

SONOS

Product Requirements Document (Version 2.0)

Team: Euphoria

Project Name: Block Party

Members:

Pedro Sosa

Mena Iskander

Miguel Delgado

Connor Shanks

Brian David Wolfe

Table of Contents

1. Introduction
 - 1.1. Potential Obstacles
 - 1.2. Functional Requirements
2. System Requirements
 - 2.1. General Overview
 - 2.2. Wireframes
3. Technical Notes
 - 3.1. UML Diagram
 - 3.2. Class Requirements
 - 3.3. RESTful API Calls
4. Use Cases
5. Test Cases
6. Reference
 - 6.1. Glossary of Terms
 - 6.2. Github

1. Introduction

Block Party is a new way of sharing music without the use of an individual app to allow for the creation of collaborative playlists.

1.1 Potential Obstacles

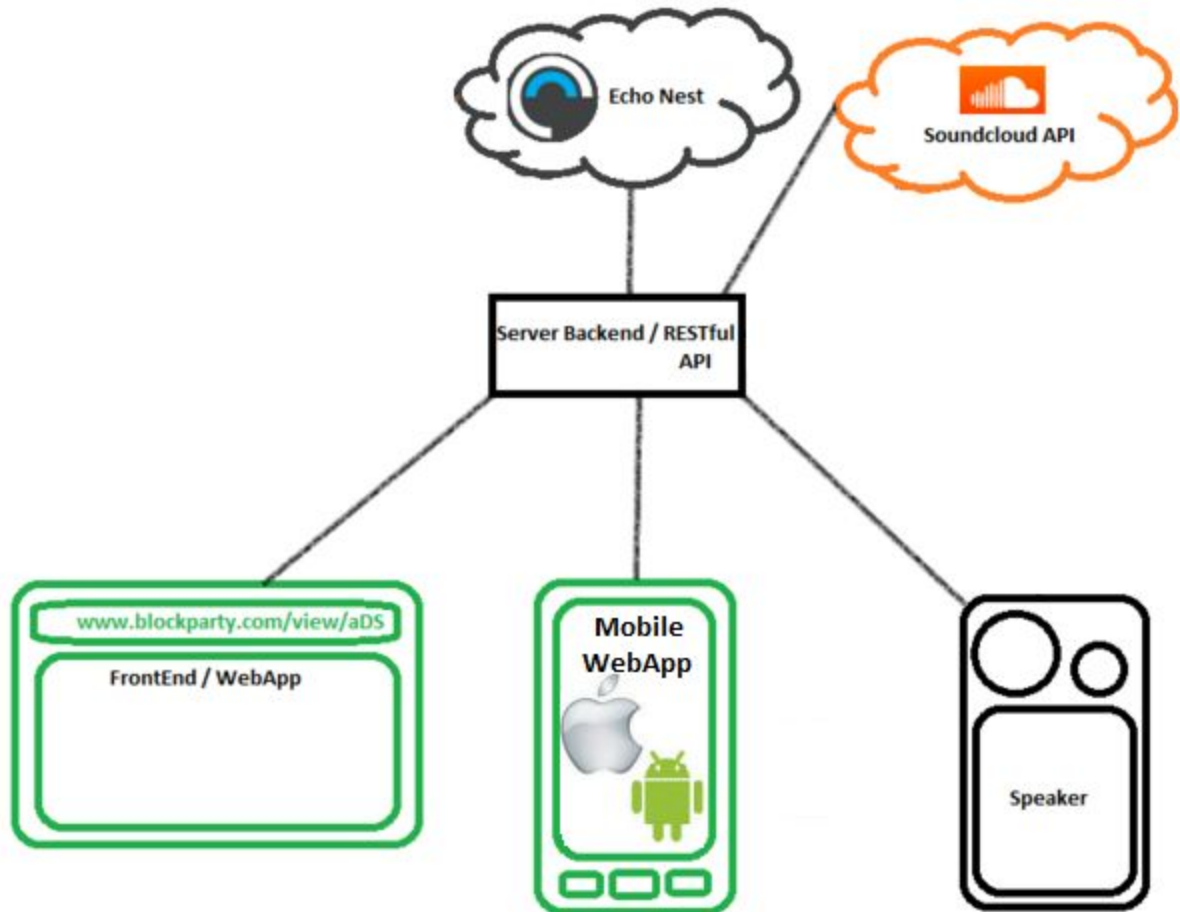
A potential obstacle would be the legal issues surrounding the use of music services' song libraries. One ideal solution would be to use music services with more lenient copyright terms such as SoundCloud. The lack of music services prevents the team from developing an application that works with a variety of platforms.

