Madhawa Sumanaweera – MS20918606

Sri Lanka Institute of Information Technology. MSc in Information Technology (Specialization in Cyber Security)

mpriyanjith@gmail.com / madhawapriyanjith@gmail.com

OAuth 2.0 framework.

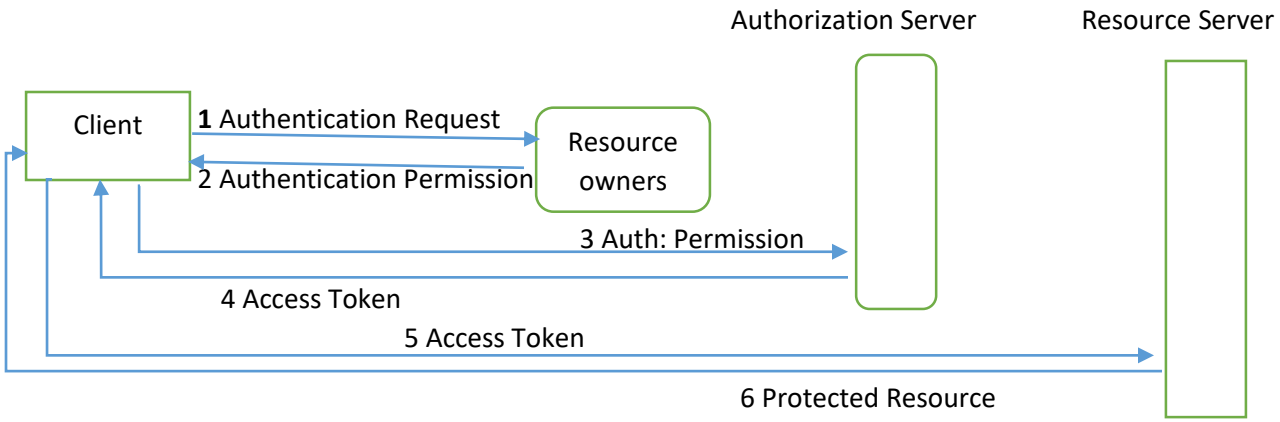
Introduction.

OAuth 2 is an industry standard application use in web, desktop and mobile base authentication system. This will use to get the user account access on HTTP services, For example GitHub, Google drive, Face Book. This will enable the user access to defined servers and application through the user accounts [1].

In mobile apps OAuth is use for Native and Browser Base Apps and device authentication process. From the token management process will use for determine the activity state and meta information of token revocation process and JOSN web token process [2].

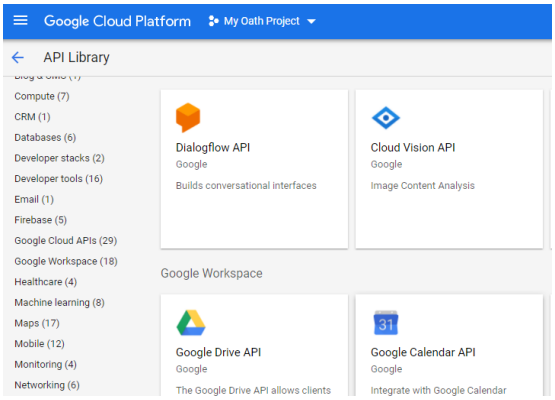
In this assignment I used Java for develop the OAuth application and will explain, how the basic background arrangement in Google Drive API to get the access to Google Drive by authenticating through Java application and use of the OAuth 2 services use for get access to Google drive through. This application is using Spring Boot and the Google Client Library for OAuth and Drive

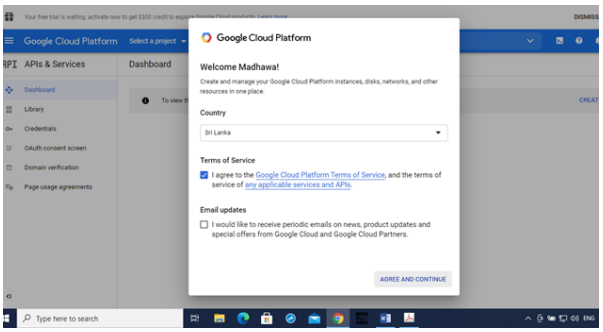
Below diagram explain the high level process OAuth application [3].

