SuperSet ID : 6314413 Shaik Balaji Mahammad Rafi

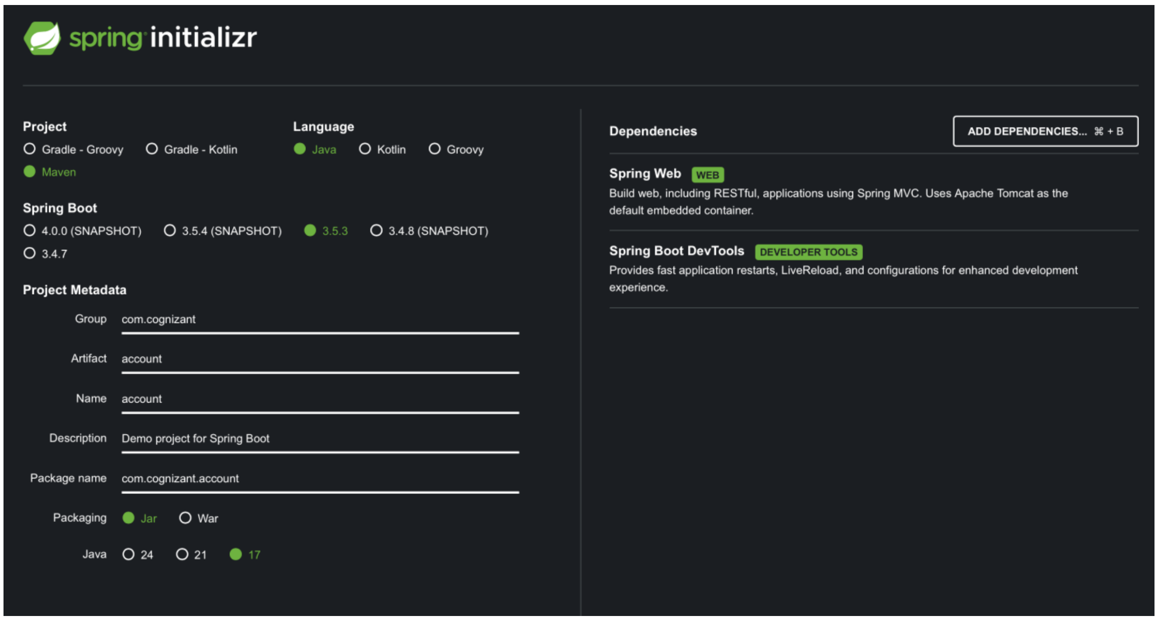
HANDSON EXERCISES - WEEK 5

Skill : Microservices with Spring Boot 3 and Spring Cloud

File : 2. Microservices with API gateway

Exercise : Creating Microservices for account and loan

account-service :



spring initializr

Project: ○ Gradle - Groovy ○ Gradle - Kotlin ● Maven

Language: ● Java ○ Kotlin ○ Groovy

Spring Boot: ○ 4.0.0 (SNAPSHOT) ○ 3.5.4 (SNAPSHOT) ● 3.5.3 ○ 3.4.8 (SNAPSHOT) ○ 3.4.7

Project Metadata

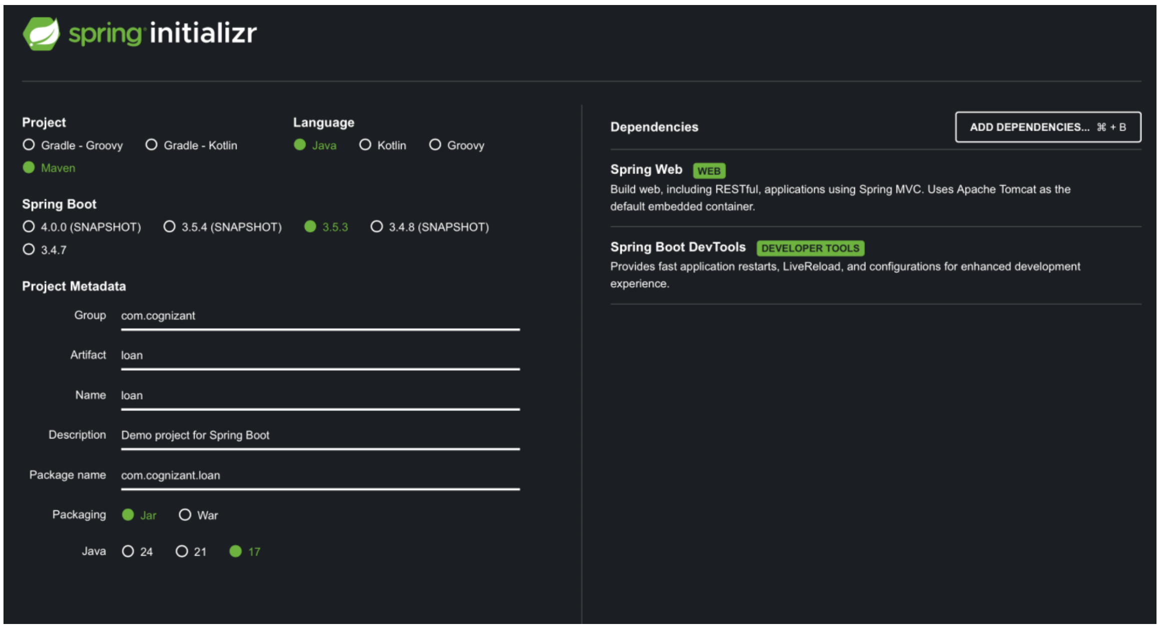
Group	com.cognizant
Artifact	account
Name	account
Description	Demo project for Spring Boot
Package name	com.cognizant.account
Packaging	● Jar ○ War
Java	○ 24 ○ 21 ● 17

Dependencies — ADD DEPENDENCIES... ⌘ + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

loan-service :



spring initializr

Project: ○ Gradle - Groovy ○ Gradle - Kotlin ● Maven

Language: ● Java ○ Kotlin ○ Groovy

Spring Boot: ○ 4.0.0 (SNAPSHOT) ○ 3.5.4 (SNAPSHOT) ● 3.5.3 ○ 3.4.8 (SNAPSHOT) ○ 3.4.7

Project Metadata

Group	com.cognizant
Artifact	loan
Name	loan
Description	Demo project for Spring Boot
Package name	com.cognizant.loan
Packaging	● Jar ○ War
Java	○ 24 ○ 21 ● 17

