Social Computing Capstone

Day 11: Trust & Safety

CSE 481p | Winter 2024

Amy X. Zhang
Instructor
Assistant Professor | University of Washington, Allen School of Computer Science & Engineering

Ruotong Wang
TA
Ph.D. Student | University of Washington, Allen School of Computer Science & Engineering

Schedule for today’s class

- Trust & Safety (30 min)
- G4 (Code and Design Spec) walk-through (15 min)
- Group work time (45 min)

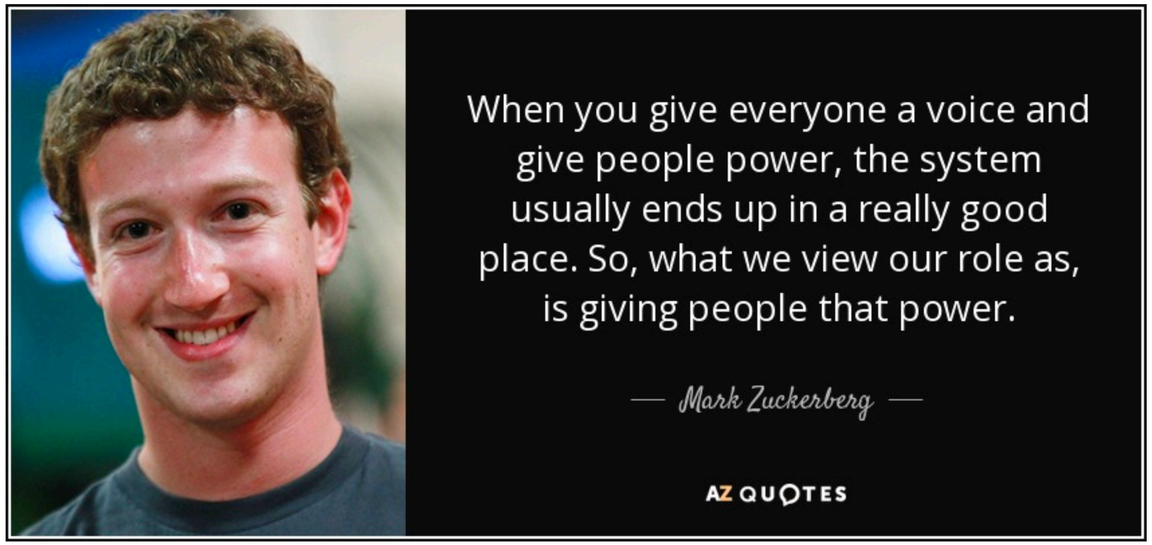
Trust & Safety
(or alternatively, content moderation at platform scale)

Content moderation exists *everywhere* we have content

We have always had content moderation. Back when we consumed content via newspapers, TV, and radio, content moderation meant having editors and regulators overseeing journalists and producers. There were fewer avenues for media consumption, and “gatekeepers” were fewer and had more power.

Today, the nature of content moderation has changed drastically alongside the explosion of social media.

~ Flashback to the early-mid 2000's ~



"The fact that we're all connected…does change the parameters…But I think, in general, it's clear that most bad things come from misunderstanding, and communication is generally the way to resolve misunderstandings—and the Web's a form of communications—so it generally should be good…"

–Tim Berners Lee, inventor of the World Wide Web, 2006, interview

Social media today

How would you address some of the many problems we've discussed in class (harassment, misinformation, incitement to violence, threats, gore, bullying, etc….)?

Mark's answer: a large team of humans for now, and the hope of AI to eventually replace them

Why humans and why not AI today? Content moderation is really tricky!

- All nudity is banned.
- Actually, nude paintings and sculptures are ok.
- Actually, nude pictures of historical significance are ok.
- Actually, pictures showing genitalia in the context of birth or health-related photos are ok.
- Actually, female breasts are ok if they're breast-feeding.
- …Also if they are protesting.
- …Also if they are showing a post-mastectomy scar.

Fury over Facebook 'Napalm girl' censorship

By Zoe Kleinman
Technology reporter, BBC News

9 September 2016

Vietnam War

Support The Guardian
Available for everyone, funded by readers
Search jobs
Sign in
Contribute →
Subscribe →

