

Observation and User Accounts and Socket.IO

No screens



Prof. Lydia Chilton
Adv Web Design Studio
4 October 2019

Say your name

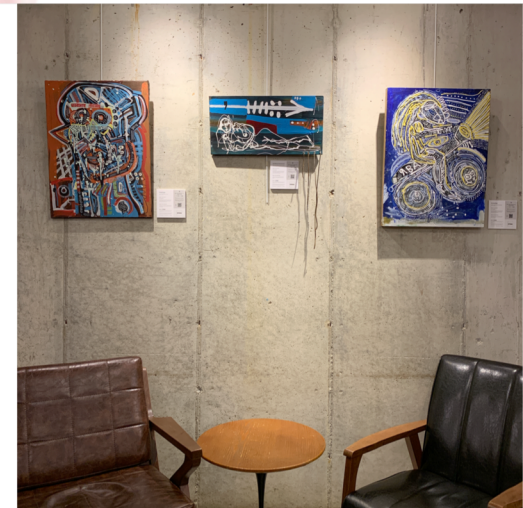
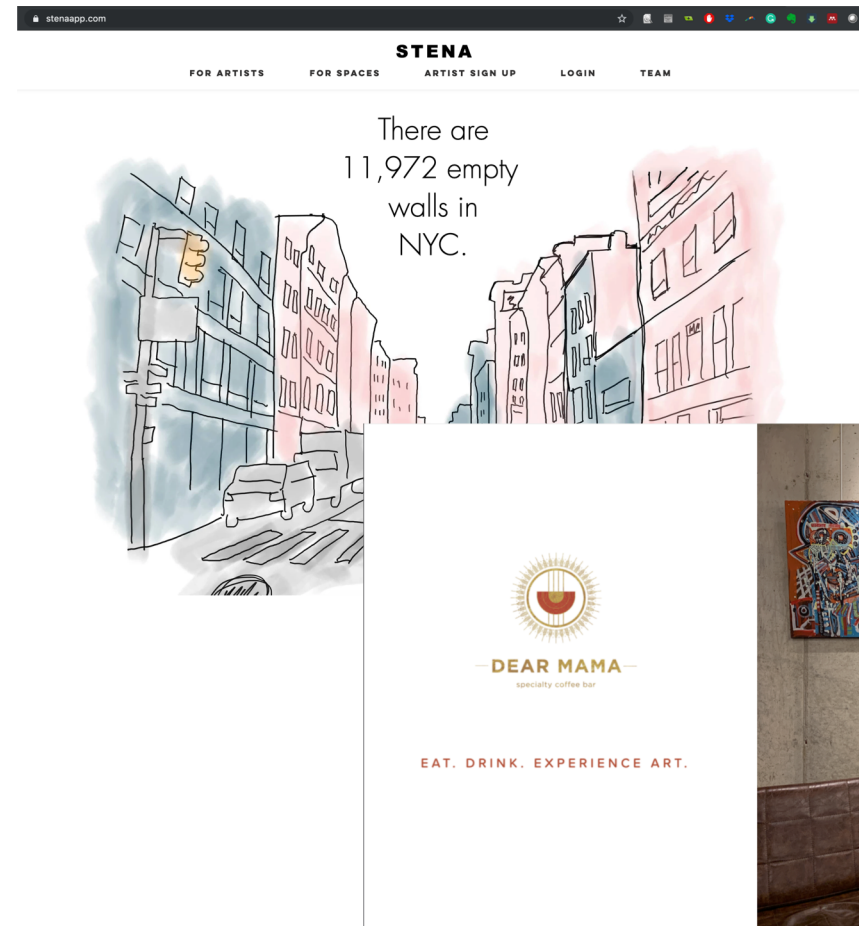


Primary Objective:

Make software
that impacts people

Not a portfolio project...

