CS 130: Software Engineering
Part A Report

Project Name:

OnAux

Team Members:
Ali Mirabzadeh (305179067), Adam Cole (004912373),
Ethan Kwan (004899710), Jake Wallin (404979154),
Joseph Clegg (704978290), Michael Jung (604598422)

Professor:
Miryung Kim

TA:
Keerti Kareti

Github URL:
https://github.com/alimz758/UCLA-CS-130---SWE---OnAux

1. Motivation

Music is inherently designed to share. Most people engage with music in some capacity, whether that be as a player, engaged listener, or passive observer. We as consumers spend hundreds of dollars to experience live music events, and that experience is so special because of that massive, collective experience. But even now, with all the innovation we have witnessed in audio streaming services and mobile devices, there is no way to truly listen to music together in a way that all present can enjoy. Sure, it is possible to make communal playlists on platforms like Spotify, but there is no easy way to share and vote on what songs should be played at any given social event.

Our vision at OnAux is to bring a platform that provides a truly collective music listening experience. There is no doubt that the music playing at an event contributes to a person's impressions and feelings that are evoked from their memories of that event. Our native web-application will allow users to create or join public listening sessions in which they can interactively request and vote on songs. OnAux creates a space for everyone to enjoy the music.

2. Feature Description and Requirements

Our product is composed of mainly two types of users: a DJ, and a set of listeners. Each side of this music-feedback interface has set of actions which is described in details below

2.1 DJ

In order to create a space for everyone to enjoy music while being able to allow users share their impressions and feelings through music we defined a role in our app as DJ. DJ is a specific type of user whose task is to create that fun and shareable music environment. DJ has a set of given tasks that would differentiate it from Listener. Below you can see DJ's tasks and actions it can make.

2.11 Create a Public Session

On the UI, there would be an option for any logged-in user to become a DJ by *Creating a Session.* By creating a session, a normal user automatically becomes a DJ where other users(Listeners) can join his/her session. The DJ has full control over the session.

2.12 Control a Session

DJ is responsible for playing music and controlling the session. S/he has two options when it comes to playing the next song:

1. Playing his own desired song by searching for the desired song through Spotify API
2. Picking from the first song of the requested list, where songs have been requested by Listeners in the session and ranked based on songs' UpVotes/DownVotes. Note that the request list would be viewable by both the DJ and Listeners. However, only the listener would make requests.

Once the DJ picks the next song from either of those options, it'd be queued and played via Spotify API.

2.13 Terminate a Session

The DJ could terminate the session as s/he pleases. There would be an option for the DJ to terminate the session. Once terminated all the Listeners would be kicked out and the session would be deleted.

2.2 Listener

By default, all users logging into OnAux are listeners. As a listener, users are able to see a list of all public sessions, see what songs are being played, and interact with DJs in an intuitive and simple way. Below we detail the possible actions a listener is able to take through OnAux:

2.21 View Session History

As a listener, users are able to see the current song playing as well as the session history of all public sessions. From this, users don't have to ask others if they know the song name or try to shazam it in a loud place - it's displayed on OnAux.

2.22 Request and Vote on Songs

As a user viewing a public session, listeners are able to send requests in to the DJ through a spotify search bar. Once displayed on the list of requests for the session, all listeners are able to upvote and downvote the song request. If the request has a higher score, the further up the list it will be displayed!

2.23 Favorite a Song

Users also have the functionality to save a song they like from a session history. Instead of having to screenshot or remember a song they like, users can save the song entry to a list on their user profile to be recorded for later.

2.24 View Liked Songs

Users can like songs in various sessions and they will all be organized in the Liked Songs page. Users will be able to remove songs from the Likes Songs list, and the Liked Songs page will be exportable to a Spotify playlist.

2.25 Exporting Likes to Spotify

After listening to songs and saving them to their profile's liked songs list, users have the option to sign-in to their personal Spotify accounts and export the songs to an "OnAux" playlist.