Functional Requirements:

Tasks	Priority	Time Estimates
Full Integration with SoundCloud	1	24 hours
Full Integration with Echo Nest	1	24 hours
Create a Backend server	1	72 hours
Create a web interface (front-end)	1	20 hours
Create an Android App	1	72 hours
Create a Song queue server	1	30 hours

2. System Architecture Overview:

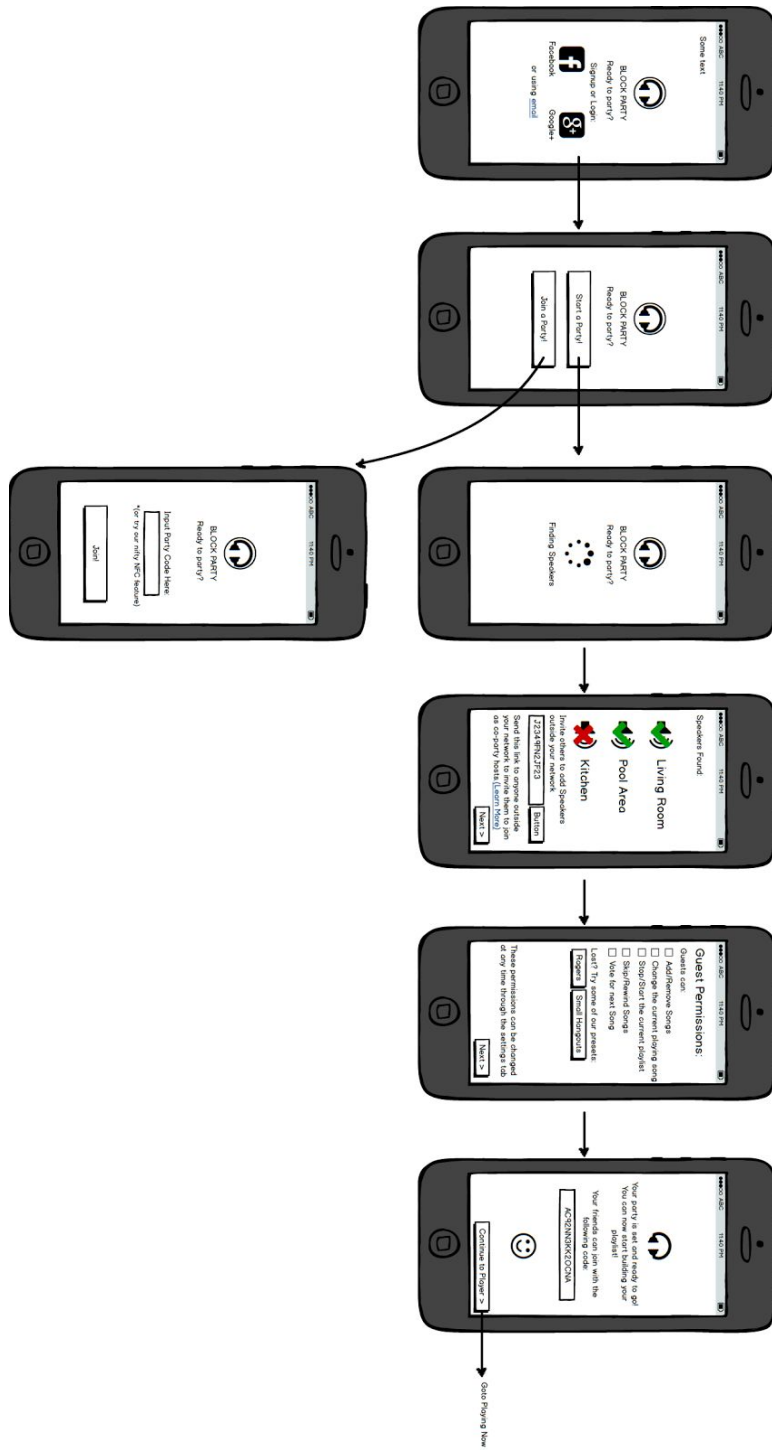
2.1 General Overview:



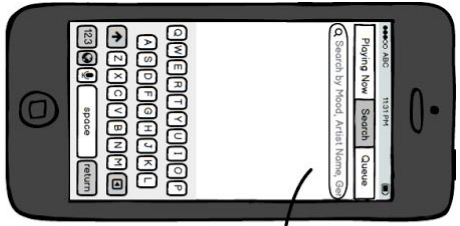
- **Server Backend / RESTful API** → *[Python Flask]* Backend will move information between the music services, front-end users, speakers, and queues.
- **Soundcloud RESTful API** → *[HTTP + Python Wrapper]* Source for music information and streaming urls for songs.
- **Front End / WebApp** → *[React]* Simple UI for clients that do not wish to download the Mobile App or do not have an android.
- **(Future Extension) Android App** → *[Java + Android Libs]* Android client application. Necessary for the host, optional for clients. Allows to search music, change the playlist, and stream music.
- **Speaker** → *[Proprietary]* The actual Sonos speaker gets the music itself from the Music Service and the playback is controlled by the Server.

2.2 Wireframes

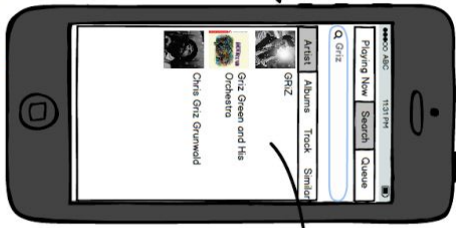
Setup Screen



Search



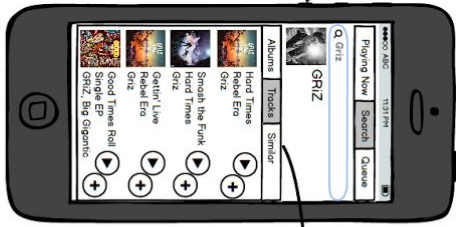
Search Function



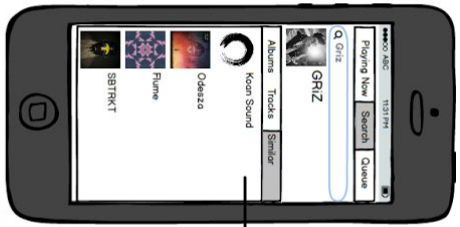
Search for Artist



Selected on Artist - Browse Albums



Selected on Artist - Browse Artist
Play Now (But don't see the playlist - just spend at current spot)
Play Next



Selected on Artists - Browse Similar Artists

Choosing any of these artists will lead you to screen #23 when you can see the artist's albums and whatnot.










