Social Computing Capstone

Day 11: Commercial Content Moderation

CSE 481p | Spring 2023

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

Kevin Feng
TA
Ph.D. Student | University of Washington, Human-Centered Design & Engineering

Schedule for today’s class

- Discussion with Lindsay Blackwell (30 min)
- G4 walk-through (15 min)
- Group work time (45 min)

Lindsay Blackwell

- Head of Trust & Safety for Sidechat and Yik Yak
- Former senior research on Twitter's Content Health team
- Former lead researcher on Facebook's Hate Speech and Violence & Incitement teams
- Research collaborator on anti-harassment projects with Oculus, Instagram, and other products
- ABD PhD on social media governance and theories of justice from University of Michigan
- Can talk about: trust and safety work within industry, commercial content moderation procedures, restorative justice and other frameworks for reconsidering social media governance, industry vs academia, online harassment research, social VR, etc.

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

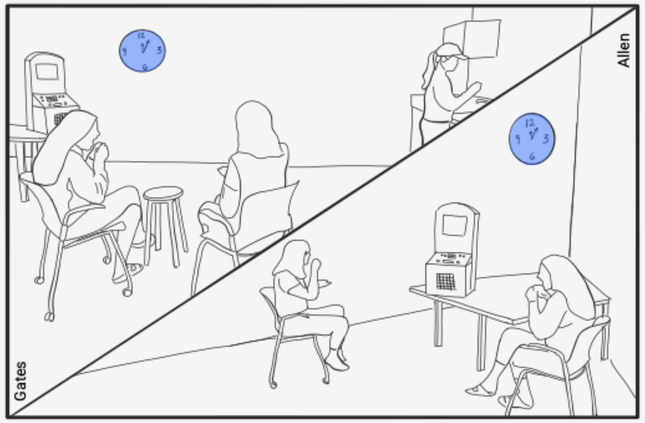
- Show the **whole interaction** including how it starts (what's the spark before you're even using the product?) and ends

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

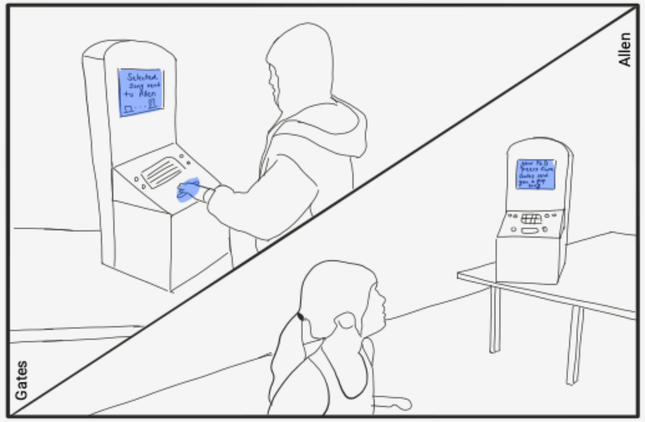
- You should have 2 or 3 flows (for different users/different scenarios)

