ExtensionChord Documentation

Table of Contents

Overview ….....1

Process ...……...1

Requirements ……2

Top-Down Overview ……2

Login Systems ……..3

Joining and Creating Room System ……4

Search System ……...4

MusicQueue Management System ……...5

Room User Management ……..5

Framework Influence ……6

Reflection/Future Plans ……7

Overview

ExtensionChord is an Android application for creating collaborative geolocation based music playlists. Users can use the app to create "Music Rooms",that other users can see and join if they are within 0.5 km of the music room. Together the users can add songs from music provider SoundCloud to create a playlist, which can then be played off of the room creator's phone. The app also leverages a voting mechanism to allow a quorum of users to decide if a song should be skipped. The application is written in Java and uses Parse as a cloud data provider.

Process

To create ExtensionChord we followed a typical XP process with a few minor tweaks to speed up the process of implementing user stories. Our first change to the XP process was to scale down the test driven development. Our non-familiarity with Parse lead to some less efficient code when we were writing tests first for our first iteration. We ended up writing the cases that we would need to test first, but not the actual code implementation. This lead to a better code quality when implementing user stories. For integration we used Git in place of SVN due to it's ability to handle merges better than SVN. For planning we had one hour long meeting every week, if necessary this meeting would go over an hour. This allowed us to thoroughly plan out what stories will be implemented, and in some cases the design that we would use. This meeting time also allowed us to prepare for each iteration meeting and double check all deliverables together. In the last few weeks refactoring for design has been huge. Initially a lot of our code was created with specific GUIs in mind, in the last few weeks we have been de-coupling this code. Paired programming has been a staple of our XP practice for the whole process. Each week the pairs rotated and worked on different use cases. We did not have code reviews in our process but due to the structure of our application, most user stories were based upon code previously written. This allowed a more nature code review to come forth as we were able to see where code could be improved.

Requirements and Specifications

A user can:

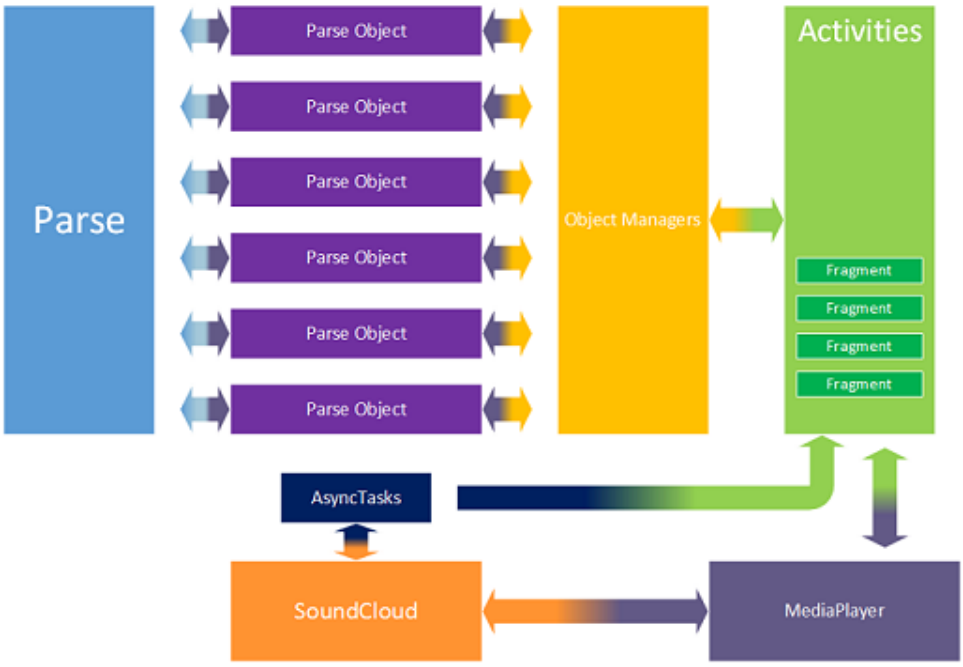
- Create an account with a unique username and password
- View all music rooms within a certain radius
- Join a music room
- See all queued songs for the music room they are in
- Contribute to the playlist by queueing up a song
- Play and pause the music
- Vote to skip a song
- Create a new music room, becoming the sole admin of the room