Dependencies — ADD DEPENDENCIES... ⌘ + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

AccountController.java :

```
package com.cognizant.account.controller;

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

import java.util.Map;

@RestController
@RequestMapping("/accounts")
public class AccountController {

    @GetMapping("/{number}")
    public Map<String, Object> getAccountDetails(@PathVariable String number) {
        return Map.of(
                "number", number,
                "type", "savings",
                "balance", 234343
        );
    }
}
```

LoanController.java :

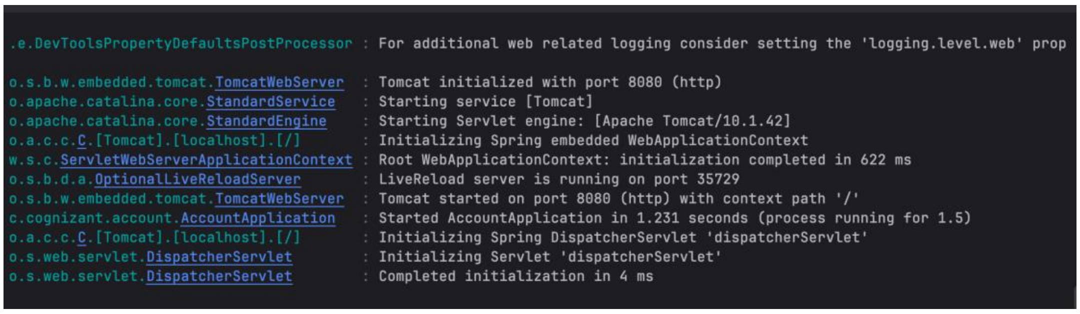
```
package com.cognizant.loan.controller;

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

import java.util.Map;

@RestController
@RequestMapping("/loans")
public class LoanController {

    @GetMapping("/{number}")
    public Map<String, Object> getLoanDetails(@PathVariable String number) {
        return Map.of(
                "number", number,
                "type", "car",
                "loan", 400000,
                "emi", 3258,
                "tenure", 18
        );
    }
}
```

Loan - application.properties :

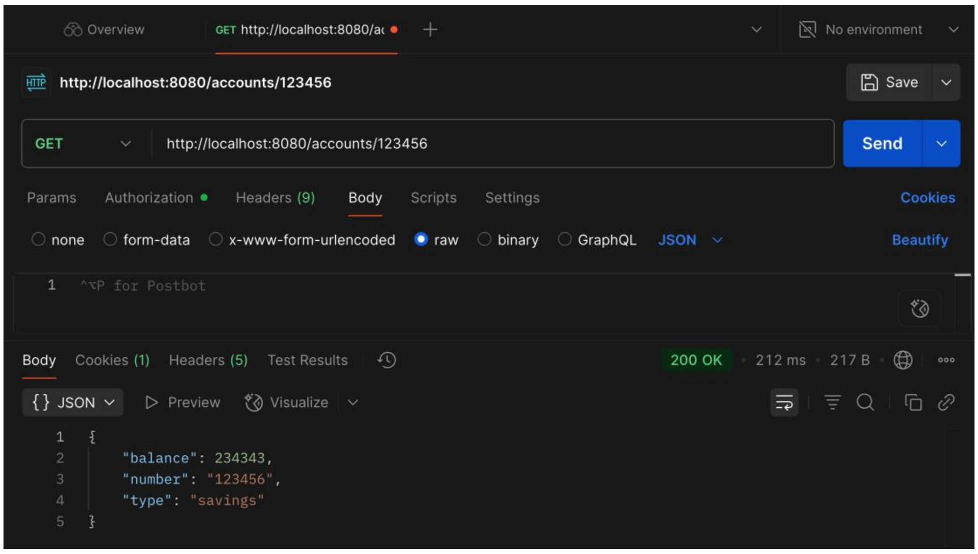
```
spring.application.name=loan
server.port=8081
```

OUTPUT :

Account : (on port 8080)



```
.e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' prop

o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
o.apache.catalina.core.StandardService   : Starting service [Tomcat]
o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.42]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 622 ms
o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
c.cognizant.account.AccountApplication   : Started AccountApplication in 1.231 seconds (process running for 1.5)
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
o.s.web.servlet.DispatcherServlet        : Completed initialization in 4 ms
```



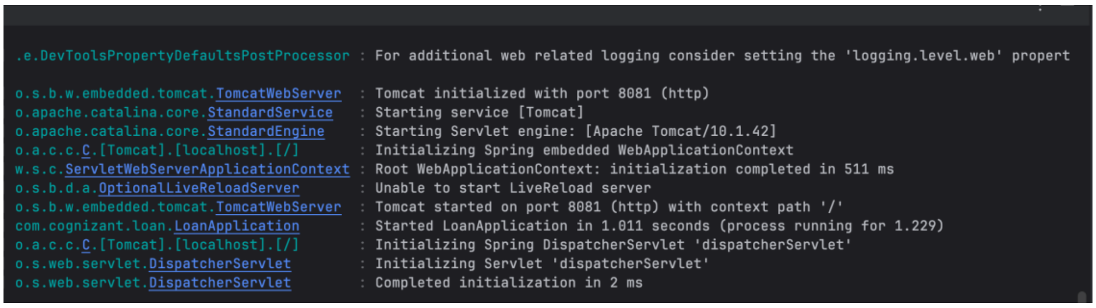
GET http://localhost:8080/accounts/123456

