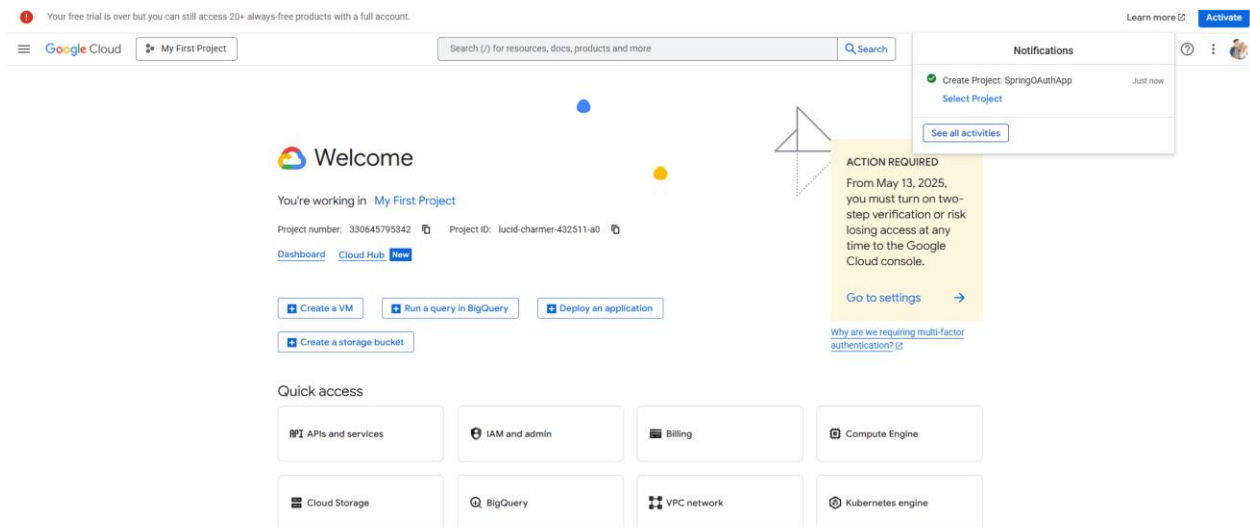
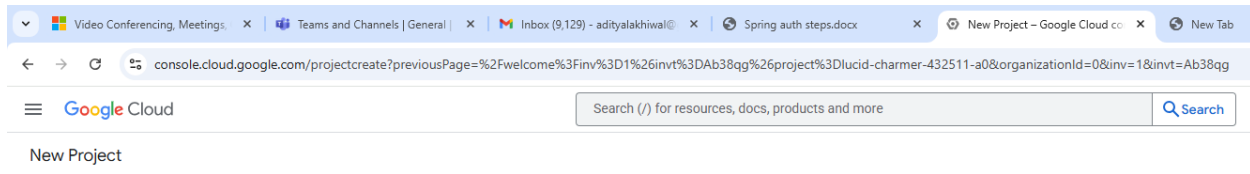


Full Stack Practical 3

Aim: Implement Springboot oauth2 Login Application

Create Project from Google Cloud



Aditya Anilkumar Lakhiwal
MSc CS F004

The screenshot shows the Google Cloud console 'Welcome' page for the project 'SpringOAuthApp'. The page includes a navigation menu, a search bar, and a 'Welcome' message. Below the welcome message, there are links to 'Dashboard', 'Cloud Hub', and 'New'. A 'Go to settings' button is also present. A yellow box on the right side of the page contains an 'ACTION REQUIRED' message: 'From May 13, 2025, you must turn on two-step verification or risk losing access at any time to the Google Cloud console. Go to settings'. Below this, there is a link 'Why are we requiring multi-factor authentication?'. The 'Quick access' section lists various services: 'APIs and services', 'IAM and admin', 'Billing', 'Compute Engine', 'Cloud Storage', 'BigQuery', 'VPC network', and 'Kubernetes engine'. A 'View all products' button is at the bottom. A black notification bar at the bottom of the page states: 'Now viewing project 'SpringOAuthApp' in organisation 'No organisation''.

The screenshot shows the Google Cloud console 'API Library' page. The page has a header with 'Welcome to the API Library' and a search bar. Below the header, there is a 'Filter' section with 'Type to filter'. The main content area is divided into two sections: 'Maps' and 'Machine learning'. The 'Maps' section lists several APIs: 'Maps SDK for Android', 'Maps SDK for iOS', 'Maps JavaScript API', 'Places API', 'Roads API', and 'Directions API'. The 'Machine learning' section lists several APIs: 'Dialogflow API', 'Cloud Vision API', 'Cloud Natural Language API', 'Cloud Speech-to-Text API', 'Cloud Translation API', and 'AI Platform Training & Prediction API'. A black notification bar at the bottom of the page states: 'Now viewing project 'SpringOAuthApp' in organisation 'No organisation''.