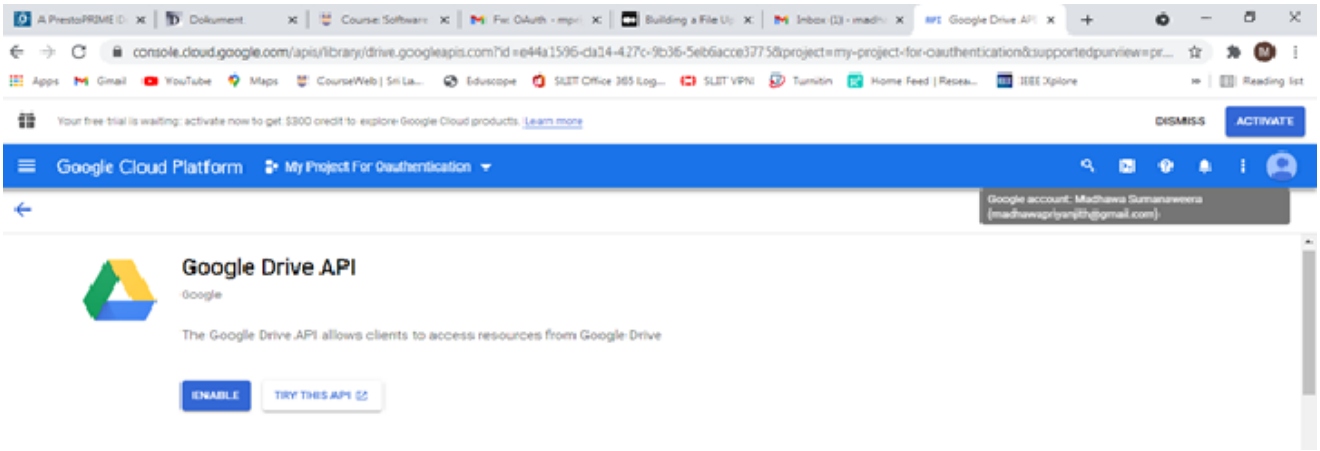


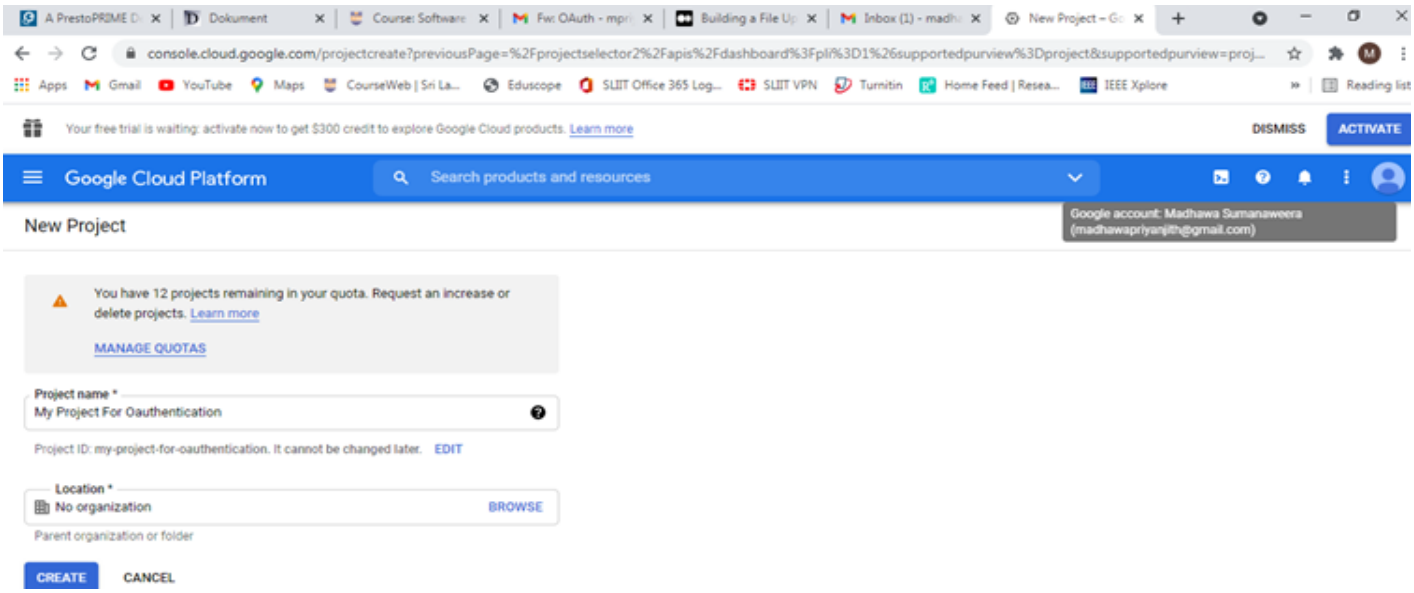
Key Arrangements.

In the initial process, need to setup a project in Google API console to get the obtain OAuth 2.0 credentials and below are the steps of creating this in Google API.

