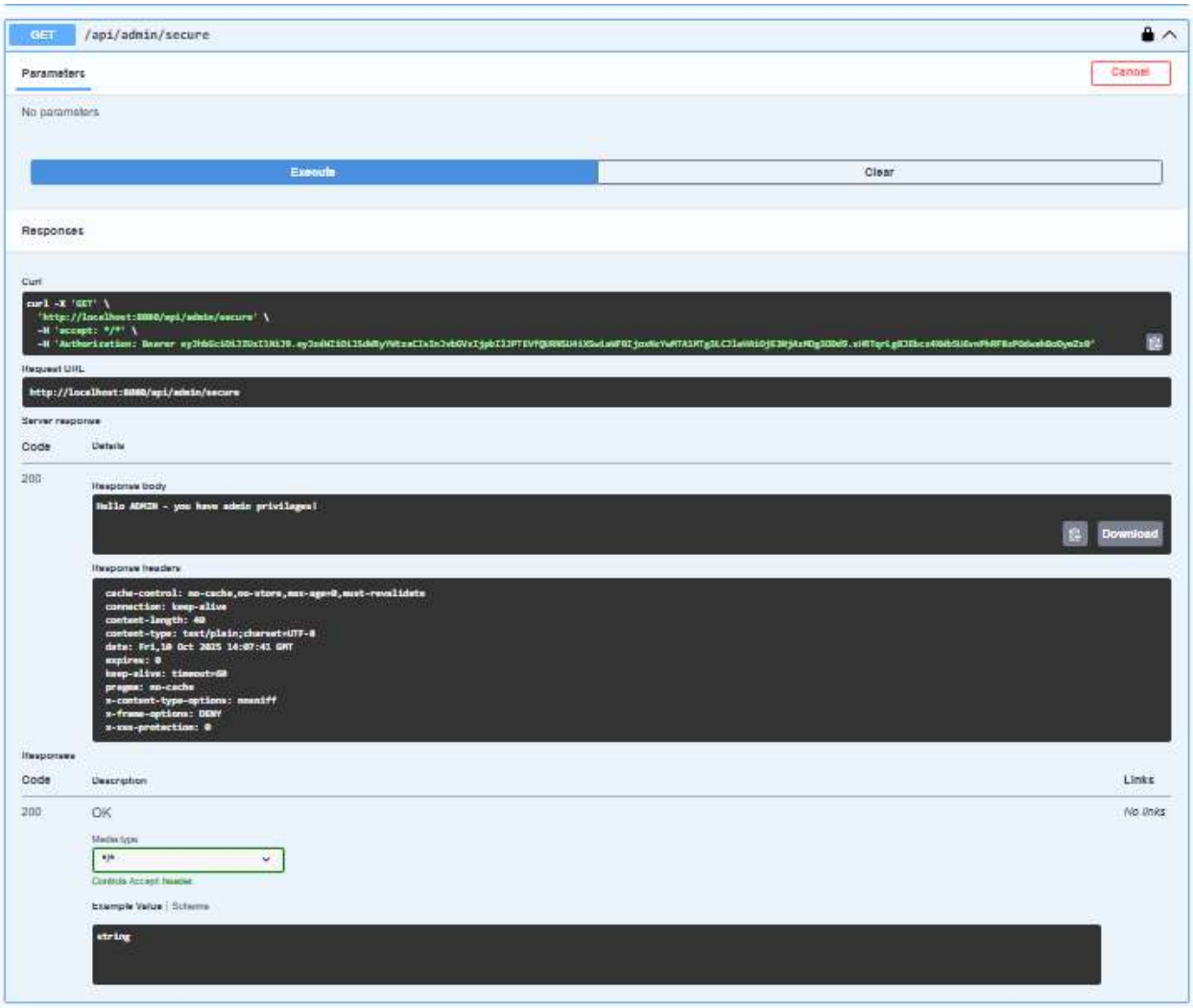
```java
package com.assignment4b.controller;

import org.springframework.web.bind.annotation.*;
@RestController
@RequestMapping("/api")
public class SecureController {
 @GetMapping("/secure")
 public String userEndpoint() {
 return "Hello USER - you are authenticated!";
 }
 @GetMapping("/admin/secure")
 public String adminEndpoint() {
 return "Hello ADMIN - you have admin privileges!";
 }
}
```

Output:


auth-controller
POST
Parameters
Cancel
Rudraksh
ROLE_USER
Execute
Clear
Responses
Code
200
Download
Code
Links
200
OK
string


Available authorizations
BearerAuth (http, Bearer)
Enter JWT token (without the 'Bearer ' prefix).
Value:
nPhRFBzPOdwahBoDymZx0
Authorize
Close


secure-controller
GET
/api/secure
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:8080/api/secure' \
-H 'accept: */*' \
-H 'Authorization: Bearer
Request URL
http://localhost:8080/api/secure
Server response
Code
Details
200
Response body
Hello USER - you are authenticated!
Download
Response headers
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-length: 35
content-type: text/plain;charset=UTF-8
date: Fri,10 Oct 2025 14:07:36 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
Responses
Code
Description
Links
200
OK
No links
Media type
/
Controls Accept header.
Example Value | Schema
string


GET
/api/admin/secure
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:8080/api/admin/secure' \
-H 'accept: */*' \
-H 'Authorization: Bearer
Request URL
http://localhost:8080/api/admin/secure
Server response
Code
Details
200
Response body
Hello ADMIN - you have admin privileges!
Download
Response headers
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-length: 40
content-type: text/plain;charset=UTF-8
date: Fri,10 Oct 2025 14:07:41 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
Responses
Code
Description
Links
200
OK
No links
Media type
/
Controls Accept header.
Example Value | Schema
string