IN4MATX 133: User Interface Software

Lecture 12:
Authentication & Authorization

Professor Daniel A. Epstein
TA Jamshir Goorabian
TA Simion Padurean

Today’s goals

By the end of today, you should be able to…

- Differentiate authentication from authorization
- Describe the utility of supporting authentication and authorization in interfaces
- Explain and implement the different stages to authenticating via OAuth
- Describe the advantages and disadvantages of OpenId

What is authentication?

- The process of establishing and verifying identity
- Identification: who are you? (username, account number, etc.)
- Authentication: prove it! (password, PIN, etc.)

What is authorization?

- Once we know a user’s identify,
  we must decide what they are allowed to access or modify

- One way is the app defines permissions upfront based on a user’s role
  - A student can access their own grades, but not modify them
  - A TA and a professor can access and modify everyone’s grades

- Another way is for the app to request certain permissions of the user
  - A Twitter app may ask, “can I Tweet on your behalf?”

Multi-factor authentication

- Should be a mix of things that you *have* or *posess* and things that you *know*
- ATM machine: 2-factor authentication
  - ATM card: something you *have*
  - PIN: something you *know*
- Password + code delivered via SMS: 2-factor authentication
  - Password: something you *know*
  - Code: validates that you *possess* your phone
- Two passwords != Two-factor authentication

Question

Which of these is an example of “good” two-factor authentication?

Ⓐ A government agency requiring a birth certificate and a passport

Ⓑ A store requiring a membership card and a PIN

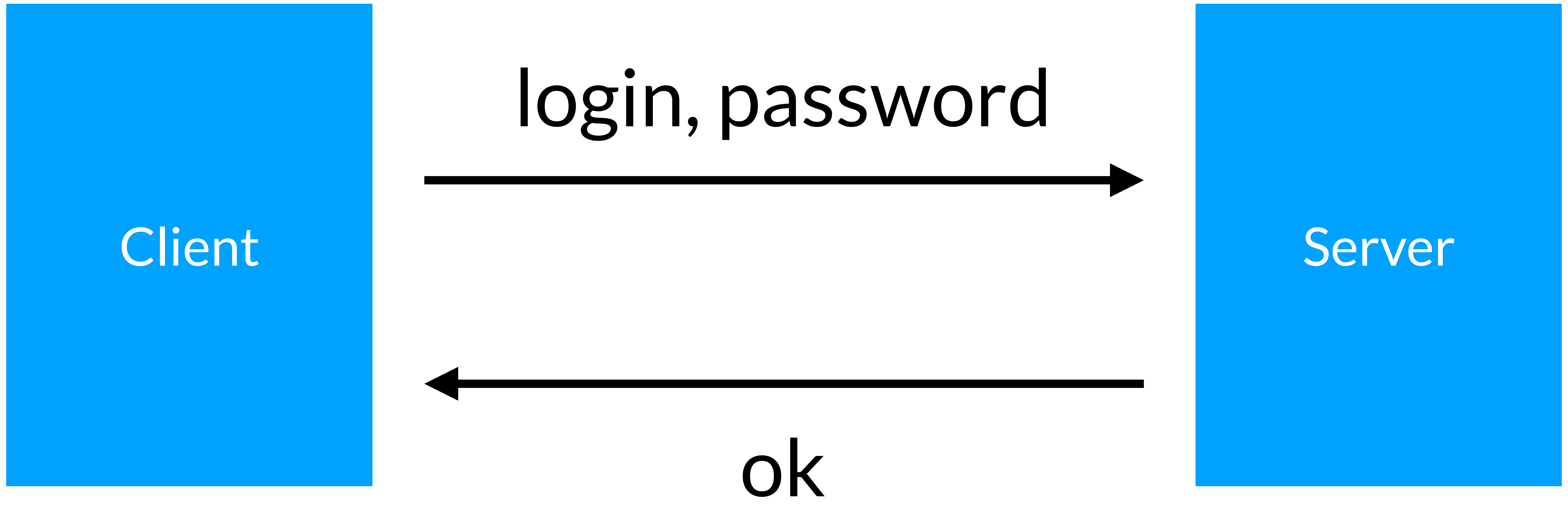
Ⓒ A website requiring a password and a security question

Ⓓ Two of the above

Ⓔ All of the above

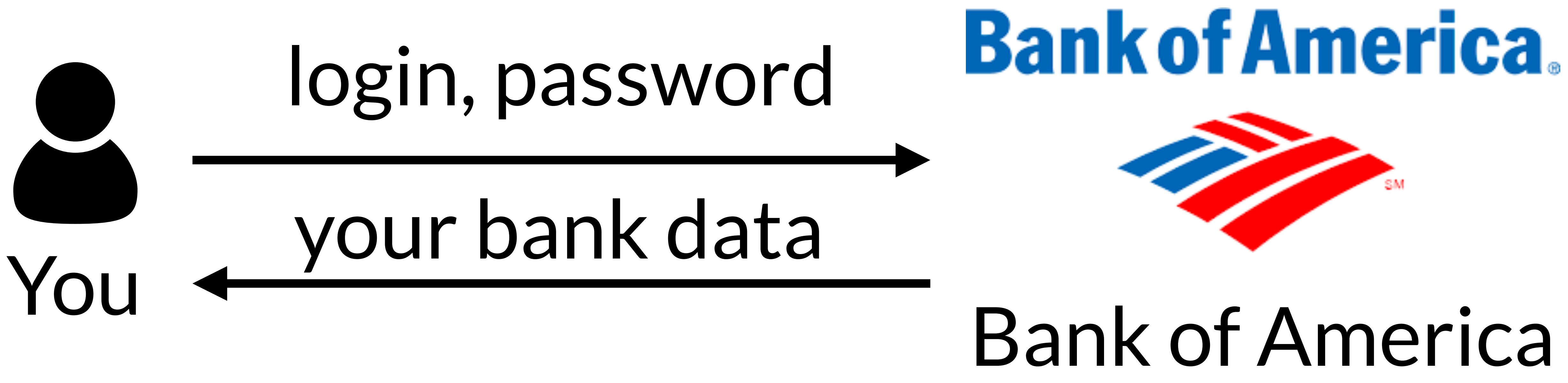
Password protocol

- Send a login and a password to a server
- Server checks your credentials and okays you
- Need to trust that the server is storing your password securely



Password protocol: sending data

- Once you've logged in,
  the server can send you whatever data you're allowed to see



login, password

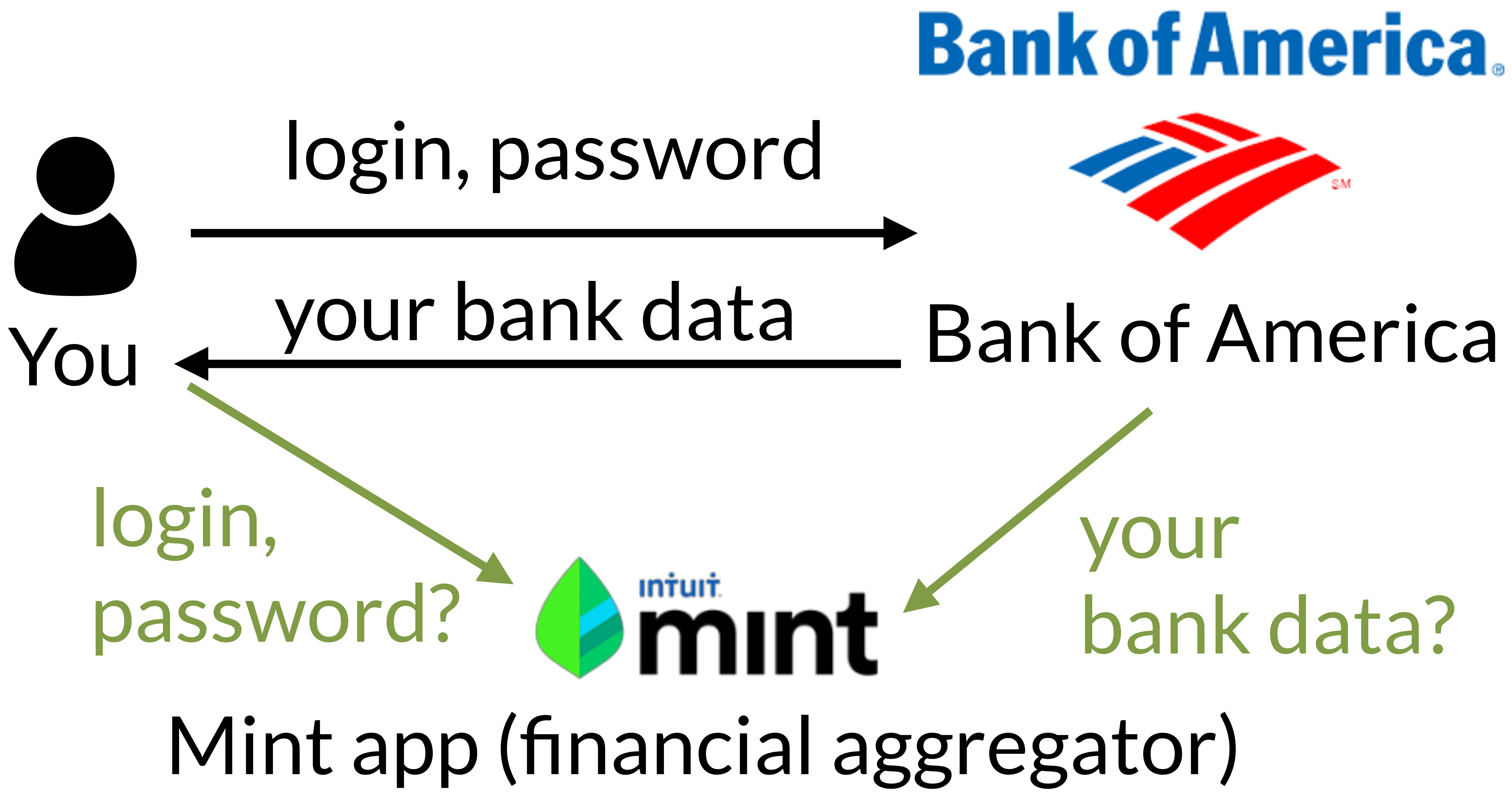
your bank data

You

Bank of America

Sending data to a third party

- You want to send data that a server has to a third party
  - You could give them your username and password…
  - Why is this a bad idea?