News | Opinion | Sport | Culture | Lifestyle

US Elections 2020 World Environment Soccer US Politics Business Tech Science Nev

Facebook

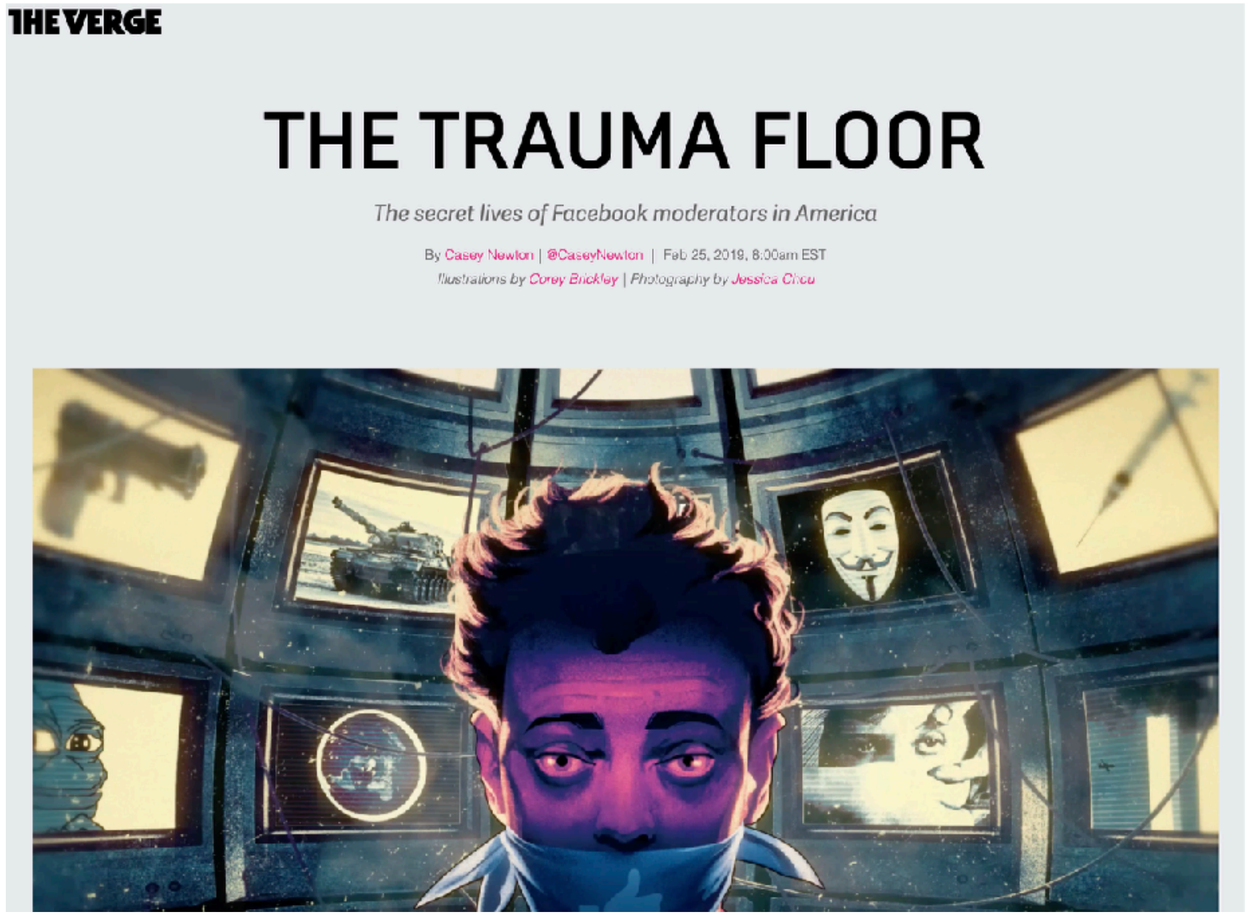
This article is more than 11 years old

Mums furious as Facebook removes breastfeeding photos

Not only that, content moderation has to contend with adversarial actors

- People trying to evade content filters
- People trying to get right up to the line but not cross it
- People learning about how the algorithm works and using that to boost their content (like SEO)
- People accusing the platforms of either over or under-moderation, or biased moderation, making platforms more hesitant about everything they do

