Capstone Final Project Report

Tempo: A Personalized Audio Experience

ECE 4991

Major Design Experience: Capstone Design

Department of Electrical and Computer Engineering
University of Virginia

Team Tempo

Joey Cohen, Bella Heintges, Thomas Keathley, Michelle Monge, Naomi Solomon
December 5, 2024

Advisor

Professor Adam Barnes, Department of Electrical and Computer Engineering

Table of Contents

Table of Figures

Figure 1: Distribution of music industry revenue in the United States.

Figure 2: Detailed final enclosure schematic

Figure 3: Functional flow diagram of the speaker hardware

Figure 4: Schematic diagram of PCB revision 2

Figure 5: External 5V DC-DC buck converter module topology [16]

Figure 6: KiCad board layout of PCB revision 2

Figure 7: 3D model of PCB revision 2

Figure 8: OSHPark board design of PCB revision 2

Figure 9: Power base plate top and side views

Figure 10: TPA3116D2 audio amplifier board layout.

Figure 11: Speaker audio and controls connections.

Figure 12: PCM1794 evaluation board layout.

Figure 13: Audio/BT base plate design.

Figure 14: Raspberry Pi 4 Model B board topology.

Statement of Work

Joey Cohen: There were various components of the project I contributed to. Initially, I designed all the wireframes for the application which provided a template for the app was built on. I also created and set up the GitHub which stores the XCode project, along with ensuring all members of the team could successfully gain access to the codebase and make edits on their personal computers. Additionally, I set up our Firebase, Spotify, and OpenAI accounts so we could access and use their API functionalities through our app. As for authentication, I created the LoginView, QuestionnaireOneView, and QuestionnaireTwoView (and all the helper views that went along with them) which integrated both the SpotifyWeb API and SpotifyiOS API to successfully login the user and exchange the proper tokens. I also integrated the Firebase backend with a server to authenticate a user through Firebase once they were logged into Spotify. This required developing a Firebase Function which serves as the backend to handle the token exchange with Spotify. This also came with handling the exchange of data for these views, obtaining and storing the user's profile picture, name, username, privacy settings, and profile preferences. Additionally, I created the user interface for the ListeningView, along with obtaining random songs from a user's Spotify library for initial testing before the RecommendationHandler was implemented. I also developed the playback functionality to allow music to play from Spotify through our app. Next, I created the SelfProfileView, obtaining data from Spotify, storing it on Firebase, and displaying it on the profile view. All of these steps had iterations of debugging and testing which were all done iteratively throughout the development process. For the final report, I wrote the software sections for the engineering insights and future work sections. I also reviewed and revised various sections throughout the paper to ensure proper grammar and descriptions of our design.

Bella Heintges: I completed practically everything related to the design and fabrication of the speaker itself. I chose and budgeted every component related to each subsystem. The only component I did not select was the Bluetooth board. I designed, organized, routed, ordered, and modeled the first PCB iteration. Further, I designed, organized, routed, ordered, and modeled the second PCB iteration. I set up the Bluetooth board to be functional, even though it was not my job. I ordered and wired the audio amplifier board and both drivers. I tested and made these components functional. I wired this system to the Bluetooth board and, again, tested for functionality. I determined the internal arrangement of all components. I designed, planned in SolidWorks, and then 3D printed five revisions of the base plate designs. I measured every component in this process. I mounted the components on the boards and then tested again. I kept track of every component, every website, every datasheet, my other group members, and contributed to every deliverable. I connected the final hardware together and proved the functionality of the entirety of our Capstone project. In terms of the proposal, I wrote everything related to the speaker. For the midterm presentation, again, I wrote everything related to the speaker, aside from two slides about the enclosure. For this report, I organized it and wrote the following: the abstract, background, project description for the internal speaker and all overall

objectives/specifications, all technical design decisions for every subsystem (power, audio, controls, mechanical), internal component speaker test plans, all of the speaker's design and manufacturing constraints, all hardware external standards, a discussion of the hardware-related timeline, all the insights related to the physical speaker, and the future work related to the physical speaker. Ultimately, I am responsible for the speaker working and for our work being included in this document.

Thomas Keathley: I worked on the hardware portion of this project. The original scope of the project included several dropped lines of work. Originally, I spent time researching bluetooth board options. I was responsible for finding and ordering the actual bluetooth used in the project as well as a backup. But my contributions early on to the other hardware elements ende up being moot. As the scope shrank I primarily became in charge of the enclosure engineering. I designed and modeled four separate iterations of the speaker box in Solidworks as well as. I cut and shaved all of the pieces, spent a considerable amount of time hand-fitting boards, glued and clamped everything together, rebuilt it after the top panel broke. I spent a considerable amount of time experimenting with different audio configurations in tools such as WinISD and Unibox. This also came along with a few miscellaneous tasks here in there such as getting professional scale prints done, spent a month trying to get the proper development software for the alternate bluetooth boards from either the vendor or Qualcomm (neither did), and finally provided equipment for testing.

Michelle Monge: I worked with the software team of the project, personally responsible for managing and debugging the *Spotify Integration* and *Authentication* branches within Github. Within the branches, designed, debugged, and implemented all relevant coding files related to the recommendation algorithm, compatibility algorithm and integration of algorithms with existing code such as: *Recommendation handler, Models, Chat View, Main Tab View, Compatibility View, Compatibility Manager, Compatibility Algorithm, Compatibility List View, Compatibility Row View, LineView, Data, WebView, Spotify Service.* Within each coding file, made sure to comment on relevant information for clarity for all members. Also responsible for integrating all branches together in terms of functionality across the app, merging the backend and frontend. This meant checking code across over 40 files within each branch and scanning for possible errors and misaligned coding practices such as conflicting definitions and invalid redeclarations. Also responsible for updating the github periodically where all of the software team could access. Additionally, made sure to put any errors into a code sharing document and created a five phased test plan for integration of all branches to reduce errors. Additionally, simulated and tested the final version of the app on a personal device noting areas for improvement and strengths and recording videos of simulations to send to group members. Specifically for the recommendations, tested various prompts to observe functionality such as a specific entry like 'sad country drive' to more ambiguous inputs such as 'forest' to see how the recommendation algorithm would react and if the result was up to our expectations of delivering a functional output. For example, if the input prompt contained the word sad, I would see if the

results only contained the word sad in the songs and did not provide a diverse output of songs. If the outputs were disappointing, the prompted role of ChatGPT would be changed. For the final report, mainly wrote and revised: *abstract, iOS objectives, iOS technical specifications, iOS application logistics across the various views, integrated product, software physical constraints, tools utilized, test plan for software, societal impact, final results with functionality, discussion of set criteria, engineering insights, future work and the appendix*. Additionally, I went in to proofread the entire document, making sure formatting and grammatical errors were taken care of to the best of my ability. Lastly, I made sure to provide a link to the Github repository, where all relevant, up to date code is within the Authentication branch.

Naomi Solomon: I worked on the software portion of our capstone project, focusing on the iOS application and its key features. I developed the recommendation handler, which integrates OpenAI API with Spotify API to deliver personalized music recommendations based on user prompts. This was rigorously tested for edge cases ensuring it can respond to a variety of inputs. I also refined the structure of the prompts sent to ChatGPT, tailoring them to extract the most accurate and meaningful results for creating the personalized playlists. Early in the project, I manually generated Spotify tokens to enable authentication and API calls before the process was automated, ensuring the app could interact with Spotify seamlessly during development. Additionally I created a ChatView and Song List View as initial testing interfaces for the recommendation handler. These views allowed us to test and validate the functionality of the recommendation system by displaying song outputs in a straightforward format. While these initial views prioritized functionality, they were later on refined to align with the overall aesthetics of the app, ensuring they seamlessly integrated with the final design. In addition I worked on building the Compatibility Web, the central feature of the community tab. The web dynamically visualizes users' musical compatibility with their friends, where the distance of lines connecting nodes corresponds to the similarity percentage in music taste. I worked on implementing this functionality using SwiftUI's Geometry Reader, which enabled me to accurately calculate the positions of nodes and draw the connecting lines. I implemented a radial layout where the position and line length were determined by a combination of angles and compatibility scores. Each user's node was represented by their profile image, dynamically retrieved and placed based on their compatibility score. Additionally, I contributed to the code sharing document that was used with Michelle Monge as we worked on debugging and testing when we couldn't physically meet. For the final report, I authored/revised the Project Description related to the iOS application, as well as sections on External Standards, Intellectual Property Issues, and the Timeline. I also revised other parts of the report to ensure clarity, accuracy, and consistency. My work spanned feature development, testing, debugging, integration, and documentation, ensuring the Tempo app delivered a polished and user-friendly experience that met the technical and functional goals of the project.

1.   Abstract

Our project integrates a specialized iOS application with an interactive Bluetooth speaker to elevate music consumption with combining personalized music recommendations and social connectivity. Using Spotify's Web API and advanced algorithms from OpenAI, the app generates a curated music stream, provides compatibility scores for social interaction, and allows users to explore shared musical interests. The Bluetooth speaker complements the app by delivering high-quality audio, enabling on-device volume control and visualization of app-related visuals. Designed with an emphasis on high audio fidelity, user engagement, and regulatory compliance, this solution bridges technology and social interaction to improve the modern music experience—helping users find their own Tempo.

2.   Background

The following sections outline the foundational elements of our project, including an overview of audio streaming and the Spotify API, the rise of recommender systems, comparisons with similar projects, and how our academic experiences have equipped us to develop this innovative solution.

2.1.   Audio Streaming and Spotify API

Society is currently experiencing a significant period of digitization, with many physical systems and artifacts becoming increasingly accessible online. Driven by this trend, the music industry has undergone substantial transformations over the years, with technological advancements leading to a widespread shift in consumer behaviors. This shift has been largely driven by the rise of music streaming platforms, which have revolutionized how artists and consumers engage with, access, and discover music.

According to the IFPI's Global Music Report, the music industry generated $28.6 billion in total revenue in 2023, with digital streaming contributing 67.3% of this figure [1]. The transition to publicly digitizing audio for user playback has then significantly transformed the industry's earnings in the United States, where streaming accounts for 84% of total revenue, while physical sales represent only 11% [2].
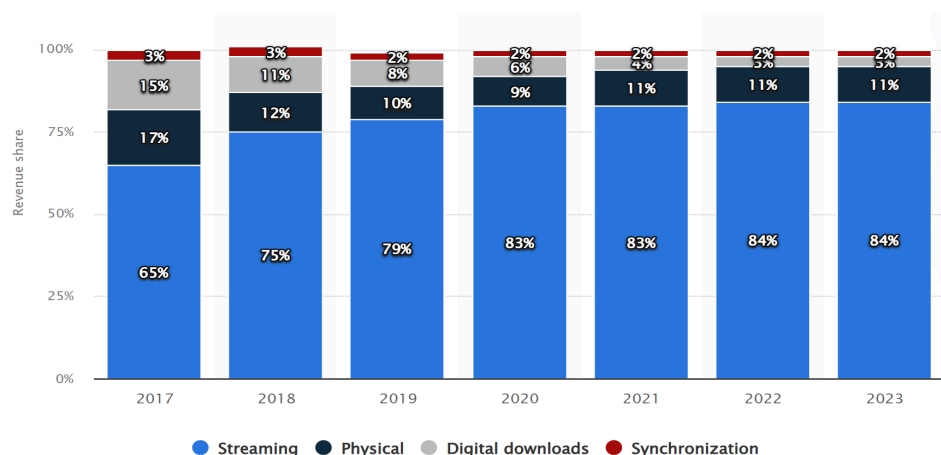
Figure 1: Distribution of music industry revenue in the United States from 2017 to 2023 [2]

Among these streaming platforms, Spotify dominates the market with over 90 million paid music subscribers [3] and about a 30% share [4], double that of its closest competitors. Spotify's overall user base is primarily comprised of Millennials and Generation Z, who together account for over 62%. Being aware of society's reliance on streaming platforms and specifically Spotify's dominance, we recognized Spotify's role corresponding to the ongoing growth of the industry, and became interested in identifying opportunities to address existing challenges to better serve this specific demographic.

While the act of streaming involves the same process across all platforms, Spotify differentiates itself from its competitors through its exhaustive collection of data regarding user history and preferences. Spotify's algorithms consider each listener's searching, listening, skipping, and saving habits, along with personal information such as geographic location and age. These statistics aid in curating recommendations, as dense algorithms use recommender systems to tailor audio to each listener's unique taste. The user data collected by Spotify is accessible via the Spotify Web Application Programming Interface (API) [5], which enables the creation of applications capable of interacting directly with the service. This API also allows for the retrieval of metadata, creation and management of playlists, and playback control. Given this, we became committed in using the capabilities of the Spotify Web API, and focused on developing a product tailored to Spotify users that addresses current limitations in Spotify's offerings, with a focus on user curation. Overall, we became interested in developing a solution that elevated music recommendations, improved social connectivity among users, and integrated a physical device for effortless playback.

2.2.   Similar Projects and Comparison

Our project integrates several innovative features that distinguish it from existing solutions on the market. While many applications and platforms offer separate services such as music recommendations, social connectivity, and audio playback through a physical device, no current solution combines all three into a cohesive physical system focused on user engagement and personalization. Our project also addresses common user complaints about Spotify as we aim to integrate a more seamless recommendation process, alongside an accessible interface designed to increase social connectivity.

Existing audio streaming platforms—such as Spotify, Apple Music, and YouTube Music—mainly deliver music recommendations based on user listening habits. Our system does not depend on user listening as our approach introduces the concept of keyword-based recommendations powered by OpenAI. Users have expressed frustration with Spotify's repetitive suggestions, often prioritizing recently played tracks rather than promoting the discovery of new artists. As former Spotify subscriber Chayka [6] observed, "The platform interface has gradually made it harder to find the music I want to listen to. With the latest app updates, I'd had enough." Unlike current platforms, our priority is safeguarding user autonomy by allowing users to request specific audio queues tailored to their needs.

In addition to a more cohesive recommendation process, social connectivity is another area where our project stands out. While platforms like Spotify offer basic social features such as collaborative playlists and user-following options, these features are limited in accessibility, scope, and transparency. There are no widely available systems that allow users to compare compatibility with friends or explore shared musical interests through meaningful metrics all year round. Our application then introduces a community-driven model with the Compatibility Web, allowing users to see five of their friends compatibility scores in relation to shared songs in the users library.

When considering the physical product associated with our Capstone project, it is important to note that no speakers currently exist with a smart application or music recommendation as a central feature. While some smart speakers—such as the Amazon Echo or Google Nest—allow users to stream music, these products do not integrate directly with recommendation systems or aim to offer a curated visual user experience. Few speaker designs incorporate visuals that correspond to the audio playback methodology, and none implement a microcontroller for additional processing. Like the application, our physical device upgrades musical experiences by displaying a looped video corresponding to the music recommendation process. Many current devices lack the ability to display visuals, and none directly incorporate an application to guide the audio recommendation process.

While current platforms fall short in their offerings, academic projects in this field also highlight gaps that our project aims to address. Some student projects have explored audio analyzers or microcontroller-based programming, but these initiatives are often limited in scope. The projects typically focus on a single feature, such as bettering algorithm development [7] or hardware design [8], and refrain from integrating all elements into a comprehensive, cohesive system ready for users to access.

Ultimately, our Capstone project will differentiate itself from current solutions by:

- Integrating an application to personalize an audio queue alongside a physical speaker.
- Incorporating visuals on an LCD display that correspond to the application.
- Permitting a user to directly request a curated stream of audio based on prompts.
- Implementing a social media interface that promotes connecting with friends online to share and compare music.
- Providing recent, relevant listening statistics alongside clear and user-friendly privacy controls.

The unique combination of features guiding our Capstone project then creates a dynamic and engaging music-listening experience that addresses user needs and thus improves music discovery.

2.3.   Relevant Coursework and Experience

Our team was confident in the feasibility of our project, drawing on our completed coursework and personal experience in application development, wiring, embedded system design, and audio enclosure design.

All team members have completed the *ECE Fundamental*s course series, which included projects focused on displaying signals from audio inputs and transmitting signals through external analog inputs. Additionally, some members have taken courses such as *Digital Signal Processing* and *Analog Integrated Circuits*, providing essential knowledge in analog and signal processing. Moreover, two team members completed the *VLSI Design & Lab* course, where they gained hands-on experience in designing and implementing digital ICs for SRAM applications. In terms of programming expertise, many members have completed the *Data Structures and Algorithms* course series, as well as the *Computer Systems and Organization* series. To further strengthen their software development knowledge, some members have also taken *Software Development*

Essentials and *Introduction to Embedded Programming*. Regarding topics related to artificial intelligence, team members are either currently enrolled in or have previously completed courses such as *Machine Learning in Image Analysis* and *Artificial Intelligence Hardware*, providing a strong foundation in AI-related concepts.

In addition to coursework, our team members have gained valuable expertise in areas closely aligned with the goals of our Capstone project. Thomas has experience in audio engineering and has successfully designed and built multiple speaker enclosures from scratch using 3D modeling tools. Bella has worked with PLC control modules for stationary power generation and has developed wiring diagrams that integrate various hardware components. Joey has created machine learning algorithms to predict diseases for the University of Virginia's medical school and to forecast the Futures market and experience designing iOS applications for consumer use. Naomi has looked into machine learning algorithms with a focus on applications related to the Tsetlin machine while also developing software to optimize energy efficiency using Asynchronous Stochastic Computing principles. Michelle has worked with machine learning models, including developing a classification-based pneumonia detection algorithm, a distracted driver detection app using computer vision, and relevant to this project, a music recommendation system.

Beyond our diverse and complementary expertise, we share a common interest for music, which motivated us to bring this Capstone project to completion.

---

3.   Project Description

The following subsections provide a comprehensive explanation of our project, detailing its performance objectives, operational mechanisms, design decisions, and testing methodologies to provide reproducibility and clarity for future implementation. Each topic will provide detailed information on (1) the physical speaker, (2) the application, and (3) integrated design, offering a cohesive understanding of all project components.

3.1.   Performance Objectives and Specifications

The following outlines the technical requirements and analytical goals of our project, designed to provide seamless integration of the iOS application and Bluetooth speaker. While the total list of design considerations remains consistent throughout, priorities differ across the software-hardware split.

3.1.1.   Physical Speaker

(1)   Objectives

The physical portion of this product was designed with an emphasis on user accessibility, synchronized visuals, and high audio quality. The objectives guiding the construction of the speaker are as follows:

- Interactivity and I/O
    - Include a power switch toggling the AC-DC power supply.
    - Provide a low-latency, Bluetooth v5.4 connection for up to 30 feet.
    - Include a 3.5mm audio jack for physical connections.
    - Implement
- Graphics
    - Configure an LCD for displaying audio based visuals.
    - Display playback information for active audio sources.
    - Design a low-latency input response.
- Audio Experience
    - Tune for a full range frequency response.
    - Minimize audio distortion (physical and electronic).

(2)   Technical Specifications

With the above objectives in mind, the device itself is powered by a supply of 15V, 10A via a 5.5/2.5mm DC jack mounted through the back panel of the enclosure. This jack is fastened to the DC plug of an AC/DC adapter, responsible for converting 110-220VAC to the necessary supply, suitable for connecting to any standard wall outlet. The speaker itself is expected to draw 30W of power, when functioning at full volume and power. Users provide an audio input after connecting to the Bluetooth module, through the functionalities of Bluetooth v5.4, with a range up to 30 feet. Users may also directly connect an audio signal from their device via a 3.5mm headphone jack connection. The speaker itself supports power, volume, and mute control on-device.

To successfully transmit and playback audio while displaying corresponding visuals, the speaker will utilize an audio amplifier board, two 8Ω full-range drivers, a Bluetooth evaluation board, a Raspberry Pi 4 Model B, and a corresponding Raspberry Pi capacitive display screen.

The related specifications of these components are as follows:

- **TPA3116D2 class-D stereo amplifier board**
  - Input voltage: 5 to 24VDC, 15V
  - Output power: 2x50W (4Ω)
  - Integrated chip: TPA3116D2
  - Efficiency: > 90%
  - Total harmonic distortion (THD): 0.01%
- **Dayton Audio PS180-8 point source full-range neo drivers**
  - Power handling: 30W (RMS), 60W (MAX)
  - Impedance: 8Ω
  - Frequency response: 48 to 25,000Hz
  - Sensitivity: 94.8dB 2.83V/1m
- **PCM1794 Bluetooth 5.4 evaluation board**
  - Input voltage: 7 to 20VDC, 15V
  - Current requirement: Above 5W
  - Integrated chip: Dual parallel, TI PCM1794
  - Communication: Qualcomm QCC5181, Bluetooth v5.4
- **Raspberry Pi 4 Model B**
  - Power supply: 5V, 3A

- ○ Storage: 8GB RAM
- ○ Processor: Quad core 64-bit ARM-Cortex A72
- ○ Frequency: 1.5GHz
- **Raspberry Pi LCD Display Screen**
  - ○ Size: 0.27"D x 6.4"W x 4.8"H (7")
  - ○ Display type: LCD/Monitor
  - ○ Resolution: 1024x600 (SVGA Wide)

These components are housed in a single enclosure with mounting for a rear facing power switch and power port as well as front facing pushbuttons, 3.5mm jack, and volume knob. The enclosure itself is composed of plywood. The overall shape is hexagonal with one front facing panel, two angled face panels (holding the drivers), two side panels, two rear-facing port panels, and a removable back plate. The final enclosure measures H9" x W24" x D12" (for detailed dimensions, see Figure 2).
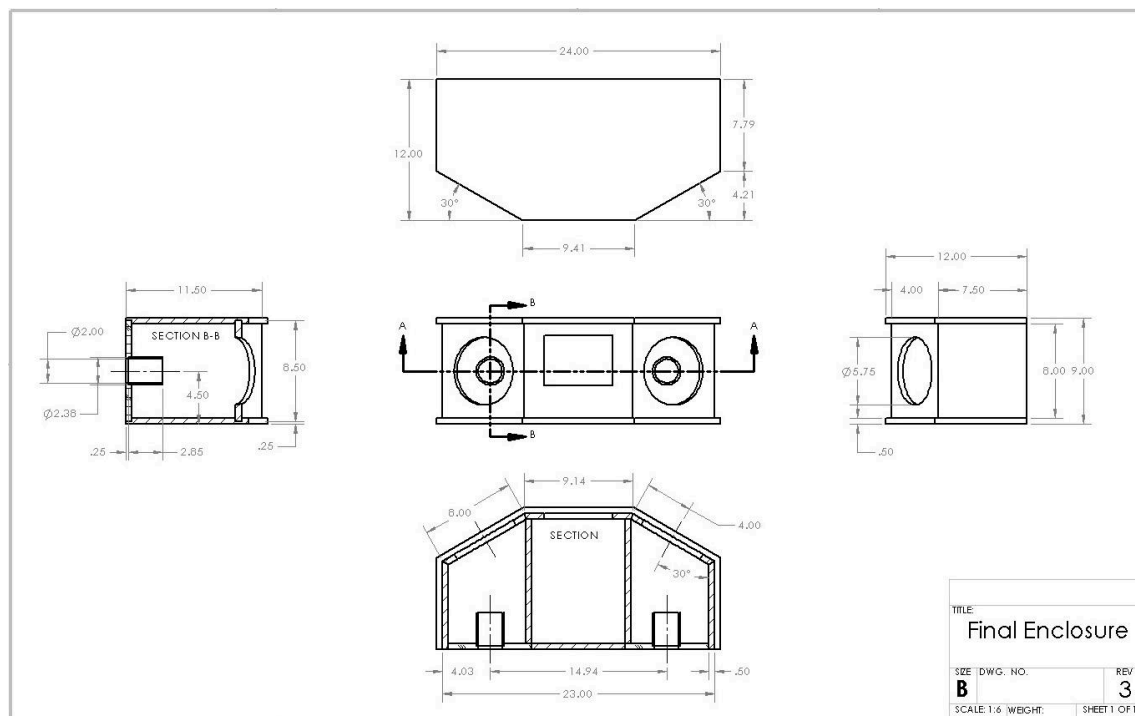
Figure 2: Detailed final enclosure schematic

(3)   *Comparison with Proposal*

The most significant modification to our proposed performance objectives relates to the processing and display of audio data. The original design called for an STM microcontroller capable of managing multiple tasks, including processing Bluetooth and data signals, handling user input communication, powering a Bluetooth module, and controlling an LCD display. In our updated approach, we have moved away from relying on a single microcontroller and instead utilized additional components to separately handle these tasks. This adjustment optimizes performance by offloading specialized processes—such as Bluetooth communication for audio signals and display control—to dedicated modules, rather than overburdening ourselves, and a single microcontroller.

We also opted to use a Bluetooth evaluation board with a built-in Qualcomm chip to handle the transmission of the audio signal to the amplifier, instead of following the original plan of converting the digital signal to analog with the selected microcontroller. Given that audio quality is a key objective, particularly within a tight timeline, we determined it was necessary to use an external board to preserve data resolution and simplify the internal hardware, while still managing the additional tasks effectively.

In subjecting an external board to handle Bluetooth communication, we selected an alternative microcontroller tool as this component would now primarily be responsible for the visuals on the LCD display. With this programming requirement reduced, we found it easier to integrate the Raspberry Pi 4 Model B in place of the originally selected STM32F769NI Discovery Kit, and order a 7" display specifically for this single-board computer. The act of configuring the screen to fit the proposed objective of "display(ing) information relevant" was still achieved, although the screen no longer showed audio data related to the Bluetooth signal or on-screen playback controls. While this objective was not fully met in terms of the content displayed, it was compensated for by the successful integration of physical controls on the enclosure and the proper functionality of the display itself.

In addition to swapping the components responsible for the controls, we also decided to update the audio amplifier board and drivers, as the increased power supply demand permitted us to select higher rated parts. Our initial design proposed using a 12V, 3A supply, but in the final

revision, we opted for a 15V, 10A supply to power the hardware. Even though this change in components necessitated greater resources, it significantly improved the speaker's ability to deliver clear audio output.

The proposal originally considered using the Sure Electronics AA-AB32155 Audio Amplifier Board to drive two 4Ω speakers, which required a lesser value of 12VDC to deliver 15W of power to each of the two channels. We revised the design to integrate 8Ω speakers, and as such, it was necessary to supply these drivers with approximately 30W (RMS) on each line; moreover, it was required that we upgrade the amplifier's power capabilities. As a result, we replaced the original board with the Texas Instruments-based TPA3116D2 model, capable of driving two 4Ω loads with a maximum of 50W of power.

These changes to the audio components significantly enhanced our speakers' performance. The revised design reproduces a much more desirable frequency range of 48Hz to 25kHz, compared to the proposed range of 160Hz to 20kHz. Additionally, the Total Harmonic Distortion (THD) specification improved with these alternative components, yielding a more favorable value of less than 1%, rather than the initial 10% distortion. With the added power, we selected higher-rated components, resulting in a more refined design that ensures clean and clear audio reproduction.

The enclosure design changed quite significantly throughout the design process. The added performance came at the cost of size, and the original plan of a portable bluetooth speaker was no longer feasible. Instead, attention was shifted towards a modest, shelf-size enclosure. Not only did the size change, but so did the geometry. Originally oriented in 180° opposition, the drivers are now oriented 60° apart in order to maximize high frequency dispersion. The size and geometry of the resonance chamber changed as well.

To summarize, while some of the hardware-related objectives were altered, we present a model that successfully integrates a display, Bluetooth functionality, and enclosure, all while enabling the output of audio. It was necessary to ensure that the system itself was built within the defined time frame, user-accessible, and permitted high-quality transmission. Further, in changing these objectives, it was necessary to alter several of the physical components, which changed a few of the related specifications of the speaker's design and relevant hardware. Ultimately, the finalized

objectives and specifications give rise to a physical product with an updated capacity for greater performance.

3.1.2. iOS Application

(1)   Objectives

The primary goals of the iOS application center around creating an innovative platform that helps elevate music discovery, increase social connections within music, and thus creating an engaging user experience. These objectives are achieved through the following key functionalities:

- **Personalized Music Discovery**: Use Spotify API key for data collection and Open AI's API key to provide specific recommendations based on user prompts.
- **Social Connectivity**: Enable users to visualize musical compatibility with friends through interactive graphical features.
- **Listening Insights**: Offer detailed analytics on listening habits, such as top songs, genres, and playlists.
- **Seamless Playback Synchronization**: Integrate with a Bluetooth speaker to synchronize music playback, display on-device visuals, and allow user-friendly controls.

(2)   Technical Specifications

The iOS application combines Spotify's vast music library with advanced AI capabilities for music discovery. Users can input prompts such as "chill acoustic vibes" within the *Personal tab* and the system generates a queue of 20 tracks, using OpenAI's GPT for sentiment analysis and Spotify API for music retrieval.

Social engagement is increased through the *Compatibility Web*, a feature that calculates compatibility scores by comparing users' saved songs on Spotify. Users can then explore their shared preferences through an interactive graphical interface which displays a user's five friends compatibility scores.

Listening insights are presented in a user-friendly format within the *Profile tab*, offering comprehensive analytics. These include a top 50 list of songs, genres, and playlists, as well as weekly listening patterns displayed in visually engaging graphs. The integration with a Bluetooth speaker ensures a seamless playback experience.

Technical considerations include Spotify API rate-limit management and caching mechanisms, ensuring uninterrupted service. The system targets playback latency under 100 milliseconds, delivering a responsive user experience. The app is also developed with SwiftUI, a developing environment backed by Apple. Our app accessibility features then include dark mode, large fonts, and auto correct, providing usability for diverse audiences.