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

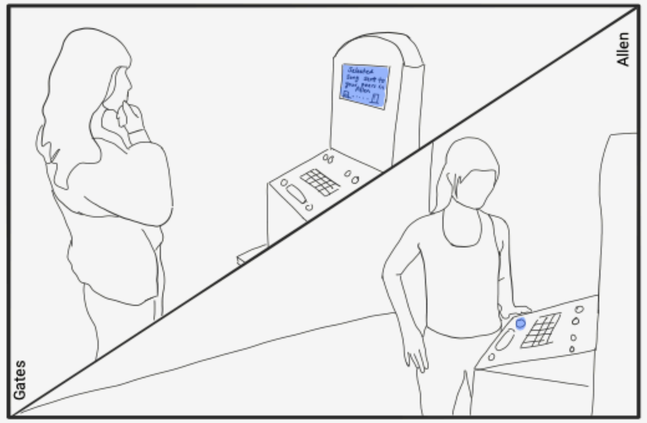
Storyboard



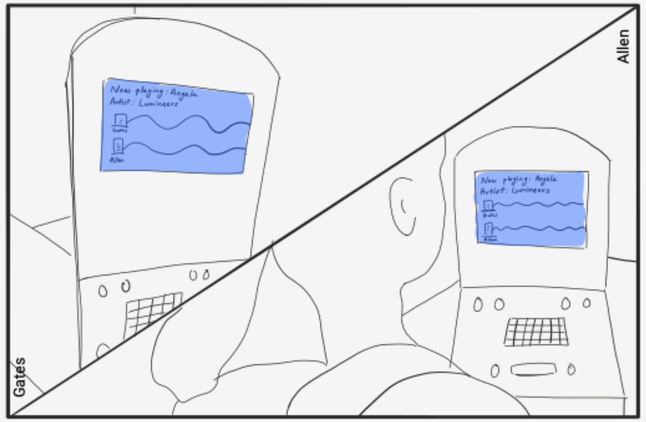
1. It is lunch time and a few Ph. D students take their breaks in both the Jake (Allen Building) and the research commons (Gates building).



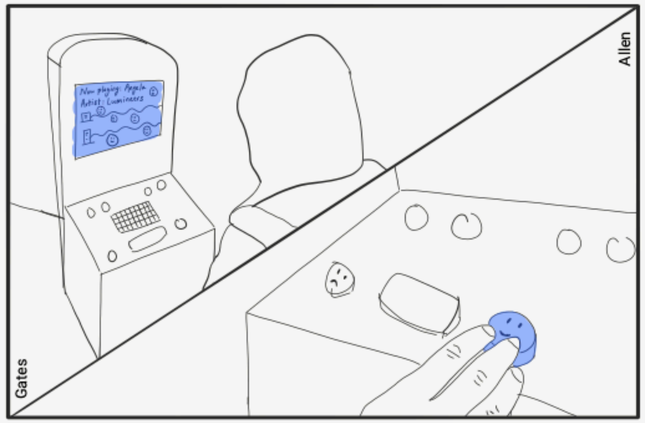
2. Emma in Gates approaches Jucebox installation and types the song “Party in the USA” which is then sent to Allen. Jessica in Allen sees a notification on the installation screen.



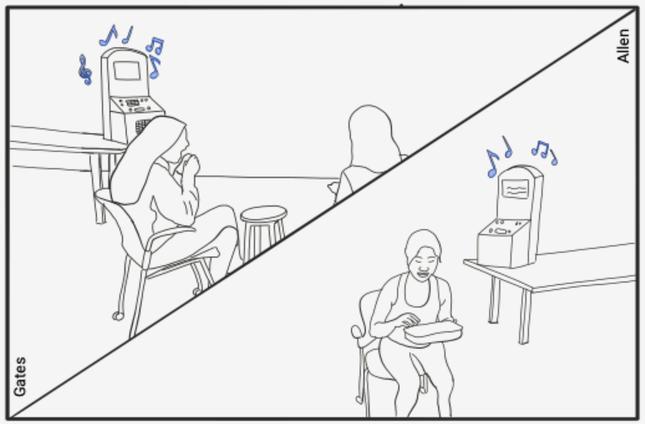
3. While Emma waits for a response from Allen, Jessica sends an accepted meme to notify Emma she accepts the song.



