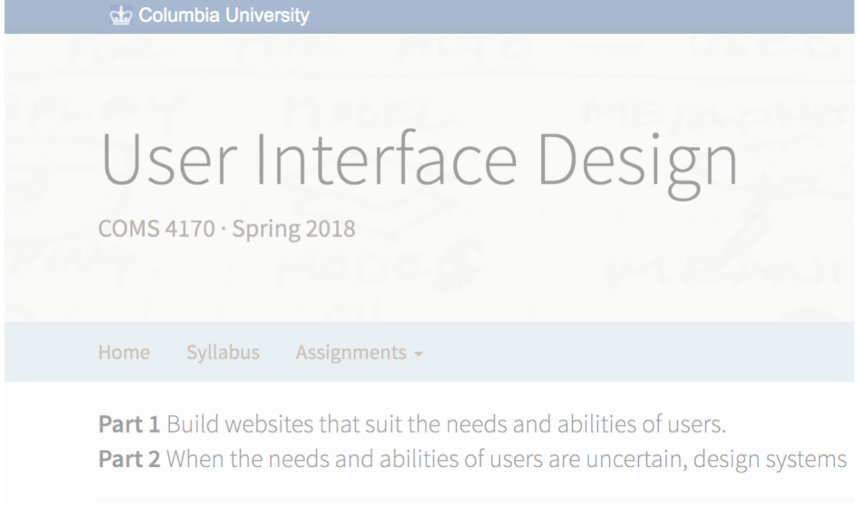
Twitter and Socket.IO

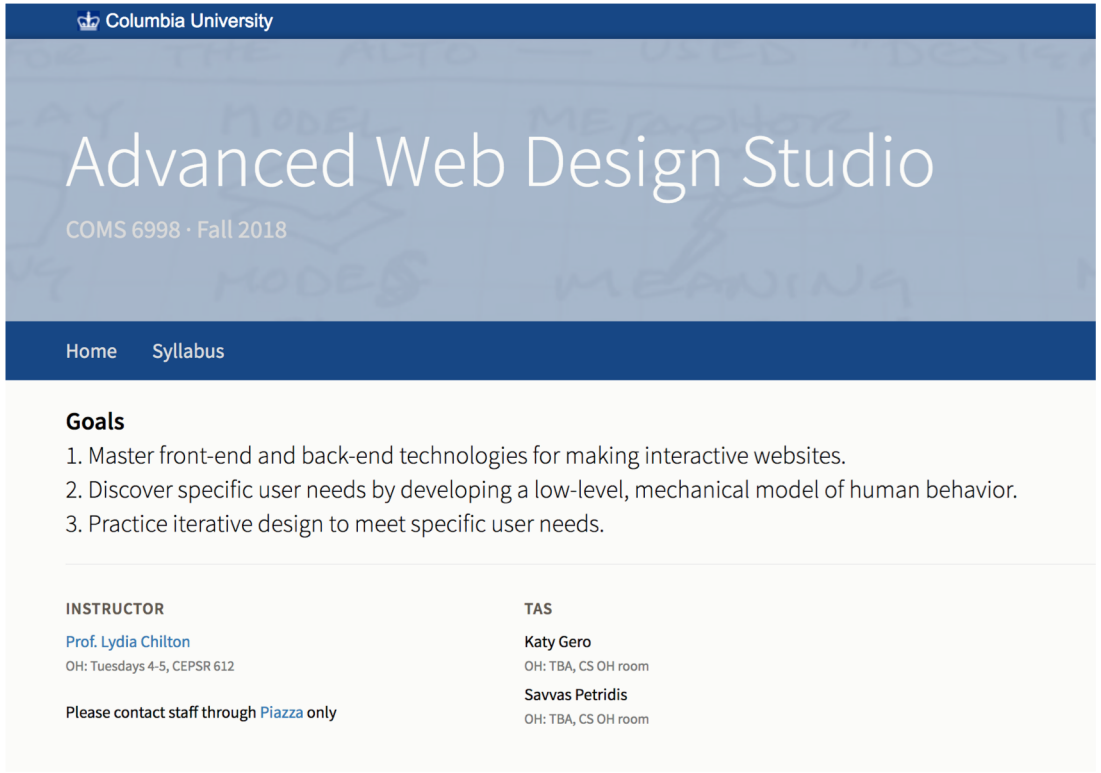
No screens

Prof. Lydia Chilton
COMS 6998
12 October 2018



You already know front-end web dev:
HTML, JavaScript, Bootstrap, jQuery

And design:
Iterative design, critique



You will learn back-end web dev:

- Server-side programming (Flask),
- **Databases (Sqlite, SQLAlchemy)**
- Real-time Communication (Socket.IO)

And practice web design by:

- **Rebuilding IMDB.com**
- Rebuilding twitter
- Pursuing your own project

Last week's goal: Implement **User Accounts**

- Extend your application by adding one feature that requires user accounts.
  - Define your own high level goal
    - Define 7-10 low level goals you will need to achieve this.

  - You must implement **user accounts**
  - **Add a feature** to your site that requires user account.
  - The **graphic design** should reflect the intent of the application to the user.
