# Polyphony

## 333 Timeline

### Rahji Abdurehman - Anqi Dong - Tom Kelly - Ben Stallworth
### TA: Deep Ghosh

***Thursday, March 6, 2014***

Met with Prof. Kernighan and had project idea approved

***Tuesday, March 11, 2014***

Met as a group and delegated foundational tasks, especially concerning design document

***Sunday, March 15, 2014***

Finalized design document and submitted it, went away for Spring Break

***Friday, March 21, 2014***

Received iOS Development Licenses, began fooling around with Xcode

***Wednesday, March 26, 2014***

Met as a group and finalized project ideas and goals, collated list of potential names for app, delegated tasks and expectations for first week

Created sketches of expected major user interaction affordances used by the app

Created first-cut specification of API of web service, to be called upon by the iOS app

Began drafting schema of relational database tables

***Thursday, March 27, 2014***

Met with our TA, Deep, and received insights about our project and advice on how to proceed

***Sunday, March 30, 2014***

Worked on developing the Storyboard, this timeline, and setting up AWS Credentials

***Tuesday, April 1, 2014***

Met with Deep: discussed blockers with respect to AWS setup, characterized progress for iOS development, noted that we might have to deal with Heroku + Amazon RDS if we can't get Elastic Beanstalk to behave

Later, managed to resolve setup issues and start instance, and push files using git to AWS Elastic Beanstalk, but actual script isn't running properly yet

### *Sunday, April 6, 2014*

Noted that Django should have mechanisms to output JSON okay, because had heard otherwise.

Had been messing around using Flask on AWS, but finally chose Django and configured everything for Django instead on the server.

### *Monday, April 7, 2014*

Noted that Django should have mechanisms to output JSON okay, because had heard otherwise.

Had been messing around using Flask on AWS, so bit the bullet and configured everything for Django instead on the server.

Did more configuration between AWS, Django, and PostgreSQL. Was able to establish connections between the database, Django, and AWS.

Worked on designing the user interface and the structure of the iOS app, and building the design in XCode

### *Tuesday, April 8, 2014*

Some notes from our weekly meeting with Deep: Making sure the view is consistent across all devices.

Long-term: Refine streaming, talking to the server, ..

Follow HTTP Request model.. give them wrapper functions for any request the app makes

This week: Have database setup, communicate with it, have a script that can do at least one action that the app needs (Joining a party, creating a room, etc.)

Be able to join a room based on what's happening within the database (dynamically)

### *Wednesday, April 9, 2014*

Finished up most of the storyboard and (initial) app design. Expect to polish before demo.

Assigned research into audio playback and contacting server from the app

### *Friday, April 11, 2014*

Set up South (we think) on the web backend to make database schema changes (adding columns) possible

Made working code that dumps data from and to the Postgres database

### *Monday, April 14, 2014*

Developed further functionality on the iPhone app

Created an alternate Heroku app to allow additional static testing (fetching static files from AWS is too annoying)

Allowed iPhone app to communicate with Django server through AWS

Started implementing music functionality

### *Tuesday, April 15, 2014*

Loaded app onto Ben's iPhone, somewhat works

Met with Deep and discussed our progress. For next week we plan to have done:

- Make a room
- Upload a list of songs
- Join a room
- See requests
- Add/remove songs (but not totally necessary yet)
- Ability to build the iOS app on a different computer

Finished implementing code to create a room, create a new user, wrangling with session cookies

Created music playback functionality

### *Wednesday, April 16, 2014*

Implemented joining a room and managing the necessary session data

### *Thursday, April 17, 2014*

Implemented listing the rooms

### *Saturday, April 19, 2014*

Updated Git Repository settings to allow building the app on multiple computers

Edited app layout

Implemented leaving a room

Figured out how to build the app on different computers

### *Monday, April 21, 2014*

Set up local servers on all of our computers for better debugging

Made scripts to automate pushing to AWS and making a local server

Added buttons on the app for new features such as adding songs and to queue

Developed new connections between the server, database, and app

Figured out how to string search fields in the database

Modified our implementation to remove redundancies and better model our desired functionality

***Tuesday, April 22, 2014***

Today we met with Deep and discussed our progress along with hiccups we've found in development

The most important thing for us to do is debug the last 3 server calls

Install the app on all devices and make sure the view is consistent over all devices (periodic pings)

Start thinking about the demo and what we'll talk about in terms of challenges in development

Also start thinking about the final document and noting as we go what content can go in it

***Wednesday, April 23, 2014***

Implemented preliminary votes functionality

***Thursday, April 24, 2014***

Wrote Django management command to clean old rooms from database

Tried writing a cron job to run database cleanup, didn't work, introduced a race condition for a while

***Saturday, April 26, 2014***

Refactored HttpRequests into separate classes

Refined setting song pool

Improved voting faciliites, add remove vote, clarified desired API from app

Created Python testing scripts for server requests

Cleaned up logic and database constraints for adding songs to community pool

***Sunday, April 27, 2014***

Tried again to get cron jobs to work, realized that it might be impossible because Elastic Beanstalk doesn't give us the location (the virtualenv) where our app is installed

Thought through and implemented blocking additions to the song queue

### *Monday, April 28, 2014*

Fixed a logical error involving the ordering of queue entries

Noticed that add requests involving many songs are slow, will try to optimize

Worked through most of the UI issues. Found music picker controller to add songs.

Fixed the iCloud item problem

### *Tuesday, April 29, 2014*

Today we met with Deep and demoed our working-yet-lacking app

We reiterated that some features we initially discussed may not be implemented, but we need to have basic parts perfect

We realized it's time to start planning for our demo and final paper, in terms of what we show and discuss

Implemented the ability to set and get the song which is currently playing, also made listener-side improvements

### *Wednesday, April 30, 2014*

Manually tuned some database/model calls to make pool list and pool add faster for typical use cases

Turned the queue into a completely local implementation

### *Sunday, May 4, 2014*

Rewrote the queue add many method to be transactional (now the entire add operation fails if anything fails)

### *Monday, May 5, 2014*

We wrote the script for our demo, delegating who would speak on what

Made final small improvements to our app in preparation of the demo

### *Wednesday, May 7, 2014*

Cleaned up a bunch of UI stuff in preparation for the demo

Changed add many to queue behaviour (clear all) to look at the old queue, and only remove votes for a song if the number of times that song appears in the queue has increased (before, all songs in the queue were cleared of their votes, which would accidentally remove songs that had been voted for again)

Rehearsed through the demo and polished our PowerPoint slides

### *Sunday, May 11, 2014*

Touched up the UI some more:

- Added some autofocus things for creating and joining parties.
- Made song titles marquee on tap
- Added color and icons
- Made upvoting and adding to queue more responsive to user
- Overhauled DJ playback, including adding a skip button

Finished final report