University of Central Florida: Senior Design 1

BrainBeats V4

Design Document

Team Sponsor

Dr. Richard Leinecker

Group 11

Aidan Fleming, Martin McCarthy,

Brandon Mrgich, Aribel Ruiz

Nov 30, 2022

Table of Contents

1 Executive Summary

BrainBeats is a web-based application built around converting brain waves into music through multiple technologies, including but not limited to electroencephalograms (EEGs) and machine learning. BrainBeats seeks to be an interactive tool for musicians to create, share, and view new music through a public web-based interface. The fourth rendition of BrainBeats is built on a react framework using Express and Node.js, and has transitioned from a MongoDB relational database to a SQL database for more efficient long-term database maintenance. BrainBeats supports Bluetooth integration with OpenBCI EEG headsets and is currently only available in chrome based browsers.

The current iteration of BrainBeats (version four) aims to present an immersive user experience, which provides music feedback in real-time to creators as they process scripts, such as viewing an image or reading a note card. We aim to develop a more inviting experience for our users, which will encourage them to return to BrainBeats on a frequent basis. This will be accomplished through machine learning models trained with EEGs, a reworking of the current BrainBeats user interface to provide more accessible and appealing content to users.

Also, the current BrainBeats version aims to provide scripting reworks for more realistic and practical music generation, including but not limited to the implementation of polyphony, standardization of input and output through MIDI, and impactful modularization of the backend's infrastructure to allow for easy implementation of future scripting concepts.

The modularization of our software will allow for more accessible development in the future, saving time avoiding code refactoring, and allowing future developers to write easily importable music generation scripts. These developments will provide BrainBeats with the opportunity to branch into a much larger scale open source project.

2 Broader Impacts

BrainBeats provides a broader scope to what it means to create music, it not only allows for greater insight into the field of composition, but it creates a stepping stone for new artists. BrainBeats allows a user to pick up music creation without ever touching an instrument, this makes music creation accessible to an audience that would previously have been unable to compose music, this includes but is not limited to those who are lacking the physical capabilities to produce music. Not only this, but it creates an easily accessible method for the creation of music for those who do not know how to play an instrument. Through BrainBeats, a user may find a passion for music and learn the foundations of composition without ever having prior knowledge of instruments, melodies, or any of the building blocks that make up a song.

We find that the methods of music generation developed by BrainBeats can be used in places like hospitals to further develop an understanding of the mind. For example, an interesting application of BrainBeats is to see how it may impact studying those who suffer from mental disorders such as Aphantasia, which is a mental disorder that prevents people from creating mental visualizations. There is a form of this disorder known as auditory Aphantasia, where people cannot recreate music in their mind, even after they have heard it multiple times. BrainBeats has the potential to show how they may process music through our correlations of EEG signals and emotional responses.

Outside of impacts in medical fields, we find that our software has the potential to create a bridge between the field of composition and computer science, allowing a further understanding of how we can improve upon music through computer generation. This can pave the way for future development of machine learning that assists composers in perfecting their craft, and can hopefully help to build a larger understanding of the impacts of emotion on composing.

3 Team Intros and Ideas

3.1 Personal Intros

3.1.1 Martin McCarthy

As somebody who has always had a strong investment in music, BrainBeats really stood out to me as a project that would allow me to invest more time into something I'm passionate about while growing my skills as a developer. The uniqueness of involving the human brain and music in a project is something that will motivate me to see the project through not only to completion but to see it done to the best of my abilities. With experience in both frontend and backend development, as well as an understanding of the framework BrainBeats is built on, I felt that this would be a perfect project to take up. While playing it safe is nice, I wanted something that would help me push my boundaries as a developer and found that a project that involves studying brain frequencies is one that would help me to expand upon my abilities not only as a programmer but help grow my toolbox of applying knowledge of computer science to a different field of work.

Currently looking to pursue further academia with a focus on artificial intelligence, I found that the opportunity to develop my skills in machine learning piqued my interest. One of the aspects of the project which I found I would be highly invested in is the machine learning involved in correlating EEG waves with a person's reactions to listening to music, as this would allow me to put my knowledge of artificial intelligence into practice.

A big part of my motivation to work on BrainBeats is that it is not a startup application. Working on a project that has had multiple teams developing it in the past will help me gain an understanding of refactoring code, which is a tool I find will be incredibly useful to me in the future. Working on a project that has had previous groups create a foundation for us motivates me to develop a product that is an excellent building block for any future group of developers to be able to work with.

3.1.2 Brandon Mrgich

For as long as I remember, I've been obsessed with music and sound. How it naturally occurs in the world, and how we can structure and compose it into works of art. I'm always eager to pick up a new instrument or play with new music software to find new sounds and ways to create music.

When I was 6, my mom told me we were walking at a mall when I saw a man playing piano. She saw the excitement I had for the instrument and went on to enroll me in piano lessons. I continued throughout the years practicing on and off, learning music that I liked, and occasionally writing my own. It wasn't until high school that I produced music in software like Audacity and Garageband which were usually inspired by film and video game scores (think John Williams and Martin O'Donnell). I became more curious about the software I was using to make music, though I didn't explore this side of music until much later.

Fast forwarding to my major change to computer science at UCF, by this time I've had lots of time to explore other audio plugins, visual studio technologies, digital audio workstations, and more. I started to teach myself various Audio Engineering practices, and once again was exposed to the plethora of software involved in music production. All of them range in complexity from simple compressors to extensive sample warping like Richard James' (Aphex Twin) SampleBrain.

The worlds of music and computer science have intersected in some pretty crazy ways so far, and I think it would reach a new peak when someone can just think and feel a song within them, to then have it be interpreted and played aloud by a computer. Especially for those times when I'm working out a melody in my head whilst improvising, to then have it keep rewriting itself differently each time I try to recreate it. I hope that through our work on BrainBeats we can get a little closer to achieving this capability.

3.1.3 - Aidan Fleming

I selected this project as it gave me the opportunity to strengthen my web development skills through a project that was interesting to me. I have been involved in music in some form for the majority of my life. Ever since elementary school I started playing music on the piano and in middle school all the way through high school I was a highly involved member of the chorus. As a result of these experiences, I was instantly intrigued by the concept of this project allowing another method of rapidly prototyping new compositions while also opening up the field to those who do not necessarily know how to play an instrument. Since this method of music creation doesn't require any prerequisite knowledge, it truly opens up the field to new composers which is exciting to me as it could bring forth new perspectives.

Technologically this project is also exciting to me as it gives me the opportunity to develop a number of new skills. I do not have much experience with web development so I am excited to develop that skill set as it should help me with future projects - both personal and professional. While simultaneously making me more marketable to potential employers. I am especially excited to learn more about using a MERN stack. I am familiar with MongoDB as I have used it for some other projects but have not had a chance to use JavaScript in any project as of yet. By taking the time to learn these technologies now, I should be able to apply those technologies in my future endeavors.

Another facet of this project which is exciting to me is working with a group and a client which is an experience I haven't had before in a major way. I am excited to implement the processes of working in a group such as Agile development and learn the details that go along with it. I have done some short-term group projects but never had a chance to truly implement multiple sprints and work on a project over a long period of time. In addition to working with a team, I think it will also be very good to have experience working with a client and ensuring the project ends up fulfilling their vision and requirements. I haven't ever had to work with a client before as they have either been personal projects or clearly defined requirements for an assigned project (whether for school or work).

Finally, I am excited to present this project to potential users. I truly feel like this project will be at the very least an interesting proof of concept to show to people and potentially encourage them to think about new ways to implement similar concepts. At

best, it would become a widely adopted tool to ease the composition process and help musicians create new pieces of music more rapidly.

3.1.4 Aribel Ruiz

There is so much value in the knowledge and resources we gain from technology and when united with a form of art, it is bound to create something beautiful. Music is a form of art that I have always held close to my heart as my childhood consisted of hours at violin rehearsals, choir, and performing within a choral academy. As I grew older my personal involvement in performing and creating music faded; but my early musical background transformed into a great appreciation of music composition and its enjoyment. The idea of taking part in a project centered around translating brain waves from an EEG into music is something that immediately caught my interest conceptually and became a huge motivation for me as it is something I am proud to be a part of.

Alongside my interest in what BrainBeats does, I believe that my skills would be very useful to our sponsor's goals for version four of the project. One of the requests from our sponsor in the pitch was that they wanted a major rework to the user interface (UI) of the web application. Since attending UCF, I have grown tremendously in my frontend development skills. I have been able to use frontend development as a tool to exercise both my previous artistic and technical skills. I have been a critical frontend developer of various projects and have had a large influence on the design and execution of UI for video games, apps, and websites. I believe my acquired development skills in HTML/CSS, Javascript, and UI development could be of great use for this project. I am hoping to continue growing both creatively and technically in frontend web development as I work on BrainBeats because it is something I have grown to enjoy and wish to pursue in the future.

Another motivation of mine to work on BrainBeats v4 is that I want to expand upon pre-existing code and gain further experience reading, processing, and understanding previous versions of a product's overall code; not just the frontend. BrainBeats gives me the opportunity to gain experience in scripting, modularizing, and possible machine learning; which I am excited to be experienced in and further develop my skills as I enter the industry and pursue a job after graduation.

3.2 Personal Ideas

3.2.1 Martin McCarthy

- Regarding User Interface:
    - Inspiration for our discovery page to be based on the Spotify user album page, which features a grid formatting with an uploadable image for the cover art. This would make it easy to display user-created music in an accessible and visually appealing format.
    - Inspiration for our user creation page to be similar to that of YouTube's Creator Studio, where users have control over insights on their created music such as views and reactions to their creations, as well as useful editing features such as changing the title, description, cover art, etc. of a song.
- Regarding scripting and backend development:
    - Implement signing in with an external media player such as Spotify or Soundcloud to play tracks for users to listen to while generating music.
        - This could be useful as a music generation scenario that is highly related to how we want to build our machine learning models.
- Regarding Machine Learning:
    - Utilizing previously created datasets to have multiple options for machine learning training, allowing us to determine which approach will produce the most "music-like" outcome.
        - For example, there's an Indian research study that provides a dataset containing EEG responses from participants who were listening to a variety of different songs along with their perceived responses to said music in an attempt to correlate enjoyment of music with EEG waves.
    - Use Spotify's API to break down the components of a song we play for research participants and use said components to build an association

between the way a song is assembled and the brain processing that occurs while listening to it.

3.2.2 Brandon Mrgich

- Research into the neurology of the brain and music.
    - Rather than the current node placement on the temporal lobes and the primary auditory complex, we would also focus our attention on locations like the hippocampus and the amygdala.
        - These regions are responsible for language and short-term memory, which according to some studies are highly active in the EEG of those who are musically fluent i.e musicians and conductors.
    - Other critical regions of the brain to our application
        - We would want to focus on specifically the right temporal lobe as it is responsible for our ability to distinguish tonal sequences, such as a melody.
    - We may want to focus on these musically fluent people when gathering EEG data, and I think it would be especially important to record EEG data whilst a conductor conducts, or a musician plays.
        - This is mainly because it has been shown that there is an increase in brain activity in other regions of the brain. In those who have performed music professionally, or for a very long time, we can expect increased activity in the occipital lobe, and on EEG the theta band.
        - I would love to record some data while I play an improvised piano melody wearing the headset.
- New methods of music generation
    - Machine learning

        - There is a large amount of publicly available data containing EEGs of people at a resting state, we can attempt to perform unsupervised learning on this data. Again on our own focus group in an activated state following audio-related tasks we've assigned.
        - We can utilize the machine learning functions of the open-source library MNEFlow which has models specifically trained on EEG data.
        - Using the data from the Music Listening-Genre EEG dataset, also known as MUSIN-G, we can alter the way MIDI events are generated by the user's enjoyment of the music in an audio-related script.
    - Polyphony
        - Given the way music is currently generated in BrainBeats, a simple Polyphony can be achieved in a few ways.
            - One of the ways previously implemented was to assign an instrument per EEG channel.
            - Another simple, though boring approach would be to look at the relative min and max peaks, and apply those intervals into min and max notes creating thirds, fourths, fifths, etc. However, this would sound clunky and stiff.
            - Lastly, though not without further research and testing, would be to group strongly correlated channels into their own instrument, and affect pitch and note length through different variables provided by the data stream.
- More scripting options
    - Audio Prompts
        - Frontend design that allows a user to drag and drop audio files to use during music generation.

        - Integration to streaming service to allow the users to search for and select a section of a song as a script.
    - A user inputs the notes they're wanting to generate, and they're scored on how accurately their brainwaves are interpreted.
- UI Related to music generation
    - A piano roll to visualize the midi generated by a user's brainwaves.
    - Minimal Audio Workspace
        - Selection of different instruments
        - Selection of effects to apply (Delay, Reverb, Compression, ADSR)

3.2.3 Aidan Fleming

- Scripting
    - Revamp of user interaction with script generation (see Frontend Ideas)
    - Allow inclusion of a photo api for users to select images as opposed to only colors as exists in the current implementation
        - Unsplash looks like a promising contender for this
        - Signed in users should also be able to upload custom images and save them within their account
    - Potentially integrate various font option for the text of scripts
        - This would allow the user an additional method of influencing the person recording from the composer's standpoint
    - Scripts should be savable and allow for different users to all upload the songs they are able to record based on them
        - The idea behind this is allowing multiple people to use the same script and develop various end products to achieve the desired effect
- Backend

    - Modularization
        - This is a necessary aspect of the project as the current implementation is disorganized and complicated to update
        - As a result we will most likely have to majorly change/remake large portions of the existing backend
            - We will most likely switch to TypeScript and an object oriented approach in an effort to achieve modularization and a “drop and play” infrastructure for music generation
    - Database
        - Should provide a robust framework and should also be consistent with the current infrastructure
        - Will store interactions so that users can have metrics and likes
- Music Generation
    - Should consider implementing methods for visualization of the generated music following the composition
        - This could be done using common visualization via any number of techniques if we can generate MIDI from the files

3.2.4 Aribel Ruiz

- UI Design
    - Have a color scheme that complements our new logo which contains navy blue, aqua/teal, and pink.
    - Incorporate design practices to help meet Web Content Accessibility Guidelines (WCAG).
    - Incorporate various logo character designs that change depending on the page the user visits.
        - Various images of our logo character are generated using an image generation model called Dalle.

        - Adds a fun aspect to the user experience and highlights how different emotions and waves produced by the brain can translate into different music during the generation process.
    - Implement design inspiration from Audius and Soundcloud websites.
    - Redesign selection settings for music generation.
        - Remove the implementation of the dropdown menu and implement a listed view for selecting settings.

4 Division of Labor

4.1 Martin McCarthy

- Project Manager of BrainBeats v4
- Handled planning of meetings, accessing time management, and ensuring target goals were reached
- Member of the research team, ensured that all of the datasets found were applicable to the project as well as the machine learning models that would be practical with them.
- Developed music generation models using machine learning
- Developed the research study for converting EEG to MIDI.
- Assisted with frontend development, working on scripts to allow for functionality in the login and sign up.

4.2 Aribel Ruiz

- Head of the frontend development team. Developed a complete rework of the original frontend system, converting from JavaScript to TypeScript.
- Developed UI and UX frameworks through Figma for reference during the development of the application.
- Handled communication between the frontend and backend to ensure the smooth handling of changes in the codebase.

4.3 Brandon Mrgich

- Head of the music generation team. Developed music generation models and restructured the previous implementation of music generation.
- Member of the research team, gathered knowledge in the use of EEGs and how to properly collect and analyze them for the purposes of BrainBeats in both music generation and how to properly guide users to utilize a BCI headset.
- Handled setting up the MySQL database for use with DigitalOcean hosting and a react application. This includes the development and management of Prisma schemas.
- Assisted with frontend development, working on navigation and making search calls to the API.
- Developed the research study for converting EEG to MIDI.

4.4 Aidan Fleming

- Head of the backend development team. Handled the development of a complete rework of the original backend system, converting the API from JavaScript to TypeScript.
- Assisted with the development and handling of JSON Web Tokens, allowing for secure user data.
- Assisted with frontend development, working on organization and styling.
- Assisted the frontend team with understanding how to make calls to the newly developed API.
- Created visualizations of the backend in order to create an understanding of object relationships.

5 Legal, Ethical, and Privacy Issues

BrainBeats has to consider the rights for songs created using BrainBeats software, this is especially relevant when handling our machine learning models, for example, if the model generates a MIDI file that strongly compares to one that is produced under a copyright there is a potential for lawsuits if the user decides to use it for personal profit. BrainBeats will have a system where anything created by a user can be used in music and other audio-related content, where it is then up to them whether that track is publicly on the BrainBeats site or privately listed in their own collection. Tracks publicly listed on the site would be free reign for anyone to use. BrainBeats will not reupload anybody else's generated music for any profitable purpose in order to ensure we do not infringe upon any copyright laws while also protecting the rights of our users.

When performing a research study in order to collect data for the development of our machine learning datasets, we must be cautious of the ethical issues involved. It is important that none of our data is influenced by our own biases in order to prevent collecting faulty data. We must also be aware that there are legal issues when conducting research, our data must be handled with respect to how our sponsor requests, and we must provide acknowledgment of our sponsorship regarding the research conducted. If we use any other datasets in the training of our models, we must also be aware of the copyrights involved in doing so, likewise, we need to make sure our research wouldn't be in violation of another study's copyright.

The dataset utilized for the first part of our machine learning model, known as the DEAP dataset, involves the use of an End License User Agreement, commonly known as an EULA. An EULA is a legal contract between the provider of the dataset and us, the user, which specifies the restrictions and rules of what we may do when handling the data provided to us. The DEAP dataset's EULA states that commercial use of the dataset is strictly prohibited, therefore a future rendition of BrainBeats which seeks to develop a commercial product must revise the machine learning models used for training with different datasets. The dataset is strictly prohibited from being distributed, because of this we must ensure that we have secure methods of producing our product as an open-sourced project while maintaining the security of the dataset. Likewise, any publication of our research must explicitly use data from consensual participants as referenced in the original DEAP dataset, and must be a "small portion" as described in the EULA. We are required to cite the use of the DEAP research paper

as a result of the public nature of our documentation of BrainBeats, and it is seen as such in the references section at the bottom of our document. The EULA provides that the developers of the DEAP dataset may alter the EULA at any time, and thus any future renditions of this project that want to continue training with the DEAP dataset must maintain awareness of this flexibility.

We must ensure that we follow the wishes of our sponsor in regard to our project being open-sourced. Because we are developing an open-sourced application, we must ensure that we do not leak any crucial information regarding our software that can expose the data of our users, this means that we must ensure we develop secure systems for storing our data such as encryption and multi-factor authentication. Likewise, we must make sure that any of the software that we implement is open-sourced as well as to not expose any of the implemented data or software that is not meant to be accessed by the public, this includes but is not limited to API use and datasets.

In the development of artificial intelligence, it is always important to understand the ethical implications of what is created. While there is much controversy in the creation of artificial intelligence, we believe that there is no ethical implication in the algorithms that we may create. We on the BrainBeats team merely want to develop machine learning models that can assist composers in creating melodies that they can then input into their own software and expand upon, rather than take over their role in the development of music as a whole. With this in mind, we are making sure to work alongside the music department at UCF, keeping the discussion open with them on their perspective of our development of machine learning models.

When developing a future machine learning model for BrainBeats, we found that it would be practical to store EEG data from users in order to be fed to future training models, this would require user agreement as well as an understanding of the legal and ethical issues that come with retaining this data. We must be aware that EEG data leaking could potentially result in serious implications, where the use of this data could be taken and used for malicious purposes.

6 Budgeting

When analyzing the cost of products for our project, many options are presented. We as a team decided on the use of certain technologies in comparison to others as a result of numerous features, this includes but is not limited to cost-effectiveness, overall utility, and necessity. An overview of the costs associated with BrainBeats is listed below along with a breakdown of each section respectively. The consistent improvement of OpenBCI technology and the desire to continuously improve the BrainBeats application means that the values of products described in this section will not stay relevant in the long-term scope of the project. With this in mind, we describe many of the resources we've utilized as an estimation.

6.1 Cost Estimation

- Research Studies - $150
- Hardware - $999
    - OpenBCI 8 Channel Cyton Board - $999
- DigitalOcean Droplet - $14 per Month
- Spectra 360 High conductivity electrode gel - $5 per bottle

6.2 Budget Analysis

In this subsection we provide an analysis of each of the estimated costs provided above. Products which cannot be predictably accounted for have been excluded from this documentation, this includes resources such as those provided to us by the senior design laboratory. In regards to research studies, to create our own dataset for the training of machine learning models, we need focused EEG samples from people performing different auditory tasks and actions. Because of this, there is a need to acquire subjects to participate in a research study. The incentive to recruit people for our study is a $10 gift card, which results in a final price of $150 spread amongst 15 total participants. In the case that there needs to be additional data collected, we would seek out an unpaid method such as members of the team developing the remainder of the dataset. To develop more accurate EEG readings from our users, we determined that an upgraded cyton board would be necessary. The cyton board is

capable of gathering 8 channels of input, twice our current board's capacity. Because our server is moving to handle more users and a higher influx of computational data, we found that it was essential to upgrade our hosting server on DigitalOcean. The new Droplet runs at the price of $14 per month, over the course of our project's development this totals out to be $140. During the testing of our application, we found the need to use one tube of electrode gel which ran at about $5 per bottle based on the supplier bought from. Overall this totaled out to a price tag of $1294.

7 Milestones

Tasks	Date Due
Develop a good understanding of brain neurology and music generation	10/14/2022
Develop a good understanding of the way that EEG operates in application to our project	10/17/2022
Have GitHub organization setup	10/21/2022
Have figma wireframes fully developed for use in our website UI	10/21/2022
Have use case diagram done	10/21/2022
Have enough participants selected for our research study	10/28/2022
Begin modularizing and refactoring code	11/01/2022
Data collection of EEG done	11/04/2022
Begin redoing UI framework	11/05/2022
Setup of DigitalOcean Droplet complete	11/12/2022
Begin reworking the backend	11/20/2022
Machine learning model created	11/30/2022
Codebase modularized and refactored	12/01/2022

Start training machine learning model	12/05/2022
Submit design document for review	12/05/2022
UI framework redone for a guest account	12/05/2022
Backend structure finalized	12/05/2022
Have familiarity with OpenBCI Headset	12/05/2022
Begin reworking of API Endpoints	12/07/2022
END OF SENIOR DESIGN 1	12/07/2022
Finish development of API Endpoints	12/14/2022
Begin documentation of API Endpoints	12/14/2022
Finish incorporation of API Endpoints with frontend	12/29/2022
Finish documentation of API Endpoints	01/07/2023
Begin adding new features to the UI for music generation	01/13/2023
Testing OpenBCI headset with new music generation	01/15/2023
Finalize script manipulation feature	01/20/2023
Finalize of music generation	02/17/2023
Present music generation module to sponsor	02/18/2023

Have sponsor add their own script to the server for manipulation	02/21/2023
Begin trial presentations	02/24/2023
Sponsor Demo	03/15/2023
Music generation using machine learning complete	03/26/2023
Begin unit testing	03/27/2023
Unit testing fully finished	04/16/2023
Final trial presentation before committee	04/18/2023
Final demonstration for committee	04/19/2023
END OF SENIOR DESIGN 2	05/02/2023

8 Requirements and Stretch Goals

For the development of the BrainBeats, we have set goals in respect to our sponsor Richard Leinecker's wishes. We came upon these goals through open discussion and an understanding of our capabilities under the timeframe we have to develop the application. We have separated this section into four sections for clarity, frontend development, backend development, machine learning development, and music generation.