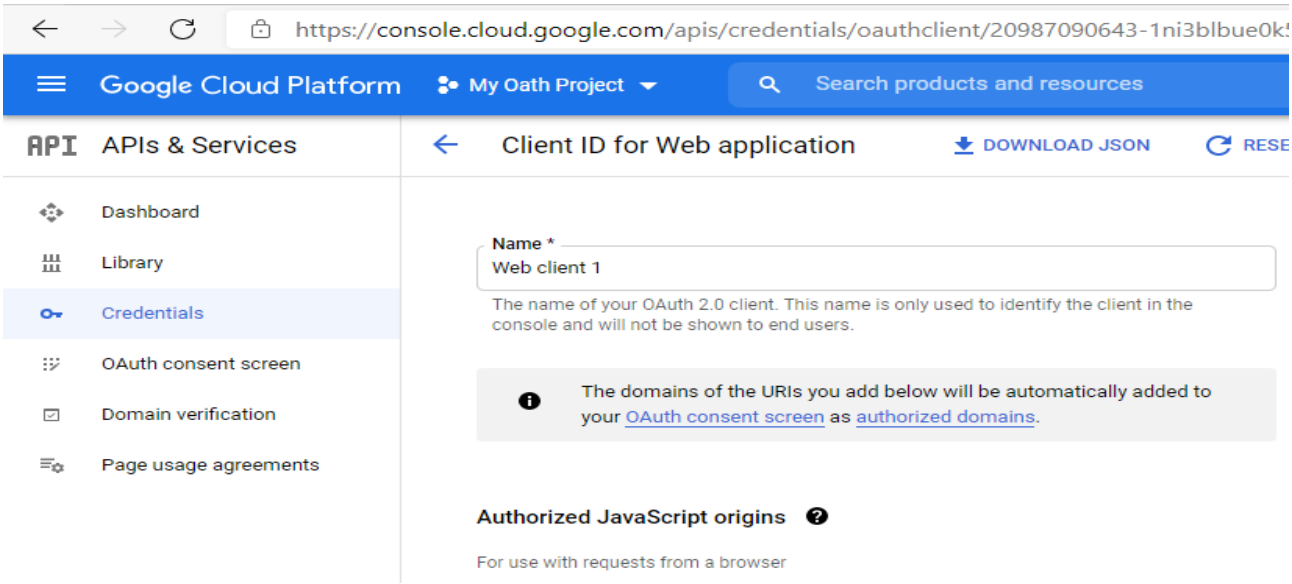


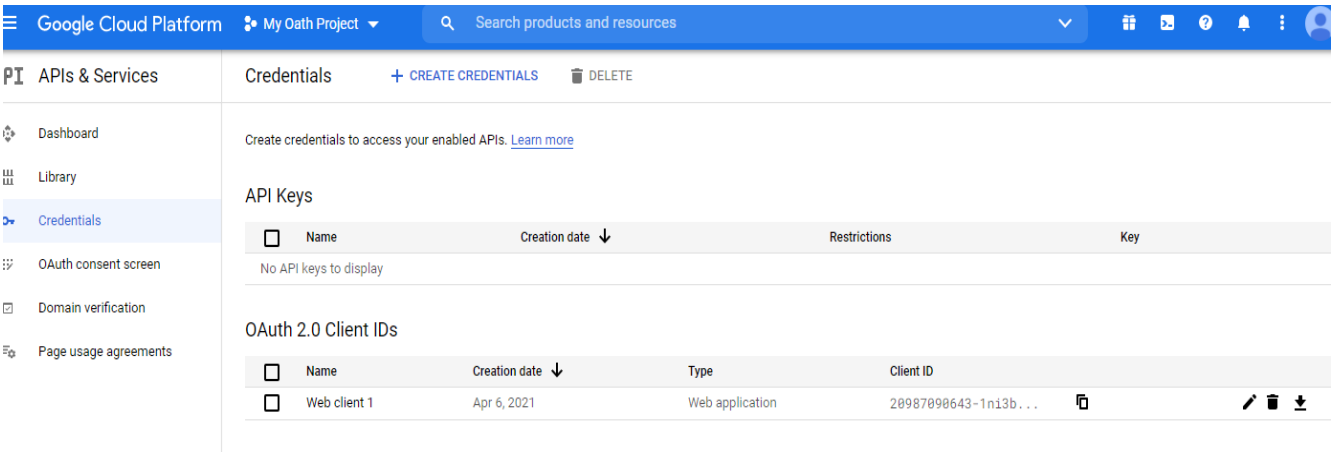






Below is the credential Obtained for my OAuth project and the client ID and other details related to this OAuth project.