Body: raw, JSON

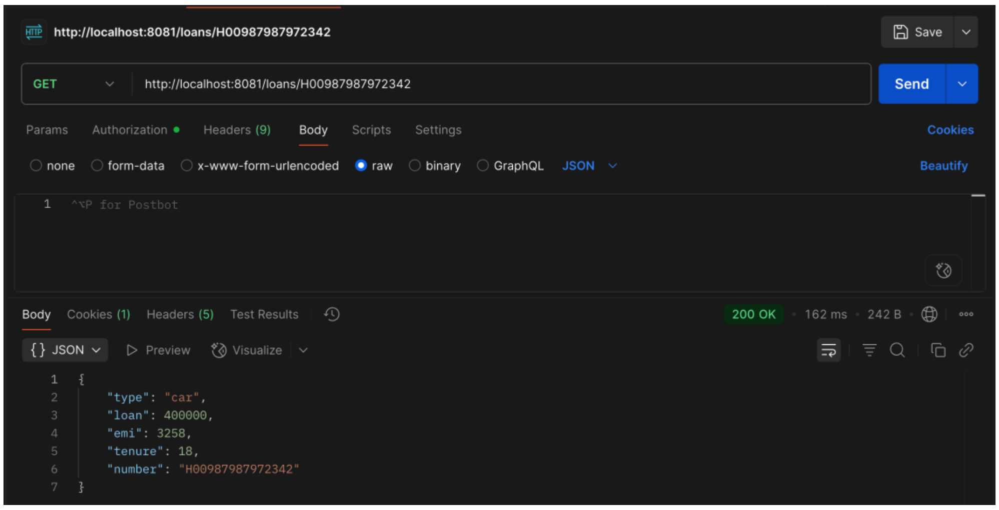
200 OK · 212 ms · 217 B

```
{
    "balance": 234343,
    "number": "123456",
    "type": "savings"
}
```

Loan : (on port 8081)



```
.e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' propert

o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8081 (http)
o.apache.catalina.core.StandardService   : Starting service [Tomcat]
o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.42]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 511 ms
o.s.b.d.a.OptionalLiveReloadServer       : Unable to start LiveReload server
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8081 (http) with context path '/'
com.cognizant.loan.LoanApplication       : Started LoanApplication in 1.011 seconds (process running for 1.229)
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
```



GET http://localhost:8081/loans/H00987987972342

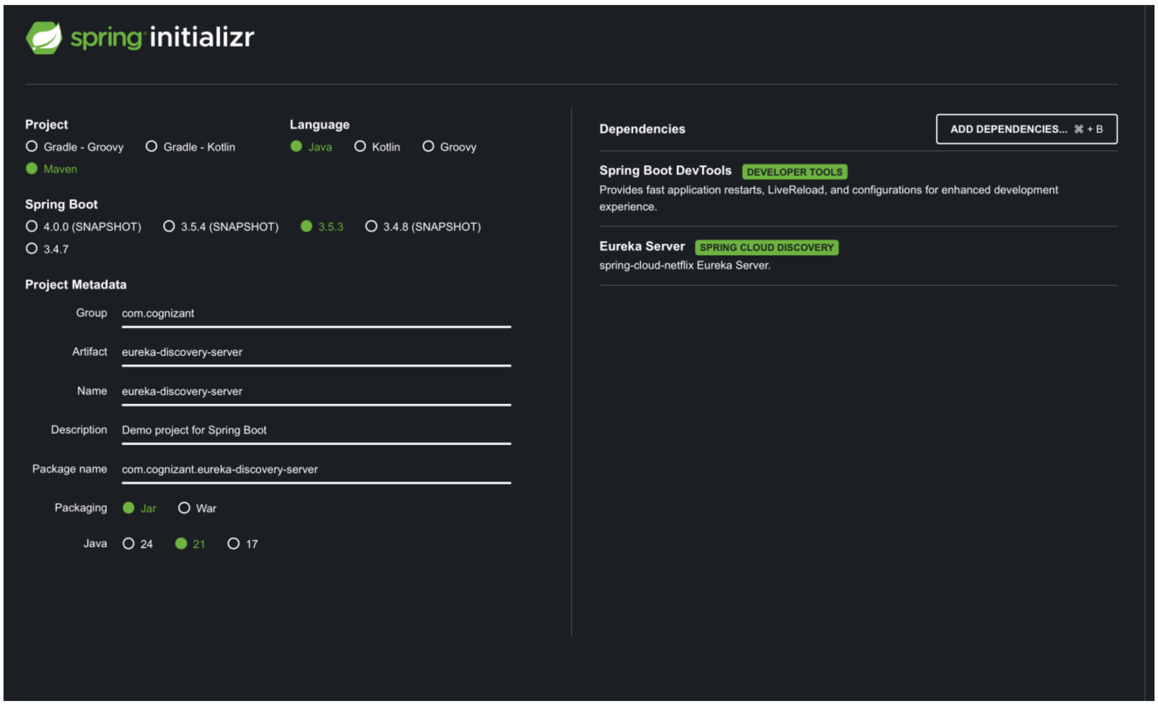
Body: raw, JSON

200 OK · 162 ms · 242 B

```
{
    "type": "car",
    "loan": 400000,
    "emi": 3258,
    "tenure": 18,
    "number": "H00987987972342"
}
```

Exercise : Create Eureka Discovery Server and register microservices

eureka-discovery-server :



spring initializr

Project
Gradle - Groovy   Gradle - Kotlin
[✓ Maven]

Language
[✓ Java]   Kotlin   Groovy

Spring Boot
4.0.0 (SNAPSHOT)   3.5.4 (SNAPSHOT)   [✓ 3.5.3]   3.4.8 (SNAPSHOT)
3.4.7

Project Metadata

Group	com.cognizant
Artifact	eureka-discovery-server
Name	eureka-discovery-server
Description	Demo project for Spring Boot
Package name	com.cognizant.eureka-discovery-server
Packaging	[✓ Jar] War
Java	24 [✓ 21] 17

Dependencies

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Eureka Server SPRING CLOUD DISCOVERY
spring-cloud-netflix Eureka Server.

pom.xml :

```
<parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>3.1.5</version>
   <relativePath/>
</parent>

<properties>
   <java.version>17</java.version>
   <spring-cloud.version>2022.0.4</spring-cloud.version>
</properties>

<dependencies>
   <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
   </dependency>

   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
   </dependency>
```

