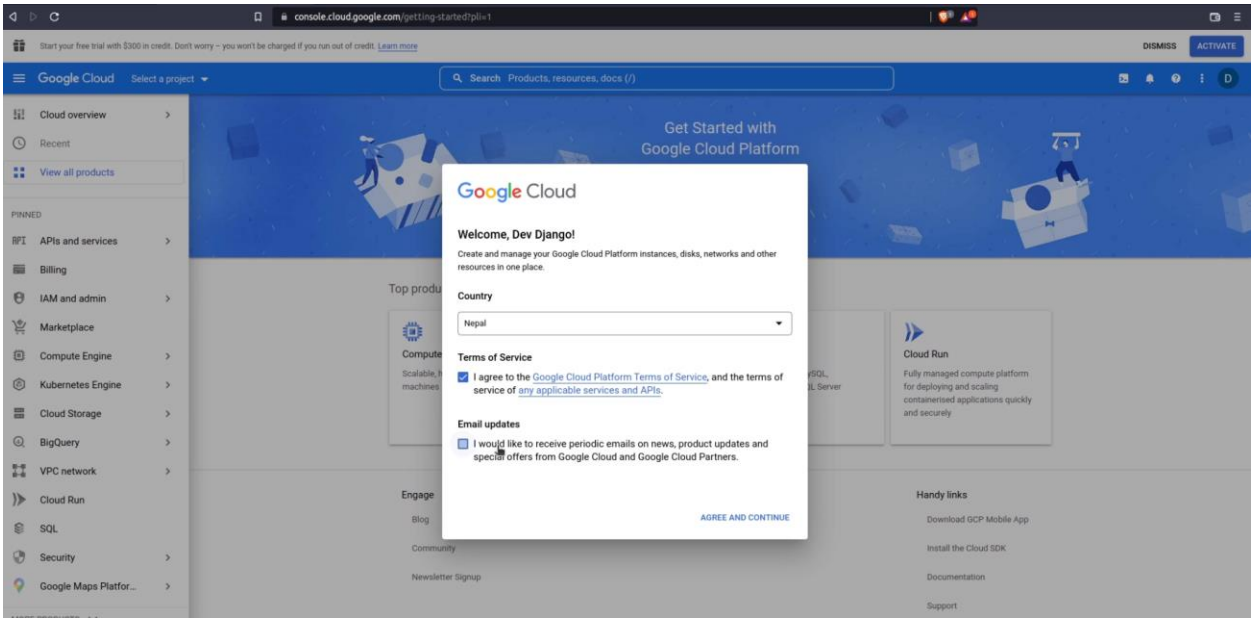
Integration of Google Workspace Admin Report in Java Spring Boot Project

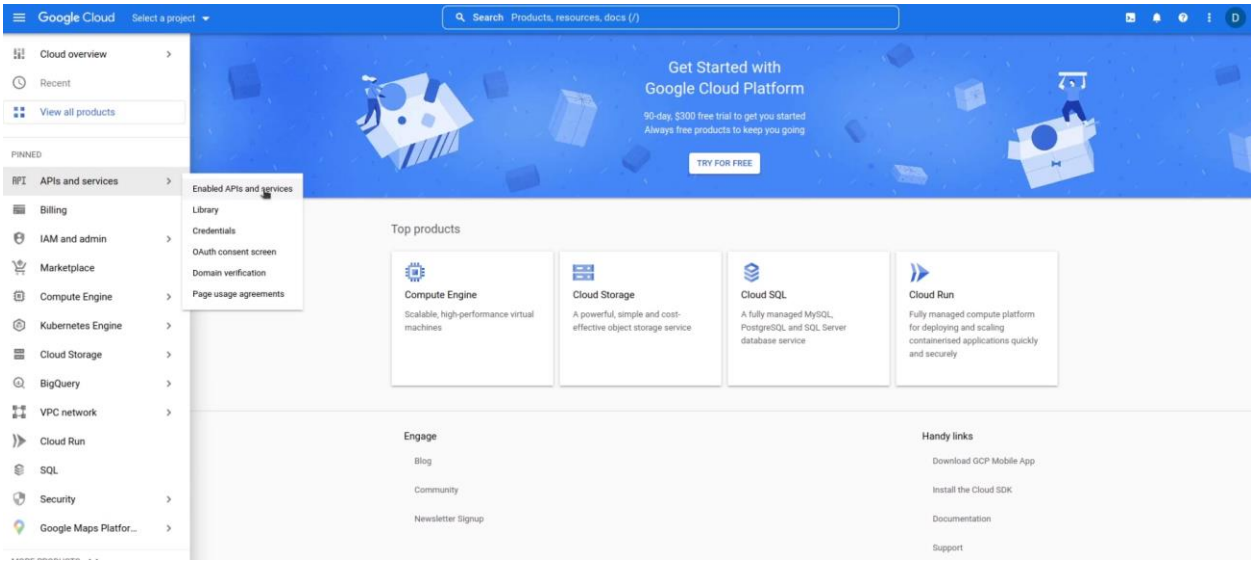
The report demonstrates how to call Google Workspace APIs in Java.