(3)   Comparison with Proposal

The final implementation of the Tempo application closely aligns with the original project proposal, achieving its primary goals while refining certain aspects during development. The AI-powered music recommendation engine integrates OpenAI's GPT and Spotify API, generating dynamic, personalized playlists based on user prompts. Thus the *Personal tab* works as intended.

The Compatibility Web provides a graphical representation of compatibility scores by analyzing and comparing users' saved, liked songs on Spotify. Unlike the initial proposal, which envisioned incorporating detailed audio features like danceability, energy, and tempo, the final implementation focuses solely on shared saved songs, offering a simpler and more streamlined approach. While the original concept included interactive nodes that allowed users to click on profiles for deeper insights, the current version focuses on presenting compatibility scores visually, emphasizing clarity and usability within design constraints.

Some features were significantly expanded compared to the initial proposal. For instance, the app now provides users with listening insights that include their top 50 songs, playlists, and genres, as well as weekly trends surrounding the users artists and songs. This enhancement offers a more comprehensive understanding of user activity. In summary, the final product successfully delivers on the core vision outlined in the proposal while incorporating thoughtful adjustments to prioritize usability.

3.2.   Design and Functional Flow

The following includes a detailed breakdown of the project's core components and their interactions. It covers the physical speaker's design, the iOS application's functionality, and the integrated product's operation.

3.2.1.   Physical Speaker: Internal

In designing the speaker, it was essential to include components capable of transmitting audio signals via Bluetooth, as well as a single-board computer capable of efficiently operating a connected LCD display. Additionally, selecting a power supply suitable for powering all subsystems was a critical consideration. With this in mind, the speaker is designed to plug into any standard wall outlet via a 100-240V AC to DC power adapter. This adapter attaches to the exterior of the speaker's enclosure, which is internally wired to the input of the (1) printed circuit board (PCB) responsible for distributing power reliably to the attached components: the (2) TPA3116D2 Class D Stereo Amplifier Board, the (3) PCM1794 Bluetooth 5.4 Evaluation Board, and the (4) Raspberry Pi 4 Model B (see Figure 3).
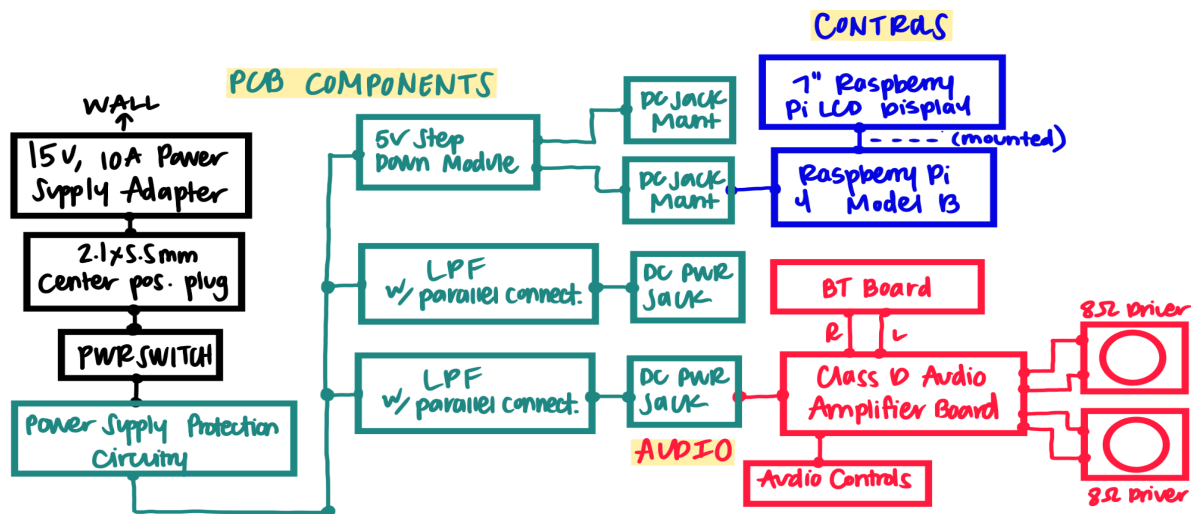
Figure 3: Functional flow diagram of the speaker hardware

(1)   Power Distribution PCB

The PCB serves as a middleman between the wall outlet, and the attached subsystems, such that appropriate voltages are seen at the input power terminals of each board. In structuring the PCB, we conducted two iterations, deciding to further integrate the second design into our project (see Figure 4).
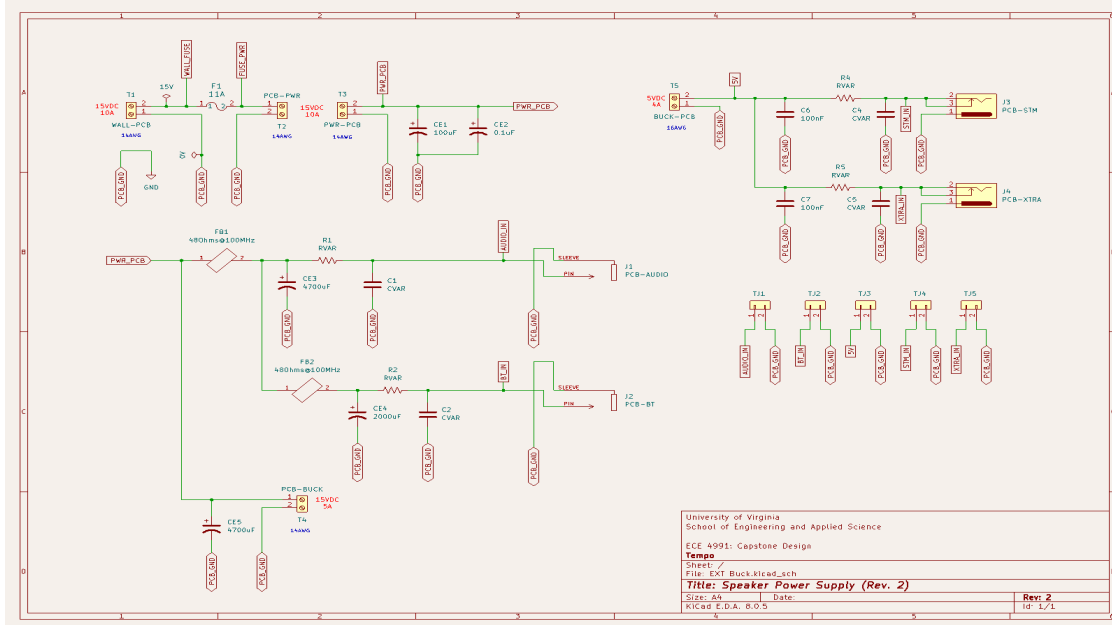
Figure 4: Schematic diagram of PCB revision 2

As previously mentioned, the power adapter converts the wall's AC voltage into a 15VDC, 10A [10] supply, which is directly connected to T1, a two-position wire-to-board terminal block that functions as the primary input for the entire system. This TE connector is rated for up to 13.5A of current and 300VDC [11], making it suitable for use in other connections as well. Consequently, the same connector is also utilized for T2, T3, T4, and T5, which handle various on-board and off-board wire connections. Following T1, the positive supply meets F1, a 12A fast-blow fuse rated up to 32VDC [12], to ensure that the aggregated components are not affected by any malfunction of the AC-DC power adapter. F1 is directly connected to T2, which is externally wired to a 20A rated, single-pole rocker switch [13], to offer the user on-device ON/OFF functionality, even if the speaker is plugged in. The secondary connection of the switch is wired again to the PCB via T3, and then passes through a pair of electrolytic capacitors connected in parallel to reduce the noise caused by the adapter's switching function. At this point in the board, it is understood that a stable 15V, 10A input exists from the wall connection.

To achieve both a stable 15V and 5V supply from this input, to power the audio amplifier, Bluetooth board, and microprocessor, the power supply is split into two parallel connections. The upper parallel connection passes through FB1, a ferrite bead rated at 48Ω at 100MHz with a 10A current capacity [14]. This component suppresses high-frequency noise and prevents

electromagnetic interference (EMI) from propagating along the power lines, which can occur due to the parallel configuration. After FB1, the circuit splits again into two separate connections, one leading to J1 and the other to J2. Before J2, another identical ferrite bead is included to further isolate the two connections and provide a stable and noise-free operation. J1 and J2, both sealed power jack connections rated for high-power applications [15], serve as stable connectors to supply the power requirements of the audio amplifier board and Bluetooth evaluation board, respectively.

The lower 15V connection from T3 is directly wired to T4, which is externally attached to a DC-DC buck converter module with 90% high-conversion efficiency [16]. Equipped with large on-board inductors, this module effectively minimizes heat generation while reducing the input supply to 5V at 4A (see Figure 5). The positive and negative output terminals are routed back to the PCB via T5, where they split into parallel connections that supply 2x5.5mm male DC jack connectors, J3 and J4, each rated for a maximum of 24V and 5A [17]. J3 is externally connected to power the microprocessor, the Raspberry Pi 4 Model B. This component then controls the display monitor via an HDMI connection.
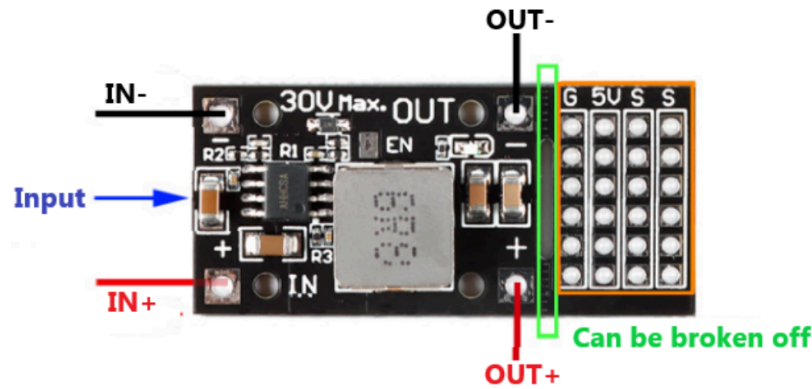
Figure 5: External 5V DC-DC buck converter module topology [16]

With the schematic finalized, the PCB design was updated and further structured in KiCad to prioritize efficient connections to the externally employed components (see Figures 6 & 7). Trace sizes were chosen based on the trace width calculator tool offered by AdvancedPCB [18], which considered the current and layer thickness, using formulas from IPC-2221 [19].
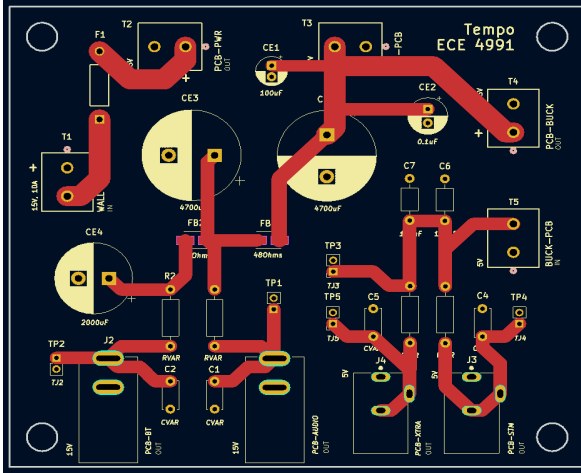
Figure 6: KiCad board layout of PCB revision 2
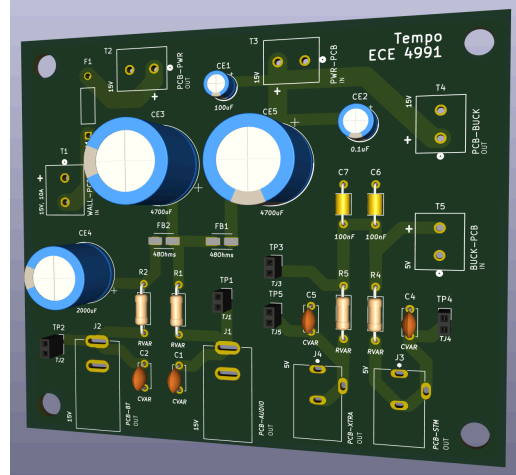
Figure 7: 3D model of PCB revision 2

Once verifying the feasibility of the PCB design, the board was ordered from OSHPark, using their 2-layer Super Swift Service to actualize three iterations of the second revision of our PCB design (see Figure 8). The board itself is 94.1 by 82.4mm, with a nominal thickness of 1.6mm [20], composed of a 175TgFR4 substrate [21]. Received and soldered, the PCB successfully offers both stable 15V and 5V supplies from the jack connections J1, J2, J3, and J4.
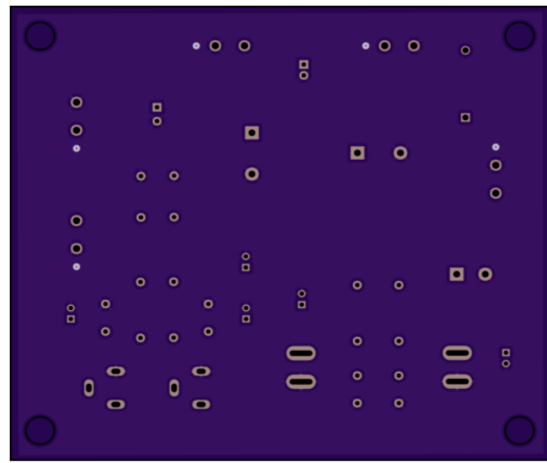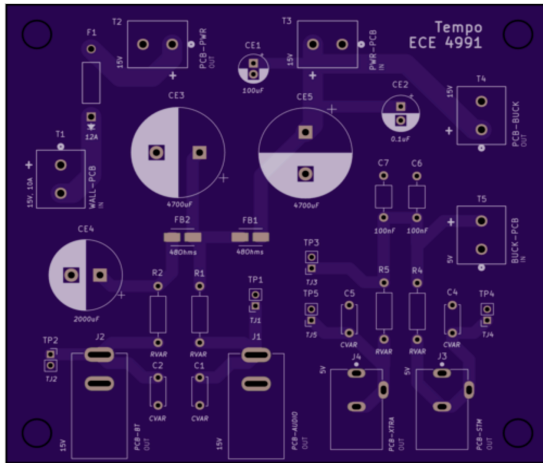
Figure 8: OSHPark board design of PCB revision 2

The PCB secures to the interior of the back plate of the enclosure, with the design of the casing discussed in a later section (see Section 3.2.2). To fasten the PCB and DC-DC converter module to the speaker such that the connections T1, T2, T3, T4, and T5 were accessible, a base plate was designed. The base plate design is composed of PLA plastic, with the following dimensions:

171.2 x 96.6 x 9.8 mm (see Figure 9). It yields a volume of about 43,073mm^3 and a surface area of 26,647.62mm^2. Cutouts of the material are implemented to ensure proper heat dissipation of both boards, alongside means of mounting the design itself.
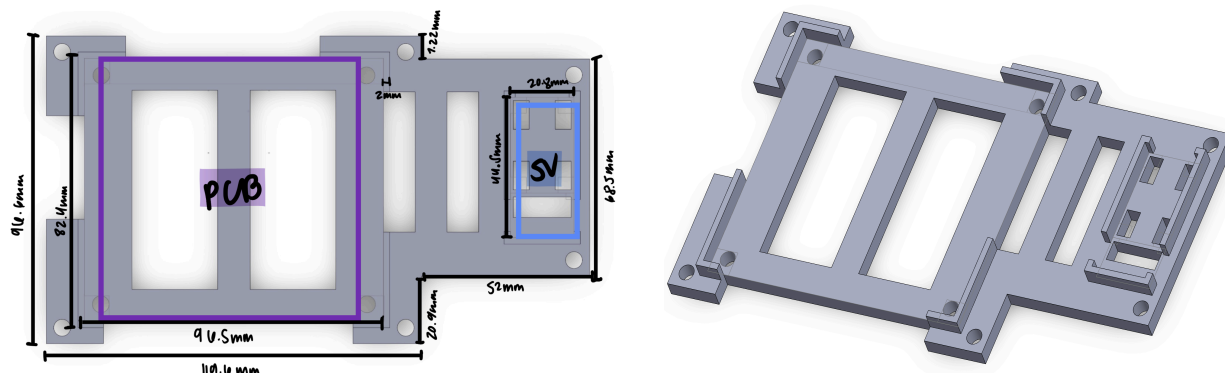
Figure 9: Power base plate top and side views

To summarize, the board is powered by an external AC-DC power adapter, which connects directly to the PCBs input. Even when plugged in, the user can direct the rocker switch to turn the speaker on or off. In conjunction with an external DC-DC buck converter module, the PCB effectively provides separate 15V and 5V connections to power the components responsible for Bluetooth audio transmission and display functionality. The components involved in power generation are fastened to a base plate, mounted internally on the back plate of the enclosure

(2)   TPA3116D2 Class D Stereo Amplifier Board

As previously mentioned, our Capstone project utilizes the TPA3116D2 Class D Stereo Amplifier Board, which operates with an input voltage range of 5-24 VDC at the power terminal connection (#5) (see Figure 10). Our circuitry supplies 15V to this screw-in terminal via the J1 power jack connector mounted on the PCB [22]. Additionally, the amplifier board incorporates the TPA3116 chip, enabling it to deliver Class D efficiency with high-quality, clean sound reproduction [23].
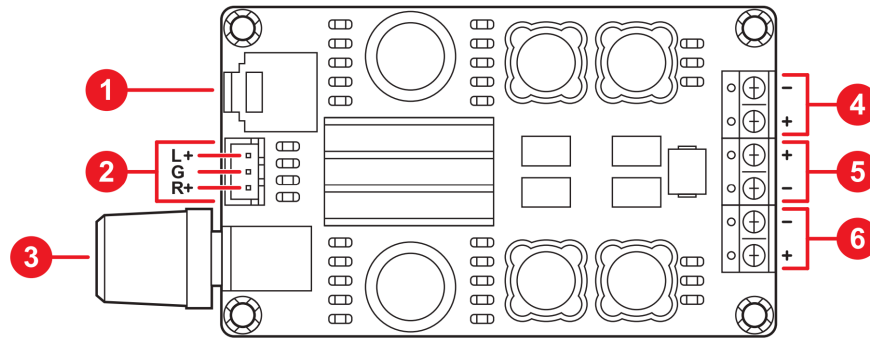
Figure 10: TPA3116D2 audio amplifier board layout [23]

Additional external safety measures were deemed unnecessary, as the board's topology includes built-in over-current, thermal, and short-circuit protection. With its two-channel stereo output, this amplifier can drive two 4Ω speakers at a maximum output power of 50W per channel. More importantly, without a heatsink on a dual-layer PCB, the amplifier is well-suited to drive our two 8Ω speakers at a maximum output power of 30W per channel [24].

The right (#4) and left (#5) speaker connections are directly fastened to the two 8Ω speaker drivers [25]. With a power handling capacity of about 30 RMS, the Dayton Audio PS180-8 Full-Range Neo Drivers are responsible for emitting the audio. These two speakers are mounted on the angled left and right front panels, through the enclosure, responsible for reproducing the full range of sound as received [26].

The speakers receive the audio signal from the amplifier, which accepts input either via a 3.5 mm stereo jack (#1) or a 3-pin socket (#2). Internally, the 3-pin socket is wired directly to the PCM1794 Bluetooth 5.4 Evaluation Board, which transmits the audio signal from the user's device (See Figure 11). Alongside this option, the 3.5 mm stereo jack input is mounted externally to provide users with the ability to directly interface from wire-in connection. The volume potentiometer of the audio amplifier (#3) is mounted outside of the enclosure, allowing the user to control the volume on the physical device.
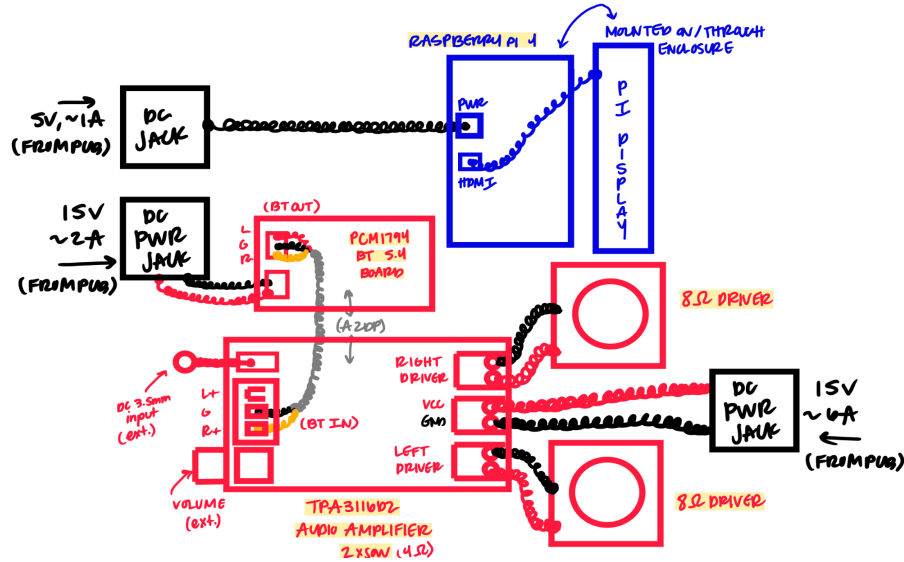
Figure 11: Speaker audio and controls connections

(3)   PCM1794 Bluetooth 5.4 Evaluation Board

As previously discussed, the performance of our speaker is directly dependent on the Bluetooth capabilities of the PCM1794 evaluation board (see Figure 12). This board supports Qualcomm chip functionalities, specifically the QCC5181 Bluetooth version 5.4 [27]. To support this component, we provide a 15V supply directly from the J2 output on the PCB (see Figure 6). Upon receiving power, the board is automatically programmed to enable its Bluetooth functionality, prompting iOS users to connect to the device titled "QCC5181." During the connection process, an onboard LED flashes blue and remains lit once a connection is established. Once connected, users are able to stream audio directly from the Tempo application.
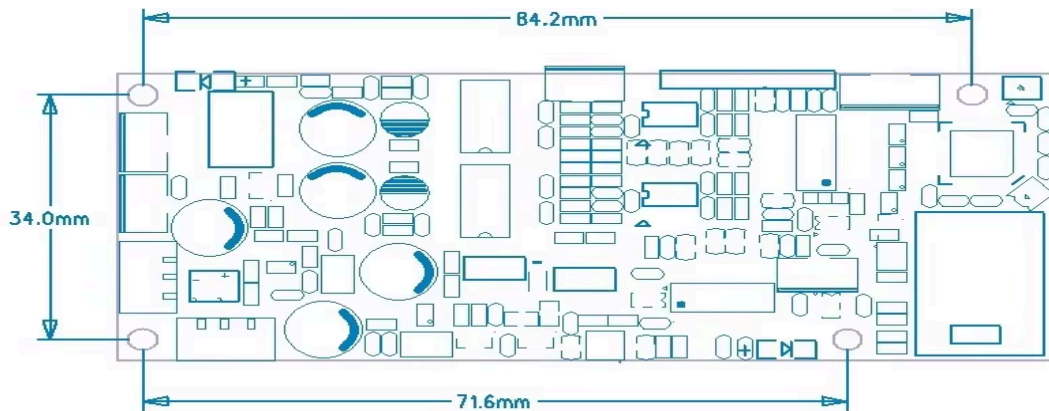
Figure 12: PCM1794 evaluation board layout

The audio amplifier and Bluetooth evaluation board secure to the bottom plate of the enclosure. To fasten both boards to the physical speaker, an additional plate was designed to provide a secure connection to the drivers and power base plate. This base plate is made of PLA plastic, with the following dimensions: 141.9 x 108.0 x 11.8 mm (see Figure 13). It yields a volume of about 49,918mm^3 and a surface area of 32,574.26mm^2. Again, cutouts of the material are implemented to provide the proper heat dissipation of both boards, alongside means of mounting the design itself.
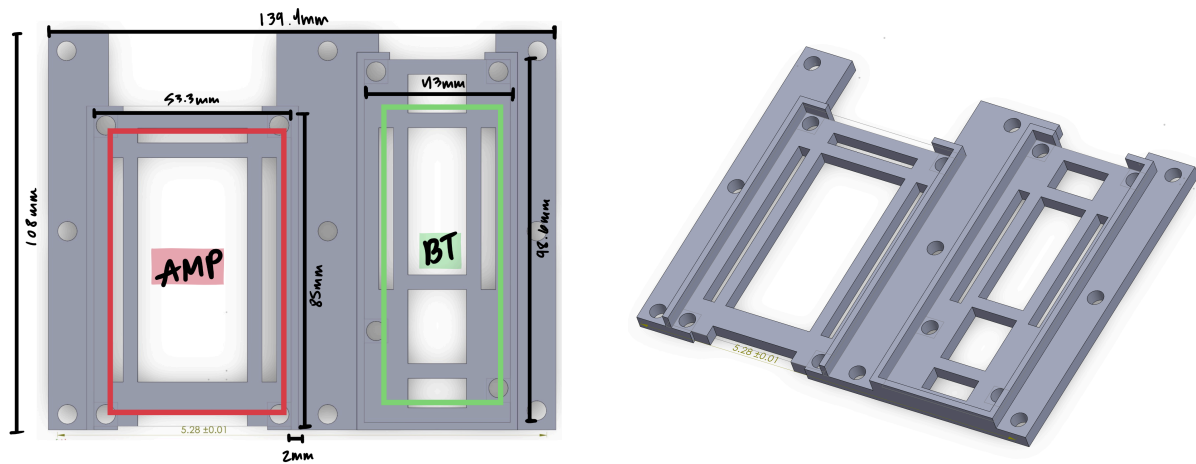
Figure 13: Audio/BT base plate design

In summary, our Capstone project drives audio by utilizing a Class-D stereo amplifier powered by a 15V supply, connected to two 8Ω Dayton Audio PS180-8 speakers mounted on the enclosure's angled front panels. The amplifier accepts input via a 3.5mm jack or a 3-pin socket, the latter linked to the Bluetooth board responsible for audio streaming. Once powered, the board automatically prompts the user to connect their iOS device. Alongside the direct line-in 3.5mm input, the user can control the volume of the audio by configuring the externally mounted volume potentiometer. Both the amplifier and Bluetooth board are securely fastened to a base plate to ensure secure connectivity.

(4)   Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B (Pi4B) was ultimately selected as the component responsible for controlling the display component, receiving a 5V power supply from J3 on the PCB, then further connected to the GPIO header. The Pi4B is powered by the Broadcom BCM2711 SoC,

featuring a quad-core Cortex-A72 (ARM v8) 64-bit processor clocked at 1.5GHz, delivering improved performance and efficiency for various computing tasks [28]. Our specific board offers 8GB of LPDDR4 SDRAM storage, alongside offering significantly enhanced CPU, GPU and I/O performance in comparison with previous models [29]. Further, the Pi4B features 2 USB 3.0 ports, 2 USB 2.0 ports, 2 × micro-HDMI ports (supporting up to 4k at 60Hz), a 2-lane MIPI DSI display port, a 2-lane MIPI CSI camera port, a 4-pole stereo audio and composite video port (see Figure 14). We configured one of the micro-HDMI ports on the device to support the speaker's display, with the operating system and data storage loaded via the micro-SD card slot.
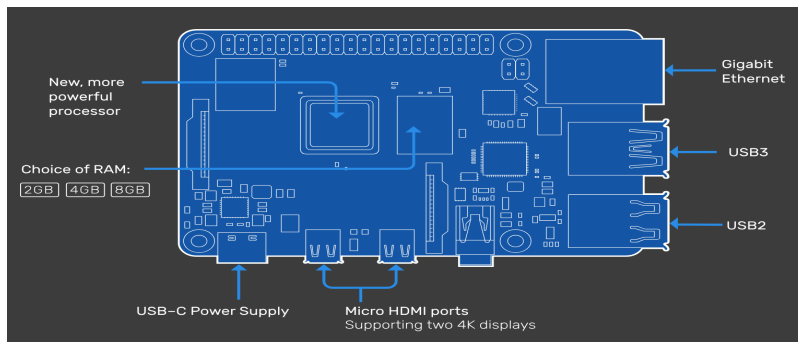
Figure 14: Raspberry Pi 4 Model B board topology [28]

The display selected was the Raspberry Pi Capacitive Touch Screen Monitor [30], which is specified for compatibility with our model, and offers driver-free 5-point touch support for Windows 10/8.1/8/7 (see Figures 15 &16). The display features a 7-inch IPS LCD panel with a maximum resolution of 1024×600. The active area measures 154.08mm by 85.92mm, while the module itself has the following dimensions: 164.9mm x 100mm x 5.7 mm. To enable video display on this monitor, the HDMI port is connected to one of the two micro-HDMI ports on the Pi4B. The Pi4B's USB port is connected to the monitor's Micro USB port to establish a sufficient power supply. This setup ensures sufficient functionality of the screen, which is mounted through the front panel of the enclosure. The display depicts a colorful looped video, corresponding to the Tempo application and audio.
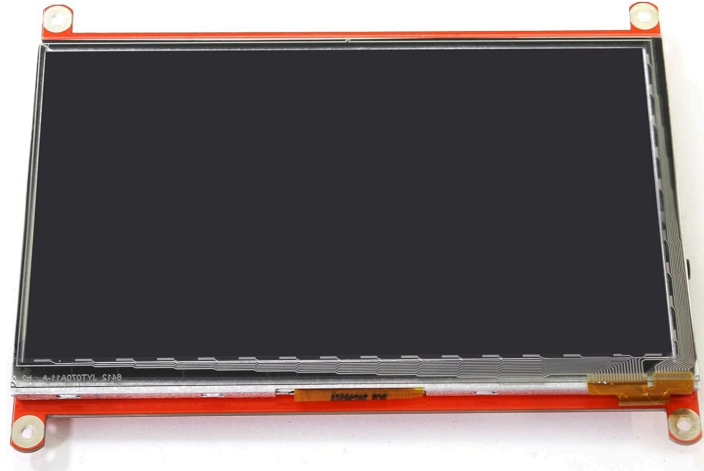
Figure 15: Raspberry Pi Screen IPS, HDMI capacitive touch screen monitor [30]
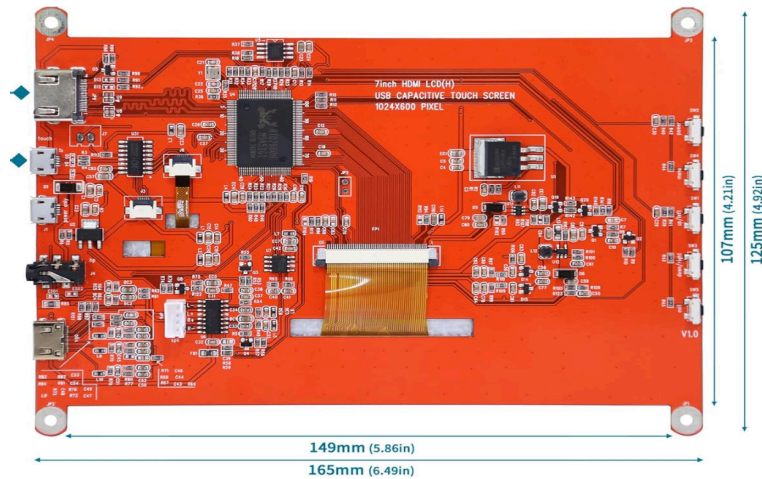
Figure 16: Raspberry Pi Screen IPS, HDMI capacitive touch screen monitor topology [30]

Initially, Raspberry Pi Imager was utilized to configure the microprocessor, writing the configured operating system to an external microSD card. The microSD card, inserted into its specified slot on the board, serves to control the communication between both devices.