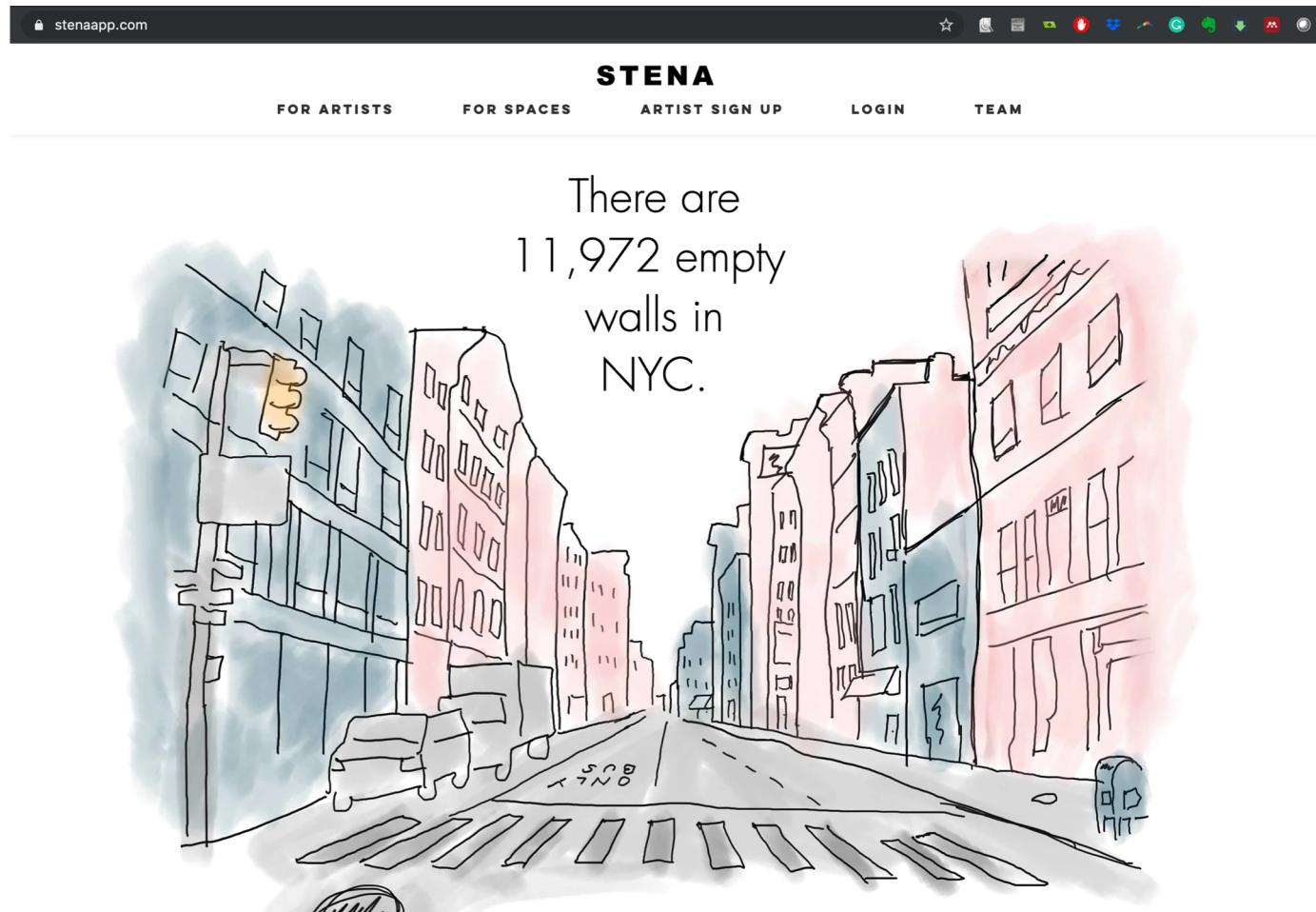
an *impact* project



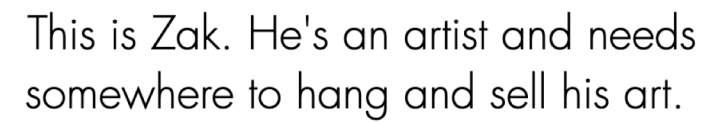
You cannot make impact with technology alone.