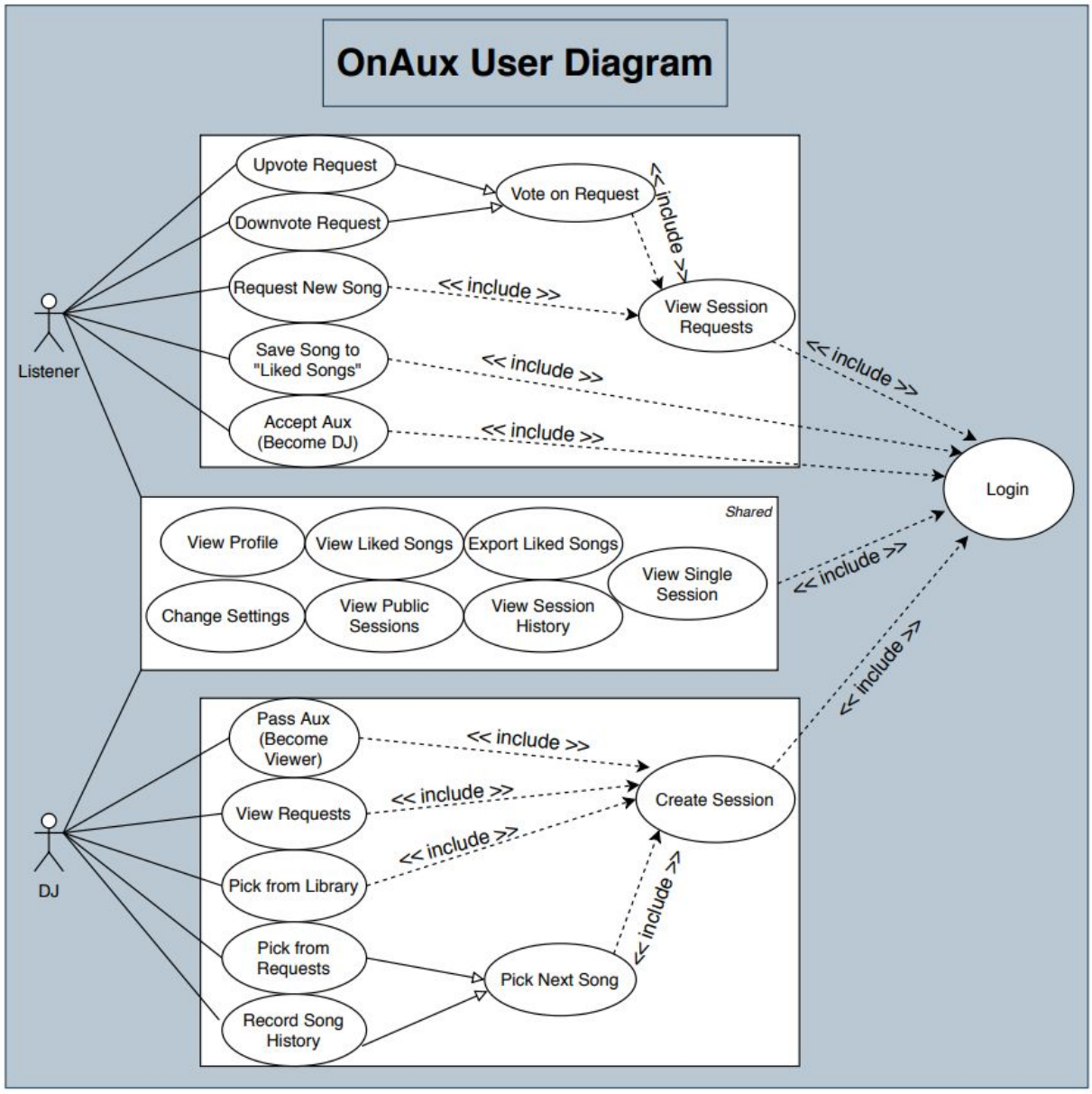
2.3 Usage Scenarios

There are a variety of social scenarios in which our application would provide utility. A popular use of our application would be social events in which there is a DJ in charge of playing music for a large gathering. In this scenario, the DJ would be able to start an OnAux session and then other users at the event would be able to join the session and request and vote on upcoming songs. The DJ would then be able to pick the next song based on the songs with the highest votes. For large gatherings such as professional sports events, the DJ may have to skip highly voted songs if the DJ deems the song inappropriate for a large public audience, so we believe there must be the ability for the host of a session to be able to disregard the request list at certain times.