The operating system's /boot/config.txt file was modified to match the 1024x600 resolution of the external monitor. To support video playback, steps were taken to ensure the successful installation of Media Player Version (MPV). Using command-line arguments within the rc.local file, MVP loops the video upon system startup. Users view a video on the monitor, featuring a saturated, iridescent background that continuously changes over time, with the Tempo

application's logo displayed on top. The video path is accessible to MPV, which enables the Pi4B's GPU to process and output the video via the HDMI port.

3.2.2   *Physical Speaker: Enclosure*

Interaction with the enclosure itself is rather straightforward. Despite using a lightweight composite, the "sandwich" construction method allows for a very sturdy box with a manageable weight. The box is intended to be placed up against a wall, which would allow the lower frequencies emanating from the ports to disperse laterally. The internal hardware can be accessed through the middle-back panel via 4 screws.

As previously mentioned, users are able to control power and volume functions on-device.

3.2.3.   *iOS Application*

The Tempo iOS application serves as the primary interface for accessing music and social connectivity features. It is organized into three main tabs—Personal, Community, and Profile—each designed to provide specific functionality.

(1)   Login Navigation

Users begin by logging into their Spotify account, which allows the app to integrate with Spotify's API to retrieve data such as liked songs, playlists, and listening history. After login, users are directed to the main interface, which provides access to the app's core features. The onboarding process is then designed to provide a secure and straightforward transition into the application while taking account of the user's preferences.
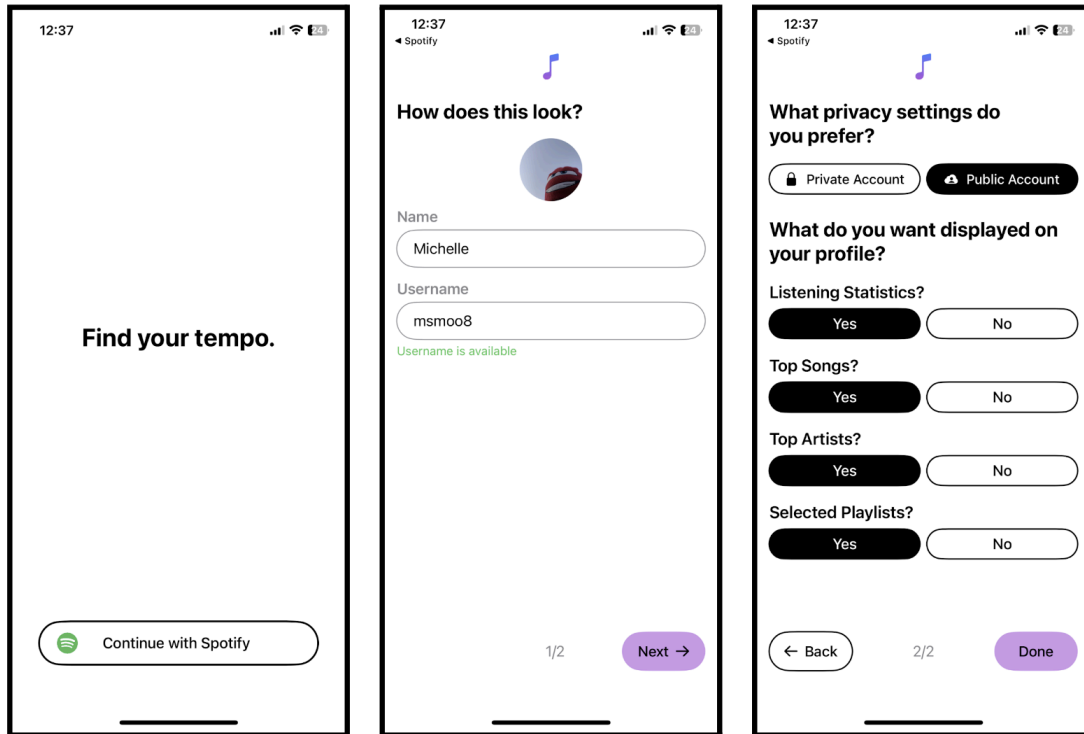
Figure 17: User view of Login Navigation

(2)   Main Tab View

The app's interface is built around a tabbed navigation bar, where users can easily switch between the three core sections: Personal, Community, and Profile. The choice of tab names reflect engineering and design decisions aimed at facilitating user experience. The "Personal" tab focuses on individuality by hosting the recommendation engine, where users can discover and listen to tailored music suggestions. This streamlined design ensures users have an intuitive and dedicated space for their personalized music experience. The "Community" tab is designed to foster connection and engagement, featuring the Compatibility Web and a list of percentage breakdowns with other users, represented visually by profile icons. This structure centralizes social features into a single, accessible space, simplifying navigation and enhancing social interaction. Finally, the "Profile" tab acts as the user's hub for managing account settings while also providing detailed listening statistics, such as genre preferences, top tracks, and artists. By adopting these descriptive and purpose-driven names, the tabs create a logical and user-friendly interface while reflecting the technical architecture and feature segmentation of the app.
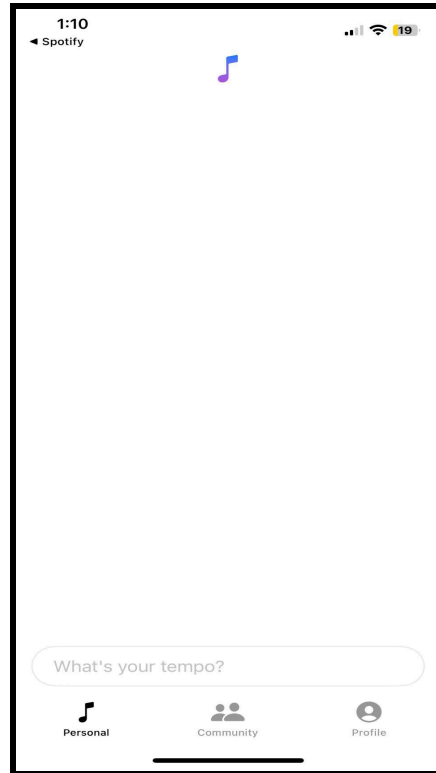
Figure 18: User view of recommendation engine in Personal Tab

(3)   Personal Tab View

The Personal tab focuses on music discovery through recommendations. Using OpenAI's GPT for natural language processing and Spotify's API for music retrieval, this feature allows users to input prompts such as "relaxing acoustic vibes" or "high-energy workout tracks." Based on these prompts, the app generates a curated queue of 20 songs tailored to the user's preferences. The song queue is presented in a straightforward format, enabling users to play, pause, or skip tracks directly within the app. When connected to the Tempo Bluetooth speaker, playback controls integrate with the device, thus allowing users to manage their listening experience consistently across platforms.
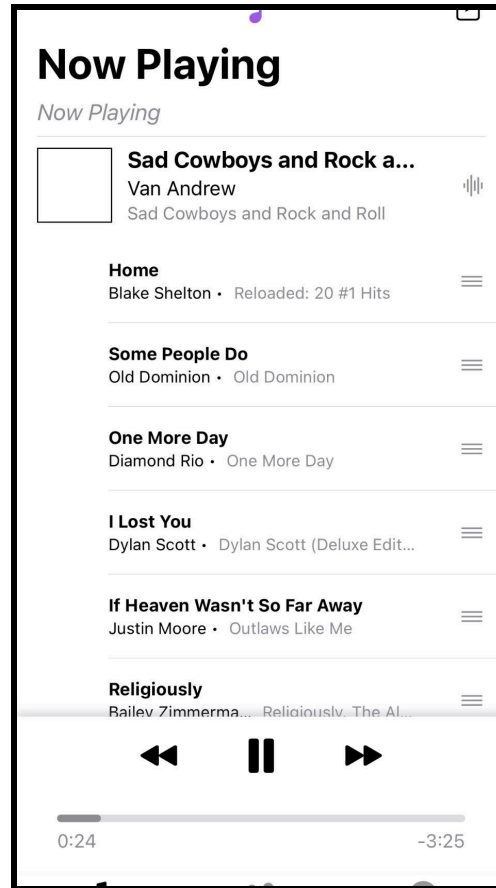
Figure 19: User view of song recommendation list

(4)   Community Tab View

The Community tab focuses on building social connections through the Compatibility Web, a visualization of the user's musical relationships. Each day, the app selects five friends from the user's Spotify follower list and displays them as nodes surrounding the user at the center of the web. The distance of each node from the center then represents the compatibility score between the user and their friend, calculated based on shared Spotify data such as liked songs. Tapping on a node then reveals detailed compatibility insights. The daily update keeps the feature engaging, while limiting the visualization to five nodes maintains clarity and prevents information overload. From an engineering perspective, displaying five friends provides a balance between providing meaningful insights and offering optimal performance on mobile devices. Overall, with five nodes, the web remains visually appealing, avoids crowding the interface, and

minimizes computational complexity, allowing for smooth rendering and responsive interactions even on devices with limited processing power.
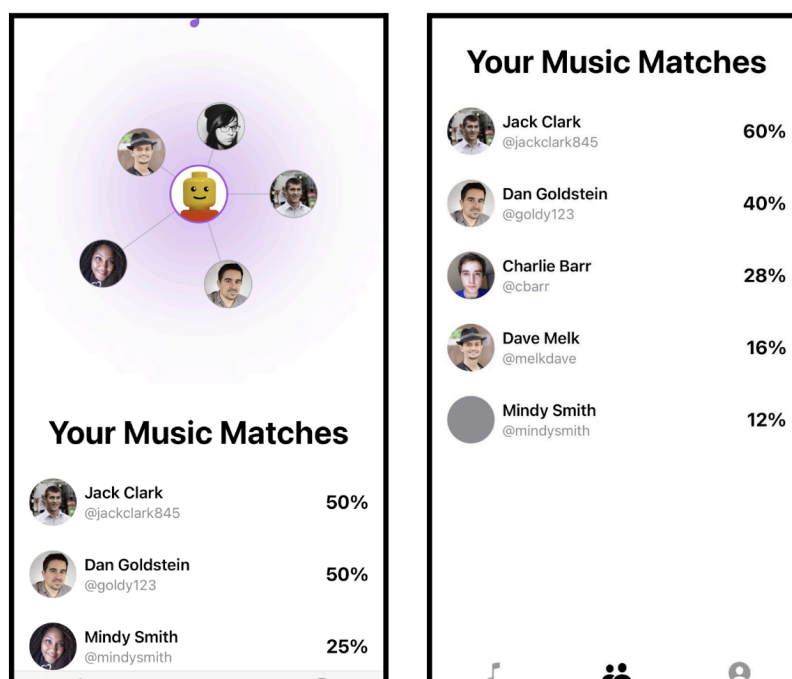
Figure 20: User view of Compatibility Tab

(5)   Profile Tab View

The Profile tab offers a comprehensive look at a user's listening habits and musical preferences, segmented into four main subsections: Summary, Songs, Playlists, and Artists. This tab serves as the central hub for exploring how users interact with their music library over time.

The Summary section provides a high-level overview of the user's activity, showcasing key statistics such as the total number of songs saved, playlists created, and artists engaged with. It also features a visual breakdown of the user's musical preferences through a detailed pie chart, which highlights their top genres. This chart makes it easy to understand the user's listening habits at a glance, offering a dynamic and engaging way to reflect on their favorite styles of music. By combining numerical data and visual analytics, the Summary section creates a balanced and informative snapshot of the user's musical journey. It emphasizes both the breadth and depth of their engagement, setting the stage for deeper exploration within the other

subsections of the Profile tab. This holistic design makes it easy for users to track their evolving tastes and better appreciate their unique relationship with music.
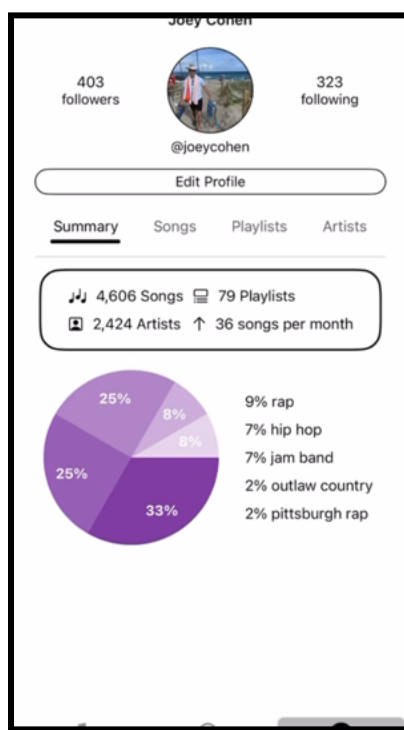
Figure 21: User view of Profile Tab view

The Songs section of the app prominently displays the user's most-played tracks, offering a personalized experience by enabling filtering of the results over various time frames: All Time, Last Six Months, or the Last Four Weeks. This modular approach allows users to explore their music preferences dynamically based on their listening history over these intervals.

Each filter presents a visually engaging and ranked list, showcasing details such as the song's title, artist, and album cover. These details help users recall their favorite tracks effortlessly and enhance their overall experience. Additionally, the design prioritizes clarity and usability by employing a tabbed navigation system, allowing quick toggling between songs, playlists, and artists. This integration ensures users have a comprehensive view of their music consumption habits, making the Songs tab a key component of the app's profile section.
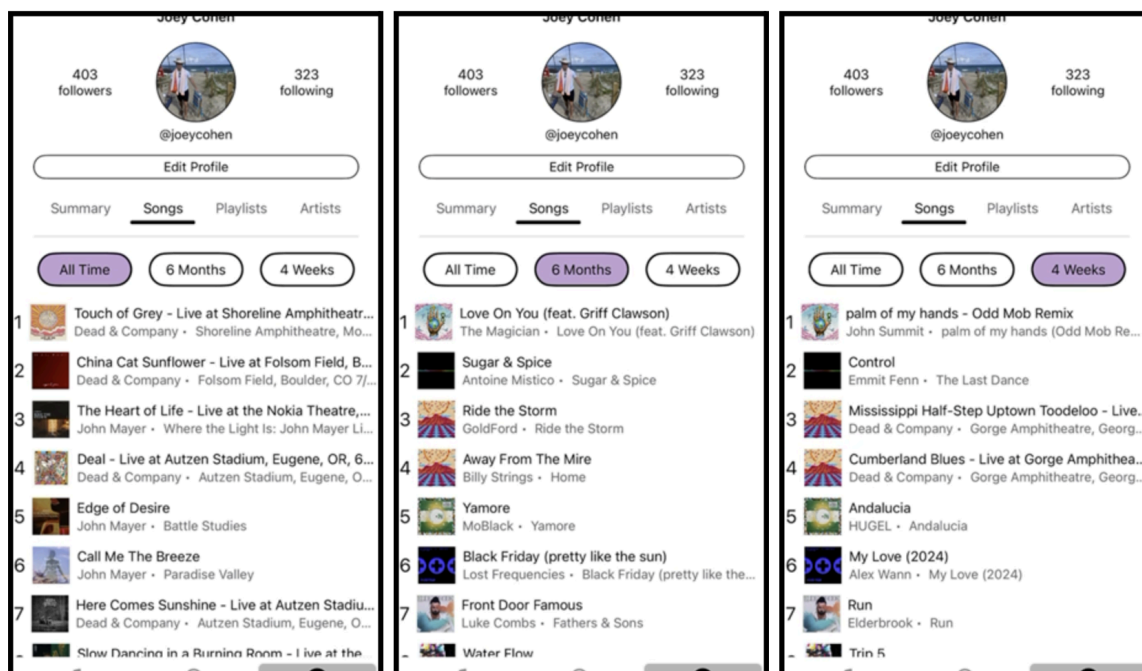
Figure 22: User view of Top Songs from All Time, 6 Months, and 4 Weeks

The Playlists section offers users a visual and organized display of all the playlists they have created or followed, serving as a snapshot of their curated music collections. This section focuses on showcasing the playlist covers, allowing users to browse their collection at a glance. Each playlist is represented by its unique cover art, giving the section a vibrant and personalized feel while reflecting the diversity of the user's musical tastes.

Although this section is primarily for viewing rather than interaction, it provides a clean and aesthetically pleasing way to appreciate the breadth of playlists the user has accumulated. Whether these playlists were created to fit specific moods, events, or themes, the display ensures that all playlists are easily visible in one place, reinforcing the app's focus on personalization and user-centric design. The section's streamlined layout emphasizes simplicity and clarity, ensuring that even users with extensive collections can quickly scroll through their playlists without feeling overwhelmed.
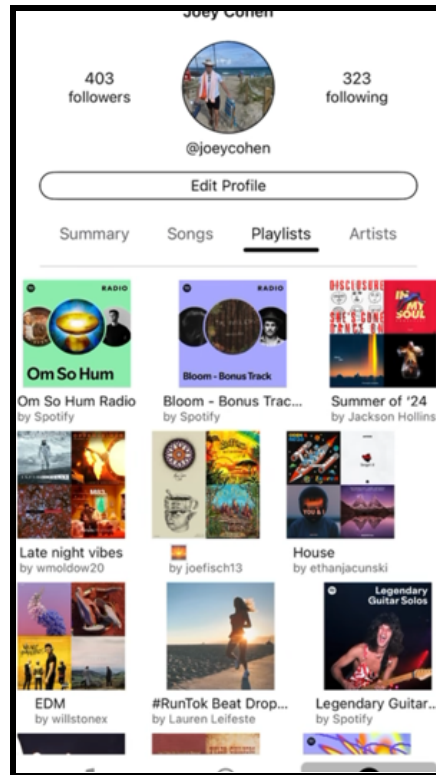
Figure 23: User view of Playlists Tab

Lastly, the Artists section highlights the user's favorite artists, organized across customizable time frames such as All Time, Last Six Months, and Last Four Weeks. This feature allows users to explore their top artists and see how their preferences shift over time, offering a clear snapshot of their most impactful musical influences.

The section focuses on presenting a ranked list of artists, creating an easy-to-read layout that visually emphasizes the user's top musical contributors. Each entry reflects the user's interaction with these artists, providing a streamlined way to revisit and reflect on their listening habits. By segmenting data into time-based categories, the section allows users to notice trends, rediscover past favorites, and celebrate newer discoveries, adding depth to their overall experience within the app.
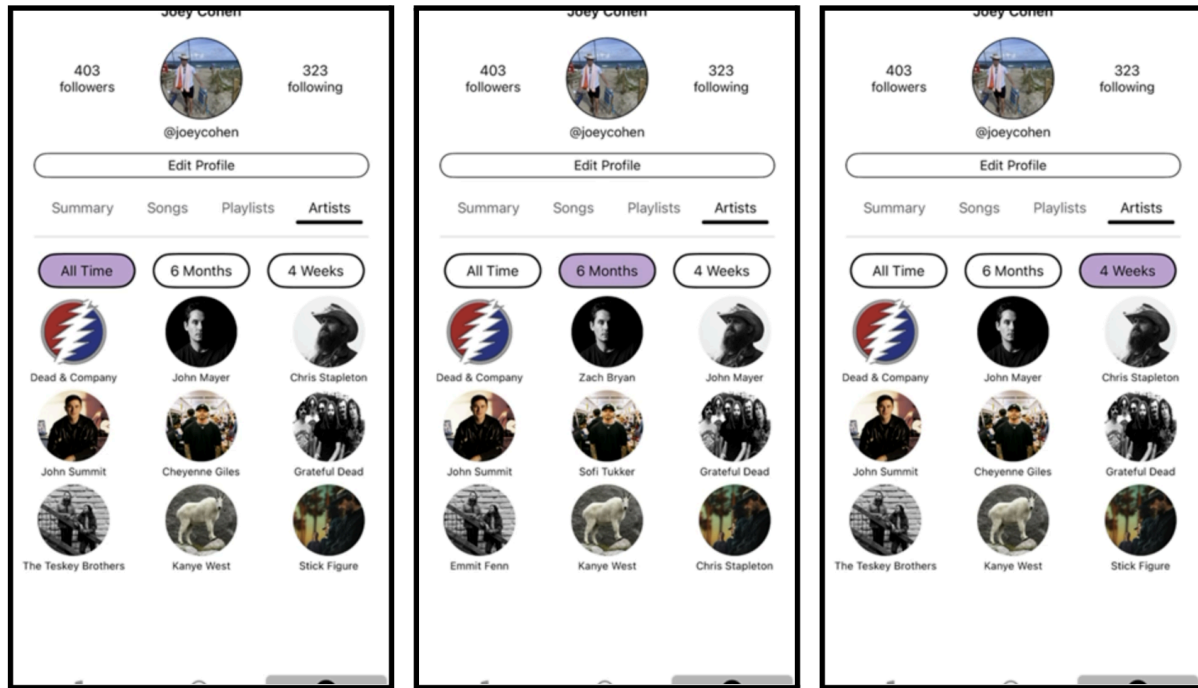
Figure 24: User view of Top Artists in All Time, 6 Months, and 4 Weeks

The iOS application then integrates music discovery, social engagement, and personalized insights.

3.2.4. Integrated Product

The integrated product will consist of the user successfully downloading the Tempo app, logging in given their spotify information, syncing the app with bluetooth and connecting to the speaker. Once connections are made, the user has access to all features such as viewing recommendations, compatibility scores with friends, and viewing their listening behaviors. The user also gains access to speaker controls such as volume adjustment and viewing basic controls on the LCD display.

3.3. Technical Design Decisions

The following discusses the rationale behind key design choices, modifications made during development, and the engineering tradeoffs considered to optimize performance and functionality.

3.3.1. Physical Speaker: Internal

(1)   Rationale Behind Design Choices

Audio Subsystem: Drivers, amplifier, and Bluetooth

Driver Selection: In choosing the drivers to integrate in our design, we focused on finding a model with specifications that adhered to predefined criteria related to its frequency range, frequency response, sensitivity, impedance, and power handling:

- Specified frequency range similar to the audio that will be transmitted, with the minimum value within this range approximately being around 50Hz.
- Minimal deviation of the frequency response when the user is stanced at different angles.
- Attributed value of sensitivity around 95dB or higher.
- Impedance value of 8Ω.
- Sufficient power handling, such that it can be driven by a reasonably-priced Class-D amplifier.

To narrow our search, we considered only passive full-range drivers due to their ability to "reproduce six or more octaves" with a single component [28]. This consideration was essential, as space constraints within the physical enclosure played a significant role in our design decisions. Full-range drivers are common in smart speaker designs because they effectively combine two speaker functions into one, with a tweeter integrated into the center of a woofer.

Additionally, we planned to implement two drivers with an impedance of 8Ω. In smaller speaker designs, driver impedance is typically either 4Ω or 8Ω. While 4Ω drivers can deliver higher power, they come with drawbacks, such as increased current draw and a higher risk of thermal overload. Considering the available current supply and the need to integrate several subsystems, 8Ω drivers were deemed a safer and more practical choice within our design constraints.

Guided by these criteria, we selected two Dayton Audio PS180-8 Point Source Full-Range Neo Drivers, opting for the 6-1/2" diameter model. This choice was informed by the principle that "the larger the surface area of the loudspeaker cone, the better it is at producing sound at long wavelengths" [29]. These drivers provided an excellent balance between size, performance, and cost, meeting our requirements for power and audio quality.

The selected model features a frequency range of 48Hz to 25,000Hz, meeting the minimum value of the range we aimed to achieve. Furthermore, its frequency response exhibited minimal deviation at different angles (see Figure 24), ensuring consistent audio performance across a wide listening area.
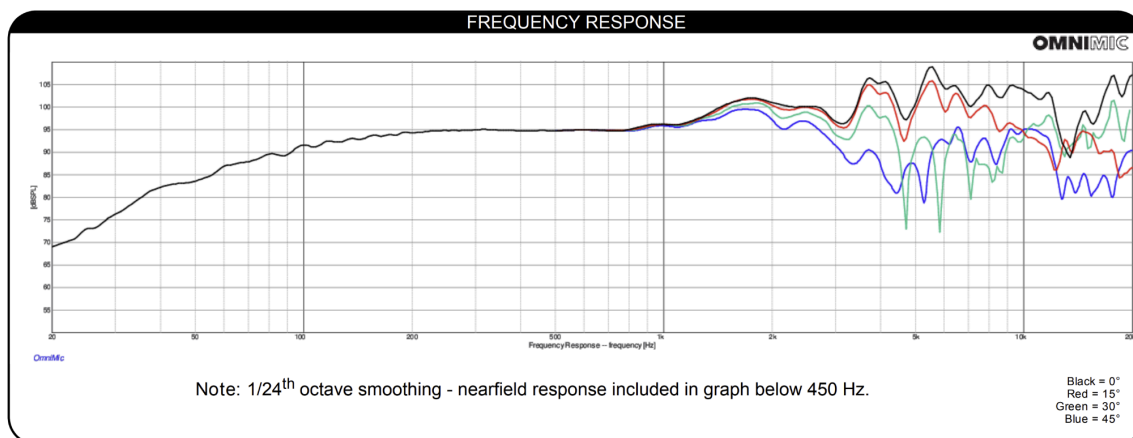
Figure 25: Frequency response of the Dayton Audio PS180-8 Full-Range Neo Driver

As seen above, the response itself is quite optimal, considering that little deviation happens overall, with alterations beginning to occur around the 3.5kHz mark. In comparison to other models, this response graph yielded the best results.

These drivers have a sensitivity of 94.8dB 2.83V/1m, meaning they produce a relatively high sound pressure level (SPL) with minimal input power, making them highly efficient for this design 30. This is beneficial as it allows the speaker system to deliver loud and clear audio without requiring excessive power, which reduces strain on the amplifier and power supply, enhances overall energy efficiency, and ensures better performance at lower power levels. Finally, these drivers operate efficiently with 30 Watts RMS power handling and have a maximum power handling of 60 Watts, providing a reliable balance between robust performance and durability.

Amplifier Selection: After determining the drivers to be utilized, it was necessary to select a complementary audio amplifier board capable of delivering power to each of the two channels connected to the speakers. The amplifier needed to efficiently handle the 8Ω impedance of the selected drivers while delivering sufficient power for optimal performance without distortion.

It was decided beforehand, to select a preconfigured amplifier circuit, one embedded on an external board with tested specifications. This would limit the time and possible mistakes that would come about in designing our own amplifier circuitry, and would allow us to take advantage of the user-control functionalities offered by the available board models. Further, we conducted our search for amplifier board models that utilized a Class-D amplifier chip, known to be generally favored for their high efficiency–greater than 90%–and low heat generation [31].

We decided upon integrating the TPA3116D2 Class D Stereo Amplifier Board, for the following reasons:

- Accepts a wide input voltage from DC 5V to 24V, allowing for flexible voltage supply options.
- Included built-in reverse power supply protection and speaker output short-circuit protection, to protect both the internal and external components.
- Offers user-control functionalities supporting a direct 3-pin jumper input, or 3.5mm headphone jack line-in input.
- Suggested to be implemented to drive two 8Ω drivers, matching our selected full-range speakers.
- Operates with an efficiency greater than 90% in driving two 4Ω loads, so greater performance is expected when driving two 8Ω speakers.
- Total harmonic distortion value of 0.01% when driving two 4Ω loads, so an even lower value expected when driving two 8Ω speakers.
- On-board volume control, via an externally mounted volume potentiometer.