You have to meet people,
learn their needs,
and help them achieve their goals.

Example of an impact project from Adv Web Design Studio Fall 2018



TEAM





— **DEAR MAMA** —
specialty coffee bar

EAT. DRINK. EXPERIENCE ART.



You just completed a full first
iteration of an impact project

[Home](#)[Cooks](#)[Eaters](#)[Suggestions](#)

Welcome to Potluck House.



Cooks

Click on Cooks to add your dish to the list for this week's potluck.



Eaters

Click on Eaters if you plan on attending to vote for cooks' dishes.



Suggestions

Click on Suggestions to suggest a dish or if you have dietary restrictions.

Full Iteration:

- Brainstorming user groups on campus
- Technical Prototype
- Graphic Design
- User testing
- Implementation
- Impact
- Pitch

But that was just a warm-up

Now,
we're going to find real user needs
through observation and interviews.

Rather than assuming we know what the problems are,

We are going to look closer at situations before deciding what is needed.

Find a group on campus to observe

- You already brainstormed 10 groups.
- Pick one that might have interesting technology needs.
- Hopefully, you already have insight into this group.
- Observe them (two sessions) doing their work.
- And be able to list 5 examples of:
 - **activities**—goal-directed actions, activities, and processes
 - **environments**—personal or shared workspaces or common areas
 - **interactions**—between people and objects
 - **objects**—things people have in their environment and use in their activities
 - **users**—the people you're observing

Let's do it now for this class.

For each category, what are 5 things we do that involve technology:

- **activities**—goal-directed actions, activities, and processes
- **environments**—personal or shared workspaces or common areas
- **interactions**—between people and objects
- **objects**—things people have in their environment and use in their activities
- **users**—the people you're observing

Does anything stick out as a critical incident?

4 Typical problem technology solves

- **Marketplaces**

- OkCupid
- ?

- **Communication**

- Slack
- ?

- **Databases**

- IMDB
- ?

- **Workflows**

- Turbo Tax
- ?

What might be some of the technology needs for Advanced Web Design Studio?

Assignment Part 1: Do Observation in Pairs


- Read article about observation
- Find a partner.
 - You will both identify a group to observe
 - And both go observe the group and write up your observations
- Read the article on Observation
- Do this process for both observations
 - Work as a team, write it up separately.
- Take pictures
- Be prepared to present interesting findings (in PPT) in small groups.

Assignment Part 2: A Chat App

- Technical Prototype of a chat application
- Must have user login with WTForms and Flask_login
- Must have real time message broadcasting with Socket.IO
- There is example code.
- Get the code to run locally
- You have to make up some random use case for it.
 - Add ONE feature for this use case.
 - Put a tiny bit of UI work into it.
- Submit a video of you using it in two browsers side-by-side

User Accounts

Be aware of "phishing" emails. CUIT will never ask for your password or private personal information via email. Use the main Columbia home page to navigate to password services. Click [here](#) for more information.

 **MICE**

UNI

lc3251

PASSWORD

.....

Contact CRF

LOGIN

clear

By using these resources, you agree to abide by Columbia University's [Acceptable Usage of Information Resources Policy](#).


 **Hacker News** new | comments | show

Login

username: Swizec

password:

login



Username

Password

☒ Remember me

[Forgot Password?](#)

LOGIN

facebook [Sign Up](#)

Log into Facebook

aspdemo@outlook.com

Password

Log In

or


Create New Account

[Forgot account?](#)

[Not now](#)

One account. All of Google.

Sign in to continue to Gmail



example.

Next

[Need help?](#)

[Create account](#)

Basic elements of Login

Front end:

User actions needed

Register
New Users

Bad: User already exists

Good: created. Now what?

