Audionyx Project Report

Group 14-5

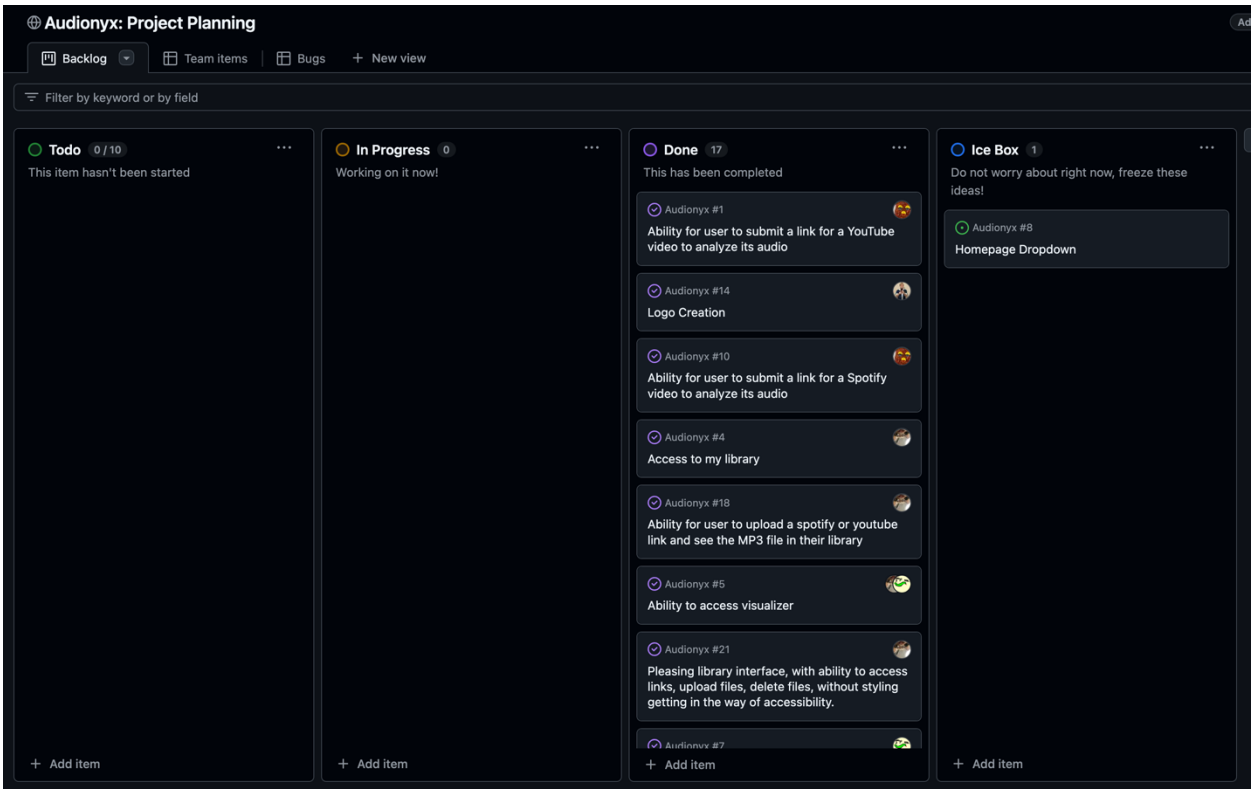
Development Team Details:

Team Members	GitHub Usernames
Abhi Chalise	Abhi6310
Agam Singh	agsi1185
Alyssa Webb	Alyssa-Webb
Connor Scott	cosc5789
Hunter Froemming	HunterFroemming

Project Description:

Audionyx is an innovative web-based audio visualizer designed to transform the way you experience music. By leveraging the powerful capabilities of Three.js and WebGL, Audionyx turns your favorite songs into interactive 3D visuals that bounce in sync with your music. One of the standout features of Audionyx is its ability to let users upload and store their music directly within the application. This storage capability ensures that users have quick and easy access to their personal music library, enhancing the overall user experience. With Audionyx, you can manage your collection, and replay your tracks anytime, all within the visualizer. As each song plays, the visualizer generates beautiful 3D animations that respond to the rhythm, tempo, and intensity of the music, creating an immersive experience for our users. Whether you're using Audionyx to relax, entertain, or simply explore your music in a new way, the platform's blend of entertaining visuals and cool music options promises to elevate your audio experience to new heights. By combining user-friendly features with unlimited music options, Audionyx stands out as a must-have tool for music enthusiasts and creative minds alike.

Project Tracker, GitHub Project Board: https://github.com/users/Alyssa-Webb/projects/4



YouTube link to our demo: https://www.youtube.com/watch?v=z-B5qZqn1U8

Github's Version Control System: https://github.com/Alyssa-Webb/Audionyx

Deployment: https://audionyx.onrender.com/login

Contributions:

Hunter Froemming: I worked on creating the Registration, Login, and Logout Pages and functionality. Along with the stylization of the mentioned pages. Along with Alyssa Webb we both worked on the stylization for every page on our sight. I also worked on modifying the index.js file to allow .css and .png files to work properly.

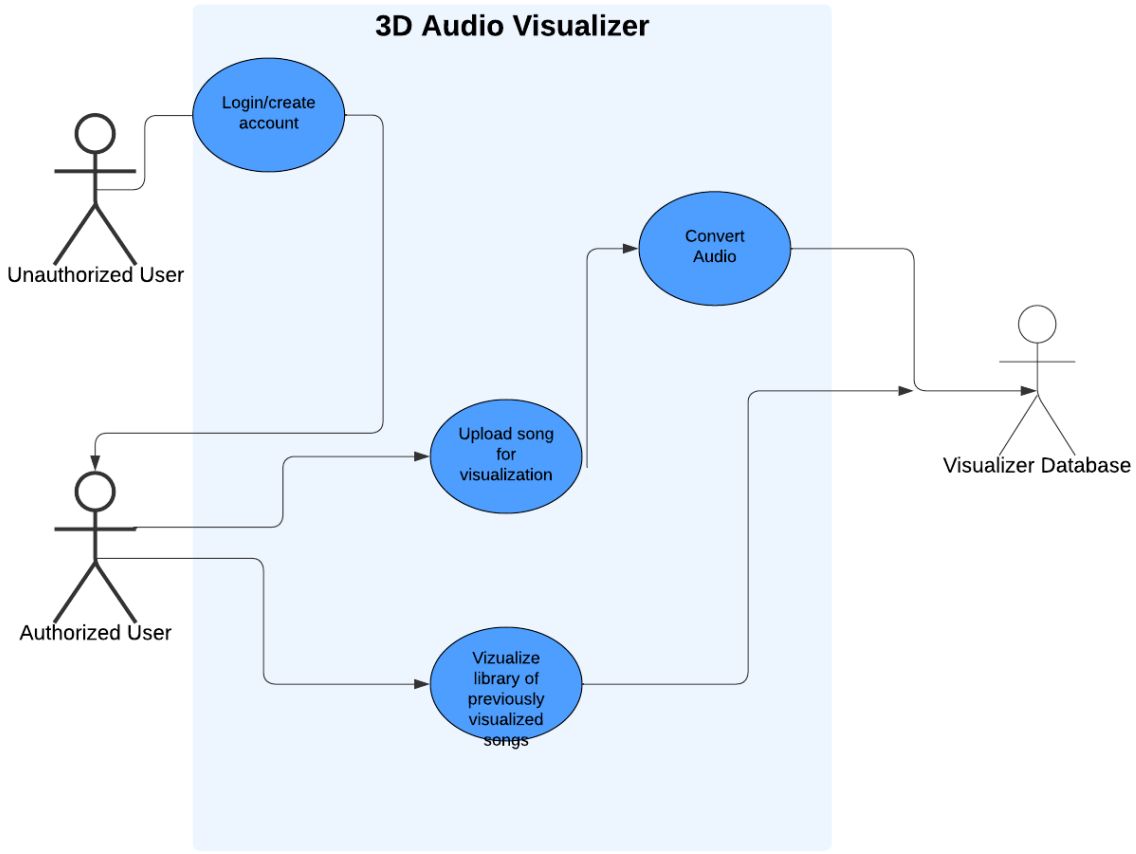
Alyssa Webb: I served as the Project Manager and Frontend Developer in development of Audionyx. My responsibilities included managing the team, ensuring tasks stayed on track, distributing work, and adhering to the project rubric. On the technical side, I focused on HTML, Handlebars, and CSS development, developing the Navigation Bar, Footer, Home, Visualizer, and My Library pages. Additionally, I collaborated with Hunter to design the Login, Register, and Logout pages. I also worked on API endpoints to make the pages interactive and implemented authentication features to ensure protection of user data and the security of our website.

Abhi Chalise: I worked as one of the backend developers responsible for creating visualization effects for Audionyx. Utilizing Three JS and WebGL rendering, I created a scene within our application that features simulated particles, and ground meshes that move and pulsate based on frequency data from user uploaded music. I also worked extensively with Agam and Connor to develop and fully integrate a decoding algorithm for base64, ensuring that it is functional with our audio buffers and allocation within WebGL.

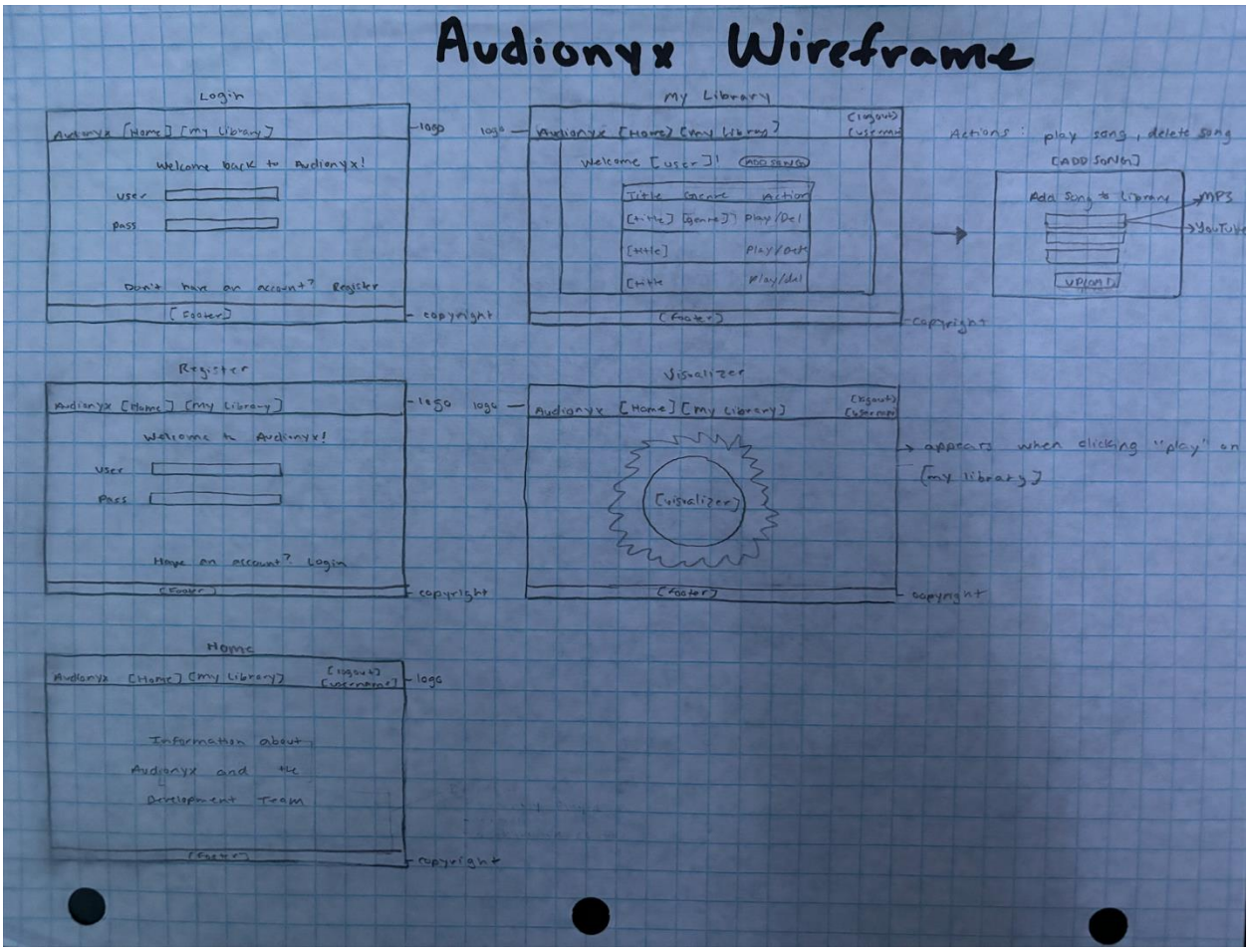
Agam Singh: I worked as the database developer, so I was responsible for designing the database, and writing queries that store everything in the database as well as the queries to pull base64 encodings from the database. I worked on writing a few routes which were used to add songs to the database for the different ways of input we accepted. I used postgres for the database. Additionally, I collaborated with Abhi and Connor to integrate the multiple parts of the projects including the converters and the visualizer to work together. Connor and I were crunched on time when it came to unexpected cloud hosting limitations when it came to integrating the converters into the project, so we decided to use mp3 files for a functional demo and prototype, which I worked on.