Bluetooth and Implementation: To begin, it was necessary to consider how to design the speaker such that the audio could be transmitted wirelessly: either via Bluetooth or Wifi. We chose Bluetooth over Wi-Fi for its lower power consumption and simpler connectivity, which align better with the compact and energy-efficient design of the speaker. Additionally, it offers sufficient bandwidth for high-quality audio streaming while ensuring ease of use for end-users without requiring complex network configurations.

Further, we selected a preconfigured Bluetooth audio board to simplify integration and ensure reliable performance without requiring extensive custom firmware development. This choice allowed us to focus on optimizing the overall system design while leveraging the board's tested and proven capabilities for audio applications.

Specifically, we integrated the Dual Parallel PCM1794 Evaluation Board, which supports QCC5181 Bluetooth v5.4. The board itself was chosen because of the following characteristics:

- Supports either DC or AC input, with a maximum of 24V; moreover, it operates with a wide voltage range input allowing for a flexible power supply.
- Automatically configured to prompt users to connect via Bluetooth when powered on, limiting the necessary configuration steps prior to its implementation.
- Requires only 5W of current, allowing us to dedicate the bulk of our supply to support other circuit functionalities.

Power Distribution: Supply, PCB, and integration

Power Supply: In determining the power supply for all of the components, it was necessary to consider the prospective power draw of the audio components. We decided to base our system on the TPA3116D2 Class D Stereo Amplifier Board, which is capable of driving two 4-Ohm speakers at 50W$_{RMS}$ per channel [32]. In our application, we found it necessary to ensure that our selected power supply would permit the amplifier board to achieve the noted 25-W$_{RMS}$ per channel, to sufficiently drive two 8-Ohm full-range speakers. To achieve this minimum 25-W$_{RMS}$ per channel, the minimum supply voltage was found to be:

$$V_{RMS} = \sqrt{Output\ Power_{RMS} \times Speaker\ Impedance}$$

$$V_{RMS} = \sqrt{25W_{RMS} \times 8\Omega} = \sqrt{200} \simeq 14.14V$$

For peak output voltage, this translates to:

$$V_{PEAK} = V_{RMS} \times \sqrt{2} = 14.14V \times \sqrt{2} \simeq 20V$$

The above calculations made it clear that at least a 15V supply voltage would be necessary to achieve this 25-W$_{RMS}$ output power per channel [33].

To estimate the nominal and maximum current draw from employing this amplifier, it was necessary to consider the efficiency associated with the Class-D topology of the unit. Assuming >90% power efficiency, the nominal current draw was found to be:

$$P_{OUT} = 2 \times Output\ Power_{RMS} = 2 \times 25 = 50W$$

$$P_{IN} = \frac{P_{OUT}}{\eta} = \frac{50}{0.9} \simeq 55.56W$$

$$I = \frac{P_{IN}}{V} = \frac{55.56}{15} = 3.7A$$

For maximum current draw, this translates to:

$$P_{OUT} = 2 \times Output\ Power_{RMS} = 2 \times 50 = 100W$$

$$P_{IN} = \frac{P_{OUT}}{\eta} = \frac{100}{0.9} \simeq 111W$$

$$I = \frac{P_{IN}}{V} = \frac{111}{15} = 7.4A$$

The above calculations made it clear that at least a 7.4A current supply was necessary to ensure the amplifier would operate with significant headroom available.

It was crucial to additionally consider the current draw from other components:

- Bluetooth: 5W requirement minimum for functionality.
- Raspberry Pi: Necessary 5V supply at 3A to support the display.

With this in mind, we deemed the 15V, 10A converter from a wall connection to be sufficient to serve as the input source.

PCB Design: The PCB design focused on delivering stable and noise-free power to various components, with provisions for both 15V and 5V supplies. To minimize interference, ferrite beads were used to suppress high-frequency noise and EMI. Further, TE terminal blocks and sealed power jack connectors were chosen for their current and voltage ratings, ensuring reliable connections. Additionally, a 12A fast-blow fuse was included to protect against potential malfunctions from the AC-DC adapter, safeguarding all connected components. On the board, trace widths were calculated to handle the current load, with KiCad's PCB layout tools and industry standards (IPC-2221) ensuring reliability and minimal resistance losses. These decisions prioritized efficiency, reliability, and safety in power distribution, accounting for high currents and potential noise sources.

External Buck Converter: We chose to integrate a pre-designed external buck converter module rather than designing one directly on the PCB to optimize time and ensure reliability.

The external module provided a high-efficiency conversion from 15V to 5V with minimal heat generation, meeting the power requirements of the Raspberry Pi and other low-voltage components. Its robust design, complete with integrated inductors and thermal management, reduced the complexity of our PCB layout and eliminated the need for extensive testing and troubleshooting typically associated with custom power circuitry. Additionally, using a pre-designed module ensured that we could focus on other critical aspects of the project, while still achieving a stable and reliable power supply for our system.

Integration Techniques: Components like the amplifier, Bluetooth board, and microprocessor were powered through distinct PCB outputs to isolate noise and optimize stability. Organized wire routing ensured that signals (audio and power) did not interfere with one another, preserving signal integrity. Modularity and separation of components simplified integration while ensuring reliable operations without electrical or thermal interference.

Controls: Microprocessor, display, and communication

Controller Selection: The Raspberry Pi 4 Model B was dedicated solely to operating the display, simplifying programming and alleviating computational demands on other components. It was chosen for its powerful quad-core processor, ample RAM, extensive I/O ports, and seamless compatibility with the display. The Pi's ease of use, versatility, and robust performance made it the ideal choice, especially under tight time constraints, outweighing the challenges associated with alternative solutions.

Connection: An HDMI connection linked the microprocessor to the capacitive touch display, while USB handled the power supply. These straightforward connections were preferred for reliability and simplicity.

Display Selection: The Raspberry Pi Capacitive Touch Screen Monitor, a 7-inch IPS panel with a 1024×600 resolution, ensured a sharp and vibrant display. The display is intended to be secured to the speaker's front panel, showcasing a looped video with the Tempo application's branding. The monitor provided a seamless integration with the Raspberry Pi, balancing functionality and aesthetics.

Mechanical: Plates, wires, and organization of components

Base Plates: Custom-designed PLA base plates were used to securely mount the PCB, amplifier, and Bluetooth board, featuring strategically placed cutouts to enhance heat dissipation. By employing separate base plates for the power, audio, and Bluetooth subsystems, the design simplified assembly while improving heat management. PLA plastic was selected for its lightweight durability, making it ideal for custom designs and effective thermal management. Each plate was precisely sized to fit its corresponding components, with cutouts promoting airflow and minimizing the risk of overheating. This organized and secure mechanical assembly helped mitigate potential issues such as overheating or electrical noise, resulting in a polished and reliable final product.

General Organization: Wires were carefully organized to minimize tangling and interference, with power and signal wires intentionally separated to ensure cleaner operation. Positioning the power plate on the back panel of the enclosure and the audio and Bluetooth plate on the bottom provided physical separation, enhancing organization and ensuring reliable performance.

(2)   Modifications and Tradeoffs

Audio Components

With respect to the audio system, two major modifications were made to the proposed design: the component selection of both the drivers and amplifier (see Figure #).
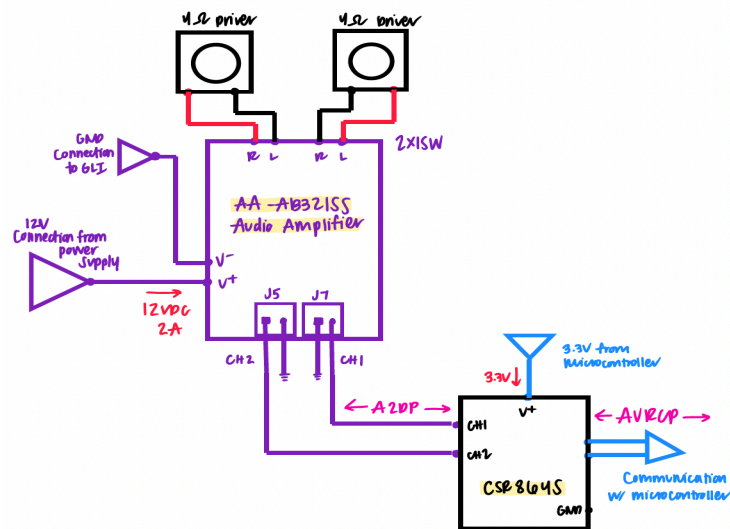
Figure 26: Proposed audio subsystem design with initial component selection

Initially, we selected speakers with a 4Ω impedance, based solely on the assumption that this was a standard value for full-range drivers in smaller audio system designs. Lacking a thorough understanding of the technical documentation associated with audio components, we sought guidance from Professor Powell. During our discussion, he explained how to interpret the graphs included in the technical documentation for our initial choice, the Dayton Audio DMA58-4 driver. Although the datasheet indicated a wide frequency range of 160 to 20kHz [34], we came to understand that its frequency response was not ideal for our design, particularly since we did not plan to include a woofer to cover lower frequencies (see Figure X). Additionally, the 4Ω driver exhibited a relatively high sensitivity of 86.2dB 2.83V/1m, when paired with our initially selected power supply.
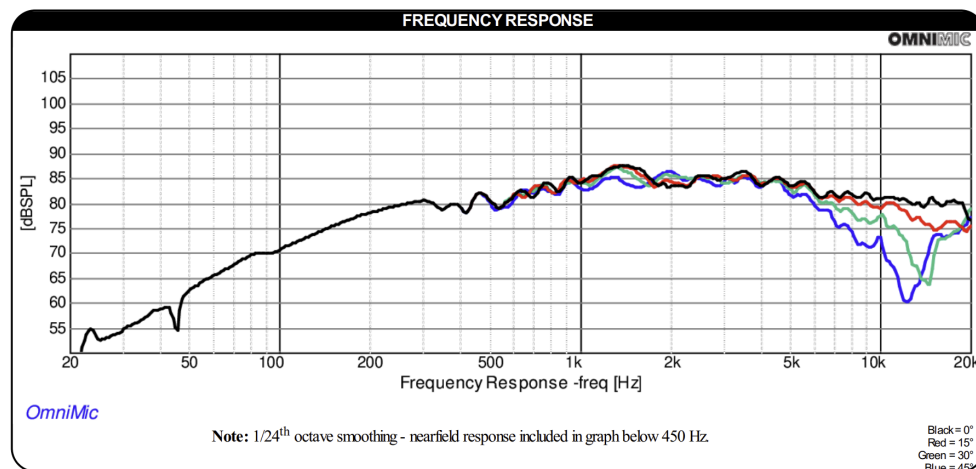
Figure 27: Frequency response of the Dayton Audio DMA58-4 driver [X]

Professor Powell recommended optimizing our design by selecting an 8Ω driver, as it would offer better performance in terms of frequency response and distortion for full-range models. Taking his advice, we revised our design to incorporate an 8Ω driver capable of transmitting the desired frequency range while maintaining a relatively high sensitivity when operating with our specified power supply.

The second major modification made with respect to the audio subsystem, was the choice of the audio amplifier. Initially, we proposed implementing a different Class-D amplifier board, the Sure Electronics AA-AB32155, capable of delivering 2x15W output at 4Ω [35]. In modifying the 4Ω drivers to the 8Ω model, these components then required 30W RMS to function normally, a power supply two times higher than the original requirement. On the basis of functionality

alone, we understood it to be necessary to modify the selected audio amplifier, but it became more apparent after meeting with Professor Powell.

In selecting the amplifier model, we based our decision solely on the output power capabilities of the board, and disregarded the specifications on the datasheet; moreover, we just assumed the operating parameters would be apt for our design. We believed the initial board's efficiency of 88% was surely good enough, and further its THD+N of 10% in driving two 4Ω loads would not affect its function when operating in the context of our design [36]. Upon learning that distortion becomes noticeable to the human ear by around 1%, it was crucial to modify our design solely on the basis of its distortion alone [37].

Ultimately, as previously stated, we chose to implement the TPA3116D2 audio amplifier board, after better understanding the parameters of efficiency and THD. This board now provided sufficient power to operate the two new drivers selected–outputting around 30W RMS per channel–while maintaining a relatively low THD [38].

Power Supply and PCB

In selecting a higher power amplifier, it was necessary to modify the power supply that served as the input, as the original design only supplied 36W to the circuit as a whole (see Figure 27).
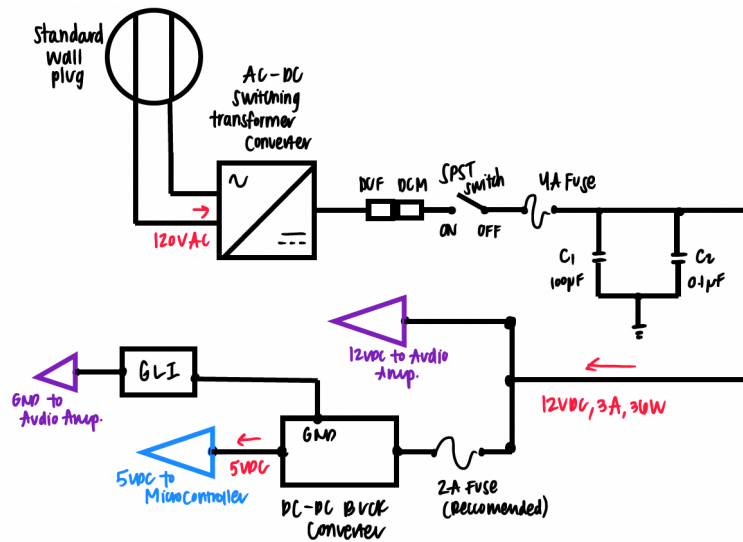
Figure 28: Proposed power distribution design with initial component selection

The original design considered an AC/DC converter [39] that outputted a stable 12V, 3A supply; and thus, we modified this converter to fit the needs of the new amplifier, choosing a converter that yielded a DC supply of 15V, 10A. Following modifying this value we ordered the first revision of the PCB, that we later deemed crucial to redesign (see Figure 28).
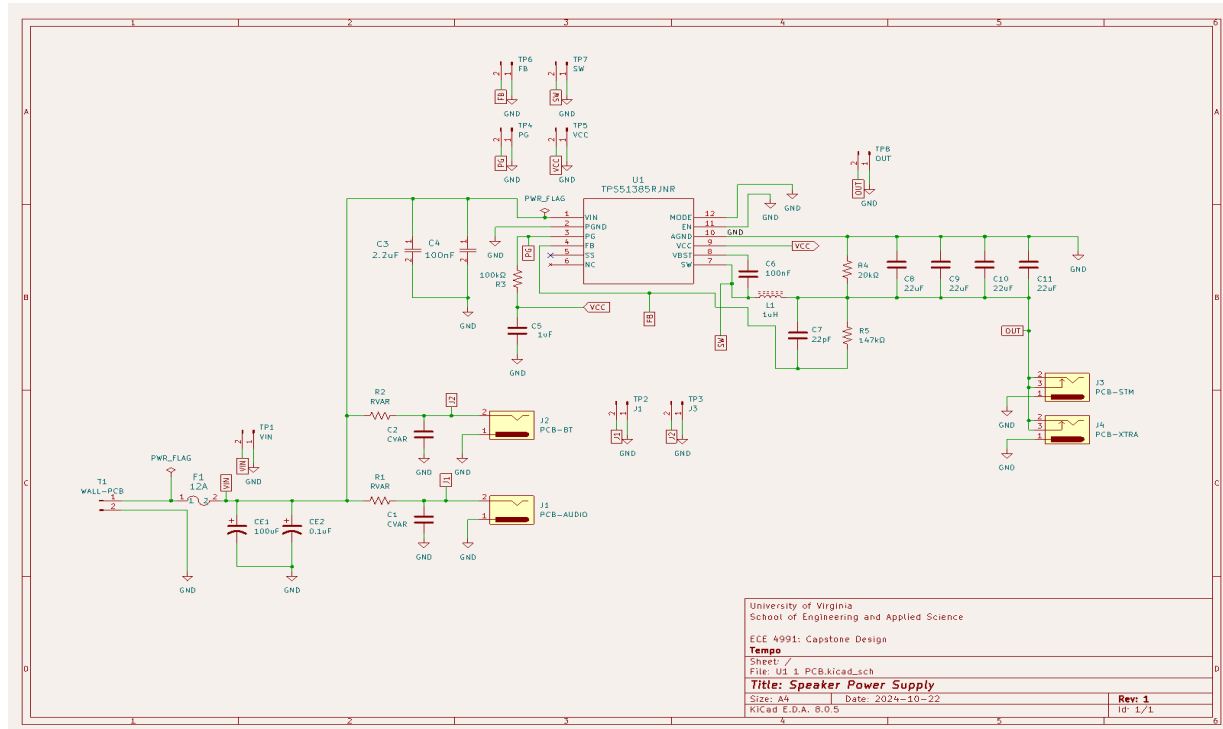
Figure 29: PCB revision 1 – Schematic diagram of the power supply circuit

The first revision of the circuit board, which defined the power distribution layout, initially integrated self-selected components for the buck converter directly on the board rather than using an external module mounted on the plate. Additionally, it did not account for an external connection to the power switch. Due to the challenges of soldering small surface-mount components and concerns about potential errors in the topology, we decided to employ an external module and redesign the PCB. This decision was crucial to ensuring a reliable 5V supply for powering the control components while improving accessibility to the stable, external module. By implementing the external module, we reduced the variables associated with potential design errors, enhancing overall system stability. The initial PCB layout for the first revision, along with its 3D model fabricated in KiCad can be seen below:
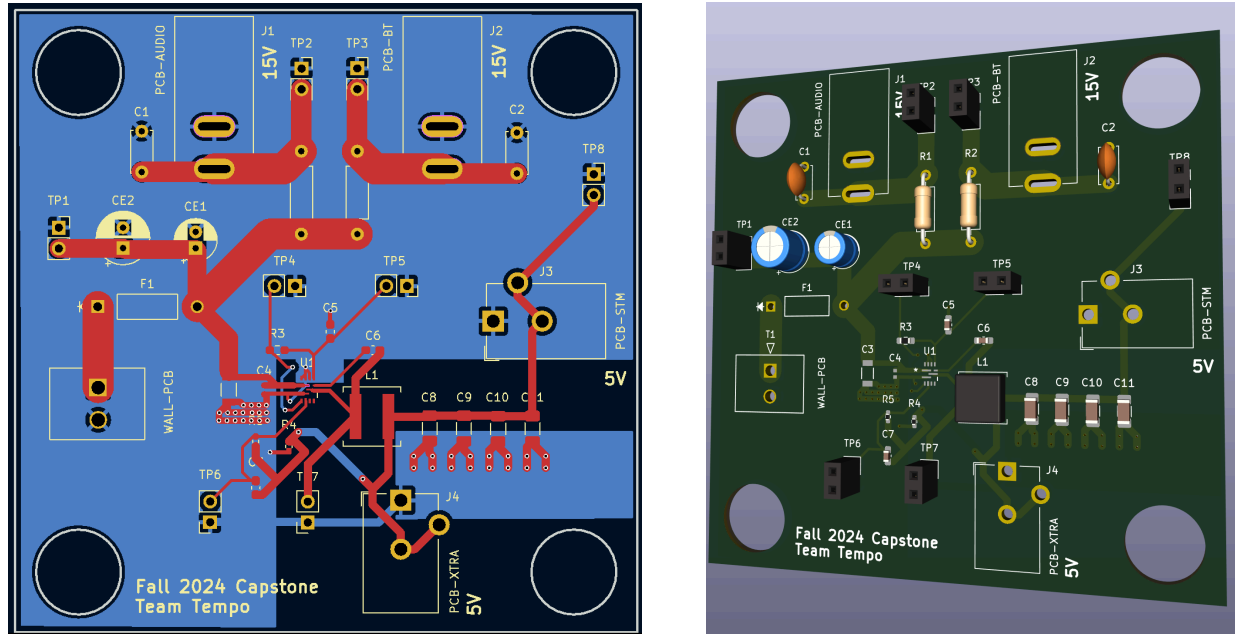
Figure 30: PCB revision 1 – KiCad board layout and 3D model

Initially, we planned to use the first revision of the PCB for power distribution and placed an order with OSHPark, receiving three iterations of the design (see Appendix 1). Although this specific physical board was not ultimately implemented, we utilized its 15V functionality to test the audio and Bluetooth components while awaiting the fabrication of the revised PCB topology.

Modifying and reordering the PCB design was a time-intensive process, but it resulted in a board that met our design requirements and provided valuable learning opportunities along the way.

Controls

The proposed subsystem responsible for the controls changed greatly, as we changed the objectives of many of the components; and thus, the components themselves (see Figure 30).
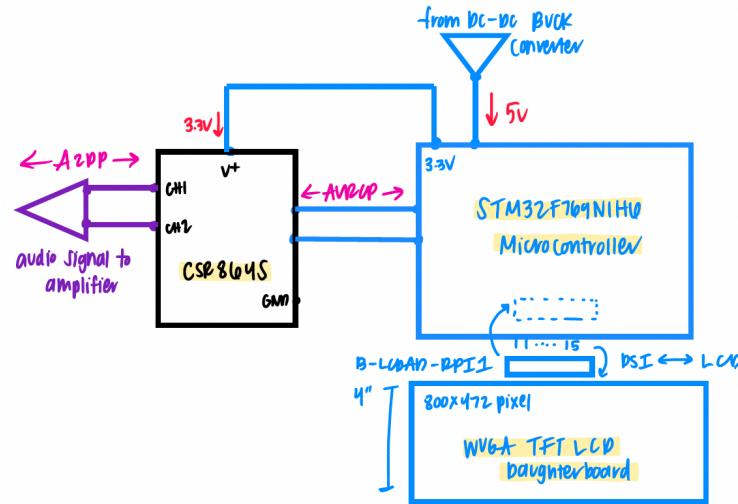
Figure 31: Proposed controls subsystem design with initial component selection

In ordering a microcontroller, we initially selected STM32F769I-DISCO, a kit consisting of a STM microcontroller with connected LCD screen, for the following reasons:

- Sufficient memory: STM32F769NIH6 Arm Cortex-M7 core, with 2MB of flash memory and 532 Kbytes of RAM [40].
- Bluetooth support: Controller supported UART for serial communication, facilitating data exchange and control signals with the originally selected CSR8645 Bluetooth module, integrated with the AVRCP profile
- Embedded LCD: The kit itself included a compatible 4-inch 800 x 472‑pixel capacitive touch TFT color LCD with serial interface.
- STM32 programming environment: Familiar environment used to program, additionally user-friendly and aligned well with our existing knowledge.

In our attempt to program the microcontroller, we faced numerous challenges, including difficulties downloading the necessary toolboxes and encountering critical errors when trying to activate the LCD display. We spent over three weeks configuring the project, but despite referencing completed projects, existing code, and technical datasheets, we were unable to successfully power the LCD. Realizing the significant time investment required to resolve these issues, we decided to shift our focus to finalizing the speaker design.

Given the persistent issues with the microcontroller, we opted for a Bluetooth evaluation board capable of operating without additional programming. This decision reduced the dependency on

the microcontroller and ensured reliable audio transmission while we explored alternatives for the controller and display components.

With the microcontroller's functionality scaled back, we reassigned its role to focus solely on the visual aspect of our design. In the final weeks of fabrication, we decided to implement a Raspberry Pi 4 Model B alongside an external 7-inch monitor. This choice was driven by the Raspberry Pi's user-friendly interface, allowing us to quickly write and program the microprocessor—a crucial factor given our time constraints.

We felt it was important to include a visual aid in our speaker's design, so despite the setbacks, we did not abandon the control or display components entirely. This approach allowed us to integrate a practical solution while working to hopefully give rise to a polished and functional screen come the day of demonstration.

3.3.2. Physical Speaker: Enclosure

(1)   Rationale Behind Design Choices

The internal hardware components are housed in the enclosure modeled below:
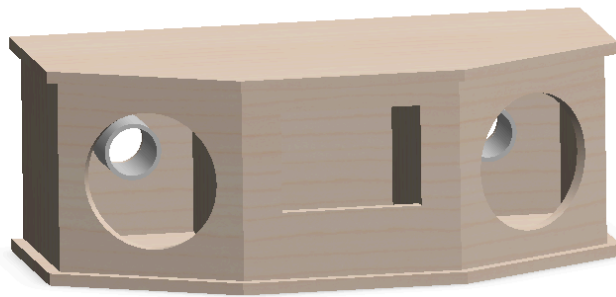
Figure 32: Final enclosure model

The overall shape is hexagonal with one front facing panel, two angled face panels (holding the drivers), two side panels, two rear-facing port panels, and a removable back plate. The ports are 2" x 3.1" pvc pipe pieces. The top and bottom panels have grooves carved out (see Figure X) that match the top profile of all the vertical panels.
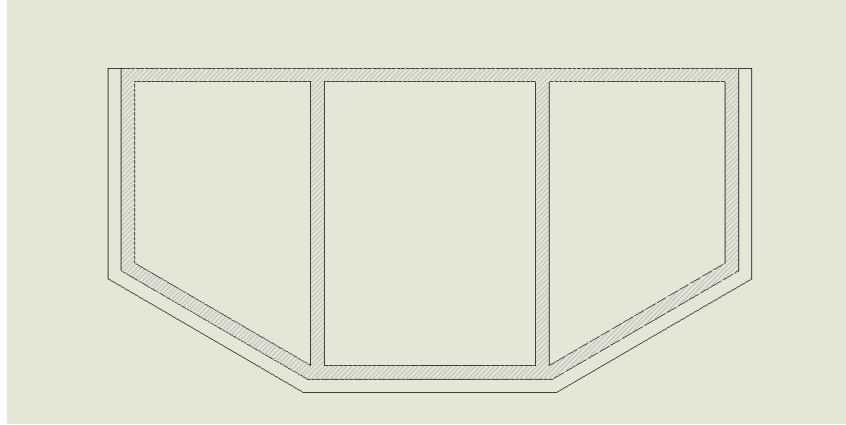
Figure 33: Bottom panel routing trace

Ideally, construction would have used ¾" mdf for the rigidity and density which enhance resonance by better reflecting sound waves. Instead, ½" Plywood is being used for several reasons. First of all, it is cheap and easy to work with. MDF is fairly easy to work with as well, but it releases a lot of fine dust, which is neither good for one's lungs, nor the contents of a classroom. It is also quite heavy, which is a concern for the consumer as well who will have to move the device around. It's also a brittle material meaning that unnoticed deviations in building might cause catastrophic damage.

The internal geometry is designed to strike a balance between the various forms of distortion present in audio systems. The following was considered:

(1) Sound Pressure Level (SPL or Sensitivity): This is the power response of an audio system. A higher SPL means that the speaker is producing more sound volume for less power [55].

(2) Box Frequency: The resonance of the speaker in its enclosure, otherwise known as tunable. With full-range drivers boosting the low frequency ranges is the main priority. In constructing an enclosure, larger boxes allow for lower tuning frequencies, but this is an inverse exponential relationship.

(3) Port Air Velocity: The max speed of the air traveling through the port of an enclosure. Port chuffing occurs when air passes through the port at over 5% the speed of sound, which mathematically equates to 17 m/s.

(4) Group Delay: Average delay of the transfer function. Humans are most sensitive to frequencies within the frequency range of 3 to 4kHz (see Figure 5).
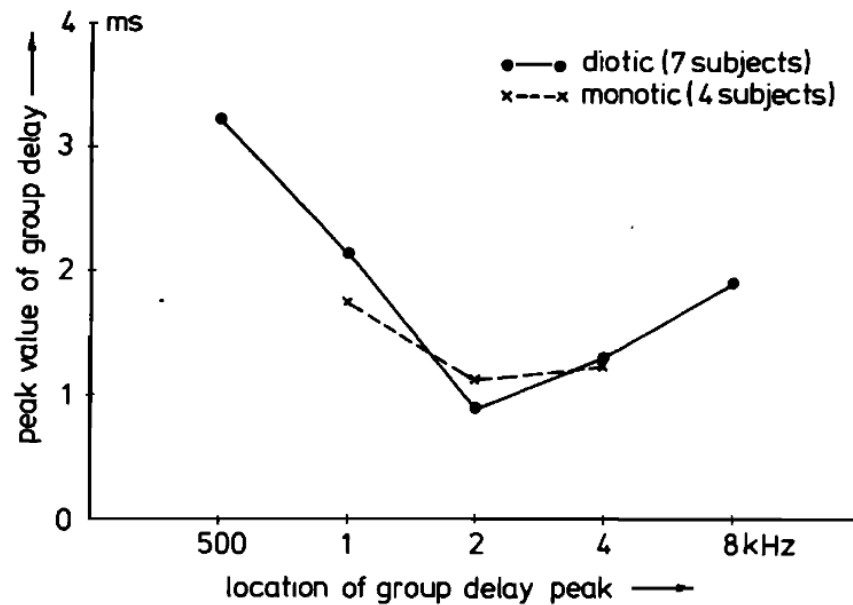
Figure 34: Audibility threshold for delay (Blauert & Laws) [56]

(5) Box Geometry and Standing Waves: The shape of the internal cavity. Standing waves are essentially when the box geometry causes static pressure points due to interference, and it was necessary to develop a design that would minimize their prevalence. As researched, the shape best to obtain an optimal frequency response is a sphere [57], however it is a difficult shape to construct, especially given our time constraints. In our application, the effect of standing waves is not necessarily prevalent, and most online calculators and free audio software (including WinISD) tend to just use the box volume.