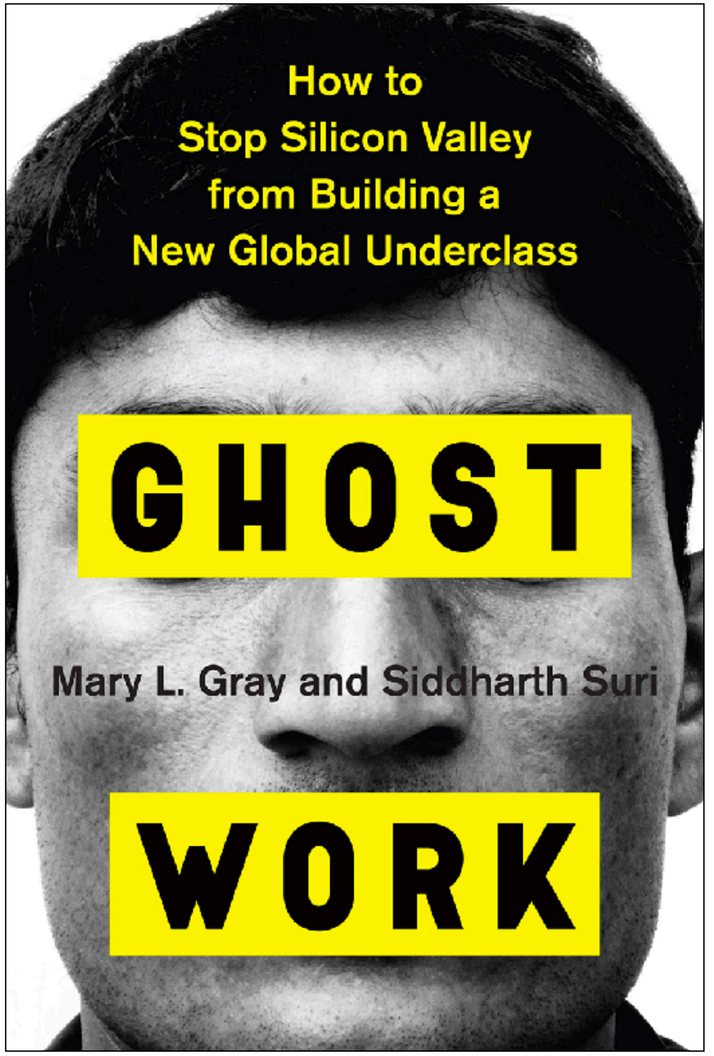
Commercial content moderation

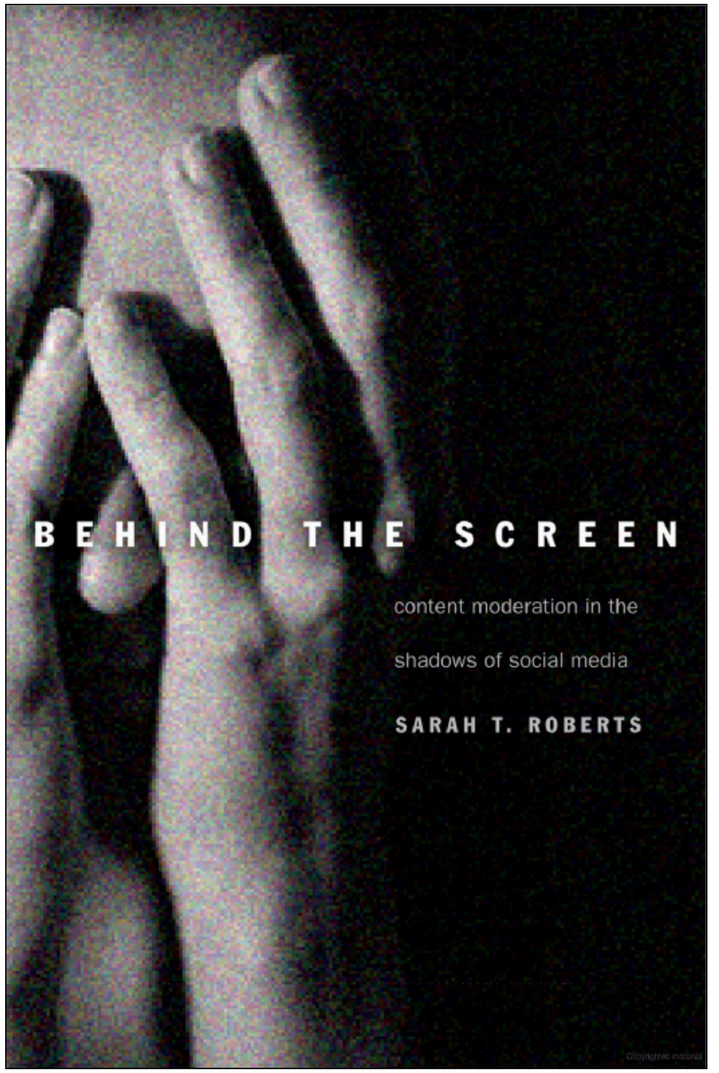


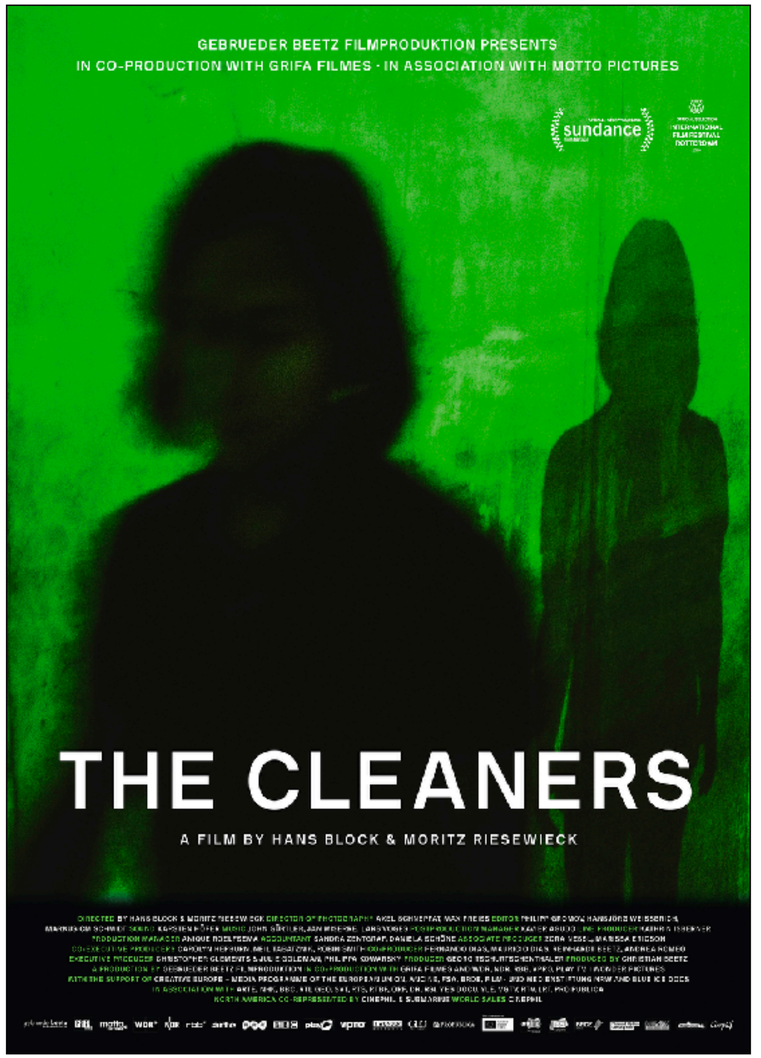
Just how many people? Facebook alone employs around 15,000 people. In total, the estimate is that around 150,000 “commercial content moderators” exist around the world.

Many are contractors, don’t have health benefits, see hundreds of pieces of content a day, taking <1 minute per piece, make close to minimum wage, and are outsourced to 3rd party companies in different places, in many cases completely different countries where wages are lower.

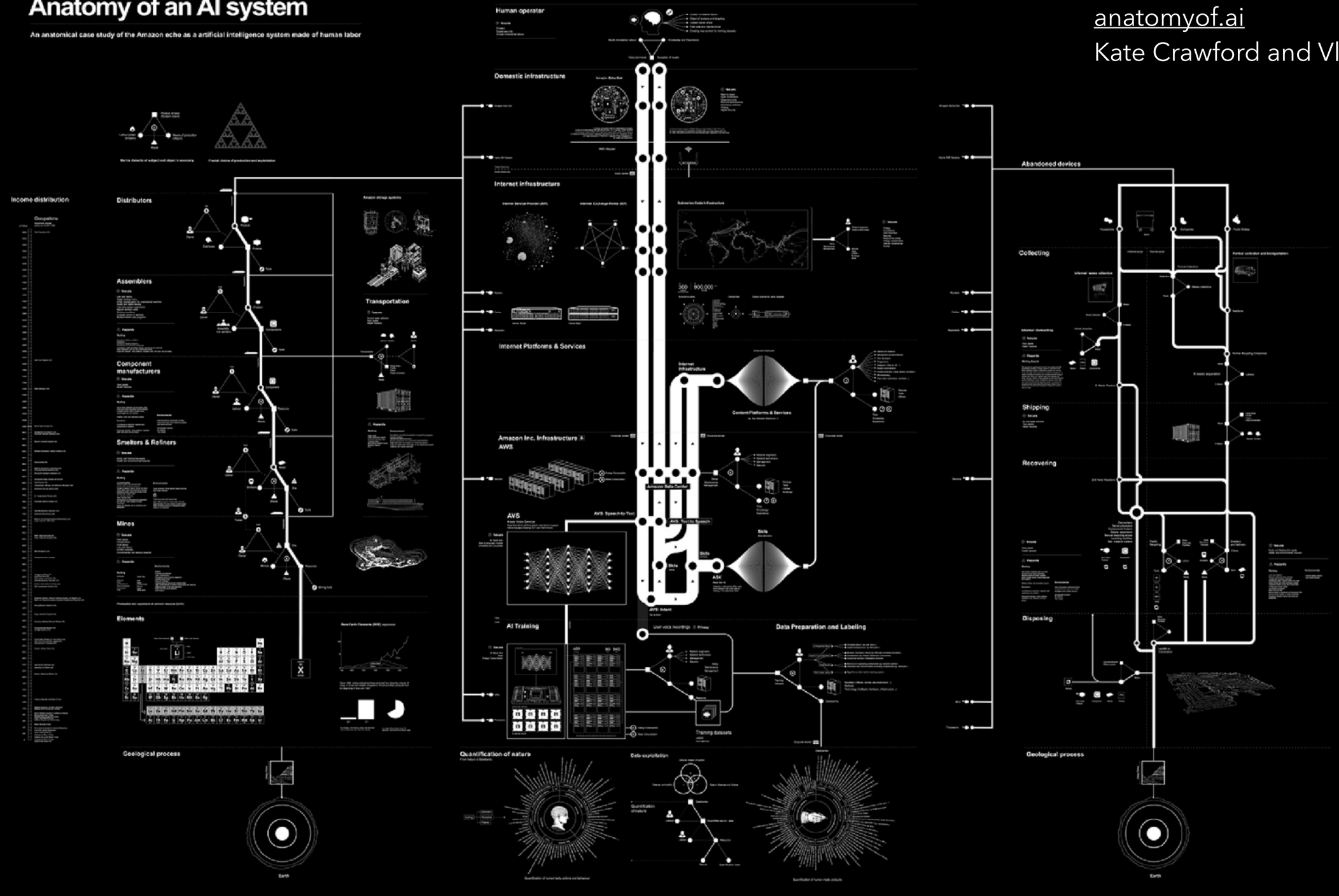
What are some potential problems with this setup?


How to
Stop Silicon Valley
from Building a
New Global Underclass
GHOST
WORK
Mary L. Gray and Siddharth Suri


BEHIND THE SCREEN
content moderation in the
shadows of social media
SARAH T. ROBERTS