Step 1: Go to Google Console dashboard.
http://console.cloud.google.com/

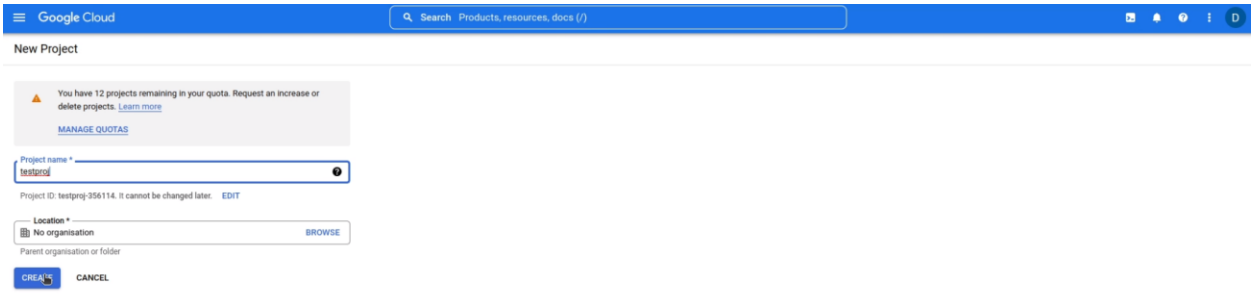


And agree to the terms.

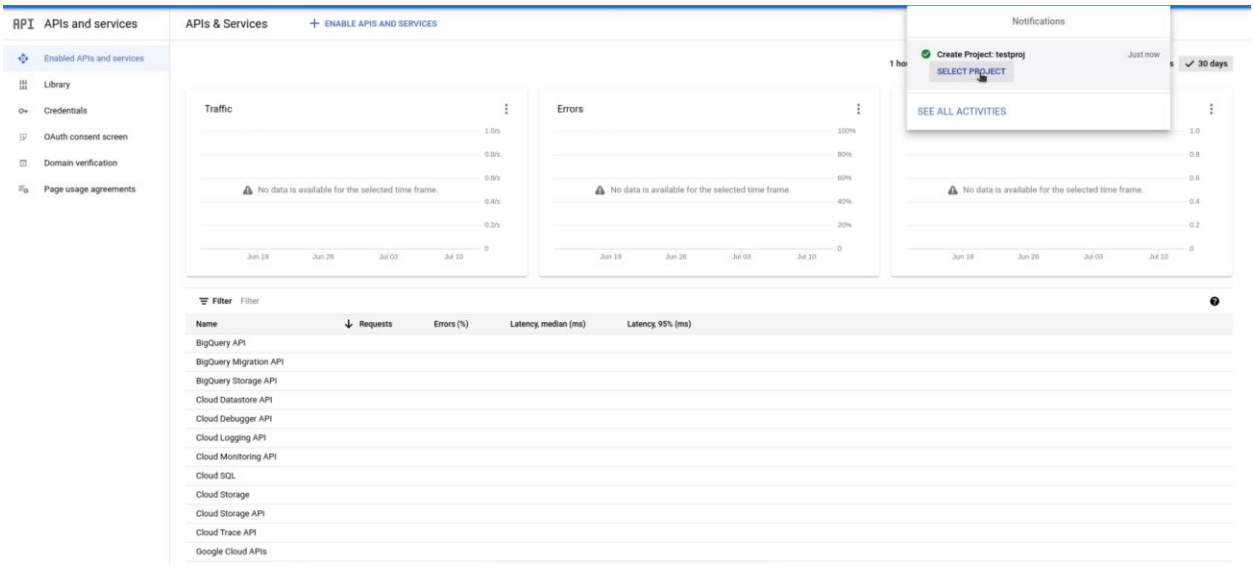
Step 2: On the left panel. Visit **APIs and Services** tab, and then to **Enabled APIs and services**.