(2)   Modifications and Tradeoffs

Much thought went into the design of the enclosure, and the greatest modification came from the shape of the box itself. Initially, without researching the effect of angled ports, we considered developing a rectangular design, with the LCD mounted on top (see Figure 35).
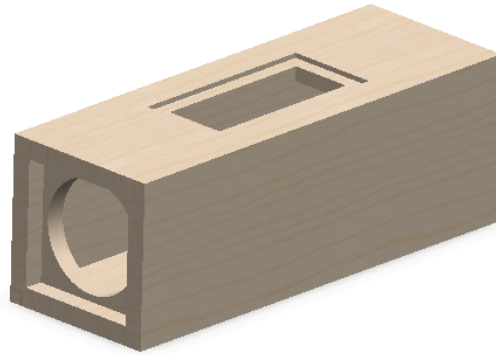
Figure 35: Project proposal enclosure design

Further, the structure was planned to be composed of Medium-Density Fiberboard, but for reasons previously stated, this was not optimal in our design. In changing the rectangular structure to be hexagonal, it was additionally necessary to mount both drivers on the front panel, rather than on each side, as originally proposed.

3.3.3. iOS Application

(1)   Rationale Behind Design Choices

The architecture of our app is designed to integrate Spotify's music recommendation features with personalized user experiences through modular and well-defined components. The block diagram illustrates how each file interacts, providing logical separation of responsibilities.
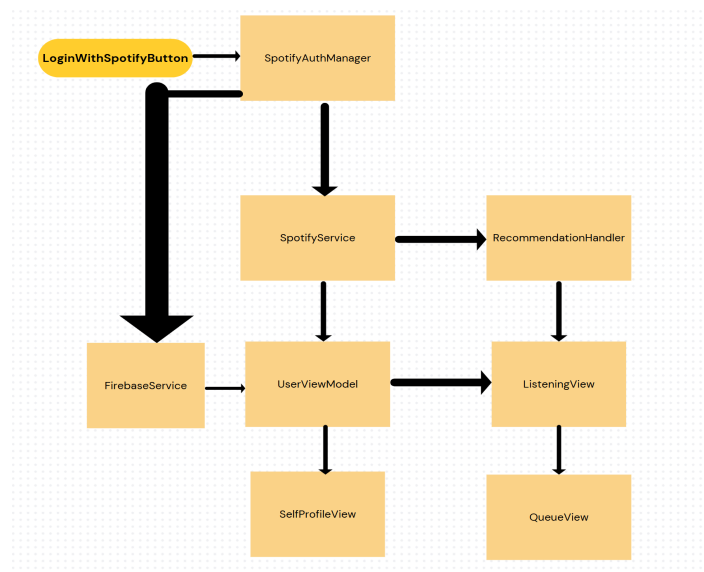
Figure 36: Block Diagram of the Tempo App Architecture

Below is an explanation of each file's role, how it connects to others, and its functionality:

LoginWithSpotifyButton.swift handles user login through Spotify's API. It provides a responsive UI button with a loading state to initiate the authentication flow. When the button is tapped, it triggers the *SpotifyAuthManager* to initiate login. This file is critical to starting the user session and connects directly to *SpotifyAuthManager* for token retrieval.

SpotifyAuthManager.swift is responsible for secure authentication, utilizing OAuth 2.0 to generate and refresh access tokens. OAuth 2.0's proven security and scalability in mobile applications make it ideal for handling user data and ensuring uninterrupted API access, as demonstrated in [41]. Once the user logs in using the Spotify button, this manager obtains the access token and stores it securely. The token is refreshed hourly through Spotify's refresh token mechanism, ensuring continuous access to the Spotify API. This file interacts with SpotifyService, supplying the token required for API calls.

SpotifyService.swift acts as the main interface for interacting with Spotify's API. It uses the token from *SpotifyAuthManager* to fetch user data, playlists, top tracks, and artists. It also plays tracks via Spotify's App Remote SDK. This file connects to *RecommendationHandler* for processing user prompts into actionable API queries and communicates with *UserViewModel* to store or display fetched data.

RecommendationHandler.swift processes user input (e.g., mood, genre, activity) into API queries. Its design aligns with studies that validate mood- and activity-based filtering as effective for music recommendation systems [42]. It fetches song recommendations by leveraging Spotify's audio features, such as valence and danceability, to tailor results to user preferences. For example, a user entering "relaxing study music" would receive tracks with low energy and high valence. This component relies on *SpotifyService* for data fetching and provides results to *ListeningView*.

UserViewModel.swift is the central state manager, responsible for tracking user preferences, profile data, and Spotify insights (e.g., top tracks and playlists). It fetches data from *FirebaseService* and *SpotifyService* and ensures UI components like *QueueView*, *ListeningView*,

and *SelfProfileView* reactively update with user actions. It acts as the bridge between the backend services and the front-end views.

FirebaseService.swift handles storage and retrieval of user data (e.g., preferences, compatibility scores, saved tracks) from Firebase Firestore. Firebase's real-time synchronization capabilities and scalability make it a practical choice for managing continuous data across user sessions, as supported by [43]. This service supports saving and loading user profiles, preferences, and top songs for persistence across sessions.This service feeds data to *UserViewModel* and receives updates from UI components.

SelfProfileView.swift is responsible for displaying the user's profile, including their name, profile picture, top tracks, and compatibility scores. It pulls data from *UserViewModel*, which aggregates information from Firebase and Spotify.

QueueView.swift manages and displays the user's active listening queue. It shows the currently playing track, the upcoming playlist, and playback controls like play/pause and skip. It relies on *SpotifyService* for playback functionality and updates dynamically as songs are played or skipped.

ListeningView.swift serves as the primary interface for users to interact with the app's music recommendation feature. By leveraging natural language processing (NLP) to process user prompts, the interface delivers a user-friendly and conversational experience. This approach has been validated as an effective method for enhancing user interaction in mobile applications [44]. It uses a conditional display mechanism to toggle between two views: *ChatView* and *QueueView*, based on the state of the send variable. Initially, it displays *ChatView*, where users can enter natural language prompts (e.g., "happy, upbeat music for working out") to request music recommendations. These prompts are processed by the *RecommendationHandler* via *SpotifyService,* which retrieves song recommendations from Spotify. The fetched songs are stored in the songs array, which is then passed to *QueueView* when send is toggled to true. Additionally, *ListeningView* ensures that the user's profile-related state (isProfile) is reset to false on appearance, indicating its distinct functionality from profile-related screens. It seamlessly integrates *SpotifyService* to fetch recommendations and provides a dynamic, chat-like interaction for users to refine their playlists, offering a smooth transition to playback within *QueueView*.

The *LoginWithSpotifyButton* initiates the user session by triggering authentication via *SpotifyAuthManager*. Once authenticated, *SpotifyAuthManager* provides tokens to *SpotifyService*, which interacts with Spotify's API to fetch user data, playlists, and track recommendations. *RecommendationHandler* translates user input into API queries, while *FirebaseService* persists user data. *UserViewModel* integrates data from these services and updates UI components such as *ListeningView*, *QueueView*, and *SelfProfileView*. Together, these components create a seamless flow between user actions, backend logic, and front-end updates.

Spotify token generation and refresh workflows are automatic, ensuring the user remains logged in without manual intervention. This decision minimizes friction in user sessions and leverages Spotify's hourly token expiration system for security.

(2)   Modifications and Tradeoffs

While our original vision included a BPM-based matching feature to adjust song playback tempo to user preferences, Spotify's API prohibits modifying songs directly or manipulating offline downloads. As a result, we pivoted to a system that curates playlists based on moods, genres, and activities, using Spotify's existing audio features alongside OpenAI's ChatGPT. To support this system, we implemented automatic Spotify token generation in the backend, as the Spotify API requires tokens to refresh every hour. This backend solution then provides uninterrupted functionality without user intervention, streamlining the authentication process.

Compatibility scoring was intentionally simplified to focus solely on shared songs rather than incorporating more complex metrics like danceability or energy levels. This decision prioritizes efficiency, enabling quick calculations without unnecessary computational overhead. Similarly, the app avoids processing or storing large amounts of data locally, instead leveraging Firebase for real-time cloud storage and synchronization. While this approach minimizes device storage requirements, it does introduce a reliance on stable network connections. By basing compatibility comparisons exclusively on shared songs, we reduced complexity while ensuring scores are grounded in universally understood metrics—shared musical interests—rather than subjective audio features.

Prompts in the ListeningView are designed to encourage users to include mood, activity, and genre because these parameters yield the most accurate Spotify recommendations based on the platform's API algorithms. For example, specifying "happy, upbeat music for working out" yields better results than generic queries like "good songs."

We utilized GitHub for version control, as it enabled efficient collaboration, branching, and issue tracking. By dividing front-end and back-end development into separate branches, we ensured parallel progress and reduced bottlenecks. However, this required careful integration to align data formats and UI behavior between the two layers. Xcode was chosen as the IDE for its seamless integration with Swift, iOS development tools, and real-device testing, which were crucial for refining the app's user experience.

Overall, our deviation from the BPM-matching feature reshaped our roadmap but allowed us to deliver a polished product within the constraints of Spotify's API. By focusing on specific recommendations, compatibility scoring, and profile management, we ensured the app remains practical, user-friendly, and aligned with real-world limitations.

3.4.   Test Plans and Verification

The following outlines the strategies used to validate the functionality of each subsystem and the overall performance of the project. It includes details on modifications to the initial test plans, along with explanations of any redesigns implemented based on testing results to ensure the system met its intended objectives.

3.4.1. Physical Speaker: Internal

Initially, to test the functionality of the PCB, first we sent the design to FreedFM, which yielded no Showstoppers (see Figure 32).
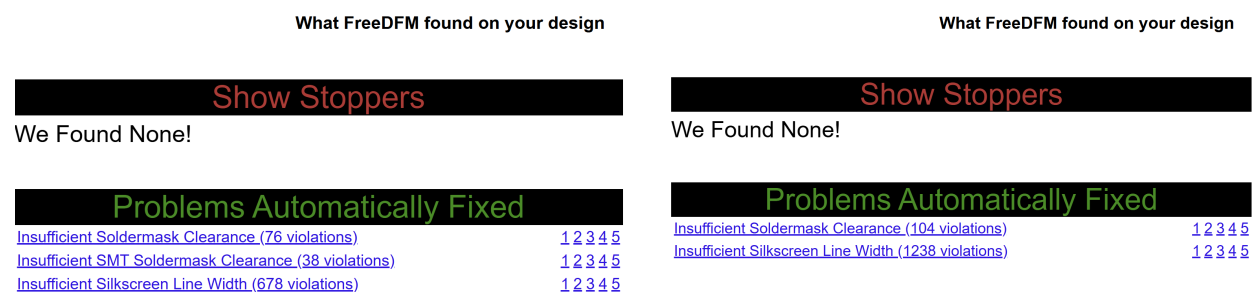
Figure 37: PCB revisions 1 & 2 – FreeDFM test results

When the board arrived, we followed the outlined plan:

1. Power Supply Circuit Testing (Standalone)
   a. Assemble the power supply circuit – AC/DC, switch, capacitors, fuse
   b. DC-DC buck converter
2. DC-DC Buck Converter & PCB
3. Audio Circuit Testing (Standalone)
   a. Amplifier, amplifier with speakers
4. Bluetooth Module Testing
5. Raspberry Pi w/ Display
6. Final Integration Testing
   a. Combine the audio circuit (amplifier and speakers), Bluetooth module, microcontroller with the LCD – Simulate power supply
   b. Integration of power supply

The power supply circuit was assembled as the foundation for the entire system, including the AC/DC conversion, switch, capacitors, and fuse. The initial testing involved verifying the AC to DC conversion, ensuring that the system provided the appropriate voltage and current needed by the rest of the components. After confirming that the power supply was working as expected, we proceeded to test the DC-DC buck converter, which regulated the voltage to the required levels for the other subsystems. We verified the buck converter's performance by measuring output voltages at different load conditions. The power supply circuit was also tested for thermal performance, ensuring that the capacitors and other components did not overheat during continuous operation. This testing led to no changes and was as expected.

The audio circuit, which consists of the amplifier and speakers, was tested standalone to verify its functionality. First, we tested the amplifier by providing a test signal from a known audio source and monitoring the output to the speakers. We confirmed that the amplifier correctly boosted the audio signal without distortion and that the speakers responded appropriately. Once the amplifier's functionality was validated, we integrated it with the speakers and ran a series of tests to check for audio clarity, volume output, and signal integrity. The testing also included examining the behavior of the amplifier at various power levels to ensure the circuit did not experience overcurrent or other issues. This testing led to no changes and was as expected.

The Bluetooth module was tested independently to ensure that it could communicate with other devices and establish a stable connection. Initial tests involved pairing the Bluetooth module with a smartphone or other Bluetooth-enabled device. Once paired, we tested the module's ability to send and receive audio signals, verifying that there was no noticeable delay or distortion in the audio. The Bluetooth module's range was also evaluated to ensure reliable communication over typical usage distances. Signal strength and connectivity stability were tested by moving the paired device at different distances and through various obstructions to confirm the Bluetooth module's performance under real-world conditions. This testing led to no changes and was as expected.

We had planned to test the Raspberry Pi with the display to ensure that the microcontroller could correctly interface with the screen and display the necessary information for user interaction. Our plan involved loading the required software and confirming that the Raspberry Pi could process input and output signals to the display. We also intended to test the touchscreen functionality, ensuring that inputs were recognized and translated into actions within the app interface. Additionally, we had planned to monitor the system's performance for any lags or errors during display interaction and verify that the visual output met our expectations. Furthermore, we aimed to test the Raspberry Pi's ability to handle the full processing load without overheating or performance degradation. However, due to time constraints and unforeseen issues, we were unable to fully carry out these tests.

The final integration testing combined all subsystems to ensure they worked seamlessly together. First, we integrated the power supply with the audio circuit, Bluetooth module, and microcontroller. The power supply was simulated under different loads, confirming that it could handle the combined current demands of all components. Next, the audio circuit (amplifier and speakers) and Bluetooth module were tested together to verify full system functionality. This included testing audio output from the Bluetooth connection and verifying that the display correctly interacted with the app, and all components powered on without issues. We also evaluated the overall stability of the system, checking for any interference, noise, or other issues that might affect performance. Finally, the integrated system was tested for power consumption to ensure that it met the specifications and could operate reliably under expected conditions. This testing led to no changes and was as expected.

3.4.2. Physical Speaker: Enclosure

The test plan measured the technical aspects mentioned above. To test, we used an Analog Discovery 2 plugged into the amplifier via a 3.5mm audio cable. Leveraging the power output from the amplifier, we used waveforms and the AD2 multimeter to run a frequency sweep for the frequency response. This was run in tandem with a phase analysis to yield the group response, a spectrum analysis measure the harmonic distortion, and port chuffing was determined by ear to be sufficient.

3.4.3. iOS Application

The test and integration plan was designed to validate the functionality of individual subsystems and provide cohesive performance across the entire app. The process followed a phased approach, with each phase addressing specific components of the system. The goal was to ensure functionality of the frontend and backend, and further integrate.

(1)   Profile and Preferences Synchronization

In this phase, testing focused on making sure user profile data and preferences were correctly synchronized with Firebase and Spotify. The *UserViewModel* needed to fetch and synchronize data such as the user's name, profile picture, and preferences. Alongside, the *SelfProfileView* needed to display this data accurately, toggling to allow users to modify their preferences, and saving these updates back to Firebase. During simulations run through XCode, synchronization issues were identified, requiring adjustments to the *UserViewModel*. Mock data was used to first simulate user profiles, providing proper retrieval and display before live data integration. These updates resolved discrepancies in data synchronization, providing a stable foundation for the user profile system and later successful login with real profiles.

(2)   Questionnaire Navigation

The second phase validated navigation between *QuestionnaireOneView* and *QuestionnaireTwoView* during the onboarding process. Testing needed to make sure that users could resume incomplete setups using *UserViewModel.fetchCurrentStep* to determine the correct starting point. Transitions between questionnaire steps were also implemented using *AnyTransition.moveTransition* for smooth navigation. Initial tests revealed inconsistencies in

handling partially completed setups, which were resolved by refining navigation logic. The *saveDataAndProceed* function in both questionnaire views was also tested to see that user data was saved correctly at each step.

(3)   Explore and Compatibility Features

Phase 3 addressed the implementation of the Explore and Compatibility features, focusing on the graphical Compatibility Web and list-based compatibility scores. The *calculateCompatibilityBetweenUsers* function was tested to see that it accurately computed scores based on shared Spotify data, such as liked songs. Compatibility scores were displayed in *ExploreView* using the proper *CompatibilityPageView*. Early tests with mock data confirmed the correctness of the calculations and display logic and integration with Firebase allowed dynamic fetching of other users' data, which exposed conflicts between compatibility and recommendation logic in the *UserViewModel*. These issues were resolved by isolating compatibility-specific functionality in its own respective file.

(4)   Listening and Recommendations

The fourth phase focused on the Listening and Recommendations subsystems. The *SpotifyService.fetchRecommendations* function was tested to see that it retrieved songs accurately based on user prompts, such as "relaxing acoustic vibes." *ListeningView* was also tested to confirm that prompts dynamically updated the song queue, and *QueueView* was validated for playback controls, including play, pause, skip, and previous track functionality. Testing revealed repetitive track suggestions and latency in queue updates. These issues were resolved by prioritizing data retrieval. Integration with the Tempo Bluetooth speaker was also tested to confirm synchronization of playback controls and real-time metadata display.

(5)   Error Handling and User Feedback

The final phase prioritized error handling and meaningful user feedback for common issues. Testing focused on asynchronous operations in *FirebaseService* and *SpotifyService*, validating that errors such as network failures or invalid inputs were handled. User-friendly alerts were implemented to notify users of issues, and loading indicators were added to improve user experience during network operations. Tests revealed missing error-handling cases and

inconsistent feedback for invalid inputs, which were addressed by refining error-handling logic across all services.

(6)   Changes in the Test Plan

The updated test and integration plan reflects an evolution from the original test plan outlined in the proposal. While the foundational structure and core objectives remain consistent, additional details and refinements were incorporated during the development process to address unforeseen challenges and optimize performance.

Profile and Preferences Synchronization

The original plan focused on testing user authentication and ensuring accurate synchronization between user profile data, preferences, Firebase, and Spotify. The updated plan builds on this by detailing the iterative debugging process during simulations in Xcode, where synchronization issues in *UserViewModel* were resolved using mock data. Specifically, the fetching and saving of user data (e.g., name, profile picture, preferences) in *UserViewModel.fetchUserProfile* and *UserViewModel.saveUserProfile* were rigorously tested. These efforts provided a stable foundation for live data integration and ensured profile synchronization worked seamlessly with real user data.

Questionnaire Navigation

Navigation testing during the onboarding process was refined in the updated plan to handle edge cases like incomplete setups. The use of *UserViewModel.fetchCurrentStep* to resume progress and smooth transitions via *AnyTransition.moveTransition* ensured a seamless user experience. Additionally, the *saveDataAndProceed* function in both *QuestionnaireOneView* and *QuestionnaireTwoView* was tested to confirm that user data was saved at each step, resolving inconsistencies in handling partially completed setups. These aspects were not explicitly outlined in the original plan, demonstrating a shift toward more robust handling of edge cases.

Explore and Compatibility Features

The original plan included testing compatibility features such as the Compatibility Web and list-based compatibility scores. However, the updated plan expanded significantly by

incorporating mock data to validate the *calculateCompatibilityBetweenUsers* function and isolating compatibility-specific logic into its own file, *CompatibilityHandler.swift*. This refinement resolved conflicts between compatibility logic and recommendation logic in *UserViewModel*. These additional steps highlighted a stronger focus on resolving integration challenges and refining functionality.

Listening and Recommendations

The original plan outlined testing for music recommendations and playback controls, but the updated plan added substantial details. It focused on resolving latency issues in *QueueView*, addressing repetitive song suggestions, and testing the synchronization of playback controls and metadata display with the Tempo Bluetooth speaker. The *SpotifyService.fetchRecommendations* function was tested extensively to ensure it retrieved songs accurately based on user prompts (e.g., "relaxing acoustic vibes"), and the dynamic updates to the song queue in *ListeningView* were validated. These refinements highlighted a deeper emphasis on optimizing performance and integrating seamlessly with hardware.

Error Handling and User Feedback

While the original test plan broadly mentioned error handling, the updated plan provided a more comprehensive approach. Testing focused on asynchronous operations in *FirebaseService* and *SpotifyService*, validating scenarios such as network failures and invalid user inputs. Improvements were made to handle errors effectively using custom alerts in *ErrorHandler.swift*, providing user-friendly notifications for common issues. Loading indicators were also implemented in key views, such as *ListeningView* and *SelfProfileView*, to enhance user experience during network operations.

(7)   Summary of Changes

The updated test plan retained the core phases outlined in the original plan but introduced refinements to address real-world challenges encountered during development.

New details included:

- Enhanced navigation testing to handle edge cases in the onboarding process.
- Iterative debugging of profile synchronization using mock data, with a focus on *UserViewModel* functions.
- Isolating compatibility logic into *CompatibilityHandler.swift* to resolve integration conflicts.
- Real-time testing of playback functionality with the Bluetooth speaker, including *QueueView* updates and *SpotifyService.fetchRecommendations*.
- A deeper focus on error handling and user feedback through *ErrorHandler.swift* and improved UI components.

These changes reflect a shift toward a more iterative and user-centered testing approach, ensuring that the system is robust, cohesive, and optimized for real-world use.

3.4.4. *Integrated Product*

Integration of the hardware and software components will consist of verifying bluetooth connectivity to where the songs can load on the app and be sent to the speaker for playback. The process will involve connecting an iPhone to the bluetooth accessory named QCC5181. From there the app can be loaded and a song from the recommendation queue can be heard through the speaker. The app will remain fully functional while the speaker is playing as the user can switch across various views while listening to the clear audio output.

4. Physical Constraints

4.1. Design and Manufacturing Constraints

4.1.1 Tempo Hardware

Designing the Tempo hardware presented several constraints, primarily related to integration, functionality, and spatial efficiency. The compact speaker enclosure required careful arrangement of components such as the Raspberry Pi, amplifier board, and Bluetooth evaluation board to ensure optimal performance while minimizing electrical noise and managing heat dissipation. Power distribution posed another challenge, as the system needed to provide stable voltage levels to multiple subsystems, necessitating the inclusion of a reliable power distribution PCB. Additionally, the design had to accommodate user accessibility for external features like the volume control and audio input while maintaining the structural integrity of the enclosure. These factors required iterative refinement of the layout and close attention to detail to ensure seamless operation.

On the manufacturing side, constraints included material selection, component mounting, and heat management. The enclosure was designed to house all components securely while providing ventilation to prevent overheating. Components like the PLA base plates were precision-crafted to fit each subsystem, incorporating cutouts for airflow and heat dissipation. The manufacturing process also required consideration of assembly complexity, as ensuring efficient and secure connections between components was vital to the system's reliability.

The biggest constraint for the design of the enclosure itself was size for several different reasons. With the original intention to be a portable speaker, the design started out quite small (See Fig #) at only H4" x W12" x D4". However performance in drivers and enclosures that small is quite poor with high corner frequencies and low bass resonance, and so quality was given priority over size. The problem is that enclosure sizes and complexity can start to creep if that is the only consideration. To that point, performance was usually limited to the minimum distortion of a given type that a human could hear.

4.1.2   Tempo Software

The software-side constraints included handling rate limits and authentication requirements for APIs. The Spotify API required careful implementation of access tokens for properly accessing music. Similarly, the OpenAI API, though effective for recommendation generation, imposed both financial and functional limitations, as extensive prompt processing could increase costs.

4.2.   Tools Utilized

4.2.1   Tempo Hardware

The following includes the tools utilized to give rise to the internal hardware related to the speaker design:

- **Component ordering:** Amazon, AliExpress, Mouser, Digikey, Switchcraft
- **Programming:** Raspberry Pi Imager, Raspberry Pi OS
- **Design and modeling:** TI Webench Simulator, KiCad, Solidworks, Ultimaker Cura, Audacity, Nebo, OSHPark, FreedFM, AdvancedPCB
- **Physical arrangement:** Soldering iron, wire cutters, scissors, screwdriver, saw, ruler
- **Testing:** Multimeter, DC supplies, current-draw module, Spotify, computer, 3.5mm jack

The following includes the tools utilized to give rise to the enclosure of the speaker:

- **Audio design:** WinISD
- **Audio testing:** Analog Discovery 2, VirtualBench, Waveforms, LabView
- **Forming Tools:** Circular saw, electric drills (various), router, electric sander, sandpaper, hand plane, chisel, hacksaw
- **Building Tools:** CLAMPS, wood glue, wood epoxy, wood filler, drywall putty, scale prints (Staples), screws, heat gun (curing), shop towels

4.2.2   Tempo Software

We utilized Xcode IDE as the development environment for writing, debugging, and testing Swift code for the iOS application. Xcode Simulator was used for testing the application on virtual devices without requiring physical hardware.

 In terms of collaboration, GitHub was sufficient for coding and saving as a group. As previously mentioned, SwiftUI served as the primary framework for designing the user interface. Additionally, Spotify API offered the access to user-specific data, such as liked songs, playlists,

and listening history, while OpenAI API was used for natural language processing capabilities including generating personalized music recommendations.
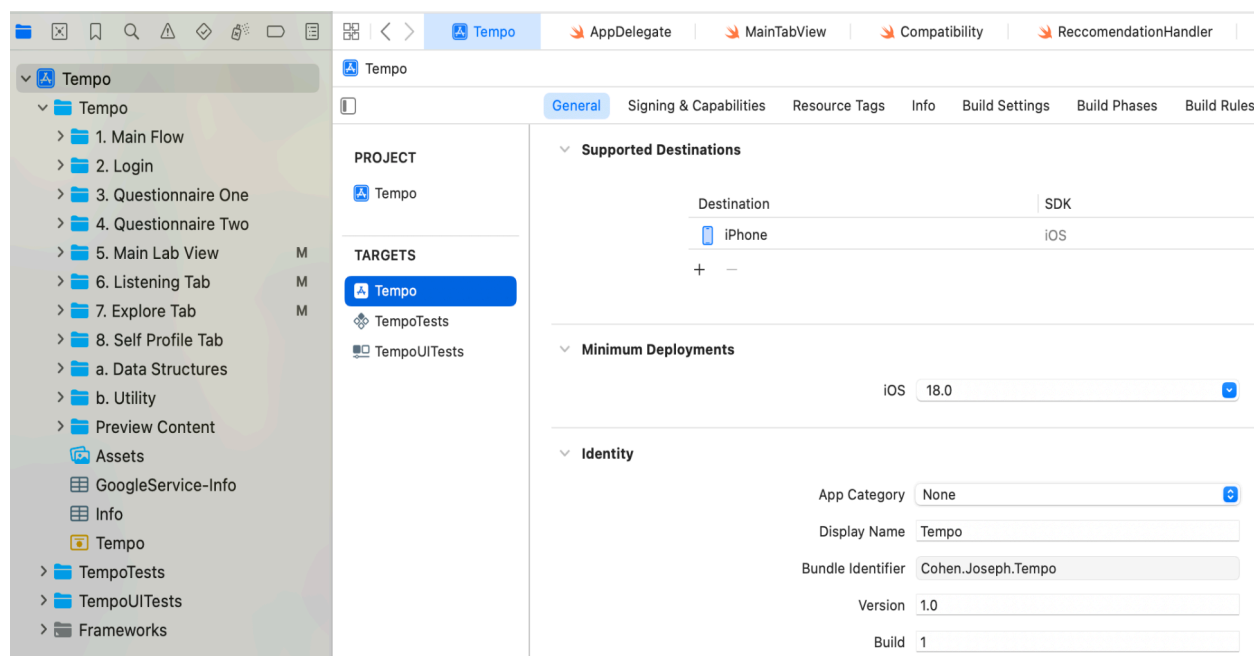
Figure 38: Xcode IDE View

4.3.   Cost and Fabrication

Figures related to our budget outline can be found in the Appendix. In total, we spent $315.94 on components in class orders, which contributed to the overall budget for the project. These expenses covered various parts essential for the assembly and functionality of the hardware, such as PCBs, the power supply, the audio system, and Bluetooth components. The detailed breakdown of each order and remaining costs is documented in the respective files (Tempo-PartsOrder1.xlsx, Tempo-PartsOrder5.xlsx, Tempo-PartsOrder9.xlsx, Tempo-PartsOrder10.xlsx).

We ordered several key components for the project to ensure the successful completion of the speaker. From TTI, we ordered 2 units of the RAPC10U for a total of $28.98, which were needed for specific audio-related functions. Additionally, we purchased a 2x50W audio amplifier from Parts Express for $188.78, which was essential for powering the speaker. To pair with the amplifier, we ordered two 8-ohm speaker drivers from Parts Express.

For the custom PCBs, we ordered 3 units of Rev. 1 PCBs from OSHPark at $100.40, followed by 3 units of Rev. 2 PCBs for $131.20. These PCBs were integral to the system's design and electrical integration. We also ordered two binding mounts and binding posts from Parts Express, totaling $64.29, to securely mount and connect the wiring and components. In total, these personal orders amounted to $513.65, ensuring all necessary hardware was available for the assembly and testing of the physical speaker. Not included is the cost of the Bluetooth board, which was about $50, and the Raspberry Pi, which we borrowed from Professor Barnes.

Not tabulated are the costs associated with the physical assembly of the speaker, as we had many of the parts on-hand. Thomas spent a bit extra to order his own tools to fabricate the enclosure, but he saw it as an investment for future work.

