FINAL PROJECT
The Slap Chamber

Pd 9 | Projection of Despair in Four Stages
James Smith, Max Korsun, Angel Ng, Connie Lei

WHAT DOES IT DO?

Our project is a website that allows many people to listen to music together and create playlists collaboratively. It will be divided into rooms that people can join and listen to the music being streamed from Youtube. Users may create their own rooms where they can make it a private room with a passcode or a public room where anyone can join. Users may join rooms from a shareable link or from a search page. Once in a room, users can submit music to the room playlist which will just be a Youtube track link. Once submitted, it will be next in the queue and it will be added to a public Youtube playlist that users can save to their actual Youtube account. If the music room runs out of new songs in the queue, it will shuffle songs in the playlist that has been constructed from all user submissions. Users will be able to add rooms to their favorites list to go back and find later.

Stretch Goals: Provide permissions to the creator of the room to automatically skip songs, change the password of the room, and remove songs from the playlist. Be able to start a vote to skip a song which will be carried out if the majority deems it so. Display the current queue of music in the room, built in youtube search function to queue music.

KEEPING USERS IN SYNC

To prevent users in a room from getting desynced due to ads or internet issues, we will be using websockets in conjunction with the Youtube API. With websockets we can ensure that everyone in a single room will be at the same spot with information provided from the youtube api. For example, we always know when a user is buffering, or when a user is watching an ad, and we can instruct the other clients to pause and wait for the other user. For the sake of simplicity, users will not be able to skip around the video because keeping users together gets unwieldy and leads to issues with trolls.

APIs:

Youtube iFrame Player API: The IFrame player API lets you embed a YouTube video player on your website and control the player using JavaScript.

Youtube Data API: Used for OAuth 2.0, managing playlists and subscriptions, searching, and much more.

Software:

WebSockets are an advanced technology that makes it possible to open an interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.

COMPONENTS

HTML files

See 'Files' section for details

Stylesheets

Bootstrap's as well as our own

Python files

See 'Files' section for details

SQLite Database

See 'Database Schema' for details

FILES

template.html

login/logout button, profile button, and homepage button on a toolbar, footer with team info, pop-out sidebar with the user's favorite rooms.

home.html

Explains how to use the program and is the search page for music rooms

login.html

Form for login and creation of accounts

Room.html

Root for specific music rooms. Plays songs and displays the current song being played, has a submission form to submit music requests. Has a button to start a vote skip.

style.css

To make the website look good

General.js

Provides basic ajax calls and needed functionality

Music.js

Provides functionality to managing the music streaming on the client side

bootstrap.min.css/bootstrap.min.js

Incorporating Bootstrap

Main.py - Flask app, handles front-end navigation and operations

- **add_session(username, password)** - creates a session cookie for the current user with valid credentials or returns error message

- **logout()** - Deletes a session key of a user if one is logged in
- **root()** - provides functionality for the home.html for the root route
- **login()** - provides functionality for the /login route
- **room()**-provides functionality for the /room route

Db.py - Handles the database, including adding rooms and songs as well as user account information

login(username, password) - Logs user in if email and password are correct

- **encrypt_password(password)** - Returns SHA-256 version of password
- **create_account(username, password)** - Creates a new entry in .profiles table
- **create_room(name, passcode(null if public))** - Creates a new room entry .music_rooms table and a new playlist table