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

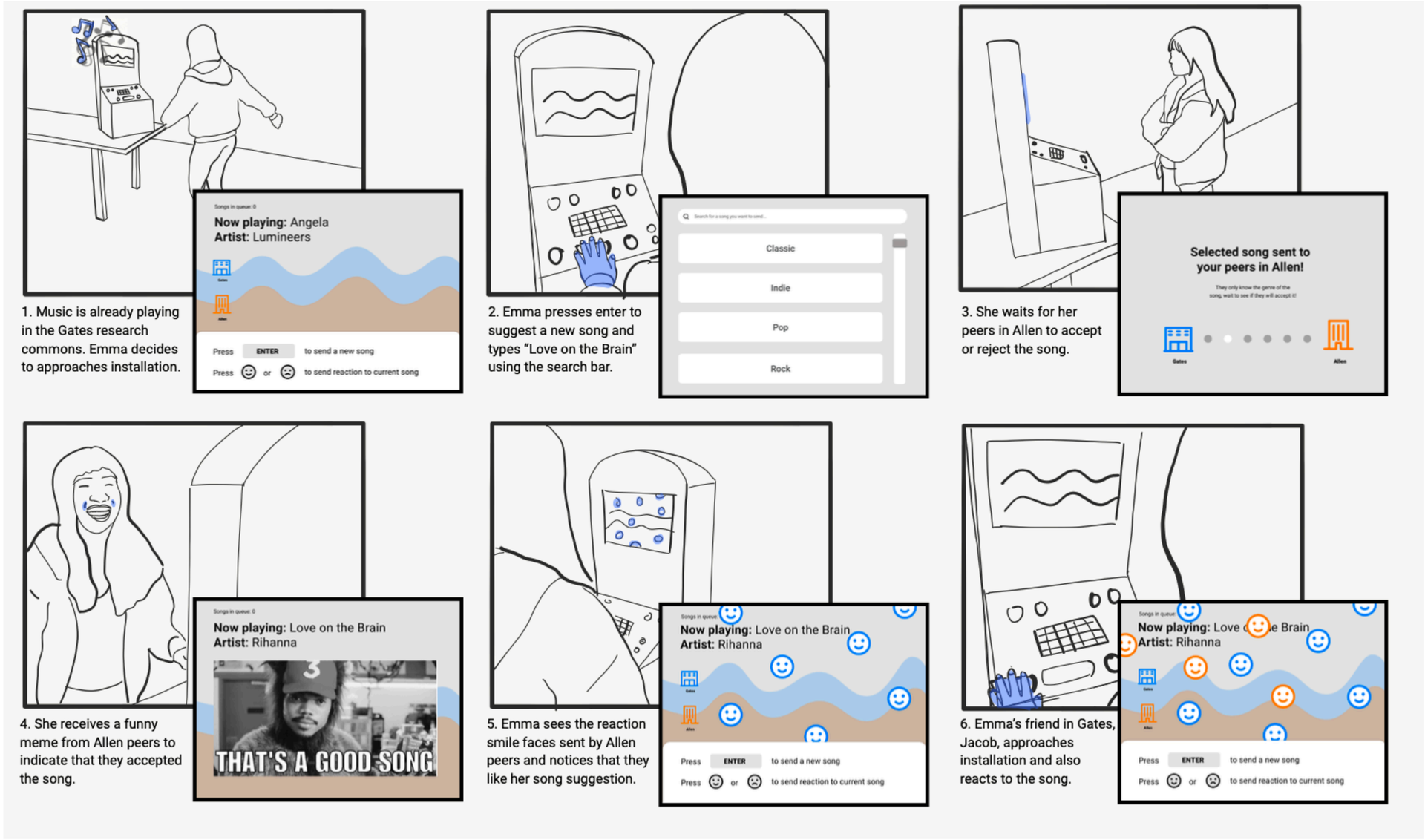


5. Jessica clicks the smiling emoji button to react to the song. The reaction emojis display on the screens in each building.



