**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | Cross-Platform Development, Interactive Application Development |
| **Assessment Title:** | Music Player |
| **Lecturer Name:** | David González (dgonzalez@cct.ie), Sam Weiss |
| **Student Full Name:** | Luis Ramirez, Telmuun Dunia, Alejo Santos |
| **Student Number:** | 2023169, 2023306, 2023197 |
| **Assessment Due Date:** | Thursday, 8 May 2025, 12:01 AM |
| **Date of Submission:** | 06/05/2025 |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

Contents

[**Introduction.** 1](#_Toc197445807)

[**CROSS-PLATFORM DEVELOPMENT.** 2](#_Toc197445808)

[**How to Run the Application.** 2](#_Toc197445809)

[**I.** **Requirements.** 2](#_Toc197445810)

[**II.** **Installation Steps.** 2](#_Toc197445811)

[**III.** **Run the application.** 3](#_Toc197445812)

[**IV.** **Additional notes.** 3](#_Toc197445813)

[**Usage.** 3](#_Toc197445814)

[**Challenges Encountered and Solutions Implemented.** 3](#_Toc197445815)

[**Reflections on the Development Process.** 4](#_Toc197445816)

[**Potential Future Improvements.** 4](#_Toc197445817)

[**INTERACTIVE APPLICATION DEVELOPMENT.** 5](#_Toc197445818)

[**Part 1: Software Development Life Cycle (SDLC).** 5](#_Toc197445819)

[**Phase 1: Requirements Analysis.** 5](#_Toc197445820)

[**Phase 2: System Design.** 6](#_Toc197445821)

[**Phase 3: Implementation.** 6](#_Toc197445822)

[**Phase 4: Testing** 7](#_Toc197445823)

[**Phase 5: Deployment.** 8](#_Toc197445824)

[**Phase 6: Maintenance.** 8](#_Toc197445825)

[**Part 2: Software Development Life Cycle.** 8](#_Toc197445826)

[**Conclusion.** 15](#_Toc197445827)

[**Appendix.** 16](#_Toc197445828)

# **Introduction.**

This project is a modern, custom and user-centric music media player, developed using the Electron framework. The design was made thinking in simplicity and some common features that we used to find in the old media player Winamp. This project is a customizable and minimalistic alternative to mainstream media players.

The core functionality includes support for a wide range of audio formats like MP3, WAV, and Ogg, as well as control such as play, pause, skip forward/backwards, and volume adjustment.

To enhance the user experience, this media player integrates a real-time audio visualizer that responds dynamically to the playing track, and additionally, has a 3-band equalizer with bass, midrange, and treble controls.

Altogether, this project is the perfect example of functionality and aesthetics working together to create the best user experience.

# **CROSS-PLATFORM DEVELOPMENT.**

## **How to Run the Application.**

This media player was made to be easy to run and enjoy.

### **Requirements.**

The basic requirements to run this project are the same as the Electron frameworks, for example:

Operating System:

* Windows 7 and above
* macOS 10.10 (Yosemite) and above
* Linux

Node.js

* Node.js 9vesion 14.x or higher recommended)
* Must be installed because Electron is built on Node.[[1]](#footnote-1)

npm (Node Package Manager) is the default package manager for Node.js

* Running this command, you can install it: npm install

Once all these requirements are completed, you are ready to download the application from GitHub, or follow these steps.

### **Installation Steps.**

1. Clone the repository:

git clone https://github.com/CCT-Dublin/music-player-ca1-aoe

1. Install dependencies:

npm install

1. Install Electron:

npm install electron --save-dev

1. **Run the application.**

To run the application, we can use the command line with the following command:

npm run start

1. **Additional notes.**

Sample songs and sound files are included in the GitHub repository. But feel free to make your playlist using the Add files button.

## **Usage.**

Here is a brief explanation of some features that you will find inside this application:

1. The application loads music files initially from the /Music path in the directory.
2. Play, pause, next, previous, shuffle and repeat button controls.
3. Create a custom playlist using local songs with the Add folder button.
4. Change the playlist order by dragging and dropping songs to the desired location.
5. Lyrics fetched using an external API and displayed in the music player.
6. Audio visualizer and equalizer sliders.
7. Light and dark modes to better suit the environment.
8. Player minimize button to have only the essential controls for the music player.