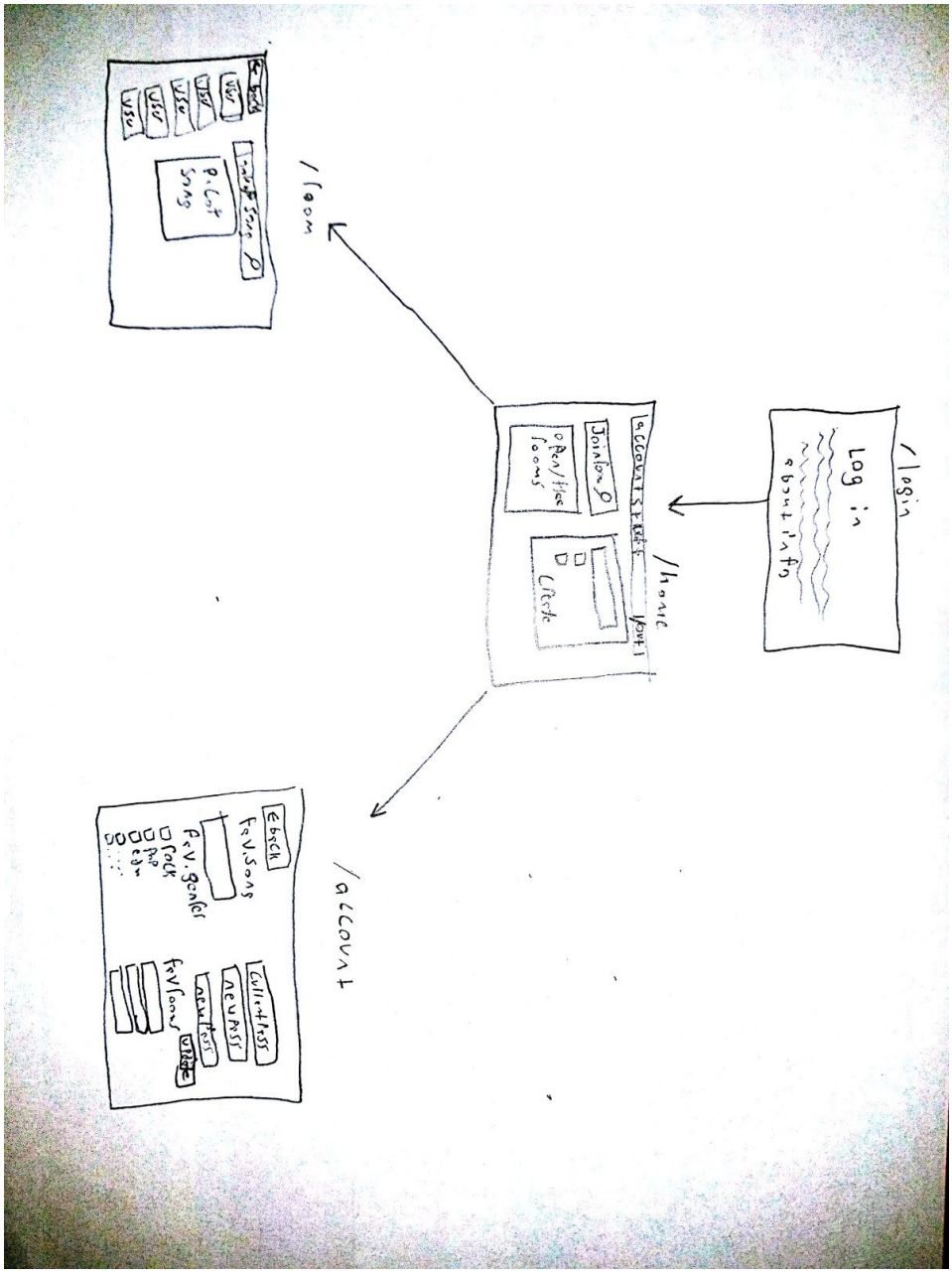
Server.py - Takes care of managing the client connections and keeping them in sync

Api.py - Communicates with the youtube API to add songs to playlists, to do Oauth, and to check validity of youtube links.

TIMELINE:

- Setting up Apache - May 19th
- Creating Playlists - May 21st
- Database and OAuth 2.0 Functionality - May 21st
- HTML Templates - May 21st
- Styling - May 25th
- Video Embedding - May 25th
- Getting video syncing working- May 28th
    - Communicating between client and server
    - Sending information from iFrame
    - Sending commands to other clients
    - Extrapolating this to many chat rooms
- Merging pieces together(lots of debugging) - June 6th
-

SITEMAP



DATABASE SCHEMA

Users

user ID	username	password*	favorites	OAuth2.0 Key
0	user	pass123	[1, 14, 12]	<key1>
1	user3	awd	[1]	<key2>
2	user1	yeet	[]	<key3>

*SHA-256 Encrypted

Rooms

room_id	room_name	password	song_ids
1	“Room 1”	“”	[0, 1, 3]
2	“Come for the party”	“We_like_to_party	[1, 2, 0]

Songs

song ID	link
0	https://www.youtube.com/watch?v=Pgd_Tgi-pbQ
1	https://www.youtube.com/watch?v=S7UAj9bl1-I
2	https://www.youtube.com/watch?v=S8gfqs1-NuE
3	https://www.youtube.com/watch?v=h-XIf-8N4eA

TASK ASSIGNMENTS

James: Video Syncing and websocket server communication
Max: Project Manager and mostly Frontend(will work where needed)
Angel: Frontend
Connie: Database and Flask