Sending data to a third party

- Now you have to trust *another* service to manage your password
- What if you don’t want them to have full access?
  - e.g., you want Mint to load your savings account but not your checking account
- What if you want to revoke access later?
  - Can change your password, but that’s not a good solution

Oauth 2.0

- Open authentication
- Goal: support users in granting access to third-party applications
  - Do not require users to share their passwords with the third-party applications
  - Allow users to revoke access from the third parties at any time

Oauth 2.0 history

- There was a 1.0
  - It was complex (worse than 2.0)
  - It had security vulnerabilities
  - It shouldn’t be used anymore
- Google, Twitter, & Yahoo! teamed up to propose 2.0
- 2.0 is not compatible with 1.0

https://en.wikipedia.org/wiki/OAuth#OAuth_2.0

Oauth 2.0 terminology

- Client
  - Third-party app who wants to access resources owned by the *resource owner* (e.g., app you develop)
- Resource owner (user)
  - Person whose data is being accessed, which is stored on the *resource server*
- Resource server
  - App that stores the resources (e.g., Spotify, Google, Facebook)
- Authorization and Token endpoints
  - URIs from where a resource owner authorizes requests

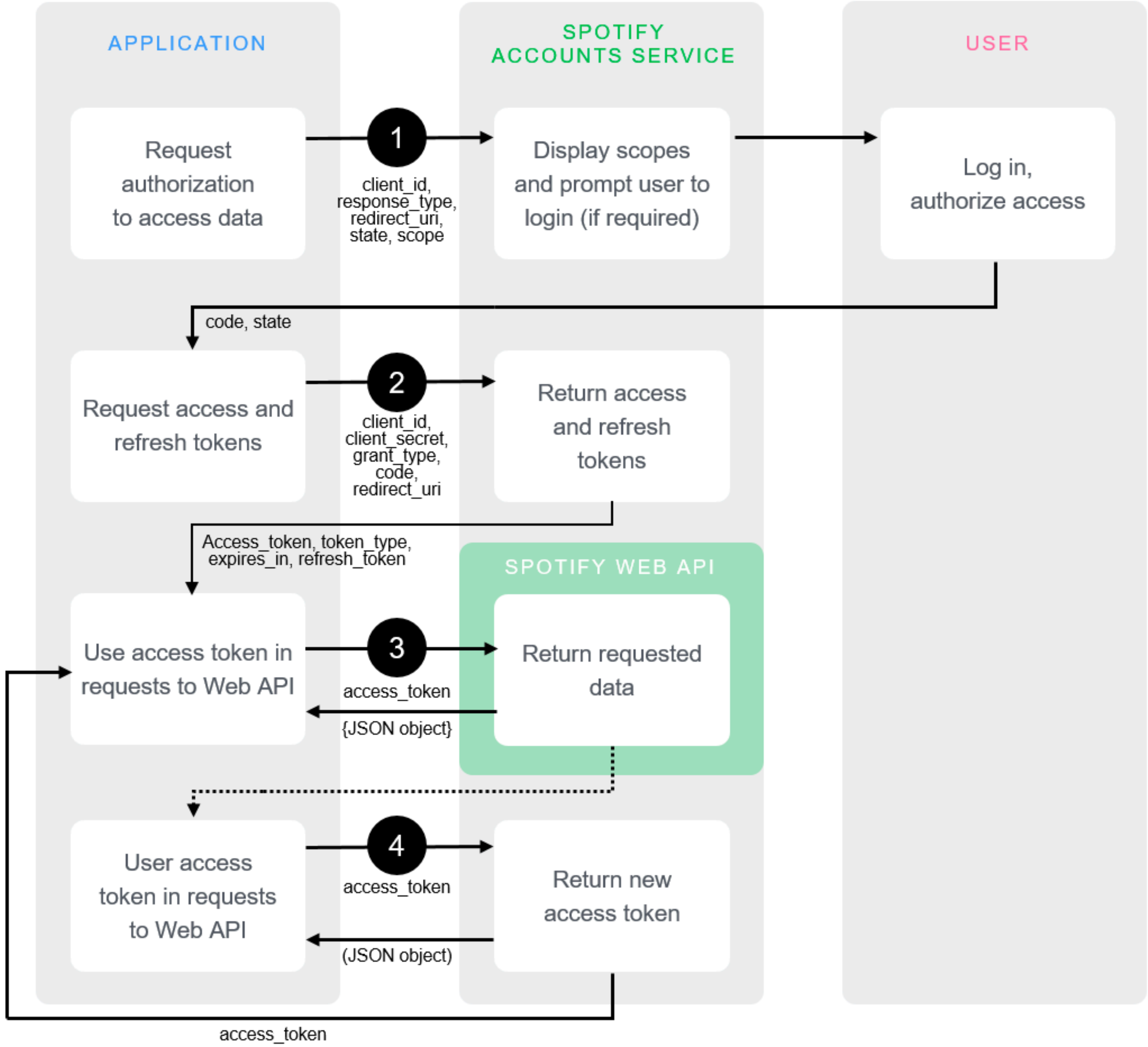
Oauth 2.0 terminology

- Authorization code
  - A string the client uses to request access tokens
- Access token
  - A string the client uses to access resources (e.g., songs on Spotify, Tweets, etc.)
  - Expires after some amount of time
- Refresh token
  - Once the access token expires, can be exchanged for a new access token