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

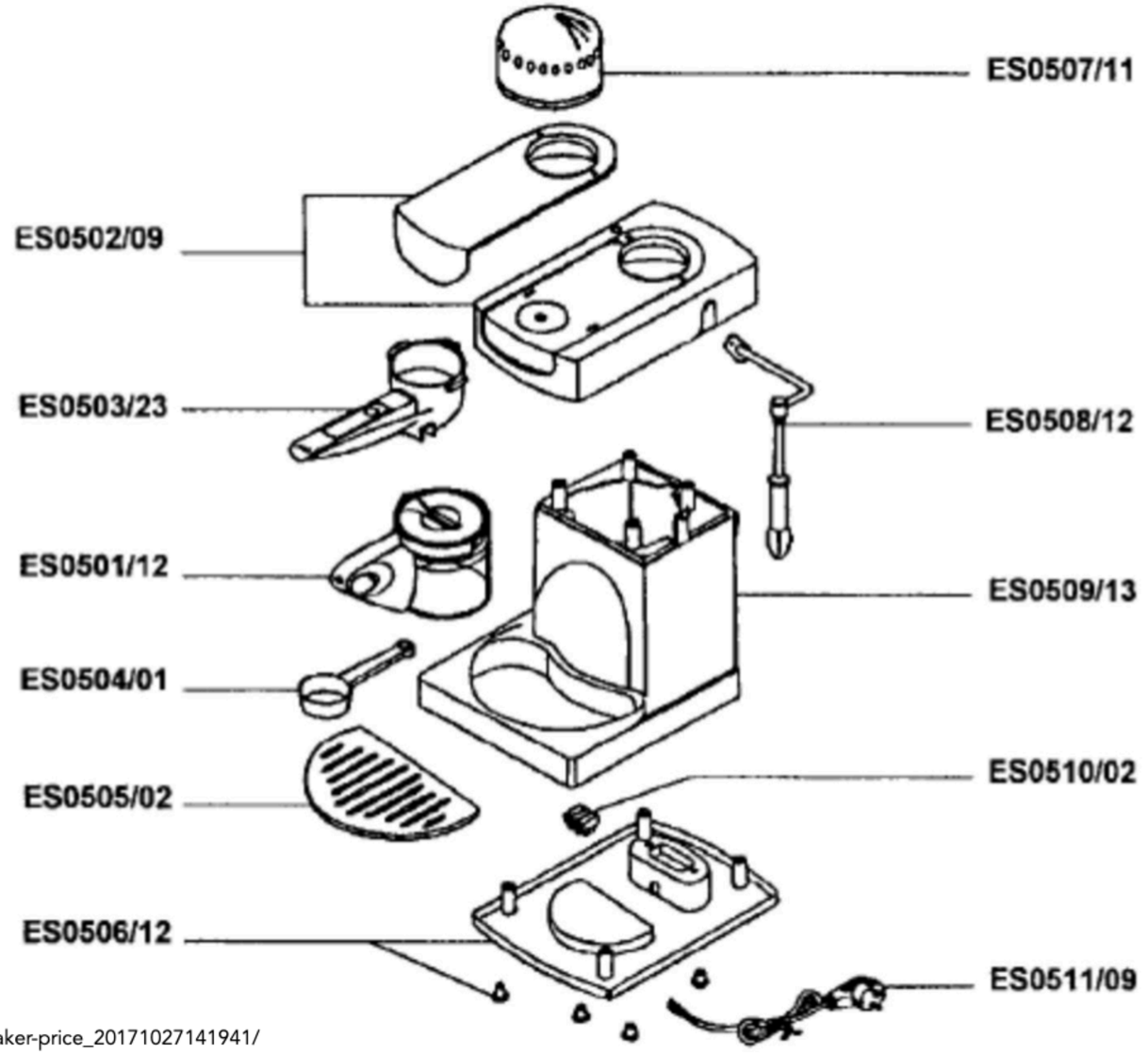
Storyboard


Songs in queue: 0
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.
Search for a song you want to send...
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!
They only know the genre of the song, wait to see if they will accept it!
Gates
Allen
3. She waits for her peers in Allen to accept or reject the song.
Songs in queue: 0
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.
Songs in queue:
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.
Songs in queue:
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



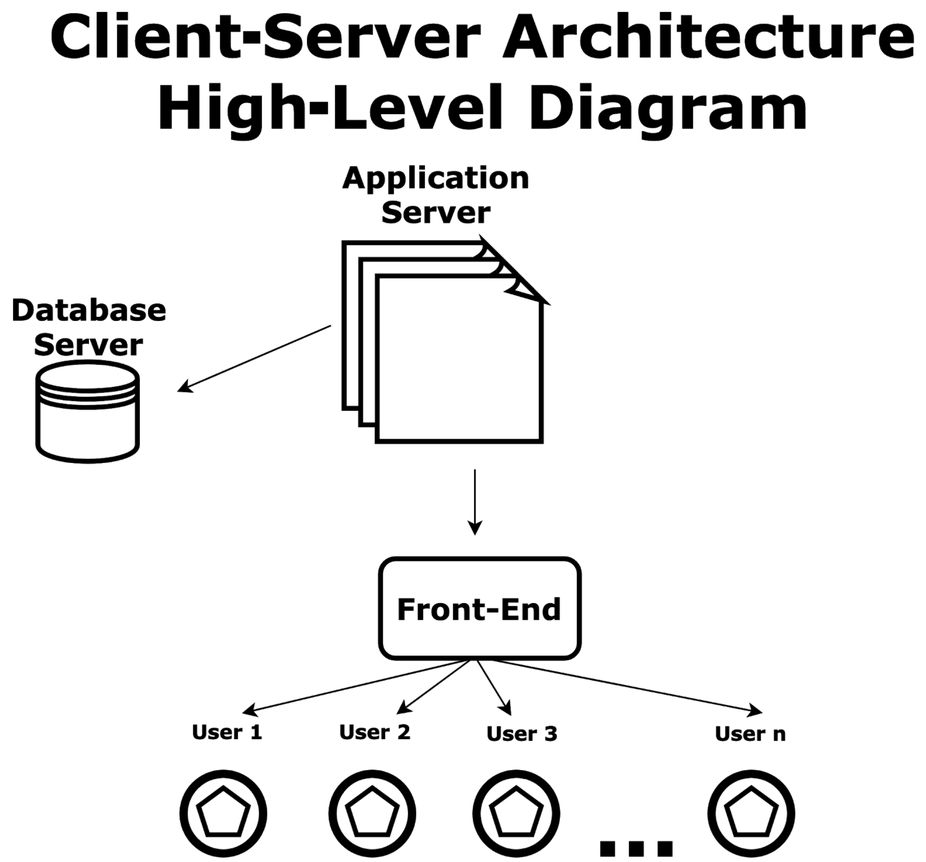
http://tiawuk.com/rowenta-coffee-maker-price_20171027141941/