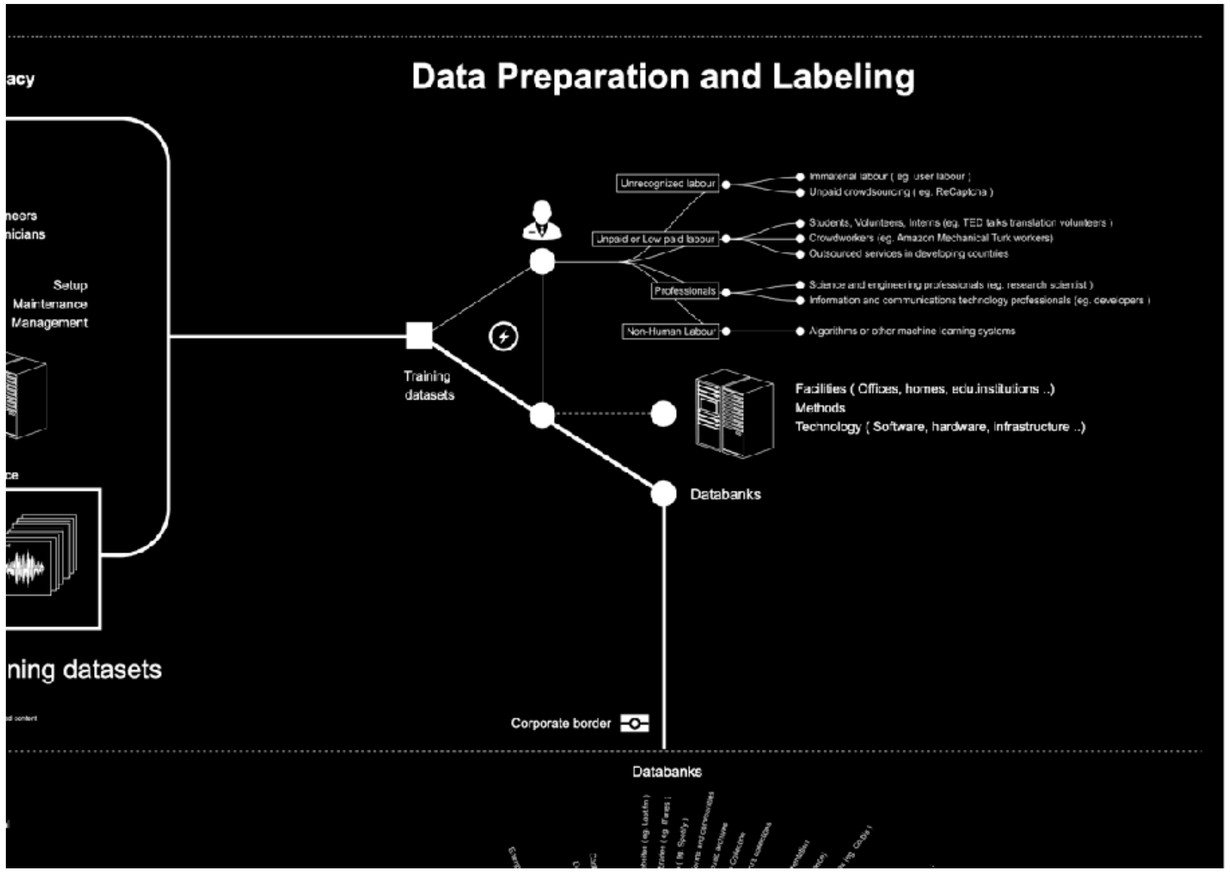

GEBRUEDER BEETZ FILMPRODUKTION PRESENTS
IN CO-PRODUCTION WITH GRIFA FILMES · IN ASSOCIATION WITH MOTTO PICTURES
sundance
THE CLEANERS
A FILM BY HANS BLOCK & MORITZ RIESEWIECK

AI content moderation doesn't mean there aren't humans



anatomyof.ai
Kate Crawford and Vladan Joler

Human resources behind AI



To train a model, you need lots and lots of data! To collect data, you need annotators to look at all that data and label it. So now you need to employ lots and lots of annotators. Sounds familiar?

There are also lots of cases where the model will be wrong. So you'll need a process where people are continually checking the output of the model and giving the correct label. Sounds familiar?

AI is biased…because people training them are biased

The Risk of Racial Bias in Hate Speech Detection

Maarten Sap[◇] **Dallas Card**[♣] **Saadia Gabriel**[◇] **Yejin Choi**[◇♡] **Noah A. Smith**[◇♡]
[◇]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, USA
[♣]Machine Learning Department, Carnegie Mellon University, Pittsburgh, USA
[♡]Allen Institute for Artificial Intelligence, Seattle, USA
msap@cs.washington.edu

Abstract

We investigate how annotators' insensitivity to differences in dialect can lead to racial bias in automatic hate speech detection models, potentially amplifying harm against minority populations. We first uncover unexpected correlations between surface markers of African American English (AAE) and ratings of toxicity in several widely-used hate speech datasets. Then, we show that models trained on these corpora acquire and propagate these biases, such that AAE tweets and tweets by self-identified African Americans are up to two times more likely to be labelled as offensive compared to others. Finally, we propose dialect and race priming as ways to reduce the racial bias in annotation, showing that when annotators are made explicitly aware of an AAE tweet's dialect they are significantly less likely to label the tweet as offensive.

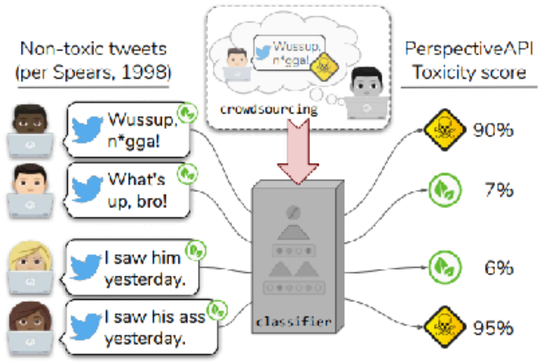


Figure 1: Phrases in African American English (AAE), their non-AAE equivalents (from Spears, 1998), and toxicity scores from PerspectiveAPI.com. Perspective is a tool from Jigsaw/Alphabet that uses a convolutional neural network to detect toxic language, trained on crowdsourced data where annotators were asked to label the toxicity of text without metadata.

You're never done. Data needs to be continually labeled

There are trillions of searches on Google every year. In fact, 15 percent of searches we see every day are new—which means there's always more work for us to do to present people with the best answers to their queries from a wide variety of legitimate sources. While our search results will never be perfect, we're as committed as always to preserving your trust and to ensuring our products continue to be useful for everyone.

Takeaway: consider the well-being of human moderators/annotators

You will always need humans involved in content moderation. The perfect AI doesn't exist for these kinds of tasks, and even if you do have a decent one (trained on labels by humans), it will constantly need to be updated with more data given data distribution shifts.

How can policies and technologies better support the well-being of these people?

- mental health support, better wages, less-precarious employment, ability for advancement, empowerment in their role as experts, visibility on the platform, supporting a local community instead of "view from nowhere"
- tools targeted to classify the worst content so people don't have to see it? or blur out parts that still capture the gist? or find near duplicates so things need only be labeled once? Connection to local child/animal support services to find the sources of this content?

Code and Design Specification

Your Code and Design Spec:

- Requirements document
- Storyboard
- Architectural Design (and optional code spec)

Requirements Doc

A requirements document describes what a use should be able to **do** with a product.

Typically has three parts, each with a primary (need to have) and secondary (nice to have) subpart:

- Functional Requirements
- Technical Requirements
- Usability Requirements

Requirements Doc

Functional Requirements

Primary:

- User will be able to select a song and send it to the other building
- User will be able to accept/reject the received song
- User will be able to react to currently playing song
- User will be able to see which building sent the currently playing song
- User will be able to see how many songs are in the queue
- User should be able to adjust the volume
- User should be able to see that someone is using the other room's installation

Secondary:

- User will be able to enter their name to be sent with their sent song
- Installations will use data of which songs were liked/disliked to choose a song of the day and/or song of the week
- User can see what volume the music is being played at in the other building

Requirements

Functional Requirements

Primary:

1. Users can see public blog posts.
2. Users can comment on public blog posts.
3. Users can subscribe to all blog posts or categories of blog posts using RSS feed readers.
4. Authors can save draft posts.
5. Admins can monitor, edit, and roll-back posts as necessary.