Oauth 2.0 steps

1. Request authorization
2. Get access token
3. Make API calls
4. Refresh access token

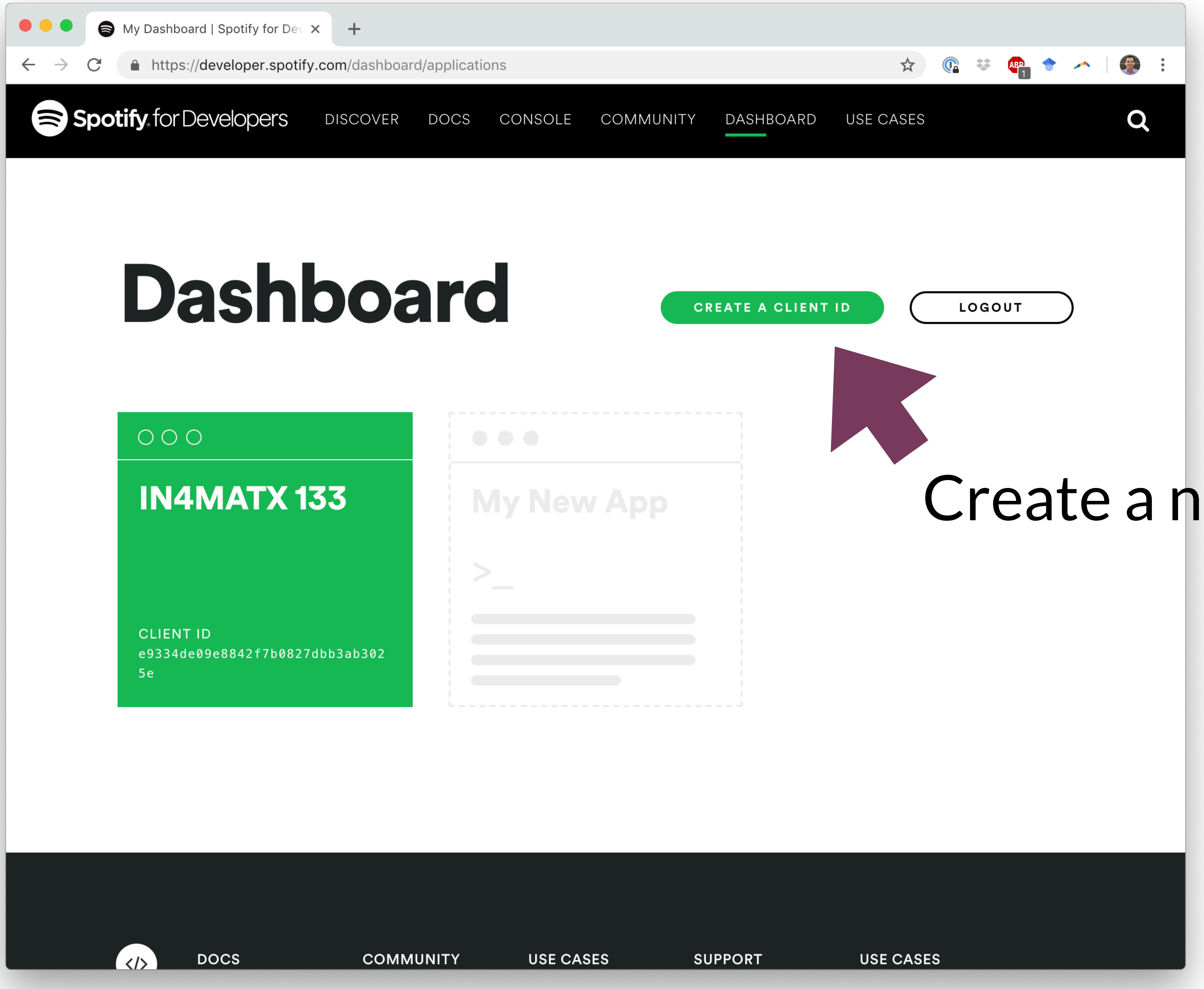
Oauth 2.0 steps



APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Oauth 2.0 and Spotify

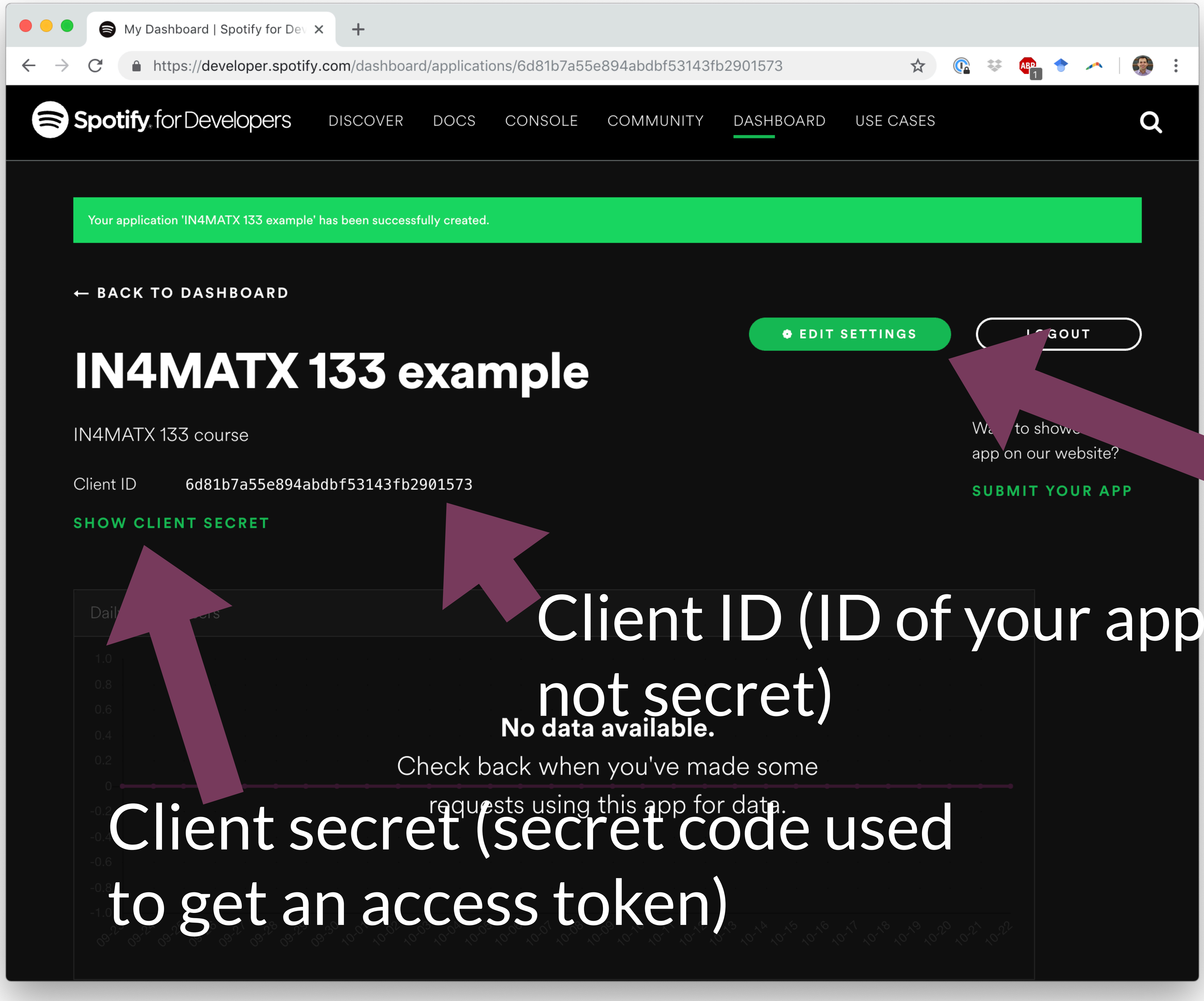


CLIENT ID
e9334de09e8842f7b0827dbb3ab302
5e

Create a new ID

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Oauth 2.0 and Spotify



My Dashboard | Spotify for De
https://developer.spotify.com/dashboard/applications/6d81b7a55e894abdbf53143fb2901573
Spotify for Developers
DISCOVER
DOCS
CONSOLE
COMMUNITY
DASHBOARD
USE CASES

Your application 'IN4MATX 133 example' has been successfully created.

← BACK TO DASHBOARD

IN4MATX 133 example

IN4MATX 133 course

Client ID 6d81b7a55e894abdbf53143fb2901573

SHOW CLIENT SECRET

EDIT SETTINGS
LOGOUT
Wa to show app on our website?
SUBMIT YOUR APP
Dail
1.0
0.8
0.6
0.4
0.2
0

No data available.

Check back when you've made some requests using this app for data.

Client ID (ID of your app, not secret)

Client secret (secret code used to get an access token)

Need to specify what URI to return to (redirect URI)

https://developer.spotify.com/documentation/general/guides/authorization-guide/

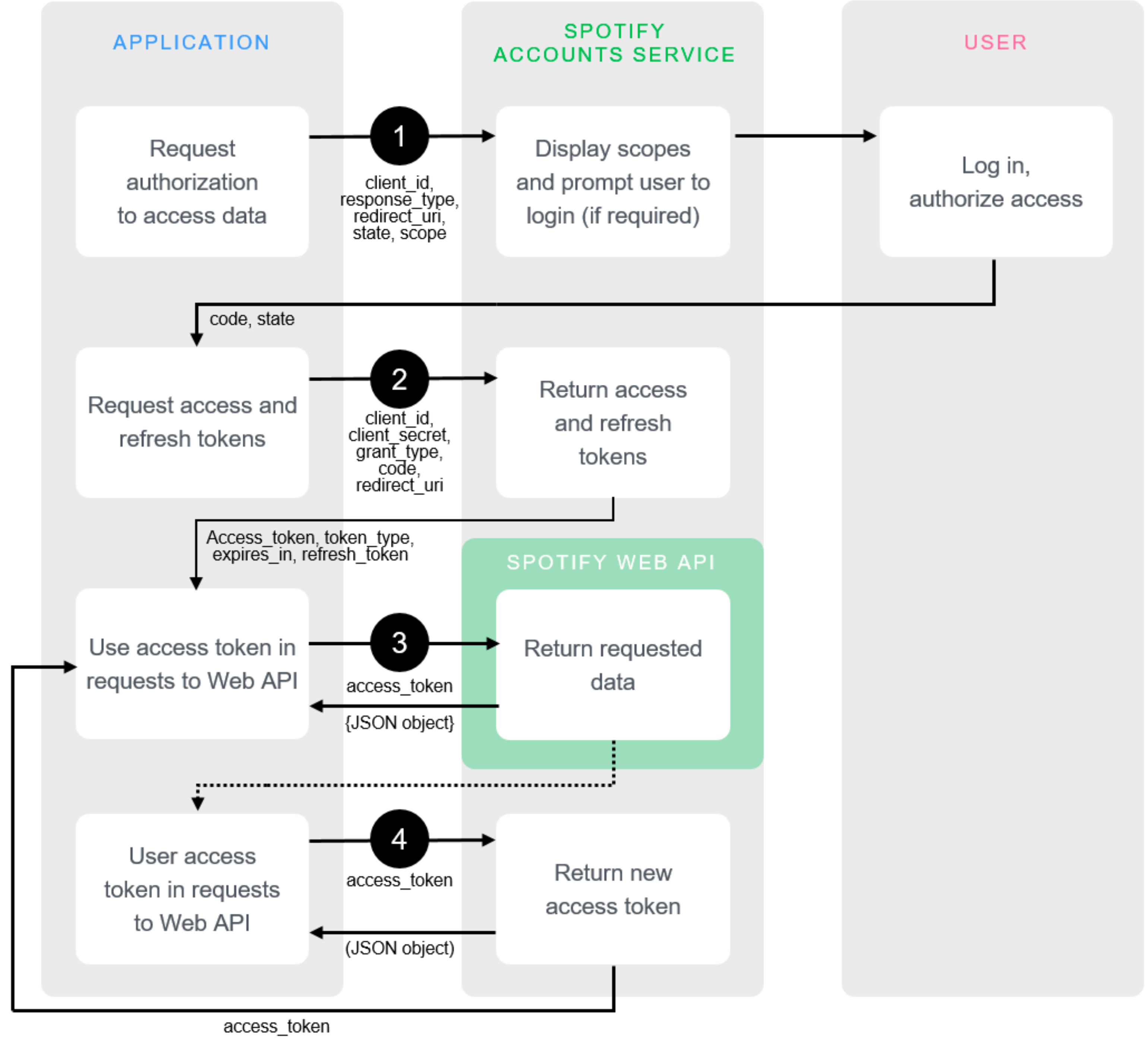
Oauth 2.0 on server-side JavaScript

- This example will walk through the Oauth flow for server-side JavaScript (like Node.js/Express)

- There are browser-side ways of doing (some parts of) Oauth

- For A3, you’ll send all browser-side requests to an Express server

Step 1: request authorization to access data



APPLICATION

SPOTIFY ACCOUNTS SERVICE

USER

Request authorization to access data

1

client_id, response_type, redirect_uri, state, scope

Display scopes and prompt user to login (if required)