Album Selected - Viewing Album Songs

Albums Similar to this Album

Playing Now

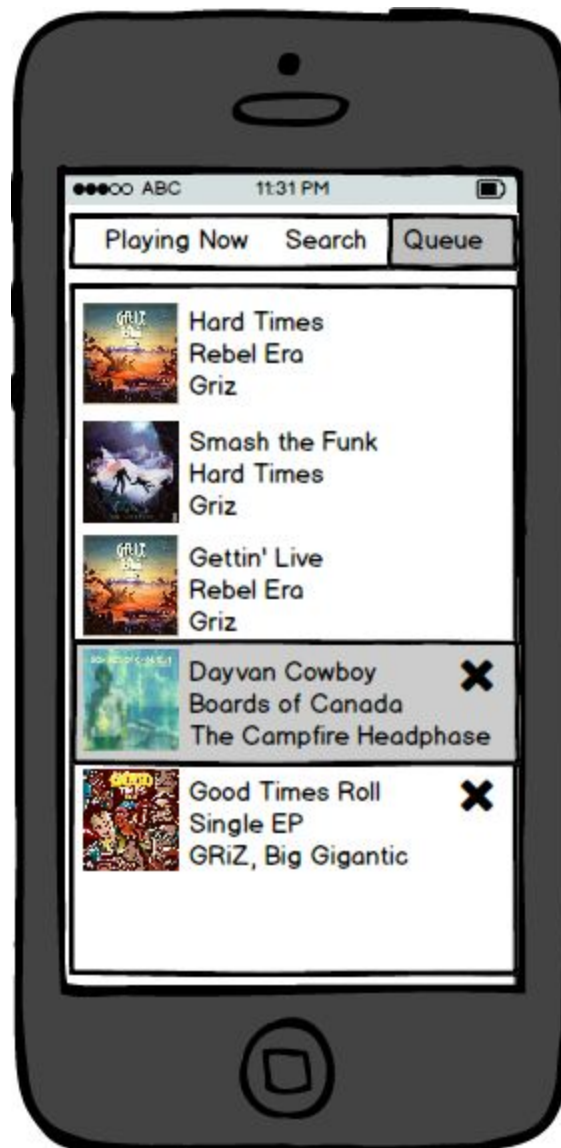


-  Thumbs Down - DownVote?
-  Thumbs Up - UpVote?
-  Music Info - Information about the song or band
-  Lyrics
-  Mode: Normal, Repeat All, Repeat One
-  Shuffle Mode
-  Previous, Play, Next

Playing Now: All the Info you want at the palm of your hand

Simple and intuitive now playing information. We want the party goes to have as much information possible about the currently playing tracks so that they do not have spend time traveling through sites or using multiple apps.