8.1 Frontend Development

This section of the project refers to design and development of the design and the functionality of the web application focusing on user interaction and accessibility. This includes but is not limited to user registration and login, as well as core features of the application such as selection of settings for music generation, recording, and uploading tracks.

8.1.1 Requirements

- Creating good practices for the frontend's codebase.
    - Provide clear commentary throughout the original JavaScript code while converting to TypeScript in order to make it easier to understand and build upon for future developers.
    - Refactorization of the current frontend codebase with a primary focus on modularization in order to allow for easier integration of new code and reduce the number of duplicate code blocks.
    - Restructure the user interface in order to remove current bugs within the codebase as well as provide good documentation for bug prevention in the future.
        - Redevelopment of the basic and advanced settings features for music generation in order to resolve an issue where a user changing settings resets the music generation process.
- Create an overhaul of the user interface's look.

- Develop a UI that is clean and professional in order to appeal to a wider user base as well as show off the effort and care for the production of our product.
    - Create an appealing home screen that catches the user's eye and expresses what the BrainBeats product is about when the user first visits the website.
    - Implement a rework of the signup and login screens so that users feel more inclined to register and return to BrainBeats for future use.
    - Develop a rework of the settings selection page for music generation in order to make it more user-friendly.
    - Rework a user profile in order to display the tracks that they have created.
        - This includes the selection of track image covers that can be uploaded to the database, they will default to the BrainBeats logo otherwise.
- Rework the music generation studio screen to implement the modularization of the scripting service.
    - This will involve the removal of the choice to select a YouTube video as a script option for music generation.
    - This includes reworking the cards users can choose to select prior to music generation.
- Allow new features for cards prior to music generation, specifically the addition of pictures rather than just colors and text.
    - Find a solution to the number of space images take up in the database, this should be done through URI encoding.

- Allow a script to be inserted into the software to be manipulated by the user.

8.1.2 Stretch Goals

- Allow the uploading of canvases to a track.
    - Users can upload a canvas to be the display image of a track when tracks are displayed in a list format.
        - This will allow for a much easier view of tracks and a nicer visual display of the track list with all the various different track covers.
- Create a new studio view when the user presses record.
    - Display a piano roll when the program is translating the user's brainwaves into MIDI notes.
- Sort public recordings.
    - Allow the user to sort public recordings on the home page using different various sort settings selected from a dropdown menu.
        - Sorting by popularity, calculated based on the number of interactions a track has within a recent time frame.
        - Sorting by the most liked tracks, this can be separated into most liked tracks within a specific time frame such as monthly, weekly, or yearly.
        - Sorting by recently played tracks, this is based upon the most recently played tracks by the entire user base rather than the user that sorts.
- Allow the user to edit recorded songs that they have saved onto their profile.
- Add confirmation to delete a track from the user's profile.

8.2 Backend Development

This section of the project refers to the database and API endpoints that will be used to run the application and serve as the bridge between the frontend and music-generation/scripting sections. This section will be primarily overseen by Aidan but will be contributed to by the entire group as necessary.

8.2.1 Requirements

- Implement a redesign of the original backend codebase, primarily focusing on modularization and code refactoring.
- Defining clear criteria for style and implementation of code, ensuring that all new code conforms to consistent requirements with clear and concise commentary allowing future renditions of the project to be able to understand what each block of code implements.
- Redefinition of Prisma schemas to work with new features in the frontend of the software.
    - Creating a script which allows users to upload images from their local computers to be stored in the database.
    - Giving users relationships to tracks stored in the database.
        - This is essential for new features in the frontend such as liking tracks and creating playlists.
- Allow storage for up to 200,000 uploaded images in the system for users to reference and utilize when scripting.
- Create an infrastructure to allow users to edit the order and timing of images in the script.
- Save created scripts in the database so that a user can access and replay them at a later date.
- Allow the user to include audio files or prescription sound effects to the script as they are created.

8.2.2 Stretch Goals

- Develop "drop-in" modules which allow for new scripts to be implemented allowing music generation without having to reconstruct the codebase.
- Allow for user selection of images to be uploaded in the script selection from an online image database.

- Developing import and export features for scripts so that users can share them amongst each other.
- Allow the addition of shapes and other aspects to the scripting tools outside of simply images, color, and text.
- Automatic backups of the database to the digital ocean droplet.
- Create a separate test database using the Prisma schemas.
- Create bulk test cases with the test database.

8.3 Machine Learning Development

This section of the project refers to the development and execution of machine learning models. This will require extensive research into the development of datasets and how to effectively develop a model that provides us with useful MIDI. When setting out to develop machine learning models for music generation we are aware that it is an ambitious task as our team does not have a strong background in the analysis of music through machine learning. Our model aims to generate MIDI based on an input of EEGs, because of this we will need to develop a model that has an understanding of both MIDI and EEGs.

8.3.1 Requirements

- Create a machine learning model which will analyze EEG waves and return music to the user in real-time, allowing for immersive and more realistic music development.
- Gather EEG data while participants listen to music, this recorded data will be used to train machine learning models.
    - This will be important for associating specific EEG readings with specific instruments, feelings, and frequencies in music.
- Utilize Python to apply neural networks to electroencephalograms.

8.3.2 Stretch Goals

- Conduct a study where we record EEG data while participants are performing music, ideally with various instruments such as a piano, guitar, etc. We will also record the music performed to attempt to associate specific nodes with specific sounds.
    - This will allow us to generate new research where readings are coming in while in a state of performing an activity, something our users will actively be participating in when creating with BrainBeats.
- Use this study to develop a machine learning model that finds correlations between EEG readings and the creation of music to provide our users with more realistic audio feedback when recording with BrainBeats.
- Development of a neural network that converts MIDI directly into EEG using the conducted dataset.

8.4 Music Generation

The music generation portion of this project refers to how BrainBeats should handle the input of brainwaves and the output of MIDI events. This includes the instruments chosen to represent the EEG data, and how many channels we consider.

8.4.1 Requirements

- Polyphony
    - Users should be able to generate music with more than one instrument i.e piano and cello.
    - This requirement was changed partway into the project because of the complexity of the machine learning approach to music generation. It was agreed to instead focus on making the generation monophonic first, allowing for future groups of BrainBeats to expand to polyphony.
- A more complex generation algorithm
    - A machine learning approach, defined above in 8.4

    - More guided research is required
        - Work alongside faculty at UCF to gain insight into the best approaches for music generation, this includes both the Artificial Intelligence research department and the Composition department.
- Play hook for real-time play
- Plugin Discovery
    - Incoming data from the headset, outgoing data (MIDI)
- Standardized input
- Standardized output
    - A midi note at a given time which would contain pitch and length and possibly velocity

8.4.2 Stretch Goals

- After recording a song you should be provided two links:
    - Link to the newly recorded song
    - Link to edit the newly recorded song
- Collaborative and Paired play
    - Users should be able to join a session and create music together as different instruments.
    - Users should be able to make an addition to someone else's publicly listed track, something to reference would be TikTok’s feature that allows users to duet with each other.

9 Research

This section of our documentation helps to provide an in-depth understanding of the subsects of our application development through a deep dive into the important background knowledge necessary to understand technologies.

9.1 Development Stack

When deciding on the implementation of a development stack to implement there are a lot of choices, the previous implementations of BrainBeats implement a MERN stack. With this in mind, we focused our research primarily on this specific stack along with the well documented LAMP stack because of its use of MySQL and Apache, which we implement in the newest iteration of BrainBeats with respect to our sponsor's wishes.

9.1.1 LAMP Stack

As one of the first open-source development stacks, the LAMP stack is known to be stable, simple, and powerful. According to Tedium, the traditional LAMP stack has been used for popular web applications such as WordPress, Wikipedia, Facebook, and Slack (Smith). Although the term LAMP stack was introduced in 1998, it is still found as a common way to develop new web applications today.

According to IBM, LAMP is a stack of four different technologies used to create websites and website applications. These four technologies include Linux for the operating system, Apache for the web server, MySQL for the database, and PHP as the programming language.

Using Linux as the operating system allows for more flexibility and configuration as opposed to other operating systems. Apache is used as the web server which processes and manages requests and responses from the client system. Apache is known to be very reliable and is used as the web server across various websites on the internet today. MySQL stores the web application data in a format easily queried with the SQL language and is useful when running large and complex sites. Finally, PHP is an open-source scripting language and it is used alongside Apache to create dynamic web pages with dynamic processes, such as pulling data out of a database. Using PHP allows for more efficient programming.

As for LAMP architecture, Linux is at the lowest level, followed by an Apache server and MySQL database in the next layer. PHP sits inside Apache as PHP works with Apache in creating web pages.

9.1.2 MERN Stack

The MERN stack follows a three-tier architectural pattern involving a frontend display tier, an application tier, and a database tier ("What is The MERN stack?"). This development stack was created by Facebook in 2013 to make the development process easier and smoother ("MERN Stack: Explained"). MERN is widely popular today and can be found being used at big companies today including Facebook, Instagram, Discord, and Pinterest. A reason the MERN stack is very modern and still being used at these large companies is that all of its components are open-source and constantly updated. The individual technologies that make up the MERN stack also have great community support, making it a great reason to be chosen as a development stack.

The four technologies that make up the MERN stack are MongoDB for the database, Express for the web server, React for the framework, and Node. Within MERN's 3-tier architecture, React is used for the frontend tier, Express and Node are used for the server tier, and MongoDB is used as the database tier.

React is the framework in charge of building the web interface of the web application through simple components that are connected to the backend server and rendered as HTML. Express is a server-side framework that runs within a Node server. Therefore, both the Express and Node components are considered the server tier. Finally, MongoDB is used for the database which processes and stores JSON documents to be retrieved later on by the application within the database tier.

Although JavaScript is most commonly used when working with all the individual components of the MERN stack, MERN is flexible with its frontend framework. Other programming languages such as TypeScript can also be used with the MERN stack instead of JavaScript.

9.2 Frontend Services

The nature of our project's development resulted in a lot of research in regards to libraries implemented by previous iterations of BrainBeats. Here we expand upon this

newfound knowledge and its potential application to our software as well as possibilities for improvement. Likewise, as we are in an ongoing process of development, we also elaborate on the implementation of new libraries and our discoveries in regards to them.

9.2.1 React

The original BrainBeats codebase is built using the React library. For this reason, we find it important to develop a strong basis of knowledge in regards to its implementation. BrainBeats will continue to be built on the React framework because of its flexibility with application development primarily due to its compatibility with Node.js.

Originally released in 2013 by software engineer Jordan Walke at Facebook, React is a free and open-source JavaScript library used for building interactive user interfaces for modern applications. React allows you the ability to create reusable and isolated pieces of code called components that are imported throughout your application (“Tutorial: Intro to React”). Other imported libraries can also be used alongside React to create various responsive user interfaces for an application.

Although commonly used with JavaScript, React can also be used with other syntax and languages such as JavaScript XML (JSX) and TypeScript. JSX, an extension of JavaScript to also stimulate HTML, is widely used with React. TypeScript supports JSX and correctly models patterns used in React’s codebase; therefore, TypeScript can also be supported with React projects (“Documentation - React”).

9.2.2 Bootstrap and React Bootstrap

Bootstrap is an open source HTML, CSS, and JavaScript framework that allows us to implement design templates for aspects of our project such as navigation, user registration forms, grid layouts, and much more. Bootstrap is highly convenient for our project as it allows for quick implementation of CSS that would take up time in our development process. Because we are developing using React, there are limitations to using the Bootstrap Javascript

React Bootstrap is a popular frontend framework that provides pre-built components used for the user interface of an application. It originally replaced Bootstrap JavaScript to work with React and has since continued to grow and develop alongside React

(“Bootstrap”). Due to its compatibility with React, pre-defined style sheets, and default accessibility, React Bootstrap allows for quick and efficient development of a user interface in the React framework that BrainBeats implements where native bootstrap would not be relevant.

9.2.3 Font Awesome

Font Awesome is an icon library and toolkit used by various designers and developers for developing and designing user interfaces. Although Font Awesome provides thousands of free and open-source icons; users can also pay an additional fee for a Pro-level subscription plan that provides access to thousands more icons and various different styles of icons.

Font Awesome offers support for various libraries, including React, and can be imported within React applications by importing the Font Awesome package into the application’s project and using a Font Awesome React component (“Setup With React”). Font Awesome icons can also be used within desktop applications such as Figma, Illustrator, or document slides. The use of Font Awesome icons within desktop applications makes it simple to convert frontend designs into a reality within an application's user face as the icons will remain consistent.

9.2.4 TypeScript

TypeScript is a programming language that is built upon JavaScript in order to make it strongly typed. This allows us to develop our project for scalability, and since the previous rendition of BrainBeats is developed in JSX, implementing TypeScript is highly practical. TypeScript offers a type system which enforces the assignment of types in order to prevent malpractice in development. There are multiple ways to define a type in TypeScript, for example when defining a string

var helloString = ‘hello world’

TypeScript will naturally infer that it is of type string, but a string can also be defined by providing the :TypeName syntax offered by TypeScript, for example:

var helloString: string

will inform TypeScript that this is a string type. TypeScript also provides an incredibly powerful tool with the ability to compose your own variable types; there are two

popular ways to do such a thing, unions and generics. Unions are useful for defining a set of values that a specific value is allowed to be, for example:

```
type PositiveOddNumbersLessThanEight = 1 | 3 | 5 | 7;
```

will only allow for integers that are odd, positive, and less than eight. Since our project is designed to be implemented for scalability in the future, this would be useful for defining values for accepted EEG nodes, in case developers working on a future rendition of BrainBeats decide to work with OpenBCI headsets that have compatibility with more nodes. Generics provide variables to types, this is very useful for arrays in JavaScript, which can contain anything in them without checking to make sure values that enter them are of the same type. For example, if we want an array of strictly strings we will define it as:

```
type ArrayOfStrings = Array<string>;
```

In comparison to JavaScript, we find that TypeScript's object oriented approach allows us to improve upon the features of JavaScript and develop structured code which is clear and concise. We believe that this makes much more sense for a project that is going to be inherited by future developers who do not have an understanding of the codebase. Objects that are strictly typed will clearly show what they do in our code and highlight their objectives when reading them, this allows us to not only write better documentation but clear things up in our code without having to write an excessive commentary on them.

9.2.5 TypeScript vs JavaScript

The current version of BrainBeats reached a point in our development where we had to ask ourselves the question: Which would be better for our purposes, TypeScript or JavaScript? This subsection of our research describes the analysis we reached when comparing TypeScript to JavaScript.

TypeScript is an open-source programming language that was developed in 2012 by Microsoft to handle complex and large-scale applications that JavaScript could no longer handle efficiently (Krishnan). TypeScript was developed as JavaScript with added features; so JavaScript code and libraries are also valid in TypeScript. Many differences remain between the two languages regardless of the similarities in use and function.

TypeScript follows an object-oriented programming language structure that supports classes, interfaces, and inheritance which provides greater modularization of a project as opposed to JavaScript. TypeScript also follows the use of types within code.

The strong use of types within TypeScript makes debugging and catching errors during compile time much more efficient for large-scale applications; improving project productivity and efficiency over JavaScript. Typescript also allows easier maintenance for future developers.

A benefit to JavaScript over TypeScript is that JavaScript has been an integral part of creating web applications longer than TypeScript has. Although TypeScript's community support is still growing, JavaScript already has huge community support. There are many more resources for creating web applications or solving issues in JavaScript than there are in TypeScript. Another difference between both languages is that JavaScript can be used and interpreted directly by browsers as opposed to TypeScript, which must first be converted to JavaScript code to be interpreted by a browser.

As mentioned previously, TypeScript is a superset of Javascript; meaning that TypeScript is JavaScript with added features. So alongside the differences between both languages, the languages are very similar in code. Example code taken from Hackr.io's "TypeScript vs JavaScript: Which is Best in 2022" can be found below that shows the same piece of code in TypeScript (Figure 7.1) and JavaScript (Figure 7.2).

```
function multiply (a, b) {
return a*b;
}
var result = multiply(a, b);
console.log('The answer is - ' +
result);
```

Figure 9.1 Example Code in TypeScript

```
<script>
function multiply (a, b) {
return a*b;
}
var result = multiply(a, b);
document.write ('The answer is
- ' + result);
</script>
```

Figure 9.2 Example Code in JavaScript

9.2.6 Previous Frontend Services

For the frontend services of BrainBeats version 3, the previous team compared React, Angular and Vue to develop the frontend of their project. The version 3 team ultimately decided that ReactJS was best for their project. React allows for separate and independent components that can be reused and updated within future versions of BrainBeats.

A significant reason for choosing React over Angular and Vue was that it had a great open-source community with lots of resources. React was also useful for the BrainBeats web application because everything is embedded in JSX. The use and benefits of React DOM made using React a more reliable and efficient way to implement the frontend of the application.

9.2.7 Updated Frontend Services

After further investigation, the version four team of BrainBeats has decided that React was the perfect choice for developing the frontend. Not only does React allow for better modularization through components, but it also allows for better updates in the future and is used in many modern-day applications at big tech companies.

The largest difference the BrainBeats version four team will be implementing differs from Version 3's frontend is that we will be using React with TypeScript instead of JavaScript for better modularization. The version four team will be rewriting and

updating the frontend of the previous version's code and as the previous code is refactored for better organization, it will also be converted to TypeScript.

BrainBeats version four will also be incorporating more outside libraries into the project for better efficiency and organization. The two main libraries that we will be implementing into the frontend are React Bootstrap and Font Awesome Icons. Using React Bootstrap will improve the responsiveness, accessibility, and ease of the user interface. Font Awesome's extensive library will allow for improved designs on the frontend and will allow for consistency between any mockups and actual implementations of the web application.

As one of the largest requirements for version four of BrainBeats was to redo the entire frontend of the application, we hope that these updates to the frontend services will assist in the modernization of the user interface and the frontend. The BrainBeats version four team believes that adding these frontend services alongside pre-existing services will encourage improved code for future versions of BrainBeats to understand and expand upon.

9.3 Backend Services

While the existing BrainBeats application had an established backend it had a number of limitations and ways it could be improved. One major aspect that could be improved was modularization of the software as well as ensuring it was compatible with modern software. In addition to this, some of the existing infrastructure was not working due to depreciated tools and remaining bugs. As a result of all these aspects, we made the decision to transition to a TypeScript backend as well as transitioning all the tools to be compatible.

The decision to fully transition the backend to a new framework may seem like a strange decision considering a functional backend already existed but during the process of reading through the existing codebase much of the methodology was quite specific to the implementation and left little room for expansion or adaptation. TypeScript was an obvious choice for a number of reasons but foremost was its object-oriented nature as opposed to JavaScript's scripting style as well as being able to specify variables further than JavaScript's limited typing options. By transitioning to TypeScript it will give us more control over the infrastructure of the program and allow for modularity to be built into the backend software.

9.4 Unit Testing

Testing is an important part of any project to ensure a level of quality and prevent timely debugging later on during future development. TypeScript is conveniently able to be tested using many common JavaScript frameworks so a large variety of testing frameworks exist. In addition to this, it is also necessary to test the API aspect of the project which can be achieved using a separate tool that we will have to choose as well.

By doing the appropriate research and making an appropriate decision early on in regards to our testing framework. This will make it much easier to implement our testing in a consistent style and prevent the loss of valuable time and resources should we have to transition on to a different framework partway through due to a lack of planning.

9.4.1 Jest for JavaScript Testing

The majority of the application will be JavaScript based so, therefore, we needed a very robust and established testing framework to ensure that all aspects of our project will work as expected. Jest provides an excellent solution to this problem. The largest advantage Jest offers is its wide range of compatible tools including TypeScript, Node, and React all of which we will be using extensively over the course of our project. In addition to this, Jest also easily allows for integration with very little post install configuration outside of codebase specific aspects. When you do need to add features the tool provides the tools you need such as very simple mocking frameworks which will allow for simulation of receiving EEG data or user input from the website without having to rely on the server or other physical inputs for testing. This tool also is excellent at providing analytics for your project with integrated code coverage reports as well as the necessary framework for generating rich exceptions to ease the debugging process as much as possible.

9.4.2 Postman for API Testing

While Jest certainly covers the majority of our testing requirements, we still needed a framework for unit testing the API calls that are being made by our software from the database to the application. We selected this framework once again due to the ease with which it will be able to be implemented considering our chosen technologies and

how the tool integrates with them. For example, Postman supports importing schemas to automate much of the testing construction for the user as opposed to having to manually transfer these aspects into the testing framework as we would have to do otherwise. The tool also supports automation of these tests after writing them, allowing quality control to be handled automatically in our CI/CD pipeline as opposed to having to manually confirm that each adjustment is still passing tests. Overall the use of this framework should greatly reduce our overall workload as it will automate much of the testing process after some initial setup that is greatly simplified via a number of integrated tools and features.

9.5 Electroencephalography (EEG)

9.5.1 What is EEG?

Electroencephalography is the recording of one's brain waves using an array of electrodes placed in specific areas of someone's head. These electrodes are then connected via a series of leads, back to some circuit board which can provide us with a data stream. It has many applications, ranging from medical uses such as the detection of seizures in those with Epilepsy to research projects utilizing EEG data for a variety of reasons. One example could be inputting a focused EEG dataset into a machine-learning model to predict a user's enjoyment of a particular genre of music.

Typically for research purposes, EEG is recorded using headwear integrated with both flat for flat surfaces, such as your forehead, and dry comb electrodes for electrodes placed on the hair. The flat electrodes are placed using a high-conductivity gel (generally consisting of glycol, organic polymers, and cellulose), in order to achieve minimal electrical impedance.

9.5.2 Impedance & Artifacts

Impedance during EEG can have a profound negative impact on the data being collected, the bands being monitored can become very distorted, and not make very much sense. This is why when performing an EEG, it is important to ensure all electrodes are firmly fastened to the head and to apply an appropriate amount of gel to those flat electrodes.

Another contributing factor to high impedance can be bad contact with the reference node. A reference node is used to calculate the potential difference between some

other electrodes in question, say located at T8, to the ground electrode. After understanding this it becomes obvious what can happen to the response from the T8 electrode, and any other electrode if the reference node is having a high impedance. We would expect to see problems in the EEG, more specifically *artifacts* are introduced into the data.

There are a few common types of artifacts in EEG we have to be aware of. In the image below are the following examples: blinking (1), bad electrode contact, often referred to as high impedance (2), swallowing (3), or bad reference node contact, this causes all channels to spike (4).

