Team MOSSY
Maddie Ostergaard (Project Manager), Sarah Yoon, Seiji Yawata

Task 1: DUE 8am Monday MORNING 12/5

- Create a design document for the project.
- Include a timeline for your project with incremental deadlines.
- Include a component map, site map and database schema (if applicable), as well as any necessary supporting documentation.
- Divide the tasks among your group members. Include a "Project Manager"
    - The Project Manager cannot have held that role for Project 0
    - Project manager should also have coding tasks, but will also make sure the group is consistently moving together.
    - Project manager should make certain the design document is coherent and that the group is adhering to the agreed design.
    - If changes need to be made to the design, project manager should be informed.
    - There will be a summary document at the end of the project created by the group, but the project manager will have certain duties pertaining to that document as well.
- Create a pdf titled **DESIGN.pdf** in the **upgraded-parakeet** repository.

Product:

MOSSY (Missing Out on Shows Sucks, Yo) is a service that takes the user's location and looks through the user's saved music on Spotify (probably artists from whom a certain minimum number of songs are saved??) and sends the user text notifications (using Twilio) about concerts in that area from these "favorite" artists (finding info from and sending links to ticketmaster).

app.py

- Renders templates
- Login → figure out OAuth (login through Spotify)
- Dashboard
    - Calls sortby(date) - default
    - Low priority: Preferences page (how many artists to display, look for different location, change password)
    - Dropdown to organize by other attributes
    - Search bar: from event list
    - Display events/links
    - Should there be a way for the user to refresh event recommendations (i.e., call sift/tix again)?

sort.py

- Take tix.py's output
- sortby(<attr>) returns list of dictionaries (each dict = one event) sorted by given attribute
- delete(<event name>) removes that event from list

sift.py

- Look through user's Spotify library
- Sort artists from greatest to least # of songs saved by each artist
- Return a dictionary of top <=10 artists, their Spotify pfp and # of songs from each

tix.py

- Get user's location from ip info API
- Search ticketmaster API for concerts by each artist in vicinity of user
- Each event would have
    - Event Name
    - Artist
    - Venue
    - Date
    - Distance from current location to venue
    - Number of songs saved from this artist
    - Link to ticketmaster page
    - Interested
    - Going
- Return a dictionary of events

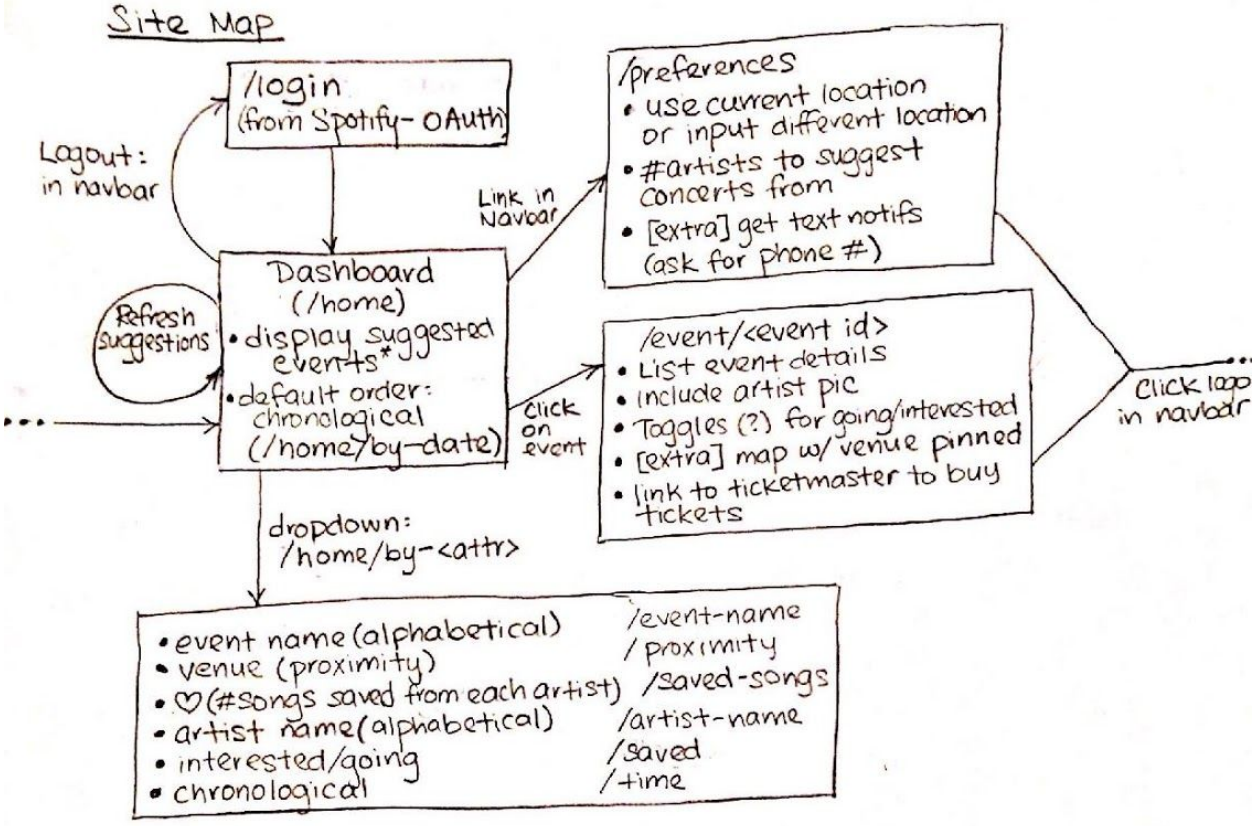
sms.py (Optional)