Aditya Anilkumar Lakhiwal

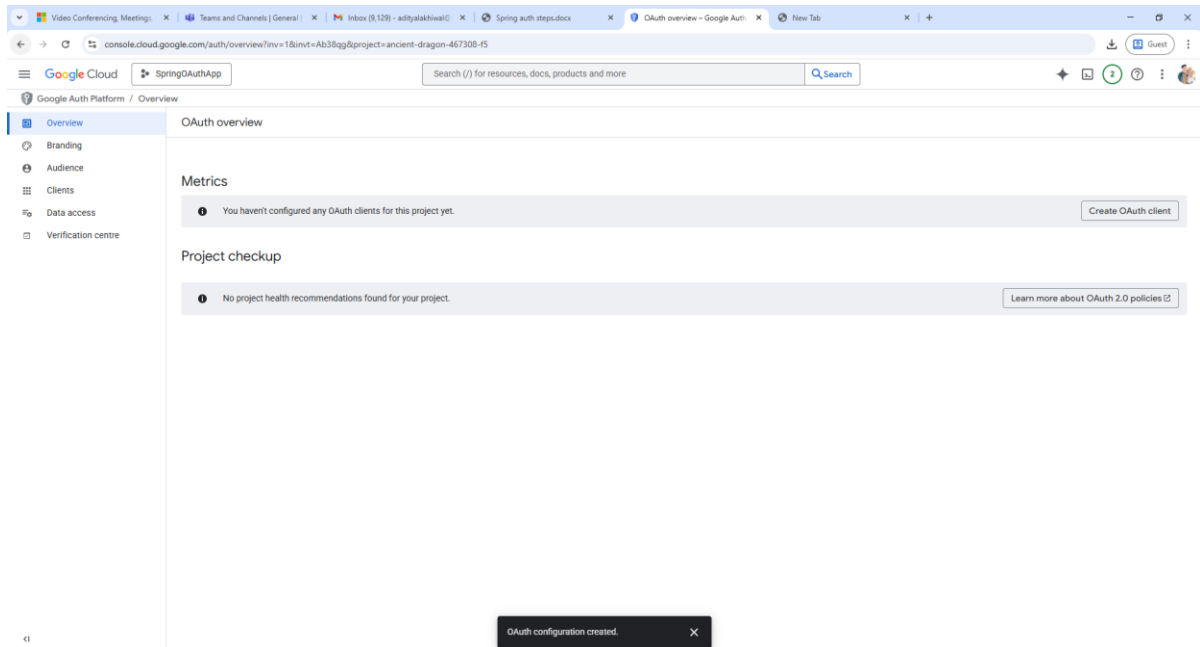
MSc CS F004

The screenshot shows the Google Cloud console interface for configuring a project. The browser tabs include "Video Conferencing, Meetings", "Teams and Channels | General", "Inbox (9,129) - adityalakhiwal@...", "Spring auth steps.docx", "Project configuration - Google", and "New Tab". The address bar shows the URL: `console.cloud.google.com/auth/overview/create?mv=1&mvmt=Ab38ag@project=ancient-dragon-467308-f5`. The Google Cloud logo and "Spring0AuthApp" are visible in the top navigation bar. A search bar is present with the text "Search (/) for resources, docs, products and more". The left sidebar shows the navigation menu with "Overview" selected. The main content area is titled "Project configuration" and displays a progress indicator with four steps: 1. App Information (checked), 2. Audience (active), 3. Contact Information, and 4. Finish. Under the "Audience" step, there are two radio button options: "Internal" (unselected) and "External" (selected). The "Internal" option is described as "Only available to users within your organisation. You will not need to submit your app for verification. [Learn more about user type](#)". The "External" option is described as "Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users that you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. [Learn more about user type](#)". A "Next" button is located below the "External" option. At the bottom of the configuration steps, there are "Create" and "Cancel" buttons. A blue tooltip message in the top right corner of the configuration area states: "You can now search for documentation, resource metadata, tutorials and API keys".

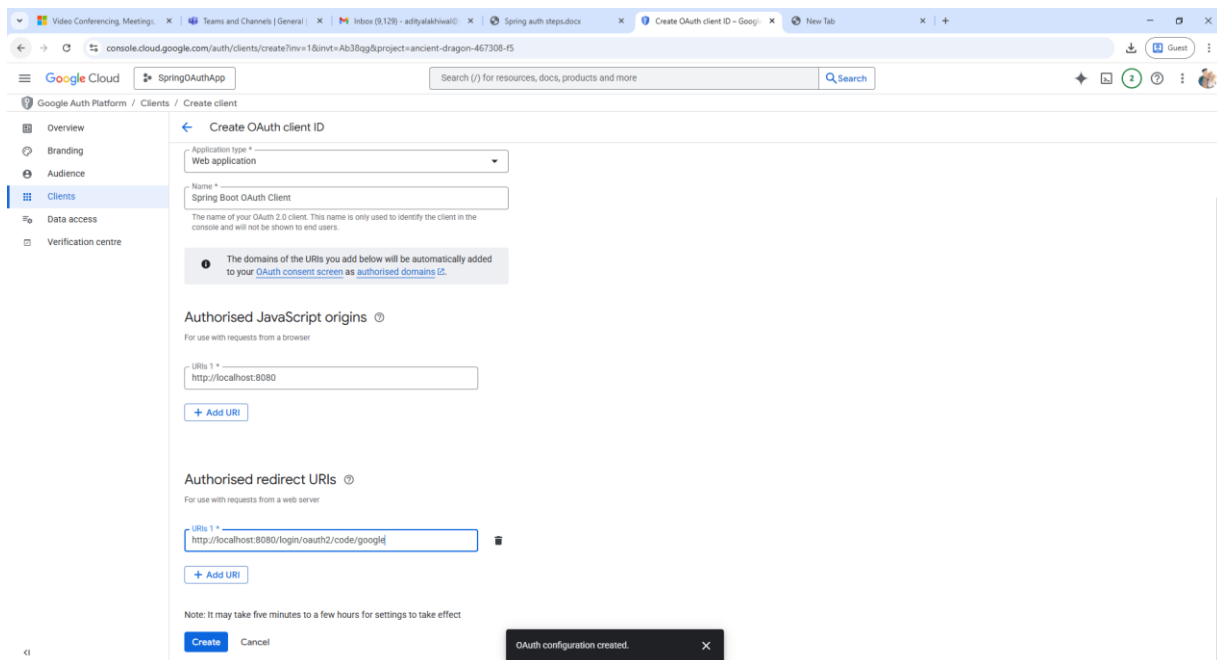
This screenshot shows the same Google Cloud console interface as the previous one, but with the configuration steps completed. The progress indicator now shows all four steps checked: 1. App Information, 2. Audience, 3. Contact Information, and 4. Finish. The "Next" button is no longer visible, and the "Create" button remains at the bottom of the configuration steps. The rest of the interface, including the browser tabs, address bar, and navigation sidebar, remains the same as in the previous screenshot.