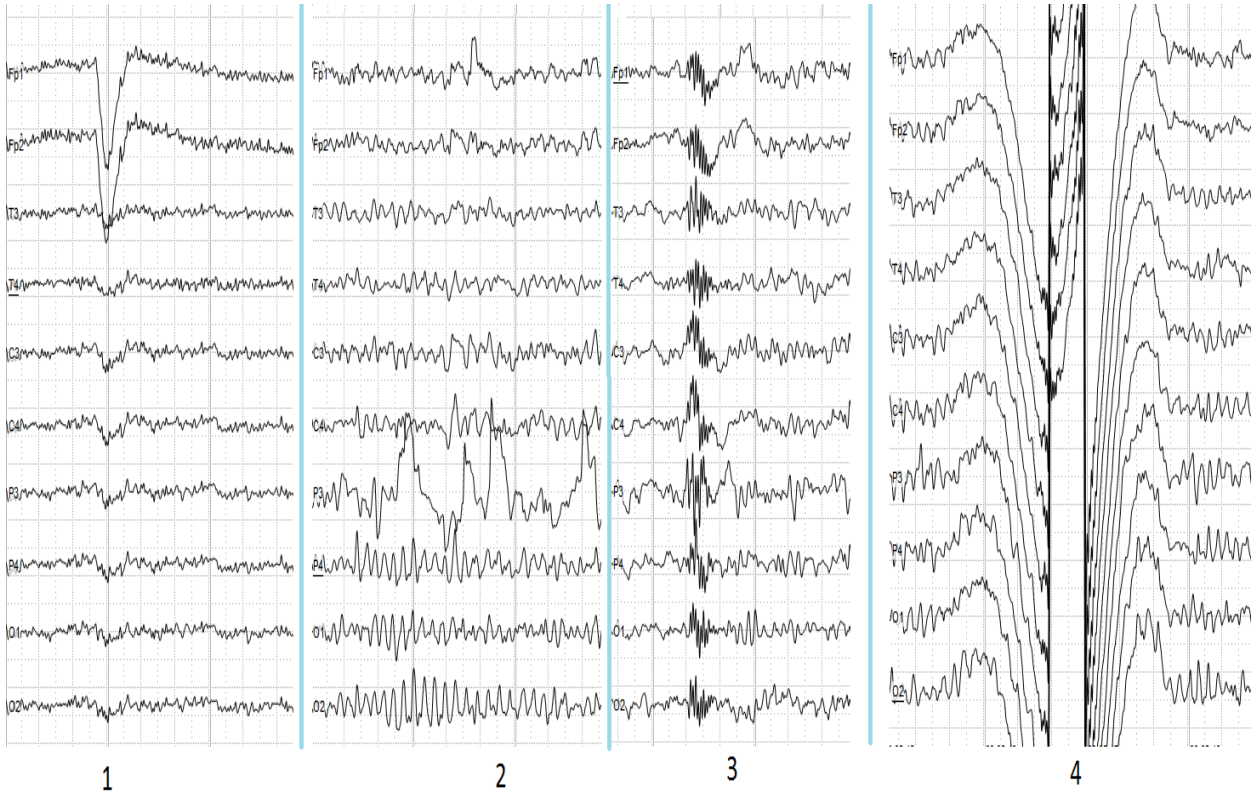

Figure 9.3 Four Types of EEG Artifacts

These artifacts can be very apparent when visualizing EEG data, and because they generally look the same across several occurrences, one can easily learn how to label these simpler events when they occur. For the BrainBeats project, when we are ready to begin processing EEG data we are going to have to be mindful of these artifacts in the data stream prior to the generation of midi.

Though it doesn't fall into the scope of this iteration, in future iterations of BrainBeats I can see the possibility of a system that monitors for artifacts like 2 and 4 in order to alert a user of a faulty electrode connection in order to prevent inaccurate recordings. Then Artifacts like 1 and 3 could have an option that the user can choose, do they want eye blinks, swallowing, or other similar events to contribute to their recording? Otherwise, I believe there would be a way to filter these from the data stream. If the user decides to keep them in the datastream, maybe those artifacts can be used in a creative sense to trigger some programmable parameter to affect the music that is being generated. Maybe one could swallow to effect a frequency filter sweep or blink to trigger a kick or snare, all while their brainwaves are creating a melody.

9.5.3 EEG Bands

In EEG, rhythmic data is separated into different bands or frequency groups. As there has not been a universal classification of these bands, there have been conflicting definitions between theoretical and applied cases. The improved model that will be referenced in this section uses the following classification of frequency distribution:

Band	Frequency (Hz)
Delta	< 4
Theta	≥ 4 and < 8
Alpha	≥ 8 and < 13
Beta	≥ 13

Figure 9.4 EEG Frequency bands: Improved definitions

Normally when processing and analyzing EEG, a fast Fourier transform is applied to the bands to accurately visualize the frequency amplitudes. Frequencies will be divided evenly between each band, specifically four per band. Additionally in this improved model, further integrating and or summations within individual bands are applied. Now, the amplitude of each band of frequencies can be measured, along with any relative maximum amplitudes in the given band. Depending on the use case, bands are subdivided, even more, resulting in more precise amplitudes.

9.5.3.1 Delta band

This band is mainly found frontally in adults, during slow-wave sleep (In the REM stage of sleep). While these occurrences are very rare, they have also been recorded during high-attention tasks. The amplitude of this band is usually high and easily visible while recording an EEG.

9.5.3.2 Theta band

Theta frequencies are associated with tiredness, and are seen more often in those who are in a resting or idle state. They are usually not contained to one specific region of the brain, dependent on age. Compared to other bands of EEG, the amplitude of Theta frequencies are relatively low.

9.5.3.3 Alpha band

The Alpha band contains the strongest amplitudes of the other bands and can be found on both sides of the posterior region of the brain. It has been shown that a greater amount of Alpha band activity can be seen on someone's dominant side. This is also the band where you can predominantly see “blinking” events.

9.5.3.4 Beta band

This band consists of primarily low amplitude waves, equally distributed on both sides of the brain. These frequencies are associated with high activity, such as focusing on a task, high anxiety, or general problem-solving.

9.5.4 Electrode Placement (10-20 System)

Electrode placements follow the nationally recognized 10-20 system. This system labels electrode placements based on the region of the brain it’s located on. For example, the frontal lobe would be called F, prefrontal Fp, Temporal T, and so on for each lobe. This model has been around since roughly 1957, and since then a higher resolution version has been used called “Modified Combinatorial Nomenclature, or MCN.

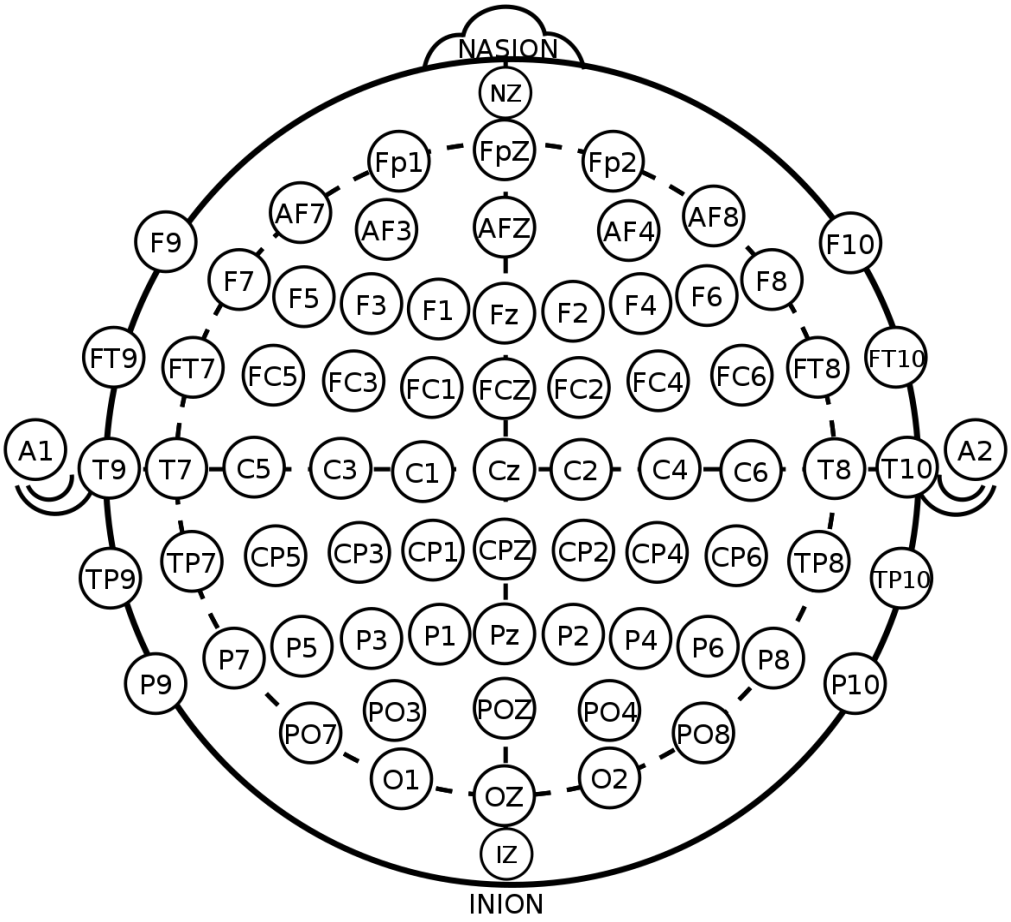

Figure 9.5 Nationally recognized 10-20 MCN EEG placement

This system adds nodes in between the already existing nodes of the 10-20 system, with a modified naming convention to accompany this. For example, there is now a TP node, symbolizing the space between the parietal lobes electrode P, and the temporal lobes electrode T. This higher resolution system also changes the names of four nodes: T3 and T4 are now T7 and T8, while T5 and T6 are now P7 and P8. One other key change is to the numbering convention. As the original 10-20 method has fewer nodes, the MCN has had to adjust the existing numbering so that the left hemisphere now contains 1, 3, 5, 7, 9, and the right hemisphere contains 2, 4, 6, 8, 10. If we are monitoring nodes F1 and T2, that means we would be monitoring the Left frontal lobe and the right temporal lobe.

9.5.5 Electrode Placement (BrainBeats Configuration)

The headwear we are working with from OpenBCI means we would have limited access to nodes listed in the 10-20 system. Of the nodes we can access, we have concluded the most important would be from the following: FPZ, FP1, FP2, FT7, FT8, T7, T8, P7, P8, O1, O2, and OZ. Of the two types of OpenBCI circuit boards we have the Ganglion Board, and Cyton Board, the Cyton Board has the most capacity, with 8 channels for electrodes. Since we have 12 channels of interest but can only support 8, we will have to cut down our list.

The easiest elimination we can make will be to utilize FPZ, rather than both nodes FP1 and FP2. However, we will not be doing this for the Occipital Lobe as this is one of the critical lobes utilized by professional musicians when listening to and performing music. For this case, we will use O1 and O2 in conjunction, rather than OZ for emphasis on the data from this region. Now that we've removed two nodes this brings us to the Temporal lobes, our T7, and T8 nodes.

Typically the left Temporal Lobe is responsible for speech and visual recognition, while the right is responsible for music processing and distinction of tonal sequences (think melodies). Previously, we discussed the reasoning for some of the questions we asked in our survey. One of which asked the participants for their dominant hand, or if they were ambidextrous. This is due to several studies showing that a person's handedness can be a sign of their dominant Temporal Lobe. While most people have a left dominant temporal lobe, there have been observed correlations that someone who is left-handed is more likely to have a right dominant temporal lobe. We ended up deciding to use both just to remain consistent in the nodes we were providing to our machine-learning model(s).

Now, with three nodes eliminated so far, we can move to P7 and P8. We won't be eliminating either of these nodes as they deal with sensory information such as vision and touch, particularly from one's hands. These nodes will play a larger part in our research study as data would not be limited to just data from regions responsible for the auditory and language processing, but also how given stimulus during recording is affecting them. Whether they are listening to a melody and afterwards playing it, or composing a short melody based on visual stimuli from images we provide, we would be getting a response from this region.

On to the last elimination, we were somewhat unsure what to choose between FT7 and FT8. After reviewing the previous selection of nodes, the thought occurred to put emphasis on the right Temporal Lobe, using FT8 which is just in front of it (Behind F2, in front of T8). This decision was made because as previously mentioned, the right Temporal Lobe has been shown to be critical in the processing of tonal sequences. This too is another significant node to us, and since we can't access the auditory complex, this would be one of the next best placements.

This leaves us with a final configuration of the following eight nodes:

1. FPZ: Frontal Lobe
2. FT8: Frontal-Temporal (Between the Frontal and Temporal Lobes)
3. T7: Left Temporal Lobe
4. T8: Right Temporal Lobe
5. P7: Left Parietal Lobe
6. P8: Right Parietal Lobe
7. O1: Left Occipital Lobe
8. O2: Right Occipital Lobe

These nodes are subject to change, dependent on the datasets we utilize for our machine learning model. Currently, many of these nodes are shared between the datasets we are considering, so we will be recommending that users use these nodes within the BrainBeats web application. They will likely be provided text upon moving to the record screen with this configuration, and if time allows, a small diagram showing these locations for the OpenBCI headband.

9.6 OpenBCI EEG Hardware

9.6.1 Circuit Boards

OpenBCI provides two different circuit boards for monitoring EEG, the four-channel Ganglion board, and the eight-channel Cyton board. They also sell an extension kit for

the Cyton board called the “Daisy” allowing for 16 channels. All of the .sch and .pcb design files are available within their documentation and on their GitHub.

9.6.1.1 Ganglion Board

The OpenBCI Ganglion board is a device which runs on the open-source software known as The OpenBCI GUI, it has four high-impedance differential inputs which can be connected to electrodes, allowing a user to measure EEG. According to OpenBCI, the Ganglion board was designed with an open-source PCB capture software called KiCAD. According to the documentation provided, the following details are given regarding the Ganglion board:

- Specs
    - Power supply: 3.3V to 12V DC battery
    - Current Draw: 14mA when idle, 15mA connected and streaming data
    - Simblee BLE Radio module
    - MicroSD Card Slot
    - Switches to manually connect/disconnect inputs to the REF pin
- Input Pins
    - GNDA: Power Supply: 0 Volts
    - D_G: Driven Ground: 1.5 Volts
    - REF: Combined Reference input.
    - +1: Channel input 1
    - +2: Channel input 2
    - +3: Channel input 3
    - +4: Channel input 4
    - AVDD: Power Supply: 3.0 Volts
- Data format

- 20-byte packets sent, 1 byte containing an id and 19 bytes of data.
- EEG data recorded at 200Hz, compressed to 100Hz for Bluetooth transmission.

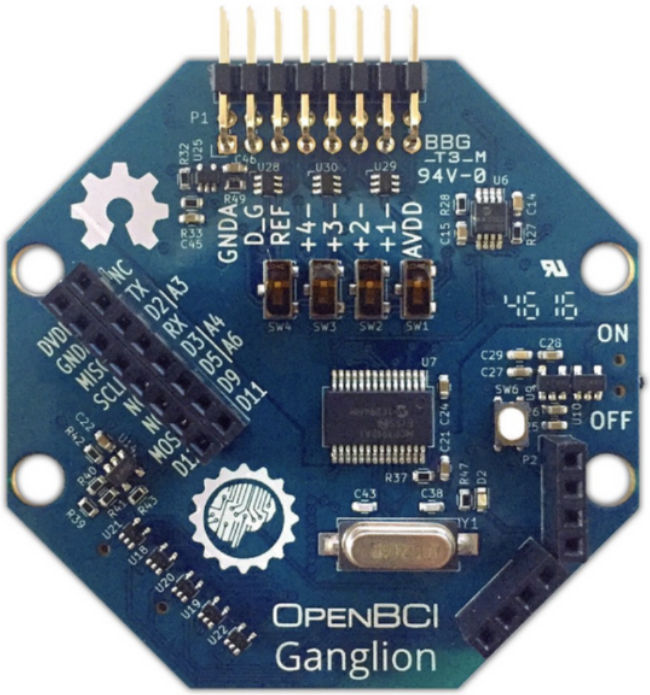

Figure 9.6 OpenBCI Ganglion Board

9.6.1.2 Cyton Board

In the OpenBCI documentation, it states that the Cyton board was designed with Design Spark, another open source software PCB capture software, different from that used during the design of the Ganglion board. The documentation on OpenBCI also provides the following details regarding the Cyton board:

- Specs
    - Power: 3-6V DC Battery
    - Micro SD card slot
    - RFduino BLE radio
    - Voltage Regulation: 3.3V, +2.5V, -2.5V

    - PIC32MX250F128B Microcontroller with chipKIT UDB32-MX2-DIP bootloader

- Input Pins
    - AVSS: 2.5 Volts
    - N1P: Channel input 1
    - N2P: Channel input 2
    - N3P: Channel input 3
    - N4P: Channel input 4
    - N5P: Channel input 5
    - N6P: Channel input 6
    - N7P: Channel input 7
    - N8P: Channel input 8
    - BIAS: Reference pin
    - AVDD: Power Supply: 3.0 Volts
- Data Format
    - 33-byte packages sent
        - Header: 1 byte for the ID and 1 byte for the Sample number
        - EEG Data: 3-26 byes taken in 3-byte increments for each EEG channel
        - Aux and Footer: 6 bytes for any extra accelerometer data, and the final byte for the booter with an exit code.
    - 24-bit signed integer (Needs to be converted to 32-bit signed integer from within the program.)

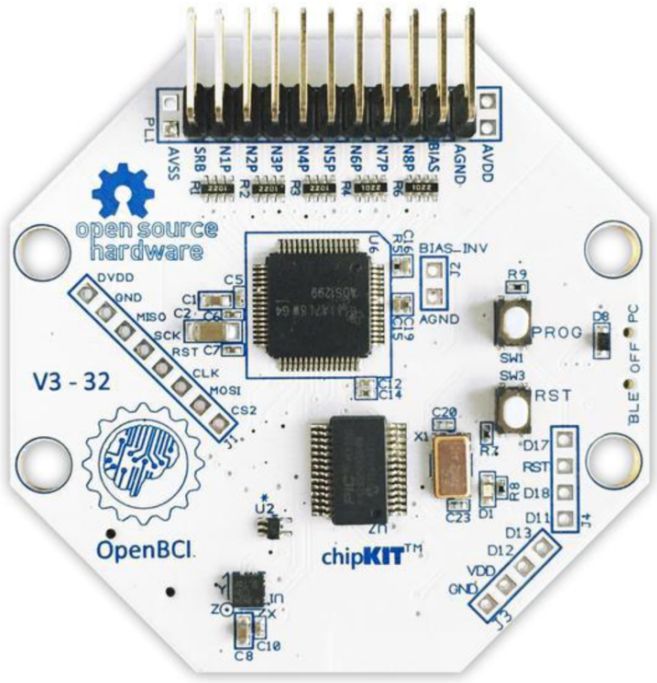

Figure 9.7 OpenBCI Cyton Board

9.6.2 Headwear

OpenBCI also provides several types of headwear, the largest one capable of supporting up to 35 nodes. They all range in size and use cases, the simplest being a velcro headband with equidistant electrode slots, and the more complex variant, a full electrode cap. Each type of headwear has the option for three different sizes. Size small ranges from 42 to 50 centimeters, size medium ranges from 48 to 58 centimeters, and size large ranges from 58 to 65 centimeters.

9.6.2.1 Ultracortex Mark IV

This headset is 3D printed, and therefore the most customizable headwear that OpenBCI offers. It supports up to 35 electrodes, a mount for the Cyton board, and a mount for the battery rack.

OpenBCI provides three different variants with their own pros and cons.

- “Print-It-Yourself” variant
    - Most cost-effective method

    - Many of the components must be printed, filed down, and glued.
    - Added $20 - $50 / Kg in printing materials, messing up components can mean restarting them entirely.
    - Large time investment spent assembling vs. working on the application
    - Comes with cables, node units, ear clips, and screws.
- “Unassembled” variant
    - Comes with 3D printed components, cables, node units, ear clips, and screws.
    - Even though no printing is required, parts still need to be glued and assembled.
- “Pro-Assembled” variant
    - Comes with all components, and the unit is fully assembled.
    - Most expensive variant, however much time would be saved by not having to print, cut, trim, and assemble many of the components.

Each Ultracortex Mark IV kit comes with six screws for the plastic components, three of the long EEG cables, two ear clips, spikey electrode units for nodes located in the hair, and two flat electrode units for nodes located in areas without hair. They also come with the optional “comfort units” which help distribute the weight of the headset and all of its components for a more comfortable fit.

9.6.2.2 Headband

This headband kit is the easiest to use, and fastest to set up and run of the available headsets. It is adjustable so it should fit most head sizes, and will be much more comfortable than other kits. This unfortunately comes at the cost of losing access to many nodes in the 10-20 System, 20 of the 78 nodes. Included in the headband kit are the two ear clips, comb-snapped EEG electrodes, flat-snapped EEG electrodes, a pack of cables for the electrodes, and the strap itself.

9.6.2.3 Electrode Cap

The Electrode Cap kit from OpenBCI comes in two variants, one where the use of electrode gel is required, and one where it is not. However, the setup and practically everything else between variants is the same. The Electrode Cap has a maximum capacity of 19 EEG channels, though, in the setup documentation it is paired with the "CytonDaisy" board using 16 of those channels.

While this is the most expensive option, it comes with many pros. Like the Ultracortex Mark IV, this headwear can grant access to many more nodes in the nationally accepted 10-20 system, though with a much lower chance of impedance. The cap has a snug fit, ensuring a more restricted electrode movement, and more secure contact with the user.

9.7 Digital Audio Workstation (DAW)

9.7.1 What is a DAW?

Prior to easily accessible technology, the creation of music was often limited to a recording studio. The modern era of technology created a lot of waves in the development of music, DAWs, or Digital Audio Workstations, providing anyone the ability to create professional-sounding music in their bedroom. They range in complexity and price ranges, where some are better suited for professional audio engineers, and others better suited for beginners. However, the primary application of them for our project is that they are compatible with MIDI files, and therefore any DAW would be compatible with a downloaded file from BrainBeats.

There are several key components and features that are shared across all DAWs that allow us to create music, and manipulate sounds. These include, but are not limited to:

- The ability to create a project which hosts multiple audio files in the same organized timeline, this includes input such as MIDI and MP4.
- The ability to separate file data by multiple components, such as different channels.

- The ability to pair with an external MIDI device to capture input for easy manipulation.
- Virtual studio technologies, audio units, and effects plugins, such as gain, compression, and limiters.

9.7.2 External MIDI Devices

MIDI devices are any device that can interface with a DAW and are capable of outputting, manipulating, and controlling MIDI. Some are developed purely to function as a MIDI controller, allowing for straightforward manipulation of these events when making music in a DAW. Others can be standalone instruments but have the capability to be used as a full, or limited MIDI device, an example of this being an electric piano. Electric pianos contain their own sound banks and can be played on their own, or can be plugged into a computer using a MIDI cable in order to write MIDI. In this example, the pitch would be changed as expected, by playing different keys on the electric piano. Velocity, which is defined by note loudness and or timbre, is affected by how hard you hit a key, allowing for dynamic control in music. Sometimes these electric pianos can have assignable knobs, capable of mapping to other MIDI parameters, or even parameters from virtual studio technologies or plugins within a DAW.

9.7.3 Virtual Studio Technology (VST)

Virtual studio technologies are audio plugins for software which come in many forms and can contain a variety of sounds. They package analog recording studio equipment and real, sampled instruments, into software to allow for incorporation with a DAW. When placing a VST on a track in a DAW, any MIDI located on the track will then be triggering the sounds contained from within this VST. This gives any musician the ability of incorporating a multitude of different sounds and instruments into their music, regardless of if they can actually play it. A piano player, while never having played the violin in their life, can use a piano MIDI controller and control many of the properties of a violin.

An example would be a very popular VST that comes from Spitfire Audio called “Spitfire Labs”. This is a free collection of sounds and instruments ranging from soft pianos and bright synths to large soundscapes and cinematic effects. These are all

designed and recorded by professional audio engineers, providing anyone with professional-sounding audio right out of the box.

9.7.4 Plugins and Audio Units

Plugins (Universal format) and Audio Units (Apple's format) are both ways to manipulate sounds from audio files like wav, mp3, flacc, aiff, as well as MIDI. They can be as simple as a gain plugin only containing the functionality of making a sound quieter or louder, or very complex such as an entire "mastering chain" of effects and plugins packed into one. When creating music in a DAW, many plugins can be placed on a given track. Once a plugin is turned on, all of the audio coming from this track will be routed through the plugin, and the resulting audio played.

9.7.5 Putting Things Together

If a musician comes to the BrainBeats site and records a track, we want them to be able to download the MDI file and import it into their DAW in order to make music with it. As an example, a musician generates a default project in a DAW called "Logic Pro X" using a tempo of 120 BPM, and four beats per measure. Based on the results of music generation on the BrainBeats software, they can then create a MIDI track and place the BrainBeats MIDI file in the timeline. After selecting the Spitfire Labs VST, and activating whatever plugins they desire, they hit the play button. Now their brainwave-generated MIDI is controlling this piano sound and they can continue to arrange other elements into this piece of music.