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

- But even then, there are subcomponents

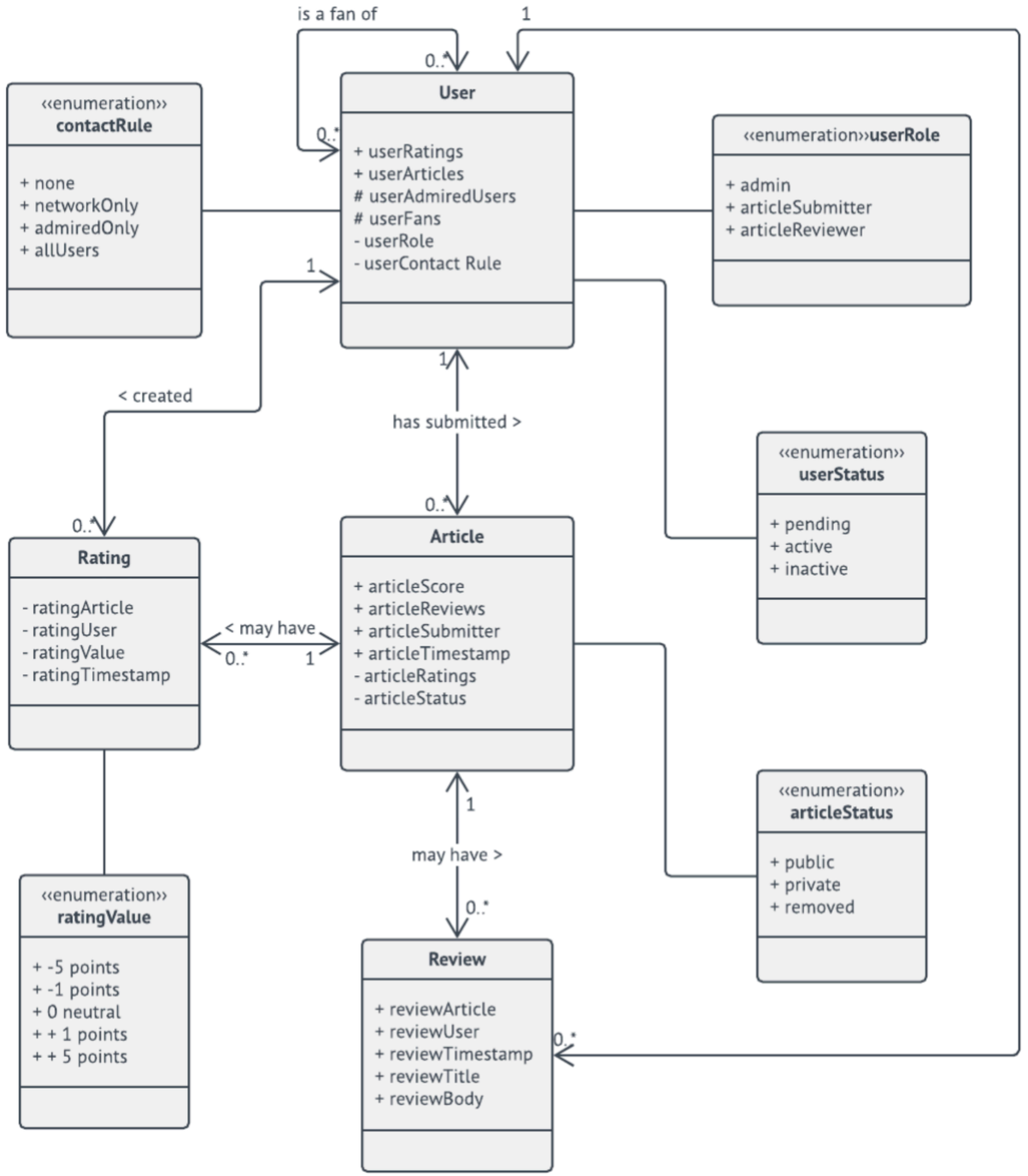
- Server vs desktop client vs mobile client, database vs web server

