```
++++++++++++++++++++++
Oauth using gmail login
++++++++++++++++++++++

Dummy credentials of gmail
 Username : raghavtesting6@gmail.com
 Password : Inspired@2023

1. Register Clientapp with google api
      go to http://console.developers.google.com/
                  |
            Select a project section
                  |
            Create a new project
            a. Enter a new project name(iNeuronApp)
            b. Ignore location text box(default will be No Organisation)
            c. click on create

2. Make iNeuronApp as the active project by selecting it
3. Go to Credentials -> click on create credentials -> click on OauthClientId->
configureConsent screen
      -> select external -> click on create
            provide the following details in the opened tab
                  a. AppName
                  b. UserSupport email
                  c. provider developer contact information
                  d. click on save and continue, save and continue

4.Go to credentials -> click on create credentials -> Select OuthClientId ->
application type : webapplication
      now select URI : http://localhost:9999 (application uri)
      now select Authorized redirect URI :
http://localhost:9999/login/oauth2/code/google (fixed uri)
        now click on create

5.Copy the generated client-id and secrete-id as shown below
      client-id : 971870997040-
8hpr3j36r5sn478blkk8rtln2d4eev50.apps.googleusercontent.com
      secrete-id: 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

6. add these information in application.properties/.yml file
spring:
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: 971870997040-
8hpr3j36r5sn478blkk8rtln2d4eev50.apps.googleusercontent.com
            client-secret: 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

7. Create a SecurityConfigApp to control the endpoints for Authentication
++++++++++++++++++++++
SecurityConfigApp.java
++++++++++++++++++++++
package in.ineuron.nitin.configuration;

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.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfigApp {

      @Bean
      public SecurityFilterChain configureFilter(HttpSecurity http) throws
Exception {
            //authorization
            http.authorizeHttpRequests()
            .antMatchers("/","/home","/login").permitAll()
            .anyRequest().authenticated()
            .and().formLogin()
            .and().oauth2Login();//Developing custom login form having hyperlink to
login as fb user
            return http.build();
      }

}

7. Add the following line in login.html page
<br>
<h1 style="color:red; text-align: center">
      <a href="/oauth2/authorization/google">Google LOGIN</a>
</h1>

8. Run the application as springboot application

request
 a. http://locahost:9999/home -> Hello, Welcome to Home page of RedBus.com
 b. http://locahost:9999/after
            |
      display login page
            |
       click on google link
            |
        it redirect to gmail login page
            |
         chooose the user
            |
        Hello,Succesfully logged into RedBus.com

 c. http:///localhost:9999/user
            |
    display the user data in Json format

++++++++++++++++++++++
JWT(JSON Web Tokenizer)
++++++++++++++++++++++
=> While implementing security in client-server application(either webapplication
or distrubuted application) the client authentication is very important.
=> What is clientAuthentication?
   Ans.Making the enduser to submit credentials or identity to the server
application and making server app validating those details is called
      "Client-Authentication".
```

eg: Logging by submitting username and password for any applications.
Logging by using faceid
Logging by using otp
etc...

Client Authentication in Client-Server application can be done in 2 ways
a. Stateful Authentication => Working with HttpSession based SessionManagement, Session Tracking technique
Server will hold the client state during a session.

refer: .png file
=> state reprsents the data,like client data(username,password) that is stored at the server app and it will be used across multiple requests.
Since the server remembers the data we say such type of authentication as "Stateful Authentication".
=> On a webapplication, if we apply session tracking then webapplicaion remembers and stores the clientdata(state) across multiple requests.
This makes a webapplication a "StateFul" WebApplication.

b. Stateless Authentication => Working with Tokens, Generating the token we can get the Support of JSON WEB Tokenizer(JWT)
Server doesn't hold client state in any angle.

refer : .png file

=> The server app will generate token, which goes to client and the token will be stored at the client side.
=> This token will be sent to Server along with every request, Generally token will have an expriy time.
=> Once the token is expired at the server side, the client sent token will fail in token validation and the response will be sent
as "login page/error page".
=> Here no state(user,pwd) of Client are stored in server app so we say as "StateLess Authentication".

Q> Where should Stateful Authentication and Where should we use Stateless Authentication?
Regular webapplication which interacts with enduser directly uses "Stateful Authentication".
eg:
Facebook.com,amazon.com,ineuron.com,redbus.com,gmail.com,flipkart.com (SessionTracking)

In Server to Server communication nothing but restful services interaction we need to go for "Stateless Authentication"
eg: flipkart interacting with paytm or gpay or phonepe.

++
Different places where Stateless Authentication will be used
++
1. Server to Server Communication(B2B)
2. While enabling the horizontal scaling for the aplication(Microservices)
3. While Implementing Resource Grant Security(OAuth Application)

refer: .png

Benefits of Working with Stateless Authentication
 a. Since it is stateless concept, it doesn't allocate any memory at the server side, so no burden on the server
 b. It is very good for Distrubuted and also for Microservices based application development.

Limitations
 a. The token is validated on evey request.
 b. If the token is passed on to other users,they can access the entire client information and perform all operations as a Authorized client.

