Social Media

Assignment 1
Concepts
Standard HTTP

- Assignment 0 uses standard HTTP request and response
  - Requests can be GET, POST, etc.
  - `req` represents request and `res` represents response.
  - A client sends request to the server, then the server sends response back to that client.

- One response per request
  - Different clients send their own “request”
  - So, each of these different requests receives their unique response

- Synchronous vs. Asynchronous
  - Assignment 0 uses synchronous HTTP
  - AJAX is standard HTTP with asynchronous component.
  - In short: synchronous requires redirect/refresh; asynchronous does not
Three Pillars of HTTP

GET /routes HTTP 1.1

Host: www.myapp.io
User-Agent: Mozilla/5.0
Accept-Language: en-us
Accept-Charset: utf-8
Content-Type: text/html
...

Request Line

Request Header

Request Body
Standard HTTP Architecture

Client

Request #1

Response #1

Request #2

Response #2

Server

Earlier Time

Later Time
OAuth 2.0

- An authentication protocol provided by third-party software (e.g. Twitter)
- Local vs. OAuth
  - Local authentication - implementing login feature(s) in your own app
  - OAuth - uses third-party to implement login feature(s)
- Why OAuth?
  - Easier to manage user’s profile (e.g. Twitter already has user’s profile)
  - Less time spending on algorithms behind authentication
  - You can add more social media features on top of existing profile information.
- Uses standard HTTP between two servers...
Architecture with OAuth

Client -> Request -> Server -> Request (client secrets) -> Twitter Server

Server -> Response (profile + token) -> Client

Client -> Response
Websocket

- “Full-duplex” communication channel over TCP
  - Standard HTTP uses “half-duplex” communication using over TCP
  - You can think of “half-duplex” as “one response per request” philosophy

- Useful for live updates, messaging, and broadcasting

- Standard HTTP vs. Websocket
  - HTTP server sends a response only to the client sends that corresponding request
  - In websocket, server can send “response” to multiple clients even with one request

- AJAX vs. Websocket
  - AJAX can do “live” updates within a single client (no broadcasting)
  - Websocket can do “live” update to multiple clients (broadcasting)
Architecture with Websocket

Client

Server

Client

Client

Client

Request

Response
Putting Things Together

- We understand standard HTTP, OAuth, and websocket work ...
- How do they connect?
  - Many social media applications use both authentication and liveness of communication
  - Some social media application utilizes other popular application (OAuth)
  - How can you get updates without refreshing in your Facebook? (possibly websocket)
- Suppose you are making a status for your Facebook profile ...
  - How can you use HTTP, OAuth, and websocket to implement this feature?
The Assignment
The Technology Stack

Handles:
- OAuth 2.0
- local authentication

Facilitates websocket programming
In order for OAuth to guarantee privacy, each application needs their own:
  ○ API Key
  ○ API Secret

These are sensitive information only shared between:
  ○ Your application
  ○ Provider (e.g. Twitter)

Add your application to Twitter (https://apps.twitter.com/)

Store API Key and API Secret to .env file.
  ○ We use this to handle configuration for Assignment 1
  ○ Requires “dotenv” node_module. (Hint: how do you install dependencies?)

Create local database to save user’s profile
  ○ Hint: how do you create schema using MongoDB?
app.use(passport.initialize());
app.use(passport.session());

/*
 * (1) Use API Key and Secret stored in .env
 * (2) Check if user’s profile exists in database
 * (3) If not (2), then store profile info to database
 * (4) If (2), then update profile info and save to database
 */

passport.serializeUser(function(user, done) { done(null, user); });
passport.deserializeUser(function(user, done) { done(null, user); });
...
Getting Started (Socket.io)

- **Look into** `public/js/chatbox.js` and `server.js`.
  - `public/js/chatbox.js` will act as client-side.
  - `server.js` will act as server-side.
- **Client and server communicates through** `emit()` **and** `on()` **functions.**
  - `emit(socket_name, function(data) { ... });`
  - `on(socket_name, function(data) { ... });`
  - `emit()` will be used to alert other side that they will be sending the data
  - `on()` will be called when one side has called `emit()`, receiving the data
- **For this assignment, you would need to:**
  - hook Passport logic in `server.js`.
  - manipulate DOM using client-side Javascript (e.g. jQuery) in `public/js/chatbox.js`. 
**emit() and on()**

- In Socket.io, `emit()` will only broadcast to `on()` with matching socket name
  - `emit(name1, ...);` will **only** broadcast to `on(name1, ...);`
  - `emit(name1, ...);` will **not** broadcast to `on(name2, ...);`
- **Special case: reserved name such as "connection" and "disconnect".**
  - These names are used in server-side (`server.js`)
  - `on("connection", ...);` will respond when client-side called `io()` function.
  - `on("disconnect", ...);` will respond when a client exits the application.
- **These are just basic, there are more ...**
Socket.io “Algorithm”

Client-Side (public/js/chatbox.js)  Server-Side (server.js)
Important Note

● Only Part II is graded
  ○ Part I is a “tutorial” for Part II
  ○ Showing proficiency Part II indicates you understood materials for Part I

● Socket.io is optional for Part II
  ○ Socket.io is mentioned in Part I to provide tools for more creativity

● Assignment 1 is heavily graded on your design skills
  ○ For this assignment, 50% of your grade depends on usability
  ○ We will look for how you applied technical skills to design level.

● This is a group assignment!
Recommendation

- Finish Part I before Part II
  - At very least, everyone (individually) should have attempted Part I
  - This gives good foundation of how to get started for Part II
- Read the resources before attempting Part I
  - Some of the resources may have answers to your problem
- Collaborate with your group for Part II
  - Everyone should put equal and fair amount of work for Part II
- Start early!
  - This assignment is huge learning curve compared to Assignment 0 and COGS 120 labs.
Questions?