    - Please iterate on the graphic design based on your feedback today.

Studio: 20 minutes
User testing the new user feature

Partner up!

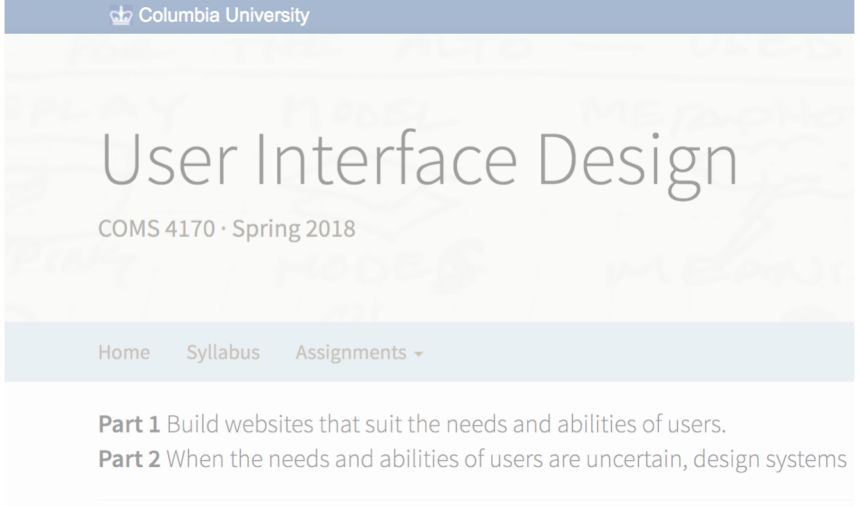
Right hand side partner goes is the **developer** first (the other person is the **user**)

Developer: tell the user who they are and what their goal is BUT NOT WHAT TO DO.
User: if the developer tries to tell you what to do, cover your ears!
User: Try to accomplish the goal. Be honest! Were you able to complete the goal.
Developer: Observe user actions and critical incidents: failures, confusions, and any stress or difficulty the user had.
Developer / User: Make clarifications ONLY as a last resort, when it is the only way to make progress.

Wait for the TAs to come around and observe.
If you finish before we call time, have a conversation on **how you implemented the user feature**.