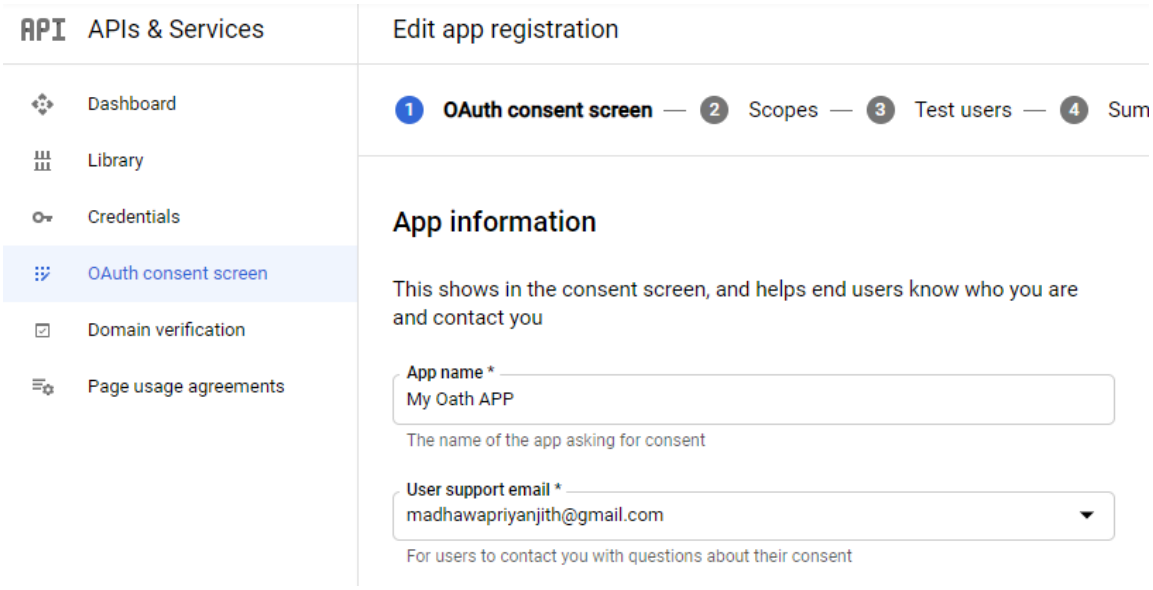


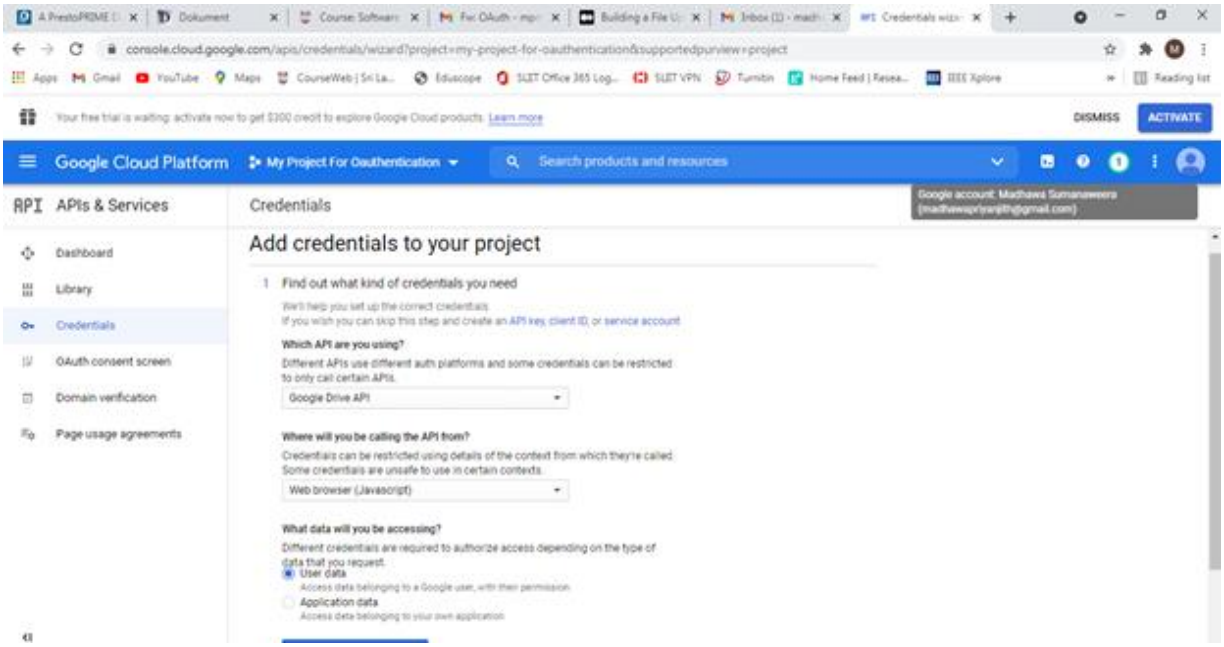


Client ID	20987090643-1ni3blbue0k5m6922ei04r2tqutlh4on.apps.googleusercontent.com
Client secret	wVucRVI-tB9QGuaNjGmHUAkj
Creation date	April 6, 2021 at 5:19:10 PM GMT+5

OAuth Consent Screen.

In this page the details of the author of the application will added for end user references.





JASON Token.

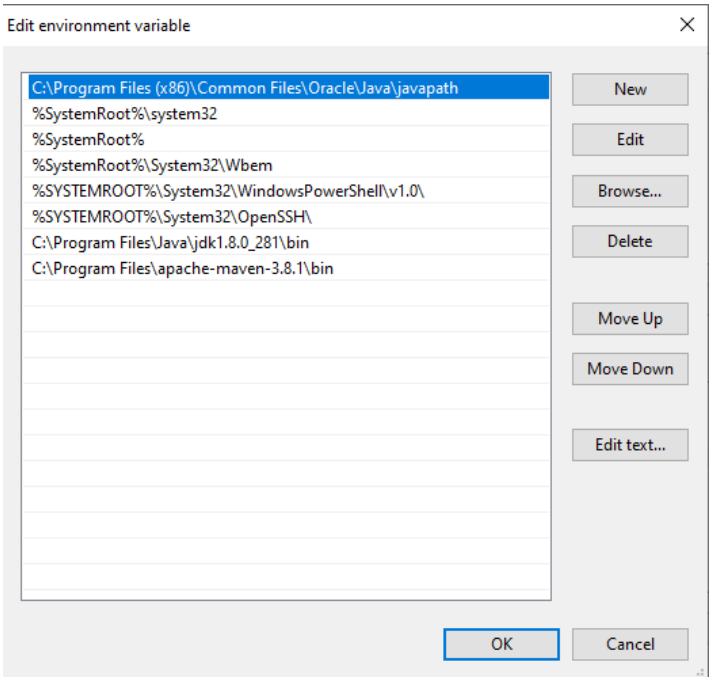
Below is the Jason token generated for this application.

{"web":{"client_id":"20987090643-1ni3blbue0k5m6922ei04r2tqutlh4on.apps.googleusercontent.com","project_id":"my-oath-project-308700","auth_uri":"https://accounts.google.com/o/oauth2/auth","token_uri":"https://oauth2.googleapis.com/token","auth_provider_x509_cert_url":"https://www.googleapis.com/oauth2/v1/certs","client_secret":"wVucRVI-tB9QGuaNjGmHUAkj"}}

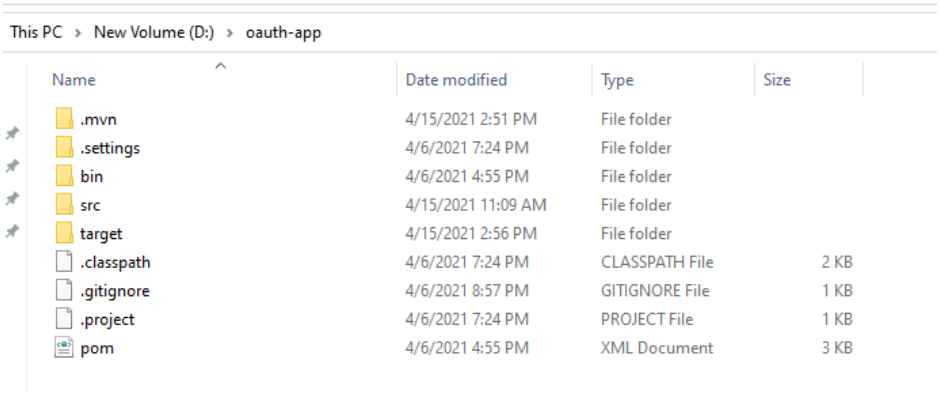
Once the details are filled save before exit.