Queue

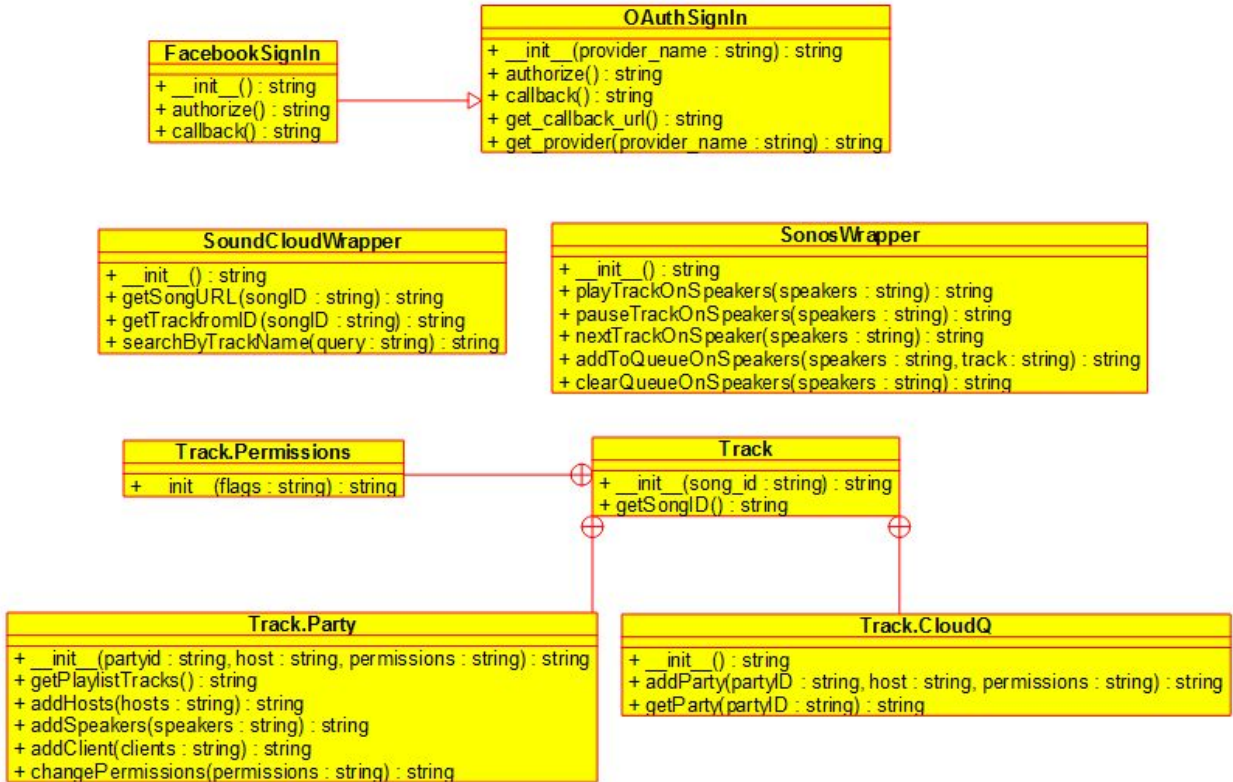


The Queue: Your party's past and future

The Queue will show the songs that have been played up to the current point in the party. Notice that unlike other music apps the past songs cannot be deleted from the Queue as they remain in the Playlist's "History". You can however choose to play them again. At the end of your party you will be able to see your party's history, save it, email it to participants or keep it to play again in the future!

3. Technical Notes

3.1 UML Diagram



3.2 Class Requirements

Tracks: Track objects contain information pertinent to a song.

<i>Member</i>	<i>Has Get/Set Method?</i>
- int ID	(<input checked="" type="checkbox"/>)
- str Title	(<input checked="" type="checkbox"/>)
- str Artist	(<input checked="" type="checkbox"/>)
- str Album	(<input checked="" type="checkbox"/>)
- str AlbumArt	(<input checked="" type="checkbox"/>)
- str Year	(<input checked="" type="checkbox"/>)
- str Genre	(<input checked="" type="checkbox"/>)
- str Service	(<input checked="" type="checkbox"/>)
- int Duration	(<input checked="" type="checkbox"/>)
- str StreamURL	(<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>)

Methods

- init(song_id, title, artist, album, year, genre, album_art, duration, service, stream_url)

Playlist: List of Tracks with a fingerprint version used to test if users have the newest playlist

- | <i>Member</i> | <i>Has Get/Set Method?</i> |
|------------------------|---|
| - int Version | (<input checked="" type="checkbox"/>) |
| - list Tracks | (<input checked="" type="checkbox"/>) |
| <i>Method</i> | |
| - init(version) | |
| - void updateVersion() | //Change the version # |

Permissions : Object that contains all the permission flags

- Static Constants*
- default_allowALL = List sets all flags to True
 - default_denyALL = List sets all flags to False

- | <i>Memeber</i> | <i>Has Get/Set Method?</i> |
|-----------------|---|
| - Playback | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| - Modify_Queue | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| - Vote | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| - Suggest | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| - PublicParty | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| <i>Methods</i> | |
| - init(flags[]) | |

Party: Object that contains the playlist and all party administrative data such as speakers, hosts, and permissions

- | <i>Member</i> | <i>Has Get/Set Method?</i> |
|---------------------------------|---|
| - int Partyid | (<input checked="" type="checkbox"/>) |
| - Hosts[] | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| - Speakers[] | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| - Clients[] | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| - Permissions | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| - Cursor | (<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>) |
| <i>Methods</i> | |
| - init(partyid,host,permission) | |
| - addTrack(track,position=-1) | |

CloudParties: List of Playlists

- | <i>Member</i> | <i>Has Get/Set Method?</i> |
|---------------|----------------------------|
| - Playlists[] | () |

Method

- *addParty(partyID,host,permissions)*
- *getParty(partyID)*

SonosWrapper: Wrapper intended to hold all the methods to communicate with the Sonos devices.

Methods

- *playTrackOnSpeakers(speakers)*
- *pauseTrackOnSpeakers(speakers)*
- *nextTrackOnSpeakers(speakers)*
- *addToQueueOnSpeakers(speakers,track)*
- *clearQueueOnSpeakers(speakers)*

SoundCloudWrapper: Wrapper intended to hold all the methods to communicate with SoundCloud

Methods

- *getSongURL(songid)*
- *getTrackfromID(songid)*
- *searchByTrackName(query)*

3.3 RESTful API Calls

Party Domain

Calls related to Party Management

- **Party/NewParty** [POST]
Parameters: Hosts, Speakers, Name, Permission
- **Party/AddSpeaker** [POST]
Parameter: PartyID, Speakers
- **Party/AddClient** [POST]
Parameter: PartyID, Client
- **Party/AddHost** [POST]
Parameter: PartyID, Host