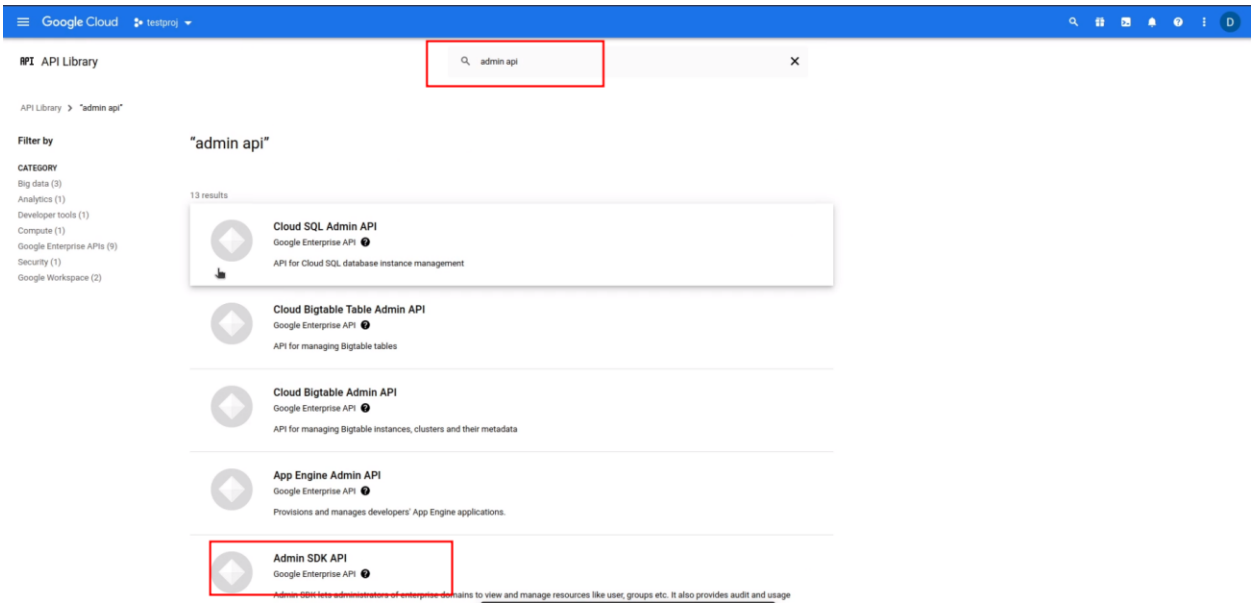
Step 3: On the next screen, **create a project** and provide a name and organization (if any).



Step 4: On the top right side of the screen, you should see a notification card to select the project, if you have one project, it's selected by default.



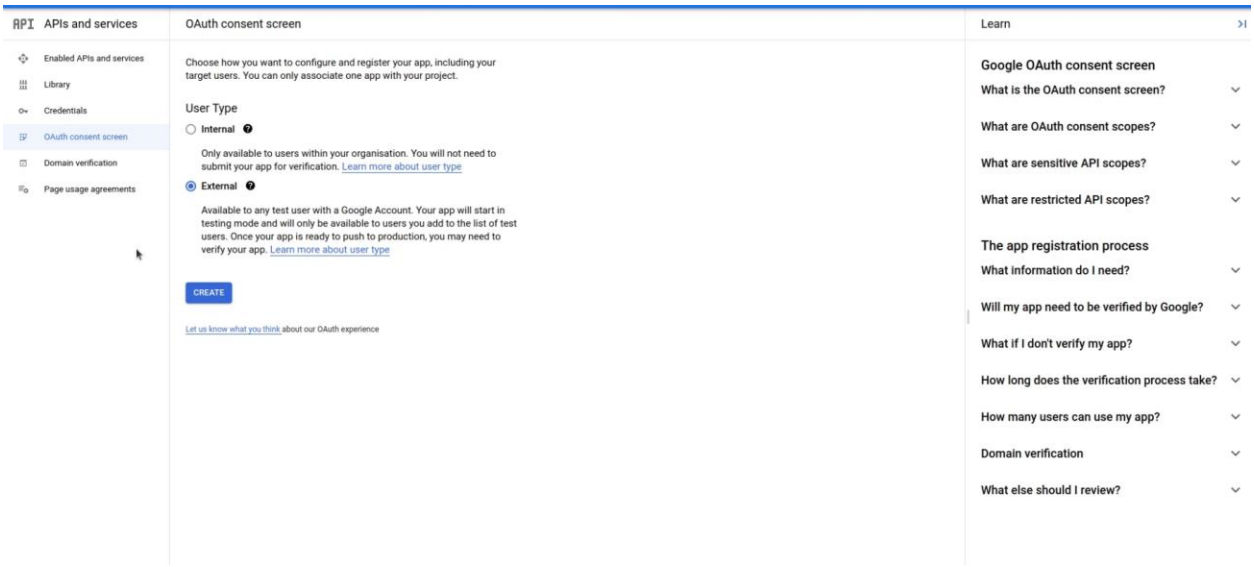
Step 5: Again, click on the Library tab and search for “AdminApi”, you should see a service named **Admin SDK Api** and enable it.