Login

Correct

Incorrect

Logout

Where to send the user next

Using app

Know they are signed in

Back end:

Stuff needed to support user actions

Database of users

Password hashing

What to implement:

Front end: User interface

Back end: Database interaction

Register
New Users

A form

Validation

feedback

Login

A form

Validation

feedback

Logout

A button

nothing

Reminder you are logged in

nothing

Using app



LIONMAIL
@COLUMBIA



What to implement:

Front end: User interface

Back end: Database interaction

Register
New Users

A form

feedback

Validation

Login

A form

feedback

Validation

WTForms

Logout

A button

nothing

Using app

Reminder you are logged in

nothing



LIONMAIL
@COLUMBIA



Flask_login

Flask_login:

log users in, log users out, know who they are

Add flask_login:

```
from flask_login import LoginManager
```

```
login = LoginManager(app)
```

```
@login.user_loader  
def load_user(id):  
    return User.query.get(int(id))
```

Use flask_login:

```
user = Users.query.get(id)
```

```
login_user(user)
```

```
<div>  
    Hi, {{ current_user.username }}!  
</div>
```

WTForms

Define forms and their validation

Back-end:

Define Form class

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField,
from wtforms.validators import ValidationError, DataRequired,
from app.models import User
```

```
class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember_me = BooleanField('Remember Me')
    submit = SubmitField('Sign In')
```

Front end:

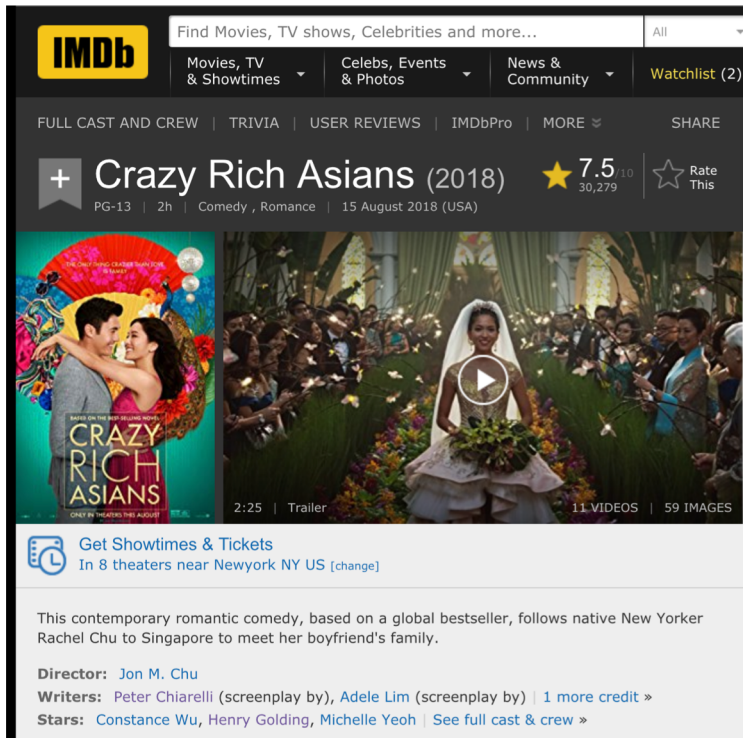
render form