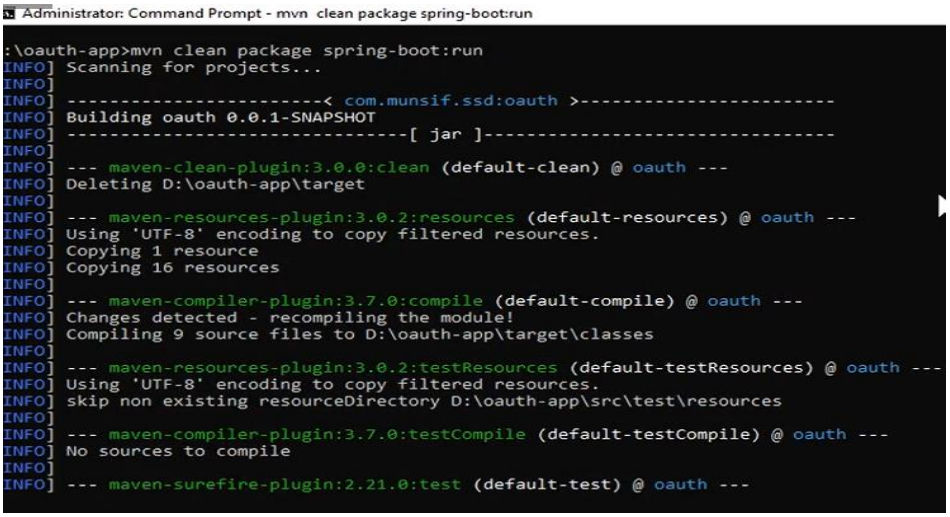
Application Processing

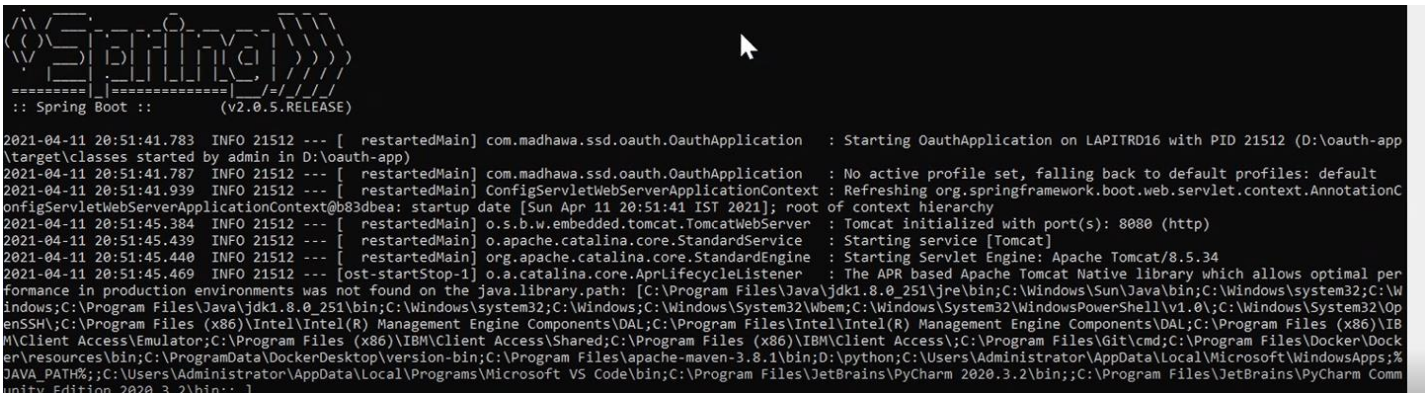
For develop this application used JDK 1.8 and Maven 3.8.1 Software. As an pre arrangement modify the environment variable as below for JDK and maven application path.



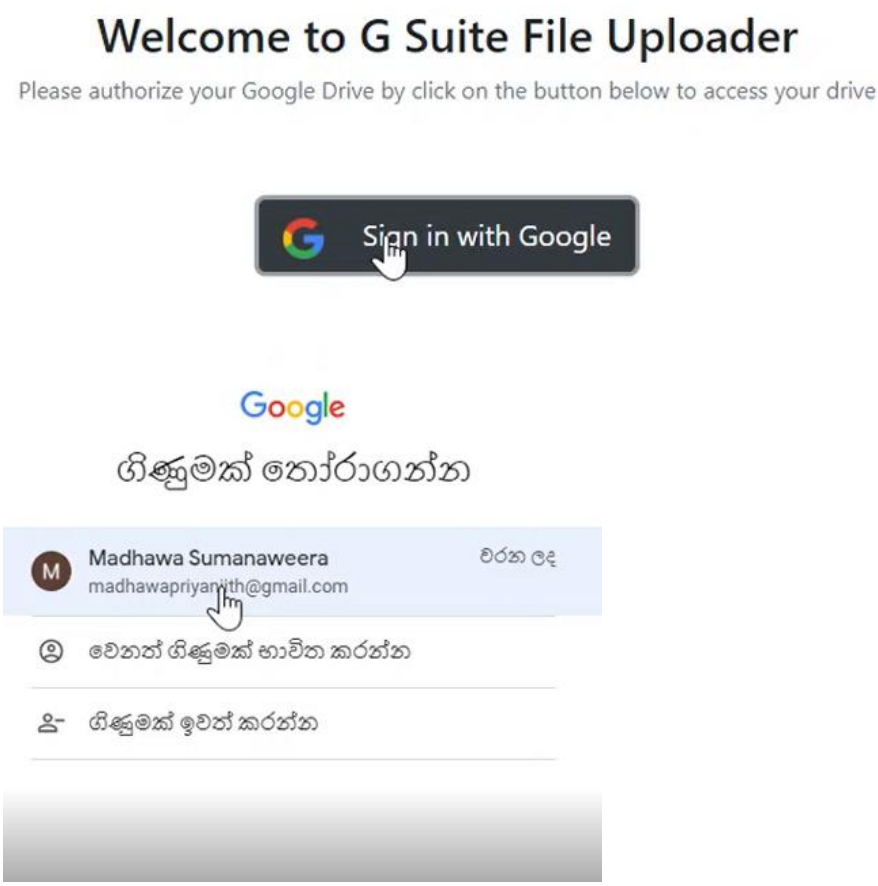
After the coding is completed (Please refer the Appendix) we are able to process the coding and Google authentication process.



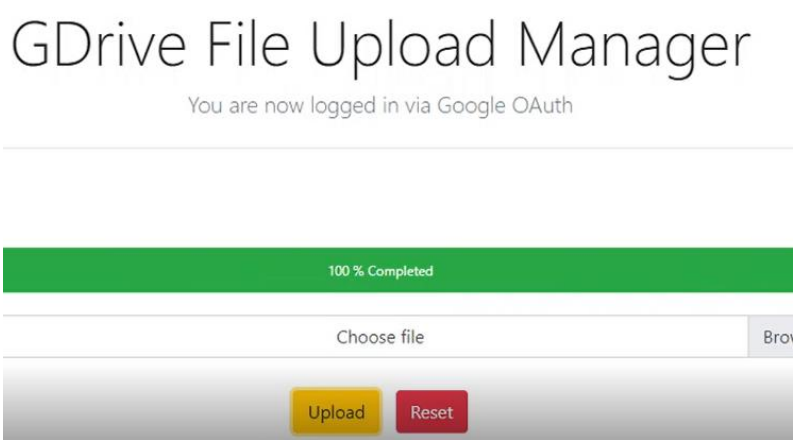




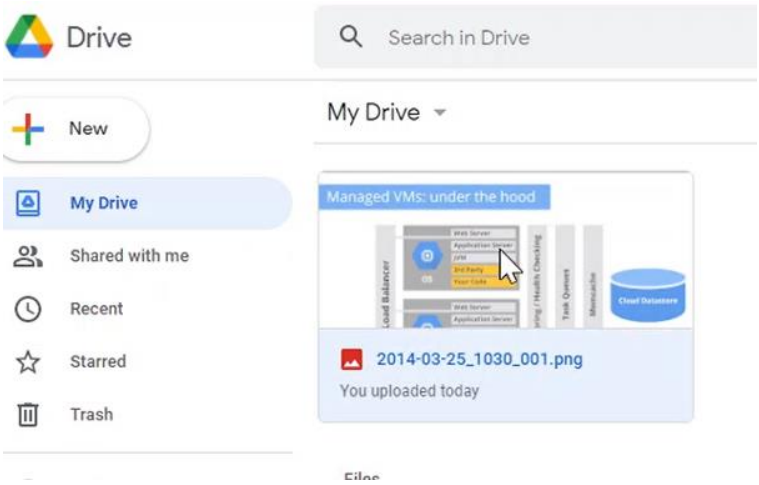
Prompt for select the google account for authentication


Welcome to G Suite File Uploader
Please authorize your Google Drive by click on the button below to access your drive
Sign in with Google
Google
ගිණුමක් තෝරාගන්න
Madhawa Sumanaweera
madhawapriyanjith@gmail.com
වරන ලද
වෙනත් ගිණුමක් භාවිත කරන්න
ගිණුමක් ඉවත් කරන්න


Google සමඟින් පුරන්න
Madhawa Sumanaweera
madhawapriyanjith@gmail.com
ඔබේ මුරපදය ඇතුළු කරන්න



Loading of the Google Drive



Appendix

Code Demonstration.

Below are the Java codes used for this OAuth Development.