Connor Scott: I served as the developer for all of our backend converters. This involved converting from YouTube to base64, Spotify to base64 and mp3 to base64. I utilized the Spotify API, NodeJS, yt-dlp, handlebars and Render. I also dabbled in helping with the storage of base64 files in the database as well as modifying the front end to monitor for on clicks to use my software. Although we struggled implementing my code at the end due to the cloud hosting limitations, it was a significant amount of work, and I had to learn a lot of new software.

Use Case Diagram:


3D Audio Visualizer
Login/create account
Unauthorized User
Convert Audio
Visualizer Database
Upload song for visualization
Authorized User
Vizualize library of previously visualized songs

Wireframes:



User Acceptance Testing Plan Results:

We developed three major test cases that significantly impacted our program:

1) MP3 File Upload

   This test verified whether users could upload an MP3 file and store it in their library. The process involved an input field to correctly accept and encode the file, store it in the library, and notify the user of a successful upload. Due to the complexity of this

feature, we needed additional tests, such as tracking if the file was uploading, and to ensure the file was correctly encoded to Base64, stored, and playable. Each step included error handling for debugging, and outcomes such as file upload failures or success messages were thoroughly tested, this included successfully uploading a file, if the file was found, and additional error handling for functionality.

2) Login and Data save success

    The second test case was whether or not the user could successfully login and see their data. This was important as a continuation of the previous test case as if we want to have the user login, and have their data saved, the files they had uploaded on their account would also be saved. This was more of a check each time we logged in since we wanted to make sure all session data was saved once the user logged out. Since all of the information for each user was stored in the data table, the information was connected to their username making it simple to trace where each audio was saved.

3) Clicking the Play button

    The final test case was to see if the user could click the play button and see the visualizer output something to their screen. This test was again a continuation of the previous two tests as the user must be capable of uploading a file, seeing it in their library, and being able to play that specific file from their library. This was tested by creating a separate page for the visualizer and clicking play and seeing if anything popped up. If not, we were able to easily check the console.log to see if there were any issues regarding API endpoints and missing data.