```
<h1>Sign In</h1>
<form action="" method="post" novalidate>
    {{ form.hidden_tag() }}
    <p>
        {{ form.username.label }}<br>
        {{ form.username(size=32) }}
    </p>
    <p>
        {{ form.password.label }}<br>
        {{ form.password(size=32) }}
    </p>
    <p>{{ form.remember_me() }} </p>
    <p>{{ form.submit() }}</p>
</form>
```

Socket.IO

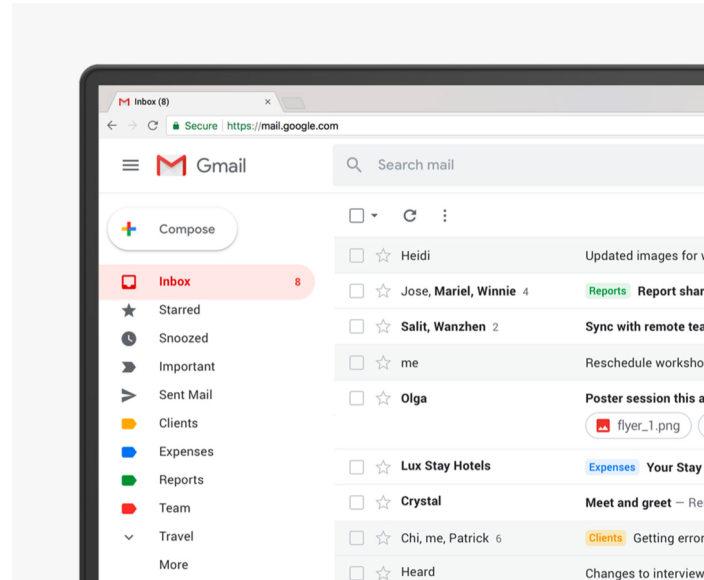
Getting new data: Pull model vs. Push model

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



Pull model –
driven by user clicks

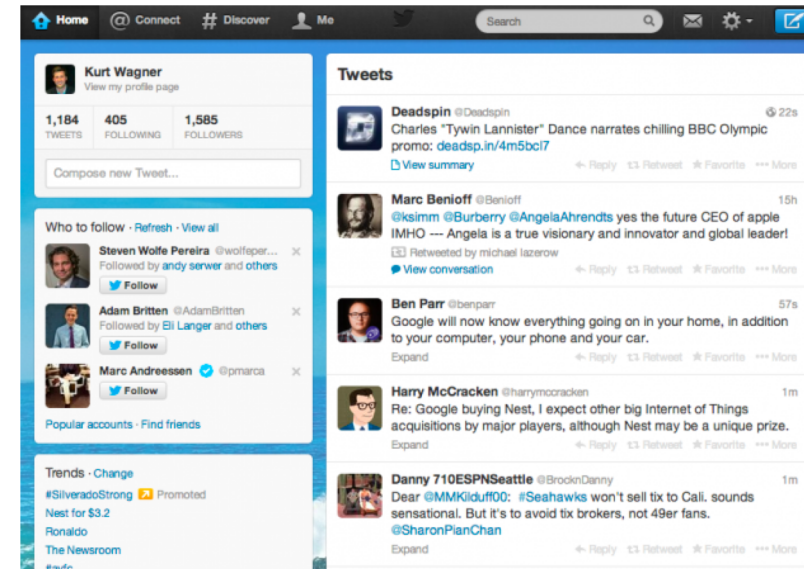
How do users get new data
from the **GMail** server?



Pull model –
Driven by a timer on the client side

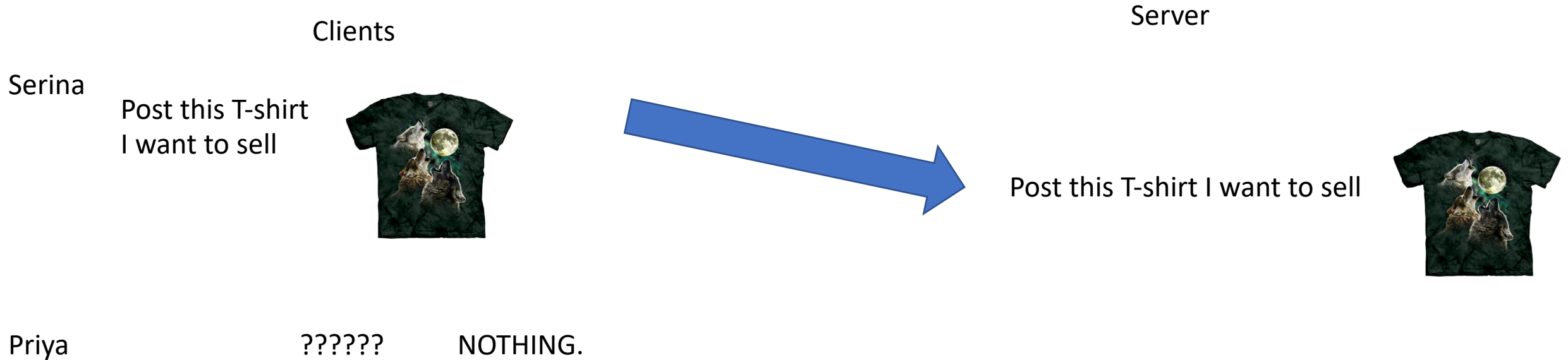
For the server to push data,
we need more than HTTP.
We need WebSockets.
SocketIO implements **WebSockets**

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



Push model –
Driven by updates on the server