Secondary:

1. Users can create profiles (i.e. gravatars).
2. Users can receive individual or aggregated emails that contain blog content.
3. Authors can forward-date posts.

Break down what the user can do into a list of discrete actions

Requirements Doc

Technical Requirements

Primary:

- Based on a user's song request, get the song and play it using the Youtube API
- LEDs on the installation light up if someone is near the other building's installation
- Motion sensor will recognize if there are people near the installation
    - Input from the motion sensor will be the output on LEDs in the opposite building
        - For example, if motion is detected near the installation in Gates, the LEDs on the installation in Allen will light up
- Functional buttons for user input (built based on existing keyboard's/number pad)
- Screen as display
- Music is played synchronously across the two buildings
- Website will be hosted on github pages

Secondary:

- Lights will be synchronized to the music
- Wavelength graphic synced with music that is playing
- Host on AWS

Requirements

Technical Requirements

Primary:

1. Cross-browser /platform support (IE, Firefox, Chrome, Safari - PC and Mac)
2. Mobile support (for advanced smartphones – iPhone,iPad, Android or when possible for older text based phones). We will let the chosen system that meets all other criteria best, set our minimum level of device support.
3. System will allow changes to minimum and maximum reservation times.

Secondary:

1. Statistics and reporting.
2. The system should be built using free open source software (FOSS) if locally hosted.
3. If hosted, the system should be built using an application framework, rather than a hard-coded. This will help the programmers to account for differences in operating systems, interfaces and displays.

Turn those user capabilities into technical components (what will the *system* do and what does it need to be able to do it?)

Requirements Doc

Usability Requirements

Primary:

- Flow of screens is intuitive
- Buttons are placed in consistency with the interface screens
- Installation is socially acceptable

Secondary:

- User will see who proposed the currently playing song in the other building
- User will see who reacts to the currently playing song

Requirements

Usability Requirements

Primary:

1. The system will fully function in major browsers.
2. The system will support mobile users in some way.
3. ADA compliance (or alternative booking support via email, phone, or online form?) We could check our interfaces against ADA software.

Secondary:

1. Beta usability testing will not be possible given the time constraints for this project. Possible use of feedback forms during the fall?

What additional things will you need to look out for as a builder + designer to make the produce usable for your target user group?

Storyboard

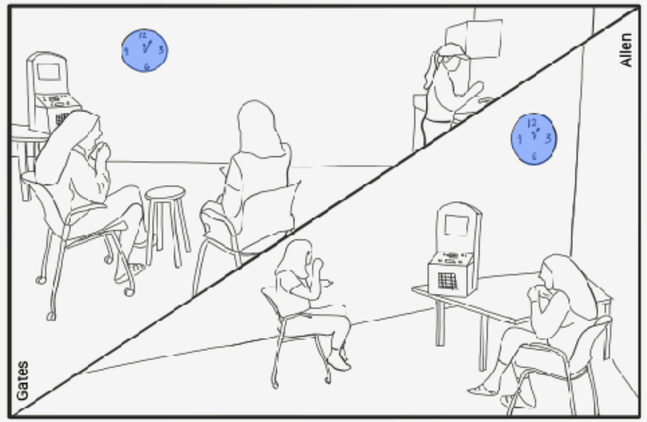
- Show the **whole interaction** including how it starts (what's the spark before you're even using the product?) and ends

- It should look something like a comic strip panel, including title and captions

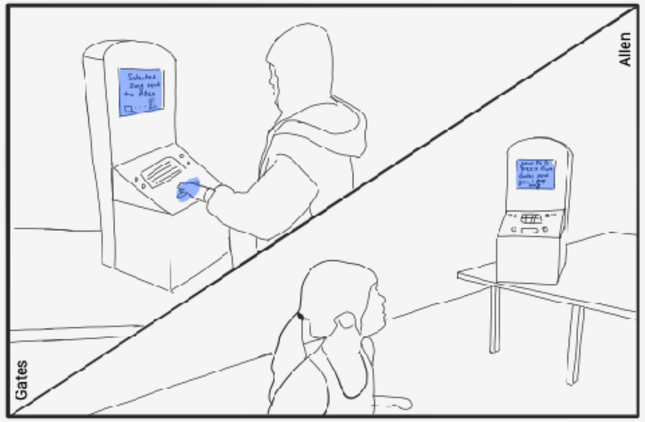
- You should have 2 or 3 flows (for different users/different scenarios)

- You can use drawings or figma mockups to describe the flow from screen to screen.

Storyboard

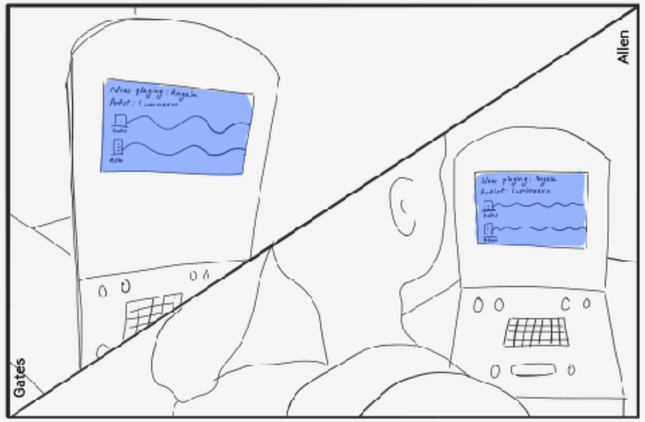


1. It is lunch time and a few Ph. D students take their breaks in both the Jake (Allen Building) and the research commons (Gates building).

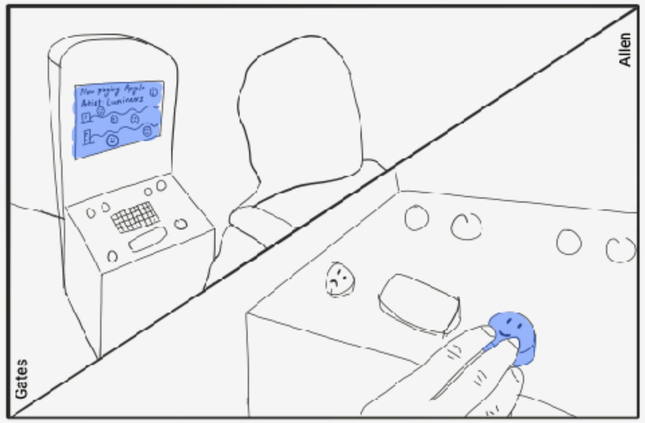


2. Emma in Gates approaches Jucebox installation and types the song "Party in the USA" which is then sent to Allen. Jessica in Allen sees a notification on the installation screen.