From this research we raise the question, "but what if we could eliminate the process of downloading and importing the MIDI from the BrainBeats web app?" We believe this could be achieved by future teams of the BrainBeats project if the primary goal was to implement this software as a MIDI interface for DAWs. BrainBeats could still be available for use on the web, but transitioning to a MIDI controller implementation would greatly expand its overall utility for the user base we are targeting.

9.8 Methods

The methods section of our research provides our acquired knowledge of the approaches and processes in regard to the development of each section of our application. This is the necessary information required in order to achieve the outcomes of the previously detailed project requirements sections.

9.8.1 Machine Learning Background

Our goal is to be able to develop and display our music generation through machine learning in real time as opposed to the previous implementation of having a lookup table that associates MIDI with specific brain waves. We have decided this route is more practical when scaling our application to have many different forms of scripts, where performing one activity may not have the same expected output in MIDI as another. EEG signals often need to be preprocessed based on knowledge of the task at hand, this is because artificial EEG signals can be emitted in sample collection. For example, eye blinking is one of the largest producers of artificial EEG signals, creating influence over datasets. Because of this, it is important to detect eye-blinking artifacts in order to prevent them from lowering the quality of our collected EEG data. Because of this, we have determined it is practical to implement a deep convolutional neural network (CNN) when analyzing our sample set. CNNs offer us the convenience of avoiding spending long periods of time extracting artifacts manually, this is because CNNs excel at image classification. CNNs allow us to analyze EEG data when converted into an image format, classifying specific aspects of an EEG sample to quickly recognize which points in the data are essential information to us, as well as the significance of those data points.

When deciding on the implementation of a specific training model for machine learning we are approached with multiple options. In order to implement these models we will need to use Tensorflow, which allows us to build and train models through the Keras API for deep learning. There are multiple factors that must be considered when taking EEGs as input, a primary one being artifacts such as blinking. We have determined that using MNE with Python is the most practical approach with this in mind. MNE is an open sourced Python package that will allow us to analyze EEG data. We find that we can use MNE to reduce artifacts prior to implementation in our machine learning models in order to make sure we have an accurate analysis of a user's EEG signals.

There are a variety of models already constructed for the analysis of EEGs through convolutional neural networks, some with extensive research done on their performance. Such as EEGNet, which is a convolutional neural network that is "robust enough to learn a wide variety of interpretable features over a range of BCI tasks." We found that referencing this model would be ideal for us because it is designed to be implemented over a wide range of EEG readings, rather than just training for a specific

purpose. This would allow us to save time in designing an effective model for our purposes. Because our data sample is over a small sample size, it makes sense for us to try to develop a model that performs well when provided with limited training. The Army Research Laboratory conducted research on the results of EEGNet and found that “EEGNet generalizes across paradigms better than the reference algorithms when only limited training data is available.”

Our mindset is that it would be impractical to generate music strictly using EEGs within the timeframe of our project, however, we do believe that building a relationship between EEGs and MIDI through the composition of melodies when a composer is reacting to a stimulus is something that BrainBeats should explore in order to allow for future implementation of music generation that is realistic and more practically suited for our application. The research study we describe in our documentation showcases the execution of this and why we find it to be an effective way of music generation. We do believe that within our timeframe it is realistic to implement machine learning, and have decided that we can do this through building relationships between EEGs and emotions and correlating these emotions to those found in the composition of music. If we were to implement this in the most ideal way, we would develop a relationship between brainwaves and the composition of music in real time. We could then use this relationship to train a model where when we input EEGs we can generate music based on prior knowledge of how composers would have produced a melody when reacting to a stimulus that is similar.

In regards to developing a neural network to implement these datasets, we are presented with multiple options. The two well explored and primarily utilized neural networks are convolutional neural networks and recurrent neural networks, each of which are expanded upon in the following sections.

9.8.1.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are algorithms built for deep learning that can give importance to images and differentiate them from others. CNNs are designed based on the functioning of neurons in the human brain and their connections to each other, with inspiration from the visual cortex of the brain. CNNs operate by converting a picture into a matrix of pixels, they are then flattened into a vector. CNNs separate RGB images into their red, blue, and green channels, but can be separated into a variety based on the different types of image spaces, Grayscale or CMYK for example. CNNs

reduce images into easily processable forms while maintaining features that allow the model to predict accurately. Because of this reason CNNs are highly efficient at scaling to larger datasets.

Implementing a CNN works in the following way:

1. Conversion of an input image into a matrix of its pixels, Height x Width x Channels. These images may need to be resized to fit with kernels.

2. We now start to construct the Convolutional Layer, this begins with selecting a filter or kernel, which is a matrix that we can select depending on the effect we want, for example, the Laplacian kernel (Figure 9.8) is very good for edge detection because it is a matrix of the second partial derivatives. Different kernels will create different feature maps as output, which is why the size of the kernel is important in avoiding the loss of pixels for training. Kernels will have the same depth as the image that is being worked with in the case that there are multiple channels to work with.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 9.8 The Laplacian Kernel

3. After selecting a kernel, it can become apparent that we would want the edges of our image matrix to have the center of the kernel be applied to them in order to have better feature recognition. If this is the case, we will want to apply padding to our matrix. For example, if we wanted to apply this practice to an 8 x 8 matrix so that a 3 x 3 filter can interact with the edges of the matrix, we would add a border of one pixel to the outside of our matrix. Convolution involves performing the dot product of the matrix with the kernel, so padding with 0 is a good idea since it will always result in a 0 output on the newly padded edges

where pixels do not actually exist. Because the feature map of a padded matrix is the same output size as the image itself, it allows for deep models to be developed with highly impactful feature maps.

4. Once we have selected a kernel and decided if padding should be applied, we perform convolution on a matrix by striding over the matrix based on a stride value we select. This means that we move the filter from left to right across the matrix from row to row, applying the dot product of the subsection of the matrix that is the same height and width of the kernel with the kernel. This goes until the image has been traversed fully. Different strides affect the output size of the feature map, which can be used to downsample our image.

5. This successfully completes convolution, which extracts the high level features of the image, for example, its edges. There can be many convolutional layers, each can be capable of extracting different features from the image, including lower level ones such as colors and gradient orientation.

6. Our next step is developing the Pooling layer, which is utilized for reducing the size of the convolved feature, with the purpose of decreasing the amount of computation required to process data. Because the convolutional layers map features based on their precise position in the images they analyze, movements in the image will result in the feature map being inapplicable. This is where pooling becomes handy, as it performs operations on each feature map that's been created in order to develop pooled feature maps.

7. Pooling begins with the selection of a pool type, the two types are Max Pooling and Average Pooling. Max Pooling outputs the max value of the portion of the image covered by the kernel, whereas Average Pooling returns the average of all the values from the portion of the image covered by the kernel. Max Pooling has much better performance than Average Pooling.

8. Once we have selected a pool type, we will apply it to the feature maps developed in the convolutional layer, which essentially creates a summarized version of the features from the input. This is useful for when small changes are applied to the original input image, such as a slight rotation or scaling up or down, because pooling creates an invariance to the translation of the new matrix.

9. The development of the layers above allows us to recognize the features of an image, now the goal is to develop a way to classify these features. To do this, we flatten the input image into a vector of a single column and perform backpropagation to every iteration of training based on the number of epochs we define. This will allow our model to detect dominating features in the image and classify them using Softmax classification.

9.8.1.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network that can take in sequential data and be trained in order to be aware of the past. Their ability to retain knowledge allows prior input to influence current inputs and thus the output. RNNs are designed to work on data that involves sequences, this is particularly good for our case because we are working with MIDI files. There are many types of RNNs, but for our purposes, a One to Many network architecture is the best fit, as we have a single input that produces multiple outputs. RNNs, like CNNs, also utilize a variety of activation functions (Figure 9.9), which are implemented in between the input and output of a current neuron in a network. They determine whether a neuron should be activated. Each activation function serves its own individual purpose and are applicable in their own specific scenarios. Almost all modern neural networks utilize non-linear activation functions because they can create complex mappings of the inputs compared to the outputs. Because most input into a neural network is going to be non-linear, such as EEGs in our case, it is important that the functions that analyze them are also non-linear.

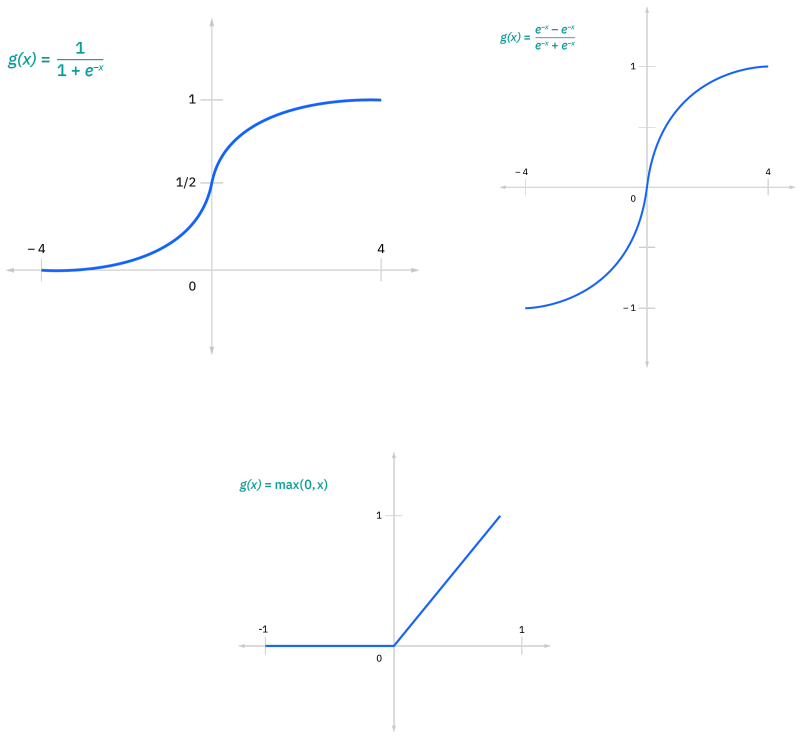

Figure 9.9 Visual Examples of Activation Functions (From left to right: Sigmoid, Tanh, Relu)

These functions allow for backpropagation, which is the term for the tuning of weight functions in order to improve the accuracy of a neural network's output. Backpropagation uses the mathematical chain rule through the computation of a gradient, the partial derivatives of a vector, on a loss function. In turn it allows a neural network to assign a lower weight to nodes that provide high amounts of error in their output.

There are many different RNN architectures, but for our purposes, a Long Short-Term Memory (LSTM) network makes the most sense. LSTMs utilize gradient descent to recognize long-term patterns, this makes them incredibly useful in situations where our network must remember information over a prolonged period of time. In the case of BrainBeats, we will use this technology to create a neural network that can:

1. Receive EEGs as input and convert these into a category of emotional states
2. Take these emotional states and convert them into MIDI

9.8.2 Keras

Keras is an API written in Python designed for deep learning which runs on TensorFlow. It is a high-level API written on top of TensorFlow 2, it provides users with building blocks for quickly developing machine learning processes. Keras is compatible with Ubuntu 16.04 and later, which is ideal for running on our server. Keras is also designed for scalability, allowing us to implement our machine learning algorithms in the future with a larger amount of scripts. Because it is built for user convenience and is primarily focused on deep learning, we find that this library will be very useful in allowing us to quickly develop machine learning algorithms.

9.8.3 Music21

Music21 is a Python library built for analyzing music quickly and in a simple manner developed at MIT. One of the aspects which stands out from the Music21 documentation is that it can analyze music that is being generated on the fly, since we are trying to analyze EEGs and convert them into MIDI in a short enough time to keep user experience high. Music21 is also able to work with a large variety of music formats, including MIDI which makes it highly applicable to our development process.

9.8.4 MNE

MNE is an open-sourced Python package that is utilized for analyzing neurophysiological data, EEGs in our case specifically. MNE provides us with analysis of EEGs through predeveloped machine learning algorithms that are capable of recognizing bad channels and removing them from the analysis. This means that it is also capable of artifact suppression, in our case this is from factors such as blinking, which are highly impactful on EEG readings. Utilizing MNE will therefore allow us to reduce our time developing code to remove artifacts from our collected datasets and user input and therefore provide us with more accurate models and output for music generation. Figure 9.10 showcases the workflow of inputting EEG data into the MNE network to be processed, it includes important API features showing their required input and output.

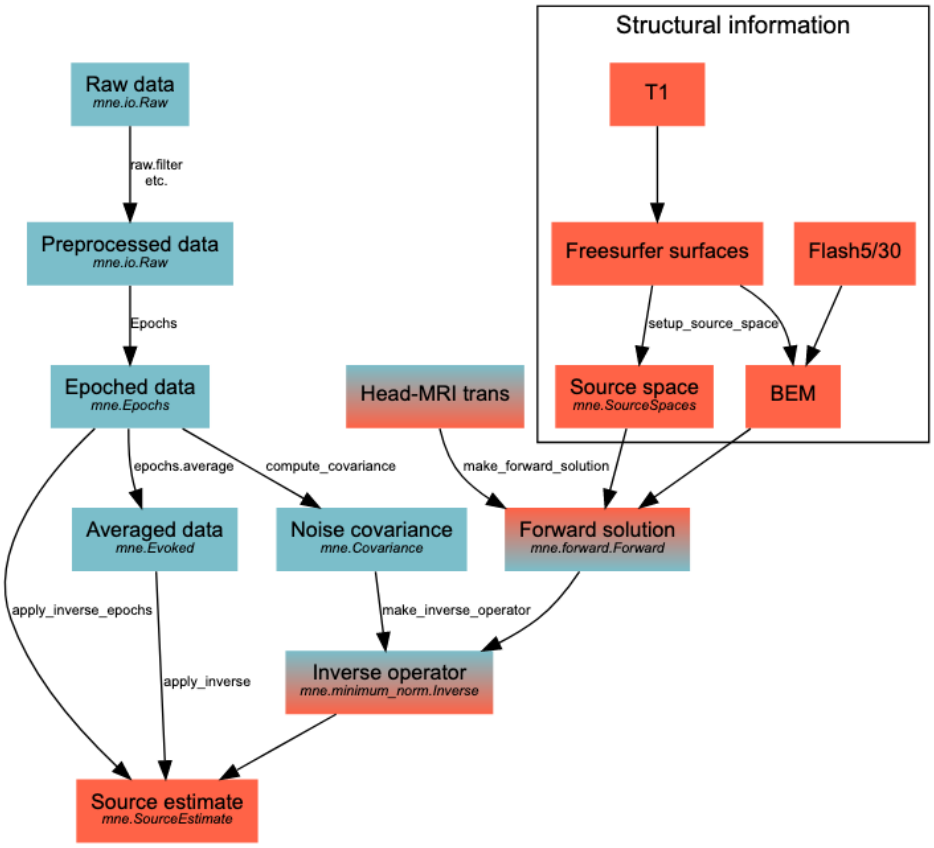

Figure 9.10 The Workflow of the MNE software

9.8.5 Datasets

There are a multitude of open source datasets available that have analysis on EEGs. We find that due to the niche nature of our project there isn't one specific dataset that can be used to convert directly into MIDI from EEG, however, there is enough available data to build correlations between EEG and MIDI through the implementation of multiple datasets. We describe these datasets in their respective labeled sections below.

9.8.5.1 DEAP Dataset

The DEAP dataset is a publicly available dataset developed for emotional analysis based on physiological stimulus. Collected at the Queen Mary University of London, the DEAP study recorded participants' EEG signals while they all viewed one minute long snippets of music videos. This was repeated over 40 samples for each of the 32 participants. Of those 32 participants, they were able to get a frontal recording of the

participants' faces while watching. Each participant described each video in terms of their arousal, their level of emotion in terms of positivity or negativity (commonly referred to as valence), whether they liked the music or not, and their familiarity with the music. These are used to label the EEG signals in terms of their enjoyment and will allow us to use them as training data to correlate a BrainBeats user's EEG input with emotions. Because the DEAP dataset monitors a person's emotions when listening to music, we find that it is very effective for our purposes, and can be utilized until there is future support for a fully developed method of data collection for a direct EEG to MIDI machine learning model.

One of the issues we originally had when searching for a model of this sort is the large room for bias, as people in different parts of the world have different tastes in music as a result of culture and other outside impacts. We found that if a dataset played music such as folk music for a specific culture, there would be significant taste or distaste for it based on impacts outside of the pure focus on the music. The DEAP dataset handles this by selecting music targeted toward the demographic of their research study. With a participant base made up primarily of European students, they selected European or North American musicians predominantly. We find that this helps in avoiding bias in terms of factors outside of the music itself.

We find that this dataset can be used in correlation with the dataset referenced in section 9.8.5.3 to generate a multi-stage neural network that takes in EEG signals, converts them into labeled emotions, and then converts these labeled emotions into music. This will be done primarily through the valence and arousal labels, as they are found in both of our datasets and allow us to build a correlation between the two. Once we've assigned these labels to our EEGs, we can feed them into the next dataset which will output us with a MIDI file.

9.8.5.2 DREAMER Dataset

The DREAMER dataset is a database of recorded EEG acquired through a study where participants were recorded reacting to audio-visual stimuli. 23 participants undertook this data collection and described their personal beliefs on their level of valence, arousal, and dominance after each stimulus was presented to them. The conductors of this experiment analyzed their classifications of arousal, valence, and dominance in comparison with other public datasets which provide these same labels and found comparable results. One of the reasons this dataset is particularly interesting is that it's

conducted with low cost EEG capturing devices. This provides us with a dataset that can be used with high accuracy results in the long-term scope of our project in the case that the compatibility of BrainBeats expands upon just OpenBCI headsets.

The source of stimuli selected for this particular dataset is audio and visual from a set of 18 movie clips. They served the purpose of attempting to elicit emotion in participants, seeking to showcase a wide range of emotions. They sought to target nine specific emotions: amusement, excitement, happiness, calmness, anger, disgust, fear, sadness, and surprise. According to their research, the length of the clips, ranging between 65 and 393 seconds, is due to the fact that video stimuli between one and ten minutes is enough to elicit a single in humans. The study utilizes the 10-20 system, with headsets placed on locations AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4, M1, and M2. We intend to keep these in mind when capturing EEG ourselves.

One of the primary interests in this dataset is its labeling of EEGs with arousal and valence. This will allow us to build a model that can correlate arousal and valence with EEG and assign labels to user input from an OpenBCI headset. We can associate these created labels with the EMOPIA dataset mentioned in section 9.8.5.3 to then convert these labels on EEG signals into MIDI.

The developers of this dataset allow for more accessible use of their work in comparison to the DEAP dataset. Because of this, we find that it is more practical to pursue model development with it despite the closer correlation to our project that the DEAP dataset does have.

9.8.5.3 EMOPIA Dataset

The EMOPIA dataset is a publicly accessible dataset that is focused on pop piano music and the perceived emotions in the genre. The purpose of this dataset was to provide a public resource that breaks the mold of standard music datasets, offering clips of music with emotional labels. This implementation allows for usage in a project that seeks to generate music. They provide clip-level emotional labels from a collection of 387 songs, they provide labels based on a clip’s level of valence and arousal. These clips acquire their labels through these levels which are separated into four quadrants, quadrant one (labeled Q1) contains clips with levels of high valence and high arousal, quadrant two (labeled Q2) contains clips with levels of low valence and high arousal, quadrant three (labeled Q3) contains clips with low valence and low arousal, quadrant

four (labeled Q4) contains clips with low arousal and high valence. Another label is included which is very useful for our purposes, this label is titled as ‘Dominant Q’, and it specifies which quadrant of valence and arousal level that dominates the song. Each song contains multiple clips, which allows us to train a model that can generate more realistic music than previous methods implemented by BrainBeats.

One area we find there is room for future improvement with this dataset involves the collection of a broader scope of instruments in order to allow for future music generation with polyphony. As this dataset is strictly implemented for piano pop, it limits the potential generation to a specific instrument, but will still provide us with a more practical output than our previous lookup table implementation. We find that this dataset’s use in our project can allow for good reference in the future development of BrainBeats which might look to be more ambitious in machine learning development after there is room for polyphony in script generation.

Using the aforementioned DEAP dataset, we can convert EEG signals into labels with high valence, low valence, high arousal, and low arousal, we can determine which quadrant these EEGs lie in as well as which one dominates the signal and then feed these results into a new model which is trained on the EMOPIA dataset. This model will output a MIDI file that should resemble a realistic melody performed on piano. Doing this will allow us to create a music generation method using machine learning which can provide our users with more realistic MIDI files to take and improve upon for their own composition purposes.

The formatting of our input from the EMOPIA dataset is read in as a CSV file, which has data that is structured like the following:

Song ID	Q1	Q2	Q3	Q4	DominantQ
ad29393c	2	1	0	0	2
…	0	2	1	3	4
…	1	1	2	0	3

Figure 9.11 EMOPIA Dataset in Table Format

These song IDs are attached to MIDI files included in the dataset which we can use for reference as well.

9.9 Security and Encryption

While simple security would be very easy to implement such as base 64 encrypting it would still be quite simple to decrypt afterwards should the software be breached. Therefore we decided it would be necessary to look into some sort of method of improving security. Some options we considered are described in more detail below. Each of these algorithms has its own benefits and drawbacks which will be explained in more detail in their relevant subsections. For our purposes, performance is a very important metric that we wanted to ensure is optimized as our hosting system will have a limited amount of resources available.

When developing any web application, security is one subject that is often overlooked or disregarded. BrainBeats version four will be an application where users can create an account, collect EEG data, store information about their tracks, and insert unique modules for music generation. The nature of our product requires that we have secure data as we are handling EEG data, something that is highly sensitive. The scope of BrainBeats version four data collection and storage will require different security practices to ensure user data is secure. Security practices implemented within BrainBeats will also encourage minimal risk of command injection or other cyber attacks in the present and future.

One goal that the BrainBeats version four team has is that version four of this application will have a robust framework that can easily be understood and added upon. With the implementation of security practices in the updated BrainBeats framework, the core of this application will be ideally secure. Therefore, any additions or updates to the security of BrainBeats should enhance what already exists and not be extensively redesigned; unless deemed necessary by a future implementation of BrainBeats where a complete application remodel is being performed.

9.9.1 Asymmetric Encryption

This method makes use of a public key used for encryption as well as a private key used for decryption. This is especially useful when keys would otherwise have to be shared, commonly increasing the chance for them being compromised. The public key

is available for use to encrypt any information while the private key is created by the individual user and therefore there is a much lower risk as this key is never shared. This is however not really a concern for us as our application is relatively low in terms of how many users it will have and this approach requires many resources and slower performance. Figure 9.12, referenced from SSL2Buy.com, provides a visualization of how asymmetric encryption is implemented.