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

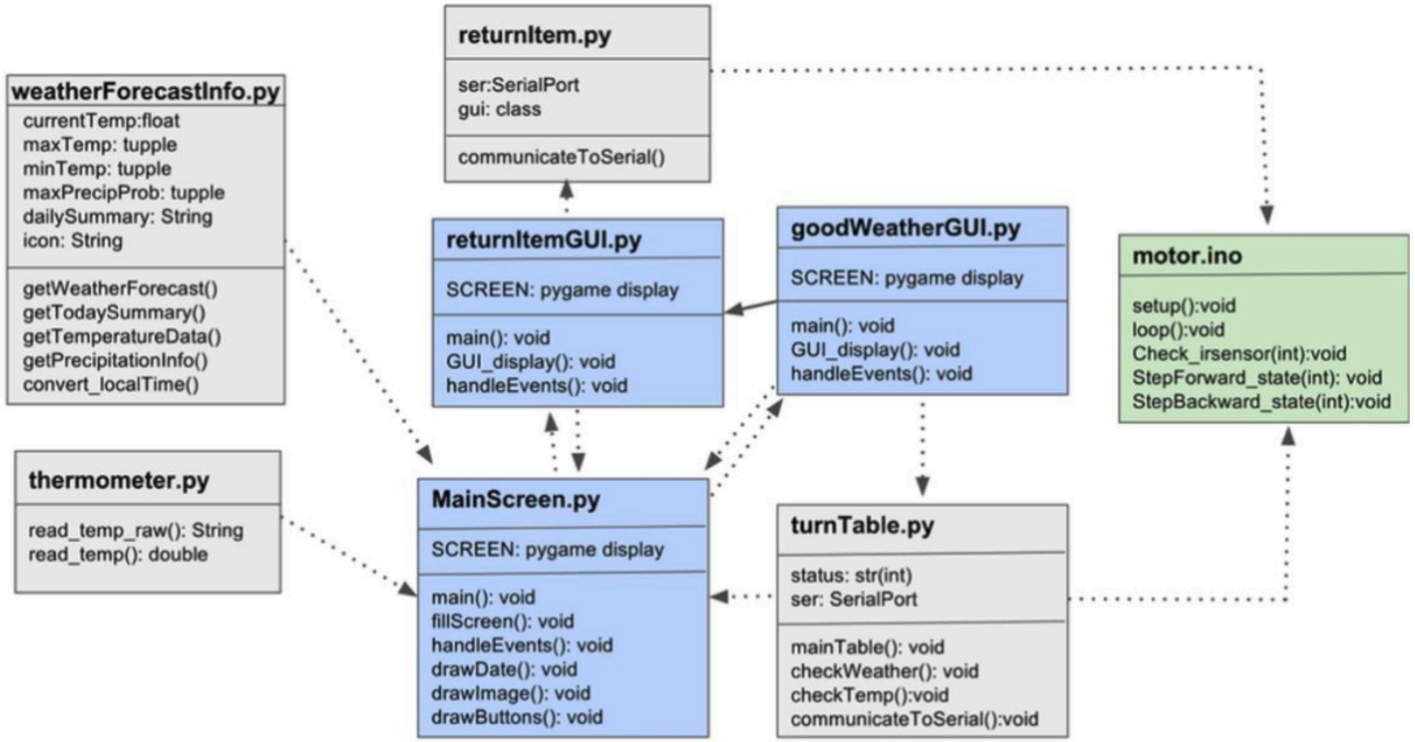


- Spend more time on spec'ing out your code architecture

- 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.