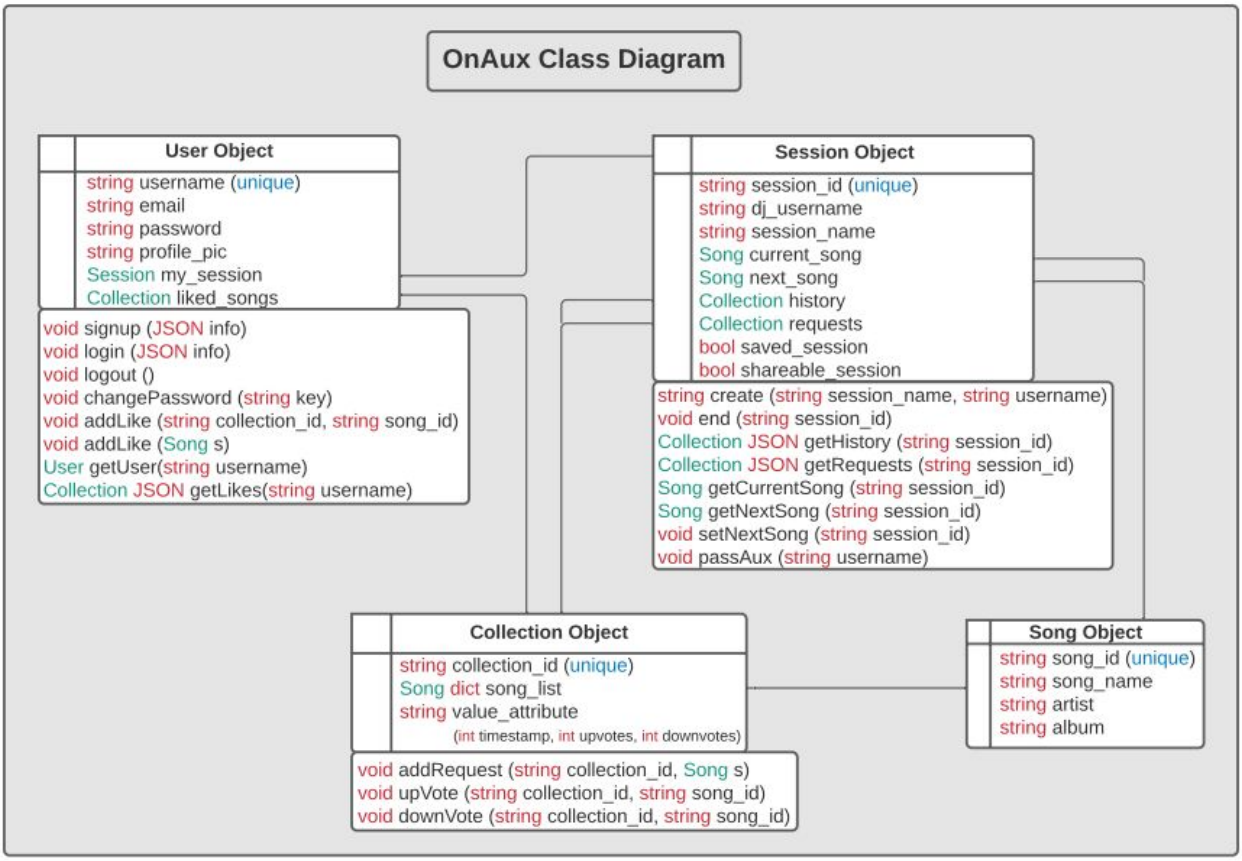
Another popular scenario would be a small group of friends all using an OnAux session to give each person the ability to directly contribute to the music being played at the gathering. In this scenario, the DJ can have special permissions removed, adding a "Community" setting to the OnAux session. For communal sessions, the highest voted song can always be played, and a tiebreaker for votes can be the earliest requested song. The creator of the session could also be able to turn off voting; in this case, all users could add songs to a queue and every song would be played in order of request time.

A third and final scenario would be that of a UCLA Student before the COVID-19 crisis. Katrina, a first year student is deciding which Fraternity to go out to one Thursday night. Half of her friends want to go to Zeta Beta Tau, and the other half want to go to Sigma Phi Epsilon. Instead of arguing and/or splitting up, Katrina pulls up OnAux and logs in. She sees that SigEp currently has a public session for their party, and so does ZBT. Katrina and her friends scroll through the history of both Aux sessions, and decide to go to SigEp since they love the music they are playing.

2.4 UML and Diagrams



Our User Diagram maps out the different relationships between Listeners and Djs in order to help map out the requirements for our platform. In our diagram we included a box around the shared requirements in the center, which include uses like exporting a liked songs list to a personal spotify account or changing settings.



Since we plan to implement our Object-Oriented structure through NoSQL Database tables in MongoDB, our class methods come in the form of an API. The API urls have been translated to readable member functions with symbolic return types, but very accurate parameters.

2.4 UI Mock-ups

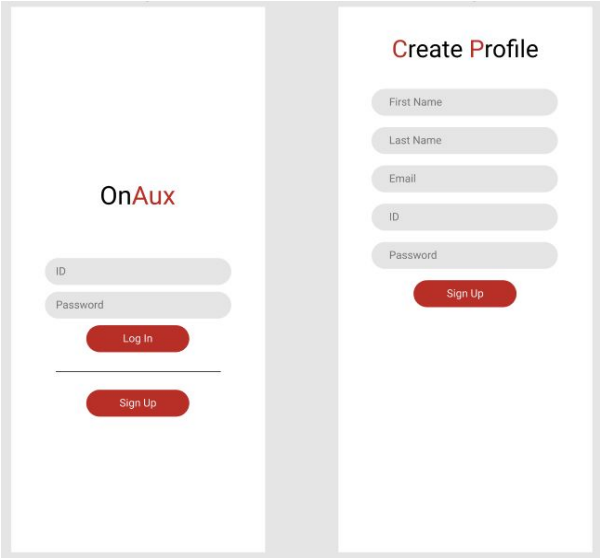


Figure 1 (left): Design for the login and user sign-up pages.

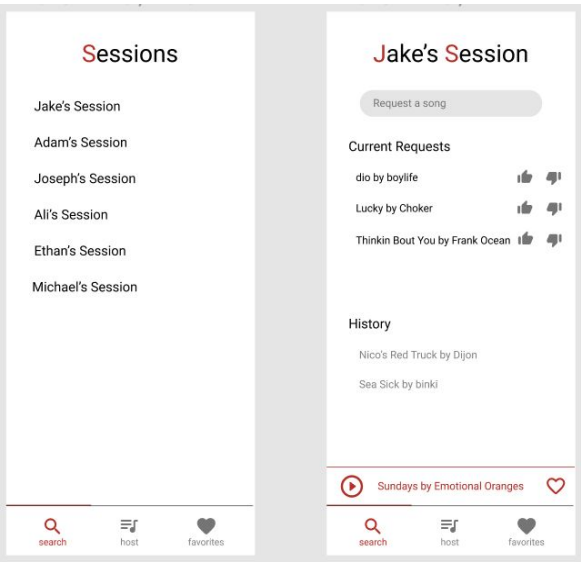


Figure 2 (right): The left screen is the design for the Session Search page where users search for an OnAux session to join. The right screen is the Session page where users search, request and vote on songs.