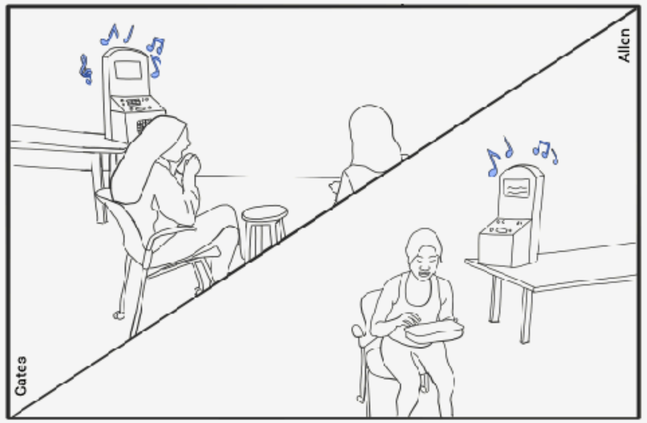
3. While Emma waits for a response from Allen, Jessica sends an accepted meme to notify Emma she accepts the song.



4. Now that the song was accepted, "Party in the USA" is playing in both buildings simultaneously.

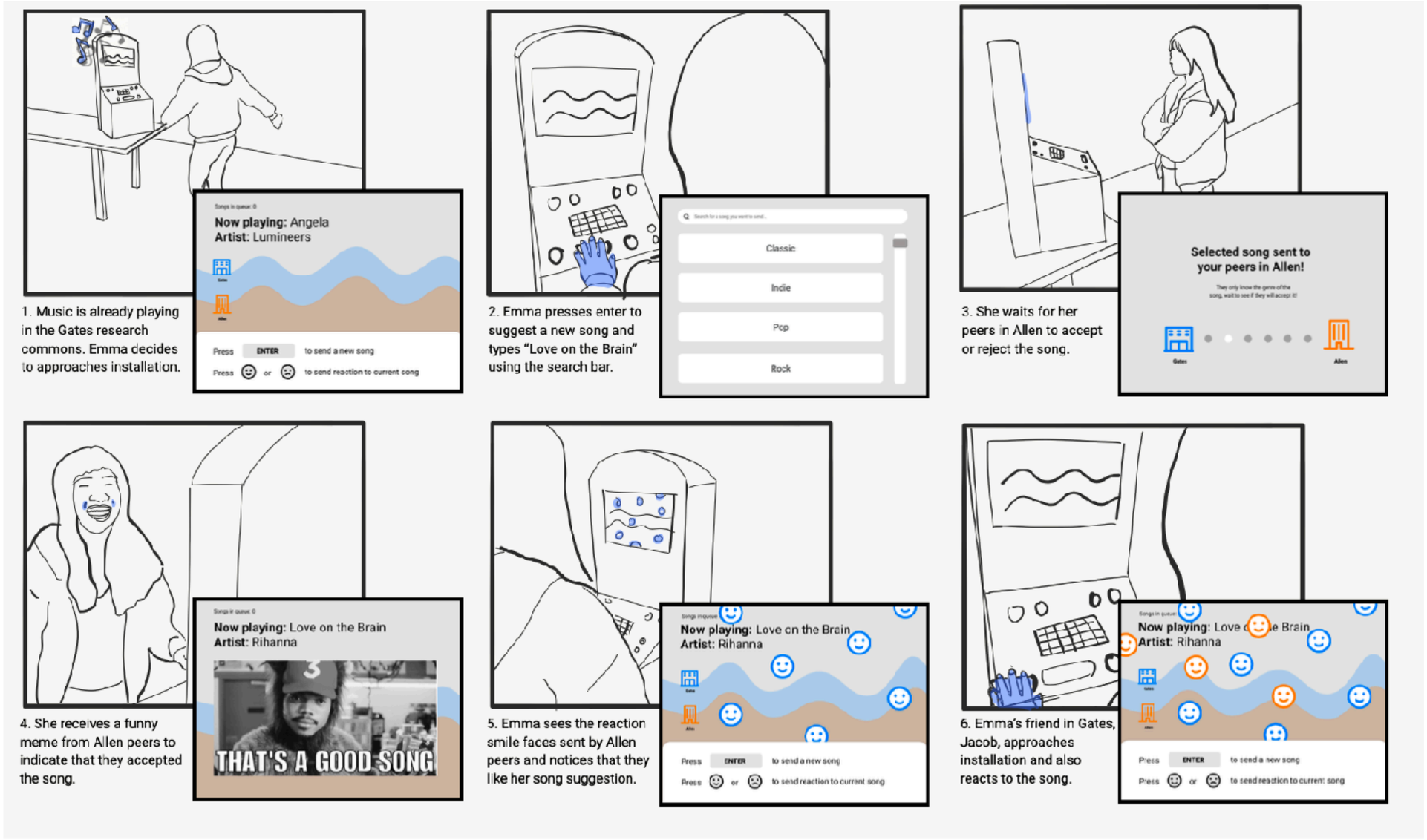


5. Jessica clicks the smiling emoji button to react to the song. The reaction emojis display on the screens in each building.