Log in, authorize access

code, state

Request access and refresh tokens

2

client_id, client_secret, grant_type, code, redirect_uri

Return access and refresh tokens

Access_token, token_type, expires_in, refresh_token

SPOTIFY WEB API

Use access token in requests to Web API

3

access_token

{JSON object}

Return requested data

User access token in requests to Web API

4

access_token

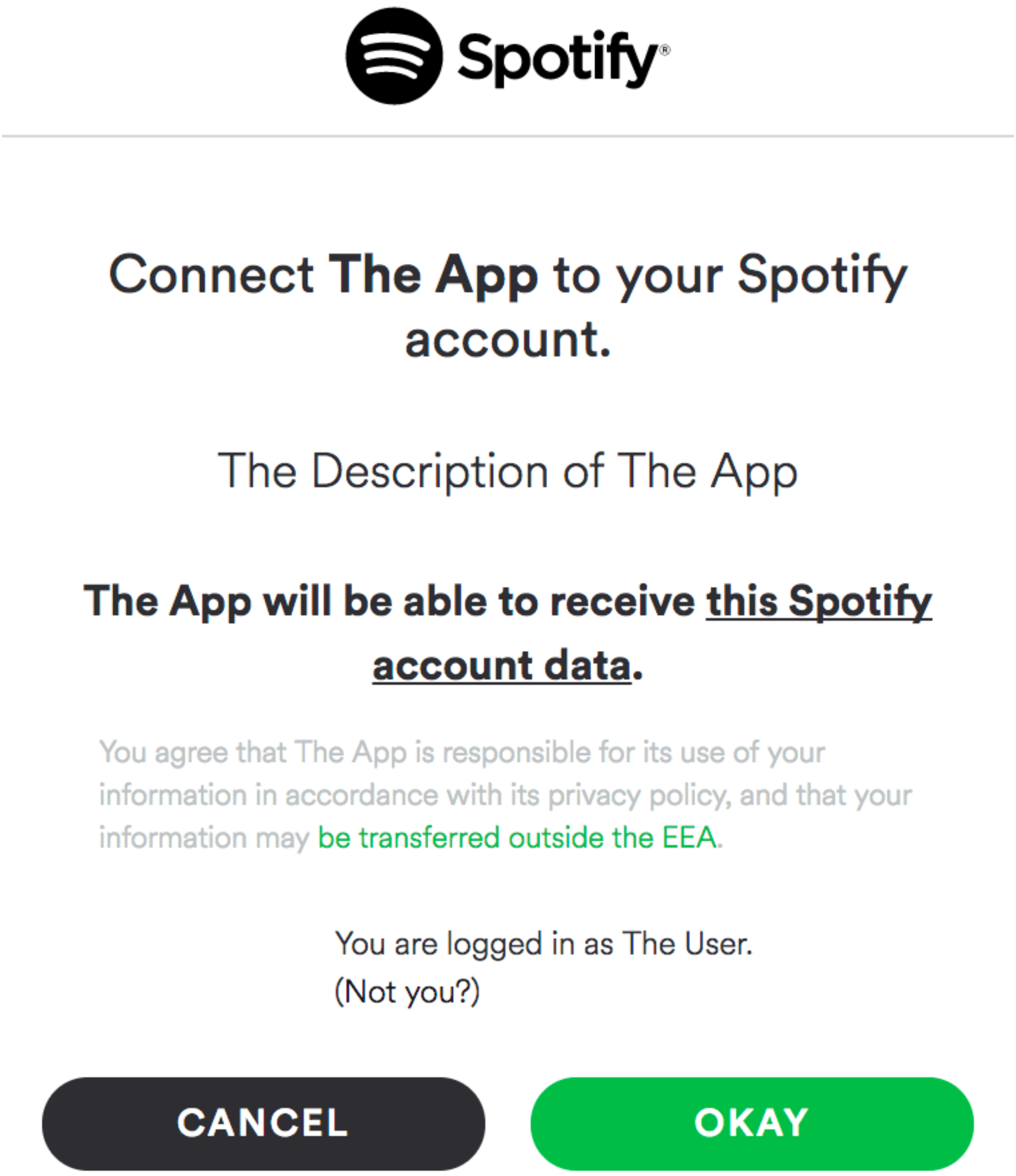
(JSON object)

Return new access token

access_token

Requesting authorization

- Make a page with links to Spotify's authorization endpoint (https://accounts.spotify.com/authorize/)
- Pass arguments in the query string
  - Client ID (public ID of your app)
  - Response type (string "code")
  - Redirect URI (where to return to)
  - Scope (what permissions to ask for)



Spotify

Connect **The App** to your Spotify account.

The Description of The App

The App will be able to receive this Spotify account data.

You agree that The App is responsible for its use of your information in accordance with its privacy policy, and that your information may be transferred outside the EEA.

You are logged in as The User.
(Not you?)

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Requesting authorization

- `https://accounts.spotify.com/authorize?` Endpoint

  `response_type=code&` “code” response type

  `client_id=6d81b7a55e894abdbf53143fb2901573&` Client id for app

  `scope=user-read-private%20user-read-email&` Scope

  `redirect_uri=http%3A%2F%2Flocalhost%3A8888%2Fcallback`

  URI to redirect to:
  http://localhost:8888/callback

- Escaping characters: `encodeURIComponent()`

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/encodeURIComponent

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Requesting authorization

- Import `fetch` library: `var fetch = require(‘node-fetch');`
- Could also use the `http.get`, etc., but we’ve used `fetch` before in A2
- `fetch(url, options)`
- `options`: dictionary of options
  - `method`: `GET`, `POST`…
  - `body`: data…
  - `headers`: Content-Type…

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch#Supplying_request_options

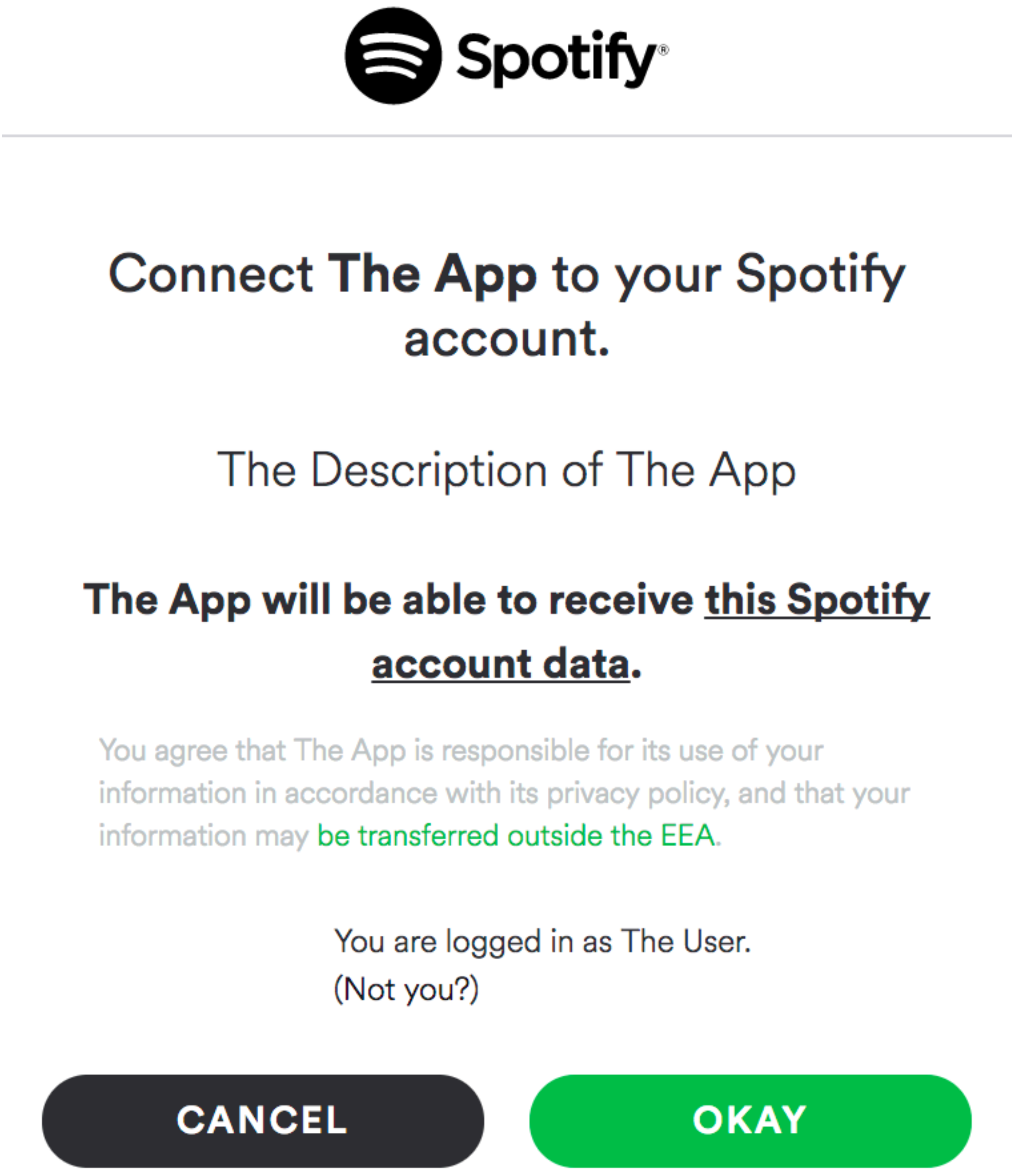
Handling response

- User clicks “okay”, browser then redirects back to your server

- The response contains additional parameters in the URL

- `http://localhost:8888/callback?code=...`

- In Express, `code` can be accessed through `req.query`



Connect **The App** to your Spotify account.