Aditya Anilkumar Lakhiwal

MSc CS F004



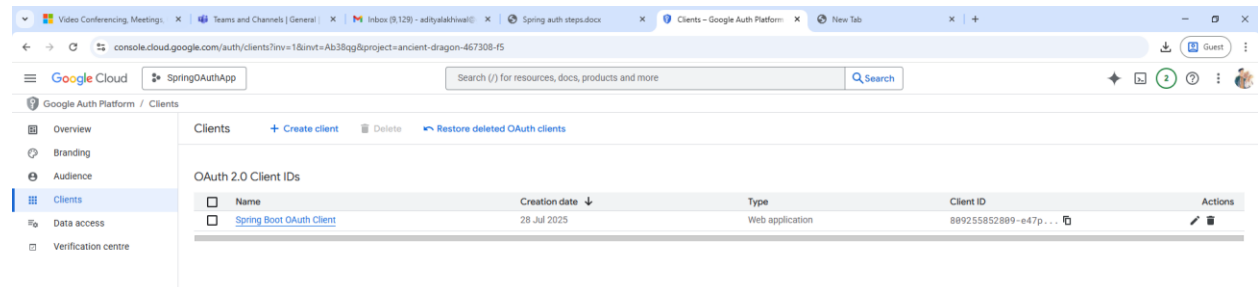
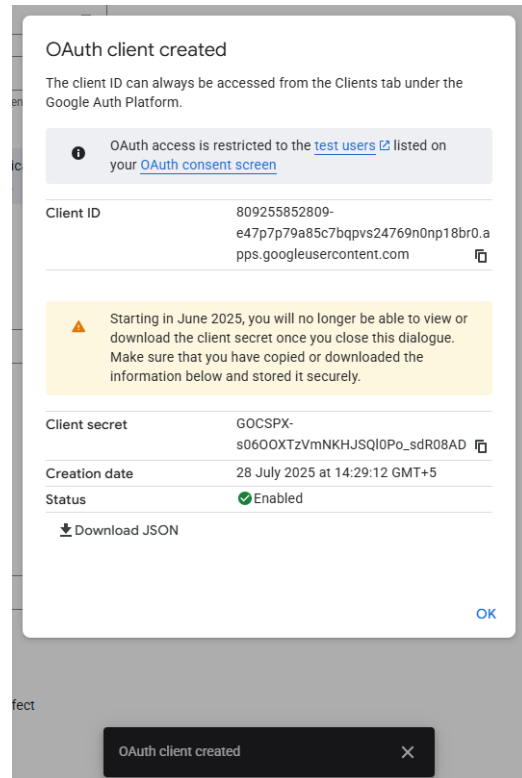
This screenshot shows the Google Cloud OAuth overview page. The browser tabs include 'Video Conferencing, Meetings', 'Teams and Channels | General', 'Inbox (9,129) - adityalakhiwal@...', 'Spring auth step:docx', 'OAuth overview - Google Aut...', and 'New Tab'. The address bar shows the URL 'console.cloud.google.com/auth/overview?nv=1&inv=Ab38qg&project=ancient-dragon-467308-f5'. The Google Cloud logo and 'SpringOAuthApp' are visible in the top navigation bar. A search bar contains the text 'Search (/) for resources, docs, products and more'. The left sidebar shows the 'Overview' menu with sub-items: Branding, Audience, Clients, Data access, and Verification centre. The main content area is titled 'OAuth overview' and contains two sections: 'Metrics' with a message 'You haven't configured any OAuth clients for this project yet.' and a 'Create OAuth client' button; and 'Project health' with a message 'No project health recommendations found for your project.' and a 'Learn more about OAuth 2.0 policies' link. A dark notification box at the bottom center says 'OAuth configuration created.'