6. In both buildings, people eat lunch and enjoy the music that was sent by Emma in Gates building.

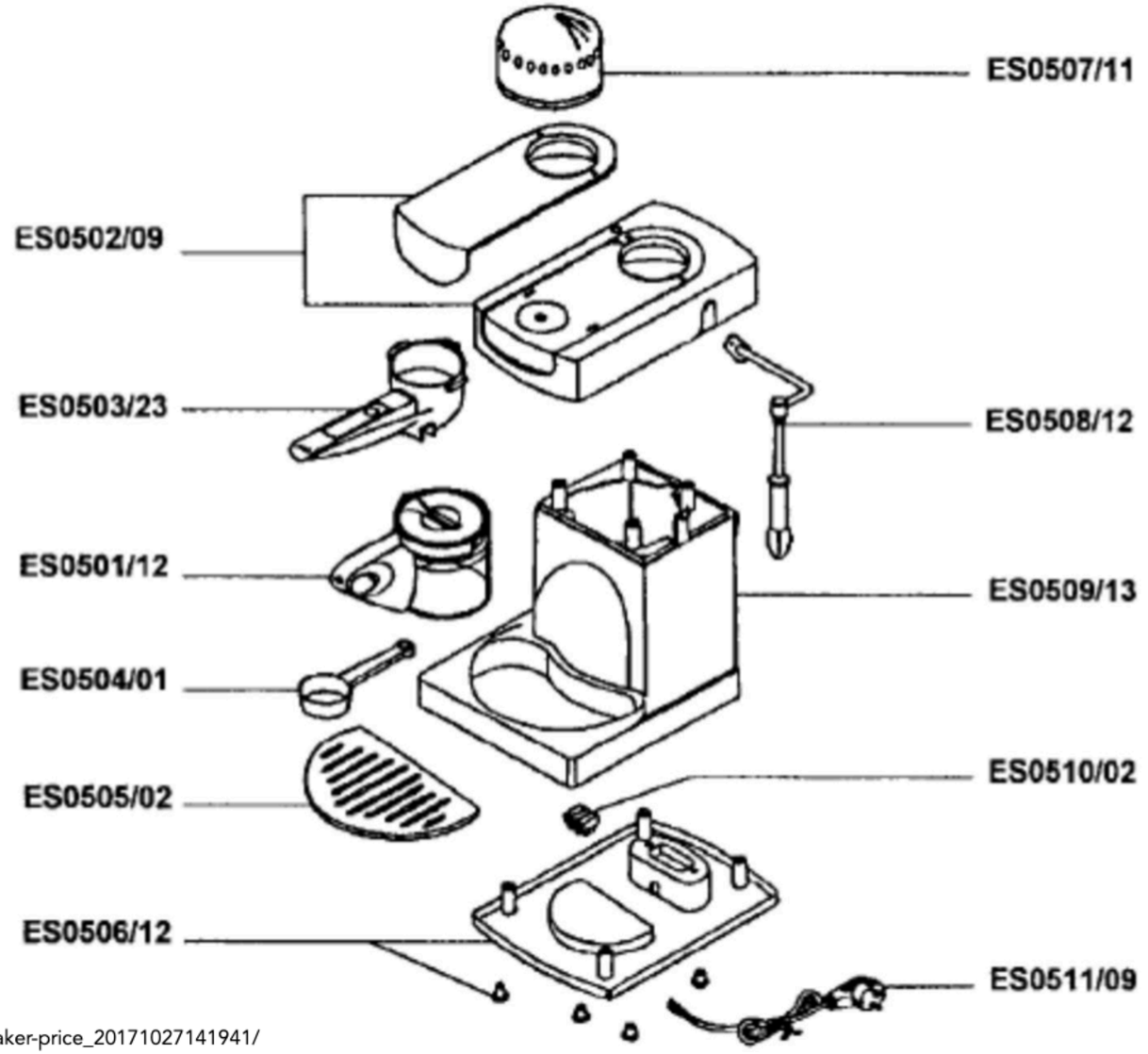
Storyboard


Now playing: Angela
Artist: Lumineers
Press ENTER to send a new song
Press or to send reaction to current song
1. Music is already playing in the Gates research commons. Emma decides to approaches installation.
Classic
Indie
Pop
Rock
2. Emma presses enter to suggest a new song and types "Love on the Brain" using the search bar.
Selected song sent to your peers in Allen!
3. She waits for her peers in Allen to accept or reject the song.
Now playing: Love on the Brain
Artist: Rihanna
THAT'S A GOOD SONG
4. She receives a funny meme from Allen peers to indicate that they accepted the song.
Now playing: Love on the Brain
Artist: Rihanna
Press ENTER to send a new song
Press or to send reaction to current song
5. Emma sees the reaction smile faces sent by Allen peers and notices that they like her song suggestion.
Now playing: Love on the Brain
Artist: Rihanna
Press ENTER to send a new song
Press or to send reaction to current song
6. Emma's friend in Gates, Jacob, approaches installation and also reacts to the song.

Architectural Design

- Diagram of components (organized into subsystems/ subcomponents)

- Show all the parts necessary



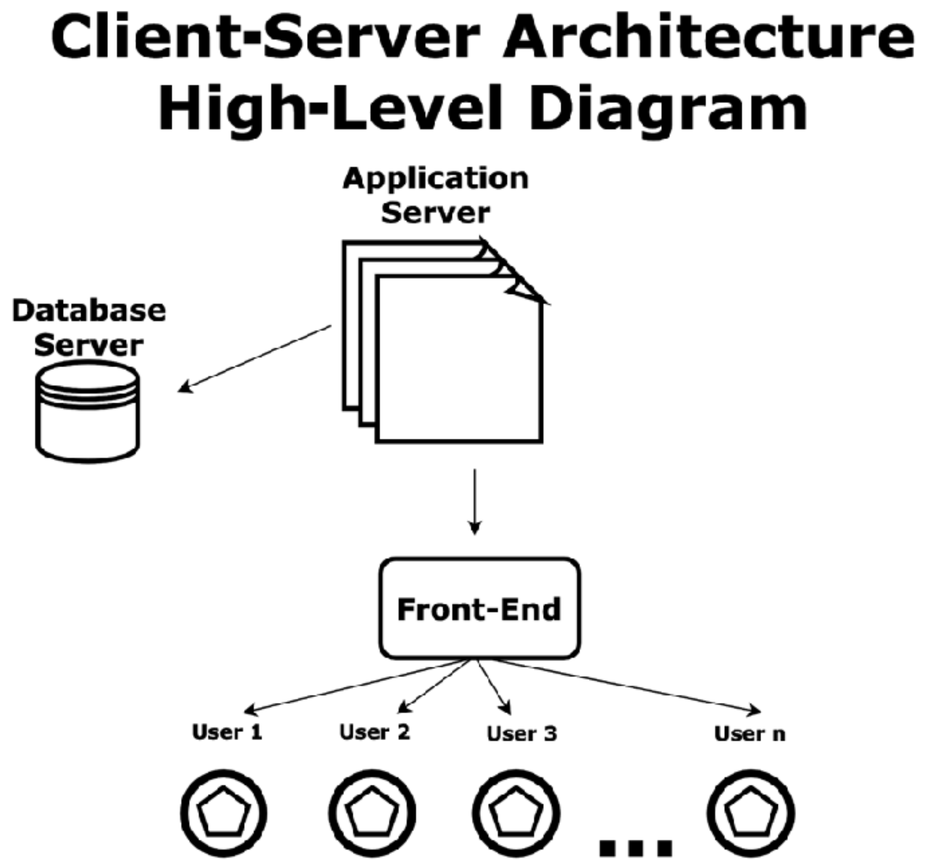
http://tiawuk.com/rowenta-coffee-maker-price_20171027141941/

- Likely not as complicated for you all since you are building software (for the most part)