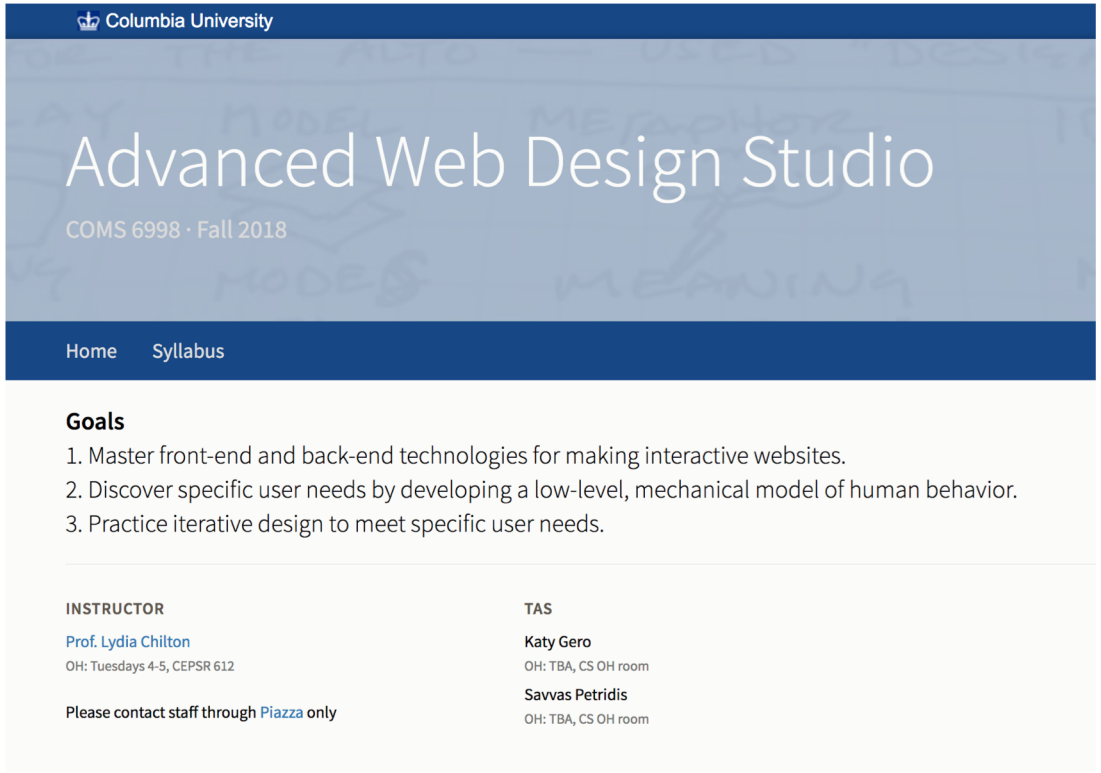
When we call time, switch roles!

Due by 9pm today on Piazza – write one thing you learned from user testing today.



You already know front-end web dev:
HTML, JavaScript, Bootstrap, jQuery

And design:
Iterative design, critique



You will learn back-end web dev:

- Server-side programming (Flask),
- Databases (Sqlite, SQLAlchemy)
- **Real-time Communication (Socket.IO)**

And practice web design by:

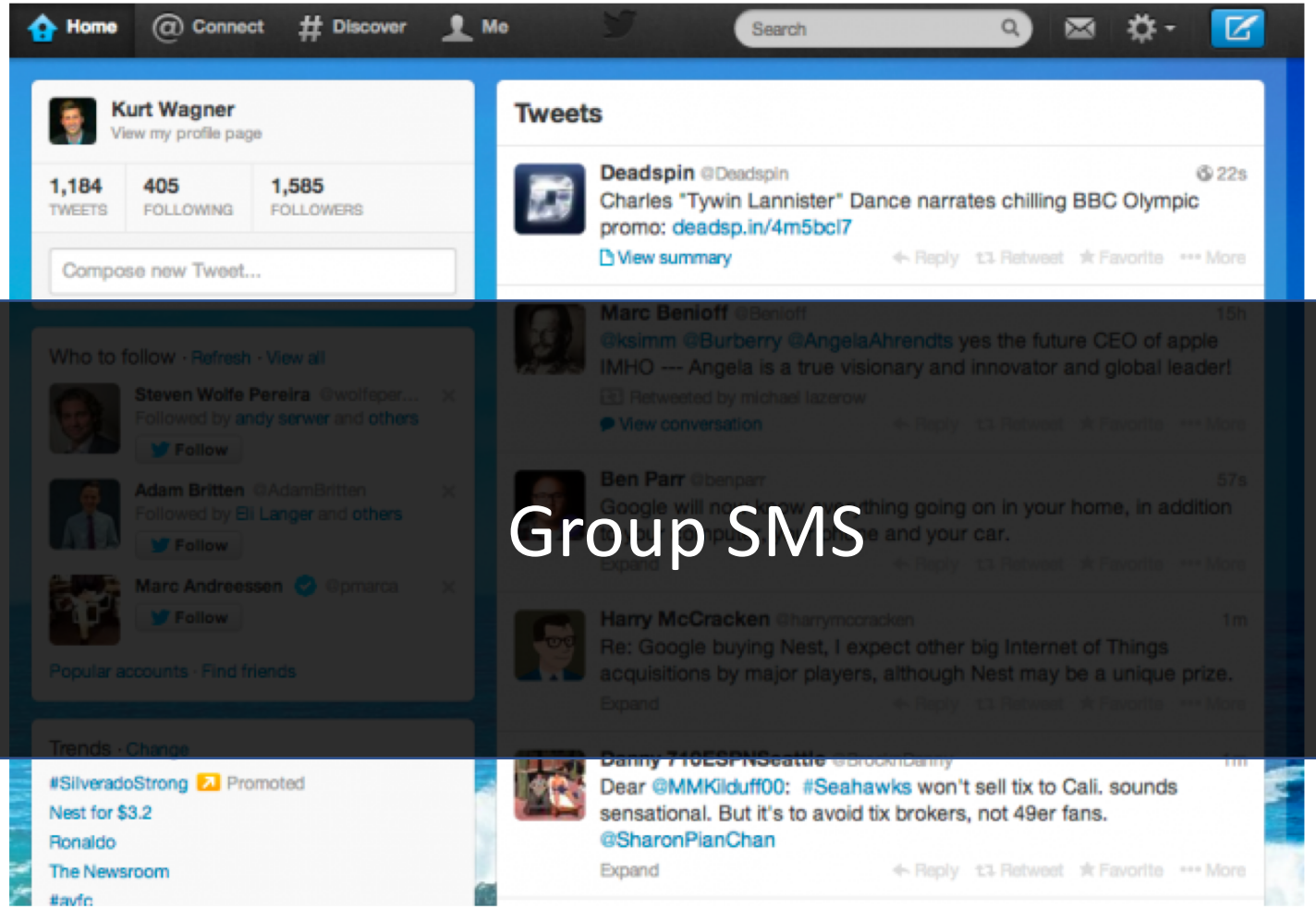
- Rebuilding IMDB.com
- **Rebuilding twitter**
- Pursuing your own project

Rebuilding Twitter

General goals are really **domains**.
By picking a **specific need** in the **domain**,
you can then **generalize** other other thing in that **domain**.