This screenshot shows the Google Cloud OAuth client creation page. The browser tabs include 'Video Conferencing, Meetings', 'Teams and Channels | General', 'Inbox (9,129) - adityalakhiwal@...', 'Spring auth step:docx', 'Create OAuth client ID - Goog...', and 'New Tab'. The address bar shows the URL 'console.cloud.google.com/auth/clients/create?nv=1&inv=Ab38qg&project=ancient-dragon-467308-f5'. The Google Cloud logo and 'SpringOAuthApp' are visible in the top navigation bar. A search bar contains the text 'Search (/) for resources, docs, products and more'. The left sidebar shows the 'Clients' menu with sub-items: Overview, Branding, Audience, Clients, Data access, and Verification centre. The main content area is titled 'Create OAuth client ID' and contains several sections: 'Application type' with a dropdown menu set to 'Web application'; 'Name' with a text input field containing 'Spring Boot OAuth Client' and a note '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 authorised domains'; 'Authorised JavaScript origins' with a note 'For use with requests from a browser' and a text input field containing 'http://localhost:8080' and an '+ Add URI' button; 'Authorised redirect URIs' with a note 'For use with requests from a web server' and a text input field containing 'http://localhost:8080/login/oauth2/code/google' and an '+ Add URI' button; and a 'Note: It may take five minutes to a few hours for settings to take effect' with 'Create' and 'Cancel' buttons. A dark notification box at the bottom center says 'OAuth configuration created.'

Aditya Anilkumar Lakhiwal

MSc CS F004



Aditya Anilkumar Lakhiwal
MSc CS F004

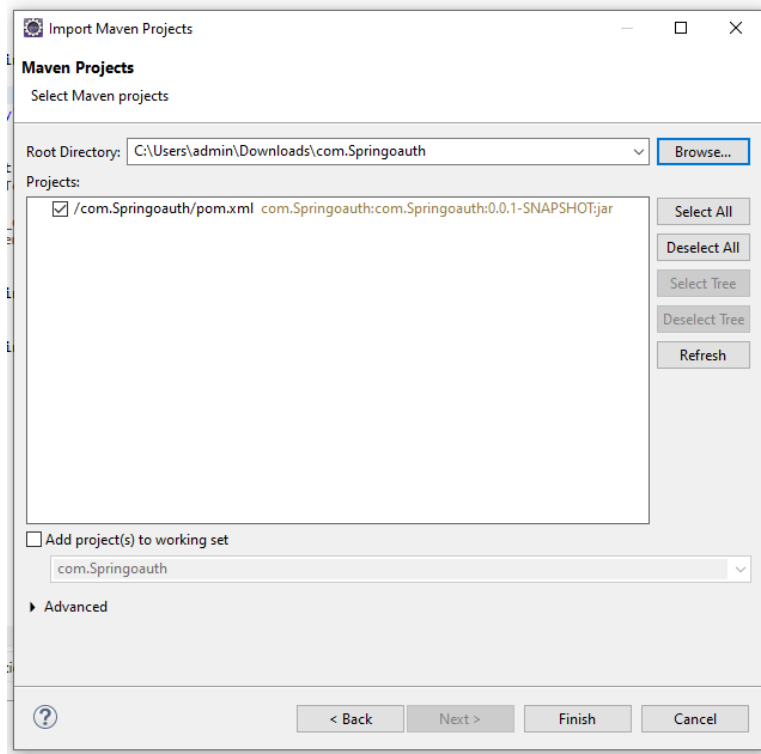
Create Maven Project:

The screenshot shows the Spring Initializr web application interface. The browser tabs at the top include 'Video Conferencing, Meetings', 'Teams and Channels | General', 'Inbox (8,129) - adityalakhiwal@...', 'Clients - Google Auth Platform', 'Spring auth steps.docx', and 'Spring Initializr'. The address bar shows 'start.spring.io'. The application has a dark theme and a sidebar menu on the left. The main content area is divided into sections for project configuration:

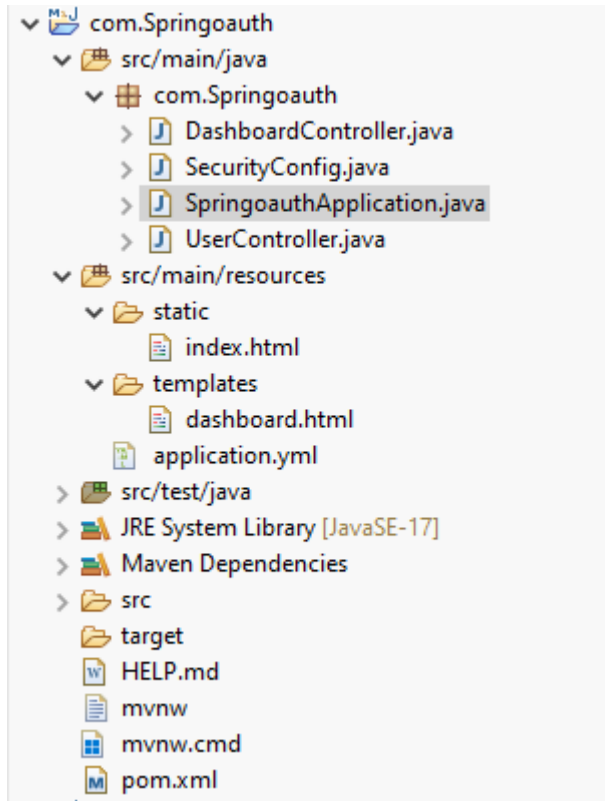
- Project:** Radio buttons for **Gradle - Groovy**, **Gradle - Kotlin**, and **Maven** (selected).
- Language:** Radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot:** Radio buttons for **4.0.0 (SNAPSHOT)**, **4.0.0 (M1)**, **3.5.5 (SNAPSHOT)**, and **3.5.4** (selected).
- Spring Boot:** Radio buttons for **3.4.9 (SNAPSHOT)** and **3.4.8**.
- Project Metadata:** Fields for **Group** (com.Springoauth), **Artifact** (com.Springoauth), **Name** (com.Springoauth), **Description** (Oauth), and **Package name** (com.Springoauth).
- Packaging:** Radio buttons for **Jar** (selected) and **War**.
- Java:** Radio buttons for **24**, **21**, and **17** (selected).
- Dependencies:** A section with a button **ADD DEPENDENCIES... CTRL + B** and a list of dependencies:
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Thymeleaf** (TEMPLATE ENGINE): A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.
 - Spring Boot Dev Tools** (DEVELOPER TOOLS): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
 - Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.
 - OAuth2 Client** (SECURITY): Spring Boot integration for Spring Security's OAuth2/OpenID Connect client features.
 - Spring Security** (SECURITY): Highly customizable authentication and access-control framework for Spring applications.