The Description of The App

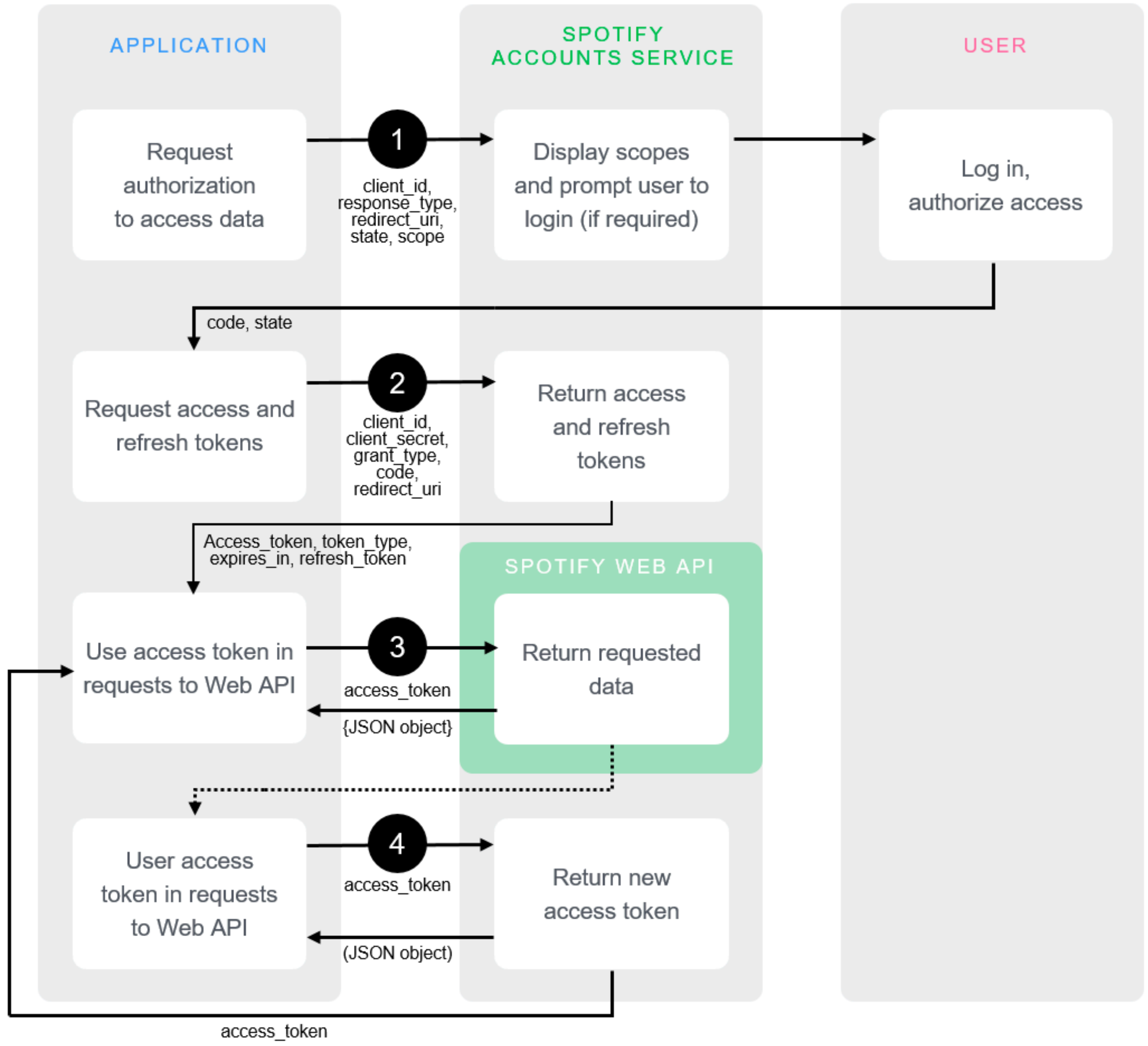
The App will be able to receive this Spotify account data.

You agree that The App is responsible for its use of your information in accordance with its privacy policy, and that your information may be transferred outside the EEA.

You are logged in as The User.
(Not you?)

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Step 2: request access and refresh tokens



APPLICATION

SPOTIFY ACCOUNTS SERVICE

USER

Request authorization to access data

1

client_id, response_type, redirect_uri, state, scope

Display scopes and prompt user to login (if required)

Log in, authorize access

code, state

Request access and refresh tokens

2

client_id, client_secret, grant_type, code, redirect_uri

Return access and refresh tokens

Access_token, token_type, expires_in, refresh_token

SPOTIFY WEB API

Use access token in requests to Web API

3

access_token

{JSON object}

Return requested data

User access token in requests to Web API

4

access_token

(JSON object)

Return new access token

access_token

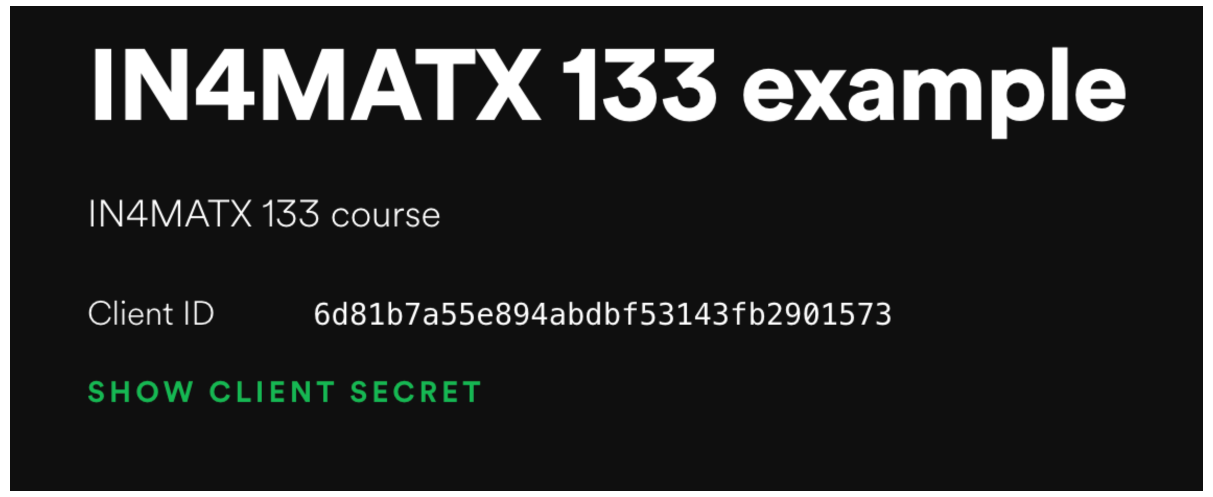
Requesting an access token

- Our goal: trade `code` for an access token
  - An access token needs to be included in API requests
- Why do we need to do this?
  - The user has granted permission for the ID we created on Spotify to access resources
  - But any website could send a user to that URL: client IDs, etc. is all public information
  - How can we verify our app uses the client ID we created on Spotify?

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Requesting an access token

- We make a `POST` request with our client's secret code and ask for an access token
  - Endpoint: https://accounts.spotify.com/api/token



IN4MATX 133 example

IN4MATX 133 course

Client ID 6d81b7a55e894abdbf53143fb2901573

- Why a `POST` request rather than a `GET`?
  - `POST` sends content in the body of an HTTP request (cannot be read by someone watching your web traffic)
  - `GET` sends content in the URI
    - `https://accounts.spotify.com/authorize?response_type=code&client_id=6d81b7a55e894abdbf53143fb2901573`

https://security.stackexchange.com/questions/33837/get-vs-post-which-is-more-secure

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Requesting an access token

- Body of `POST` request requires 3 parameters
  - Grant type (string “authorization_code”)
  - Code (returned as a parameter in the response from the authorization request)
  - Redirect URI (must be the same as before)
- Header of `POST` request requires 2 parameters
  - Authorization (concatenation of client ID and client secret, as a Buffer)
  - Encoding (via Content-Type, as “*application/x-www-form-urlencoded*”)

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Requesting an access token

- Making the body: URLSearchParams
  - `params = new URLSearchParams();`
  - `params.append(‘grant_type’, ‘authorization_code’);` etc.
- Header: a dictionary
  - `‘Content-Type':'application/x-www-form-urlencoded'`
  - `'Authorization': 'Basic ' + Buffer.from(my_client_id + ':' + my_client_secret).toString('base64')`