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>

</dependencies>


<dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>${spring-cloud.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
</dependencyManagement>
```

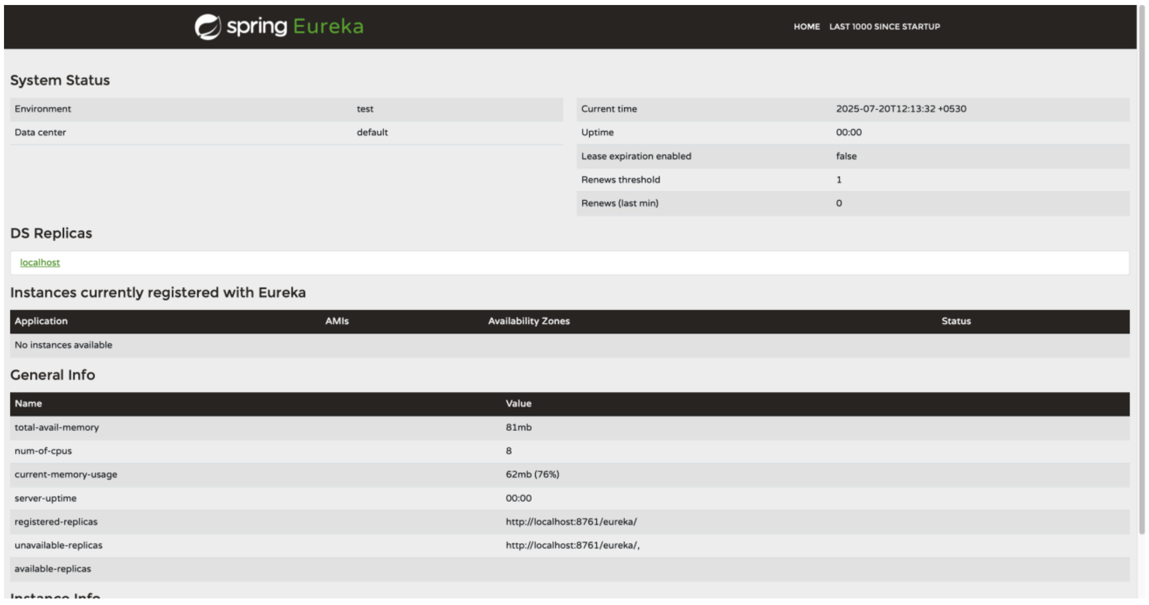
EurekaDiscoveryServerApplication.java :

```
package com.cognizant.eureka_discovery_server;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class EurekaDiscoveryServerApplication {
    public static void main(String[] args) {
      SpringApplication.run(EurekaDiscoveryServerApplication.class, args);
    }
}
```

resources/application.properties :

```
spring.application.name=eureka-discovery-server
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
logging.level.com.netflix.eureka=OFF
logging.level.com.netflix.discovery=OFF
```

OUTPUT ~ No services are registered yet



System Status

Environment	test	Current time	2025-07-20T12:13:32 +0530
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

General Info

Name	Value
total-avail-memory	81mb
num-of-cpus	8
current-memory-usage	62mb (76%)
server-uptime	00:00
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/,
available-replicas	

pom.xml (in both account and loan microservices)

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

AccountServiceApplication.java :

```
package com.cognizant.account;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@EnableDiscoveryClient
@SpringBootApplication
public class AccountApplication {

    public static void main(String[] args) {
      SpringApplication.run(AccountApplication.class, args);
    }

}
```

resources/applciation.properties :

```
server.port=8080
spring.application.name=account-service
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

LoanServiceApplication.java :

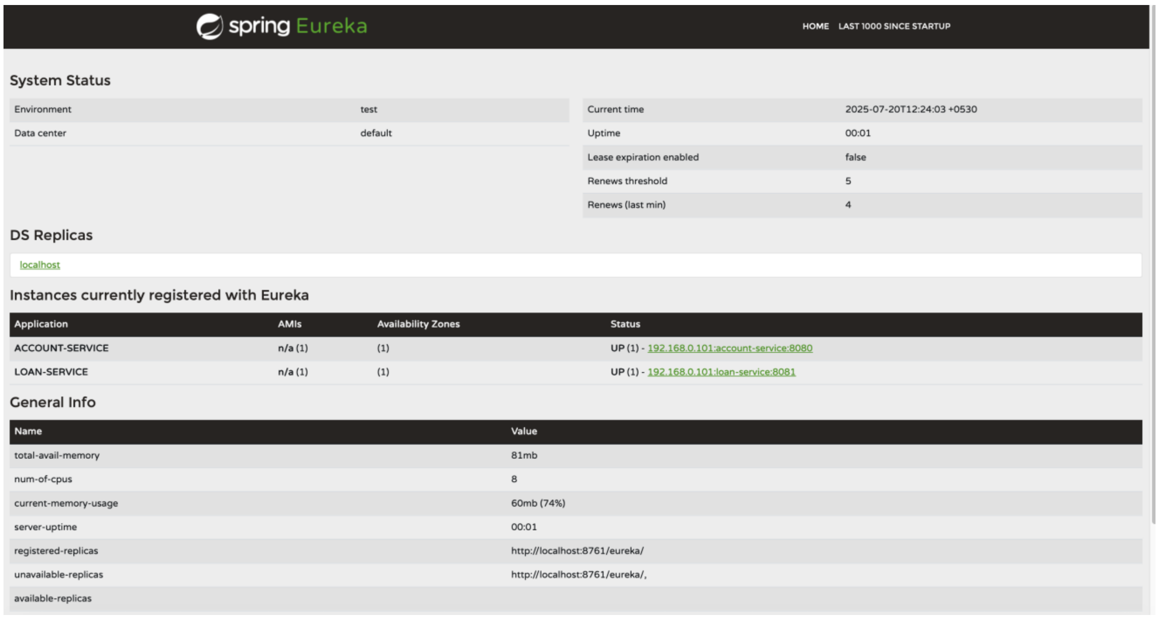
```java
package com.cognizant.loan;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@EnableDiscoveryClient
@SpringBootApplication
public class LoanApplication {

    public static void main(String[] args) {
        SpringApplication.run(LoanApplication.class, args);
    }

}
```