At the bottom, there are three buttons: **GENERATE CTRL + G**, **EXPLORE CTRL + SPACE**, and **...**.

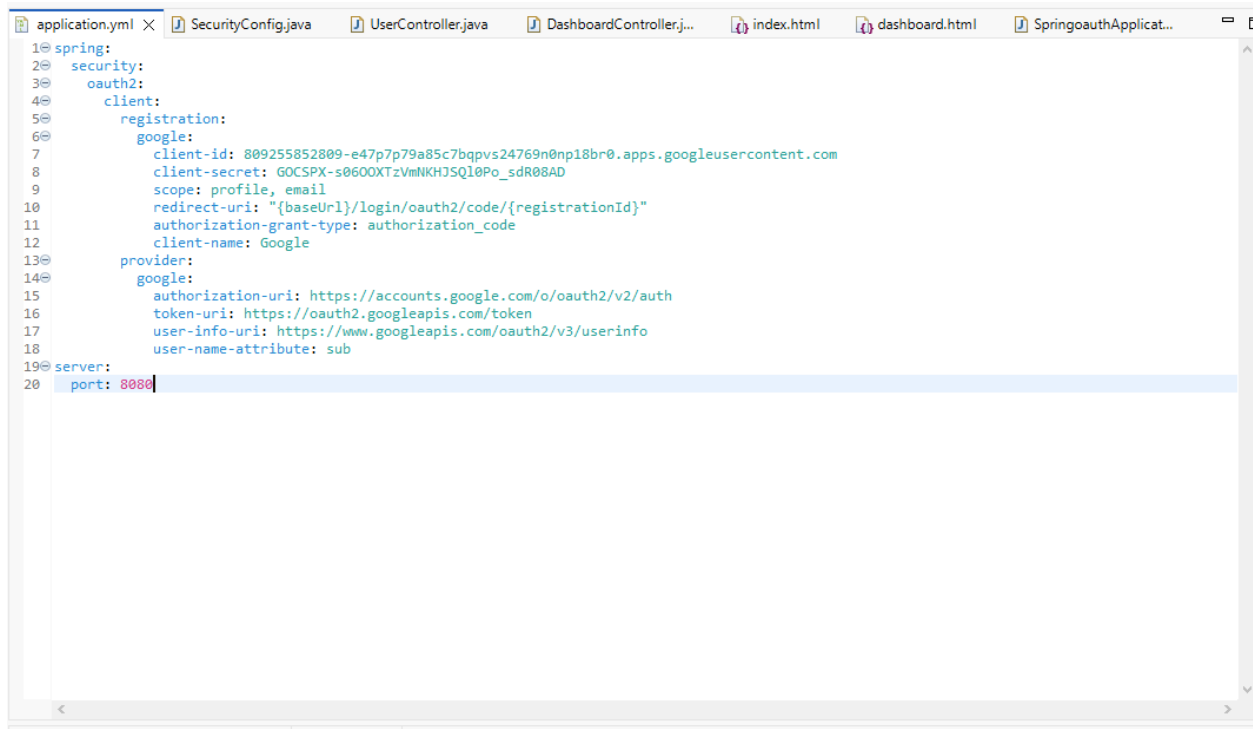
Import Maven Project:



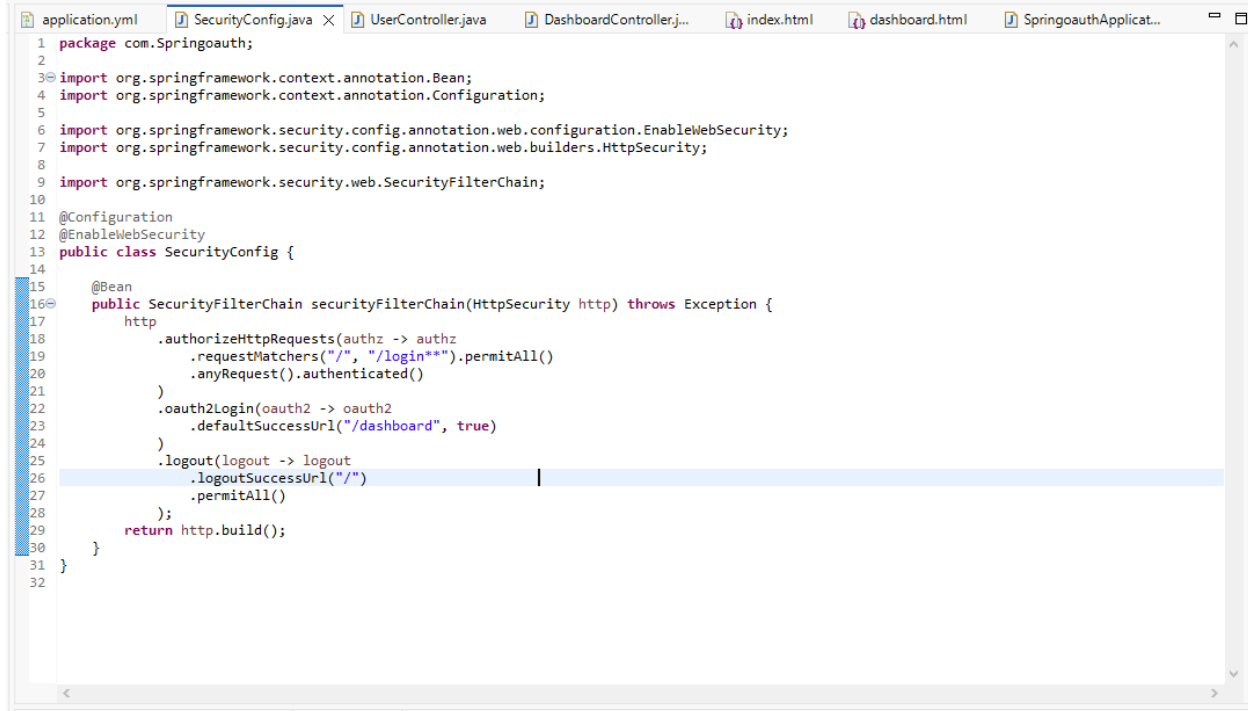
Structure:



application.yml

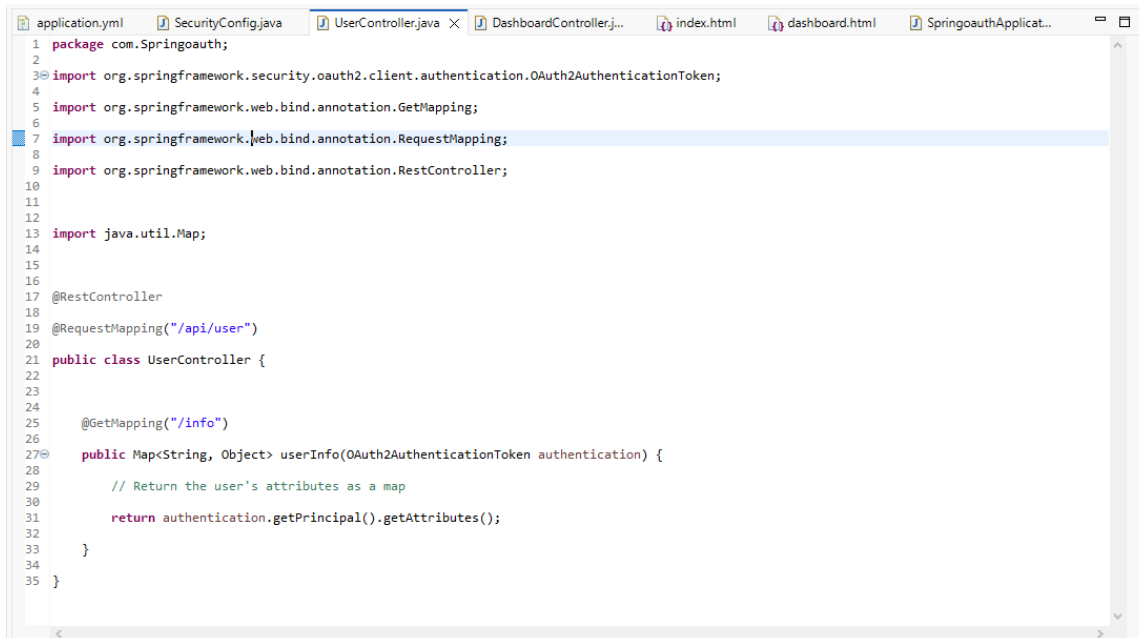


SecurityConfig.java:

A screenshot of an IDE window showing the SecurityConfig.java file. The file is part of the com.Sprigoauth package. It imports several Spring Framework classes: Bean, Configuration, EnableWebSecurity, HttpSecurity, and SecurityFilterChain. The SecurityConfig class is annotated with @Configuration and @EnableWebSecurity. It contains a @Bean method named securityFilterChain that takes an HttpSecurity object and returns a SecurityFilterChain. The method configures the HttpSecurity object with authorizeHttpRequests, oauth2Login, and logout methods. The authorizeHttpRequests method is configured to permit all requests to the root and login endpoints. The oauth2Login method is configured with a default success URL of /dashboard. The logout method is configured to permit all requests to the root endpoint.

```
1 package com.Sprigoauth;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5
6 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
7 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
8
9 import org.springframework.security.web.SecurityFilterChain;
10
11 @Configuration
12 @EnableWebSecurity
13 public class SecurityConfig {
14
15     @Bean
16     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
17         http
18             .authorizeHttpRequests(authz -> authz
19                 .requestMatchers("/", "/login**").permitAll()
20                 .anyRequest().authenticated()
21             )
22             .oauth2Login(oauth2 -> oauth2
23                 .defaultSuccessUrl("/dashboard", true)
24             )
25             .logout(logout -> logout
26                 .logoutSuccessUrl("/")
27                 .permitAll()
28             );
29         return http.build();
30     }
31 }
32
```