- But even then, there are subcomponents

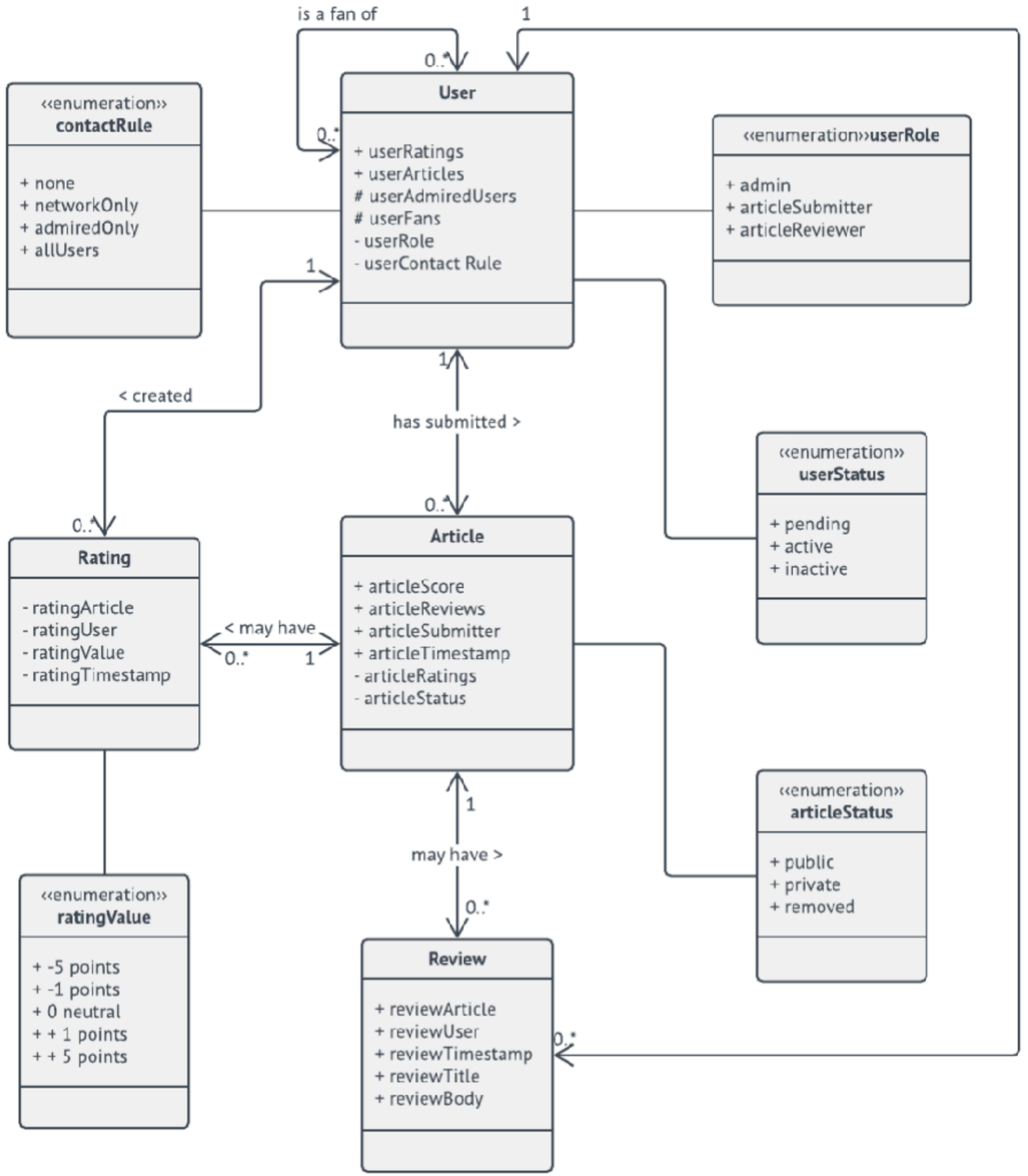
- Server vs desktop client vs mobile client, database vs web server

(Side note: If you're looking for a way to scope down, consider removing your backend and faking the data from a database)

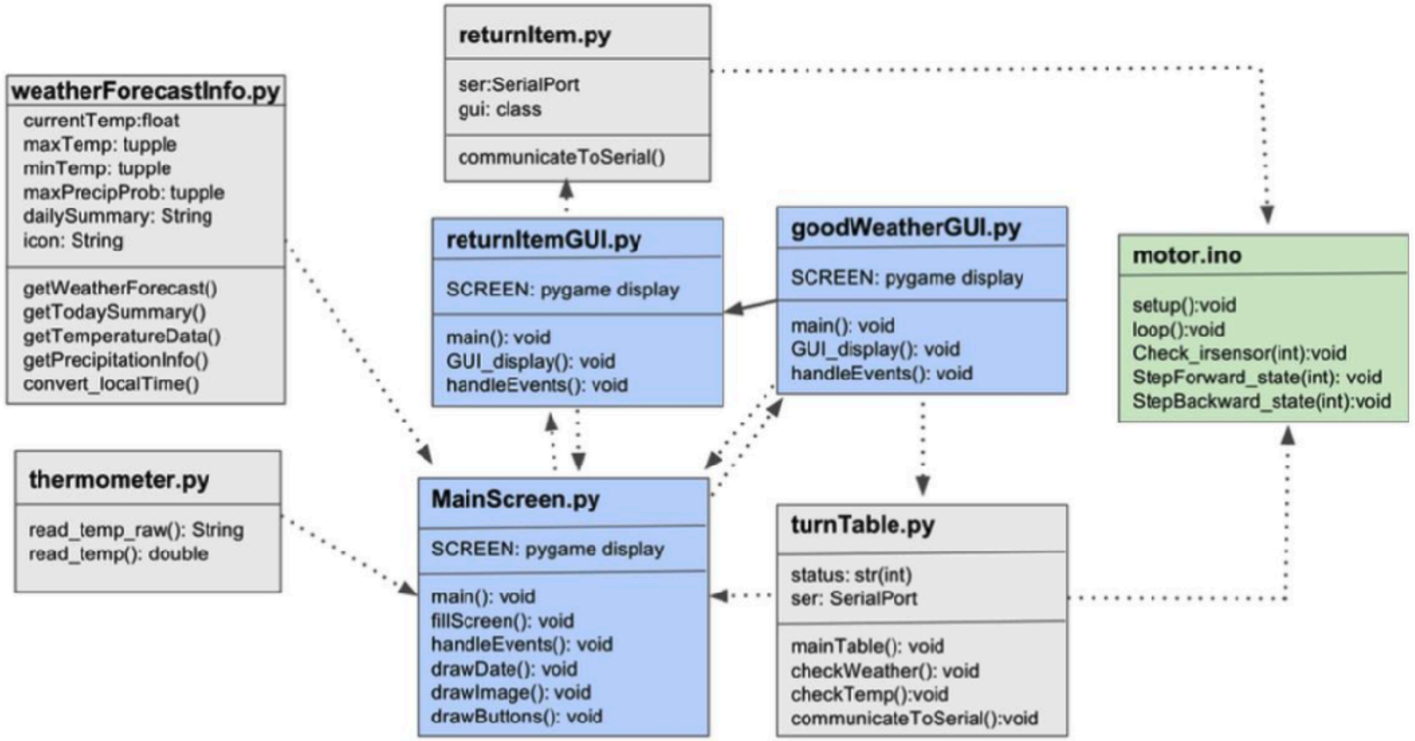


- Spend more time on spec'ing out your code architecture

- UML diagram describes different classes, attributes, methods, and relationships between them



Example UML diagram (Unified Modeling Language)



http://poe.olin.edu/2015/wsmart/firmware--software.html

Void highlightSelection()
- Parameters: none
- If up/down arrow key is pressed, the selection gets highlighted

Void sendSong()
- Parameters: building name
- Call getSelection() to get the song name, and based on the name, get its correlated genre
- Create a new message to send to the other user that will only contain the genre of the song and which building it was from
- If "enter" key gets pressed, the song gets sent to the other building
    - The song object will be sent to a database (firebase) and the other building will be subscribed to any changes in database and that's how the message will be sent
    - Prior to sending the song, if the building's "attemptsLeft" is at 3 (which is the max number of retries), the user will see a pop up that says they have to wait for a few minutes before trying again.

String getSelection()
- Parameters: if movement came from search or from the genre selection
- If parameter = search
    - Get the user input from text box and return it
- If parameter = genre
    - Get the user input from the div and return it

String addToQueue()
- Parameters: song object
- If queue is at max length, return string that says "queue is maxed"
- If queue still has space, add the song object to the queue and return string "song queued"

String removeFromQueue()
- Parameters: song object
- If queue is already empty, return string "queue is already empty"
- If queue has songs to remove, remove the song from the queue and return string "song removed"

Code Spec [Optional for G4 but probably something you want to do anyway after determining your architectural design]

- Write out in more detail each method and what it will do.
    - You can add more details like what arguments it takes and what it returns.