## **Challenges Encountered and Solutions Implemented.**

|  |  |
| --- | --- |
| Challenge | Solution |
| Audio file compatibility | Implemented file format checking for .mp3, .wav and .ogg types. |
| UI size issues | Fixes to the media player app window not matching what is able to be seen on the screen. |
| Lyrics display is larger horizontally than the display | Optimized the fetched data from the external API, where the text is horizontally too long to fit in the display screen, so we broke the line according to the available size. |
| Cross-platform compatibility | Used Electron framework that supports multiple OS platforms. |
| Window controls | Fixed an issue where the playback buttons were breaking the window control buttons due to ipcRenderer issues. |

## **Reflections on the Development Process.**

Learning how Electron works and putting what we have been learning to a real world project was a rewarding experience.

Using GitHub for version control helped manage code collaboration smoothly. And organizing team meetings to discuss possible new features to add, as well as solutions to current issues/bugs, was beneficial.

Gained experience with Electron app development and GitHub branching and version control.

What could have been done differently was that the Early-stage planning of the project architecture could have reduced code refactoring later. Lacked a vision of the end product at the beginning, which made us change certain features several times over the course of the project.

## **Potential Future Improvements.**

* Add support for more audio formats (FLAC, AAC, and M4A files)
* Album art display.
* Improve lyrics integration
* Dark mode auto detection.
* Implement an advanced equalizer and visualizer animations.
* Implement .mp4 video playback feature.
* Improve performance and reduce overall compute load.

# **INTERACTIVE APPLICATION DEVELOPMENT.**

## **Part 1: Software Development Life Cycle (SDLC).**

The Waterfall methodology provides us with a clear and structured approach to managing our tasks. Since the project's scope and requirements (such as local audio playback, UI design, and lyric integration) were well-defined from the beginning, Waterfall allowed us to follow a step-by-step process with minimal need for mid-project changes. Each phase (planning, design, implementation, testing, and final deployment) was completed before moving to the next, which helped us stay organized and focused on completing one task at a time.

This linear structure made it easier to assign responsibilities among team members and manage progress, especially since we had limited time and a fixed set of features to deliver. The predictability and documentation focus of Waterfall also helped us in writing up our final report and maintaining clarity in our GitHub repository.

### **Phase 1: Requirements Analysis.**

In this initial phase, we defined the exact scope and functionality required for our music media player.

* Folder selection: Users must be able to select a folder from their file system. The app will persist this choice between sessions using local storage.
* Supported file types: Audio files with extensions .mp3, .wav, .ogg will be recognized and played.
* Playback controls: The app must allow users to play, pause, skip forward/backward, control volume, mute/unmute, and seek within a track.
* Track Info: Extract and display metadata such as title, artist, album, and duration using ID3 tags.
* Playlist view: A sidebar will list all detected tracks, allowing selection and playback.
* System integration: Implement tray icon with mini controls, desktop notifications for track changes, and support for media keys.
* UI/UX: Responsive layout, light/dark theme toggle, keyboard shortcuts, and resizable window with a mini media player.

### **Phase 2: System Design.**

Here we planned how the app will be built, focusing on structure, components, and technology stack:

Architecture Design:

* Main process (using electron): Handles file access, app lifecycle, tray integration, and media key binding.
* Renderer Process (HTML/CSS/JS): Manages UI rendering and playback logic.
* IPC (Inter-Process Communication): Used to send messages between renderer and main processes.

### **Phase 3: Implementation.**

This phase involves writing code based on the approved design:

* App Initialization:
  + Set up Electron app with basic menu and windows.
  + Implement folder selection using dialog.showOpenDialog.
  + Save folder path to local storage.
* File Handling:
  + Scan the selected folder for valid audio files.
  + Use music-metadata to extract ID3 tags.
  + Populate playlist with track info.
* Playback Functionality:
  + Load the selected track in the audio player.
  + Implement play, pause, skip, volume, seek, and mute.
  + Update UI in real-time during playback.
* User Interface:
  + Styled with CSS for dark/light themes.
  + Added keyboard shortcuts (e.g., Space for Play/Pause).
  + Made the window resizable with responsive layout.
* System Integration:
  + Tray icon with mini player.
  + Notifications when track changes.
  + Media keys (play, pause, next, previous) using Electron global shortcuts.

### **Phase 4: Testing**

Testing focused on functionality, performance, and cross-platform compatibility:

* Unit Tests:
  + Folder selection logic.
  + Metadata extraction.
  + Playlist rendering.
* Manual and End-to-End Testing:
  + Test playback controls on all OS (Windows, macOS, Linux).
  + Simulate user flows: folder selection → play → skip → pause.
  + Test UI responsiveness at different window sizes.
* Edge Case Handling:
  + Empty folder or unsupported files → shows an error.
  + Long file names → truncate in UI.
  + Missing metadata → display file name as fallback.

### **Phase 5: Deployment.**

We used electron-builder to create an installable version for Windows.

* Windows file .exe.
* Full GitHub repo.
* Documentation (README, user guide, developer notes).

### **Phase 6: Maintenance.**

Although the project has a defined endpoint, we prepared for future needs:

* Bug fixes: Team agreed to monitor GitHub Issues for 1 month after release.
* Feature requests (future): Shuffle, drag-and-drop playlist, equalizer.
* Dependency updates: Node/Electron libraries will be reviewed frequently.
* Backup strategy: Project repository backed up in Google Drive and GitHub.

## **Part 2: Software Development Life Cycle.**

The project was developed in the GitHub environment, therefore, the SDLC breakdown and individual contributions are well documented and available at:

<https://github.com/CCT-Dublin/music-player-ca1-aoe>

**Project overview.**

The objective of this project was to design and build a desktop media player using Electron, capable of playing local audio files with essential playback features, playlist management, equalizer controls, audio visualizations, and lyric display. The project was collaboratively developed using GitHub for version control and code collaboration, ensuring a well-managed workflow across different tasks and components.

Overall Project Work Breakdown.

|  |  |
| --- | --- |
| Component | Description |
| App Initialization (Electron Setup) | Setting up Electron’s main and renderer processes, including preload.js |
| UI & Layout Design | Designing the user interface using HTML, CSS, and JavaScript |
| Playback Features | Implementing play, pause, stop, next, previous buttons |
| Playlist Management | Loading music files, displaying playlist, and folder selection |
| Equalizer & Visualization | Connecting audio filters and creating animated visual bars |
| Lyrics Integration | Fetching lyrics from an external API and displaying them |
| Window Controls | Implementing custom close, minimize, and maximize buttons |
| Dark/Light Theme | Allowing user to toggle between UI themes |
| GitHub Collaboration | Using branches, pull requests, and issue tracking for coordination |

**Individual Team Member Contributions.**

**Telmuun – *Backend & Core Logic Lead.***

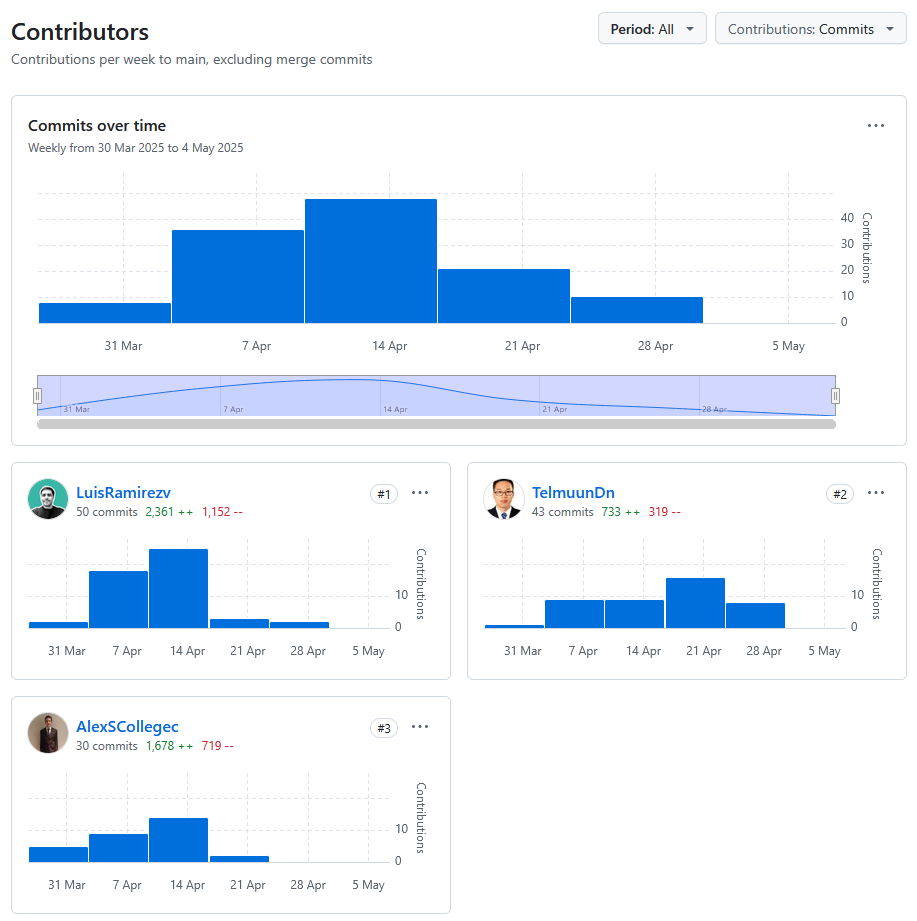
* Set up the **Electron main process**, including window creation and IPC communication.
* Implemented the logic for **handling audio playback**, equalizer filters, and visualization pipeline.
* Integrated the **file system access** to load music files and handle folder selection.
* Worked on **lyrics fetching functionality** using the external API.

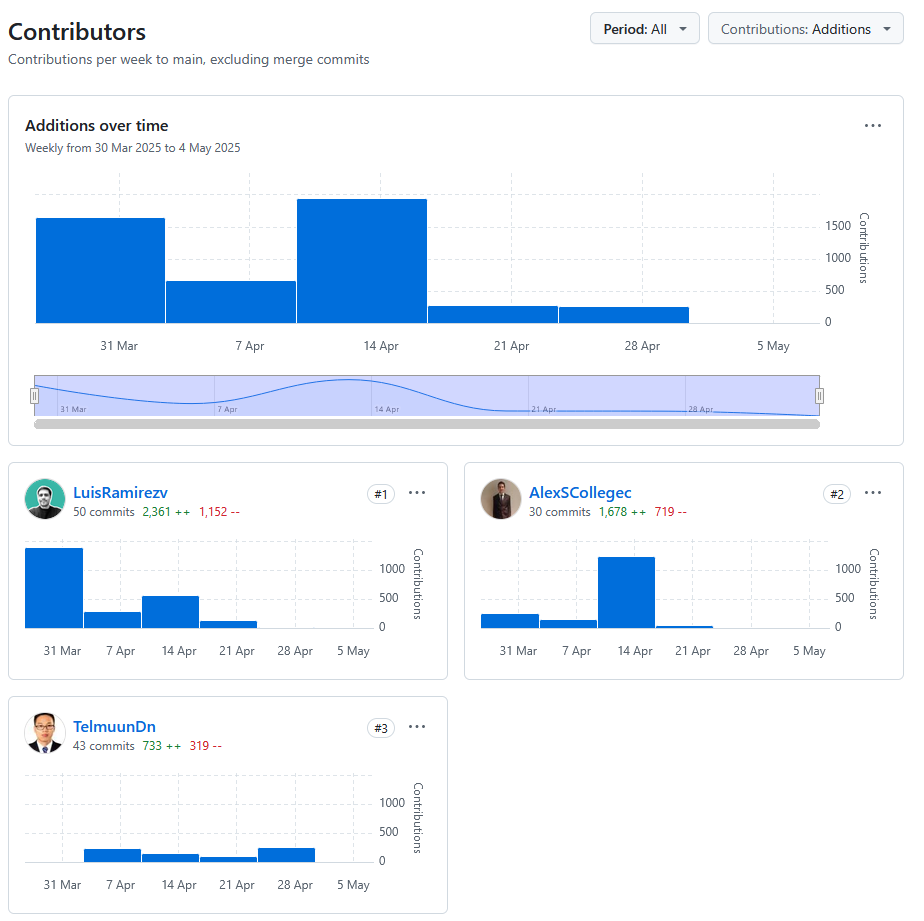
**Luis – *Frontend & UI Design Lead.***

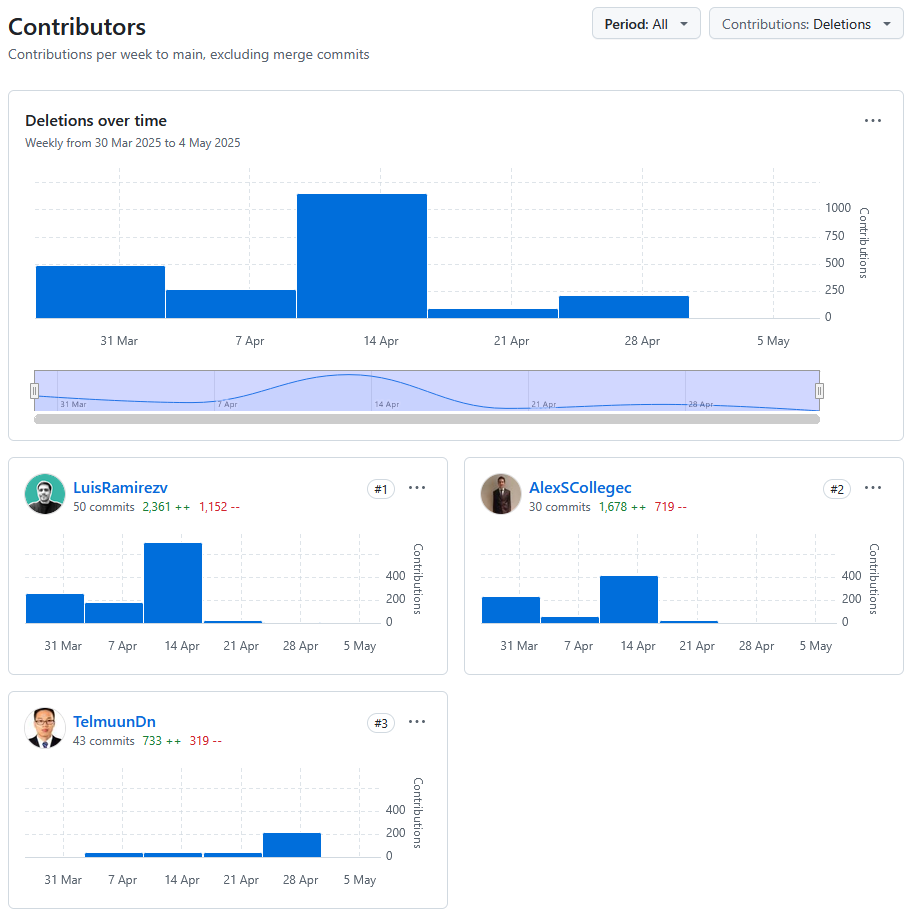
* Designed and styled the **user interface**, including playlist layout and playback controls.
* Implemented **dark/light theme toggling** and worked on improving user experience (UX).
* Developed the **equalizer control sliders**, volume bar, and song timer display.
* Created visual feedback animations for playback (e.g., animated EQ bars).

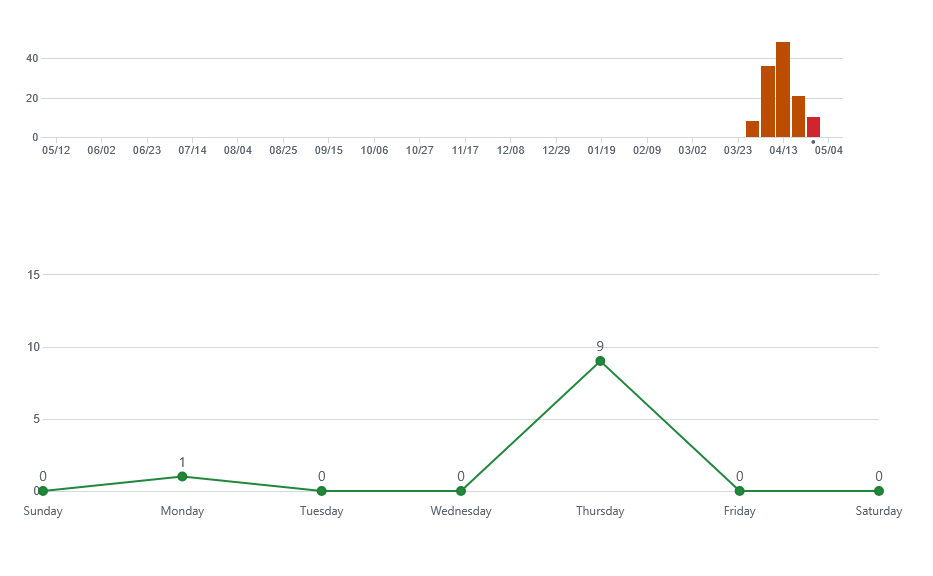
**Alejo – *Integration & GitHub Workflow Manager.***

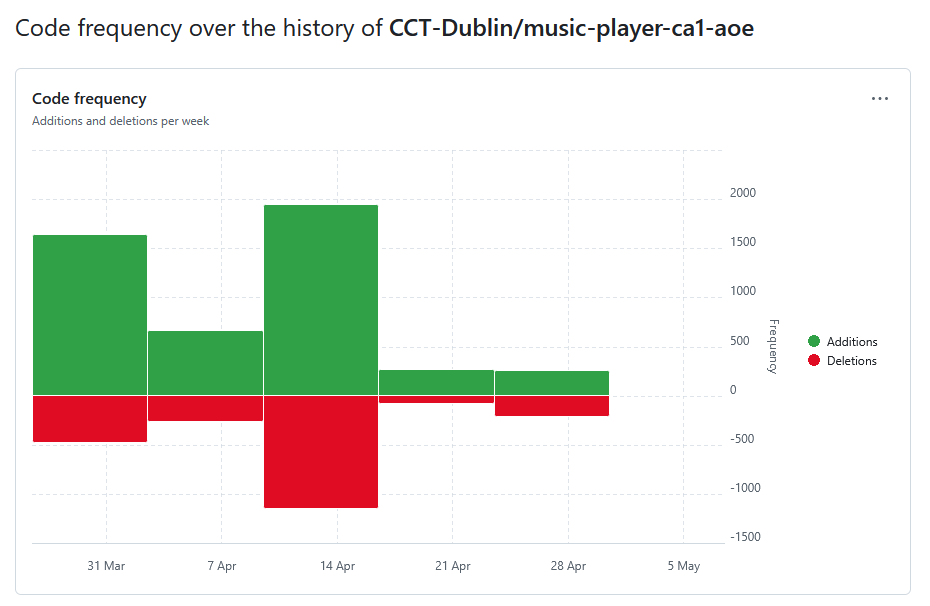
* Coordinated the **version control** workflow using GitHub (branches, merges, conflict resolution).
* Managed the **preload.js** and bridge between the main and renderer processes.
* Worked on **playlist logic**, handling dynamic creation of playlist items and event listeners.
* Assisted with **debugging and testing** to ensure smooth functionality across all features.

Contributions in terms of commits.

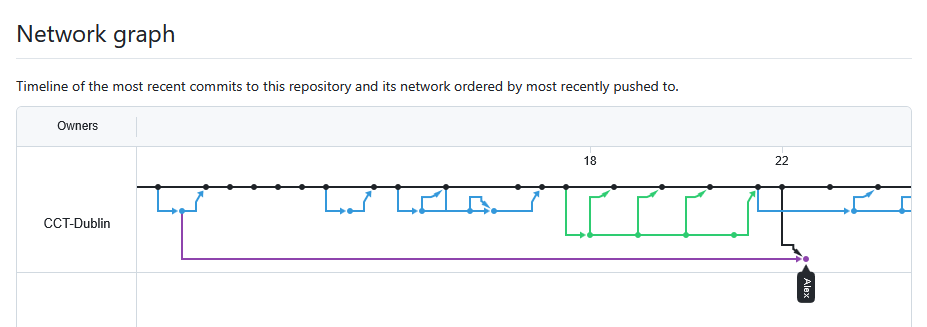
Contributions in terms of additions.

Contribution in terms of deletions.

Commit activity during the timeline of the project.

Code frequency during the lifetime of the project.

1. Network Graph of branches and versioning.

After the project was worked on a personal branch, the working prototype would be tested and reviewed by team members, and if agreed upon, pushed to the main branch. For full graph, please visit the link: <https://github.com/CCT-Dublin/music-player-ca1-aoe/network>

# **Conclusion.**

The development of this Electron-based media player has been a gratifying journey that combined creativity, collaboration, and technical problem-solving. By leveraging web technologies within a desktop application framework, we successfully created a functional, and user-friendly music player capable of handling local audio files, visualizing playback, applying equalizer settings, and even displaying lyrics. Along the way, we deepened our understanding of JavaScript, Electron, audio processing APIs, and project workflow using GitHub.

While the current version of the application meets its core objectives, it also lays a solid foundation for future enhancements. From improved user customization to expanded media support and smarter integrations, the project offers many avenues for continued development. Overall, this experience has not only strengthened our technical capabilities but also emphasized the importance of planning, teamwork, and adaptability in real-world software development.

# **Appendix.**

1. A screenshot of a video player

   AI-generated content may be incorrect.*Screenshot of the app running as intended.*
2. **A diagram of a company

   AI-generated content may be incorrect.**Architecture diagram. The following diagram has been generated by Mermaid.js from <https://gitdiagram.com/CCT-Dublin/music-player-ca1-aoe>

* The external API used to fetch the lyrics online:

<https://lyrics.ovh/>

* The external API used to fetch some of the buttons:

<https://fontawesome.com/>

1. You can install it from: <https://nodejs.org> [↑](#footnote-ref-1)