resources/applciation.properties :

```
server.port=8081
spring.application.name=loan-service
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

OUTPUT :

Eureka Dashboard ~ http://localhost:8761



System Status

Environment	test	Current time	2025-07-20T12:24:03 +0530
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	5
		Renews (last min)	4

DS Replicas

localhost

Instances currently registered with Eureka

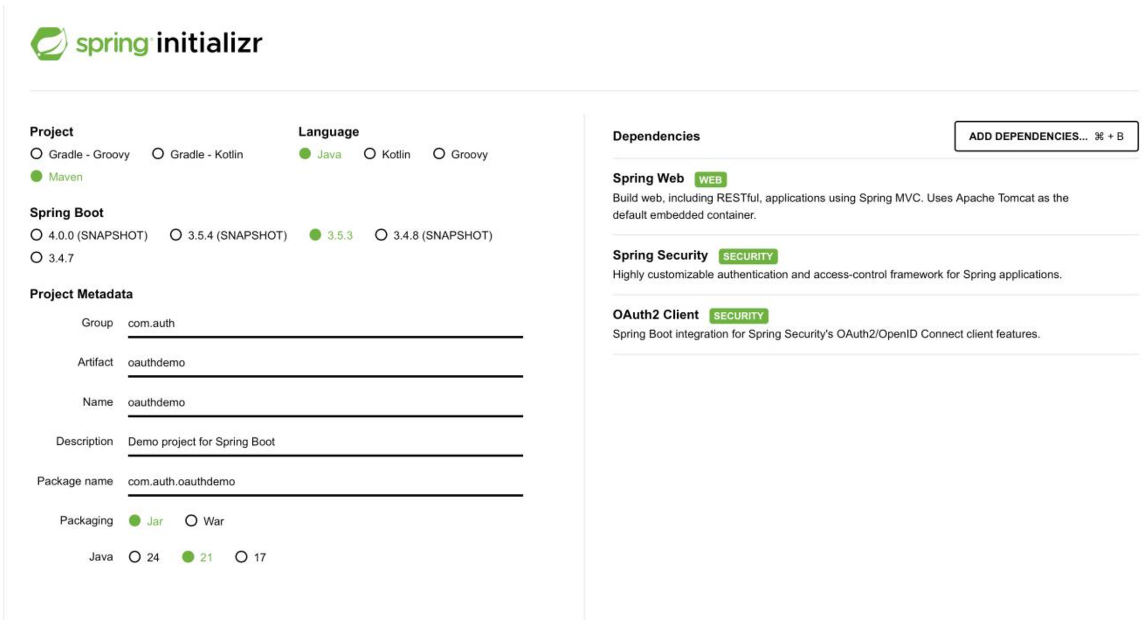
Application	AMIs	Availability Zones	Status
ACCOUNT-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.101:account-service:8080
LOAN-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.101:loan-service:8081

General Info

Name	Value
total-avail-memory	81mb
num-of-cpus	8
current-memory-usage	60mb (74%)
server-uptime	00:01
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/,
available-replicas	

File : 0. Sample Microservices exercises

Exercise 1 : Implementing Centralized Authentication with OAuth 2.1/OIDC


spring initializr

Project
Gradle - Groovy | Gradle - Kotlin | [✓ Maven]

Language
[✓ Java] | Kotlin | Groovy

Spring Boot
4.0.0 (SNAPSHOT) | 3.5.4 (SNAPSHOT) | [✓ 3.5.3] | 3.4.8 (SNAPSHOT) | 3.4.7

Project Metadata

Group	com.auth
Artifact	oauthdemo
Name	oauthdemo
Description	Demo project for Spring Boot
Package name	com.auth.oauthdemo
Packaging	[✓ Jar] War
Java	24 [✓ 21] 17

Dependencies — ADD DEPENDENCIES... ⌘ + B

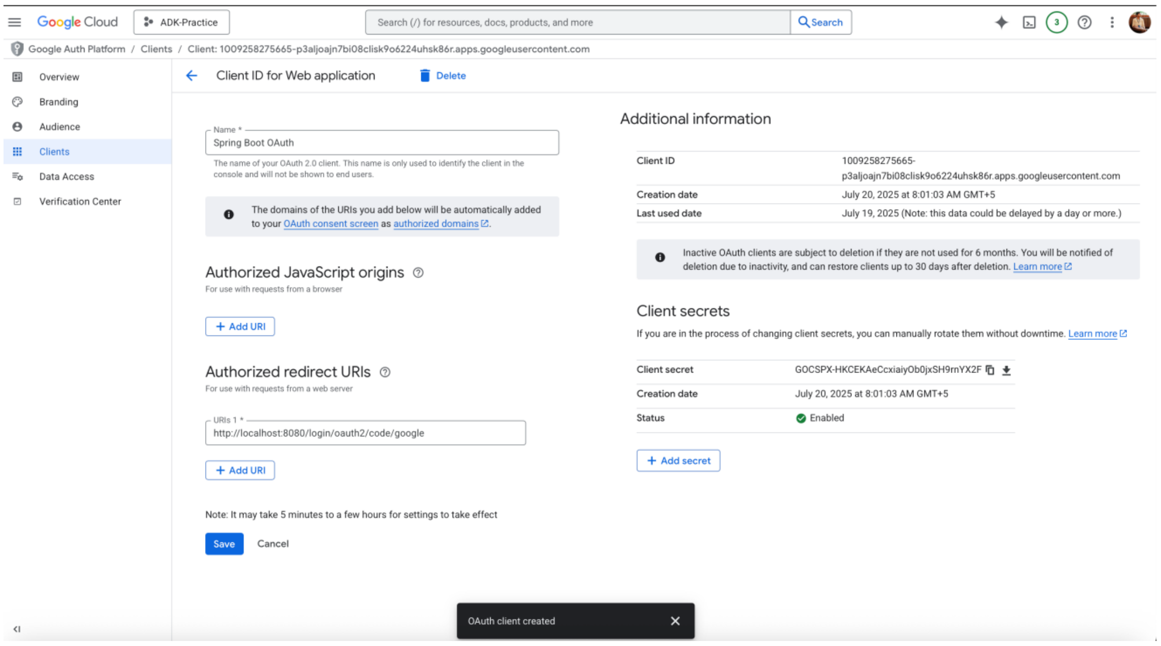
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Security SECURITY
Highly customizable authentication and access-control framework for Spring applications.

OAuth2 Client SECURITY
Spring Boot integration for Spring Security's OAuth2/OpenID Connect client features.

Google Cloud Console Setup :

- Enabled OAuth Content Screen and set user type to External
- Created OAuth 2.o Credentials.



Google Auth Platform / Clients / Client: 1009258275665-p3aljoajn7bi08clisk9o6224uhsk86r.apps.googleusercontent.com

Client ID for Web application — Delete

Name *: Spring Boot OAuth
The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins
For use with requests from a browser
+ Add URI

Authorized redirect URIs
For use with requests from a web server
URIs 1 *: http://localhost:8080/login/oauth2/code/google
+ Add URI

Note: It may take 5 minutes to a few hours for settings to take effect

Save Cancel

Additional information

Client ID	1009258275665-p3aljoajn7bi08clisk9o6224uhsk86r.apps.googleusercontent.com
Creation date	July 20, 2025 at 8:01:03 AM GMT+5
Last used date	July 19, 2025 (Note: this data could be delayed by a day or more.)

Inactive OAuth clients are subject to deletion if they are not used for 6 months. You will be notified of deletion due to inactivity, and can restore clients up to 30 days after deletion. Learn more

Client secrets
If you are in the process of changing client secrets, you can manually rotate them without downtime. Learn more

Client secret	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
Creation date	July 20, 2025 at 8:01:03 AM GMT+5
Status	✓ Enabled

+ Add secret

OAuth client created ✕

resources/application.yml :