UserController.java

A screenshot of an IDE window showing the UserController.java file. The file is part of the com.Sprigoauth package. It imports several Spring Framework classes: OAuth2AuthenticationToken, GetMapping, RequestMapping, and RestController. It also imports java.util.Map. The UserController class is annotated with @RestController and @RequestMapping("/api/user"). It contains a @GetMapping method named userInfo that takes an OAuth2AuthenticationToken object and returns a Map of user attributes. The method is annotated with a comment indicating it returns the user's attributes as a map.

```
1 package com.Sprigoauth;
2
3 import org.springframework.security.oauth2.client.authentication.OAuth2AuthenticationToken;
4
5 import org.springframework.web.bind.annotation.GetMapping;
6
7 import org.springframework.web.bind.annotation.RequestMapping;
8
9 import org.springframework.web.bind.annotation.RestController;
10
11 import java.util.Map;
12
13 @RestController
14 @RequestMapping("/api/user")
15 public class UserController {
16
17     @GetMapping("/info")
18     public Map<String, Object> userInfo(OAuth2AuthenticationToken authentication) {
19         // Return the user's attributes as a map
20         return authentication.getPrincipal().getAttributes();
21     }
22 }
23
24
25
26
27
28
29
30
31
32
33
34
35

```

DashboardController.java

Aditya Anilkumar Lakhiwal

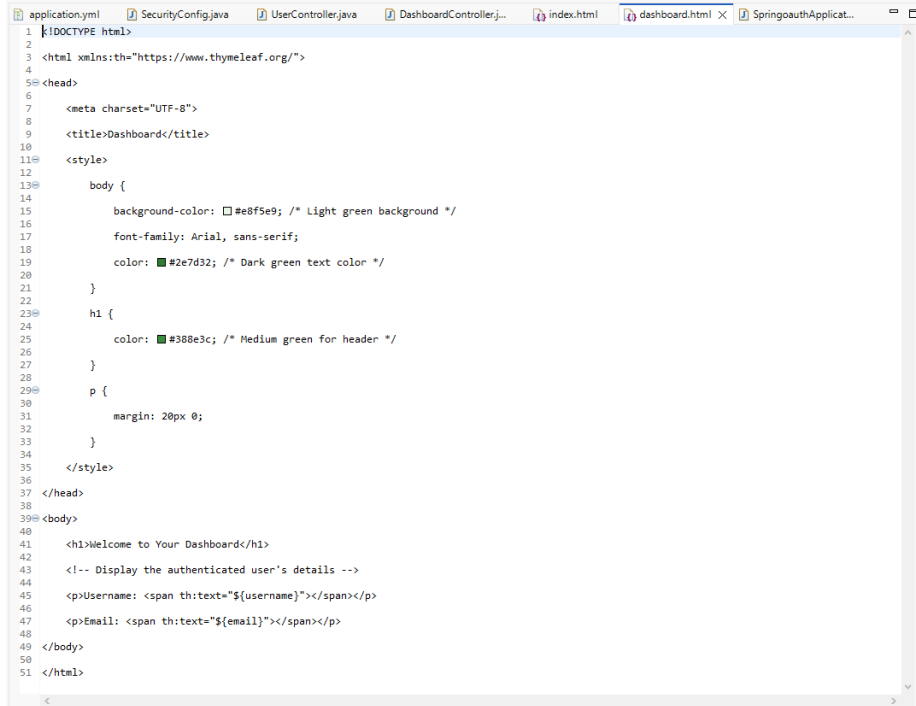
MSc CS F004

```
application.yml  SecurityConfig.java  UserController.java  *DashboardController...  index.html  dashboard.html  SpringoauthApplicati...
1 package com.Springoauth;
2
3 import org.springframework.security.core.annotation.AuthenticationPrincipal;
4
5 import org.springframework.security.oauth2.core.user.OAuth2User;
6
7 import org.springframework.stereotype.Controller;
8
9 import org.springframework.ui.Model;
10
11 import org.springframework.web.bind.annotation.GetMapping;
12
13
14 @Controller
15
16 public class DashboardController {
17
18
19
20
21     @GetMapping("/dashboard")
22
23     public String dashboard(@AuthenticationPrincipal OAuth2User principal, Model model) {
24
25         // Extract user details from OAuth2User
26
27         String username = principal.getAttribute("name");
28
29         String email = principal.getAttribute("email");
30
31
32         // Add user details to the model
33
34         model.addAttribute("username", username);
35
36         model.addAttribute("email", email);
37
38
39
40
41         return "dashboard";
42     }
43 }
44
45 }
```

index.html inside static folder:

```
application.yml  SecurityConfig.java  UserController.java  DashboardController.j...  index.html  dashboard.html  SpringoauthApplicat...
1 <!DOCTYPE html>
2
3 <html xmlns:th="https://www.thymeleaf.org/">
4
5 <head>
6
7     <meta charset="UTF-8">
8
9     <title>Home</title>
10
11 <style>
12
13     body {
14
15         background-color: #e8f5e9; /* Light green background */
16
17         font-family: Arial, sans-serif;
18
19         color: #2e7d32; /* Dark green text color */
20
21     }
22
23     h1 {
24
25         color: #388e3c; /* Medium green for header */
26
27     }
28
29     a {
30
31         color: #1b5e20; /* Dark green for links */
32
33         text-decoration: none;
34
35         padding: 10px;
36
37         border: 2px solid #1b5e20;
38
39         border-radius: 5px;
40
41         display: inline-block;
42
43         margin-top: 20px;
44
45     }
46
47     a:hover {
48
49         background-color: #a5d6a7; /* Lighter green on hover */
50
51     }
52 }
```

dashboard.html inside templates folder:

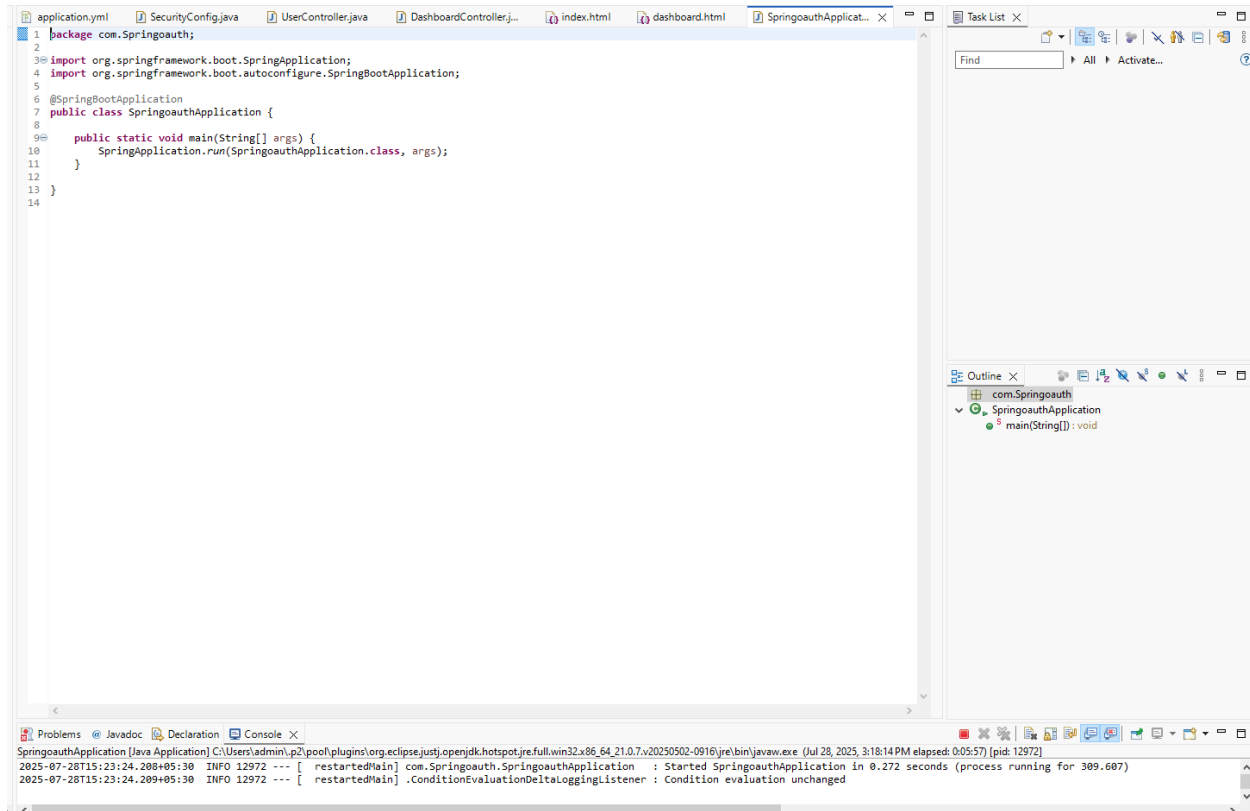


```
1 <!DOCTYPE html>
2
3 <html xmlns:th="https://www.thymeleaf.org/">
4
5 <head>
6
7   <meta charset="UTF-8">
8
9   <title>Dashboard</title>
10
11 <style>
12
13   body {
14
15     background-color: #e8f5e9; /* Light green background */
16
17     font-family: Arial, sans-serif;
18
19     color: #2e7d32; /* Dark green text color */
20
21   }
22
23   h1 {
24
25     color: #388e3c; /* Medium green for header */
26
27   }
28
29   p {
30
31     margin: 20px 0;
32
33   }
34
35 </style>
36
37 </head>
38
39 <body>
40
41   <h1>Welcome to Your Dashboard</h1>
42
43   <!-- Display the authenticated user's details -->
44
45   <p>Username: <span th:text="${username}"></span></p>
46
47   <p>Email: <span th:text="${email}"></span></p>
48
49 </body>
50
51 </html>
```

SpringoauthApplication.java

Aditya Anilkumar Lakhiwal

MSc CS F004



```
1 package com.SpringoAuth;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class SpringoAuthApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(SpringoAuthApplication.class, args);
11     }
12 }
13
14
```

com.SpringoAuth
SpringoAuthApplication
main(String[]) : void

SpringoAuthApplication [Java Application] C:\Users\admin\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.21.0.7.v20250502-0916\jre\bin\javaw.exe (Jul 28, 2025, 3:18:14 PM elapsed: 0:05:57) [pid: 12972]
2025-07-28T15:23:24.288+05:30 INFO 12972 --- [restartedMain] com.SpringoAuth.SpringoAuthApplication : Started SpringoAuthApplication in 0.272 seconds (process running for 309.607)
2025-07-28T15:23:24.289+05:30 INFO 12972 --- [restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged

Output:

