UC Santa Barbara

NEXT

CS 48 - Computer Science Project - Spring 19

Vision Statement

Authors:
Cristian Gomez - `c_k_c@ucsb.edu`
Edward Zhong - `ezhong@ucsb.edu`
Kerem Celik - `kerem@ucsb.edu`
Ramiro Pinto - `rpintoprieto@ucsb.edu`
Steven Booke - `stevenbooke@ucsb.edu`

April 5, 2019

Contents

1 Project Background **2**

2 Project Outcome **2**

3 Initial Project Milestones **3**

4 Planning **3**

4.1 Definition of Components 4

4.2 Communication Protocol 5

4.3 User Stories 6

1 Project Background

Hiring a DJ to perform at a venue or a party is a cost-ineffective and inefficient way of playing music. The DJ must either play what the crowd requests or in most cases will only play music that the DJ enjoys. The crowd as an aggregate knows what they want to listen to and they should be responsible for choosing music. Entrusting the venue host with choosing a DJ that will know and then play what the crowd desires is simply outdated in the age of smartphones.

Solutions today include mobile apps like TouchTunes and Rockbot. However, both of these apps only allow business like restaurants and stores to allow users to choose their songs and even require in-app purchases to actually select the next song. There does not exist a solution where users can host their own venues/events and allow other users to select songs.

Users should be able to utilize their phones as an impromptu jukebox and allow other nearby users to select and vote on the next songs for free.

2 Project Outcome

A voting-based system to determine the next song(s) to play enables the crowd to control what they listen to; the most popular song (desired by the most people) will always be played. Additionally, a skip feature can enable users to vote to skip the current song being played. Some venues may want to limit the type of music being played, e.g. a middle school dance would likely only want non-explicit songs to be played. Users of the app, not only enterprise/business partners can host their own ad-hoc "venues" where other users can connect and vote on the next few songs. Hosts can choose the threshold for number of votes required to skip the current song and/or queue the next song. In certain cases, the host can grant permission to only a handful of specific users to vote or select the next song. A scoring system will reward users for consistently suggesting songs that have been selected by voters or punish users for selecting songs that are frequently skipped, in the form of cosmetic flairs. A scoreboard will serve as a means of comparing user contributions and keep users engaged and motivated in the service. Per-venue statistics exposed to the host after an event will allow the host to get insightful data about the music played and the engagement of the audience.

3 Initial Project Milestones

Sprint 1

1. Client wire mocks
2. Basic client integration with music player
3. Basic server integration with database layer

Sprint 2

1. Basic client and server integration
2. Location aware venue search

Sprint 3

1. Song voting and skipping support
2. Scoreboard and statistics system
3. Demo preparation

4 Planning

Potential technologies:

Elasticsearch for storage and location-aware retrieval of venues

Spring for REST API for server

React for mobile/hybrid web app

Napster Developer API for music library API

4.1 Definition of Components

The project consist of 3 components:

- **Audience Client:** Users that can vote for the songs in the play-list and propose new songs. It is connected to the music API in order to search for songs to propose.
- **Host Client:** User who is has set up the venue. In addition to the audience clients functionality, it is the client that plays the songs. It is connected to the music API to play the song and search for new songs and vote.
- **Server:** The server is the connection point for the users. It stores the venue play-list and updates it according to the proposed songs and votes.

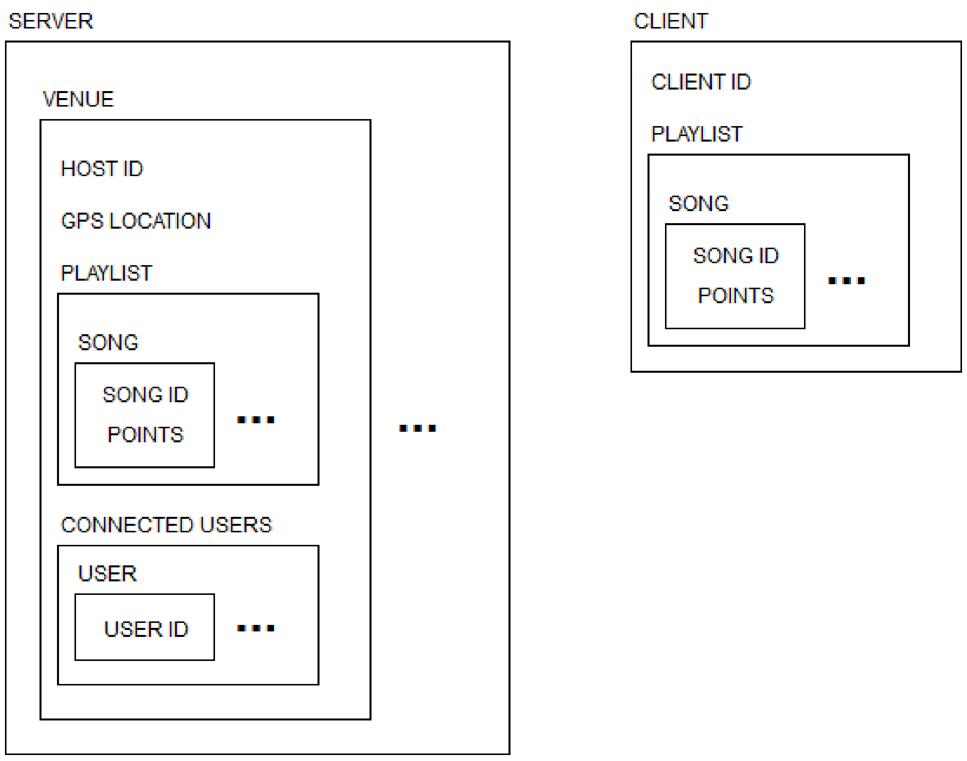

Figure 1: Components OOP scheme

4.2 Communication Protocol

1. **Initialize venue**: Host client sends GPS location and server stores client ID and creates new venue ID.
2. **Enumerate venues**: Audience client sends GPS location and receives a list of nearby venues sorted by distance and size.
3. **Send song**: Audience client sends server a song suggestion. Server stores client ID and song ID.
4. **Vote song**: Audience client sends server a vote for a song. Server stores client ID and song ID.
5. **End venue**: Host client send server request to close venue for new song requests and receive statistics

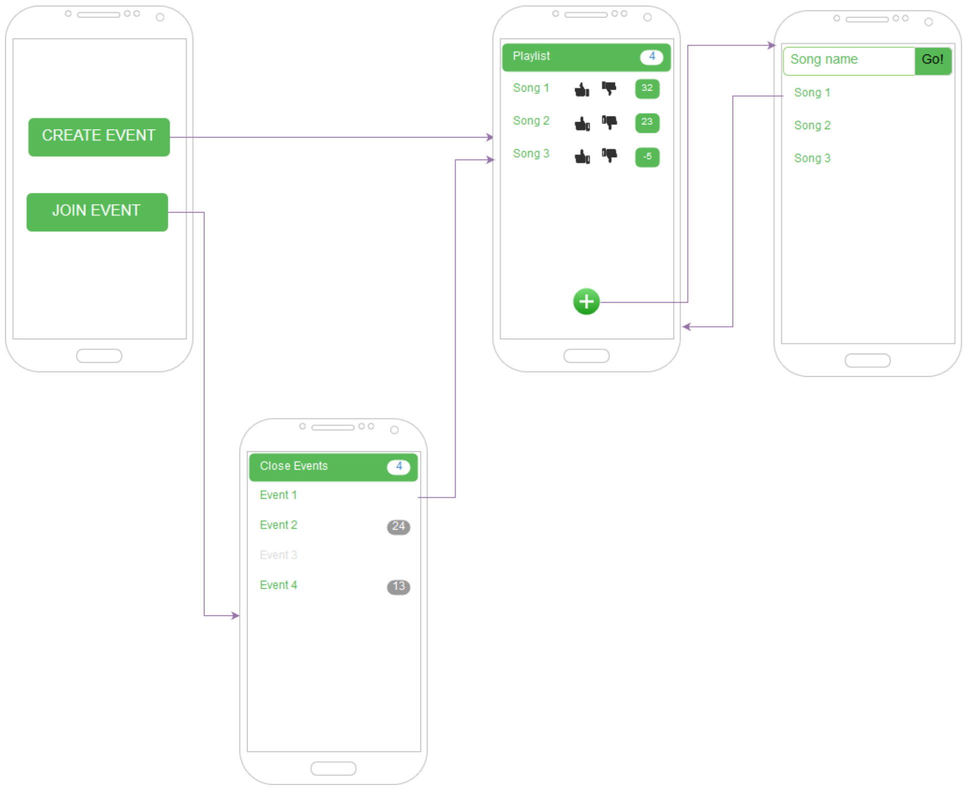

Figure 2: First draft interaction diagram

4.3 User Stories

As a host client I want to:

1. Create a venue
2. Propose songs
3. Vote for or against songs
4. Get the next song to play

As an audience client I want to:

1. Connect to a venue
2. Propose songs
3. Vote for or against songs