```
spring:
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: 1009258275665-
p3aljoajn7bi08clisk9o6224uhsk86r.apps.googleusercontent.com
            client-secret: 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
            scope: openid, profile, email
            redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"
            client-name: Google
        provider:
          google:
            authorization-uri: https://accounts.google.com/o/oauth2/auth
            token-uri: https://oauth2.googleapis.com/token
            user-info-uri: https://openidconnect.googleapis.com/v1/userinfo
            user-name-attribute: sub
```

com.auth.oauthdemo.config/SecurityConfig.java :

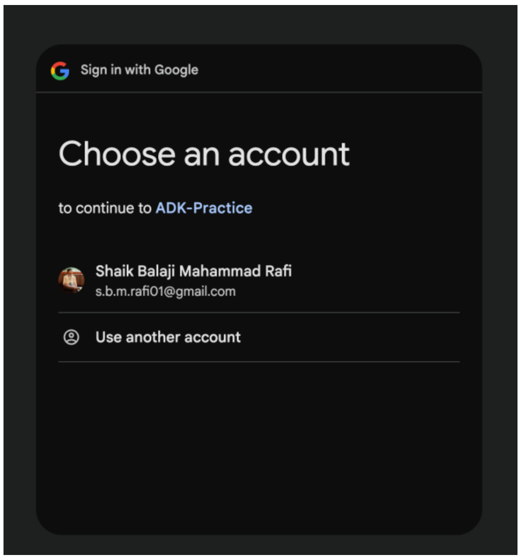
```java
package com.example.oauthdemo.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests(authorize ->
                authorize.anyRequest().authenticated()
            )
            .oauth2Login(); // enables OAuth2 login

        return http.build();
    }
}
```

com.auth.oauthdemo.controller/UserController.java :

```java
package com.auth.oauthdemo.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.security.Principal;
```

```java
@RestController
public class UserController {
   @GetMapping("/user")
   public Principal user(Principal principal) {
      return principal;
   }
}
```

OUTPUT : **http://localhost:8080/user**



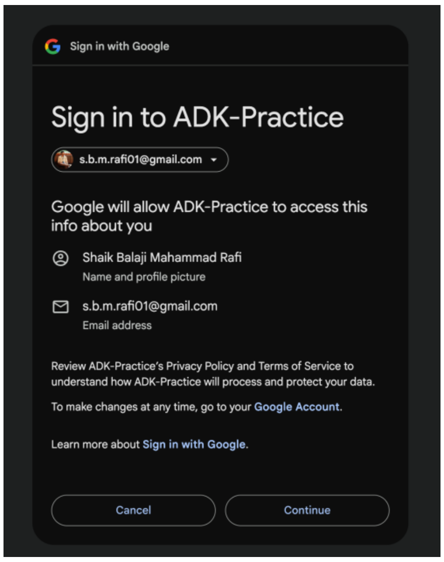
Sign in with Google

Choose an account

to continue to ADK-Practice

Shaik Balaji Mahammad Rafi
s.b.m.rafi01@gmail.com

Use another account



Sign in with Google

Sign in to ADK-Practice

s.b.m.rafi01@gmail.com

Google will allow ADK-Practice to access this info about you

Shaik Balaji Mahammad Rafi
Name and profile picture

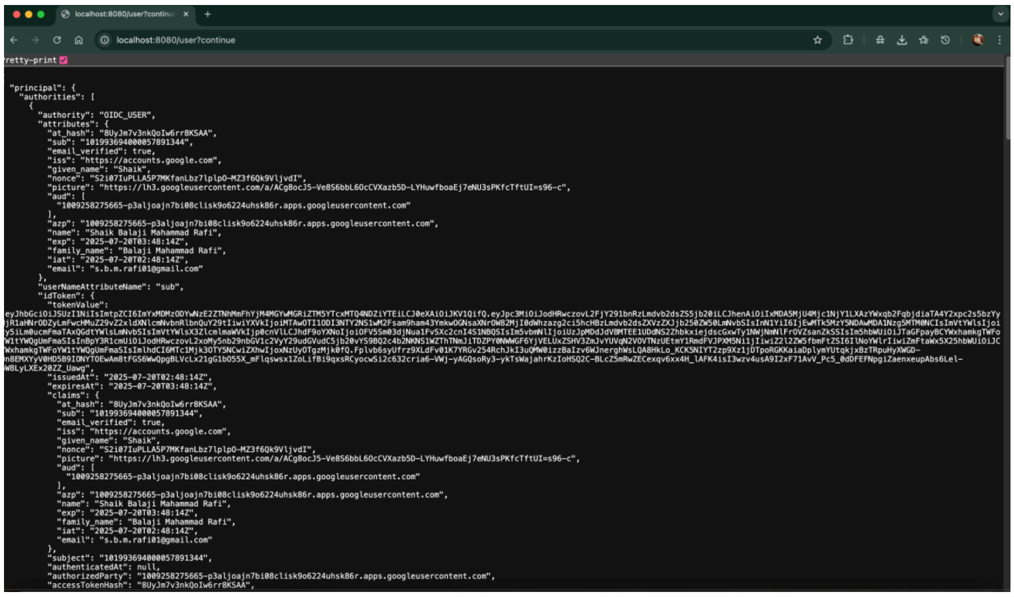
s.b.m.rafi01@gmail.com
Email address

Review ADK-Practice's Privacy Policy and Terms of Service to understand how ADK-Practice will process and protect your data.

To make changes at any time, go to your Google Account.

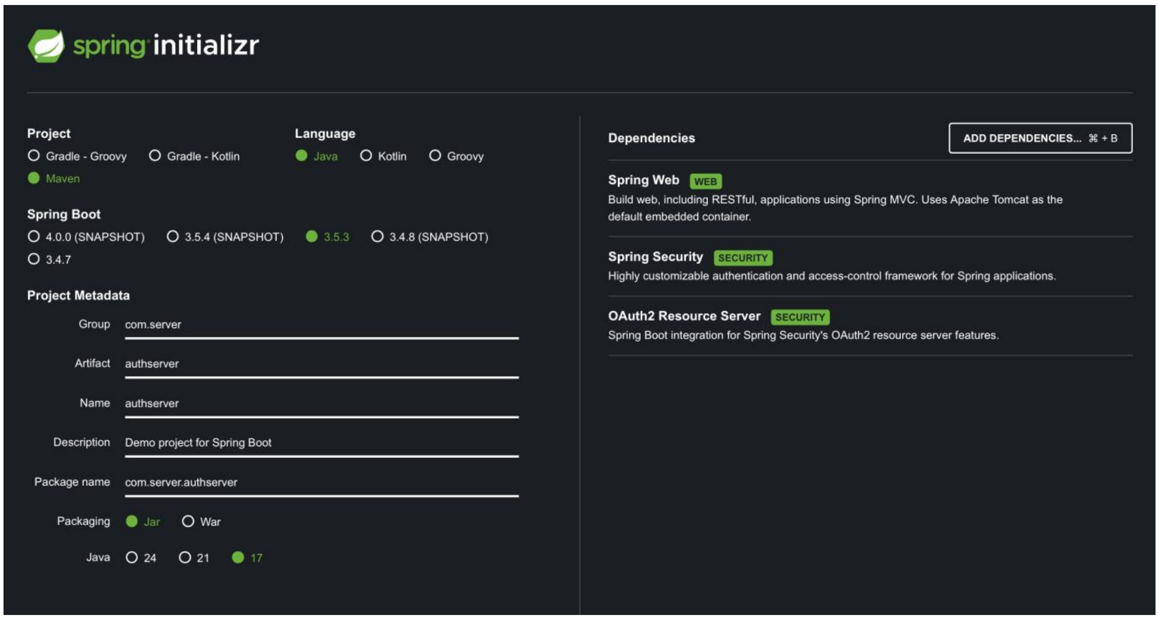
Learn more about Sign in with Google.

JSON Data :



