Kulpreet Chilana
Dylan Joss
Yonglin Wu

6.170: Software Studio
Fall 2014
YouTube Jukebox: Real-Time Music Playlists for Everyone

Idea

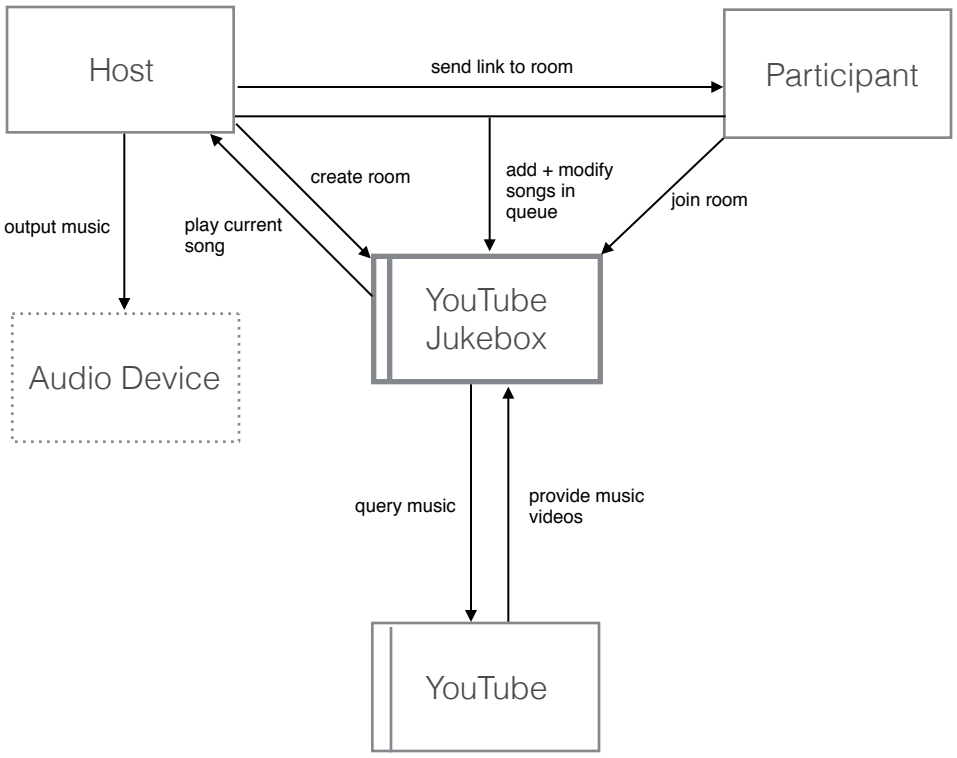
YouTube Jukebox is a web application that provides a simple way to collaboratively create music playlists in real-time using the many music videos available on YouTube. Once a piece of music finishes playing, it disappears from the playlist.

Purpose

The purposes of YouTube Jukebox are as follows:

- **Allow users in close proximity of each other to all have control of what they're listening to.** Many times we find ourselves in a situation where only one person has direct control of what music is being played (e.g. DJ at a party, driver during a car ride). This application enables multiple users to decide what music should be played. Users can view what music is playing on a playlist and what music will be played, as well as add music.
- **Allow friends to connect with each other's music tastes in real-time.** YouTube Jukebox provides a concrete way for friends to connect with each other by knowing what kind music each other is listening to when they are using the service.
- **Make it easier to utilize YouTube as a service for listening to music.** YouTube is a general platform for hosting videos, but has also become one of the most popular websites for listening to music. YouTube Jukebox aims to highlight and build a platform on top of YouTube specifically for the purpose of accessing this large library of music

Context Diagram



Note: A host is a also a participant.

Concepts

Room - a collection consisting of exactly one host, zero or more listeners, and exactly one queue. Is associated with a particular URL within the application. Representative of a group of users sitting in the same physical space.