Tempo's software fabrication process had minimal costs attributed to its development and function, but the API-related costs, particularly from OpenAI, were a constraint. During development, $25 was spent on the OpenAI API for testing and implementation. The group did not use the entire $25 added to the OpenAI account; less than 7 dollars was actually spent dependent on our token usage.

In total, we went about $350 over budget, due to unforeseen changes, updated orders, and not having a clear sense of what the design of the project would ultimately entail on the hardware side. The software budget was minimal, but we did not consider possible hardware-related budget implications. This lack of clarity led to additional component purchases and revisions, which added to the overall cost. In the future, it is necessary to establish a more defined and detailed project plan early on, with clear hardware and design specifications, to prevent unexpected costs. Regular communication and alignment among team members will also help avoid miscommunication, ensuring that all hardware requirements are accurately accounted for from the start.

4.4.   Production Consideration

If the Tempo hardware were to be mass-produced, several production considerations would need to be addressed to optimize cost, scalability, and quality. Component sourcing would be a critical factor, requiring reliable suppliers to ensure consistency and minimize lead times for key parts

like the Raspberry Pi, Bluetooth evaluation board, and amplifier. Custom PCBs and base plates would benefit from economies of scale, reducing per-unit costs while maintaining high-quality manufacturing standards. Simplifying the design for assembly would be essential, potentially involving pre-assembled modules and snap-fit enclosures to streamline production and reduce labor costs. Material selection would also play a significant role, as alternative options to PLA or higher-grade plastics might improve durability or thermal performance in a consumer-grade product. Additionally, rigorous quality control processes would need to be implemented at every stage to ensure that each unit meets performance and safety standards, while packaging design would need to focus on protecting the product during transit and providing a polished, market-ready presentation.

To address software-related constraints in a production environment, efforts would focus on optimizing API calls, exploring alternative natural language processing models, and improving data handling to reduce dependency on external services. These steps would overall help provide a cost-effective and scalable solution

5.   Societal Impact

Although Tempo can aid in the discovery of music, help increase social connectivity, and offer interactive accompaniment hardware, the project also raises societal concerns related to cultural trends, safety, environmental, and economic factors. These considerations are all critical not only for maintaining the ethical integrity of the system but also for addressing potential risks to stakeholders such as users, musicians, content creators, regulators, and environmental agencies.

5.1.   Bias, Creativity, and Cultural Trends

Artificially intelligent recommendations have the potential to change the way users discover and interact with music entirely. While these recommendations offer convenience and personalization, the current algorithms, such as those used by Spotify, risk confining users to narrow content that already aligns with their existing tastes. These isolating filter bubbles in music recommendation systems result from a combination of factors, including algorithmic design priorities such as engagement metrics, and commercial, profitable interests. This system can then be designed to maximize user interaction and retention, at the cost of promoting and prioritizing diversity. As a result, users may become less likely to encounter new or unfamiliar music, limiting their cultural exposure and reinforcing a cycle of sameness. Additionally, certain artists who post their music on Spotify may not be recommended if they do not fit into these filter bubbles [45]. This issue contains broader implications for cultural consumption as it shapes not only individual listening habits but also societal trends in music discovery and which voices get prioritized over others [46].

To address this, Tempo's recommendations do not rely on existing user preferences or listening history to generate recommendations, reducing the risk of filter bubbles within the system. With this recommendation system, users may have less frustrations over repetitive content and can diversify their listening experiences[47]. Additionally, musicians of all genres may benefit from increased visibility as the recommendations do not prioritize mainstream, already liked content. Although no algorithm is unbiased, as the code is a direct reflection of the developer and their

inherent bias, our algorithms attempt to reduce this bias and make music discovery more equitable for all.

In analysis of stakeholders, music consumers benefit from broader exposure to diverse musical content, artists and musicians gain increased visibility - particularly independent and emerging artists who did not have the chance previously. Additionally in terms of impact for secondary stakeholders such as music labels, the labels will need to adapt to a more democratized recommendation landscape and will have to monitor shifts in cultural trends in a way they have not before.

5.2.   Safety & Privacy

With the system, safety and privacy are another key concern. The application collects and processes sensitive user data in the profile and compatibility tabs such as listening habits and social interactions, which raises questions about data security and ethical usage. With this responsibility in mind to protect data, all user data will be stored and saved on Firebase, a reputable platform backed by Google that stores and syncs data in real time. As a result, protecting privacy not only builds user trust but also aligns with ethical obligations to safeguard sensitive information [48]. Additionally, within the app, users have the option to put their profiles on private, also controlling what data they want visible such as top songs, top genres, and playlists regardless of being on public or private mode. Thus, the Tempo app provides a safe environment that respects privacy.

Overall, end users receive protections of personal data and increased control over privacy settings, and platform administrators such as Firebase hold the responsibility for maintaining security protocols.

5.3.   Health & Wellness

From a physical health perspective, the project mitigates potential risks associated with Bluetooth radiation. By encouraging users to interact with a speaker rather than devices held close to the body, there is a lesser exposure to the radiation [49]. While current research suggests radiation caused by headphones is likely minimal and further impacts would have to be studied within the next several decades, promoting speaker use aligns with public health priorities.

Additionally, the system will integrate volume control and smart sound adjustments to help minimize noise pollution, a growing concern linked to hearing damage, stress, and sleep disturbances [50]. These measures then make sure the system prioritizes both individual safety and broader community welfare.

As a result, users experience reduced exposure to radiation with the benefit of controlled audio levels, and medical professionals can support the use of the speaker, considering effects on well-being and potentially study long term effects of technology use.

5.4.   Environmental Impact

AI systems and their associated infrastructure consume significant energy, contributing to carbon emissions which needs to be addressed. The energy consumption of AI systems, particularly in data processing and storage, contributes significantly to carbon emissions. Open AI's Chat GPT consumes over half a million kilowatts of electricity each day where a single ChatGPT conversation uses about 50 centimeters of water, equivalent to one plastic bottle [51]. Our project then prioritizes energy-aware AI usage to minimize its environmental footprint. The recommendation system is dependent on ChatGPT 3.5, a less powerful and complex model than the most recent 4.0 model, thus keeping in mind of energy consumption. Additionally, the recommendations are the only part of the app which depends on AI, as the compatibility and profile tabs were coded to not be dependent on AI, only to fetch data from Firebase.

To elaborate, environmental protection agencies and regulators can gain access to an implementation of sustainable AI practices in consumer technology through our transparent reporting. Additionally, climate research institutions can gain valuable data on energy consumption patterns in AI driven consumer products as all of our data is accessible and stored on Firebase.

5.5.   Economic Considerations

The project has significant potential to contribute to the music and technology industries, providing innovation and economic growth. However, affordability and accessibility must be prioritized to make sure the platform is inclusive and not excessively expensive to reach a broader audience. As our target customer is for users with Spotify Premium, the app will be of

no additional cost to users than what they already pay for Spotify. Additionally, since the recommendations use ChatGPT 3.5, which is available to the public for free, there is no additional cost in this regard. Overall, a free-tier application model has made the system accessible. Overall, end users benefit from cost effective access through Spotify Premium subscriptions, eliminating additional fees. Additionally, musicians and content creators gain improved, equitable visibility through the reduced bias of the recommendation system for profit. In terms of platform partners, companies involved in the process of this project will gain visibility from the project's popularity.

Secondary stakeholders consist of music industry professionals such as producers, labels and distributors who may experience shifts in content discovery patterns and will have to keep up to date with AI technology and user behaviors to propose new strategies for profit [52]. Although shifting away from traditional promotion and distribution strategies may be difficult at first, music labels can develop programs that emphasize artistic development and quality content creation, knowing that recommendations are based on musical merit rather than existing popularity or marketing spend. This might involve investing more resources in artist development and production quality, since visibility will be earned through the inherent appeal of the music rather than inherent promotional algorithms.

5.6   Summary

By addressing the societal impacts the project seeks to balance technological innovation with ethical responsibility. Incorporating safeguards, transparent practices, and inclusive features ensures the system benefits a wide range of stakeholders while minimizing risks. A focus on sustainability, diversity, and accessibility will enable the platform and speaker to make a positive and lasting contribution to society.

6.   External Standards

6.1.   Software Standards

Tempo adhered to various external standards and regulations to provide high quality, reliability, and safety for the end-user. A key focus was meeting Apple's App Store Review Guidelines, which saw that the app maintained a high standard of performance, design, and security. Compliance with these guidelines included designing a clean, responsive, and easy to use user interface aligned with Apple's Human Interface Guidelines, implementing security protocols for protecting user data, and optimizing app performance to run across all supported iOS devices [53] In light of growing concerns over data security and user privacy, the Tempo app also adhered to the ISO/IEC 27001 standard, which outlines best practices for implementing and maintaining an information security management system (ISMS) [54]. The app incorporated encryption for sensitive user data, and secure authentication mechanisms to minimize vulnerabilities. These measures then saw that user information, including musical preferences and social connections, was handled securely.

6.2.   Hardware Standards

For the physical Bluetooth speaker, the project adhered to Bluetooth Classic standards, utilizing the A2DP protocol for audio streaming and AVRCP for remote playback control [55]. These standards then enabled reliable communication between the speaker and iOS devices, providing a consistent and high-performance audio output. Additionally, IEC 60065 standards for audio and video equipment safety and IEC 61000 standards for electromagnetic compatibility (EMC) were considered to see that the speaker met safety and interference requirements for operation in diverse environments [56][57]. To guarantee high audio performance, the speaker also followed IEC 60268, which outlines testing and measurement methodologies for audio equipment [58]. These standards allowed the development team to verify the speaker's performance against defined benchmarks for sound quality. Furthermore, components within the speaker were designed to comply with RoHS directives, restricting the use of hazardous substances to protect both users and the environment [59]. The speaker also adhered to FCC Part 15 regulations,

mitigating electromagnetic interference, and achieved compliance with CE Marking standards to meet European safety and environmental legislation [60].

By adhering to these diverse external standards, Tempo offers a secure, and legally compliant product. These considerations laid the foundation for trust, reliability, and operational integrity, allowing Tempo to comply in both the app and hardware regards.

7.   Intellectual Property Issues

In evaluating Tempo's patentability, it is important to assess relevant prior art and determine how Tempo's design compares to existing intellectual property. This section references three U.S. patents with claims similar to Tempo's core features and analyzes their impact on the app's patentability.

Patent 1: Natural Language Query for Content Recommendations

The patent reference is of the U.S. Patent 9,762,607: *System and method for content recommendation based on natural language input* [61]. Relevant claims are as follows where claim 1 (Independent) describes a system that processes natural language input to retrieve and recommend content based on context, preferences, and metadata. Additionally, claim 5 (Dependent) further specifies that the system incorporates machine learning algorithms to refine recommendations based on user feedback.

Tempo's recommendation system aligns with claim 1 as it processes natural language inputs (e.g., "chill tracks for studying") using OpenAI's GPT; however, Tempo differs in that it does not depend on user preferences or behavioral data, relying instead on direct query processing and Spotify API data retrieval. Since claim 5 specifically includes user feedback for refining recommendations, it does not overlap with Tempo, which generates recommendations exclusively based on input queries. Additionally, Tempo introduces a novel use of Open AI's GPT for conversational queries, which is not explicitly covered by this patent. However, the core idea of natural language-based recommendations may limit Tempo's ability to secure broad claims in this area.

Patent 2: Social Graph for Visualizing Compatibility

The patent reference is of the U.S. Patent 10,284,828: *System and method for visualizing relationships in a social network* [62]. Relevant claims consist of claim 1 (Independent) which describes a system for generating and displaying a social graph that visualizes relationships between users based on shared attributes (e.g., interests, preferences). Additionally, claim 3

(Dependent) specifies that the graph updates dynamically based on user activity or changes in preferences.

Tempo's Compatibility Web overlaps with claim 1, as it visualizes relationships between users based on shared musical compatibility (calculated using Spotify data). However, Tempo's approach differs in its daily-refresh feature, which randomly selects five friends for exploration, as well as its focus on promoting music discovery rather than static relationship metrics. Claim 3 partially aligns with Tempo but does not address the app's use of randomized selection as part of its design. Additionally, Tempo's Compatibility Web introduces unique design features that distinguish it from existing social graph systems. Its focus on dynamic refreshes and exploration then creates opportunities for patentable claims, though the foundational concept of visualizing relationships may already be covered.

Patent 3: User Interface for Music Analytics and Insights

The patent reference is of the U.S. Patent 9,646,362: *System and method for presenting personalized media analytics* [63]. Relevant claims are claim 1 (Independent) which describes a user interface that organizes and displays personalized music data, such as top genres, playlists, and listening history, over various time periods. Also, claim 4 (Dependent) includes additional functionality for comparing user statistics with those of others.

Tempo's Profile Tab aligns with Claim 1 in its presentation of user insights, such as top songs, playlists, and genres across customizable timeframes (e.g., 4 weeks, 6 months). However, Tempo does not incorporate direct user-to-user comparison of statistics as described in Claim 4. Instead, it integrates insights into the Compatibility Web for a more interactive exploration of social connections. While the concept of personalized analytics is well-covered, Tempo's unique integration of this data into its broader recommendation and social engagement system could form the basis for specific patentable claims.

Patentability of the Tempo Application

Based on the analysis of these patents, Tempo introduces innovative features that differentiate it from prior art, particularly in the use of natural language queries. Unlike traditional recommendation systems that rely on user behavior, Tempo uniquely uses OpenAI's GPT to

process conversational inputs. Additionally, Tempo consists of the unique Compatibility Web as Tempo's dynamic, daily-refreshed visualization for exploring social connections introduces a novel twist on social graph systems. Additionally, tempo combines user analytics with discovery and social engagement features, creating a cohesive ecosystem not explicitly addressed in the referenced patents.

However, several challenges could limit Tempo's patentability such as the reliance on external platforms. Many of Tempo's features depend on third-party platforms such as Spotify's API and OpenAI GPT, which are protected under their respective intellectual property. Tempo cannot patent the underlying technology provided by these platforms, such as Spotify's data retrieval algorithms or GPT's natural language processing capabilities. Any patent claims must then focus on Tempo's unique implementation or integration of these tools, such as the specific algorithm used to link GPT outputs with Spotify's music data. Additionally, since both Spotify's API and OpenAI GPT are publicly available, the novelty of Tempo's features must derive from how these tools are uniquely combined or enhanced. Without distinctive implementation details, the claims risk being deemed too broad or obvious. Additionally, licensing agreements for these platforms may restrict proprietary claims over how their tools are used.

Overall, while the Tempo app incorporates several innovative features, its reliance on pre-existing technologies narrows the scope of potential claims. Specific features, such as the integration of conversational queries with interactive social exploration, may present opportunities for intellectual property protection. However, the app must clearly differentiate its innovations from the underlying capabilities of Spotify and OpenAI's platforms to provide valid and enforceable patent claims.

In evaluating the patentability of Tempo's Bluetooth speaker, it is essential to assess its unique hardware features against relevant prior art to identify opportunities for intellectual property protection. Below is an analysis of three U.S. patents with claims relevant to the speaker's design and functionality, followed by an assessment of the hardware's patentability.

The first relevant patent is U.S. Patent 9,835,421, "Audio Device with Integrated Display for Enhanced Playback Control" [64]. This patent describes an audio device with an integrated display that presents playback information, such as track title and artist metadata, alongside user

input options for volume, track selection, and playback control. It further specifies that the display dynamically updates based on metadata received in real time from a connected device. Tempo's Bluetooth speaker aligns with these claims by incorporating an LCD display to show playback information and allow user interactions. However, Tempo differentiates itself by integrating with the Tempo app to display synchronized social and musical compatibility data. Unlike the prior art, Tempo's design enables real-time visualization of music analytics and social compatibility derived from Spotify data, creating a novel user experience. While the core concept of an integrated display for playback information overlaps with the prior art, the unique interaction between the app, speaker, and display presents opportunities for patentable claims.

The second relevant patent is U.S. Patent 10,125,389, "Modular Audio System with Network Synchronization" [65]. This patent covers a modular speaker system that streams audio and displays user-specific data retrieved from a connected service. Tempo's Bluetooth speaker shares similarities with this system, as it pairs with the app to stream music and display user-specific insights. However, Tempo diverges by integrating a dynamic, daily-refreshed Compatibility Web that promotes social exploration and advanced music analytics. Additionally, Tempo focuses on user interaction through tactile controls and real-time app integration, going beyond static playback and analytics. These distinctive features may allow Tempo to claim originality in the way its hardware enhances interactivity and social music discovery.

Finally, U.S. Patent 9,476,230, "Audio Device Featuring Optimized Enclosure for High-Fidelity Output" [66], describes an enclosure design incorporating passive radiators for enhanced sound quality and a dedicated amplifier with high-fidelity audio drivers. Tempo's speaker shares these characteristics, utilizing an MDF enclosure with passive radiators and a Class-D amplifier to deliver low-distortion, high-efficiency sound. However, Tempo's innovation lies in its integration of these acoustic features with its app-driven LCD interface, enabling a blend of high-quality audio and synchronized, dynamic visual content. While acoustic enhancements are widely covered in prior art, Tempo's combination of audio fidelity and real-time, app-connected visualization distinguishes it from existing designs and could support specific patent claims.

Overall, Tempo's Bluetooth speaker introduces several innovations that differentiate it from prior art. These include the dynamic integration of app data, such as the Compatibility Web and music

analytics, with the hardware's interactive LCD display. By emphasizing its unique hardware-software synergy, Tempo has the potential to secure patentable claims, particularly in areas where its features enhance interactivity, music discovery, and user engagement. Nonetheless, challenges may arise from the reliance on standard components like Bluetooth modules and microcontrollers, requiring patent applications to focus on the novel implementations of these components within the Tempo ecosystem.

8.   Timeline

Below, we present our initial Gantt chart, designed to visually outline our project timeline. This chart is organized by weeks and divided into sections for enhanced clarity and readability. It was created to ensure that tasks and milestones were easily trackable and comprehensible. The Gantt chart from our proposal has been included here alongside the final version, allowing for a direct comparison to highlight the differences and adjustments made over the course of the semester. Throughout the project, our timelines evolved due to unforeseen challenges, adjustments in scope, and new insights gained during the development process. For ease of presentation and to maintain legibility in the report, we have ensured that the charts are formatted appropriately.

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Sept. 9 - Sept. 13				
				M	T	W	R	F
1.0	General Market, Product, and Software Research							
1.1	Physical Design and Composition of Speaker		Bella, Thomas					
	Market Research for Similar Products							
	Final Selection for Firmware Features							
	General Physical Speaker Design Profile							
1.2	Embedded System Architecture		Thomas					
	Platform Selection for Microcontroller							
	Finalize Physical I/O							
1.3	Communication Protocols		Thomas					
	Selection for Internal Communication Protocol							
	Selection for External Communication Protocol							
1.4	Spotify API and OpenAI Integration in SwiftUI		Michelle, Naomi					
1.5	Firebase API Integration with SwiftUI		Joey					
1.6	Connection of SwiftUI and GitHub		Joey					
1.7	Best App Development Practices and Techniques		Naomi					
1.8	Audio Processing Algorithims		Bella, Naomi					
	Choice for Analog Processing Methods							
	Choice for Digital Processing Methods							
	Specifications for Recommendation and Mood Integraion							
1.9	~~Song-Blending Techniques~~		Bella					

Figure 39: General Market, Product and Software Research Sept. 9 - Sept. 13

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Sept. 9 - Sept. 13					Sept. 16 - 20					Sept. 23 - Sept. 27					Sept. 30 - Oct. 4					Oct. 7 - Oct. 11				
				M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
2	Hardware Planning and Design:																											
2.1	Determination of Design Parameters and Constraints		Group																									
	Design Specifications for Physical Speaker																											
	Design Specifications for Phone Application																											
2.2	Platform Part Selection and Design		Thomas, Bella																									
	Part Selection for Hardware Subcomponents																											
	Part Selection for Audio Components																											
	Documentation of Subcomponent Specifications																											
	Platform Selection for Firmware Development																											
	Power Delivery Selection																											
2.3	PCB Design: I/O & Power Supply		Thomas, Bella																									
	I/O PCB Design																											
	Power Delivery PCB Design																											
2.4	Enclosure Part Selection and Design		Thomas																									
2.5	Documentation of Parts, Schematics, and Drawings		Thomas, Bella																									
2.6	Hardware Development Test Plan Formation		Thomas, Bella																									

Figure 40: Hardware Planning and Design Sept. 9 - Oct 11

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Sept. 9 - Sept. 13					Sept. 16 - 20					Sept. 23 - Sept. 27				
				M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
4	Software Planning and Design																	
4.1	Determination of Design Parameters and Constraints		Group															
4.2	Design of the App UI		Group															
4.3	Design UI and Backend Relationship		Michelle															
4.4	Design Song Recommendation Algorithim with Spotify and OpenAI API		Joey															
4.5	~~Design Blending Algorithim~~		Bella, Michelle, Joey															

Figure 41: Software Planning and Design Sept. 9 - Sept

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Oct. 14 - Oct. 18				
				M	T	W	R	F
3	Hardware Development							
3.1	Platform Assembly	✔	Bella					
3.2	Microcontroller Setup	✔	Thomas, Bella					
3.3	Enclosure Assembly	✔	Thomas					
3.4	Implementation of Additional Hardware Features	✔	Thomas, Bella					

Figure 42: Hardware Development Oct 14 - Oct 18.

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Oct. 21 - Oct. 25					Oct. 28 - Nov. 1				
				M	T	W	R	F	M	T	W	R	F
3	Hardware Development												
3.1	Platform Assembly	✔	Bella										
3.2	Microcontroller Setup	✔	Thomas, Bella										
3.3	Enclosure Assembly	✔	Thomas										
3.4	Implementation of Additional Hardware Features	✔	Thomas, Bella										

Figure 43: Hardware Development Oct 21 - Nov 1

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Nov. 4 - Nov. 8				
				M	T	W	R	F
3	Hardware Development							
3.1	Platform Assembly	✔	Bella					
3.2	Microcontroller Setup	✔	Thomas, Bella					
3.3	Enclosure Assembly	✔	Thomas					
3.4	Implementation of Additional Hardware Features	✔	Thomas, Bella					

Figure 44: Hardware Development Nov 4 - Nov 8

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Sept. 9 - Sept. 13					Sept. 16 - 20					Sept. 23 - Sept. 27					Sept. 30 - Oct. 4				
				M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
7	Deliverables																						
7.1	Project Proposal		Group																				
7.2	Posters Due		Group																				
7.3	Poster Session		Group																				
7.4	Initial PCB Design and Submission		Group																				
7.5	Midterm Design Review		Group																				
7.6	Final Project Report		Group																				
7.7	Final Project Video		Group																				
7.8	Final Project Demo		Group																				

Figure 45. Deliverables Sept 9 - Oct 4

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Nov. 4 - Nov. 8				
				M	T	W	R	F
3	Hardware Development							
3.1	Platform Assembly	✔	Bella					
3.2	Microcontroller Setup	✔	Thomas, Bella					
3.3	Enclosure Assembly	✔	Thomas					
3.4	Implementation of Additional Hardware Features	✔	Thomas, Bella					

Figure 46. Hardware Development Nov 4 - Nov 8

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Oct. 21 - Oct. 25					Oct. 28 - Nov. 1				
				M	T	W	R	F	M	T	W	R	F
6	Integration and Testing												
6.1	App Functionality Across Devices and iOS Software Versions		Joey										
6.2	Spotify Functionality within App		Michelle										
6.3	Reccomendation Process		Group										
6.4	Ensure Physical Fit and Proper Component Interface		Bella										
6.5	Operation within General Electronic Parameters		Naomi										
6.6	Microcontroller Operations		Naomi, Thomas										
6.7	Enclosure Results: Resonance Damping		Naomi										
6.8	Seperate PCB Design Testing: Speaker Hardware		Michelle, Bella										
6.9	Connection of Device to Speaker		Group										
6.10	Playback of Audio through App to Speaker		Group										

Figure 47. Integration and Testing Oct 21- Nov 1

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE
5	Software Development		
5.1	Creation of Tempo SwiftUI App	✔	Joey
5.2	Integrate App with GitHub	✔	Naomi
5.3	Integrate App with Firebase API	✔	Michelle
5.4	Integrate App with Spotify API	✔	Naomi
5.5	Build Home/Login View	✔	Joey
5.6	Authenticate User through Spotify API	✔	Joey
5.7	Retrieve and Organize User Spotify Data using Spotify API	✔	Michelle
5.8	Build Song Queue Main View	✔	Joey
5.9	Integrate App with OpenAI API	✔	Naomi
5.10	Code Song Recommendation Algorithm with Spotify and OpenAI API	✔	Joey
5.11	Integrate App with Python Packages (for blending algorithm)	✔	Michelle
5.12	Code Blending Algorithm	✔	Michelle
5.13	Combine Song Recommendation and Blending Algorithm	✔	Naomi
5.14	Display Recommended Songs in Song Queue Main View	✔	Joey
5.15	Build Control Playback View	✔	Michelle
5.16	Connect Control Playback View with Spotify API	✔	Naomi
5.17	Build User Profile View	✔	Michelle
5.18	Connect User Profile Data to Profile View Using Spotify API	✔	Joey
5.19	Build Settings View	✔	Naomi
5.20	Connect Logout with Spotify API + Other General Settings to Settings View	✔	Michelle
5.21	Design Data Model for Compatibility Metrics	✔	Michelle , Naomi
5.22	Develop Compatibility Calculation Algorithm	✔	Joey, Naomi
5.23	Integrate Compatibility Algorithm with Spotify API	✔	Michelle, Naomi, Joey
5.24	Implement Compatibility Web Visualization	✔	Joey, Michelle

Figure 48. Software Development Sept 30 - Nov 15

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE
6	Integration and Testing		
6.1	App Functionality Across Devices and iOS Software Versions		Joey
6.2	Spotify Functionality within App		Michelle
6.3	Reccomendation Process		Group
6.4	Ensure Physical Fit and Proper Component Interface		Bella
6.5	Operation within General Electronic Parameters		Naomi
6.6	Microcontroller Operations		Naomi, Thomas
6.7	Enclosure Results: Resonance Damping		Naomi
6.8	Seperate PCB Design Testing: Speaker Hardware		Michelle, Bella
6.9	Connection of Device to Speaker		Group
6.10	Playback of Audio through App to Speaker		Group

Figure 49. Integration and Testing Nov 4 - Nov 29

The final Gantt chart reflects the adjustments and updates made to our project timeline throughout the semester. It highlights shifts in task durations, added milestones, and changes in project priorities that arose as the work progressed. These modifications were necessary to accommodate unexpected challenges and ensure the successful completion of the project. The chart is organized to clearly display the final distribution of tasks and their completion timelines, providing a comprehensive overview of how our project evolved from the initial plan to its conclusion. It is displayed below.