https://www.w3schools.com/nodejs/met_buffer_from.asp

https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams

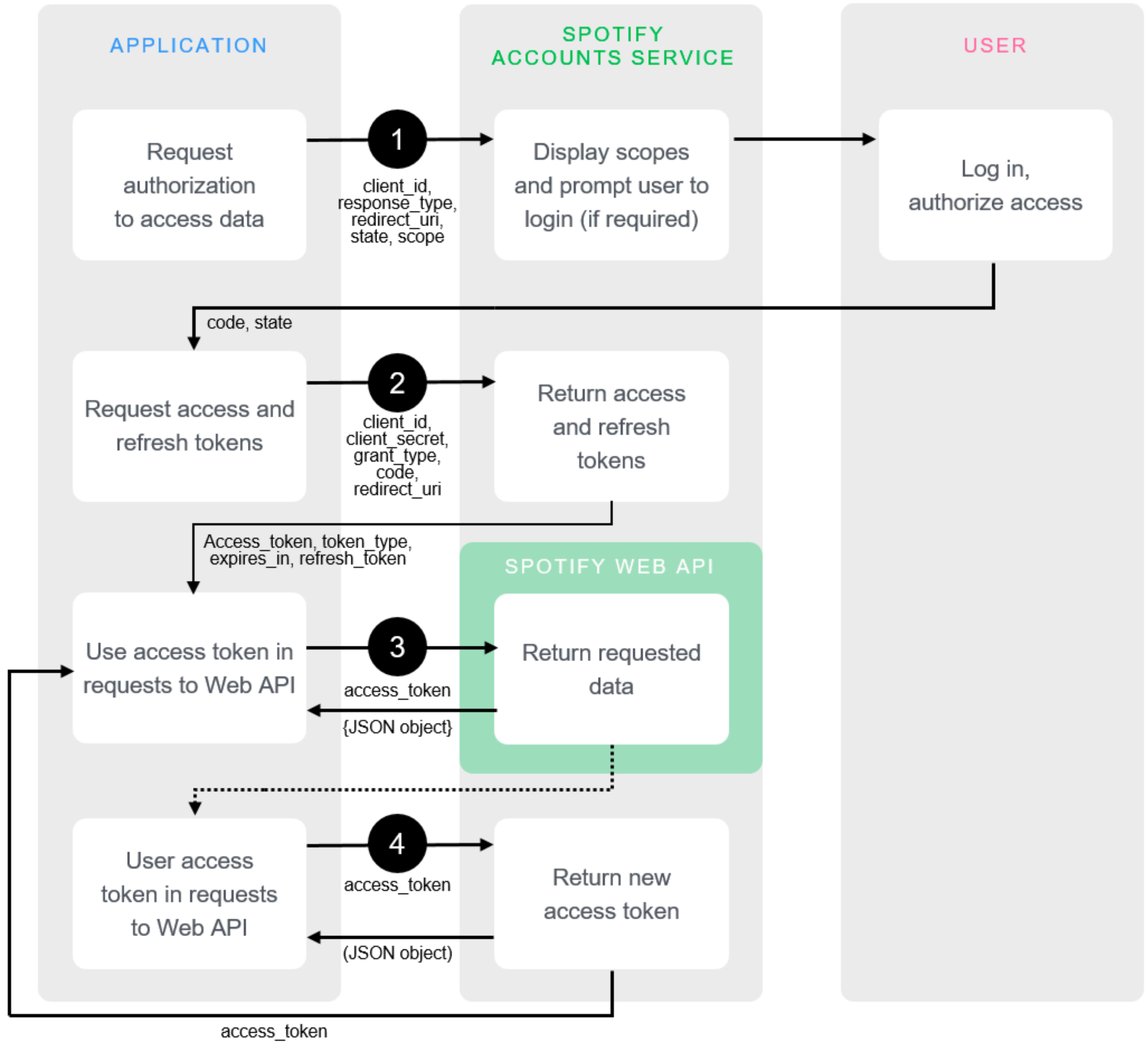
https://developer.spotify.com/documentation/general/guides/authorization-guide/

Handling response

- In the response body, Spotify sends back:
  - Access Token (needed to make API calls)
  - Expires in (how long the access token is good for)
  - Refresh Token (once the Access Token expires, this can be used to get a new one)
- What would you do with these tokens?
  - Store them in a database for later access
  - In A3, we'll store them in a text file (bad form, but easier)

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Step 3: use access token in requests to web API


APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token

Making an API request

- Pass the access token in the header
    - Much like the client id and secret, but no need to convert it
    - `'Authorization': 'Bearer ‘ + access_token`
- Make a `GET` request to one of the API endpoints
    - e.g., https://api.spotify.com/v1/me
    - Will return a JSON object with the requested resource
        - e.g., birthdate, email, a profile image

https://developer.spotify.com/documentation/web-api/reference/users-profile/get-current-users-profile/

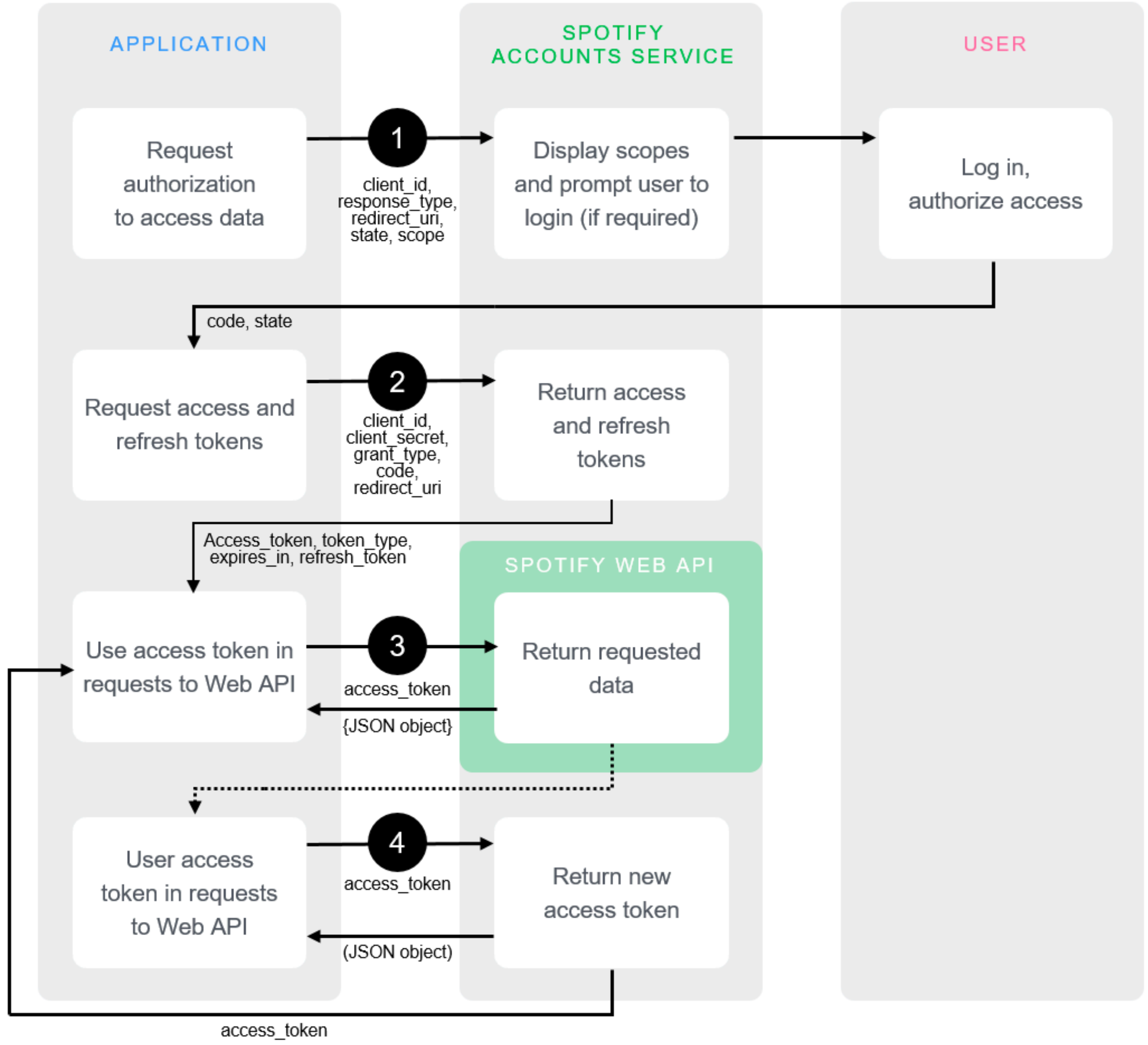
https://developer.spotify.com/documentation/general/guides/authorization-guide/

Making an API request

- Spotify has endpoints for artists, albums, tracks, and more
- Often specify a subresource in the URI
  - e.g., https://api.spotify.com/v1/albums/{id} for a specific album

https://developer.spotify.com/documentation/web-api/reference/

Step 4: refresh access token


APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token

Refresh token

- Tokens typically expire after a fixed amount of time
  - One hour for Spotify tokens
  - After that time, all API requests will return with code `401` (Unauthorized)
- A user can use the refresh token to get a new token
- Why do tokens expire?
  - To allow a user to revoke their privileges