Step 6: Once it is enabled, go to the OAuth **consent screen.** Here, we have two options for User Type, the Internal and the External. For the demo purpose, we’ll set it as External User Type.

Internal User Type: Only available to users within your organization.

External User Type: Available to any test user with a Google Account.



Step 7: On the next screen, we'll have four steps that follow

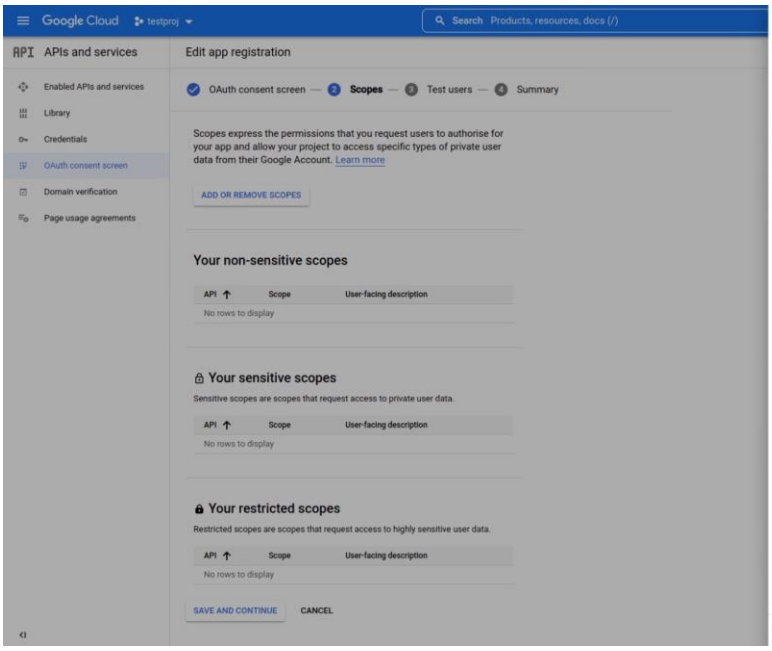
- OAuth consent screen
- Scope
- Test users
- Summary.

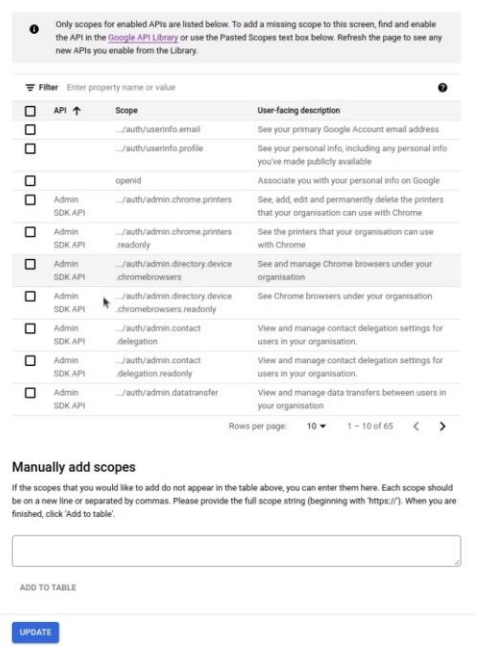
On the OAuth screen

Fill the required app information, application domain, and Developer contact information. After all, save and continue the fields.