 # Application Content

```
package com.madhawa.ssd.oauth.constant;

import java.util.Collections;
import java.util.List;

import com.google.api.client.http.HttpTransport;
import com.google.api.client.http.javanet.NetHttpTransport;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.jackson2.JacksonFactory;
import com.google.api.services.drive.DriveScopes;

public class ApplicationConstant {

	public static HttpTransport HTTP_TRANSPORT = new NetHttpTransport();
	public static JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();

	public static final List<String> SCOPES = Collections.singletonList(DriveScopes.DRIVE);

	public static final String USER_IDENTIFIER_KEY = "MY_TEST_USER";
	public static final String APPLICATION_NAME = "SSD OAuth Spring App";
	public static final String PARENT_FOLDER_NAME = "OAuth Demo App Uploaded";
}
```

 # Main Controller.

```
package com.madhawa.ssd.oauth.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.multipart.MultipartFile;

import com.madhawa.ssd.oauth.model.UploadFile;
import com.madhawa.ssd.oauth.service.AuthorizationService;
import com.madhawa.ssd.oauth.service.DriveService;

@Controller
public class MainController {

	private Logger logger = LoggerFactory.getLogger(MainController.class);

	@Autowired
	AuthorizationService authorizationService;

	@Autowired
	DriveService driveService;
```

```
/**
 * Handles the root request. Checks if user is already authenticated via SSO.
 *
 * @return
 * @throws Exception
 */
@GetMapping("/")
public String showHomePage() throws Exception {
	if (authorizationService.isUserAuthenticated()) {
		logger.debug("User is authenticated. Redirecting to home...");
		return "redirect:/home";
	} else {
		logger.debug("User is not authenticated. Redirecting to sso...");
		return "redirect:/login";
	}
}

/**
 * Directs to login
 *
 * @return
 */
@GetMapping("/login")
public String goToLogin() {
	return "index.html";
}

/**
```

```
 * Directs to home
 *
 * @return
 */
@GetMapping("/home")
public String goToHome() {
	return "home.html";
}

/**
 * Calls the Google OAuth service to authorize the app
 *
 * @param response
 * @throws Exception
 */
@GetMapping("/googlesignin")
public void doGoogleSignIn(HttpServletResponse response) throws Exception {
	logger.debug("SSO Called...");
	response.sendRedirect(authorizationService.authenticateUserViaGoogle());
}

/**
 * Applications Callback URI for redirection from Google auth server after user
 * approval/consent
 *
 * @param request
 * @return
 * @throws Exception
 */
```

```
@GetMapping("/oauth/callback")
public String saveAuthorizationCode(HttpServletRequest request) throws Exception {
	logger.debug("SSO Callback invoked...");
	String code = request.getParameter("code");
	logger.debug("SSO Callback Code Value..., " + code);

	if (code != null) {
		authorizationService.exchangeCodeForTokens(code);
		return "home.html";
	}
	return "index.html";
}

/**
 * Handles logout
 *
 * @return
 * @throws Exception
 */
@GetMapping("/logout")
public String logout(HttpServletRequest request) throws Exception {
	logger.debug("Logout invoked...");
	authorizationService.removeUserSession(request);
	return "redirect:/login";
}

/**
 * Handles the files being uploaded to GDrive
 *
```

```
	 * @param request
	 * @param uploadedFile
	 * @return
	 * @throws Exception
	 */
	@PostMapping("/upload")
	public String uploadFile(HttpServletRequest request, @ModelAttribute UploadFile uploadedFile)
throws Exception {
		MultipartFile multipartFile = uploadedFile.getMultipartFile();
		driveService.uploadFile(multipartFile);
		return "redirect:/home?status=success";
	}
}
```

 # Upload File

```
package com.madhawa.ssd.oauth.model;

import org.springframework.web.multipart.MultipartFile;

public class UploadFile {

	// private static final long serialVersionUID = 1L;
	private MultipartFile multipartFile;

	public MultipartFile getMultipartFile() {
		return multipartFile;
	}
```

```
	public void setMultipartFile(MultipartFile multipartFile) {
		this.multipartFile = multipartFile;
	}
}
```

 # Authorization Service

```
package com.madhawa.ssd.oauth.service.impl;

import java.io.IOException;
import java.io.InputStreamReader;

import javax.annotation.PostConstruct;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.google.api.client.auth.oauth2.Credential;
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeRequestUrl;
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
import com.google.api.client.googleapis.auth.oauth2.GoogleTokenResponse;
import com.google.api.client.util.store.FileDataStoreFactory;
import com.madhawa.ssd.oauth.constant.ApplicationConstant;
import com.madhawa.ssd.oauth.service.AuthorizationService;
import com.madhawa.ssd.oauth.util.ApplicationConfig;
```

```
@Service
public class AuthorizationServiceImpl implements AuthorizationService {

    private Logger logger = LoggerFactory.getLogger(AuthorizationServiceImpl.class);
    private GoogleAuthorizationCodeFlow flow;
    private FileDataStoreFactory dataStoreFactory;

    @Autowired
    private ApplicationConfig config;

    @PostConstruct
    public void init() throws Exception {
        InputStreamReader reader = new InputStreamReader(config.getDriveSecretKeys().getInputStream());
        dataStoreFactory = new FileDataStoreFactory(config.getCredentialsFolder().getFile());

        GoogleClientSecrets clientSecrets = GoogleClientSecrets.load(ApplicationConstant.JSON_FACTORY, reader);
        flow = new GoogleAuthorizationCodeFlow.Builder(ApplicationConstant.HTTP_TRANSPORT,
ApplicationConstant.JSON_FACTORY, clientSecrets,

    ApplicationConstant.SCOPES).setDataStoreFactory(dataStoreFactory).build();
    }

    @Override
    public boolean isUserAuthenticated() throws Exception {
        Credential credential = getCredentials();
        if (credential != null) {
```

```
                boolean isTokenValid = credential.refreshToken();
                logger.debug("isTokenValid, " + isTokenValid);
                return isTokenValid;
            }
            return false;
        }

        @Override
        public Credential getCredentials() throws IOException {
            return flow.loadCredential(ApplicationConstant.USER_IDENTIFIER_KEY);
        }

        @Override
        public String authenticateUserViaGoogle() throws Exception {
            GoogleAuthorizationCodeRequestUrl url = flow.newAuthorizationUrl();
            String redirectUrl =
url.setRedirectUri(config.getCALLBACK_URI()).setAccessType("offline").build();
            logger.debug("redirectUrl, " + redirectUrl);
            return redirectUrl;
        }

        @Override
        public void exchangeCodeForTokens(String code) throws Exception {
            // exchange the code against the access token and refresh token
            GoogleTokenResponse tokenResponse =
flow.newTokenRequest(code).setRedirectUri(config.getCALLBACK_URI()).execute();
            flow.createAndStoreCredential(tokenResponse,
ApplicationConstant.USER_IDENTIFIER_KEY);
        }
        @Override
```

```
        public void removeUserSession(HttpServletRequest request) throws Exception {
//                  HttpSession session = request.getSession(false);
//      session = request.getSession(true);
//      if (session != null) {
//          session.invalidate();
//          logger.info("Logged Out...");
//      }
                    // revoke token and clear the local storage
                    dataStoreFactory.getDataStore(config.getCredentialsFolder().getFilename()).clear();
          }

}
```