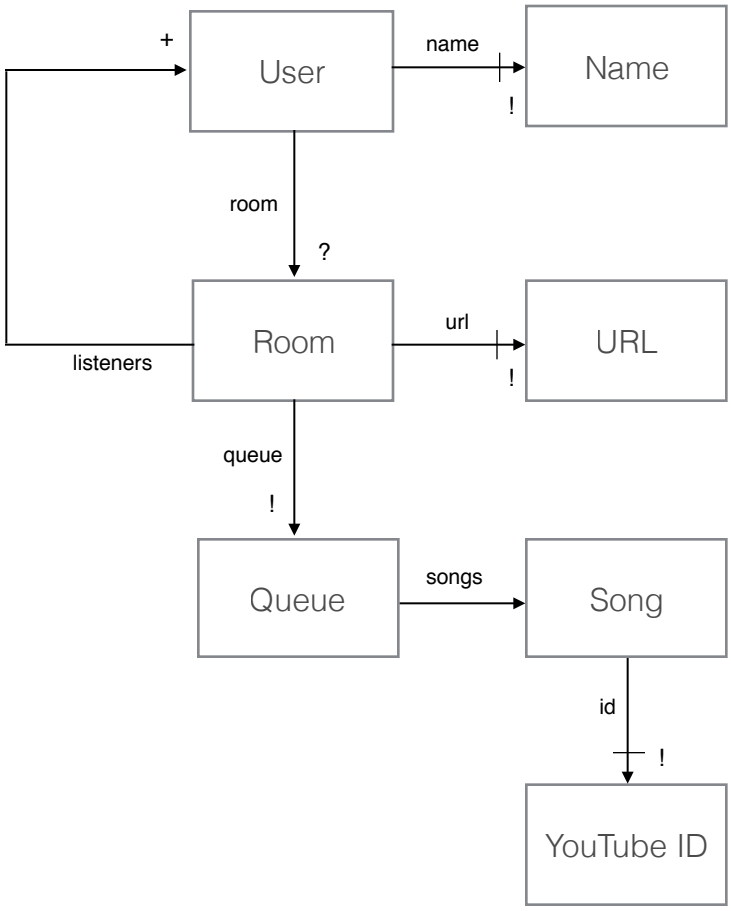
Queue - the ordered sequence of music to be played. Once a song finishes playing, it is removed from the queue. Interchangeable with ‘playlist’ throughout this document. Users can search YouTube to add music videos to the queue and it will be played on the host’s device.

Room Link - the URL to a particular room. Anyone with the link can add music to the room’s queue. The host shares the link with participants.

Host - the person who creates a room and the only one who plays the music. May view the queue, add songs to the queue, or delete songs from the queue. A host must register an account before creating a room.

Participant - anyone with the link to a room shared by a host. May view the queue, add songs to the queue, or delete songs from the queue. A participant need not register an account before joining a room.

Data Model



Design Challenges

Issue: Should access to rooms be restricted somehow? If so, how?

Possible Solutions

- A host can choose to share the link with whoever he wants. A person with the link can view and edit the queue. This has the advantage of not having to check user credentials when someone tries to access the room link. The downside is that an adversary with the link can maliciously modify the queue.
- Only specified user accounts can access the room. This has the advantage of security. The downside is the need to check credentials whenever someone tries to access the room, and requiring everyone who wants to use the service to register an account.

Resolution: we chose the first option as security is not a primary focus of this project, and the consequences of the malicious adversary having access to the room are minimal.

Issue: Should the focus of the application primarily be allowing users in the same room to create playlists or allowing users to remotely listen to the same music?

Possible Solutions

- The application is primarily for users in the same room to contribute to deciding what music to listen to. This ties in well with the "queue" concept of YouTube Jukebox, as users are listening to the same music, and the music will be taken off the queue once it is played. The downside is that users must be in the same physical proximity to take full advantage of the service.
- The application is primarily for users to remotely listen to the same music queue. This has the downside of having to deal with the complexity of users listening to different music at the same time, and the concept of music disappearing after it's been played doesn't work well with the remote focus as users do not know what others are listening to. There may be weird situations such as some music suddenly disappear while a user is still listening to it, or multiple pieces of music disappear at the same time because multiple users are listening at the same time. This is more suitable for a static playlist that doesn't change when music is played, not for a real-time music queue for YouTube Jukebox.

Resolution: we chose the first option as we thought the disadvantages of the second option conflicted with the original intent of the application: real-time, collaborative music playlists.

Issue: Which user(s) should be able to output music from their computer, and who should be able to edit the queue?

Possible Solutions

- Any participant can output music from the room, any participant can edit the queue. The advantage is uniformity. The downside is synchronization: suppose one user finishes listening to a song before another. By our model, the finished song is dropped from the queue, interrupting the second user.
- Only the host can output music from the room, any participant can edit the queue. The advantage is avoiding the synchronization issues of the above solution, and distinguishing the host from other users by giving him more application privileges.

Resolution: We chose the second option as it avoided the synchronization issues of the first, and because our application is primarily for users in the same room. It doesn't make sense to have multiple people play the same music in the same physical location.

Issue: Should we require user accounts?

Possible Solutions

- Don't require accounts at all. The advantage is that anyone can start using the service right away, and sharing among a large group of people is more effective. The downside is that the host can't use the application to share the link. The host has to send the link through text or online messengers or physically telling participants the room link.
- Require accounts for hosting/creating rooms but not for joining rooms. The advantage is that it becomes easier to associate room links with hosts (the room link can just contain the host's username), and anyone with the link can immediately join rooms. The downside is not being able to share the link within the application.
- Require accounts for hosting/creating rooms as well as for joining rooms. The advantage is consistency: an account is required before a user can perform any action on the site. The downside is the startup barrier for users getting started with the service.

Resolution: We chose the second option. We want to combine the advantages of being able to share to a large group of people at the same time and being able to identify the host and give specific privileges to the host (e.g. playing music).