Scopes

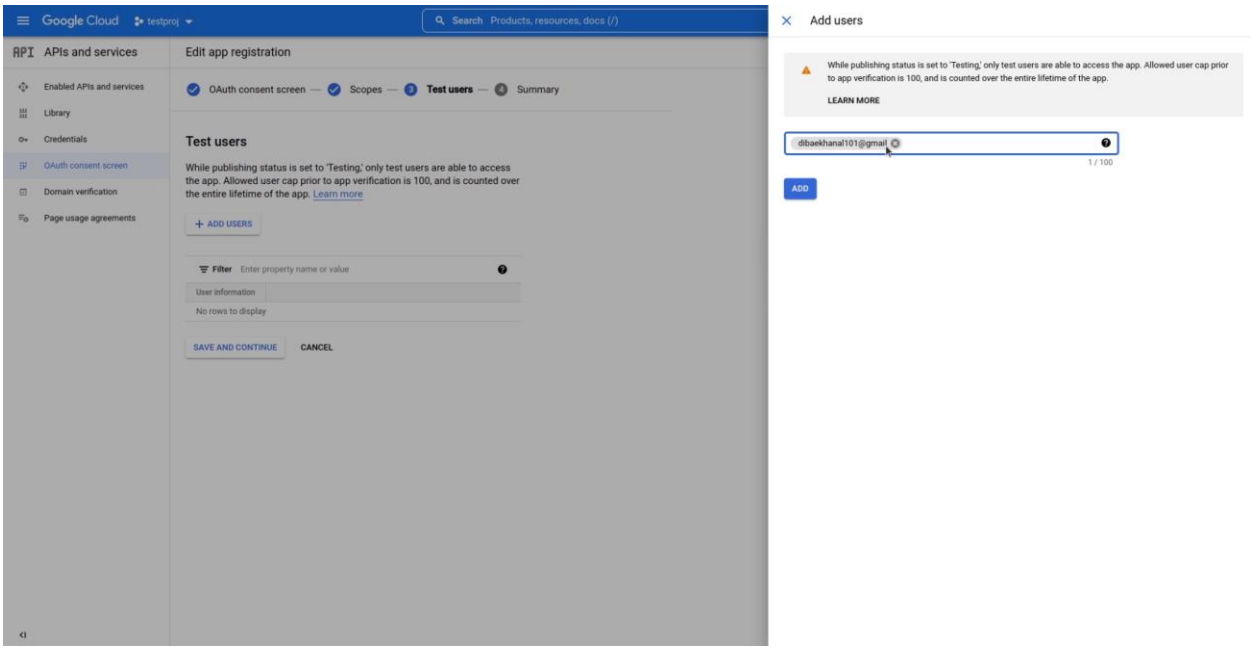
Here, we can set permission for users to authorize for the application. Click on **Add or Remove Scope** to control the permission or add scopes manually. Now, save and continue.