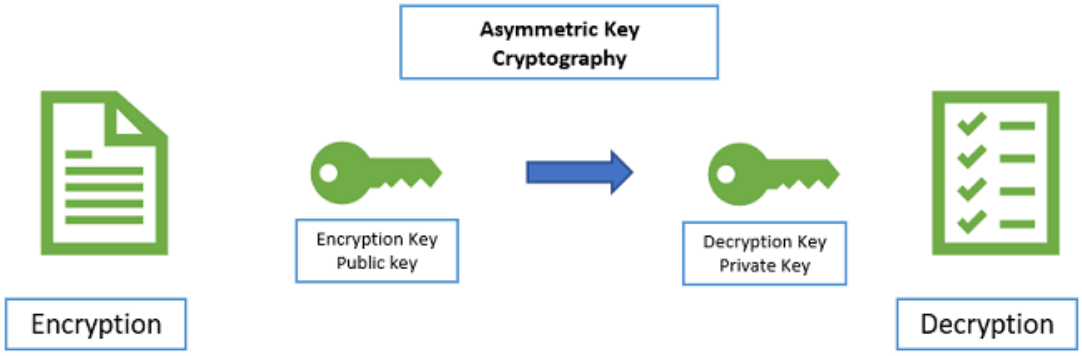

Figure 9.12 Asymmetric Encryption Visualization

9.9.2 Symmetric Encryption

Symmetric encryption uses one key to encrypt and decrypt all information, this can be useful in circumstances where the key is not being shared often. It offers rapid performance and low resource usage but is inherently less secure than its counterpart. Especially since it relies on one key, if that key is lost then it could compromise a large aspect of your security. Figure 9.13, referenced from SSL2Buy.com, provides a visual of how symmetric encryption works. Luckily for our aspect, the tool should not need to share the key very often and therefore this approach offers excellent benefits with not too much risk for our small system and can be easily implemented within our planned framework. The next step was to choose a specific encryption algorithm.

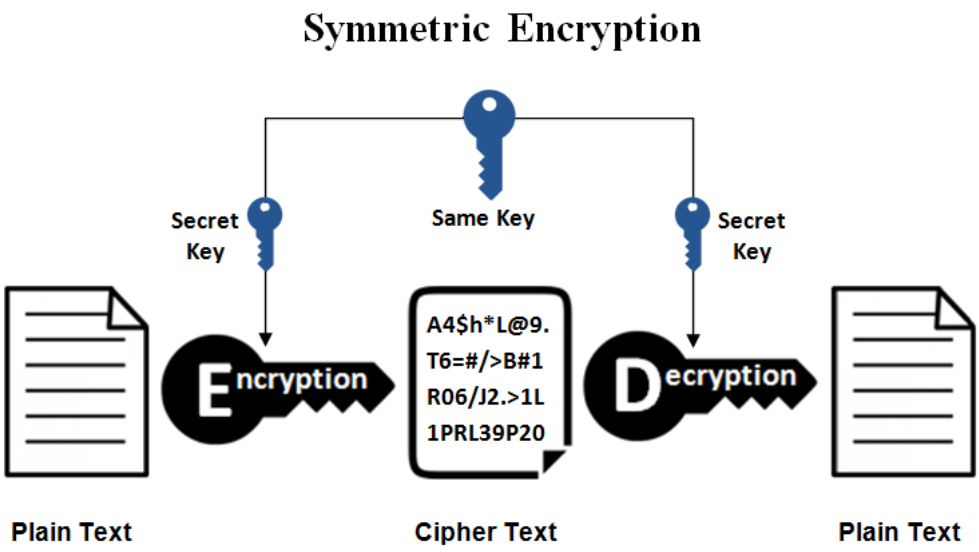

Figure 9.13 Symmetric Encryption Visualization

9.9.3 Advanced Encryption System (AES)

One of the most common modern symmetric encryption algorithms is AES which was created to replace the Data Encryption Standard as it became outpaced by modern malicious attacks. It corrected the failures of DES by using a variable key length giving the user the option to create a key length that matched their needs for security and ease of use on resources. This algorithm was made for handling large amounts of data with ease and therefore is quite useful for many modern applications.

9.9.4 Triple Data Encryption Standard 3DES

3DES instead decided to enhance the existing DES algorithm to preserve its usefulness in the modern time. It is quite simple in its approach since it applies the DES algorithm thrice to each block of information as the name suggests. This triples the 56-bit key into a more secure 168-bit one. This is a much slower process than other processes that are more modern. This is still considered to be a less secure option though as the blocks of data are quite still smaller than with AES so it is more susceptible to brute force decryption.

Ultimately as a result of the reasons listed above, we reached the decision to proceed with encryption by implementing AES. This decision was helped since it is one of the most commonly implemented methods and therefore has no shortage of resources and

existing libraries to make use of this method in a TypeScript backend. However, this technology only provides encryption and we still need to have authorization for users via another tool.

9.9.5 Hashing

The process of hashing data is a common and essential implementation of security within an application. Hashing is a form of encryption that involves passing data through a formula, or hash function, which produces a result known as a hash value.

A principal aspect of hashing data is the hash function that is used to encrypt data. The key to the functionality of a hash function is that the function must produce the same hash value given a piece of data. Therefore no matter how many times a unique piece of data is passed through the hash function, the same hash value unique to the data should be produced by the function every given time. Hash functions also typically generate hash values of the same length regardless of the amount of data that is passed to it.

By transforming the original data into a hash value, the original data is disguised and protected from hackers or cyberattacks. Hashing data can also benefit an application through faster data retrieval if a hash value is used to map object data (Zola). After the data is hashed it can then be looked up in a hash table or hash map in the form of a key and value pair.

BrainBeats version four plans to implement hashing primarily when dealing with data related to user information such as usernames and passwords. Hashing user data ensures that the user's personal information is kept secure from outside data breaches as the data being stored will be encrypted in the form of hash values. Hashing this data will also allow for quick data retrieval for verification during the login process.

The BrainBeats version 3 team implemented hashing data within the application through a standard library based on JavaScript called bcrypt.js. BrainBeats version four is an updated and remodeled version of BrainBeats version 3 but written in TypeScript. As TypeScript is a superset of JavaScript, the bycrypt.js library should function properly within TypeScript and will also be used within version four of BrainBeats.

9.9.6 Salting

The term salting is another form of security used for encrypting passwords specifically. In other words, it is a form of password hashing that increases the amount of security and minimizes the risk of personal information such as a password getting leaked during a potential data breach.

Salting is the process of adding a unique value to the end of a password, referred to as salt, in order to create a different hash value (Nohe). Although not necessary, it is an additional step that can be done during the hashing process for additional security. This additional step can be especially useful against brute force attacks. An example of a brute force attack is when a hacker uses a computer to test every possible combination of characters in an attempt to guess a user's password. By adding the process of salting when hashing a password, the attempt to guess a user's password becomes increasingly more complicated. Adding a salt to the end of a password before hashing produces a new unique hash value which disguises the real hash value of the user's password.

One vulnerability to salting is when the salt is discovered by a hacker, the hacker can add the salt to the end of every possible combination of characters to discover the user's password (Nohe). To avoid this, the salt added to the password should be unique to that password. Adding a unique salt to every password adds additional security that prevents further attacks against a hacker trying to collect data.

As mentioned in the previous section, 8.2.1 Hashing, bcrypt.js is a standard JavaScript library that should also work within TypeScript and supports salting. Bcrypt automatically generates the salt with the hash through the "bcrypt.hash" function. Therefore, version four of BrainBeats will be implementing salting within our application through the bcrypt library.

9.9.7 Multiple Layers of Verification

Having multiple layers of verification is a critical security measure to avoid cyber attacks including command injection attacks. Command injection attacks can occur when an application passes unsafe user-supplied data. Many command injection attacks can be avoided through sufficient input validation. Having multiple layers of

validation can have other benefits besides security including having clean and concise code which will also allow for easier debugging when experiencing errors (Nidecki).

The measures taken to ensure multiple layers of validation will start within any functions that are written to pass and accept data within BrainBeats. As the BrainBeats version four team reworks the version 3 code into TypeScript, all functions will be inspected to ensure that they only accept expected input. To ensure that only the expected input is accepted into functions, input will be validated before being passed into its corresponding function. For example, if the user must input a password when creating an account, the password must follow specific guidelines and restrictions. Some restrictions that can be put on user input for a password are the types of characters the user is allowed to use and the length of the password.

After providing restrictions to the user before they enter input, the input should then be parsed using a function created specifically for the input to check whether or not the input follows the guidelines and restrictions specified. Taking these additional steps in the code will allow for greater protection of BrainBeats and will fight against cyber attacks, specifically command injection.

9.9.8 Information Secrecy

Providing too much information to users when unnecessary can allow hackers to find vulnerabilities in an app much easier than if the information was kept secret. The use of console.log in web development, warnings, or detailed error messages can result in a side-channel attack against the application.

A side-channel attack is a cyberattack when extra information made available to an attacker is gathered and exploited to identify how an algorithm or protocol is implemented (“Side-channel attack”). If an attacker knows the way an algorithm is implemented, the attacker can then use that information to develop an attack against the application and have better knowledge of how to gather and leak information that is meant to be protected.

An example of how a side-channel attack can occur is accidentally giving the user access to variable or function names used to transfer and validate data. The user may receive access by improperly providing error messages, such as entering the variable name where the error occurred instead of creating a separate error message for what

has occurred with less information. Warnings in the application's log can also give an attacker an idea as to where your application may be vulnerable.

To avoid side-channel attacks, the BrainBeats version four team will make an attempt to address all warnings and assure any error messages or warnings with valuable information are removed before launching the updated website onto the server. Another approach the team can take to ensure unnecessary information is not being shared with the user is only providing the amount of information the user needs. For example, if the user enters a password that is too long as input when creating an account, the user should receive an error message that says the password entered is too long, In this scenario with password restrictions, the user should not be provided with every single restriction in the error message when only one restriction was not met.

Ensuring there is some secrecy in the information being provided to the user will make it more difficult for attackers to gather information on the inner workings of the application and develop an attack.

9.9.9 JWT Authorization Implementation

Our team made the decision to use JSON web tokens to share keys because it is a common internet standard that allows us to securely communicate JSON data between endpoints. This is achieved by tokenizing a given session and using that token to authorize the user across the web application. This also happens to have the added benefit of being an open standard and therefore not incurring any additional costs to the team. Likewise, the open nature of JWT provides us with a large resource base to gain knowledge on the development of web-based applications using it. Figure 9.14 showcases the process of converting an encoded JWT to a decoded one in order to view the contents inside of it.

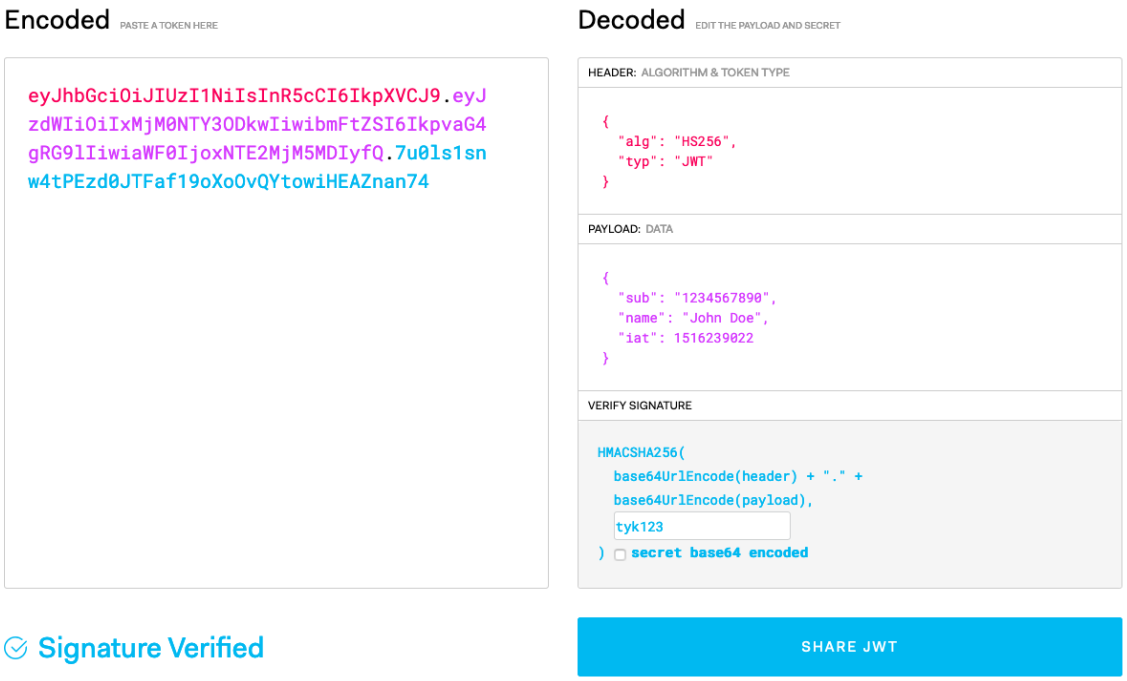

Figure 9.14 Visual Example of JWT Encoding and Decoding

Another interesting benefit to making use of this standard is the option to differentiate types of users therefore by including an identifier in the header. This would allow for a very simple implementation of differing levels of access. For example, guests could be given access to listening to recorded pieces but not recording while a signed in user could be given access to recording songs and posting them to the library. Finally, an administrative user could have permission to remove songs and control sitewide settings.

9.10 Design and Accessibility

Music is a form of art that we believe should be accessible to all people regardless of ability or situation. BrainBeats v4 aims to execute good design practices and prioritize accessibility for its users to maximize everyone’s chances of properly using and enjoying our product. The goal of BrainBeats version four’s desire to improve accessibility led us to the incorporation of the Web Content Accessibility Guidelines (WCAG) into our UI design, which will allow all sorts of users, even those with disabilities, to enjoy and use the software. We find this to be especially important with our desire to see our product used long-term in a variety of ways rather than just for the composition of music.

The Web Content Accessibility Guidelines (WCAG) prioritizes the four principles of accessibility which include having content that is: perceivable, operable, understandable, and robust (Henry and Dick). Giving people the ability to generate music using BrainBeats is something the BrainBeats v4 team wants to ensure is inclusive and user-friendly. Without the principles of accessibility, a wide range of people with disabilities will not have proper access to BrainBeats. Below is a more in-depth description of how WCAG addresses the four principles of accessibility and how the BrainBeats v4 team attempts to incorporate them into our project.

Due to time constraints, the BrainBeats v4 team is unsure if we will be able to implement every single practice of WCAG; but we hope to incorporate as many guidelines as we can that we believe will significantly improve the accessibility of our target audience and follow the principles of accessibility.

9.10.1 Perceivability

The information and user interface provided to a user must be presentable in an easily perceivable format; meaning users must be able to recognize the information with their senses.

At a glance, the guidelines for the WCAG that address perceivable content include providing alternatives to the user so that there is no confusion on what something on the website is or does. Some guidelines that address having perceivable content are providing text alternatives for non-text content and providing captions or other alternatives for multimedia (Henry and Dick). This is important to increase the understanding of the content for the user in the chance that there is confusion on what a button does or what a graph represents. Other guidelines include creating content that can be presented in different ways without losing meaning and making it easier for users to see and hear content (Henry and Dick). Both of these guidelines address accessibility with perceivable content. It is important to provide different options for users and help users who may be using assistive technology such as text-to-speech.

BrainBeats v4 addresses accessibility by having perceivable content that clearly identifies the function and use of all buttons, graphs, and tools on the website. To increase the perception of content, BrainBeats wants to first prioritize providing significant contrast between foreground and background content so that the website’s content is easily visible to the user. After assuring content has significant contrast and

is easily visible, providing clearly identified labels and headings will increase the understanding, identification, and perception of content. Significant contrast is important for BrainBeats version four plans to use a dark background throughout the majority of the website for a dark theme. An example of good contrast on a dark background can be seen when a light blue is used on top of a black background (Figure 9.15). An example of bad contrast on a dark background can be seen when a dark red is used on top of a black background (Figure 9.15). Making interactive elements such as buttons easily identifiable using both of these methods will also address the perception of BrainBeats.

Figure 9.15 Good Contrast (on the left) vs. Bad Contrast (on the right).

In addition, BrainBeats will not be using color alone to convey information in an attempt to increase accessibility for users who may have color blindness or the inability to differentiate between colors. An example of using color alone to convey information is to have a red outline over a text box when the user has entered invalid information for a username or password. To make the content in this situation more perceivable, using symbols such as an ‘x’ next to the text box alongside the red-colored border increases the permeability of the content and allows a wider range of users to see and detect what may be wrong and why their information is not being properly submitted due to an error.

Although creating significant contrast, clearly identifying content, and making interactive elements easily identifiable is of the highest priority when addressing the perceivable principle of accessibility, the BrainBeats v4 team also aims to make the layout and design of our content understandable and consistent amongst different viewport sizes. By doing so, we would be addressing the WCAG guideline of creating content that can be presented in different ways without losing meaning.

9.10.2 Operability

The user interface must be easily operable through its components and navigation; therefore the interface should not require an action that the user can not perform in order to properly navigate.

There are various guidelines within WCAG that address the operable principle of accessibility. Due to time constraints, BrainBeats v4 will be prioritizing some guidelines over others. Giving users enough time to read and use content, helping users navigate and find content, and avoiding content that can cause seizures or physical reactions are guidelines included in the Web Accessibility Initiative's "WCAG 2.1 at a Glance" that BrainBeats hopes to prioritize in order to increase accessibility and make the BrainBeats site operable.

To give users enough time to read and use content, the BrainBeats v4 team will provide controls for content that starts automatically and avoid content that only appears on the website for a limited amount of time. By doing so, the user will have the ability to move to the next page or read the next message at their own pace instead of having timed content on the screen.

Flashing lights and certain visual patterns have the possibility of triggering seizures or physical reactions from users with photosensitive epilepsy or other conditions. To minimize the possibility of someone being triggered by content on BrainBeats, the BrainBeats v4 team will be assured that no flashing lights or visual patterns be displayed on the site for user music generation or when users visit the site. We will also assure that no content moves too quickly, such as images in a carousel throughout the site or on the home page.

BrainBeats also plans to provide clear and consistent navigation options as well as the use of overlays, such as tooltips, within the React-Bootstrap library to help users navigate and find content. Tooltips will allow the user to hover over the content and

display a message with guidance on what the content is or does. For example, a button that says ‘Cancel’ may be unclear to the user but if we add a tooltip that specifies ‘Exit Music Generation’, then it may give the user a better idea as to what will happen when the user presses cancel within the Music Generation studio. The BrainBeats v4 team hopes that incorporating overlays into the website will make the website more operable and easier to understand.

Some other guidelines that address the operable principle are making all functionality available from a keyboard and making it easier to use inputs other than the keyboard. These are guidelines that the BrainBeats v4 team plans to add as a stretch goal after prioritizing guidelines that the BrainBeats v4 team feels has a greater impact on operability for our users.

9.10.3 Understandability

Users must be able to properly and easily understand the information being presented and how to operate the user interface; confusing user interfaces or displays of information should be avoided.

To make content understandable, BrainBeats will follow guidelines in “WCAG 2.1 at a Glance” which recognizes the following points: make text readable and understandable, make content appear and operate in predictable ways, and help users avoid and correct mistakes (Henry and Dick).

As mentioned previously when discussing the perceivable principle of accessibility, BrainBeats’ user interface design will include sufficient contrast between foreground and background to assure that text is always readable. Clearly defined headings and labels will also assure that the text is always easy to understand and clearly defined.

Making content appear and operate in predictable ways is assuring that the process of using our program is easy to understand. For example, the user will have to press sign up for a signup screen pop-up to appear rather than just a hover. Aspects of the website will only appear when clearly directed to appear unless it is a tooltip overlay being used to convey information to the user on what content is or does. Otherwise, there will be no random pop-ups that will confuse the user on what they should or should not press.

The use of color, symbols, labels, and common design practices in our user interface will also make the functionality of our website easier to understand and more predictable. A common design practice would be placing a submit button on the right side and a cancel button on the left side of a page. This is common on many sites to place the buttons in this order and if BrainBeats were to deviate from this design practice, it may confuse users who already have experience using the internet and visiting websites. The same idea goes for a navigation bar being placed and locked onto the top of a screen or a delete button being represented by a trash can icon.

Input assistance is also useful for helping users avoid and correct mistakes. For example, having clear instructions on what the user should enter as input is important and helps the user avoid mistakes when using the program. Through clear instructions, the user also will have a reference to what might have gone wrong when they receive an error using the program. Error identification messages provided to the user also allow the user to have a better understanding of their mistakes and what steps they should take to correct them.

9.10.4 Robustness

The technology must be robust enough so that it can be interpreted properly by a large range of user agents and assistive technologies over time. As technology and user agents advance, the users should still be able to access the content reliably.

Guidelines according to WCAG that address the robust principle of accessibility may include parsing, name and role determination, and status messages (Henry and Dick).

In regards to parsing, elements must have complete start and end tags, must be nested according to their specifications, and must not contain duplicate attributes or IDs unless specifications allow for it. To have robust code, BrainBeats plans to incorporate these design practices to create clean, adaptable, and robust code. This is especially important because the code we add to BrainBeats v4 will be transferred onto future versions of BrainBeats.

User interface components may also have their names and roles programmatically determined to be more robust. States, properties, and values of user interface components that can be set by the user may also be programmatically set so that notifications of these changes are able to be made available to assistive technologies used by the user.

BrainBeats v4 is passionate about developing organized and robust code so that future developers can understand the code and more accessibility features can be added as technology grows. These practices that address the robust principle of accessibility will be considered throughout the development process of BrainBeats v4. Although the BrainBeats v4 team is unsure if we will be able to include notifications of changes to item components in version four of BrainBeats, it is something we will be considering throughout the development process and have added as a stretch goal.

9.11 Focused Study

9.11.1 Change of direction

The study referenced below was originally scheduled to be undergone during early November of 2022, allowing the machine learning team time so that we could upload these recordings to UCF's AARC Newton Cluster in order to conduct unsupervised learning between Senior Design 1 and 2. Unfortunately, due to a lack of willing participants as well as other time constraints during development, we found it impractical to perform this study. Instead, we felt it was more practical to be reliant upon publicly available research studies and their corresponding datasets for our machine-learning methods as a result of the aforementioned issues.

We do however find that the research study method we have designed is important enough to continue to be referenced in our documentation for future reference despite the lack of data collection from it. We believe that the implementation of this study can be provided for a more unique and focused dataset that caters to what BrainBeats sets out to accomplish.

9.11.2 Survey Description

In order to gather participants for our focused study, we designed a survey and issued it out to interested members of the composition team at UCF. The following is the description and overview we provided to willing participants:

"*BrainBeats Research Study: Interpretation of EEG data for the generation of MIDI.*

BrainBeats is a recurring Senior Design project in the Department of Computer Science. The main focus of this project is to provide several options for users to generate music using their brainwaves. One of these options utilizes machine learning

models trained on EEG data. This study consists of connectivity with an OpenBCI headset to a subject's head, consisting of eight nodes, and conductive gel, followed by exposure to various types of music. Following this, the subject will perform a menial task to reset their brain activity, which will then be followed by having the subject then try to recreate the song in their head to the best of their ability. This will be performed over 3 trials with 3 separate songs (A slow melody, a single-instrument song, and a song with two instruments). During this process we will actively record EEG waves from the participant, noting down recordings primarily on channels: A1, A2, T1, T2, F1, F2, and FPZ based on the 10-20 system for categorizing EEG node placement on the brain."

Participant safety was a priority in the implementation of the research study. Because of this, we also provided participants with a statement and link to the conductive gel documentation, to make them aware of possible skin reactions. While harmless in most cases, the gel can sometimes irritate those with sensitive skin, and a small possibility to grow to a skin reaction. We felt it was important to include this in the study details because we would need to apply this to any flat electrodes we use.

"For allergy and other health-related information on the conductive gel, please reference the link here. (The gel is required to achieve lower electrical impedance on the nodes for a cleaner signal)."

This was followed by a statement and link to our research methods so participants would have a better idea of the tasks they would be performing, and why.

The link we provided for the *'Spectra 360 Electrode Gel'*, sent participants to a safety data sheet provided by Parker Laboratories, Inc. This sheet breaks down the classifications of the gel such as Hazardous Ingredients Content, Hards Identification, First Aid Methods, Physical and Chemical Properties, etc. The document states there are no known significant effects and no hazardous components contained in the gel.

