Tutorials are interaction sessions for you to ask questions and for the instructor to check your understanding of SPA requirements analysis and API design. In order to benefit from tutorials, *you are expected to come prepared with written solutions.*

API Documentation

Q1. . . . .

Identify a few strategies to optimize the evaluation. Describe how Query Evaluator would evaluate the following query:

assign a, a1, a2; statement s1, s2, s3; variable v1, v2;
Select <s1, a, a1, v2> such that Uses (5, "y") and Uses (s1, v1) and Affects (a1, a2) with a1.stmt#=20 such that Parent (s1, s2) pattern a2 ("a", _) such that Next (s2, s3) and Modifies (s1, "x") and Modifies (a, v2) pattern a (v1, _"x"_)

Assume the following list of answers for selected clauses:

Uses (5, "y") is true

Uses	
s1	v1
2	y
5	y
5	z
7	y
8	y

Parent	
s1	s2
2	9
5	6
5	7
7	8

Next	
s2	s3
6	7
7	8
7	12
8	9

Modifies(s1,"x")
s1
6
5
7
20

Modifies	
a	v2
11	y
20	x

Pattern a(v1, _"x"_)	
a	v1
11	y
20	x
4	y

Q2. . . . .

Based on the following description of the OAuth 2.0 authorization framework, discover and document the APIs for:

- Authorization Server: Client Registration, Authorization Code Grant and Implicit Grant interaction scenarios
- Resource Server: Client accessing the hosted resource.

Optional: Draw a sequence diagram for the flow of the Authorization Code Grant. Label the arrows in the diagram with the API operations discovered.

The OAuth 2.0 authorization framework enables a client (i.e., a third-party application) to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the client to obtain access on its own behalf.

The flow of interactions in this framework is as follows: An application (e.g., a website, a browser extension, or a mobile application) wants to access a resource (e.g., your Facebook profile) which is hosted on a resource server. To access the resource, the application should register as a client of the authorization server. Upon successful registration, the client seeks permission from the authorization server. The authorization server then proceed to obtain an approval from the resource owner (you) and issue a token to the client. Henceforth, the client makes requests to the resource server using this token.

Tokens are random strings generated by the authorization server and are issued to a client when the client requests for them. There are 2 types of tokens, as shown in Table 1.

There are three possible interaction scenarios between the resource owner, resource server, the client application and the authorization server:

Client Registration
If an application wants to retrieve data from a resource server using OAuth 2.0, it has

	Access Token	Refresh Token
Purpose	To access the resource server	To renew an expired access token
Used between	Client and resource server	Client and authorization server

Table 1: Types of tokens

Example	
Resource Owner	you
Resource Server	a Google server
Client	a website
Authorization Server	a Google server

Table 2: An example for Authorization Code Grant

to register as a client of the authorization server. The registration requires the name of the application, redirect URLs of the client for receiving the tokens, and the types of authorization (grant types) that will be used by the client.

Authorization Code Grant
It is used when the client is a website. It allows the client to obtain a long-lived access token.

An example scenario based on Table 2 is given below:
A website wants to obtain information about your Google profile.

1. You are redirected by the client (the website) to the authorization server (Google).
2. If you authorize the access, the authorization server sends an authorization code to the client in the callback response.
3. This code is exchanged for an access token between the client and the authorization server.
4. The website is now able to use this access token to query the resource server (Google again) and retrieve your profile data.

You never see the access token but it is stored by the website (e.g., in session). Google also sends other information with the access token, such as the token lifetime and eventually a refresh token. This is the ideal scenario and the safer one because the access token is not passed on the client side.

Implicit Grant
It is typically used when the client is implemented in a browser using a scripting language such as Javascript. This grant type does not allow the issuance of a refresh token.

An example scenario based on Table 3 is given below:
A browser extension wants to obtain information about your Facebook profile.

Example	
Resource Owner	you
Resource Server	a Facebook server
Client	a browser extension implemented usingAngularJS
Authorization Server	a Facebook server

Table 3: An example for Authorization Code Grant

1. You are redirected by the client (the browser extension) to the authorization server (Facebook).
2. If you authorize the access, the authorization server sends a redirection URI, which contains the access token, to the client.
3. This access token can now be retrieved and used by the client to query the resource server.

Example of callback:
`http://example.com/oauthcallback#access_token=MzJmNDc3M2VjMmQzN`

Example of query:
`https://graph.facebook.com/me?access_token=MzJmNDc3M2VjMmQzN`