Test Users

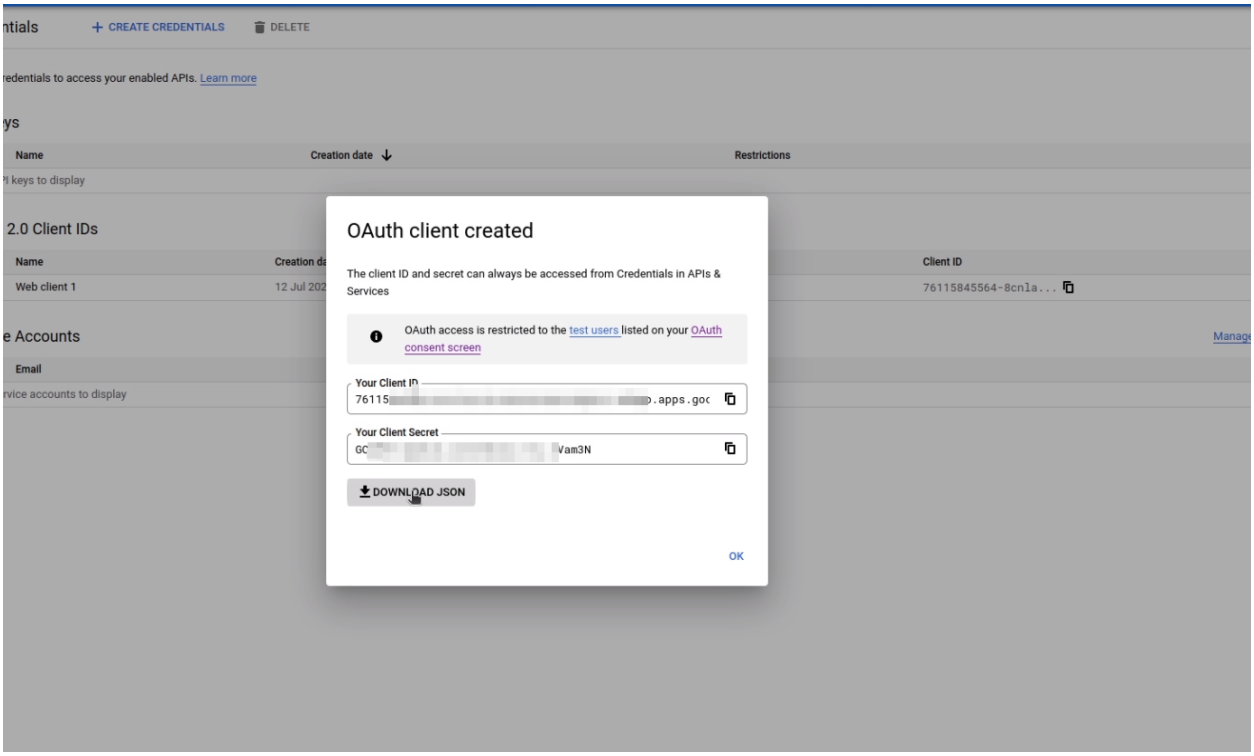
We can as well set a test user to access the app. On the right panel, add the email address and Add.



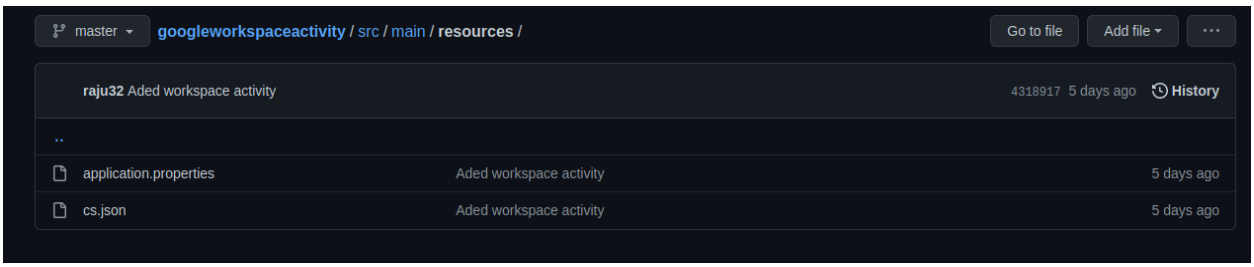
Summary
Summarize the registration and save the changes with the Back **to Dashboard** button.

***Step 8*:** On the left panel, click on **Credentials,** and **create credentials**. On the next screen, select the application type, here we are interested in Web applications. Also, provide an application name. On the **Authorized redirect URIs**, add your domain name with the API url. Suppose, my production application's domain name is https://example.com, then the request from the web server should be https://example.com/Callback. Henceforth, we can add multiple URIs as well. Finally, create the OAuth ID settings.

This will generate the Client ID and the secret Keys.



WIth this credential, we can configure the application. Download the json credential generated from the OAuth client and add the file to the following path**: /src/main/resources/**




```
@SpringBootApplication
public class GoogleActivityReportApplication {
    /** Application name. */
    private static final String APPLICATION_NAME = "Google Admin SDK
Reports API Java Quickstart";
    /** Global instance of the JSON factory. */
    private static final JsonFactory JSON_FACTORY =
GsonFactory.getDefaultInstance();
    /** Directory to store authorization tokens for this application. */
    private static final String TOKENS_DIRECTORY_PATH = "tokens";

    /**
     * Global instance of the scopes required by this quickstart.
```

```
     * If modifying these scopes, delete your previously saved tokens/
folder.
     */
    private static final List<String> SCOPES =
Collections.singletonList(ReportsScopes.ADMIN_REPORTS_AUDIT_READONLY);
    private static final String CREDENTIALS_FILE_PATH = "/cs.json";
```

The **TOKENS_DIRECTORY_PATH** is the directory to store authorization tokens for this application and saves the token generated for next retrieval.