An admin can:

- Set password protection for the music room
- Delete songs from the queue
- See all users in the room
- See which user submitted which songs
- Boot users from the room
- Promote another user to be a fellow admin

Architecture and Design

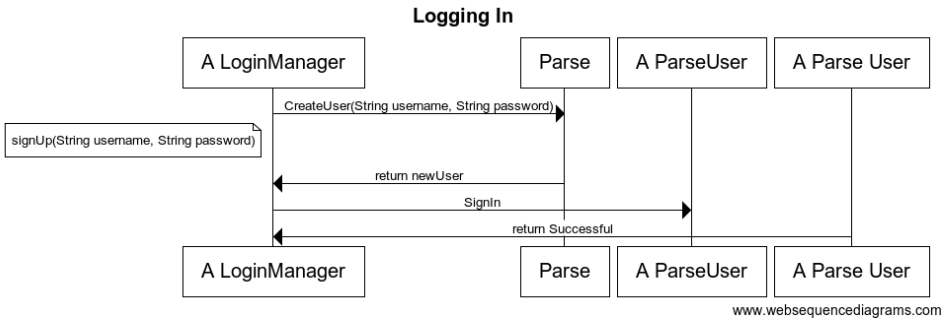
Top-Down Overview of ExtensionChord Architecture



The above figure illustrates an overview of the basic architecture of the ExtensionChord project. On the left we had the Parse backend which is our third party API layer that holds all our data. The Parse objects are the objects that represent data tables in the Parse database as well as the relationships between the objects. All the ParseObject classes have the ability to create new Parse objects, edit the them, create relationships, save, and delete the objects in the Parse Cloud. As a layer of abstraction between the GUI and the Parse objects themselves we have the Object Managers. The Object Managers such as `LoginManager` and `RoomManager` provide interfaces for creating, deleting, and editing these Parse objects without having to deal with the way in which Parse asynchronously saves its data. This prevents the GUI from hanging when changes to the data is made. On the right we have the activities which represent the GUI interfaces. Inside the activities we have the Android fragments which are also part of the GUI layer, but work as interchangeable portions of the GUI that exist during the activity lifecycle. Together these portions of the interface communicate with the Object managers , asynchronous tasks, and the Android `MediaPlayer`. The `MediaPlayer` interacts with the GUI layer to play the streaming music from the music provider SoundCloud. The GUI layer of activities also interacts with the AsyncTask layer. To avoid blocking the UI thread when interacting with SoundCloud, the activities start asynchronous tasks which interact with SoundCloud in another thread, and then the AsyncTasks use callback methods in the activities to handle the SoundCloud data. Finally, we have the SoundCloud layer, which exists solely on the SoundCloud cloud. Both the `MediaPlayer` and AsyncTasks interact with the SoundCloud layer through SoundCloud's HTTPS API.

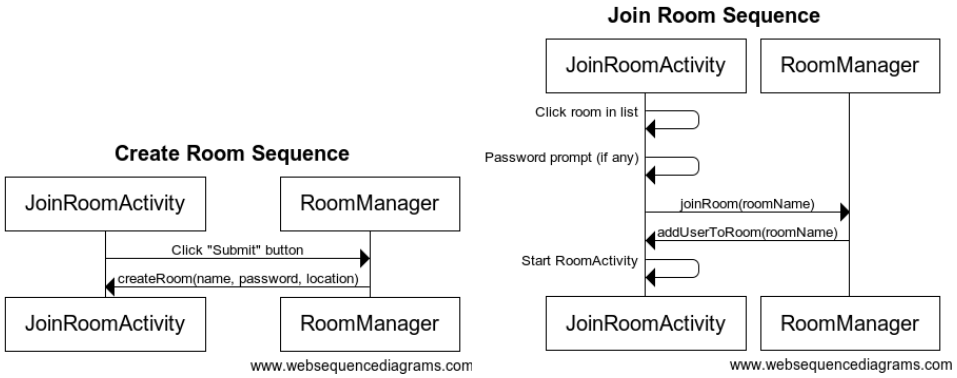
Login, Sign-up, and Logout Systems

All account activity such as logging in and out occurs within the `MainActivity` activity. When the `MainActivity` is first launched it checks to see whether the user is logged in or not. If the user is not logged in, it presents a login screen with a sign up button at the bottom. The app checks if a user is logged in by checking to see if a global variable called `ParseCurrentUser` is set to a username. If it is not, then the user is forwarded to `JoinRoomActivity` where the user can begin using the main app functionality. When the user clicks on the login button `signUpButton` is called within the `MainActivity` class, they are taken to `SignUpActivity` where they can sign up. This method shows a form where the user enters a username, password, and password confirmation which then creates an account for them that can be signed into. The user will now be sent back to the `MainActivity` where they can fill out the username and password field. This then fills the global Parse variable `ParseCurrentUser` with that username which allows the app to know the current user at all times. After logging in, the user will be forwarded to `JoinRoomActivity`, which is where they can log out at anytime. There is a logout button at the top which when clicked, will set the `ParseCurrentUser` to null and send the user back to `MainActivity`.



Joining and Creating Rooms

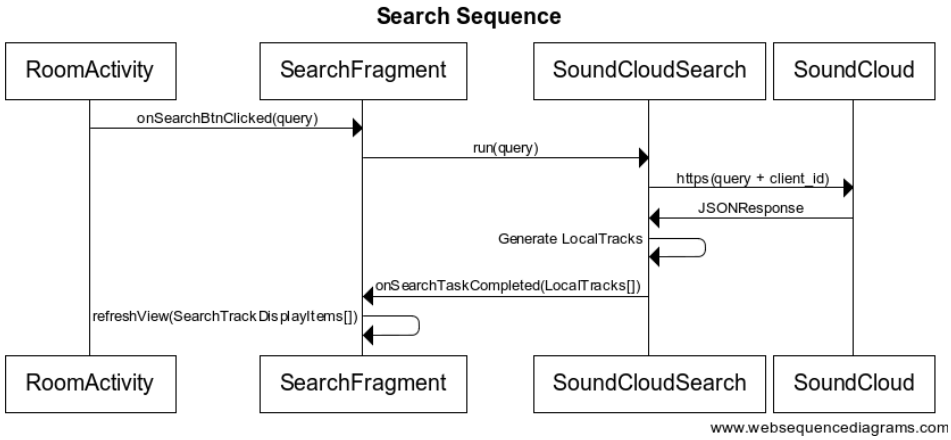
Joining and creating rooms is all done from the `JoinRoomActivity` activity. When this activity is launched, the app uses Android's location services to scan for any existing rooms within a 0.5 km radius. All rooms within this radius are displayed to the user and may be joined by tapping any of the corresponding list items. This list can be refreshed at any time clicking the "Refresh" button. Should the user wish to create their own room, they simply need to click the "Create Room" button, enter a room name, an optional password, and click "Submit" to actually create the room at the user's current location and save it on the cloud using Parse. After having done that, the list of nearby rooms may be refreshed by clicking the "Refresh" button. The newly created room may be joined in the same way as described previously. After selecting a room to join, a `RoomUser` object is created to associate the user with a `ParseRoom` and the user is forwarded to `RoomActivity` where they will be able to interact with the room.