The section we are mainly interested in is *"First Aid Measures."* In this section, multiple procedures are listed for if the gel was Inhaled, Ingested, came into eye contact, or skin contact. As we would need to use gel for electrodes placed on participants, we would have to consider the possibility of irritation and an even smaller probability of allergic reaction. The document states that if either of these events were to happen, to wash with soap and plenty of water. To prepare for this, we will be conducting this very near

a bathroom so in the case a participant has skin irritation or some sort of reaction, we can walk with them next door for them to thoroughly wash areas where the gel has been applied.

9.11.3 Survey Questionnaire

The purpose of the questions we prepared for the study was mainly to gauge where we would expect to see heightened activity in different regions of the brain, depending on the answers to these questions.

1. Which genres of music do you actively listen to?
    a. Electronic
    b. Pop
    c. Hip Hop / Rap
    d. R&B
    e. Classical
    f. Rock
    g. Latin
    h. Other
2. Have you ever played an instrument? If so, for how long?
    a. Less than a year
    b. 1 to 3 years
    c. 3 to 10 years
    d. Greater than 10 years
    e. I have not played an instrument
3. Are you capable of playing a melody on a MIDI keyboard?
    a. Yes

b. No

4. How many instruments do you play?
    a. 0
    b. 1
    c. 2
    d. 3
    e. 4
    f. 5
    g. > 5
5. How often do you listen to music?
    a. Rarely
    b. Sometimes
    c. Frequently
    d. Always
6. Do you write, score, or compose music? If so, for how long?
    a. Less than a year
    b. 1 to 3 years
    c. 3 to 10 years
    d. Greater than 10 years
    e. I have not written, scored, or composed music.
7. What is your dominant hand?
    a. Left-Hand
    b. Right-Hand

    c. Ambidextrous

8. Do you give BrainBeats permission to use your responses for the BrainBeatsV4 project and future research studies?

9.11.4 Study Methods

When creating our research study, we wanted to include a link to a separate doc containing our methods for this study. Contained in those methods was this introductory paragraph highlighting the purpose of the study, what we would need from them, and what would be gained. We then went into more detail, breaking down our arranged procedure once participants have been welcomed into the lab.

9.11.5 Study Overview

BrainBeats is an immersive application that allows composers to develop MIDI directly from their thoughts. In order to make the application more practical we are conducting a research study to improve our generation of MIDI via machine learning. The purpose of this study is to gain a better understanding of the correlation between the reaction to a stimulus and the composition of melodies. We will then utilize this correlation to build a machine learning model that attempts to accurately take EEG readings based on reaction to a stimulus and convert it into MIDI. This would allow us to generate music in real-time based on a user’s reactions to a song or image of choice.

9.11.6 Study Execution

We will begin by setting up an OpenBCI headset for participants, specifically targeting nodes T7, T8, FPZ or FP1 and FP2, FT7, FT8, P7, P8, OZ or O1, and O2. The selection of nodes will depend on the participant's dominant hand and experience as a composer. This is because research has suggested that there is a correlation between these attributes and the amount of brain activity in specific regions.

We will be applying two methods to our study. One is focused on visually stimulating a participant with an image, while the other is with audio. In both, we will have the participant focus their attention on the stimuli for 15 seconds. After which we will shut off the stimuli, then have them close their eyes until the readings stabilize.

This is where things will change depending on the method. During the image method, once they open their eyes, we will ask them to compose a monophonic melody with a

length of somewhere between 30 to 45 seconds. While they play, they will still be able to see the image, so now we have a dataset of the solo image, and a dataset of the image paired with the performance. During the audio method, once they have opened their eyes, we will ask them to play the melody they heard to the best of their ability. Each method will then be repeated, now with a different image and a different section of the melody.

In both methods, we ask participants to perform on a midi-piano which we will use to record their melodies utilizing a program known as LogicProX. In the case that a participant cannot perform due to a lack of familiarity with performing on a MIDI piano but provide their own instrument to perform on, they are allowed to perform on them to complete these steps with their instrument. In the case that this occurs, we would then record the raw audio with an H1n Zoom microphone, to be converted from an audio file into MIDI, where we would then match it up to the same interval in their EEG much like the implementation with the MIDI piano. Both of the methods described below are synced to each other via a timestamp acquired through any available way to record time.

9.11.6.1 Image method

1. Display an image, have participant look at the image for 15 seconds, and record their EEG readings while observing
2. The participant will be asked to close their eyes until their EEG readings stabilize from reacting to the stimulant
3. Have participants compose a melody within the range of 30 to 45 seconds based on their reaction to the image, and record their EEG readings while performing
4. Repeat all of the previous steps using a different but similar image.

9.11.6.2 Audio Method

1. Participants will listen to a 15 second snippet of a music snippet of a simple melody played on the piano, we will record EEG readings while the participants are listening.

2. The participant will be asked to close their eyes until their EEG readings stabilize from reacting to the stimulant
3. Participants will be expected to reenact the snippet based on their memory of the song.
4. Repeat all of the previous steps using a different segment of the song.

Both the EEG and MIDI generated during the study will be captured for use in the development of music generation models. These two files are then synced with each other in an external software.

Unfortunately, after sending out the survey and methods document, the statement we made about electrode placement depending on the participant's dominant hand was somewhat irrelevant. This is because the electrode placement between participants needed to be consistent in order to avoid bias in the data. It was concluded that regardless of a participant's handedness, rather than choosing between the T7 and T8 nodes we will be utilizing both of them. Thus we arrived at the node configuration from another previous section, for the study and general usage of the application.

10 Design

This section of the documentation describes the design approaches implemented in the development of our product in regard to the frontend and backend. It highlights our decision-making process in the steps of our project in order to reach the end goal of creating clean, concise, and user-friendly software.

10.1 frontend Application

The inspiration for the new BrainBeats user interface is influenced by various websites used for social media and music sharing such as Snapchat, Instagram, Spotify, and Audius. The BrainBeats version four team aimed to create a design that was accessible and followed good design practices for accessibility such as using contrasting colors and clear labeling of information. In order to make a professional-looking website, it was decided that a neutral color scheme should be used with accent colors inspired by the new BrainBeats logo.

The main priority of the new BrainBeats user interface is functionality for its users and creating good use of the screen space. To ensure the proper functionality of the new user interface, requirements were first gathered on what the user will be doing on the website and how they will be doing it. These requirements included creating an account and signing in, learning more about the BrainBeats application, changing settings for music generation, and recording music using the EEG headset. Looking at the user interface for a previous version of BrainBeats assisted greatly in understanding what purpose the UI served in the user experience.

10.1.1 Wireframes

The implementations of wireframes allow for a focus on space allocation, the functionality of the website, the priority of components, and the intended behavior of the site. Using the requirements and knowledge gained from the earlier BrainBeats user interface, wireframes were developed for the pages of the application that see the most user interaction and thus require the most importance.

The first priority was deciding what the home page of the website will contain. The home page will appear the same to both guests and registered users when they first visit the site, so a login and sign-up button was important to include and can be found

within a navigation bar at the top of the screen (Figure 10.1). A collapsible sidebar was also decided to be the most efficient way to navigate the site's different pages such as home, an about us page for learning more about BrainBeats, a page for searching existing tracks, and creating a new recording (Figure 10.1 and Figure 10.2). The large dark rectangle at the center of the screen would represent a graphic containing information about the website, such as a slogan, and at the bottom would be featured tracks that other users have uploaded to the BrainBeats site (Figure 10.1).

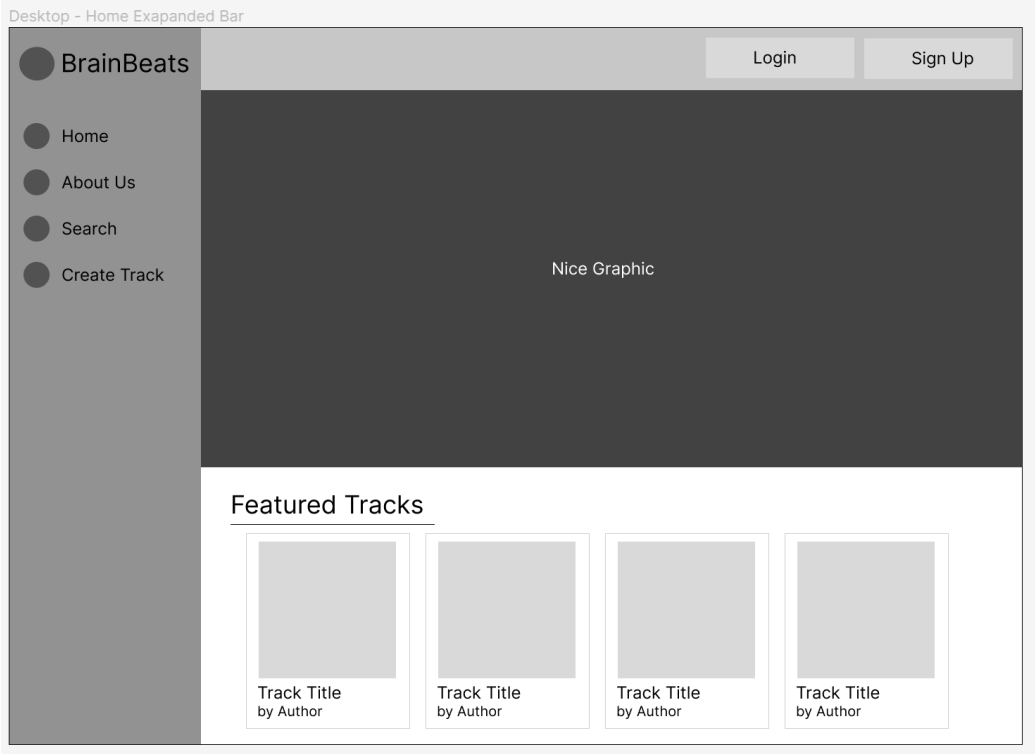

Figure 10.1 Landing Page for Guest/User (Expanded Sidebar) Example

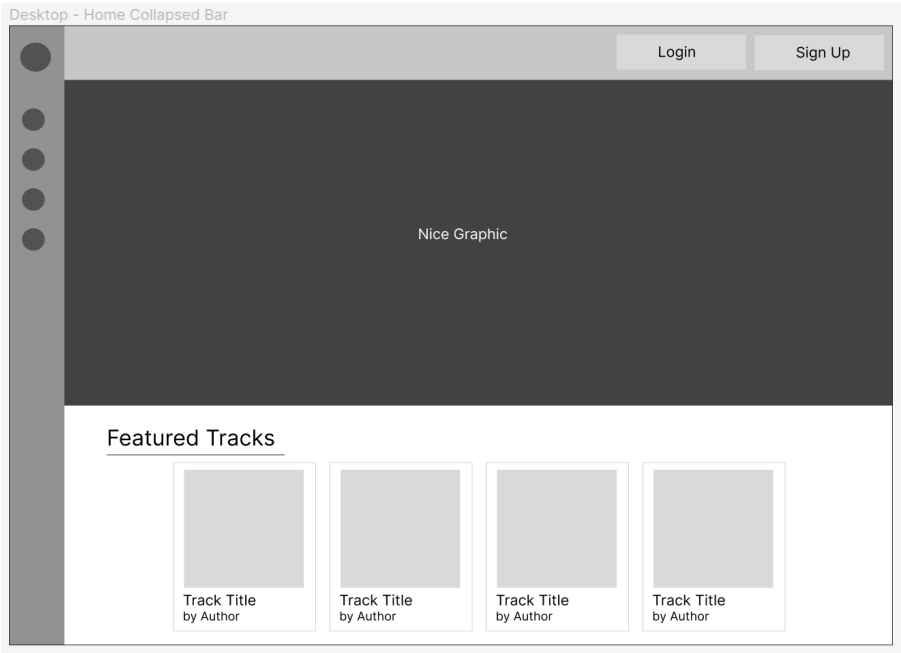

Figure 10.2 Landing Page for Guest/User (Collapsed Sidebar) Example

Next, wireframes for a login and signup page were created. A very simple user interface inspired by Snapchat and Instagram was designed to create a straightforward and easy user experience for users who want to register or sign in to BrainBeats. In our wireframes for the login and signup pages, circles represent a logo and rectangles represent input text boxes or buttons (Figure 10.3).

Figure 10.3 Login and Sign-Up Example Pages

Once the user wants to create a new track, the user will be prompted to a settings page that will allow the user to choose between basic settings or advanced settings (Figure 10.4). If the user chooses to go to advanced settings, the user will be able to

adjust music settings for instruments and other settings using a series of drop-down menus represented by gray rectangles (Figure 10.5).

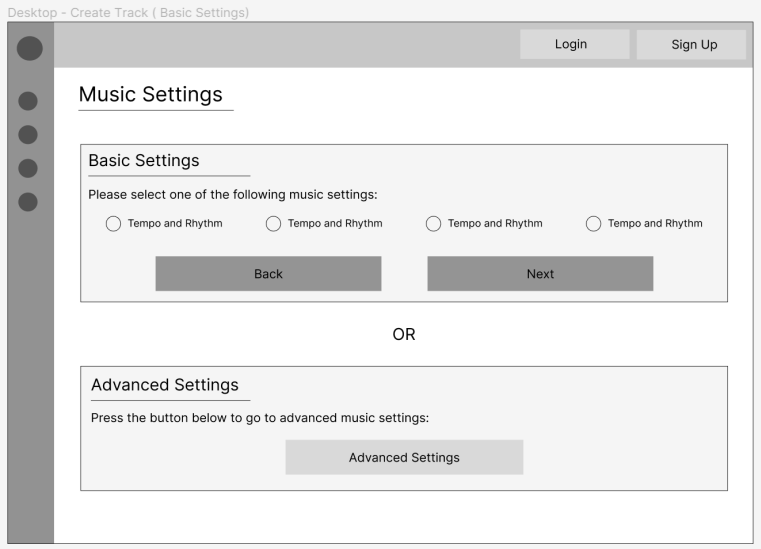

Figure 10.4 Choosing Basic or Advanced Settings Example Page

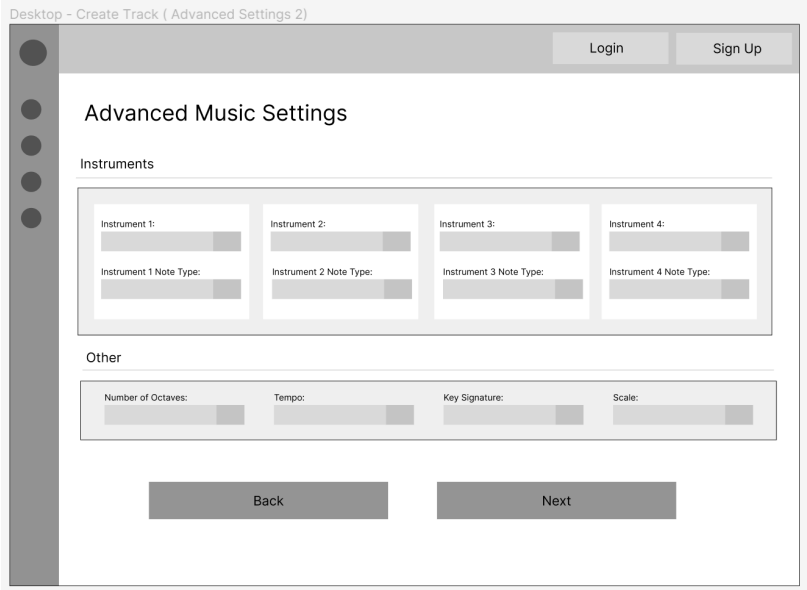

Figure 10.5 Advanced Settings Example Page

Once the user continues from basic or advanced settings, they would be directed to a page for recording music that presents the option to create visual cards that can influence their EEG recordings or skip this step and go straight to recording music

using buttons at the bottom of the Record Music page (Figure 11.6). The user will then be directed to the Recording Music page to record their track (Figure 11.7).

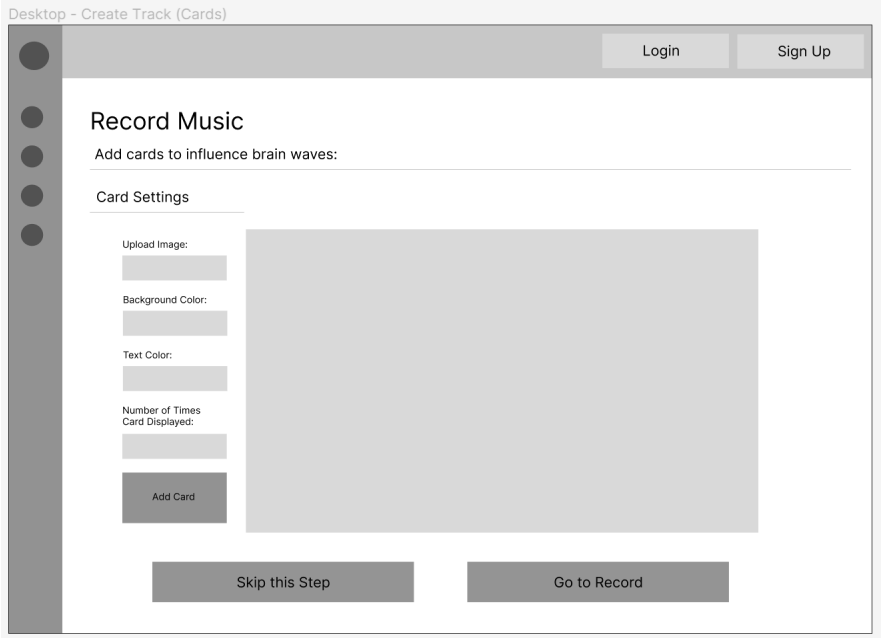

Figure 10.6 Record Music: Card Settings Example Page

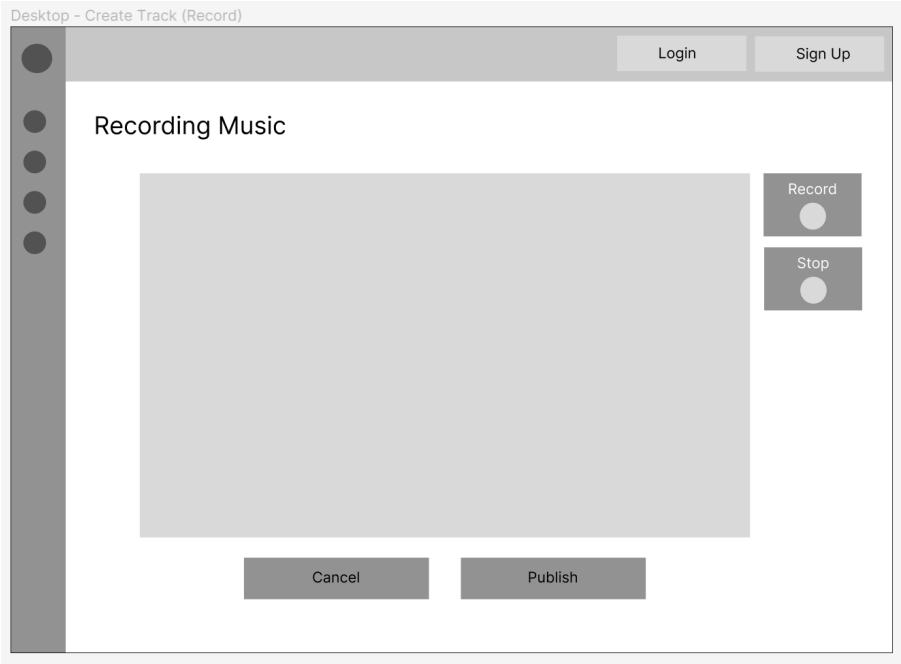

Figure 10.7 Recording Music Example Page

10.1.2 - Mockups

Mockups allow for greater design elements of a user interface to be displayed and to show an idea of what the final user interface will look like. The inclusion of colors, logos, icons, and images differentiates mockups from wireframes and allows for a greater focus on the appeal and aesthetic of the website as opposed to the requirements and functionality established within the wireframes.

To create a modern and sleek user interface, the BrainBeats version four team decided on a timeless color scheme; a series of grays, black, and white. To add color and personalize the user interface of BrainBeats, we decided to incorporate colors present within our logos such as teal and pink. The Hex code, or hexadecimal format for identifying colors, that we used for our accent colors was #43ABBB for teal and #CE8394 for pink.

For greater accessibility, we chose to create a significant contrast between any blue and gray colors used on our website as some people with color blindness will have trouble distinguishing these two colors. The significant contrast between colors can be seen on the landing page; as the sidebar is an extremely dark gray color and the highlighted tab on the sidebar menu is a bright neon teal (Figure 10.8 and Figure 10.9). The contrast in colors can also be seen on the music settings page between the light gray foreground and dark gray background (Figure 10.10).