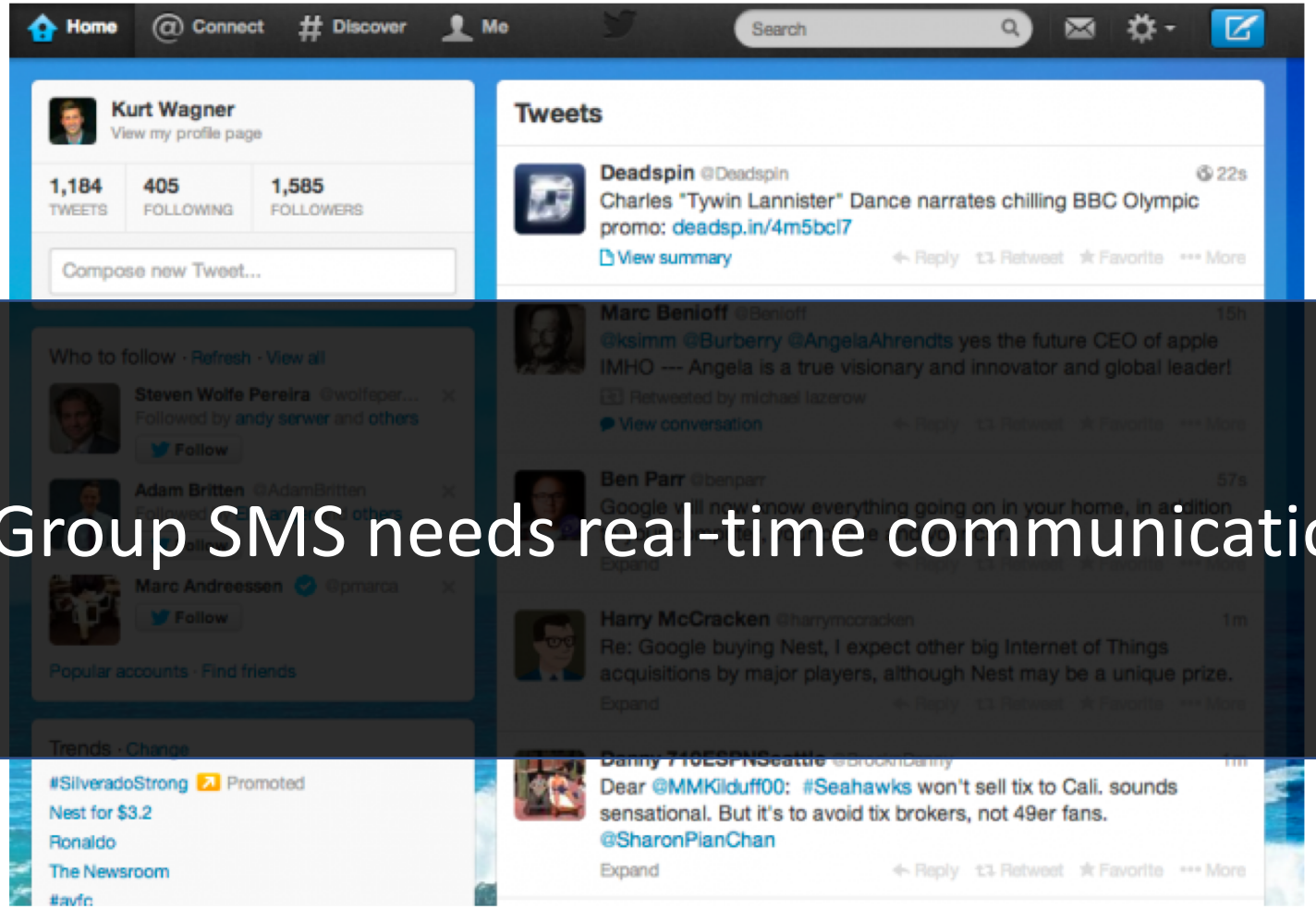
	Domain	Specific Need	Generalized to
amazon	Online shopping	Uncommon books	Clothes, Food, Amazon Fresh Other sellers 
facebook	Social Network	Harvard students to look up each other's classes	Ivy League US Colleges Everybody 
twitter	Social Network		Images, Political movements, Celebrities connecting with fans advertising

What specific user need did Twitter **initially** serve?



Group SMS

We prototype/build the riskiest feature first. What is the **riskiest features** necessary for this?