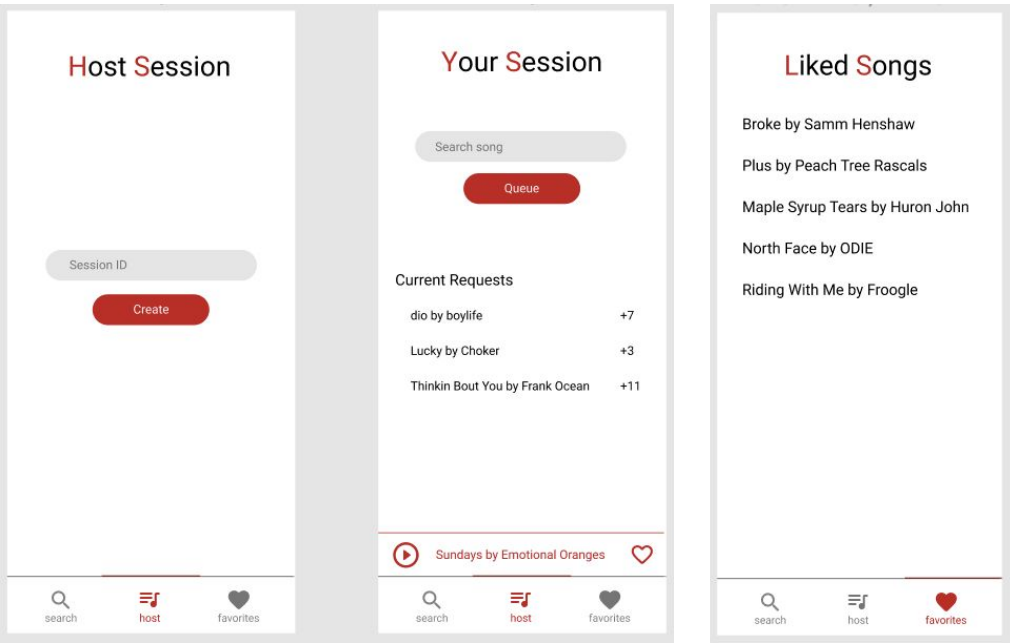


Figure 3: The left screen is the Host Session where users can create an OnAux session. After creating a session, the user would then be taken to the middle screen which is the Session page. The right screen is the Liked Songs page where users can view the songs they have favorited from OnAux sessions. This page will also allow the user to export the songs to a Spotify playlist.

Above we display some UI Mock-ups that were created by our frontend team. The first picture details the login and signup page for creating an account. Next we have a wireframe of our Discover page where listeners are able to view public sessions. The adjacent picture shows the details of Jake's Session, where the current requests are shown with the ability to upvote and downvote. Furthermore, we can see the displayed music history and the currently playing song. The bottom left picture are mockups for creating your own session and viewing the requests coming in and their scores, as well as a search bar to queue a different song. Lastly, a wireframe of a user's liked songs are shown on the favorites tab.

3. Feasibility

While there are a variety of platforms for listening to music individually, none of them offer a means of delegating song selection to a crowd; each of these platforms relies on a single user to decide which songs to queue, and that user is

essentially left to make the decision on their own accord, or by word-of-mouth. There is accordingly ample opportunity for a centralized platform to handle this functionality. Though OnAux' combination of features is unique, the underlying application relies on technologies standard to the arena of mobile development.

At its core, OnAux is a mobile application implemented in React Native communicating with a backend based in Node.js and MongoDB. The application will communicate by means of custom API requests as well as persistent socket connections, essentially the bread and butter of mobile development. OnAux will use the Spotify API for the tasks of searching for songs to request, exporting liked songs, and obtaining a track history for each session.