Client **Pull** Info from the server (by asking)



Client Pull Info from the server (by asking)

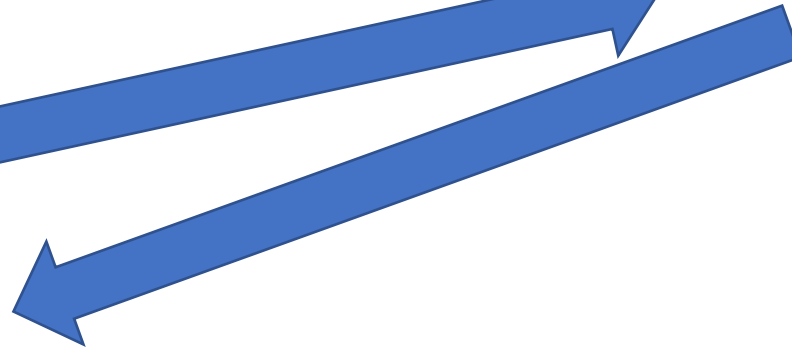
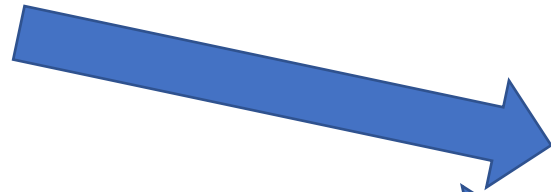
Clients

Server

Serina: Post this T-shirt
I want to sell



Post this T-shirt I want to sell



Grant: Any new t-shirts?
(refresh the page)

“BUY IT!!!”



Can we simplify this?

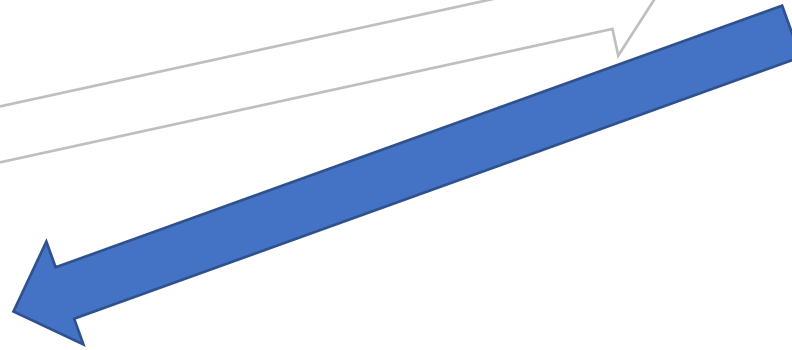
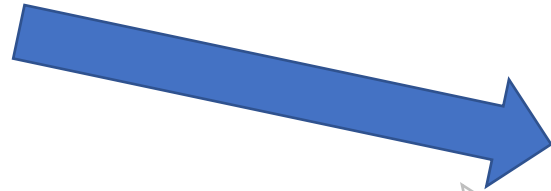
Clients

Server

Serina: Post this T-shirt
I want to sell



Post this T-shirt I want to sell



Grant: Any new t-shirts?
(refresh the page)

“BUY IT!!!”



Can we simplify this?

Clients

Server

Serina: Post this T-shirt
I want to sell



Julia:
"BUY IT!!!"



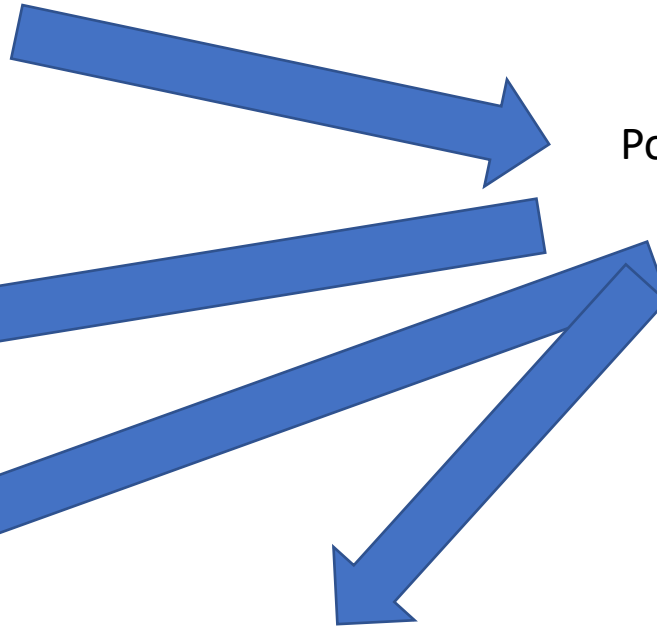
Grant: "BUY IT!!!"



Chris:
"BUY IT!!!"



Post this T-shirt I want to sell



Socket.IO uses **pushes** new info to users

Clients

Server



SocketIO is a framework to send and receive messages

Clients

Server

Serina
Grant
Chris:

```
25
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
```

```
8
9 from flask_socketio import SocketIO, send
10 socketio = SocketIO(app)
11
```

Serina:

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

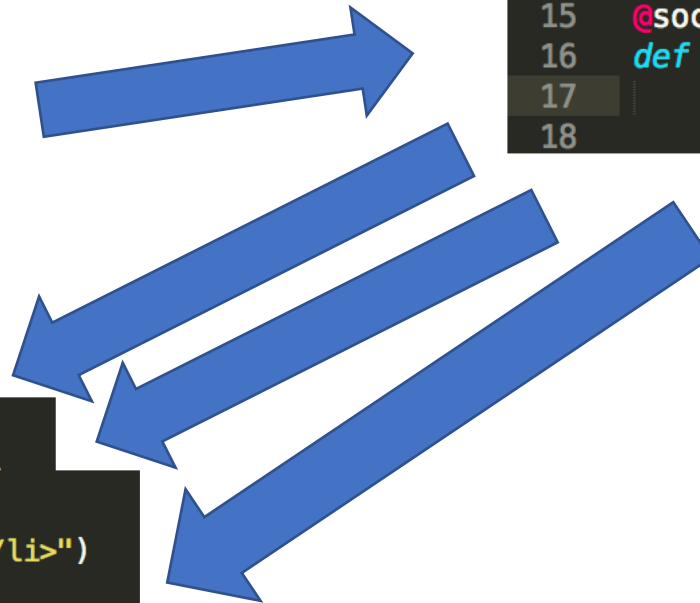
```
14
15 @socketio.on('message')
16 def handleMessage(msg):
17   send(msg, broadcast = True)
18
```

Grant:

```
35
36 socket.on('message', function(msg){
37
38   socket.on('message', function(msg){
39     console.log(msg)
40   })
41
42   socket.on('message', function(msg){
43     $("#messages").append("<li>" + msg + "</li>")
44   })
45
```

Chris:

Serina:



Step 0.

- Download the example code with User Accounts and Socket.IO
 - Get it to run on your machine ASAP.
 - You will have to pip install libraries, and do some config stuff.
 - No Digital Ocean (yet!)

If you feel like your brain is split in half...

That's okay!

Engineering! Yay!



Humanity! Double yay!

RIGHT NOW

- Identify your teammate, and talk about what groups you each want to observe
- Reply to the piazza post called “My Observation group is”
 - Your names
 - What group each of you would like to observe this week.
- Only one of you needs to reply to the post. (this is how we record participation for today)