The following string **CREDENTIALS_FILE_PATH** saves the generated tokens. If you are modifying the Scope, delete your previously saved tokens/ folder.

```
private static Credential getCredentials(final NetHttpTransport
HTTP_TRANSPORT) throws IOException {
       // Load client secrets.
       InputStream in =
GoogleActivityReportApplication.class.getResourceAsStream(CREDENTIALS_FILE_
PATH);
       if (in == null) {
             throw new FileNotFoundException("Resource not found: " +
CREDENTIALS_FILE_PATH);
       }
       GoogleClientSecrets clientSecrets =
GoogleClientSecrets.load(JSON_FACTORY, new InputStreamReader(in));

       // Build flow and trigger user authorization request.
       GoogleAuthorizationCodeFlow flow = new
GoogleAuthorizationCodeFlow.Builder(
                   HTTP_TRANSPORT, JSON_FACTORY, clientSecrets, SCOPES)
                   .setDataStoreFactory(new FileDataStoreFactory(new
java.io.File(TOKENS_DIRECTORY_PATH)))
                   .setAccessType("offline")
                   .build();
       LocalServerReceiver receiver = new
LocalServerReceiver.Builder().setPort(8888).build();
       return new AuthorizationCodeInstalledApp(flow,
receiver).authorize("user");
    }
```

The following function sends the HTTP request to the Google Authorization server along with a credential file and authenticates the request.

```
public static void main(String... args) throws IOException,
GeneralSecurityException {
       final NetHttpTransport HTTP_TRANSPORT =
GoogleNetHttpTransport.newTrustedTransport();
       Reports service = new Reports.Builder(HTTP_TRANSPORT, JSON_FACTORY,
getCredentials(HTTP_TRANSPORT))
                     .setApplicationName(APPLICATION_NAME)
                     .build();
```

The **Reports.Builder** is instantiated along with HTTP_TRANSPORT, JSON_FACTORY and getCredentials obtains data from the server.

```
String userKey = "all";
String applicationName = "admin";
```

Now, the report is generated based on the userKey, and applicationName.

Upon running the application, we can get the logs of users' login activity within the registered domain.

Login logs:

```
Attempting to open that address in the default browser now...
Logins:
2022-07-06T14:28:24.428Z: sushil@securityj-class.com (login_success)
2022-07-05T05:06:21.755Z: sushil@securityj-class.com (login_success)
2022-07-04T15:33:44.956Z: sushil@securityj-class.com (login_success)
2022-07-04T14:39:55.415Z: sushil@securityj-class.com (login_success)
2022-07-04T14:34:57.395Z: sushil@securityj-class.com (login_success)
2022-07-04T14:34:56.583Z: sushil@securityj-class.com (password_edit)
2022-07-04T14:34:24.407Z: sushil@securityj-class.com (login_success)
2022-07-04T14:33:10.227Z: sushil@securityj-class.com (login_success)
2022-07-04T08:00:05.406Z: sushil@securityj-class.com (login_success)
2022-07-04T06:05:36.990Z: sushil@securityj-class.com (login_success)

Process finished with exit code 0
```

Admin logs:

```
Admin:
2022-07-06T14:28:27.875Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)
2022-07-04T16:12:14.611Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)
2022-07-04T16:12:04.564Z: sushil@securityj-class.com (GENERATE_CERTIFICATE)
2022-07-04T16:10:40.739Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)
2022-07-04T16:08:29.131Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)
2022-07-04T15:52:27.201Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)
2022-07-04T15:33:49.519Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)
2022-07-04T14:40:00.253Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)
2022-07-04T14:35:42.471Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)
2022-07-04T08:09:06.387Z: sushil@securityj-class.com (ALERT_CENTER_VIEW)

Process finished with exit code 0
```

Application name for which the events are to be retrieved.