https://developer.spotify.com/documentation/web-api/

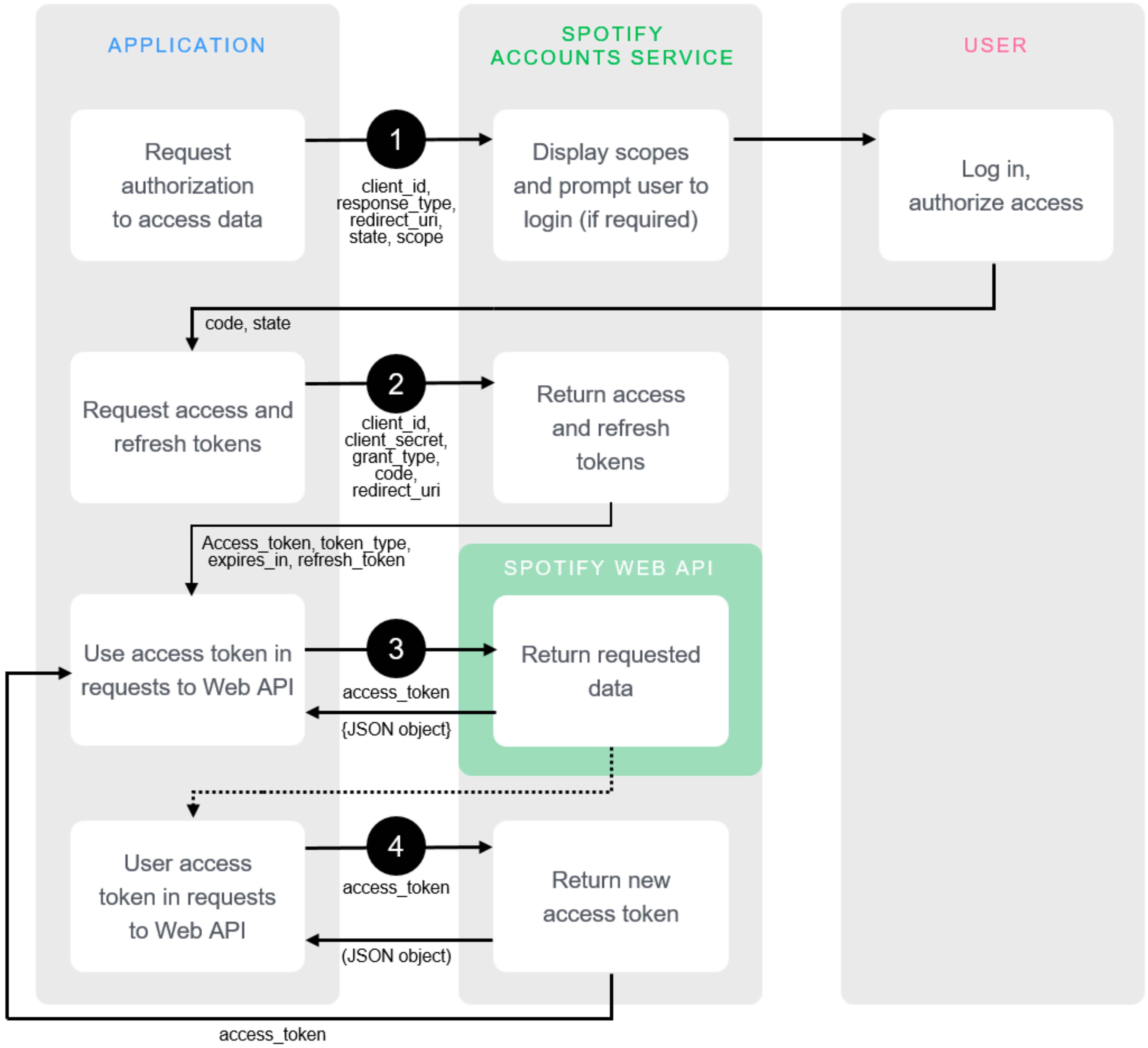
https://developer.spotify.com/documentation/general/guides/authorization-guide/

Refresh token

- Same endpoint as requesting an access token
  - Endpoint: https://accounts.spotify.com/api/token
- Similar parameters; header with encoding and authorization
  - `‘Content-Type':'application/x-www-form-urlencoded'`
  - `'Authorization': 'Basic ' + Buffer.from(my_client_id + ':' + my_client_secret).toString(‘base64')`
- Different body parameters
  - “refresh_token” as “grant_type”, the token itself as “refresh_token”

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Oauth 2.0 steps



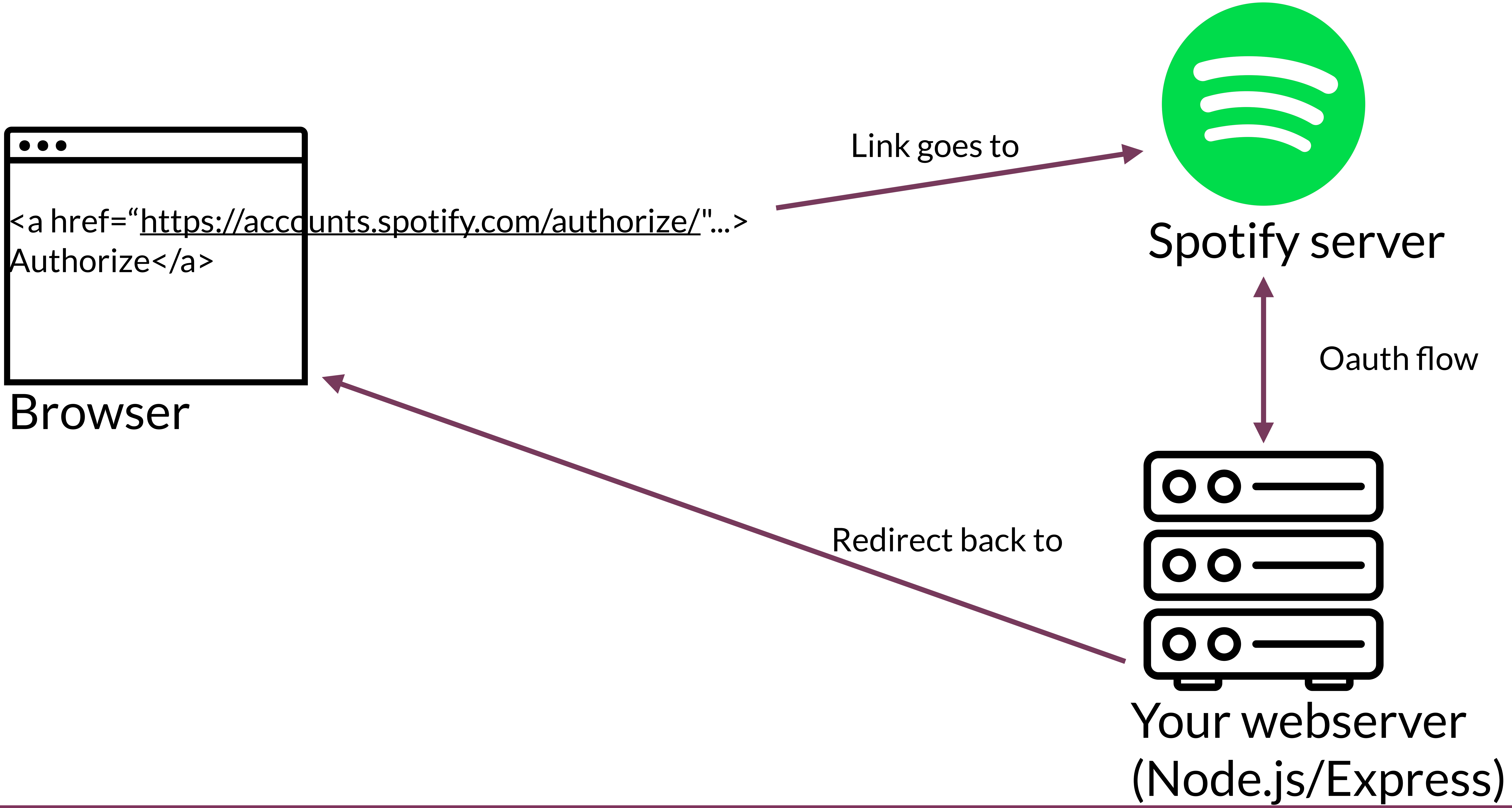
APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token

https://developer.spotify.com/documentation/general/guides/authorization-guide/

Authorizing from the browser

- Create a link to the authorization endpoint (https://accounts.spotify.com/authorize/)
    - Which will redirect to your server-side JavaScript
- Once tokens have been received, redirect back to client-side JavaScript