- Would require app.py to ask for phone # at registration and for permission to send SMS notifications under Preferences
- Using twilio API, sends texts to users with notifications about upcoming events (same info as what is displayed in the Dashboard)
- Would have to set frequency for how often recommendations get refreshed

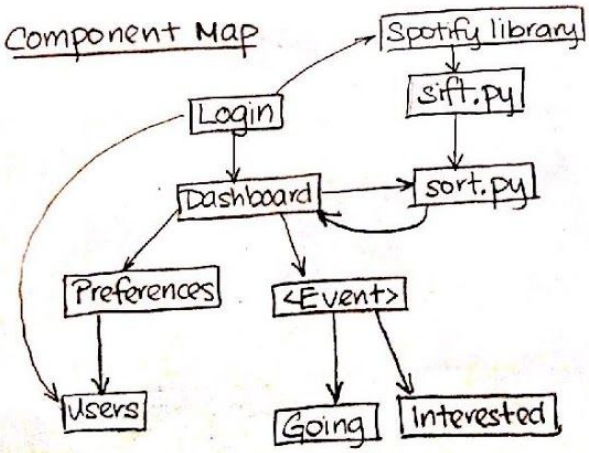
MOSSY

Maddie Ostergaard
Sarah Yoon
Seiji Yawata

Site Map



Component Map



APIs and Where We're Using Them

- Spotify [https://developer.spotify.com/web-api/]
- Ticketmaster [http://developer.ticketmaster.com/]
- Ipinfo [http://ipinfo.io/developers/getting-started]
- Twilio (potentially) [https://www.twilio.com/docs/api/rest]

Database Schema (?)

- Users
    - Usernames // location (can either be current or a saved location) // # of artists to look for
- Events they're going to
    - User // event
- Events they're interested in
    - User // event

Tasks

- Figure out how Spotify OAuth works → save usernames to database
- Add to Spotify login stuff with our own database stuff
- Write sift.py (figure out how to get info from user's Spotify library, how to search for events from ticketmaster with artist name, fake area)
- Use real geographic area (from ip info API)
- Write sort.py (using fake account data)
- Make sift.py/sort.py work with database stuff
- Add preferences form → info goes to database
- Figure out how to change generic location descriptor (i.e. city name → zip code)
- Take output from sift/sort to generate dashboard
- Write page for individual event
- Work on beautifying html files for home, event pages, preferences, etc. using BS (bootstrap)

Timeline

TASK	PERSON	DEADLINE (Mr. DW's new favorite word) *due in the morning*
Figure out how Spotify OAuth works → save usernames to database	Seiji	Monday
Add to Spotify login stuff with our own database stuff	Maddie	Tuesday
Write sift.py (figure out how to get info from user's Spotify library, how to search for events from ticketmaster with artist name, fake area)	Sarah	Tuesday
Use real geographic area (from ip info API)	Seiji	Tuesday/Wednesday
Write sort.py (using fake account data)	Sarah	Wednesday
Make sift.py/sort.py work with database stuff	Maddie	Wednesday
Add preferences form → info goes to database	Maddie	Thursday
Figure out how to change generic location descriptor (i.e. city name → zip code)	Sarah	Thursday
Take output from sift/sort to generate dashboard	Maddie	Thursday
Write page for individual event	Sarah	Friday
Work on beautifying html files for home, event pages, preferences, etc. using BS (bootstrap)	Seiji	Sunday