 # Drive Service

```
package com.madhawa.ssd.oauth.service.impl;

import javax.annotation.PostConstruct;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import com.google.api.client.auth.oauth2.Credential;
import com.google.api.client.http.FileContent;
import com.google.api.services.drive.Drive;
import com.google.api.services.drive.model.File;
import com.madhawa.ssd.oauth.constant.ApplicationConstant;
```

```
import com.madhawa.ssd.oauth.service.AuthorizationService;
import com.madhawa.ssd.oauth.service.DriveService;
import com.madhawa.ssd.oauth.util.ApplicationConfig;

@Service
public class DriveServiceImpl implements DriveService {

	private Logger logger = LoggerFactory.getLogger(DriveServiceImpl.class);

	private Drive driveService;

	@Autowired
	AuthorizationService authorizationService;

	@Autowired
	ApplicationConfig applicationConfig;

	@PostConstruct
	public void init() throws Exception {
		Credential credential = authorizationService.getCredentials();
		driveService = new Drive.Builder(ApplicationConstant.HTTP_TRANSPORT,
ApplicationConstant.JSON_FACTORY, credential)
				.setApplicationName(ApplicationConstant.APPLICATION_NAME).build();
	}

	@Override
	public void uploadFile(MultipartFile multipartFile) throws Exception {
		logger.debug("Inside Upload Service...");
```

```
                String path = applicationConfig.getTemporaryFolder();
                String fileName = multipartFile.getOriginalFilename();
                String contentType = multipartFile.getContentType();

                java.io.File transferedFile = new java.io.File(path, fileName);
                multipartFile.transferTo(transferedFile);

                File fileMetadata = new File();
                fileMetadata.setName(fileName);

                FileContent mediaContent = new FileContent(contentType, transferedFile);
                File file = driveService.files().create(fileMetadata,
mediaContent).setFields("id").execute();

                logger.debug("File ID: " + file.getName() + ", " + file.getId());
        }

}
```

 # Application Config

```
package com.madhawa.ssd.oauth.util;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.io.Resource;
import org.springframework.stereotype.Component;
```

```
@Component
@PropertySource("classpath:application.properties")
@ConfigurationProperties
public class ApplicationConfig {

	@Value("${google.oauth.callback.uri}")
	private String CALLBACK_URI;

	@Value("${google.secret.key.path}")
	private Resource driveSecretKeys;

	@Value("${google.credentials.folder.path}")
	private Resource credentialsFolder;

	@Value("${myapp.temp.path}")
	private String temporaryFolder;

	public String getCALLBACK_URI() {
		return CALLBACK_URI;
	}

	public void setCALLBACK_URI(String CALLBACK_URI_p) {
		CALLBACK_URI = CALLBACK_URI_p;
	}

	public Resource getDriveSecretKeys() {
		return driveSecretKeys;
	}
```

```
	public void setDriveSecretKeys(Resource driveSecretKeys) {
		this.driveSecretKeys = driveSecretKeys;
	}

	public Resource getCredentialsFolder() {
		return credentialsFolder;
	}

	public void setCredentialsFolder(Resource credentialsFolder) {
		this.credentialsFolder = credentialsFolder;
	}

	public String getTemporaryFolder() {
		return temporaryFolder;
	}

	public void setTemporaryFolder(String temporaryFolder) {
		this.temporaryFolder = temporaryFolder;
	}

}
```

 # OAuth Application.

```
package com.madhawa.ssd.oauth;

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

@SpringBootApplication
```

```
public class OauthApplication {

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

 # Google OAuth

```
google.secret.key.path=classpath:keys/gsuite_drive_keys.json
google.oauth.callback.uri=http://localhost:8080/home
google.credentials.folder.path=file:D:\\oauth-app\\src\\main\\resources\\credentials

# Logger Level
logging.level.com.munsif.ssd.oauth=DEBUG

# Spring MultiPart File
spring.servlet.multipart.max-file-size=50MB
spring.servlet.multipart.max-request-size=50MB
spring.http.multipart.enabled=true
#spring.resources.cache.cachecontrol.max-age=1

myapp.temp.path=D:\\oauth-app\\src\\main\\resources\\tempfolder
```

Referance

1) https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2
2) https://oauth.net/2/
3) https://medium.com/@munsifmusthafa03/building-a-file-upload-service-to-your-google-drive-using-oauth-2-0-d883d6d67fe8