```
"principal": {
  "authorities": [
    {
      "authority": "OIDC_USER",
      "attributes": {
        "at_hash": "8UyJm7v3nkQoIw6rr8KSAA",
        "sub": "101993694000057891344",
        "email_verified": true,
        "iss": "https://accounts.google.com",
        "given_name": "Shaik",
        "nonce": "S2i07IuPLLA5P7MKfanLbz7lplpO-MZ3f6Qk9VljvdI",
        "picture": "https://lh3.googleusercontent.com/a/ACg8ocJ5-Ve8S6bbL6OcCVXazb5D-LYHuwfboaEj7eNU3sPKfcTftUI=s96-c",
        "aud": [
          "1009258275665-p3aljoajn7bi08clisk9o6224uhsk86r.apps.googleusercontent.com"
        ],
        "azp": "1009258275665-p3aljoajn7bi08clisk9o6224uhsk86r.apps.googleusercontent.com",
        "name": "Shaik Balaji Mahammad Rafi",
        "exp": "2025-07-20T03:48:14Z",
        "family_name": "Balaji Mahammad Rafi",
        "iat": "2025-07-20T02:48:14Z",
        "email": "s.b.m.rafi01@gmail.com"
      },
      "userNameAttributeName": "sub",
      "idToken": {
        "tokenValue":
eyJhbGciOiJSUzI1NiIsImtpZCI6ImYxMDMzODYwNzE2ZTNhMmFhYjM4MGYwMGRiZTM5YTcxMTQ4NDZiYTEiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iLCJhenAiOiIxMDA5MjU4Mjc1NjY1LXAzYWxqb2FqbjdiaTA4Y2xpc2s5bzYy
jR1aHNrODZyLmFwcHMuZ29vZ2xldXNlcmNvbnRlbnQuY29tIiwiYXVkIjoiMTAwOTI1ODI3NTY2NS1wM2Fsam9ham43YmkwOGNsaXNrOW82MjI0dWhzazg2ci5hcHBzLmdvb2dsZXVzZXJjb250ZW50LmNvbSIsInN1YiI6IjEwMTk5MzY5NDAwMDA1Nzg5MTM0NCIsImVtYWlsIjoi
y5iLm0ucmFmaTAxQGdtYWlsLmNvbSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJhdF9oYXNoIjoiOFV5Sm03djNua1FvSXc2cnI4S1NBQSIsIm5vbmNlIjoiUzJpMDdJdVBMTEE1UDdNS2ZhbkxiejdscGxwTy1NWjNmNlFrOVZsanZkSSIsIm5hbWUiOiJTaGFpayBCYWxhamkgTWFo
W1tYWQgUmFmaSIsInBpY3R1cmUiOiJodHRwczovL2xoMy5nb29nbGV1c2VyY29udGVudC5jb20vYS9BQ2c4b2NKNS1WZThTNmJiTDZPY0NWWGF6YjVELUxZSHV3ZmJvYUVqN2VOVTNzUEtmY1RmdFVJPXM5Ni1jIiwiZ2l2ZW5fbmFtZSI6IlNoYWlrIiwiZmFtaWx5X25hbWUiOiJC
WxhamkgTWFoYW1tYWQgUmFmaSIsImlhdCI6MTc1Mjk3OTY5NCwiZXhwIjoxNzUyOTgzMjk0fQ.Fplvb6syUfrz9XLdFv01K7YRGv254RchJkI3uQMW0izzBaIzv6WJnerghWsLQA8HkLo_KCK5NIYT2zp9Xz1jDTpoRGKKaiaDplymYUtqkjxBzTRpuHyXWGD-
n8EMXYyV0HD5B9IONYTOEwAm8tFGS6WwQpgBLVcLx21gG1bO55X_mFlqswsx1ZoLifBi9qxsRCyocwSi2c632cria6-VWj-yAGQsoRy3-ykTsWajahrKzIoHSQ2C-BLcZ5mRwZECexqv6xx4H_lAFK4isI3wzv4usA9I2xF71AvV_Pc5_0dDFEFNpgiZaenxeupAbs6Lel-
W8LyLXEx20ZZ_Uawg",
        "issuedAt": "2025-07-20T02:48:14Z",
        "expiresAt": "2025-07-20T03:48:14Z",
        "claims": {
          "at_hash": "8UyJm7v3nkQoIw6rr8KSAA",
          "sub": "101993694000057891344",
          "email_verified": true,
          "iss": "https://accounts.google.com",
          "given_name": "Shaik",
          "nonce": "S2i07IuPLLA5P7MKfanLbz7lplpO-MZ3f6Qk9VljvdI",
          "picture": "https://lh3.googleusercontent.com/a/ACg8ocJ5-Ve8S6bbL6OcCVXazb5D-LYHuwfboaEj7eNU3sPKfcTftUI=s96-c",
          "aud": [
            "1009258275665-p3aljoajn7bi08clisk9o6224uhsk86r.apps.googleusercontent.com"
          ],
          "azp": "1009258275665-p3aljoajn7bi08clisk9o6224uhsk86r.apps.googleusercontent.com",
          "name": "Shaik Balaji Mahammad Rafi",
          "exp": "2025-07-20T03:48:14Z",
          "family_name": "Balaji Mahammad Rafi",
          "iat": "2025-07-20T02:48:14Z",
          "email": "s.b.m.rafi01@gmail.com"
        },
        "subject": "101993694000057891344",
        "authenticatedAt": null,
        "authorizedParty": "1009258275665-p3aljoajn7bi08clisk9o6224uhsk86r.apps.googleusercontent.com",
        "accessTokenHash": "8UyJm7v3nkQoIw6rr8KSAA",
```

Exercise 2 : Configuring Authorization Servers and Resource Servers



Project: Gradle - Groovy, Gradle - Kotlin, [✓ Maven]

Language: [✓ Java], Kotlin, Groovy

Spring Boot: 4.0.0 (SNAPSHOT), 3.5.4 (SNAPSHOT), [✓ 3.5.3], 3.4.8 (SNAPSHOT), 3.4.7

Project Metadata

Group	com.server
Artifact	authserver
Name	authserver
Description	Demo project for Spring Boot
Package name	com.server.authserver
Packaging	[✓ Jar] War
Java	24 21 [✓ 17]

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Security SECURITY
Highly customizable authentication and access-control framework for Spring applications.

OAuth2 Resource Server SECURITY
Spring Boot integration for Spring Security's OAuth2 resource server features.

resources/application.yml :