Searching for Music

Searching for music occurs within the `RoomActivity` activity, and more specifically the `SearchFragment` that resides inside the `RoomActivity`. Together these classes represent the activity section in our top-down diagram. When the `RoomActivity` is first launched it opens up to the search interface which consists of a textbox and a search button. When the user clicks on the search button `onSearchBtnClick` is called within the `RoomActivity` class. This method starts a new `ProgressDialog` which displays to the user that data is being fetched, and calls `onSearchBtnClick` within the `SearchFragment`. This method grabs the text that the user enters and creates a new `SoundCloudSearch` object with the specified query. The `SoundCloudSearch` class extends `AsyncTask` and does all its work in a new

thread. When the work is completed, it calls a callback method inside the `SearchFragment` called `onTaskCompleted`. The `SoundCloudSearch` works by getting a specific SoundCloud HTTPS request page that is generated by the query and our unique SoundCloud Client ID. The page consists of an array of JSON objects which are then parsed, and put into `LocalTrack` objects. The `LocalTrack` objects are then passed back to the `SearchFragmen`t through the previously specified callback method. The `LocalTrack`s returned are then wrapped in `SearchTrackDisplayItem`s and displayed in the `SearchFragment`.



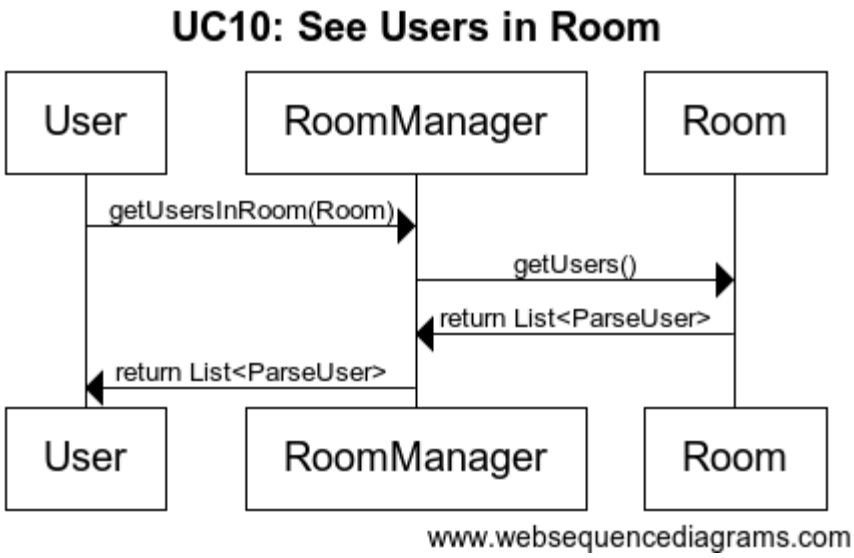
Music Queue Management

The music queue is stored within the `ParseMusicQueue` object. It is a ParseObject, as such it is stored remotely in the Parse database. The `ParseMusicQueue` is essentially a list which stores `ParseTrack` objects. These objects store the track's name, id, and various other information used for displaying information about the track to the user. The `ParseMusicQueue` queries Parse for the List of tracks to obtain the tracks. Every room always contains one `ParseMusicQueue` object. The interaction with the queue takes place primarily within the `RoomActivity` and the `ViewMusicQueue` fragment. The `RoomActivity` uses the queue within the MediaPlayer object. The MediaPlayer requires a callback to be set upon completion of each song, this is where the queue is called to obtain the next song and play it. The `ViewMusicQueue` fragment loads the `ParseMusicQueue` from Parse and displays the tracks along with their various information.

Room Users Management

Once a `RoomUser` object is created, it contains a field which lists its current room, as a `RoomUser` can only be a part of one room at a time and a boolean that indicates whether or not the user has Admin privileges in that room. This allows the app to check to see if the user is part of a room, and what other users are part of that room as well by querying the current room field. In the ViewRoomUsers fragment in the `RoomActivity`, this query is performed in order to present a list of users who are also in that room. From there additional tasks can be performed if the user is an admin by long pressing a user to bring up a context menu. The context menu allows an Admin to promote another user to Admin or to boot a user out of the

room. Once a user is booted from a room, or leaves voluntarily through the sidebar, the `RoomUser` object will be deleted.