Playback Domain

Calls related to playback Management

- **Playback/Play** [GET]
- **Playback/Pause** [GET]
- **Playback/Next** [GET]
- **Playback/Prev** [GET]

Playlist Domain

Calls related to playlist Management

- **Playlist/GetQueue** [GET]

- **Playlist/GetQueueVersion** [GET]
- **Playlist/GetQueue** [GET]
- **Playlist/ReArrangeQ** [POST]
 - Parameter: SongID, SongPosition, NewPosition
- **Playlist/Mode?mode={normal|repeat|repeatSingle|shuffle}** [GET]

Search Domain

Calls related to Music Searching

- **search?q={query}** [GET]

Sonos Play Domain

*Proxy call that redirects Sonos speakers to actual music streams
(necessary to overcome certain speaker's constraints)*

- **play/<service>/<songid>.mp3** [GET]

4. Use Cases

- As a Host user I can create a party to interact musically with the people around them
- As a Host user I can set permissions to allow other users access to different portion of the party
- As a Host user I can generate a playlist given a variety of different music services collaboratively
- As a Host user I can modify the playlist by adding/removing songs and changing the order in which they are played as well as other common music streaming options
- As a Host user I can guarantee permissions by requiring Facebook login for identify verification
- As a Host user I can control which room and which speakers the music is playing from, including all of Sonos' functionality for the music playing experience
- As a non-Host user I can view the playlist of a given party as a web app without downloading an app
- As any kind of user I can download the Euphoria app to avoid accessing a web page although not required
- As a non-Host user I can vote on the current and future music selection
- As a non-Host user I can view the current playing track and its lyrics
- As a non-Host user I can suggest songs to be played
- As a non-Host user I can download a copy of the history of music played.
- As a non-Host user I can search for almost any song.

5. Tests

Login:

- 1) Can users login via Facebook?
- 2) Can we pull Facebook data for use in our app?

Party - Host:

- 1) Can the host moderate the playlist?
 - Add / Remove song.
 - Change song order.
 - Toggle random.
 - Toggle repeat.
- 2) Can the host moderate the guest list?
 - Add / Remove users.
 - Adjust guest permissions.
- 3) Can the host control the broadcasting to speakers?
 - Assign playlist to speaker.

Party - Guest:

- 1) Can the guest join a party?
 - Can the guest join an open party?
 - Can the guest join a private party with authentication?
 - Without authentication, will the guest be denied access?
- 2) Can the guest vote on songs?
 - Is the vote counted correctly?
 - Can the vote only once?
 - Can they change their vote?
- 3) Can the guest receive the music stream?
 - Is the stream synchronized?
- 4) Can the guest interact with the playlist?
 - Can they download the current playlist?
 - Can they share the current song?-
 - Can the guest suggest songs?

6. Glossary

6.1 Glossary of Terms

- **API (Application programming interface)** - The set of routines, protocols, and tools for building software applications.
- **AWS (Amazon Web Services)** - A cloud computing platform that provides large computing capacity more quickly and cheaply. It is offered by Amazon.
- **Bootstrap** - A CSS, HTML and JS framework used by web developers to make developing for the front-end easier.
- **EchoNest** - A music intelligence and data platform used by developers to supply song metadata to their applications.
- **Flask** - A lightweb Python web framework based on Werkzeug and Jinja 2.
- **MySQL** - An open source relational database management system.
- **React** - A javascript library for building user interfaces by Facebook and Instagram. It is the V in the MVC framework (Model View Controller).
- **Sonos** - Founded in 2002, this company makes a variety of wireless audio products (“smart speakers”). Its’ product lineup includes the Play:1, Play:3, Play:5 and a couple other products.
- **SoundCloud** - Social sound platform where artists can upload their own musical creations.

6.2 Github

https://github.com/Migueld37/SONOS_Capstone/

Google Drive:

<https://drive.google.com/open?id=0B6qnjy96aIJGNnBCeThBbGUyb2M>