TASK NUMBER	TASK TITLE	COMPLETE	ASSIGNEE	Sept. 9 - Sept. 13				
				M	T	W	R	F
1.0	General Market, Product, and Software Research							
1.1	Physical Design and Composition of Speaker		Bella, Thomas					
	Market Research for Similar Products							
	Final Selection for Firmware Features							
	General Physical Speaker Design Profile							
1.2	Embedded System Architecture		Thomas					
	Platform Selection for Microcontroller							
	Finalize Physical I/O							
1.3	Communication Protocols		Thomas					
	Selection for Internal Communication Protocol							
	Selection for External Communication Protocol							
1.4	Spotify API and OpenAI Integration in SwiftUI		Michelle, Naomi					
1.5	Firebase API Integration with SwiftUI		Joey					
1.6	Connection of SwiftUI and GitHub		Joey					
1.7	Best App Development Practices and Techniques		Naomi					
1.8	Audio Processing Algorithims							
	Choice for Analog Processing Methods		Bella, Naomi					
	Choice for Digital Processing Methods		Bella, Naomi					
	Specifications for Recommendation and Mood Integraion		Bella, Naomi					
1.9	~~Song Blending Techniques~~		Bella					

Figure 50: General Market, Product and Software Research

TASK NUMBER	TASK TITLE	COMPLETE	ASSIGNEE	Sept. 9 - Sept. 13					Sept. 16 - 20					Sept. 23 - Sept. 27				
				M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
2	**Hardware Planning and Design:**																	
2.1	**Determination of Design Parameters and Constraints**		Group															
	Design Specifications for Physical Speaker																	
	Design Specifications for Phone Application																	
2.2	**Platform Part Selection and Design**		Thomas, Bella															
	Part Selection for Hardware Subcomponents																	
	Part Selection for Audio Components																	
	Documentation of Subcomponent Specifications																	
	Platform Selection for Firmware Development																	
	Power Delivery Selection																	
2.3	**PCB Design: I/O & Power Supply**		Thomas, Bella															
	I/O PCB Design																	
	Power Delivery PCB Design																	
2.4	**Enclosure Part Selection and Design**		Thomas															
	Internal Volume Design Report		Thomas															
	Construction and Attachment Methods		Thomas															
	Basic 3D Model w/ Alternatives		Thomas															
	Advanced Part Modeling		Thomas															
	Advanced 3D Model		Thomas															
2.5	**Documentation of Parts, Schematics, and Drawings**		Bella, Thomas															
	Compile and Document BOM		Bella															
	Finalize Audio, Enclosure, and Control Specs		Bella, Thomas															
	Finalize Audio Transmission Flow and Document Necessary Protocols		Bella, Thomas															
	Document Final Subcircuit Design: Power, Audio, Controls		Bella															
	Document Connections between Subcircuits		Bella															
2.6	**Hardware Development Test Plan Formation**		Thomas, Bella															
	Physical Component Assembly: Power, Audio, Control																	
	Programming Functionality: BT to Amp/Microcontroller, DAC, SPI/UART																	
	Formation of Interactive Display and Screens																	

Figure 51: Hardware Planning and Design Sept 9 - Sept 27

TASK NUMBER	TASK TITLE	COMPLETE	ASSIGNEE	Sept. 9 - Sept. 13					Sept. 16 - 20					Sept. 23 - Sept. 27				
				M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
4	**Software Planning and Design**																	
4.1	**Determination of Design Parameters and Constraints**		Group															
4.2	**Design of the App UI**		Group															
4.3	**Design UI and Backend Relationship**		Michelle															
4.4	**Design Song Recommendation Algorithim with Spotify and OpenAI API**		Joey															

Figure 52: Software Planning and Design Sept 9 - Sept 27

TASK NUMBER	TASK TITLE	COMPLETE	ASSIGNEE	Sept. 23 - Sept. 27					Sept. 30 - Oct. 4					Oct. 7 - Oct. 11				
				M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
	Software Development and Testing																	
.1	**Creation of Tempo SwiftUI App**		Joey	█														
.2	**Integrate App with GitHub**		Joey		█													
.3	**Integrate App with Firebase API**		Joey			█												
.4	**Integrate App with Spotify API**						█											
	Authenticate User through Spotify API		Joey											█	█	█	█	█
	Retrieve and Organize User Spotify Data using Spotify and Firebase API		Joey															
	Design Data Model for Storing Music Information		Joey															
.5	**Build The User Authentication Views**																	
	Build AuthView with Firebase/Spotify Integration		Joey						█									
	Build AuthQOneView with Firebase/Spotify Integration		Joey							█	█							
	Build AuthQTwoView with Firebase/Spotify Integration		Joey								█	█						
	Build AuthQThreeView with Firebase/Spotify Integration		Joey															
	~~Build AuthQFourView with Firebase/Spotify Integration~~		Joey															
	~~Build AuthQFiveView with Firebase/Spotify Integration~~		Joey															
	~~Build AuthQSixView with Firebase/Spotify Integration~~		Joey															
	~~Build AuthQSevenView with Firebase/Spotify Integration~~		Joey															
	Design Data Model for Storing User Information		Joey															
.6	**Build The Listen Tab Views**																	
	Build ChatView with Firebase/Spotify Integration		Joey/Naomi															
	Build ListenView with Firebase/Spotify Integration		Joey/Naomi															
	Build HistoryView with Firebase/Spotify Integration		Joey															
.7	**Build The Search Tab Views**																	
	Build ExploreView with Firebase/Spotify Integration		Joey/Michelle															
	Build OtherProfileView with Firebase/Spotify Integration		Joey															
	Build FollowersView with Firebase/Spotify Integration		Joey															
	Build FollowingView with Firebase/Spotify Integration		Joey															
	Build PlaylistView with Firebase/Spotify Integration		Joey															
.8	**Build The Profile Tab Views**																	
	Build SelfProfileView with Firebase/Spotify Integration		Joey															
	Build EditSelfProfileView with Firebase/Spotify Integration		Joey															
	Build FollowRequestView with Firebase/Spotify Integration		Joey															
	Build SettingsView with Firebase/Spotify Integration		Joey															
.9	**Song Recommendation Algorithm**																	
	Integrate App with OpenAI API		Naomi															
	Code the pipeline from ChatGPT to a Song Queue		Naomi															
	Design Data Model for Song Recommendations		Naomi															
.10	**User Compatibility Algorithm**																	
	Develop Compatibility Calculation Algorithm		Michelle															
	Integrate Compatibility Algorithm with Spotify API		Michelle															
	Design Data Model for Compatibility Metrics		Michelle															

Figure 53: Software Development and Testing Sept 23 - Oct 11

TASK NUMBER	TASK TITLE	COMPLETE	ASSIGNEE	Oct. 14 - Oct. 18					Oct. 21 - Oct. 25					Oct. 28 - Nov			
				M	T	W	R	F	M	T	W	R	F	M	T	W	R
5	**Software Development and Testing**																
5.1	**Creation of Tempo SwiftUI App**		Joey														
5.2	**Integrate App with GitHub**		Joey														
5.3	**Integrate App with Firebase API**		Joey														
5.4	**Integrate App with Spotify API**																
	Authenticate User through Spotify API		Joey														
	Retrieve and Organize User Spotify Data using Spotify and Firebase API		Joey		█	█	█	█									
	Design Data Model for Storing Music Information		Joey		█	█	█	█									
5.5	**Build The User Authentication Views**																
	Build AuthView with Firebase/Spotify Integration		Joey														
	Build AuthQOneView with Firebase/Spotify Integration		Joey														
	Build AuthQTwoView with Firebase/Spotify Integration		Joey														
	Build AuthQThreeView with Firebase/Spotify Integration		Joey											█	█		
	Build AuthQFourView with Firebase/Spotify Integration		Joey														
	Build AuthQFiveView with Firebase/Spotify Integration		Joey														
	Build AuthQSixView with Firebase/Spotify Integration		Joey														
	Build AuthQSevenView with Firebase/Spotify Integration		Joey														
	Design Data Model for Storing User Information		Joey						█	█	█	█	█				
5.6	**Build The Listen Tab Views**																
	Build ChatView with Firebase/Spotify Integration		Joey/Naomi														█
	Build ListenView with Firebase/Spotify Integration		Joey/Naomi														
	Build HistoryView with Firebase/Spotify Integration		Joey														
5.7	**Build The Search Tab Views**																
	Build ExploreView with Firebase/Spotify Integration		Joey/Michelle														
	Build OtherProfileView with Firebase/Spotify Integration		Joey														
	Build FollowersView with Firebase/Spotify Integration		Joey														
	Build FollowingView with Firebase/Spotify Integration		Joey														
	Build PlaylistView with Firebase/Spotify Integration		Joey														
5.8	**Build The Profile Tab Views**																
	Build SelfProfileView with Firebase/Spotify Integration		Joey														
	Build EditSelfProfileView with Firebase/Spotify Integration		Joey														
	Build FollowRequestView with Firebase/Spotify Integration		Joey														
	Build SettingsView with Firebase/Spotify Integration		Joey														
5.9	**Song Recommendation Algorithm**																
	Integrate App with OpenAI API		Naomi						█	█	█	█	█	█	█	█	█
	Code the pipeline from ChatGPT to a Song Queue		Naomi		█	█											
	Design Data Model for Song Recommendations		Naomi		█	█											
5.10	**User Compatibility Algorithm**																
	Develop Compatibility Calculation Algorithm		Michelle		█				█	█	█	█	█	█	█		
	Integrate Compatibility Algorithm with Spotify API		Michelle														
	Design Data Model for Compatibility Metrics		Michelle		█				█	█	█	█	█	█			

Figure 54: Software Development and Testing Oct 14 - Nov 1

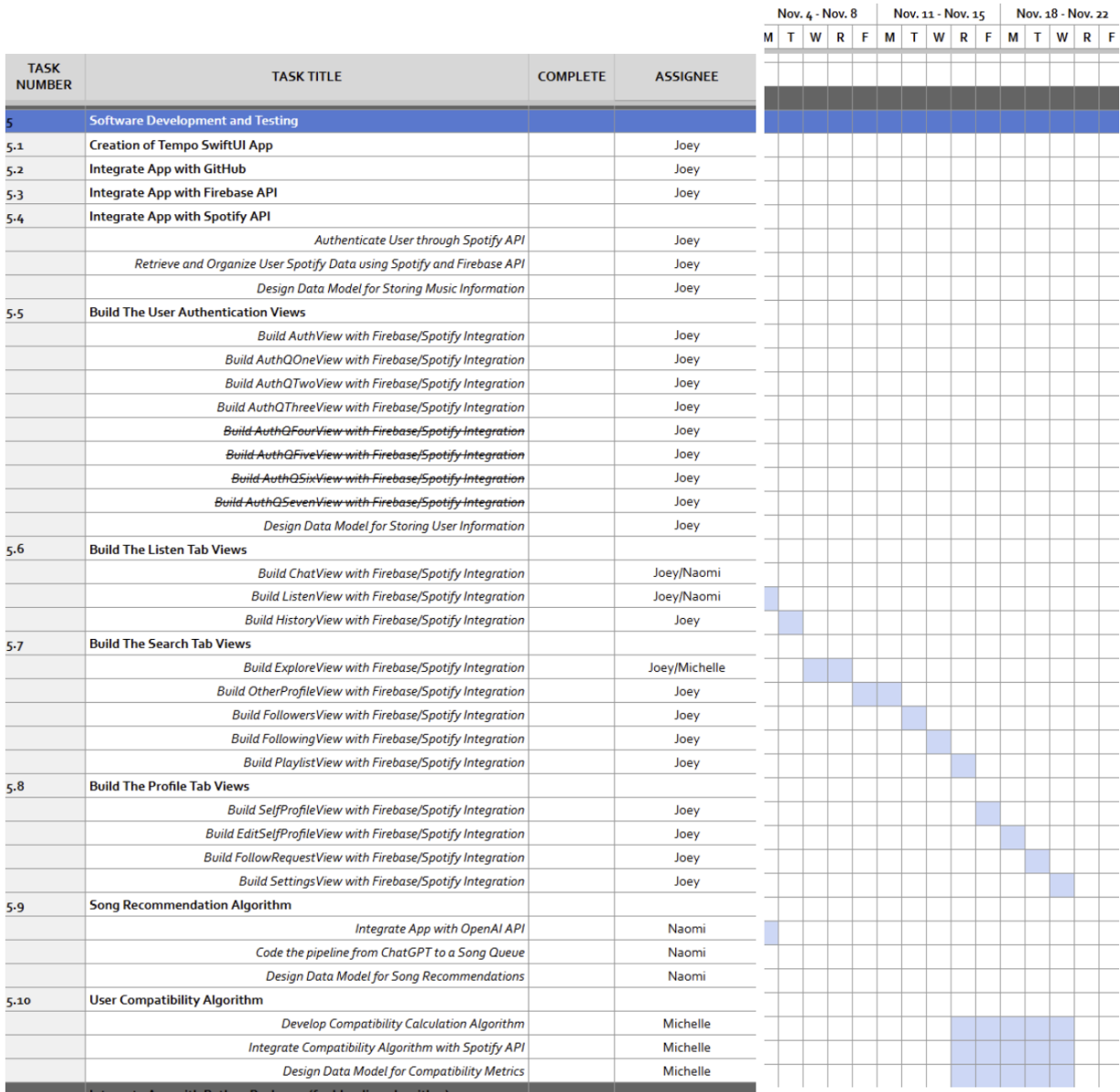| TASK NUMBER	TASK TITLE	COMPLETE	ASSIGNEE	Nov. 4 - Nov. 8					Nov. 11 - Nov. 15					Nov. 18 - Nov. 22				
				M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
5	**Software Development and Testing**																	
5.1	**Creation of Tempo SwiftUI App**		Joey															
5.2	**Integrate App with GitHub**		Joey															
5.3	**Integrate App with Firebase API**		Joey															
5.4	**Integrate App with Spotify API**																	
	Authenticate User through Spotify API		Joey															
	Retrieve and Organize User Spotify Data using Spotify and Firebase API		Joey															
	Design Data Model for Storing Music Information		Joey															
5.5	**Build The User Authentication Views**																	
	Build AuthView with Firebase/Spotify Integration		Joey															
	Build AuthQOneView with Firebase/Spotify Integration		Joey															
	Build AuthQTwoView with Firebase/Spotify Integration		Joey															
	Build AuthQThreeView with Firebase/Spotify Integration		Joey															
	~~*Build AuthQFourView with Firebase/Spotify Integration*~~		Joey															
	~~*Build AuthQFiveView with Firebase/Spotify Integration*~~		Joey															
	~~*Build AuthQSixView with Firebase/Spotify Integration*~~		Joey															
	~~*Build AuthQSevenView with Firebase/Spotify Integration*~~		Joey															
	Design Data Model for Storing User Information		Joey															
5.6	**Build The Listen Tab Views**																	
	Build ChatView with Firebase/Spotify Integration		Joey/Naomi															
	Build ListenView with Firebase/Spotify Integration		Joey/Naomi	█														
	Build HistoryView with Firebase/Spotify Integration		Joey	█														
5.7	**Build The Search Tab Views**																	
	Build ExploreView with Firebase/Spotify Integration		Joey/Michelle			█												
	Build OtherProfileView with Firebase/Spotify Integration		Joey				█	█										
	Build FollowersView with Firebase/Spotify Integration		Joey						█									
	Build FollowingView with Firebase/Spotify Integration		Joey							█								
	Build PlaylistView with Firebase/Spotify Integration		Joey								█							
5.8	**Build The Profile Tab Views**																	
	Build SelfProfileView with Firebase/Spotify Integration		Joey											█				
	Build EditSelfProfileView with Firebase/Spotify Integration		Joey												█			
	Build FollowRequestView with Firebase/Spotify Integration		Joey													█		
	Build SettingsView with Firebase/Spotify Integration		Joey														█	
5.9	**Song Recommendation Algorithm**																	
	Integrate App with OpenAI API		Naomi	█														
	Code the pipeline from ChatGPT to a Song Queue		Naomi															
	Design Data Model for Song Recommendations		Naomi															
5.10	**User Compatibility Algorithm**																	
	Develop Compatibility Calculation Algorithm		Michelle								█	█	█	█				
	Integrate Compatibility Algorithm with Spotify API		Michelle								█	█	█	█				
	Design Data Model for Compatibility Metrics		Michelle							█	█	█		█	█			

Figure 55. Software Development and Testing Nov 4 - Nov 22

TASK NUMBER	TASK TITLE	COMPLETE	ASSIGNEE	Sept. 23 - Sept. 27				
				M	T	W	R	F
6	Integration and Testing							
6.1	App Functionality Across Devices and iOS Software Versions		Group					
6.2	Spotify Functionality within App		Group					
6.3	Recomendation Process		Group					
6.4	Compatibility Comparison		Group					
6.5	Ensure Physical Fit and Proper Component Interface		Bella					
6.6	Operation within General Electronic Parameters		Naomi					
6.7	Microcontroller Operations		Naomi, Thomas					
6.8	LCD Display and Functionality		Bella, Naomi, Thomas					
6.9	Enclosure Results: Resonance Damping		Naomi					
6.10	Seperate PCB Design Testing: Speaker Hardware		Michelle, Bella					
6.11	Connection of Device to Speaker		Group					
6.12	Playback of Audio through App to Speaker		Group					
6.13	LCD Display with Audio Playback Capabilities		Group					

Figure 56. Integration and Testing Sep 23 - 27

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Nov. 4 - Nov. 8					Nov. 11 - Nov. 15					Nov. 18 - Nov. 22					Nov. 25 - Nov. 29				
				M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
6	Integration and Testing																						
6.1	App Functionality Across Devices and iOS Software Versions		Joey																				
6.2	Spotify Functionality within App		Michelle																				
6.3	Reccomendation Process		Group																				
6.4	Ensure Physical Fit and Proper Component Interface		Bella																				
6.5	Operation within General Electronic Parameters		Naomi																				
6.6	Microcontroller Operations		Naomi, Thomas																				
6.7	Enclosure Results: Resonance Damping		Naomi																				
6.8	Seperate PCB Design Testing: Speaker Hardware		Michelle, Bella																				
6.9	Connection of Device to Speaker		Group																				
6.10	Playback of Audio through App to Speaker		Group																				

Figure 57. Integration and Testing Nov 4 - 29

TASK NUMBER	TASK TITLE	TEST	ASSIGNEE	Dec. 2 - Dec. 6				
				M	T	W	R	F
6	Integration and Testing							
6.1	App Functionality Across Devices and iOS Software Versions		Joey					
6.2	Spotify Functionality within App		Michelle					
6.3	Reccomendation Process		Group					
6.4	Ensure Physical Fit and Proper Component Interface		Bella					
6.5	Operation within General Electronic Parameters		Naomi					
6.6	Microcontroller Operations		Naomi, Thomas					
6.7	Enclosure Results: Resonance Damping		Naomi					
6.8	Seperate PCB Design Testing: Speaker Hardware		Michelle, Bella					
6.9	Connection of Device to Speaker		Group					
6.10	Playback of Audio through App to Speaker		Group					

Figure 58. Integration and Testing Dec 2 - 6

The updated Gantt chart highlights several key differences from our original plan, reflecting the adjustments made to accommodate unforeseen challenges during the project. One notable change was the extended timeline for developing the compatibility web, which took longer than anticipated. Originally scheduled for earlier completion, integration and testing also continued until the final days of the project, culminating on December 6.

These delays were primarily due to issues such as retrieving tokens from Spotify, which introduced unexpected complications, and the user models, which proved to be finicky and required additional troubleshooting. Despite these obstacles, the extended timelines allowed us to refine these components and ensure their functionality. The updated chart captures these shifts, demonstrating our flexibility and commitment to delivering an effective final product.

The timeline for the hardware development of our project was significantly impacted by several key factors, including delays caused by two separate PCB orders, communication challenges within the team, and an uneven distribution of responsibilities among the group members. Initially, our project was hindered by the need to reorder the PCB designs twice, due to issues discovered in the first version that required revision. These revisions took longer than

anticipated, causing delays in testing and assembly. Additionally, the lack of consistent communication between group members exacerbated the situation. As a result, there was often a lack of clarity on the status of hardware components, design iterations, and assembly schedules. This misalignment contributed to confusion and further delays in completing the hardware design on time.

Another challenge we faced was the disproportionate workload placed on a single group member for the hardware-related tasks. While the group collectively contributed to the software and conceptual aspects, one member took on the majority of the responsibility for the selection and design of hardware components, such as the power distribution system, audio amplifier, and microcontroller. This created a bottleneck in the design process, as the rest of the team was not fully integrated into the decision-making process, which could have led to more efficient problem-solving and more timely progress. Furthermore, the decision to change the microcontroller in the later stages of the project added complexity to the design, resulting in additional time needed for integration and testing. These hardware-related delays, compounded by communication gaps and uneven workload distribution, significantly impacted the project's overall timeline, and we had to adjust expectations and prioritize critical tasks to meet the final goals.

9. Final Results

9.1. Functionality

Bluetooth functionality across the integrated system was thoroughly tested, serving as the ultimate proof of successful software and hardware integration. To initiate the system, the AC-DC power converter was plugged into a standard wall outlet, delivering power to the connected components. The rocker switch wired to the PCB was flipped on, which sequentially powered the amplifier, Bluetooth module, and supporting circuitry. The Bluetooth module became discoverable, and an iPhone was used to pair with the device, identified by its designated name in the phone's Bluetooth settings.

Using the app interface, a text prompt was entered into the search bar. The system successfully processed the input, generating a song recommendation, which was streamed wirelessly to the speaker. Upon playback, the system demonstrated seamless functionality, with no delay in initiating audio output. The sound emitted was distortion-free, meeting expectations for clarity and quality. At approximately 25% of the amplifier's maximum volume, the system consumed around 5.2W of power, excluding the display's operational load.

During testing, the sound quality was evaluated both for individual components and the fully assembled enclosure. The audio fidelity remained consistent and robust, reaffirming that the mechanical assembly and electronic design worked harmoniously. The system not only met but exceeded initial performance benchmarks, delivering an integrated and user-friendly experience.

We are in the process of configuring the display, given the previously mentioned troubles with the STM microcontroller. The Raspberry Pi system has been written to an SD card, and inputted into the board. The display has arrived and we are in the process of testing that functionality now. We aim to have everything arranged on-device come December 9th, for demonstration purposes.

Photos depicting the current status of our product can be found in the Appendix, alongside photos depicting the process of testing.

9.2.   Discussion of Set Criteria

The success of our capstone project was assessed based on the performance and integration of the iOS application with Bluetooth speaker equipped with a touchscreen LCD display. The predefined rubric can be found in the Appendix, but the grade calculation is as follows:

- A (90-100%): 90-100 points
- B (80-89%): 80-89 points
- C (70-79%): 70-79 points
- D (60-69%): 60-69 points
- F (Below 60%): 0-59 points

For the iOS app, successful performance includes delivering stable functionality, reliable music recommendations, and user profile management with minimal bugs, providing a smooth experience. In analysis, the Tempo app delivered a good, functional app. Any latency can be attributed to storing immense amounts of data to firebase where for example, the profile tab needs a minimal amount of time to process all the user information; however, the main goals of the app in terms of implementing a recommendation queue, compatibility web, and a breakdown of profile statistics were operational to our liking and do not negatively impact user experience. We are content with the fact that playback works for long periods of time and there are minimal glitches in the overall app. Additionally, our app is aesthetically pleasing, with calming shades of purple and an easy, organized tab system where users can navigate from tab to tab without questioning the various tabs' purpose or overloading the user with too much information at the same time.

The Bluetooth speaker is then expected to provide a clear audio output and maintain stable connectivity, with a touchscreen display that accurately shows basic playback information and settings in real-time. Additionally, we expect the speaker to align with app changes for a cohesive experience. For analysis, the Bluetooth speaker produced a great output where sound has minimal distortion, nor are there any connectivity errors. We were able to connect the app to the Bluetooth easily on the first try and found no connection errors throughout our testing

process. The enclosure also looks visually appealing and is appropriate for all electronics to fit inside. The display is functional, although not mounted, but we are pleased with our process in overcoming issues with microcontroller programming and hitting set objectives.

Compliance with established standards is also necessary for the project's success. This includes adherence to regulations for Bluetooth devices and iOS app development guidelines to provide operational compliance within regulatory frameworks. Overall, our team did an excellent job in storing and keeping track of documentation, and compiling to various standards, thus, the overall product is safe to use. To summarize, we are content with the final product and would rate ourselves in the 90-100 points range in developing and implementing a functional product.

10.   Engineering Insights

10.1.   Physical Speaker

Throughout the design and assembly of the physical speaker, we gained valuable technical and personal insights. From a technical perspective, we learned the importance of material selection in terms of acoustics and structural integrity. Our decision to use PLA for the base plates was driven by the need for a lightweight, durable solution that could support various components while also ensuring effective thermal management. This experience deepened our understanding of how material properties, such as weight, strength, and heat resistance, affect not only the ease of assembly but also the overall performance of the product. Additionally, we recognized the need to design custom cutouts in the base plates for proper airflow and heat dissipation, an aspect often overlooked in early designs but crucial to preventing component overheating.

From a power management standpoint, we realized the complexity of ensuring that multiple components receive stable and correct power supplies without causing interference. The careful routing of power and signal lines on the PCB taught us how electromagnetic interference (EMI) can degrade audio quality and the importance of isolating these paths to reduce noise. The integration of an external DC-DC buck converter to step down the voltage with minimal energy loss highlighted how efficient power regulation can directly impact system performance and thermal efficiency. We also learned how critical component selection—such as using ferrite beads for noise suppression and ensuring proper trace widths for current capacity—can directly affect the longevity and reliability of the system.

Another technical takeaway was how to optimize the internal layout of components within a confined space. Placing the speaker drivers at precise angles and within specific areas of the enclosure ensured even sound dispersion and improved overall acoustics. This reinforced our understanding of sound wave propagation and how enclosure design influences audio quality.

On a personal level, this project taught us the value of iteration and testing in the design process. Early on, we encountered challenges with component placement and heat management, but through trial and error, we refined our approach and gained a deeper appreciation for the

practical aspects of hardware design. Additionally, working through power and signal integrity issues taught us the importance of meticulous attention to detail—how small design decisions, like component placement and routing, can have significant impacts on performance. We also developed greater patience and perseverance, as many of the issues encountered were solved through persistent experimentation and problem-solving.

Throughout this project, we learned firsthand how critical communication and planning are to the success of any engineering endeavor. Early on, we encountered significant challenges in meeting deadlines and maintaining effective communication within the team. At times, we struggled to stay in touch regularly, which led to delays in decision-making and created confusion about the project's direction. Our lack of clarity about the final product compounded these issues, as team members often had different expectations and visions for the outcome. Disagreements over design choices and priorities further slowed progress, and we realized that not having a unified understanding of the end goal resulted in inefficient use of time and resources. Some group members had to take on significantly more than initially expected, as others refrained from contacting or adding to the project, which caused the project as a whole to fall behind. From this experience, we learned the importance of establishing clear goals, regular check-ins, and transparent communication from the outset of a project. Proper planning, with well-defined milestones and responsibilities, would have helped mitigate many of the challenges we faced. This experience reinforced the idea that success in team-based projects relies not only on technical knowledge but also on strong communication, effective collaboration, and a shared vision.

Overall, this project deepened both our technical understanding and personal growth, reinforcing the importance of careful design, testing, and continuous improvement in building a successful product.

10.2.   iOS Application

On the software side, there were various lessons we learned both technically and generally about the engineering process. Technically, we refined our skills in tools like Xcode and GitHub, where the evolution of development practices required us to stay updated on how to build the software. Xcode, as the primary tool for developing iOS applications, presented unique challenges. The

continuously changing requirements of Apple's iOS ecosystem meant we needed to see that our app was compatible with the latest software versions while still functioning for users on older versions. Although we had experience with XCode in the past, it required us to do thorough research before we began development and a continuous process of learning and adapting as we integrated new features. For instance, implementing features that relied on iOS's latest APIs taught us how to handle version-specific fallbacks, making sure no user was left behind due to compatibility issues.

GitHub, on the other hand, added the complexities of team collaboration in software development. Unlike simple collaboration tools like Google Docs, GitHub requires careful planning to avoid overwriting each other's work. We divided the project into multiple branches—Main, Authentication, and Playback—so each team member could work independently while contributing to the same codebase. This structure forced us to think critically about feature dependencies and integration points. Additionally, merge conflicts became a practical lesson in how to resolve overlapping changes and refine our code review process to improve collaboration. It forced us to meticulously plan each step of the code before writing anything which challenged our ability to think through problems and structures before actually facing them. This is different from individual projects where it is easier to explore and test different paths without needing to constantly consult with group members. However, it highlighted the ongoing theme that communication is key to group success. Luckily, we didn't have any serious issues with integrating each group member's code bases since we did a good job communicating from the start, but it was an aspect of the project that was constantly in the back of our minds.

Beyond these tools, integrating Firebase was another significant learning experience. Firebase served as the backbone for user authentication and real-time data synchronization. Its flexibility allowed us to focus on implementing features rather than worrying about backend infrastructure. However, its power came with a learning curve. We needed to understand Firebase's security rules deeply to protect user data, especially as we began implementing features like user profiles and data sharing with Spotify. Additionally, managing real-time database updates taught us how to optimize queries to minimize latency and keep the app responsive, even as the user base grew. However, creating an advanced data structure that can efficiently handle both Firebase updates

and integration with the Spotify API data was beyond the scope of our project. We did not fully understand the complexities of handling data in a way that did not take away from the user experience, such as long loading times. Although we were able to integrate all major aspects into our project, there is still room to improve in terms of improving latency for example.

The Spotify API introduced the most challenges and learning opportunities. Learning OAuth 2.0 authentication, a type of authentication pipeline, helped us securely access user accounts, but it also required us to design a seamless login experience that balanced security with usability. Spotify requires security to be properly handled in order to access user's data through their API. There were various steps, such as exchanging and constantly refreshing tokens, and securely storing these tokens on servers. Additionally, the Spotify documentation for these services became misleading which caused us to circle back at times for functionality. For example, we initially only integrated the SpotifyWeb API since the research we conducted made it seem like all the functionality required, such as playback and data querying was available through that API. However, once we integrated playback, we learned we also had to use the SpotifyiOS API. This pivot required us to completely redo the authentication pipeline since we now had to log in to a user through both APIs simultaneously. Managing how and when to use each API, along with handling tokens from both APIs, was then the largest learning curve of the project for the software. A security aspect that also came with blending Firebase and Spotify is figuring out a way to allow a user logged into Spotify to access their data in Firebase. Since we don't want any user to access anyone's data in Firebase, we had to create a pipeline that ensured users only had access to data they were allowed to see. This required us to take the Spotify token, which was obtained once a user logged into Spotify, to then securely log them into Firebase so that they are authenticated to access certain information. We were not anticipating this integration, however, it became required to provide the proper handling of user data.

Working with the OpenAI API fortunately was more straightforward as the documentation of OpenAI was accommodating towards developers.. Even though there was a slight learning curve with seeing that we used the correct syntax, it did not pose any extreme limitations as compared to the Spotify API. Integrating AI-generated content overall required learning on how to craft precise prompts and understanding how different parameters influenced the quality of the API's responses. Due to the nature of natural language processing, a single query can yield a wide

range of responses. Thus, we had to test various prompts in order to achieve the best output from OpenAI's models. This process can be constantly improved, however, it took many iterations and refinements to curate a prompt that provided the best music recommendations based on a user's request.

Although the technical challenges were crucial to reflect on, the lessons learned about the engineering process—particularly in time and resource management, communication, and maintaining team morale—are invaluable for improving efficiency, collaboration, and seeing the successful execution of future projects. One of the biggest lessons we took away was time management. The initial thought process for approaching a long-term project was to break the code into sections and perfect each section before moving on to the next one. This approach seemed logical, but it is not necessarily the best way to handle a long-term project. This is because we wasted time at the beginning of the semester perfecting the authentication flow, which then caused us to rush through the rest of the application. We also focused on animations, aesthetics, and functionality all at once since we tried to hit each mark the first time around. Ironically, we ended up having to redo the authentication flow later on, due to the SpotifyWeb and iOS API pivot, where the perfection initially made was outdated since it was redone anyway.