```yaml
spring:
  security:
    oauth2:
      resourceserver:
        jwt:
          issuer-uri: https://issuer.example.com
```

com.server.authserver.config/ResourceServerConfig.java :

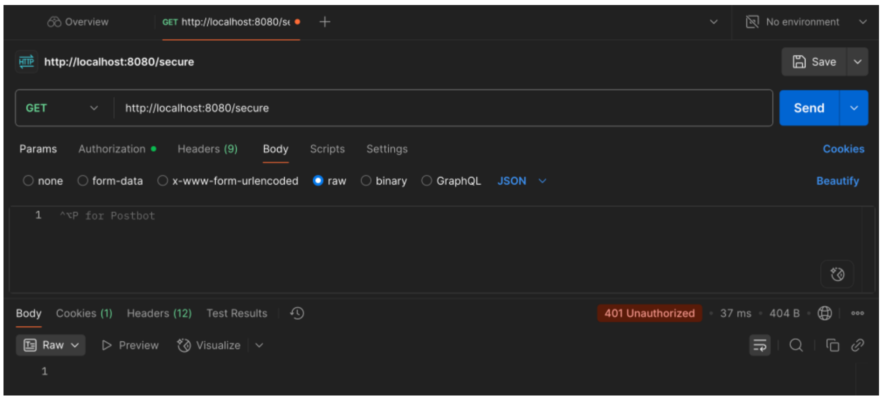
```java
package com.server.authserver.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.oauth2.server.resource.authentication.JwtAuthenticationCo
nverter;

@Configuration
@EnableWebSecurity
public class ResourceServerConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
                .authorizeHttpRequests(auth -> auth
                        .anyRequest().authenticated()
```

```
            )
            .oauth2ResourceServer(oauth2 -> oauth2
                .jwt()
            );

        return http.build();
    }
}
```

com.server.authserver.controller/SecureController.java :

```
package com.example.demo.controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class SecureController {
    @GetMapping("/secure")
    public String secure() {
        return "This is a secure endpoint";
    }
}
```

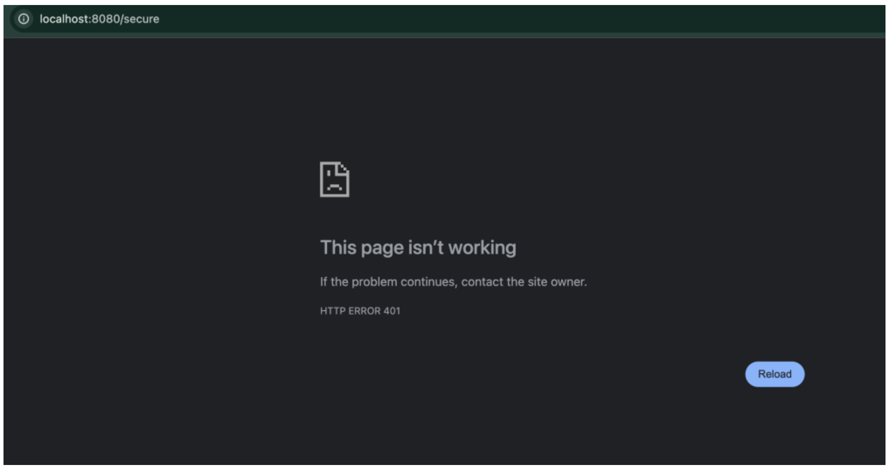
OUTPUT:



GET http://localhost:8080/secure

Authorization: raw, JSON

```
401 Unauthorized · 37 ms · 404 B
```



localhost:8080/secure

This page isn't working

If the problem continues, contact the site owner.

HTTP ERROR 401

Due to my preparation for the upcoming Agentic AI Hackathon by Hack2Skill, I was only able to dedicate this amount of time to practice. Thank you for your understanding.