This API is the most specialized out of the tools we are using, though Spotify provides extensive documentation, as well as support for web applications, iOS, and Android, and modules exist specifically for the integration of API functionality into React Native. Thus it will not be difficult to integrate into our backend or frontend. Our separation of functionality into a mobile client and a backend server application will leave both parts flexible to changes in design or implementation made along the way.

4. Capability

Our team will be able to implement this project due to a diverse background of members. We have experienced members in backend and frontend development and some with full stack development. We are using MERN stack for our development as we all have experience building applications with it before. Meet the team:

Ali has done both full-stack and backend development through internships and personal projects. He has done full stack development at Accutive with Java + Spring Boot + React.js and backend at with Java + Rest.li + SQL/Hive at LinkedIn. He has made back-end services with Node.js, Express.js and SQL and NoSQL databases and so he is comfortable with working on architecting the backend services for OnAux.

Adam is an experienced computer science major that has dabbled in many different languages, industries, and classes. At UCLA, he has taken abstract computation classes like Machine Learning and Computer Vision, as well as frontend classes like Graphics. Adam also has internship experience in many

different realms of tech. Well versed in IoT, cloud computing, and AWS architecture, he also has experience in web applications. For OnAux, we plan to use React Native to create the frontend experience. Adam has developed in Ionic and Django, giving him enough experience to assist the core frontend team. Adam's primary role on the team will be in backend Architecture, structuring the backend system and the API interface for the frontend, as well as the NoSQL database system we plan to use. For the proposal, Adam created the Class Diagram as well as the Use Case Diagram.

Jake is a confident and useful developer and computer scientist. His main language is Python, as he is a machine learning and data analytics specialist. He has taken Machine Learning, Computer Vision, and Artificial Intelligence courses at UCLA, as well as doing his own personal projects in his free time. Jake is very familiar with AWS, as his internship this summer required him to build predictive models using AWS Forecast. I am also familiar with backend frameworks such as AWS RedShift and PostgreSQL, as well as the management of API usage with AWS Cloudwatch. This summer I also extensively used the MapBox API in order to develop heat maps for customer locations. I will be in charge of integrating the Spotify API into the OnAux application so that the backend and frontend can seamlessly work together and play over 1 million songs within the app.

Joseph has done mobile development on iOS, Android, as well as cross platform Flutter applications. He did mobile development at UCLA Radio for two years, implementing a chat mechanism based on persistent socket connections between mobile clients and a centralized server. He also has experience implementing client-server communication through HTTP requests to a backend API from his work on PairsQuoter, an iOS application for select tasks in finance. Furthermore, he has worked with the React framework on frontend applications at UCLA Radio and CreativeLabs, both UCLA student organizations, and while he has not written a React Native application before, the underlying design principles should transfer well. For Part A of the project, Joseph drafted UI designs and contributed to writing this report.

Michael has done mobile development on iOS and Android, making a myriad of apps from games to earthquake rescue methods. He has created a Flappy Bird clone where, instead of tapping the screen to move the sprite up and down, the user yells louder and softer. Furthermore, he has worked with a team to create an app that utilizes cellular and bluetooth connections to try and detect other devices and push a list of found devices and the locations of them to aid in

search and rescue operations. For OnAux, Michael drafted UI designs with Joseph with Figma and will help mostly with the frontend of the app, and the backend if necessary.

Ethan is a versatile programmer with a focus in Data Engineering and Machine Learning. He has built a predictive analytics model for a retail client and visualized results for an inventory optimization report. Realizing the importance of front-end development, he has supplemented his data skills by creating an interactive web-dashboard using Dash, a Python web framework, to help Blaze develop a monetization strategy for internal API access. He also has experience with databases like AWS Redshift and MongoDB. His main function at OnAux is to implement the APIs that will allow users to interface with Spotify and interactively vote on each other's song requests.