Since software can be iterated easily compared to hardware, it is necessary to break up the software into drafts or waves. The first wave should be high-level without any code such as breaking down all the components of the app and explicitly researching exactly what is required of each component as if you are implementing it. We believe this should be done without coding because it allows you to inexpensively interchange components as research is conducted. This is an important first wave since it may have prevented our pivot of adding the SpotifyiOS API more than halfway through the semester. If we conceptually broke down all the requirements of the app, then we maybe would have caught that we needed to use the SpotifyWeb API for the data querying and the SpotifyiOS API for playback. Although we did comprehensive research beforehand to know what we needed and how it would come together; however, this stage I'm describing takes it a step further. It should be done comprehensively enough that when you do go to code, there will not be any big surprises or research needed to implement the components. Each portion of the app should then have exact inputs and outputs and the exact mapping for

how the inputs will connect to the outputs, along with how the various components will all connect together.

Next, the second wave should be taking this outline and purely implementing the functionality aspects of it. No aesthetics and no animations. Simply get the functionality working for the entire app. Then, circle back to the beginning and add robustness, efficiency, and security in the same manner by iteratively going through each component of the app and refining the first wave functionality. Then, it is now acceptable to take a third pass through the app to add animations and aesthetics. This new approach allows for improvements to be made in small increments so that large-scale functionality, like authentication, does not need to be redone after spending much of the semester perfecting it before any other parts of the app were even started. Additionally, it is the most optimal way to see that all the functionality is completed by the end of the long-term project. There was functionality we initially set to achieve that we did not end up completing. For example, implementing more, accurate user data in the explore view and the non-critical views such as queue history, settings, followers/following, and follow requests did not happen. We do not believe this is because we bit off more than we can chew in the timeframe of this project. If we took the iterative approach described above, then all functionality would have been completed. There is a part of us that is grateful we made that mistake because it is such an important lesson to learn how to properly organize a long-term project. I would recommend any future capstone groups take the iterative approach instead of the perfection at each-step approach.

Another big lesson we learned was through communication. We split up the work in a way that makes sense, however, we were initially vague about the exact requirements we expected from each member. For example, for authentication, we split it up where some members worked on the UI while others worked on the backend functionality. Splitting the work this way was not the issue, but the requirements voiced for each member were the issue. We simply stated that we should get that portion done and then integrate it together the next time we meet; however, that posed issues when we combined the two functionalities together because the two sets of code were not necessarily compatible with one another where a lot of time was spent on integration of all the code. We should have stated explicitly the inputs and outputs for the front and back end to see that once we complete them, the two codebases are integrated together seamlessly. This

caused delays initially because we had to spend extra weeks fixing the code so that the front end worked with the backend. It is overall important to take time as a group to outline what both components should function as and how they will eventually fit together. Then, each member will go off and execute exactly how we outlined it. This will save us a lot of time and effort in future long-term group projects. Once we got past the authentication flow, the rest of the app integration went by smoothly which allowed us to get the project completed with the core functionality intact. However, this is an extremely important lesson and emphasizes the importance of direct, clear communication between group members.

Another important aspect we learned is accounting for imperfection. Particularly with planning out our Gantt chart, it is not good enough to simply account for how long we believe each task will take. Although this is a good baseline to start with, it is important to understand that that time frame is based solely on what we believe needs to get done. However, real-world execution requires all aspects to be completed, including portions that we did not initially anticipate needed to get done. No one is perfect and no one, especially at our level of experience, can foresee all the pitfalls and setbacks. Thus, when designing the breakdown of the semester, be as comprehensive as possible, but also account for unforeseen events. If you end up blocking too much time for a part of the project, then the worst-case scenario is more time to work on other components. However, if you do not block off extra time for mistakes, then there are parts of the project that will either be left out or hastily added at the end of the semester. The Spotify integration with the web and iOS API is a perfect example of this.

Regardless of the challenges we faced, we constantly kept the group morale high as we saw it as a crucial aspect of our success. Even if something did not come out the way we imagined, or there were issues vocalized, we made sure to keep the perspective that it was us versus the problem instead of us against each other. This small perspective switch is important because a lot of time can be wasted bickering with group members on a particular issue, instead of actually solving the issue. We do not believe the actual problem is ever the reason a group fails, nor is it realistic that a group will never have an issue. It is how the group handles the problems when they do come up that decides the success of the group.

In the end, these lessons extended beyond individual tools or technologies. They shaped how we approached problems, planned our workflows, and collaborated as a team. This project gave us practical experience in navigating the complexities of modern software development, from technical execution to team dynamics, leaving us better prepared for future challenges.

11.   Future Work

11.1.   Physical Speaker

As we approach demo day, our current focus is on finalizing the physical assembly of the speaker and completing key integration tasks. We are in the final stages of mounting all hardware components, including securely placing the power distribution PCB, amplifier, Bluetooth board, and Raspberry Pi into the speaker enclosure. The enclosure itself still requires painting to achieve a professional and polished appearance, which is one of the last aesthetic steps before final assembly. Additionally, we are preparing to mount the 7-inch display, ensuring it is properly connected to the Raspberry Pi and functioning as intended. By demo day, we aim to have all hardware fully integrated, the enclosure painted, and the display mounted and operational, ensuring a smooth, visually appealing, and functional presentation of the product.

In future iterations of our prototype design, there are several enhancements and improvements we hope to explore. These include refining the enclosure design to optimize acoustic performance and further improve the overall aesthetic. We plan to experiment with additional Bluetooth range improvements and fine-tuning of the audio signal processing for even better sound quality. Additionally, we aim to streamline the integration between the hardware components and software, improving efficiency and reducing potential latency or errors. Future work will also involve testing with a wider range of devices to ensure compatibility and performance under different conditions.

Further, in future iterations, we aim to expand the display's capabilities, with a focus on achieving full on-device playback functionality via Bluetooth. Our goal is to enable users to not only stream music directly from the app but also interact with the device through the touchscreen for a more intuitive experience. We envision a setup where users can control playback, browse through songs, and input commands directly on the screen, reducing the reliance on external devices. Further, we plan to integrate more interactive features, such as feedback for user input or on-screen notifications, enhancing the overall usability and experience.

Ultimately, our goal is to elevate the product's reliability, performance, and user experience, building on the foundation established in this prototype.

11.2. iOS Application

Although we completed the core functionality of the application, there are many improvements to be made to our design. Additionally, there were difficulties not foreseen at the beginning of the project that we had to handle as the project developed. Many of these pitfalls were stated in the previous section because they led to the majority of our engineering insights from the project. Essentially all aspects of the app could be improved in the future. Starting with the functionality, we would implement all views we initially set out to complete. This includes the ListenHistoryView, OtherProfileView, FollowRequestView, and SettingsView. Additionally, we would finish the ExploreView so that we can search users and filter users by genre and, or user category such as followers and other people on the app.

Various aspects of the social media functionality do not work as intended. Although you can view your profile, the followers and following count is not implemented accurately, nor is it possible to follow other users in the app. Also, the settings that a user specifies when going through the questionnaire during authentication, such as private or public accounts and choosing to display their top songs, playlists, and artists, are not as accurate as we would have liked and contain minor delays. These minor functionalities would be implemented in future iterations of the app. This overall limits the social media aspect of the app and would be prioritized in the future.

The next critical aspect we would fix in the future is the bugs and glitches. The genre breakdown in the SelfProfileView works in some respects, but it is not production-ready. The pie chart shows the percentage of genres for the five largest genres, but the labels on the right side do not entirely reflect the genres within the pie chart as we would like. The genres instead display the top five niche genres which should be fixed. Another critical bug that would be updated is the playback. Its functionality works, but it is not as robust as we would like. It requires the Spotify app to be open in order for playback to work, but if the user swipes out of the Spotify app, they need to redo the authentication process to get playback to run properly again. Additionally, there are issues with the data where sometimes the song cover will not display on the queue of songs

or playlists will not properly be loaded in the user's profile view. In future versions, these bugs would need to be handled where the app can properly be used in a production setting.

Lastly, we would reconfigure how data is handled on the app. We would keep using Firebase, but we would use the user's iPhone cache also where we can load frequently used data there and it would not take long to retrieve data from Spotify and Firebase. Currently, it takes a small amount of time to load data to a user's profile. Utilizing the phone's cache would make it where data can persist on the user's local storage for faster retrieval. By taking care of all these critical issues in the future, we are confident that the Tempo app would be stable enough to be put on the Apple App Store.

References

[1] "Industry Data." IFPI. Accessed: Sep. 16, 2024. [Online.] Available:
https://www.ifpi.org/our-industry/industry-data/

[2] "U.S. Music Industry - Revenue Distribution by Source 2023." Statista. Accessed: Sep. 17,
2024. [Online.] Available:
https://www.statista.com/statistics/186304/revenue-distribution-in-the-us-music-industry/

[3] "U.S. Paid Music Subscribers 2023," Statista. Accessed: Sep. 17, 2024. [Online.] Available:
https://www.statista.com/statistics/707103/paid-streaming-music-subscribers-usa/

[4] A. Turner. "Spotify Users: How Many People Have Spotify? (2024)" Bankmycell.com.
Accessed: Nov. 20, 2024. [Online.] Available:
https://www.bankmycell.com/blog/number-of-spotify-users/

[5] "Web API Documentation." Spotify. Accessed: Sep. 17, 2024. [Online.] Available:
https://developer.spotify.com/documentation/web-api

[6] A. Ross. "Why I Finally Quit Spotify." The New Yorker. Accessed: Nov. 20, 2024. [Online.]
Available: https://www.newyorker.com/culture/infinite-scroll/why-i-finally-quit-spotify

[7] K. Allawadi and C. Vij, "A Smart Spotify Assistance and Recommendation System," 2023
International Conference on Advancement in Computation & Computer Technologies
(InCACCT), Gharuan, India, 2023, pp. 286-291, doi: 10.1109/InCACCT57535.2023.10141810.

[8] A. T. Su, T. Thet Aung, & H. H. Khaung Tin. "Trends and Patterns in Spotify's Most Played
Songs: A Comprehensive Analysis." 2024 5th International Conference on Advanced
Information Technologies (ICAIT), Yangon, Myanmar, 2024, pp. 1-6, doi:
10.1109/ICAIT65209.2024.10754921.

[9] "How to build a speaker box," Available:
https://www.diyaudioandvideo.com/Guide/BuildSpeakerBox/. [Accessed: Dec. 6, 2024].

[10] "15V 10A power supply adapter," Available:
https://www.amazon.com/15V-10A-Power-Supply-Adapter/dp/B0BS3PKP54/. [Accessed: Dec.
6, 2024].

[11] TE Connectivity, "Product datasheet: 282837-2," Available:
https://www.te.com/usa-en/product-282837-2.datasheet.pdf. [Accessed: Dec. 6, 2024].

[12] Littelfuse, "Fuse 251 datasheet," Available:
https://cdn1-originals.webdamdb.com/12956_127233578. [Accessed: Dec. 6, 2024].

[13] Lowe's, "Hillman single pole rocker light switch," Available: https://www.lowes.com/pd/Hillman-Single-Pole-Black-Compatible-with-LED-Rocker-Light-Switch/1000896654. [Accessed: Dec. 6, 2024].

[14] Laird, "EMI filtering and shielding solutions," Available: https://www.mouser.com/datasheet/2/987/Laird_10092020_Laird_MCP_Catalog_EMI_Filtering_and-1947796.pdf. [Accessed: Dec. 6, 2024].

[15] Switchcraft, "Sealed 10A power jacks," Available: https://www.mouser.com/catalog/specsheets/Switchcraft_%20Sealed_10A_%20Power_%20Jacks.pdf. [Accessed: Dec. 6, 2024].

[16] DFRobot, "DC-DC buck converter overview," Available: https://www.mouser.com/pdfDocs/Product_Overview-DFRobot-DC-DCBuckConverter1.pdf. [Accessed: Dec. 6, 2024].

[17] SameSky Devices, "PJ-102AH product resource," Available: https://www.sameskydevices.com/product/resource/pj-102ah.pdf. [Accessed: Dec. 6, 2024].

[18] Advanced Circuits, "Trace width calculator," Available: https://www.advancedpcb.com/en-us/tools/trace-width-calculator/. [Accessed: Dec. 6, 2024].

[19] IPC, "IPC-2221A: Generic standard on printed board design," Available: https://www.ipc.org/TOC/IPC-2221A.pdf. [Accessed: Dec. 6, 2024].

[20] OSH Park, "Super swift services," Available: https://docs.oshpark.com/services/super-swift/. [Accessed: Dec. 6, 2024].

[21] OSH Park, "Two-layer substrate: Kingboard KB6167F," Available: https://docs.oshpark.com/resources/two-layer-substrate-Kingboard-KB6167F.pdf. [Accessed: Dec. 6, 2024].

[22] Parts Express, "Quick reference guide," Available: https://www.parts-express.com/pedocs/more-info/320-699--quick-reference-guide.pdf. [Accessed: Dec. 6, 2024].

[23] Texas Instruments, "TPA3116D2 datasheet," Available: https://www.ti.com/lit/ds/symlink/tpa3116d2.pdf. [Accessed: Dec. 6, 2024].

[24] Dayton Audio, "PS180-8 6.5" point source full-range neo driver," Available: https://www.parts-express.com/Dayton-Audio-PS180-8-6-1-2-Point-Source-Full-Range-Neo-Driv-295-344. [Accessed: Dec. 6, 2024].

[25] Dayton Audio, "Driver specification sheet," Available: https://www.parts-express.com/pedocs/specs/PS180-8%206-1_2_%20Point%20Source%20Full-Range%20Neo%20Driver%20Specification%20Sheet.pdf. [Accessed: Dec. 6, 2024].

[26] "Aliexpress product listing," Available: https://www.aliexpress.us/item/3256807365018556.html. [Accessed: Dec. 6, 2024].

[27] Adams Desk, "Raspberry Pi 4 Model B hardware," Available: https://kb.adamsdesk.com/hardware/raspberry-pi-4-model-b/. [Accessed: Dec. 6, 2024].

[28] Raspberry Pi, "Raspberry Pi 4 datasheet," Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf. [Accessed: Dec. 6, 2024].

[29] "Amazon product listing: Raspberry Pi 4 Model B case," Available: https://www.amazon.com/dp/B0BZGW51FY. [Accessed: Dec. 6, 2024].

[30] E. Hammer-Lahav, "OAuth 2.0: The protocol for secure API authorization," IEEE Internet Computing, vol. 16, no. 1, pp. 12–19, Jan. 2012. Available: https://doi.org/10.1109/MIC.2012.12. [Accessed: Dec. 6, 2024].

[31] L. Chen, W. Wu, and Z. Zhang, "Personalized music recommendation systems: Techniques and applications," ACM Transactions on Information Systems (TOIS), vol. 28, no. 3, pp. 9:1–9:24, June 2010. Available: https://doi.org/10.1145/1734206.1734208. [Accessed: Dec. 6, 2024].

[32] M. B. Srivastava and C. E. Wills, "Leveraging cloud-based Firebase for scalable mobile applications," IEEE Internet of Things Journal, vol. 7, no. 4, pp. 3121–3133, Apr. 2020. Available: https://doi.org/10.1109/JIOT.2020.2969001. [Accessed: Dec. 6, 2024].

[33] S. Yang, L. Zhang, and W. Liu, "Conversational AI in mobile applications: Natural language processing for enhanced user interaction," IEEE Transactions on Consumer Electronics, vol. 65, no. 3, pp. 419–427, Aug. 2019. Available: https://doi.org/10.1109/TCE.2019.2928326. [Accessed: Dec. 6, 2024].

[34] Y. Benkler, The Wealth of Networks: How Social Production Transforms Markets and Freedom. Yale University Press, 2006. Social Science Computer Review, vol. 26, no. 2, pp. 259–261. Available: https://doi.org/10.1177/1084713807301373.

[35] U.S. Department of Health and Human Services, "Surgeon General's Advisory on Social Media and Youth Mental Health," [Online]. Available: https://www.hhs.gov/sites/default/files/sg-youth-mental-health-social-media-advisory.pdf. [Accessed: Sep. 12, 2024].

[36] F. J. Z. Borgesius, D. Trilling, J. Möller, et al., "Should we worry about filter bubbles?" Internet Policy Review, vol. 5, no. 1, 2016. Available: https://doi.org/10.14763/2016.1.401. [Accessed: Dec. 6, 2024].

[37] D. Gao and D. Yu, "Challenges and cracks: Ethical issues in the development of artificial intelligence," Science, Technology and Society, vol. 29, no. 3, pp. 415–434, 2024. Available: https://doi.org/10.1177/09717218241246372. [Accessed: Dec. 6, 2024].

[38] J. Li, "Analysis of the trend of Spotify," BCP Bus. & Manag., vol. 34, pp. 919–926, 2022, https://doi.org/10.54691/bcpbm.v34i.3112. [Accessed: Dec. 6, 2024].

[39] National Geographic, "Noise Pollution," National Geographic Education, [Online]. Available: https://education.nationalgeographic.org/resource/noise-pollution/. [Accessed: Sep. 12, 2024].

[40] C. Gordon, "ChatGPT and generative AI innovations are creating sustainability havoc," Forbes, Mar. 12, 2024. [Online]. Available: https://www.forbes.com/sites/cindygordon/2024/03/12/chatgpt-and-generative-ai-innovations-are-creating-sustainability-havoc/. [Accessed: Sep. 12, 2024].

[41] Apple Inc., "App Store review guidelines," Apple Developer Documentation, 2024. Available: https://developer.apple.com/app-store/review/guidelines/. [Accessed: Sept. 19, 2024].

[42] International Organization for Standardization, "ISO/IEC 27001: Information technology — Security techniques — Information security management systems — Requirements." Available: https://www.iso.org/standard/54534.html. [Accessed: Sept. 19, 2024].

[43] Silicon Labs, "Bluetooth® A2DP and AVRCP profiles," 2021. [Online]. Available: https://www.silabs.com/documents/public/application-notes/AN986.pdf. [Accessed: Sept. 12, 2024].

[44] International Electrotechnical Commission, "IEC 60065: Audio, video and similar electronic apparatus - Safety requirements." Available: https://webstore.iec.ch/publication/1008. [Accessed: Sept. 19, 2024].

[45] International Electrotechnical Commission, "IEC 61000: Electromagnetic compatibility (EMC) - Part 1: General." Available: https://webstore.iec.ch/publication/6000. [Accessed: Sept. 19, 2024].

[46] International Electrotechnical Commission, "IEC 60268: Sound system equipment." Available: https://webstore.iec.ch/publication/1894. [Accessed: Sept. 19, 2024].

[47] European Commission, "Directive 2002/95/EC (RoHS)." Available: https://ec.europa.eu/environment/waste/rohs_eee. [Accessed: Sept. 19, 2024].

[48] European Commission, "CE Marking." Available: https://ec.europa.eu/growth/single-market/ce-marking_en. [Accessed: Sept. 19, 2024].

[49] United States Patent and Trademark Office, "U.S. Patent 9,762,607: System and method for content recommendation based on natural language input." Available: https://patents.google.com/patent/US9762607B2. [Accessed: Nov. 29, 2024].

[50] United States Patent and Trademark Office, "U.S. Patent 10,284,828: System and method for visualizing relationships in a social network." Available: https://patents.google.com/patent/US10284828B2. [Accessed: Nov. 29, 2024].

[51] United States Patent and Trademark Office, "U.S. Patent 9,646,362: System and method for presenting personalized media analytics." Available: https://patents.google.com/patent/US9646362B2. [Accessed: Nov. 29, 2024].

[52] U.S. Patent 9,835,421, "Audio Device with Integrated Display for Enhanced Playback Control," issued Dec. 5, 2017.

[53] U.S. Patent 10,125,389, "Modular Audio System with Network Synchronization," issued Nov. 13, 2018.

[54] U.S. Patent 9,476,230, "Audio Device Featuring Optimized Enclosure for High-Fidelity Output," issued Oct. 25, 2016.

[55] Ballou, G. (2008). *Handbook for sound engineers* (7th ed.). Focal Press.

[56] J. Blauert, P. Laws; Group delay distortions in electroacoustical systems. *J. Acoust. Soc. Am.* 1 May 1978; 63 (5): 1478–1483.

[57] Olson, H. F. (1951). Direct radiator loudspeaker enclosures. *The Journal of the Acoustical Society of America*, 38. https://doi.org/10.1121/1.1917331

Appendix

PCB revision 1 – OSHPark board design
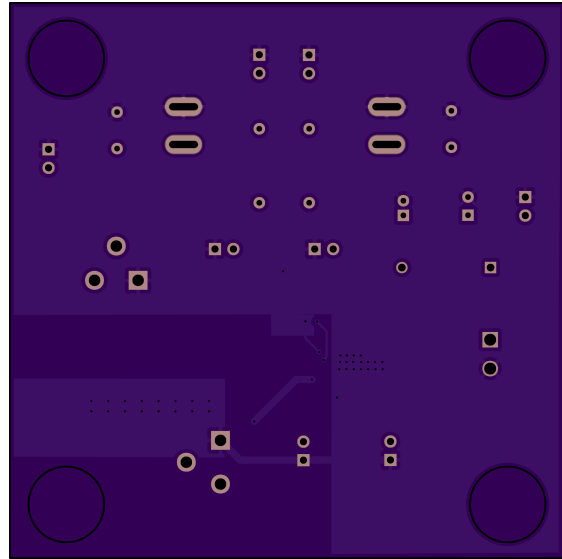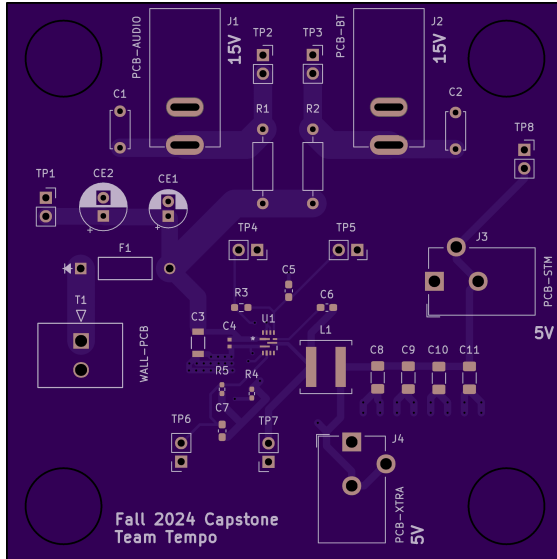
Project Rubric

iOS Application Performance (40 points)

Criteria	Excellent (36-4)	Good (32-35)	Satisfactory (28-31)	Needs a lot of improvement (24-27)	Did not attempt (0)
Functionality	Application delivers consistently stable performance with seamless music recommendations and profile management. No significant bugs present.	Application functions well with occasional minor issues that don't impact core functionality. Some non-critical bugs may be present.	Basic functionality is operational but with noticeable performance issues. Several bugs affect user experience.	Core features work inconsistently. Significant bugs impair basic functionality.	App did not work at all.
User Experience	Provides an exceptionally smooth and intuitive interface.	Generally positive user experience with minor navigation or interface issues	Usable interface with some friction points in navigation or interaction.	Poor user experience with significant usability issues.	Users could not access any part of the app easily.

Bluetooth Speaker Integration (40 points)

Criteria	Excellent (36-4)	Good (32-35)	Satisfactory (28-31)	Needs a lot of improvement (24-27)	Did not attempt (0)
Audio & Connectivity	Consistent, high-quality audio with stable Bluetooth connection. No disconnection issues.	Good audio quality with occasional minor connectivity interruptions.	Acceptable audio quality with periodic connectivity issues.	Frequent audio quality or connectivity problems.	No sound was produced.
Display Integration	LCD video does not lag, performs well, great synchronization with app changes.	Display functions well with minimal lag in updating information.	Display shows basic information with noticeable delay in updates.	Display information is frequently inaccurate or significantly delayed.	Display did not work.

Technical Compliance (20 points)

Criteria	Excellent (18-20)	Good (16-17)	Satisfactory (14-15)	Needs a lot of improvement (12-13)	Did not attempt (0)
Standards Adherence	Full compliance with all Bluetooth device regulations and iOS development guidelines.	Mostly compliant with minor deviations that don't affect functionality.	Meets basic compliance requirements with some notable exceptions.	Multiple compliance issues requiring significant attention.	No compliance, danger to the public.
Documentation	Fantastic organization of documentation, someone could quiz us on it.	Good documentation with minor gaps in technical details.	documentation present but lacking detail in key areas.	Incomplete or inadequate documentation	Missing documentation, unreliable.

Budget Outline Figures

Number	Start	Cost	Remaining	Link
Class Orders				
Parts Order 3	$500	$101.76	$398.24	Tempo-PartsOrder1.xlsx
Parts Order 5	$398.24	$55.30	$342.94	Tempo-PartsOrder5.xlsx
Parts Order 9	$342.94	$83.12	$259.82	Tempo-PartsOrder9.xlsx
Parts Order 10	$259.82	$75.76	$184.06	Tempo-PartsOrder10.xlsx
		Total spent:	$315.94	

Budget Outline: Class Orders

Personal Orders				
	TTI	RAPC10U	2	$28.98
	Parts Express	2x50W audio amplifier	1	$188.78
		8Ohm speaker drivers	2	
	OSHPark	Rev. 1 PCB	3	$100.40
	Parts Express	Binding mount	2	$64.29
		Binding posts		
	OSHPark	Rev. 2 PCB	3	$131.20
				$513.65

Budget Outline: Personal Orders

Rev. 1 PCB : Rev. 1 Information					
B	RAPC10U power barrel	J1, J2	2		28.98
	SK10U jack plug	J1, J2 (connect)	2	13.89	27.78
	PCB barrel jack mount	J3, J4	2	1.02	2.04
	PCB jack plug	J3, J4 (connect)	2	1.83	3.66
	Connector header	T1	1	0.29	0.29
	Connector housing	T1 (connect)	1	0.19	0.19
	Connector crimp	T1 (connect)	3	0.24	0.72
	12A fuse	F1	1	1.41	1.41
	100uF electrolytic capacitor	CE2	1	0.21	0.21
	0.47uF electrolytic capacitor	CE2	1	0.1	0.1
Check	100nF ceramic capacitor	C4 (CINX)	1	0.63	0.63
Check	2.2uF ceramic capacitor	C3 (CIN)	1	0.22	0.22
	100k resistor	R3 (RPG)	1	0.1	0.1
	1uF ceramic capacitor	C5 (CVCC)	1	0.1	0.1
	100nF ceramic capacitor	C6/C4 (CBOOT)	1	0.1	0.1
	20kOhm resistor	R4 (RFBB)	1	0.1	0.1
	147kOhm resistor	R5 (RBFT)	1	0.1	0.1
	Step-down regulator	U1	1	1.04	1.04
	1uH inductor	L1	2	3.1	6.2
	22uF ceramic capacitor	C8, C9, C10, C11 (COUT)	4	0.37	1.48
	22pF capacitor	C7 (CFF)	1	0.1	0.1
	100k resistor	R3 (RPG)	1	0.1	0.1
	8 test pins	TP1, ..., TP8	8	0.21	1.68

Budget Outline: Revision 1 PCB

Rev. 2 PCB: 📄 *Rev. 2 Information*					
	DC-DC power module	U1	1	4.9	4.9
	Ferrite beads	FB1, FB2	2	0.5	1
	4700uF electrolytic cap	CE3, CE5	2	2.05	4.1
	2000uF electrolytic cap	CE4	1	2.26	2.26
	0.1uF ceramic cap	C6, C7	2	0.19	0.38
	TE, wire to PCB connector	T1, T2, T3, T4, T5	4	0.8	3.2
	12V to 5V USB power module		1	5.99	5.99
	RAPC10U power barrel	J1, J2	2		28.98
	SK10U jack plug	J1, J2 (connect)	2	13.89	27.78
	PCB barrel jack mount	J3, J4	2	1.02	2.04
	PCB jack plug	J3, J4 (connect)	2	1.83	3.66
	12A fuse	F1	1	1.41	1.41
	7 test pins (2 extra)	TP1, ..., TP5	7	0.21	1.47

Budget Outline: Revision 2 PCB

External					
	AC/DC wall adapter		1	33.98	33.98
	16 AWG wires		1	5.24	5.24
	18 AWG wires		1	3.95	3.95
	Port to wire leads		1	6.75	6.75
	M5 screws		1	5.99	5.99
	Washers		1	5.79	5.79
Controls					
	DSI - LCD connector		1	14.26	14.26
	Discovery kit		1	87.5	87.5
	2 pin jumper (1)		1	1.6	1.6
	2 pin jumper (2)		1	2.1	2.1
T	BT evaluation board, 15V		1	49.59	49.59
T	BT evaluation board, 5V		1	25.8	25.8
Audio					
B	Audio amplifier		1	10.98	10.98
B	Speaker driver		2	84.15	177.8
B	Binding board		2		$19.96
B	Binding posts		2		$25.96
B	Shipping (Parts Express)				$18.37
Extra					
	TE, wire to PCB connector		2	0.8	1.6
	Power jack socket threads (6)		1	10.9	10.9

Budget Outline: Additional Costs

Final Product (Current Status)

Link to Github Repository: https://github.com/JCohen72/Tempo/tree/authentication