Group SMS needs real-time communication

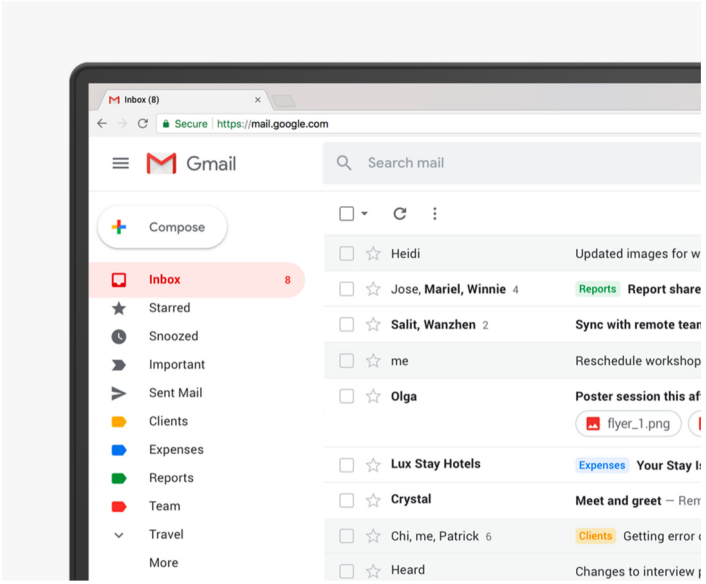
Getting new data:
Pull model vs. **Push** model

For the server to push data,
we need more than HTTP.
We need WebSockets.
SocketIO implements **WebSockets**

How do users get new data
from the **IMDB** server?

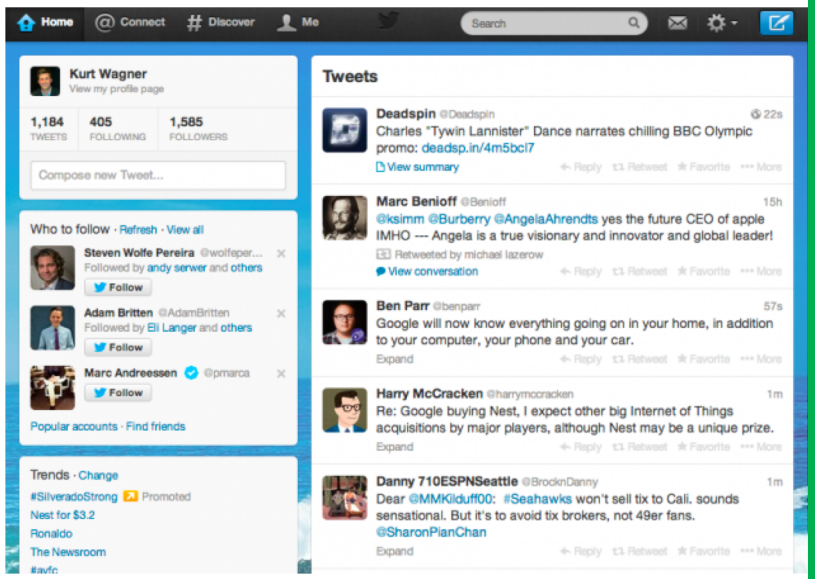
Pull model –
driven by user clicks

How do users get new data
from the **GMail** server?



Pull model –
Driven by a timer on the client side

How do users get new data
from the **Twitter** server?



Push model –
Driven by updates on the server

SocketIO is a framework to send and receive messages

Client side

```
26  $(document).ready(function(){
27
28    var socket = io.connect('http://localhost:5000/')
29    socket.on('connect', function(){
30      console.log("User has connected")
31    })
32
```

```
36  socket.on('message', function(msg){
37    $("#messages").append("<li>"+msg+"</li>")
38  })
39
```

```
40
41  $("#sendButton").on('click', function(){
42    var myMessage = $("#myMessage").val()
43    socket.send(myMessage)
44  })
45
```

Server side

```
 9  from flask_socketio import SocketIO, send
10  socketio = SocketIO(app)
11
```

```
15  @socketio.on('message')
16  def handleMessage(msg):
17      send(msg, broadcast = True)
18
```

For next week

- Implement the **real-time synchronous group chat aspect of Twitter**
  - Must have user accounts
  - Must have a database of history
  - Chats must appear in real-time using Socket.IO
    - must include message and the send's name
  - Needs to have a homepage of all messages
  - Needs to have pages for individual users messages
  - Users must be able to reply to a message (stretch goal)
- Don't need to implement:
  - Hashtags / trending topics
  - Profile pictures
  - search

Due by 9pm today on Piazza – write one thing you learned from user testing today.