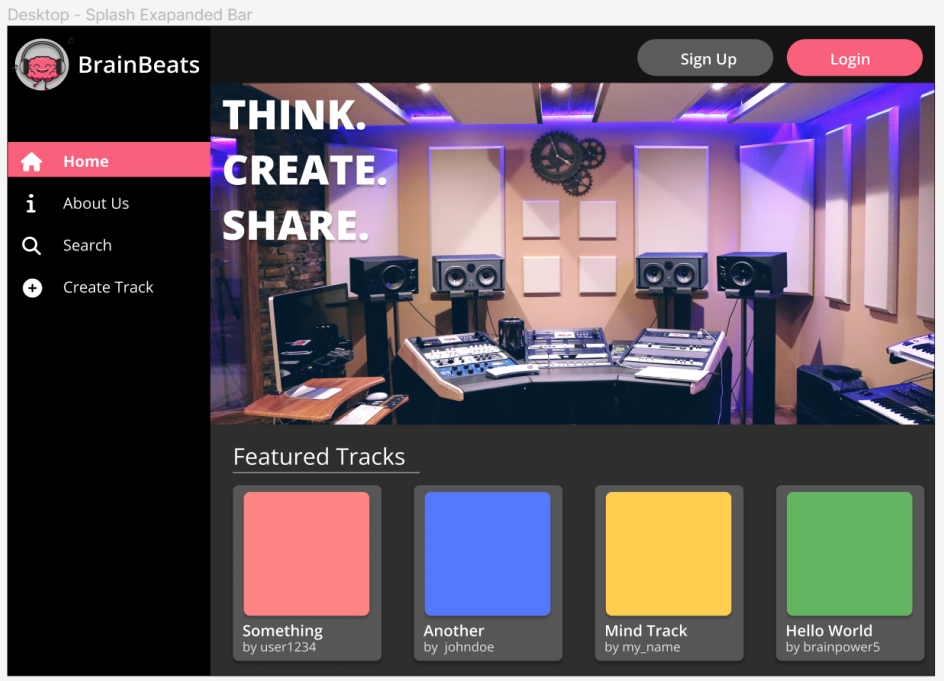

Figure 10.8 Landing Page for Guest/User (with Color) Expanded Bar Example

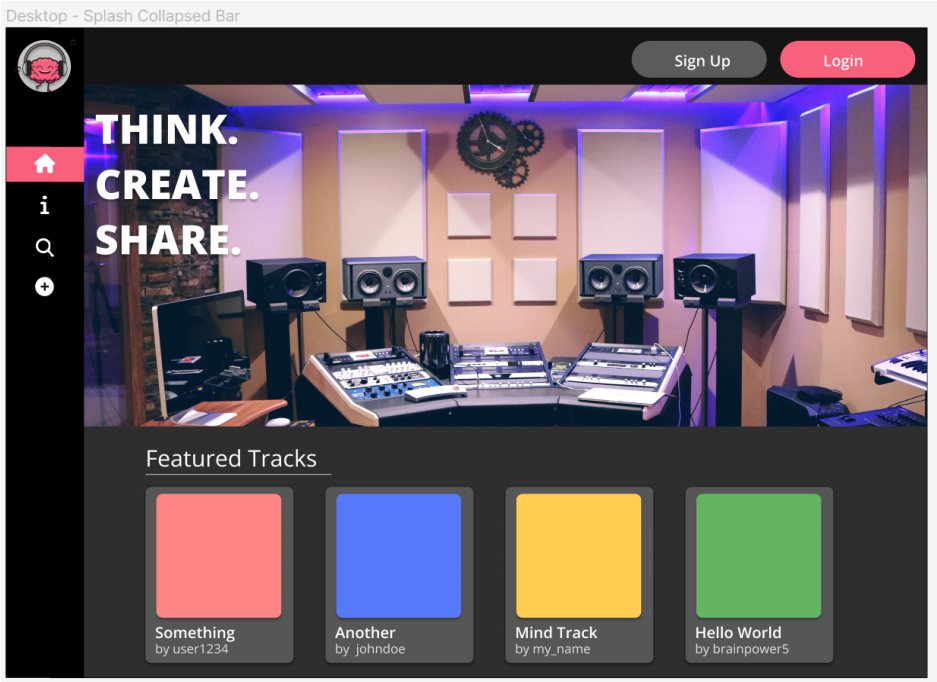

Figure 10.9 Landing Page for Guest/User (with Color) Collapsed Bar Example

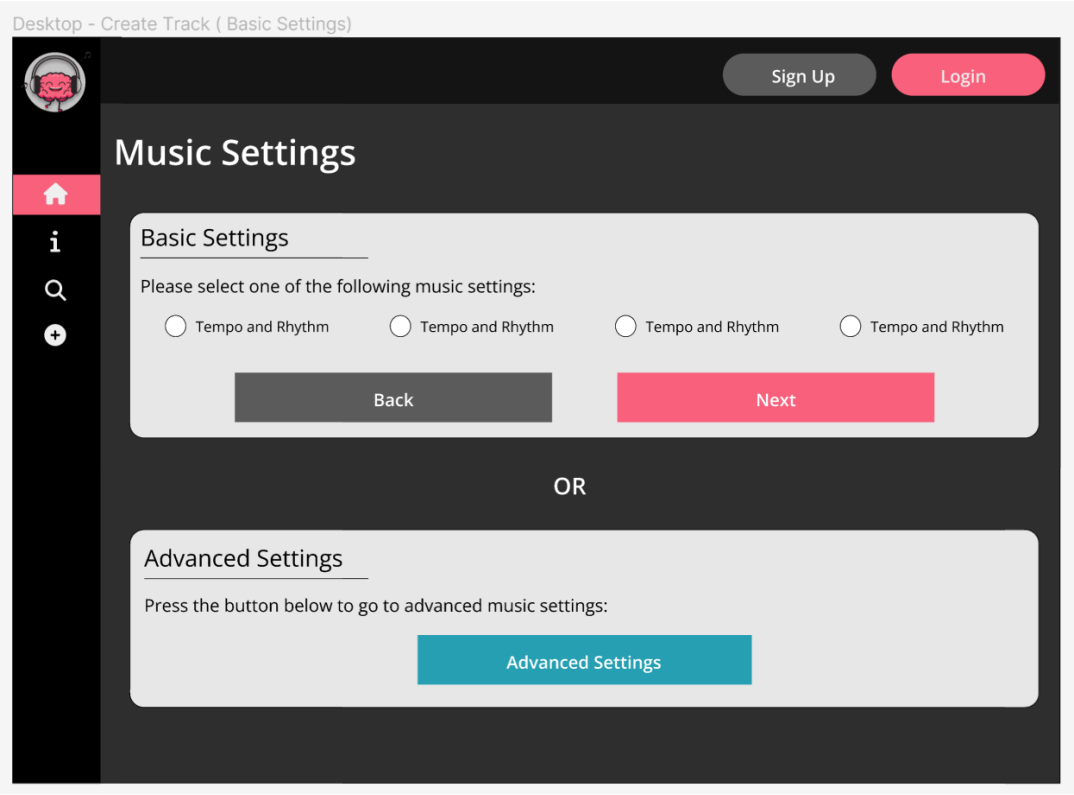

Figure 10.10 Choosing Basic or Advanced Settings (with Color) Example Page

A simple login and sign-up page was designed to be straightforward but also include our logo and some accent colors. The login and sign-up pages will also include a top navigation bar with a back button at the top left corner. A left arrow at the top left of the navigation bar represents a back button that will direct the user to the landing page from the login page (Figure 10.11). The logo at the top left of the navigation bar of the sign-up page will also direct the user back to the landing page (Figure 10.12).

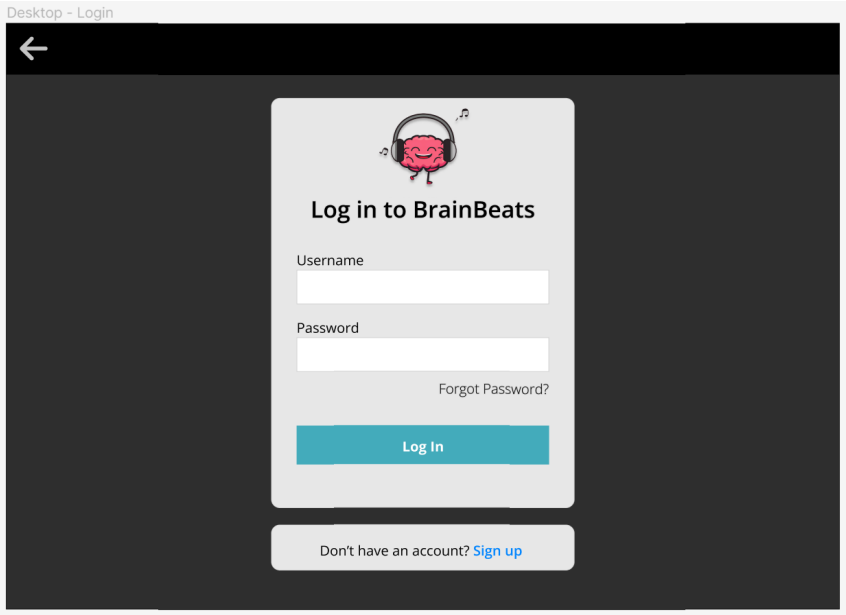

Figure 10.11 Login Page (with Color) Example

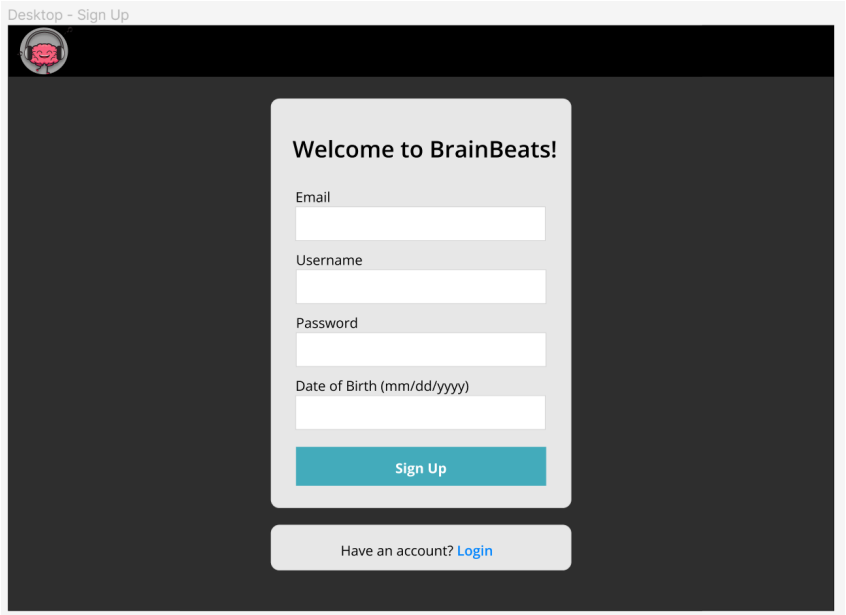

Figure 10.12 Login Page (with Color) Example

The use of color to separate information is also a design principle being used within version four of BrainBeats. An example of the Advanced Music Settings page exhibits the use of colors to separate instruments for easier visibility for the user (Figure 10.13). The use of labels also makes it easy for the user to navigate what settings they are adjusting (Figure 10.13).

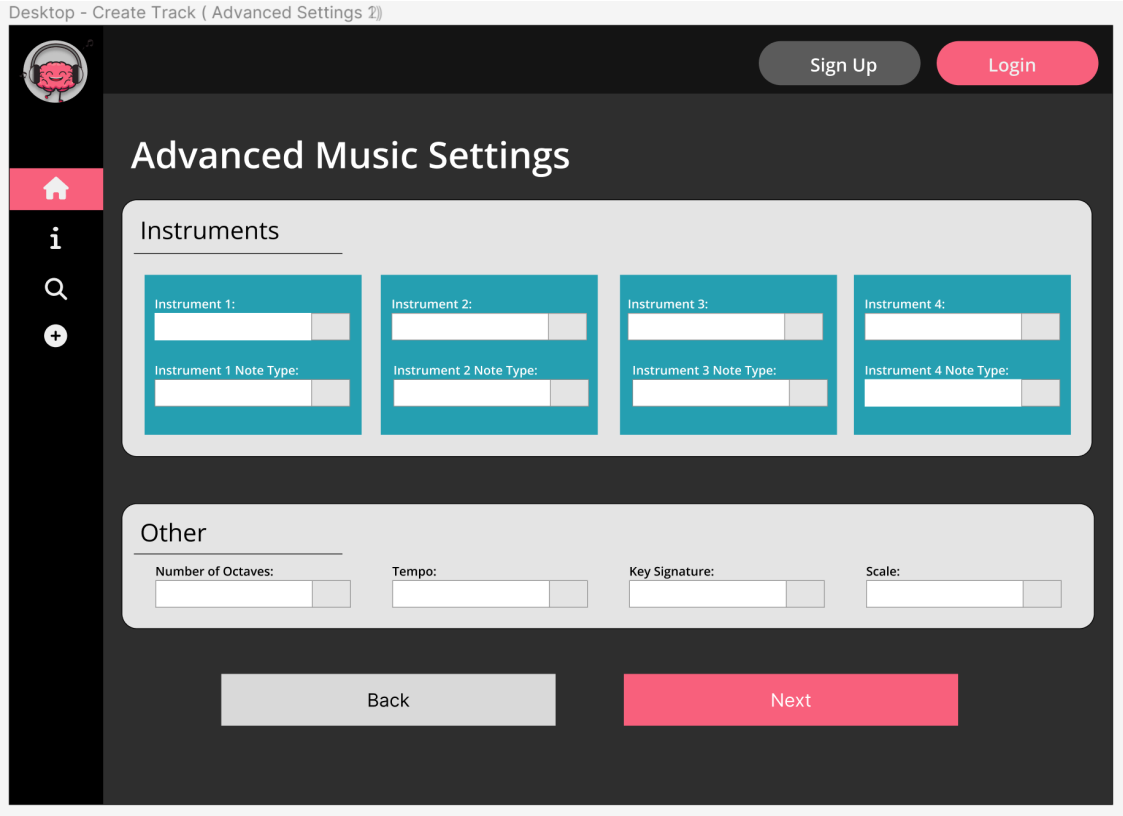

Figure 10.13 Advanced Music Settings Page (with Color) Example

10.2 Database Services

One of the central parts of our project involved establishing a database for the app to use in order to ensure that all of the other aspects of our project could function properly as well as maintain information from the users as well as the songs they have created for other users to have access to within our application.

While the previous team was instructed to make use of MongoDB to conform with standards, the requirements of the project have now been adjusted to ensure that it is maintainable by our sponsor after our team graduates. This results in us having to switch from a MongoDB to a MySQL database. This of course presents some challenges and also will result in us having to reimplement the existing codebase to ensure compatibility.

10.2.1 MongoDB to MySQL 8

In the previous iteration of the project the team had opted to make use of MongoDB as their database management software as opposed to MySQL. This was a decision made by them as a way of attempting to integrate more modern technologies, however, they failed to consider other aspects of maintainability. One of the concerns our sponsor expressed was his ability to make updates and changes to the code after the conclusion of the project as he would then become the primary maintainer. As a result, he made it very clear that it was important to him that we used technologies that he was familiar with. Therefore we made the decision that transferring from MongoDB to MySQL 8 was a necessary step because our sponsor was much more familiar with it and therefore would have a much easier time maintaining it than previously.

10.2.2 Database Schema

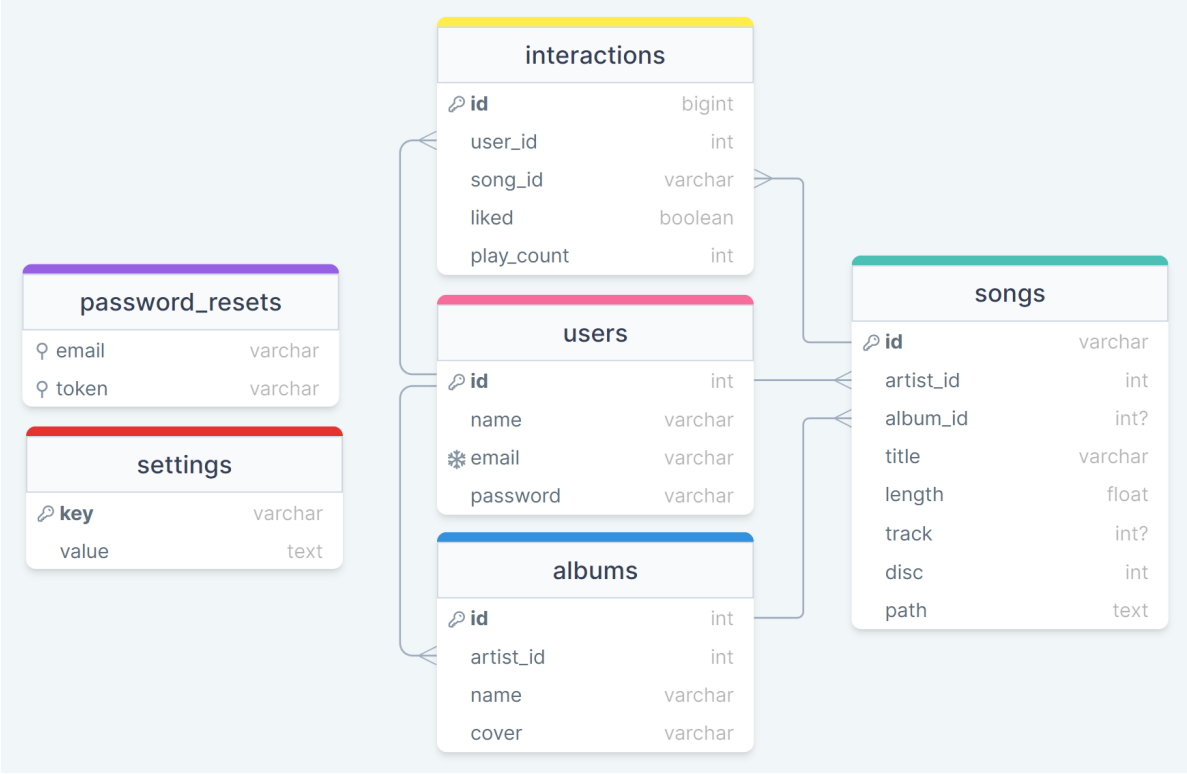

Figure 10.14 Database Schema Diagram for BrainBeats

The center of our schema is the user, which holds basic information about the user and connects them to both their albums, songs, and also their interactions with other user's pieces. The album is simply an organizational item to allow a user to group created

songs together, songs each contain information on title, length, and also information to access the created piece. Finally, interactions allow a user to like a given piece as well as keep track of how often a given user has listened to a particular song creating metrics for a creator to gauge a piece.

10.3 Hosting and Backend Services

10.3.1 Use Case Diagram

The BrainBeats use case diagram specifies the interactions between the user base and the server the application runs on. It describes the actions that the user can take that lead to the database connection. The diagram is split into two major sections which have their own individual interactions with the database, the guest user, and the registered user. Their respective use cases referenced in Figure 11.15 are described as follows:

- Guest User
    - Path to registration
        - If this is followed the Guest User becomes a Registered User and then follows its path.
    - Listening to a public track
    - Searching for public tracks
    - Recording a track
        - Downloading a track
- Registered User
    - Path to Login
    - Option to reset a password in the case that it is forgotten
    - User Verification Process
    - Listening to public tracks and personal private tracks
    - Searching for public tracks

- Recording a track
    - Downloading a track
    - Uploading a recorded track to the database
- Liking a public track
- Creating a playlist of tracks
- Viewing another user's profile
- Editing the personal profile

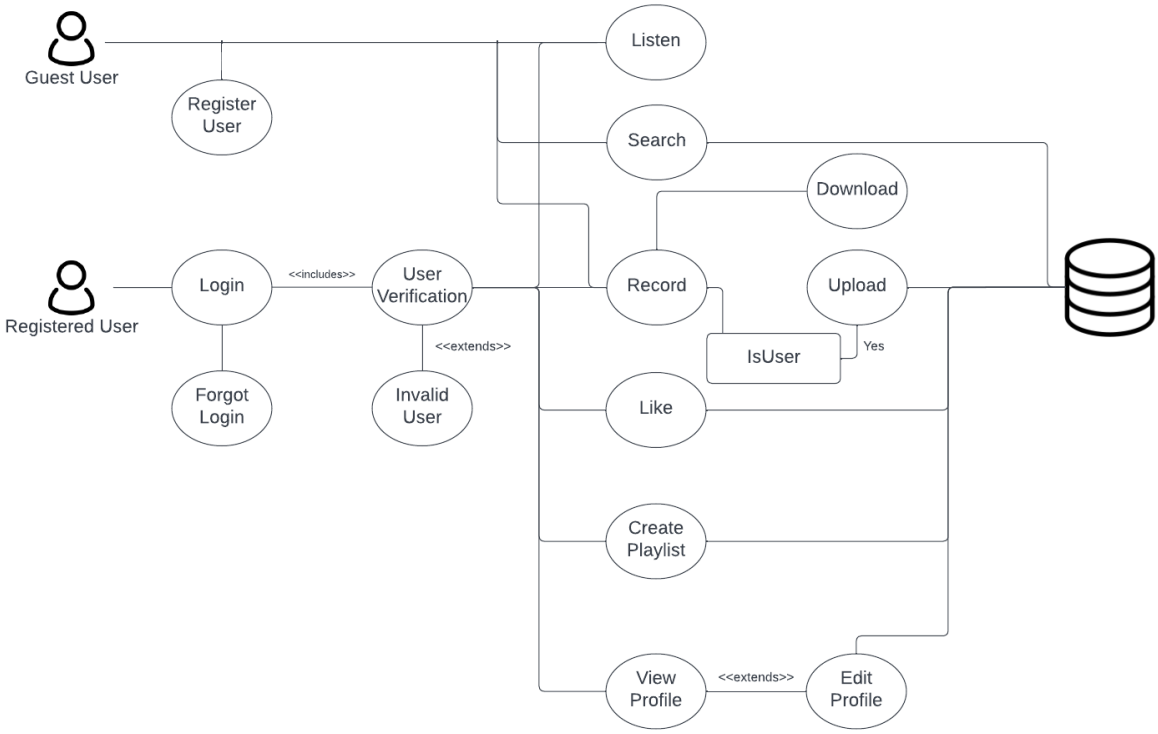

Figure 10.15 BrainBeats Use Case Diagram

10.3.2 - Digital Ocean

One of the central parts of our technical system design is the hosting of our software so that it can be accessed outside of a local network by many users and from a variety of devices. As mentioned in the research section of our document our hosting service is Digital Ocean for a variety of reasons such as ease of use and the familiarity our

project sponsor has with the platform which should ensure that the maintenance process is as simple as possible following the conclusion of this iteration of the project.

One of the issues we had to overcome was the server which was used by the previous team crashing intermittently after we began to update the codebase. We discovered that the previous iteration also had this issue but it had been latent and that the droplet we were using did not have enough resources to host the required technologies, specifically MySQL. As a result of this we decided to upgrade the Digital Ocean droplet plan in order to be able to support the technologies we planned on making use of. This ended up being sufficient and resolved our issues so that our web app could be hosted publicly for access to our user base.

10.3.3 Network Architecture

10.3.3.1 Host

The BrainBeats application will be contained on a Droplet hosted by DigitalOcean. DigitalOcean provides developers with numerous plans to host applications within their cloud infrastructure. The BrainBeats team made this decision as a result of our sponsor’s preference, primarily because of its compatibility with MySQL and general familiarity with the product.

10.3.3.2 Specifications

BrainBeats does not expect heavy user traffic at any given time of the day due to the niche nature of our product as well as the low level of promotion as a public product. However, we still want BrainBeats to be as efficient and responsive for the user as possible. The previous iteration of BrainBeats utilized the Premium Intel Tier One droplet, which is described below. After a discussion with our sponsor, it was decided to upgrade the droplet to the Premium Intel Tier Two level, just to have the overhead if it should be needed.

The BrainBeats droplet implements multiple technologies in order to cooperate between the local development environment, the database, and the frontend. This includes

- Nginx
- Nodejs

- MySQL

The original droplet, Premium Intel Tier One, contained the following specifications:

- The server is based out of New York City
- 1 Intel vCPU
- 1,000 Transfer
- 1 GB Memory
- 25 GB Disk
- The server is built on a Linux with the following distribution:
    - Ubuntu 22.10
    - x64 Architecture

The upgraded droplet, Premium Intel Tier Two, contains the following specifications:

- The server is based out of New York City
- 1 Intel vCPU
- 2,000 Transfer
- 2 GB Memory
- 50 GB Disk
- The server is built on a Linux with the following distribution:
    - Ubuntu 22.10
    - x64 Architecture

10.4 - Scripts

10.4.1 - About Scripts

Scripts in BrainBeats are ways for a user to change the way their brain waves make music. Currently, in the application there is only one script which consists of a lookup

table. The amplitude for a given segment of EEG is mapped to a lookup table, and if the value does not directly map to a pitch and ends up between two pitches, the floor value is taken.

10.4.2 Modularization

In order to allow for scripts to be easily created by other developers, and possibly users, we must define a standard input and output for them. We want this to work in a way that all they need to focus on is the logic for that script, then our system will handle providing the EEG data stream, and processing the midi according to the parameters they provide as an output to their script. For every interval of EEG data, there should be a corresponding interval of MIDI data or rest, if there should not be a note playing.

10.4.2.1 Standard Input

The input for this system will consist of an object which will contain some information for each EEG node the user is recording, with its corresponding data buffer. We would have to monitor and forward that buffer to the scripts in real time, with as little latency as possible.

The input specifications for each node are as follows:

- Name
    - The name of the node, for example, the following string: “T8”.
- Active
    - A boolean which says if a given node is active or not. Helps in the case of a user using different node counts and configurations.
- Data
    - The buffer containing the EEG data for this node.

10.4.2.2 - Standard Output

The output will consist of all of the MIDI parameters that the creator of a script decides to change. Our system will be responsible for taking that output and generating MIDI based on those parameters. These parameters could consist of pitch, note length,

velocity, attack, decay, sustain, release, etc. If a parameter is not provided by the script to our system, it should assign a default value, it must not force a script to make edits to each parameter.

For example, If I make a script that only changes the pitch of notes, my output would be some sort of object with the parameter “pitch”, and some value it should be set to, let's say “A4”. BrainBeats should then be able to replace the default value for the pitch change with this value and play the note.