To implement this token based authorization, we need to generate and validate the tokens by using the support of "JWT".
JWT is an OpenSource API, that supports generating the tokens based on the given client details and secrete-key(password)

JWT
 => it is used for stateless authentication, which is capable of generating tokens.
 => Information is :: https://jwt.io/
 => Sample JWT token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.
SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
 => The content which is not readable is called "Encoded Content".
 => JWT token message is "Encoded Content".
 => Decoding the content means => getting readable content from the encoded content.
 => JWT toke contains 3 parts
      a. HEAD   : JWT Specific information like algorithm,type, etc..
      b. PAYLOAD   : Claims info like clienid,clientname,issuername,expirydata
      c. SIGNATURE : BASE64Encoded(header) + BASE64Encoded(body/payload) mixed with Secrete-key.

            refer :.png

To generate the token and to get the claims information from generated token we use "JWT" API.

maven depenancy
a. jjwt
b. jaxb-api

pom.xml
+++++++
```
<dependency>
      <groupId>io.jsonwebtoken</groupId>
      <artifactId>jjwt</artifactId>
      <version>0.9.1</version>
</dependency>

<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
      <groupId>javax.xml.bind</groupId>
      <artifactId>jaxb-api</artifactId>
      <version>2.3.1</version>
</dependency>
```

++++++++++++
JWTUtil.java

++++++++++++
```
package in.ineuron.utility;

import java.util.Base64;
import java.util.Date;
import java.util.concurrent.TimeUnit;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

public class JWTUtil {

	public static String generatToken(String id,String subject,String secretekey)
{

		return Jwts
			.builder()
			.setId(id)
			.setSubject(subject)
			.setIssuer("PWSKILLS")
			.setIssuedAt(new Date(System.currentTimeMillis()))
			.setExpiration(new Date(System.currentTimeMillis() +
TimeUnit.MINUTES.toMillis(1)))
			.signWith(SignatureAlgorithm.HS256,Base64.getEncoder().encode(sec
retekey.getBytes()))
			.compact();

	}

	public static Claims getClaims(String secretkey,String token) {
		return Jwts
				.parser()
				.setSigningKey(Base64.getEncoder().encode(secretkey.getByte
s()))
				.parseClaimsJws(token)
				.getBody();
	}

	public static boolean isTokenValid(String secretkey,String token) {
		Date tokenExpiryDate = getClaims(secretkey, token).getExpiration();
		Date date =new Date();
		boolean result = date.before(tokenExpiryDate);
		return result;
	}

	public static String getSubject(String secretekey,String token) {
		return getClaims(secretekey, token).getSubject();
	}

	public static Date getExpiryDate(String secretekey,String token) {
		return getClaims(secretekey, token).getExpiration();
	}

}
```

++++++++++++++
JWTTestApp.java
++++++++++++++

```
package in.ineuron.test;

import in.ineuron.utility.JWTUtil;
import io.jsonwebtoken.Claims;

public class JWTTestApp {

	public static void main(String[] args) throws Exception {

		String token = JWTUtil.generatToken("TRA_SCTN", "UPI-PIN", "7234");
		System.out.println(token);

		Claims claims = JWTUtil.getClaims("7234", token);
		System.out.println("Subject info :: " + claims.getSubject());
		System.out.println("Client id    :: " + claims.getId());
		System.out.println("Exprity date :: " + claims.getExpiration());
		System.out.println("IssuedDate time :: " + claims.getIssuedAt());

		System.out.println("IS Token valid      :: " +
JWTUtil.isTokenValid("7234", token));
		System.out.println("SubjectInfo         :: " +
JWTUtil.getSubject("7234", token));
		System.out.println("ExpiryDate and Time :: " +
JWTUtil.getExpiryDate("7234", token));
	}
}

Output
HEAD         : eyJhbGciOiJIUzI1NiJ9.
DATA/Payload :
eyJqdGkiOiJUUkFfU0NUTiIsInN1YiI6IlVQSS1QSU4iLCJpc3MiOiJQV1NLSUxMUyIsImlhdCI6MTY5NTU
0Mzc4OSwiZXhwIjoxNjk1NTQzODQ5fQ.
SIGNATURE    : UKLLsnWEjhAsDJtOOCZnkh-HhkNKzgHwzK3Kh5QBFWQ
Subject info :: UPI-PIN
Client id    :: TRA_SCTN
Exprity date :: Sun Sep 24 13:54:09 IST 2023
IssuedDate time :: Sun Sep 24 13:53:09 IST 2023
IS Token valid      :: true
SubjectInfo         :: UPI-PIN
ExpiryDate and Time :: Sun Sep 24 13:54:09 IST 2023

++++++++++++++++
RealtimeScenario
++++++++++++++++
1. Client------STATELESS Authentication----------RESTAPI

	request -> GET: http://localhost:9999/hello
	reponse -> {
		    "timestamp": "2023-09-24T08:44:23.278+00:00",
		    "status": 403,
		    "error": "Forbidden",
		    "message": "Access Denied",
		    "path": "/hello"
		}

	request ->POST     : http://localhost:9999/authenticate
		    select body tab,choose raw and select JSON type
		    {
			    "username":"admin",
```

```
                        "password":"admin@123"
                    }
                   click on send
      repsonse -> {
                    "jwt":
                    "eyJhbGciOiJIUzI1NiJ9.

      eyJzdWIiOiJhZG1pbiIsImV4cCI6MTY5NTU4MTMxMSwiaWF0IjoxNjk1NTQ1MzExfQ.
                    qQoboA3ZF8lwoPubFspKOqARKU6Y8KWI2n_em5gJXrA"
                  }


2. Client -----STATELESS Authentication(token)----------RESTAPI
      a. send the request with token
            request -> GET: http://localhost:9999/hello
            Choose Authorization tab
            a. select Bearer token
            b. paste the


token :eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImV4cCI6MTY5NTU4MTMxMSwiaWF0IjoxNjk
1NTQ1MzExfQ.qQoboA3ZF8lwoPubFspKOqARKU6Y8KWI2
                    click on send

      b. response form the endpoints
            Hello World

++++++++++++++++++
JUnit with Mockito
++++++++++++++++++
Wednesday
Thursday

++++++++++++++++++++++++++++++++
Springboot with React integration
++++++++++++++++++++++++++++++++
Saturday
Sunday
```