Framework Influence

Two of the main external APIs that we used were the Soundcloud HTTPS API and the Parse API. The Parse API had the most influence over the design of our system as we used to manage both the music objects and the music rooms. Since these management actions require reaching out to a backend database over Wi-Fi or data, these methods have to be asynchronous so that they don't block our UI thread. Parse offers a convenient way to set up callbacks for these methods, but the code ended up mucking up our GUI classes. This lead to the development of the managers to create abstraction for the GUI as well as allowing it to separate itself from these asynchronous callbacks. The use of the Soundcloud HTTPS API influenced the data that we needed to store for in our `ParseMusicQueue` objects. It also influenced how we search for songs and song playback. Since the API wasn't native to Android, it required the development of more wrapper classes inside our system.

Future Plans

"Working on this project really taught me a lot of things I didn't think I needed to learn. I'm going to be interning with a financial firm this summer, and in anticipation they sent me "Clean Code" by Robert C. Martin, and I was surprised to see a lot of the same things in that book that I learned in this class. I honestly thinks the lessons I've taken from the project about teamwork and collaboration are what makes the difference between a programmer and a computer scientist."

- Lucas Pritz

"Working on this project has taught me a lot about the Android framework as well as designing a larger project. This was one of the first times working on such a large project from scratch, and actually worrying about the design of the system. I feel as though my refactoring skills have greatly increased and I have the confidence in myself to create great code with good design. I look forward to being able to take these skills and applying them to even larger project in future internships and jobs."

- Lucas Rosario

"I was surprised how much I learned and grew as a computer scientist while working on this project. I never thought I would have the chance to make a mobile app for a class so I'm very glad I was presented with this opportunity. I also used to hate writing unit tests but I've come to realize how important they really are after having broken them countless times after refactoring or redesigning. All in all, I had a blast working with Android and learning how to properly design the system with the future of the app in mind. I learned a great deal while working on this project and will look back to my experiences for guidance for years to come."

- Brett Hohler

"Prior to this class I had never developed a piece of software from scratch before. It was incredibly satisfying to see the project develop from a concept into a finished project over several iterations. I enjoyed the process we used and the project as a whole gave me valuable experience of what it's like to work in a software-development team. I feel like the experience of working on a real piece of software that I actually found interesting was more enjoyable than developing irrelevant code to perform some contrived functionality. I had previously done a lot

of work on how to design a program but had never had to convert it into a functional piece of software before, I think that finally making that transition was very educational and gave me a better grasp of the software development process as a whole."

- Evan Mackenna-Ramsay

"This project was a great learning experience for me, and allowed me to grow as a developer. It was interesting starting from nothing and developing a large project. It was the first time I had to suffer with my own mistakes in not making code maintainable enough. This lead to a lot of frustration, but in the end was a very good eye opener in how important it is to keep code maintainable for future revisions. Parse was a very interesting library to use, and was very useful in our project. It taught me quite a bit about asynchronous actions and making sure that they complete in a certain order or how to keep track of them all. It was good to get more experience with unit testing as well as it is a pivotal component in the computer science world. In the end I learned a lot from this project and managed to make something that is useful."

- Jakub Wlodarczyk

"This project was an amazing opportunity and experience for me as a mobile developer. It was amazing to take an idea, which I found very practical and valuable, and watch and help it grow into a working Android application. I got to work with many new technologies that I probably would not have otherwise, and work with many different people on one project which I had never done with before. I learned a lot when it came to looking for fixes and refactoring my code to save these troubles in the future. I see myself definitely applying what I have learned throughout my career"

-Sumant Sabada