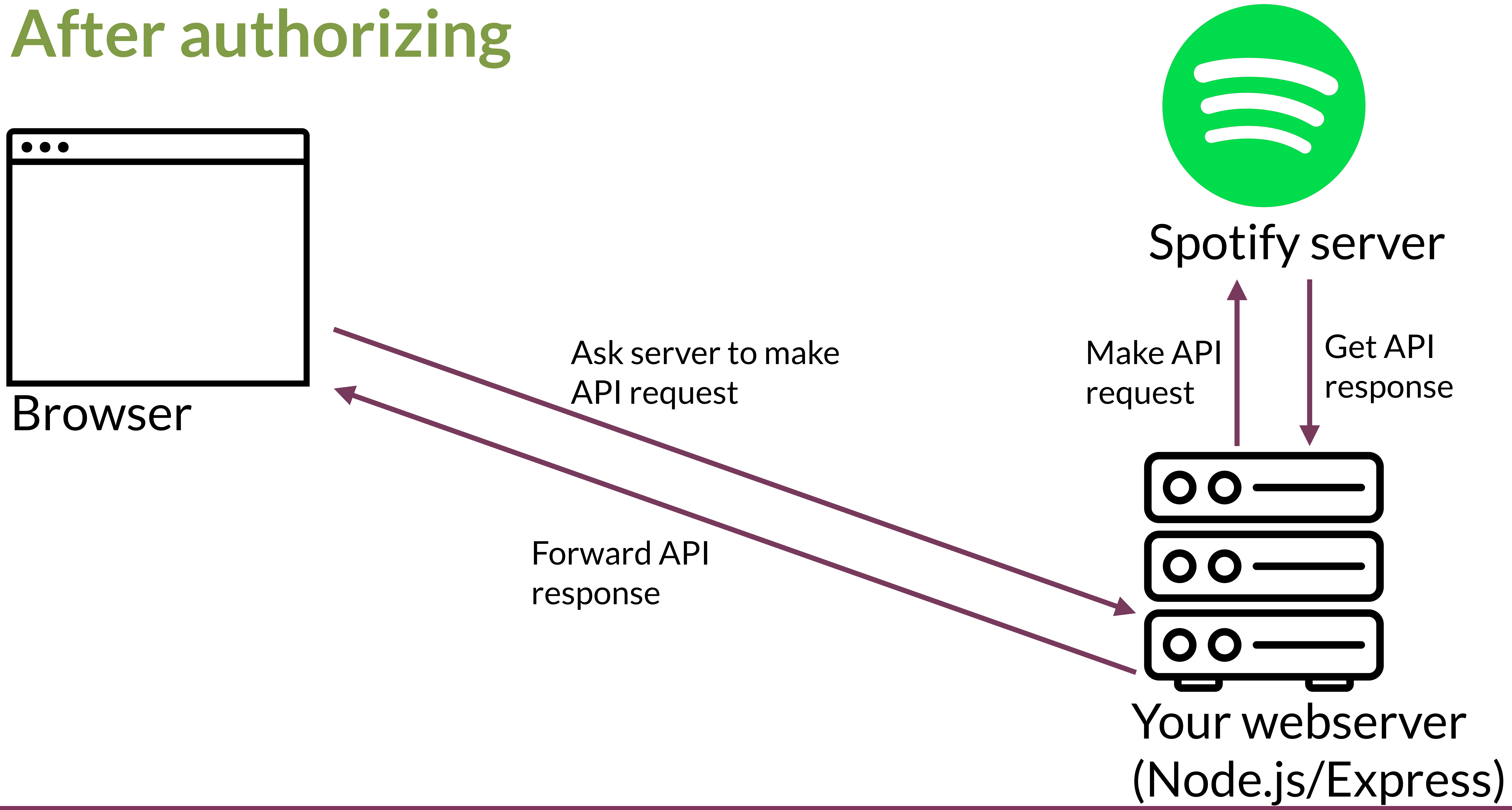
Authorizing from the browser



<a href="“https://accounts.spotify.com/authorize/"...>
Authorize</a>

Making an API request from the browser

After authorizing

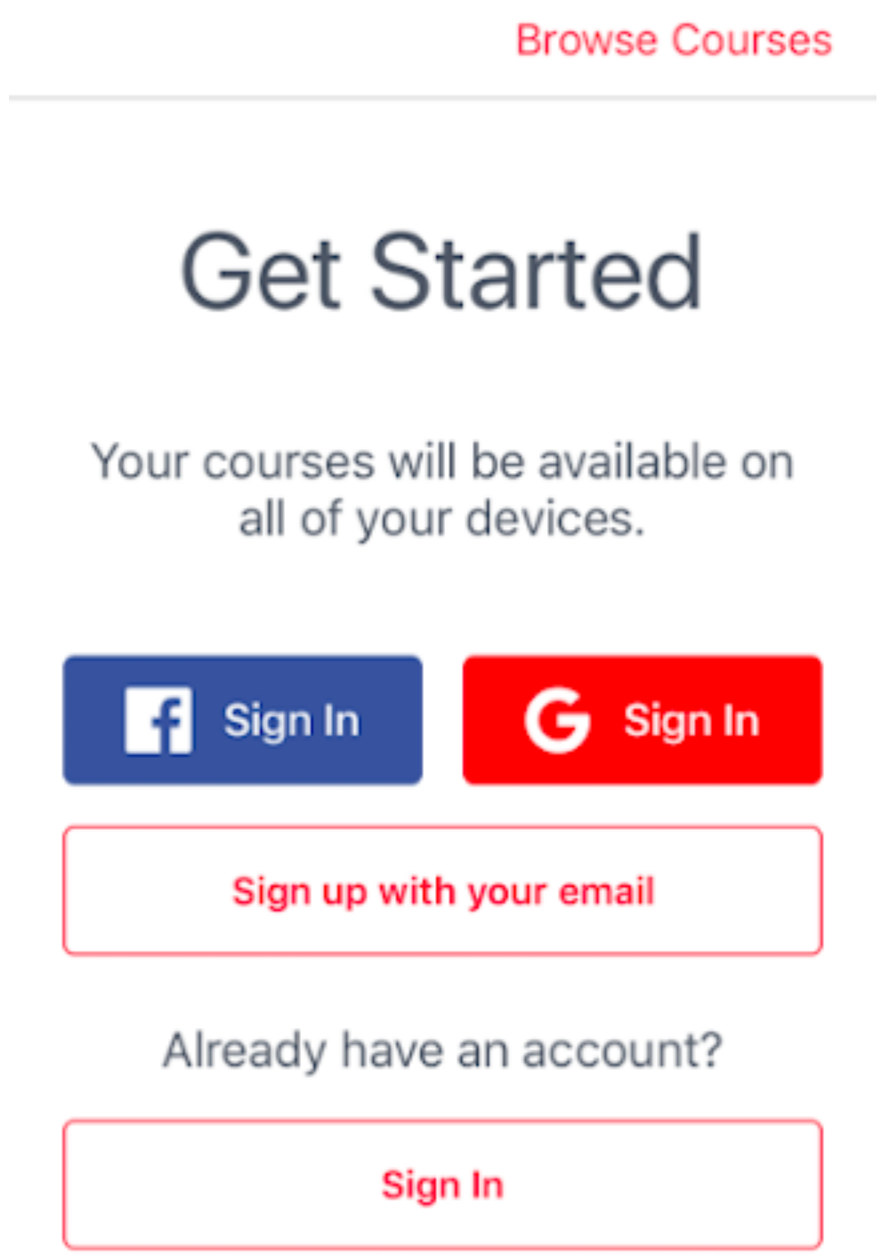


Making an API request from the browser

- How does the browser indicate that it wants the server to make an API request?
  - All web servers communicate in HTTP
  - Make an HTTP request to the server, asking it to make the API request
  - It returns the response

OpenID Connect

- Ever seen a button with “sign in with Google”, etc.?
- Implemented with OpenID Connect
  - Added layer on top of Oauth



Get Started

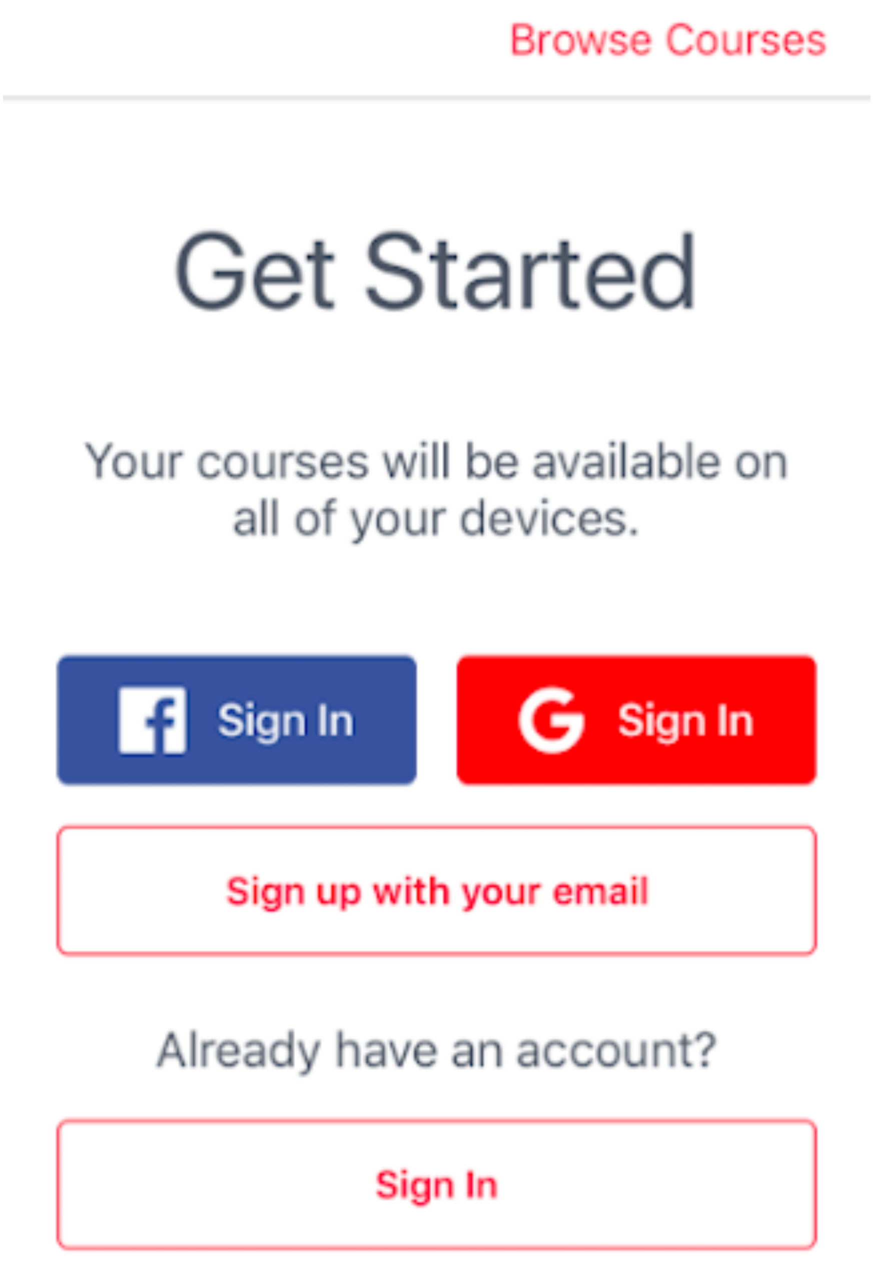
Your courses will be available on all of your devices.

Already have an account?

https://openid.net/connect/

OpenID Connect

- Benefits:
  - No need to get an ID for every service
  - Only one password to remember/store
- Drawbacks
  - Facebook/Google/etc.
    gather (more) information about you
    and and the websites you go to



Get Started

Your courses will be available on all of your devices.

Already have an account?

Today’s goals

By the end of today, you should be able to…

- Differentiate authentication from authorization
- Describe the utility of supporting authentication and authorization in interfaces
- Explain and implement the different stages to authenticating via OAuth
- Describe the advantages and disadvantages of OpenId

IN4MATX 133: User Interface Software

Lecture 12:
Authentication & Authorization

Professor Daniel A. Epstein
TA Jamshir Goorabian
TA Simion Padurean