Enums	
access_transparency	The Google Workspace Access Transparency activity reports return information about different types of Access Transparency activity events.
admin	The Admin console application's activity reports return account information about different types of administrator activity events.
calendar	The Google Calendar application's activity reports return information about various Calendar activity events.
chat	The Chat activity reports return information about various Chat activity events.
drive	The Google Drive application's activity reports return information about various Google Drive activity events. The Drive activity report is only available for Google Workspace Business and Google Workspace Enterprise customers.
gcp	The Google Cloud Platform application's activity reports return information about various GCP activity events.
gplus	The Google+ application's activity reports return information about various Google+ activity events.
groups	The Google Groups application's activity reports return information about various Groups activity events.
groups_enterprise	The Enterprise Groups activity reports return information about various Enterprise group activity events.

jamboard	The Jamboard activity reports return information about various Jamboard activity events.
login	The Login application's activity reports return account information about different types of Login activity events.
meet	The Meet Audit activity report returns information about different types of Meet Audit activity events.
mobile	The Device Audit activity report returns information about different types of Device Audit activity events.
rules	The Rules activity report returns information about different types of Rules activity events.
saml	The SAML activity report returns information about different types of SAML activity events.
token	The Token application's activity reports return account information about different types of Token activity events.
user_accounts	The User Accounts application's activity reports return account information about different types of User Accounts activity events.
context_aware_access	The Context-aware access activity reports return information about users' access denied events due to Context-aware access rules.
chrome	The Chrome activity reports return information about Chrome browser and Chrome OS events.

data_studio	The Data Studio activity reports return information about various types of Data Studio activity events.
keep	The Keep application's activity reports return information about various Google Keep activity events.

Common Issues and their solutions while setting this app up

Issue#1:
If you are facing the type of error given below then it means after running app the account you are choosing to authenticate with is not under jclass.solutions project with admin rights.
For Example: I am authenticating with irfan@gmail.com which is not official account by jclass.solutions project.

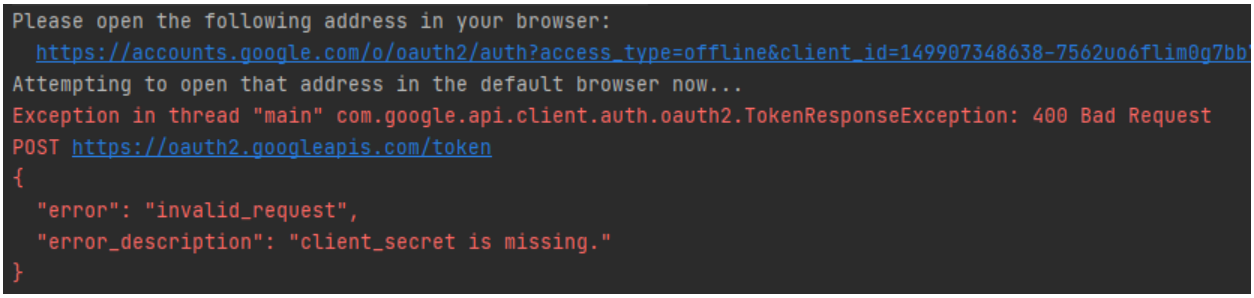
```
Attempting to open that address in the default browser now...
com.google.api.client.googleapis.json.GoogleJsonResponseException: 401 Unauthorized
GET https://admin.googleapis.com/admin/reports/v1/activity/users/all/applications/chat?maxResults=10
{
  "code": 401,
  "errors": [
    {
      "domain": "global",
      "location": "Authorization",
      "locationType": "header",
      "message": "Access denied. You are not authorized to read activity records.",
      "reason": "authError"
    }
  ],
  "message": "Access denied. You are not authorized to read activity records."
}
```

Solution:
If you want to run the GoogleActivityReportApplication, Admin must have to create user@jclass.solutions account and grant admin rights to it for the specific user.

NOTE: Once you resolved this issue you must have to delete file from token folder and then re-run the application.

Issue#2:

If you are facing the below issue it means that Client_secret is missing from the json file you have downloaded at Step 8.



Solution:

Just verify from the json file if it is missing then you have to re-download the same file. If it still missing then you have to drop the previous **OAuth 2.0 Client ID** and add a new one and make sure your **Authorized redrirect URIs** is http://localhost:8888/Callback and download the newly created json and configure in the GoogleActivityRepportApplication.

EXTRAS:

To Try different google account activities you can go to the below url and try different parameters in application name.

https://developers.google.com/admin-sdk/reports/reference/rest/v1/activities/list