The output specifications for a MIDI event are as follows:

- Required Input:
    - Pitch, which defines the desired amount of pitch for a given note after conversion from EEG to MIDI.
        - For example, the selection choice A4 represents an A 440 Hz pitch.
- Optional input, all of these values have defaults which are dependent on tempo and the user selected music generation type in the case that a user leaves these settings unaltered. More information on the selection of a generation type is referenced in Section 7.6: Music Generation Options.
    - Length, which determines the length of a note.
    - Velocity, which determines the loudness of a note when performed. This correlates to how hard a key would be struck.
    - Attack, which determines the amount of time it takes for a signal to rise to 100% amplitude from zero.
    - Decay, which determines the amount of time a signal takes to return back to a sustained level from 100% amplitude.
    - Sustain, which determines the sustained amplitude level at which decay will return to from 100% amplitude.
    - Release, which determines the amount of time it takes a signal to return from the sustained amplitude to zero.

10.6 - Coding Conventions

One of the primary goals of BrainBeats v4 is to implement modularization of the code provided to us from the third version of the project. In order to do this, we find it important to implement the following coding conventions to allow for easy expansion and future development of the project:

In terms of casing:

- Constants are written in snake case:
    - const CONST_NUM = 5;
- Variables are written in camel case and should have their proper variable type as defined by TypeScript rather than using typing by inference:
    - number variableNum = 5;
- Classes are written in upper camel case:
    - Class InputValues( ) {}
- Types are written in upper camel case:
    - type IntegersFromThreeToSix = 3 | 4 | 5 | 6;
- Methods are written in camel case:
    - function processInput( ) { }

Each type is defined per the TypeScript conventions for typing in order to maintain code readability and have a fundamental understanding of what specific lines of code are supposed to do.

We are developing with an object oriented approach, any object that can be modularized are defined in a class, for example:

```
Class InputValues {
        inputName:string
        midiPath:string
        …
}
```

10.7 Improvements Upon Previous Technology

10.7.1 Login and Logout

The implementation of JSON Web Tokens was found in the codebase of the project we inherited, however, they did not have full functionality in terms of storage and removal of local storage upon logging out. We found that it was important to implement a feature that allowed users to log out of their accounts to improve the security of our user base, this became especially important with the implementation of revisable scripts in our system. If a user was unable to logout on a publicly accessible device, we find that it could be the case where a user may have their previously created script be altered without their consent. With the implementation of a logout feature came the need to develop methods for removing the JSON Web Token from a user's local storage upon logging out as opposed to expiring after a prolonged period of time. We found that this was a more practical approach for the future development of BrainBeats as well, considering that there is a need for more user security upon expansion of the application's purposes.

10.7.2 Sign Up

Registration for BrainBeats previously implemented a system of collection of an email, username, password, first name, last name, and date of birth. However, their use of a username appeared to be obsolete following registration, because of this we determined there was no longer a need to have a username and we reverted to strictly collecting emails for users to both register and use at login. The previous implementation of BrainBeats collected to date of birth along with a follow up age requirement to use the application, this meant that if you were under the age of 13 you were unable to register for BrainBeats. We found this to be an unnecessary requirement, as there is a potential that a younger person may find interest in

composition, and likewise BrainBeats as a result. Because of this, we removed the age requirement, and since there is no check on a user's age, we removed the collection of date of birth as a whole. This topic is expanded upon in section 10.7.3. This resulted in a clean-up of the database, giving a user a more clear definition with just an email, username, first name, last name, and password.

10.7.3 Abandoned Features

Throughout the process of updating the existing codebase some of the previous features were found to be lacking or no longer contributive within the scope of our final product. As a result, these aspects were edited, replaced, or in some cases removed altogether in pursuit of improving the final result to more closely fit our sponsor's idea of what the project should be able to achieve and how it would function. This section is dedicated to both identifying the features that existed in previous iterations as well as explaining the reason for their removal and what if anything took their place.

In the previous iteration of the project, one of the central features for scripting was the inclusion of an import option for YouTube videos as an alternative to their simplistic text card based system. Ultimately this feature was found to be underused and generally not within the scope of the intended concept for the project. While it was certainly an improvement over the text card system it required the user to essentially either:

a) Find an appropriate video which fit their specifications (usually resulting in a result which couldn't truly showcase their intended script)

b) Create a script in another software and upload it as a YouTube video so that it could be accessed within the application which puts more stress on the user than is truly intended by the project sponsor

While removing this feature was a task assigned to us by our project sponsor we wanted to maintain the existence of a better option for users than a simple text card option that was the alternative so we decided to implement an image importing option which could then be further augmented with text if so desired by the user. This was the chosen solution as it provided even finer control to the user compared to the previous system while also ensuring that the user did not have to turn to outside software to achieve basic intended functionality within the application.

This process was achieved through one of two primary methods. In the primary case, the user is presented with the option to select from a library of stock images provided by Unsplash and their associated API. This provides users with a wide range of options that are readily available and range a wide variety of content and options without having to provide high quality images themselves since that may be out of the range of the average user. The secondary case, however, deals with those that have access to greater resources by allowing the user to upload a custom set of images to then augment and time within the tool to provide full control over the process. This is ideal for those who want to evoke feelings of a particular type or gauge reactions to a certain perhaps regional influence in terms of the EEG data and BrainBeats produced from it by various people.

Both of these potential use cases of course share a common thread in how they resolve the shortcomings of the previous iteration in that they both grant the user greater control by essentially removing the reliance of the tool on outside software either by providing access to images within the application or allowing a user who wants that greater aspect of control to upload images and compile them into a script within the application as opposed to having to rely on other software to achieve this functionality.

10.7.3.2 Music Generation via Lookup Tables

The previous rendition of BrainBeats made use of lookup tables which would essentially read in levels of brain activity and simply assign a music note to a given range by snapping it to fit within the given bounds based on how each note was defined prior to recording. While this was a functional system it lacked the sophistication of generating music based on a person's brain waves and more in terms of playing music if you were able to control the level of brain activity. This was sufficient for their iteration as it was largely focused on transferring the application to a web based system instead of the application they were given but now that we have more time to dedicate to this system it would be nice if possible to upgrade it to be more in line with the vision of our sponsor.

In order to combat this problem our group decided to look into different machine learning approaches to best collect feelings from a user and use that as a basis for composition as opposed to the rudimentary lookup table approach that was currently implemented. This was achieved by using a dataset that we found connecting EEG

data to emotions and then once this was established any number of techniques could be used to convert this information into music representative of the user. This process will be described in more detail in the following sections but essentially this method allows for the compositions to be based upon how the user feels as opposed to how much activity there is in general which is often a much more accurate metric of what would be the desired output given various image based prompts which will be the method of composition in this medium.

10.7.3.3 - Unnecessary User Login Requirements

The previous system which handled user login collected a number of redundant data fields and was making checks which put unnecessary restrictions upon who could make use of the application. The most obvious example of this was the fact that it collected user age as well as an email address for verification which put artificial restrictions on the user base beyond what was necessary. None of the generated music will contain explicit lyrics or other content that would need to be age restricted and the email data despite being a required field to create and use an account didn't have any practical use within the code base beyond email verification of accounts.

These different aspects required different solutions in order to properly resolve them. In terms of the unnecessary data such as age, we simply decided to cut without replacement as they were not relevant and beyond putting artificial restrictions on our user base also presented as potential security concerns as we were holding more personal information than was necessary and would expose users to greater risk should a security exploit arise or be discovered within our program at a future date.

In terms of the email address and user name being required for authentication and login, we decided that it would be best to simplify this process to reduce the number of lost accounts while still maintaining the integrity of each account. We decided to stop using a username and simply base each user's login on their email and an associated passphrase. This still allowed us to make use of email authentication and ensure that a single user cannot make many accounts for themselves and overflow the system while also simplifying the sign-up/login process for new users while simultaneously saving our database from storing excess and unnecessary information within thereby increasing performance compared to the previous iteration of the project.

10.8 Testing

Testing is of course one of the most important aspects of the development process. In this section, we describe our testing methodology as well as documentation of our test cases, containing expected and actual results. This documentation will take the form of a basic table describing the requirements and results of each test case as well as defining its success or failure.

10.8.1 Testing Methods

10.8.1.1 API Testing

This will be done by making use of Postman as detailed in our research section, this technology will allow testing of our HTTP requests even without requiring the frontend and backend to both be fully developed and integrated.

10.8.1.2 Database Testing

The database portion of our program will also need to be tested. Specific functions that we intend to test include the maintenance of data integrity as well as being able to request, access, and write data under our expected level of use. We will test the responsiveness and integrity of our database under this situation.

10.8.1.3 Unit Testing

Throughout developing software, each function within our codebase will be tested using pre-written test cases to ensure compliance prior to integration with other tested sections to isolated individual issues and ensure integration issues that arise are kept to a minimum and not a result of individual subsection’s failures.

10.8.1.4 Integration Testing

This is the final tier of testing we will be doing within our project. Once all the individual portions of our code base have completed unit testing we will begin integrating them and ensure that data is maintained as it moves from section to section and is changed

and manipulated as expected, which will be defined in the documentation in the following section.

10.8.2 Test Case Documentation

This section defines our test cases in greater detail. The form of each test case will be as follows: test number, testing category, testing objective, expected results, and status. Each of the test cases to follow will be executed in the spring semester and will not be declared successful until all the team members agree and conclude that it meets our specifications as listed in the description category of the table. Once the project passes all the test cases the project will then be placed under testing by our sponsor to ensure that it meets his specifications and if there is time will also be tested by a number of potential users we have established within the composition program of the college and have expressed interest in the tool.

Test Number	Testing Category	Testing Objective	Expected Results	Status
1	API	Ensure GET endpoints function as intended and return accurate results.	Using Postman sends a GET request for each API endpoint that has one and ensure that the results are accurate as per documentation and the database.	TBD
2	API	Ensure POST endpoints function as intended and return accurate results.	Using Postman send a POST request for each API endpoint that has one and ensure that the results are accurate as per documentation and the database.	TBD

Test Number	Testing Category	Testing Objective	Expected Results	Status
3	Database	Ensure the database can support large amounts of user data sufficient for at least 500 active users.	The database should be filled to store an equivalent amount of test data for at least 500 users and should still function.	TBD
4	Database	Ensure the database does not have unreasonably long runtimes under the expected load.	The database should be tested with requests which are made of varying sizes configurable with HTTP headers and should be found to have a maximum response time of one second.	TBD
5	Unit	Ensure each function and component for the front/backend operates as intended individually throughout the application.	When each function is developed it should be tested using dummy data for each possible state and ensured to be returning correct values, implemented in an appropriate medium as needed.	TBD
6	Integration	Ensure the web application connects and interfaces with the API appropriately	Manual testing of frontend API based features including: search, track creation, and interactions.	TBD

Test Number	Testing Category	Testing Objective	Expected Results	Status
7	Integration	Ensure that the frontend application properly interfaces with the backend and MySQL instance.	Automated testing upon build and deployment will ensure that the backend and database will function properly.	TBD
8	Integration	Ensure EEG data is collected from the headset properly.	Log into the application and ensure that EEG data is returned.	TBD
9	Integration	Ensure the production environment functions properly and that the UI is connected.	Deploy the project to the production server and manually test functionality of the UI and API requests to ensure proper and complete functionality.	TBD

11 Consultants

As a part of the project we decided that it would be beneficial to reach out to individuals who had more experience in their technologies and composition techniques than the members of our group. As a result of this, we decided that we should communicate with several professors at the university, especially those with experience in machine learning. In addition, we also wanted to reach out to some professors in the music department for assistance in deciding what the most important aspects and features should be included so that the tool would be most useful and easily applicable for composers in the music department.

We considered this to be a worthwhile endeavor as, despite all of our group members having some experience in a wide variety of software applications and even each of us having some familiarity with composition, we still lacked a depth of knowledge in the relevant subject matter compared to the professors we consulted on this with. By having these people as available contacts and resources of information we hope to greatly ease the process of implementing the software as well as ensuring it complies with those who make up our intended users.

We would like to preface this section with a thank you to the faculty who took time out of their schedules to share an interest in a project we believe helps to advance both the fields of computer science and composition. We think that further communication between these two fields will open doors to many new innovations in music.

11.1 UCF Professors

UCF has a large ensemble of faculty to reach out to, each is very experienced in their own field and therefore are excellent resources for information regarding both the machine learning and composition techniques that our team sought out to implement. The BrainBeats team maintained conversation with professors, gaining insight into these techniques, the faculty that we gained fantastic advice from are referenced below, separated into their respective fields and what they aided us with.

11.1.1 Machine Learning Consultants

As mentioned above one of the aspects we wanted to receive consultations on was machine learning which we planned to use for the music generation. In the following

subheading, each professor we talk to is listed along with their specialties and what information they provided to us in an effort to assist with the implementation of our project and ensure that we were taking the correct approach.

Dr. Liqiang Wang is a professor who invests research efforts into big data computing, According to his UCF homepage, “Professor Wang’s research focuses on big data computing and analytics techniques in the following aspects: (1) improving accuracy and security of big data analysis models; (2) optimizing performance and scalability of big data processing and parallel computing systems, including multi-threading, HPC, Cloud and GPU platforms; (3) using program analysis and deep learning techniques to detect and prevent programming errors and execution anomaly in big data and/or parallel programs.” Because EEG and MIDI are both relatively large files, there is a need to know how to handle big data in computation with machine learning. Because of this, we have kept open communication with Dr. Wang, who has expressed a general interest in the project. His experience directly applies to processing out information in an efficient manner and ensuring that it is optimized as much as possible to limit our required resources for training.

Another UCF professor we consulted with was Dr. Lotzi Bölöni who is the co-director of the AI Things Laboratory and has also written a number of publications covering concepts such as robotics, deep reinforcement learning, vision-based end-to-end learning, and computer vision. Dr. Bölöni’s expertise in machine learning has lended a great hand in helping us to understand the methods of building an effective deep learning model for our purposes. Dr. Bölöni provided us with knowledge on the effective ways to create a neural network with specific parameters and gave us very useful information regarding potential datasets. In regards to this, he presented our team with machine learning models which converted EEG into motor functioning, which led us to understand how these projects collected their data and also showed us where to go to find publicly available datasets for machine learning.

11.1.2 Composition Consultants

In addition to this it was also important for us to consult with a number of music professors as our application has an obvious audience with composers and other musicians. As a result of this, we needed to confirm that the software we are developing meets the requirements of these individuals. The difficulty with this, however, was quickly evident as the technical experience of these individuals was

much less than those we had spoken to in the CS department as is to be expected. Therefore a large part of these conversations was attempting to quantify their abstract requests into actionable and testable requirements which we could use to ensure that our final project would be useful not only to them but also to composers in general assuming they would have similar requests. Each of the professors also provided us with contacts in their composition classes so that we could conduct our research study which is described in its own section later in this document. The following subsections will define what each of the professors did to help us achieve our resource both in defining requirements and distributing our research survey so we could find appropriate participants.

Dr. Scott Lubaroff largely provided us with research participants as well as referring us to Professor Burtzos as he believed that his experiences would be more directly applicable to our project. So despite not being able to provide us with much assistance in the early stages of our project that we are currently in, he provided us with the correct contacts to gather the information that were necessary for us as well as giving us access to his student distribution so that we could collect research participants.

Dr. Alexander Burtzos is the Endowed Chair of Composition Studies and a professor at UCF, where he teaches composition, orchestration, film scoring, video game scoring, and music technology. As a result, his specific skill set was nearly a perfect match for our project considering that he specialized in composition studies and also conveniently had prior experience with music technologies which ensured that his understanding of general technical terms was greater than the average non-technical professor at UCF. Our team has had conversations with him regarding the composition program’s interest in our project and how we can develop the project to cater to the needs of a composer. His hands-on work with the composition team gave us a connection to students who have an interest in the BrainBeats product, expanding our network and providing us with more education in the field of composition.

12 Project Summary and Conclusions

BrainBeats serves as a web application providing a new method of creating and sharing music. Users of all experience levels can visit the website as a guest or registered user to create music using different methods of music generation through EEG data. Using a compatible EEG headset, BrainBeats records and transforms brain waves from an EEG headset to music influenced by user-selected settings. After users have recorded their music tracks, registered users can upload and share tracks with other BrainBeats registered users. Although guest users would not be able to share and upload tracks, they would still have the option to download their tracks onto their own devices.

Throughout the implementation of BrainBeats version four, the entire previous version's project will be remodeled into TypeScript and modularized for future versions of BrainBeats to improve and expand upon. The modularization of version 3 will include a remodel of the user interface to provide a more modern, user-friendly, and accessible display of the application following the principles of accessibility.

Scripting rework, standardization of input and output through MIDI, and impactful modularization of the backend infrastructure is prioritized to allow for easy implementation of new scripts in the future development of the application.

Alongside modularization and redoing of the user interface, BrainBeats version four also aims to create a new means of music generation for its users involving machine learning. A study was designed to collect data for the machine learning model but due to time constraints and a lack of currently available participants, the BrainBeats version four will be using data collected from previous studies done online to train a machine learning model. The study we designed can still be found in our documentation for future versions of BrainBeats who may consider implementing a machine learning model consisting of data from the designed study.

BrainBeats version four source code, documentation, and instructions will be added onto the pre-existing GitHub created by BrainBeats's sponsor Dr. Richard Leinecker at https://github.com/RickLeinecker/BrainBeatsv4.

13 References

Chang, Won-Du et al. “Detection of eye blink artifacts from single prefrontal channel electroencephalogram.” *Computer methods and programs in biomedicine* vol. 124 (2016): 19-30. doi:10.1016/j.cmpb.2015.10.011

Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. “MEG and EEG data analysis with MNE-Python.” *Frontiers in Neuroscience*, *7*(267):1–13, 2013. doi:10.3389/fnins.2013.00267.

Vernon J. Lawhern, Amelia J. Solon, Nicholas R. Waytowich, Stephen M. Gordon, Chou P. Hung, Brent J. Lance. “EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces.” *J. Neural Eng*, 1-15, 2016. arXiv:1611.08024

Sharma, Devansh. "Image Classification Using Convolutional Neural Networks: A step by step guide." *Analytics Vidhya*, 11 Jan. 2021, www.analyticsvidhya.com/blog/2021/01/image-classification-using-convolutional-neural-networks-a-step-by-step-guide/.

Brownlee, Jason. "How Do Convolutional Layers Work in Deep Learning Neural Networks?" *Machine Learning Mastery*, 17 Apr. 2020, machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/.

IBM Cloud Education. "Recurrent Neural Networks." *IBM*, 14 Sept. 2020, www.ibm.com/cloud/learn/recurrent-neural-networks#toc-types-of-r-q1VkG6gm.

Chollet, François et al. "Keras." 2015. https://keras.io

Hsiao-Tzu Hung, Joann Ching, Seungheon Doh, Nabin Kim, Juhan Nam, and Yi-Hsuan Yang. “EMOPIA: A Multi-Modal Pop Piano Dataset For Emotion Recognition and Emotion-based Music Generation.” *International Society for Music Information Retrieval*, 1-8, 2021. arXiv:2108.01374

”DEAP: A Database for Emotion Analysis using Physiological Signals”, S. Koelstra, C. Muehl, M. Soleymani, J.-S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, I. Patras, IEEE Transactions on Affective Computing, vol. 3, no. 1, pp. 18-31, 2012

IBM Cloud Education. “Lamp Stack Explained: Master the Basics and Get Started Quickly.” *IBM*, 9 May 2019, https://www.ibm.com/cloud/learn/lamp-stack-explained.

Smith, Ernie. “LAMP stack history: It's everywhere, but developers hate it” *Tedium,* 1 Sept. 2021, https://tedium.co/2021/09/01/lamp-stack-php-mysql-apache-history/.

“What is The MERN stack?” *MongoDB,* https://www.mongodb.com/mern-stack. Accessed 22 Nov. 2022.

“MERN Stack: Explained” *We Are Capicua,* https://wearecapicua.com/mern-stack. Accessed 22 Nov. 2022

Krishnan, Abhimanyu. “TypeScript vs JavaScript: Which is Best in 2022” *Hackr.io*, 15 Nov. 2022, https://hackr.io/blog/typescript-vs-javascript.

“Tutorial: Intro to React” *React,* https://reactjs.org/tutorial/tutorial.html#what-is-react. Accessed 22 Nov 2022.

“Documentation - React” *TypeScript*, 28 Nov. 2022, https://www.typescriptlang.org/docs/handbook/react.html.

“Bootstrap” *React*, https://react-bootstrap.github.io/. Accessed 22 Nov 2022.

“Setup With React” *Font Awesome*, https://fontawesome.com/docs/web/use-with/react/. Accessed 22 Nov 2022.

Henry, Shawn Lawton, and Wayne Dick. “WCAG 2.1 at a Glance” *Web Accessibility Initiative (WAI)*, 5 June 2018, https://www.w3.org/WAI/standards-guidelines/wcag/glance/.

Zola, Andrew. “What is hashing and how does it work?” *Tech Target*, 3 June 2021, https://www.techtarget.com/searchdatamanagement/definition/hashing#:~:text=Hashing%20is%20the%20process%20of,the%20implementation%20of%20hash%20tables.

Nohe, Patrick. “The difference between Encryption, Hashing and Salting” *The SSL Store*, 24 March 2021, https://www.thesslstore.com/blog/difference-encryption-hashing-salting/.

Nidecki, Tomasz Andrzej. “Secure coding practices – the three key principles” *Acunetix*, 29 Nov. 2021, https://www.acunetix.com/blog/web-security-zone/secure-coding-practices/.

“Side-channel attack”, *Wikipedia*, 21 Oct. 2022, https://en.wikipedia.org/wiki/Side-channel_attack.

International Federation of Clinical Neurophysiology, et al. Clinical Neurophysiology. Vol. 145, Elsevier Science B.V.

“Cyton Specs: OpenBCI Documentation”, *OpenBCI Documentation RSS*, 26 July 2022, https://docs.openbci.com/Cyton/CytonSpecs/.

“Cyton Data Format: OpenBCI Documentation”, *OpenBCI Documentation RSS*, 27 July 2021, https://docs.openbci.com/Cyton/CytonDataFormat/.

“Ganglion Specs: OpenBCI Documentation”, *OpenBCI Documentation RSS*, 26 July 2022, https://docs.openbci.com/Ganglion/GanglionSpecs/.

“Ganglion Data Format: OpenBCI Documentation”, *OpenBCI Documentation RSS*, 27 July 2021, https://docs.openbci.com/Ganglion/GanglionDataFormat/.

“Ultracortex Mark IV: OpenBCI Documentation”, *OpenBCI Documentation RSS*, 28 Sept. 2022, https://docs.openbci.com/AddOns/Headwear/MarkIV/.

“OpenBCI EEG Headband Kit Guide: OpenBCI Documentation”, *OpenBCI Documentation RSS*, 12 July 2022, https://docs.openbci.com/AddOns/Headwear/HeadBand/.

“Electrode Cap Getting Started Guide: OpenBCI Documentation”, *OpenBCI Documentation RSS*, 27 Oct. 2022, https://docs.openbci.com/AddOns/Headwear/ElectrodeCap/.

“Gelfree Electrode Cap Guide: OpenBCI Documentation”, *OpenBCI Documentation RSS*, 20 Jan. 2022, https://docs.openbci.com/AddOns/Headwear/GelfreeElectrodeCap/.

Back, David. “Standard MIDI-File Format Spec. 1.1, Updated.” Standard MIDI File Format, Updated, http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html.

"Symmetric Encryption." *SSL2Buy*, www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences.

S. Katsigiannis, N. Ramzan, “DREAMER: A Database for Emotion Recognition Through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices,” IEEE Journal of Biomedical and Health Informatics, vol. 22, no. 1, pp. 98-107, Jan. 2018. doi: 10.1109/JBHI.2017.2688239