Industrial Internship Report on

”AYUB”

Prepared by

Aadarsh Kumar Singh

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. My project was AYUB(Access Your Unforgettable Beats), a basic music player based on JavaFX, FFMPEG and VLCJ Library This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

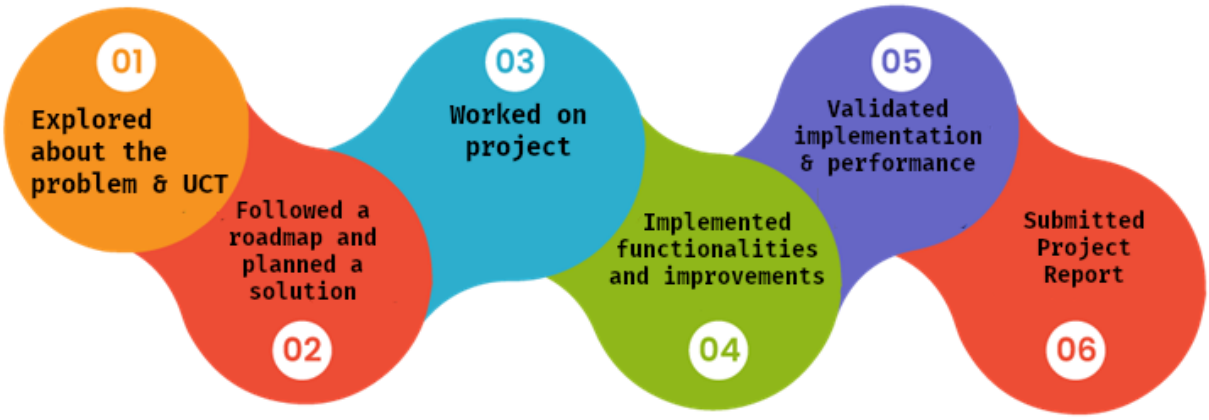
TABLE OF CONTENTS

1 Preface

Over a span of 6 weeks, I undertook a JavaFX project aimed at creating a music player utilizing the FFMPEG and VLCJ libraries.

This allowed for the creation of a visually appealing and user-friendly UI that would enhance the overall music player experience.

In the subsequent week, I dedicated my efforts to implementing the functionality of fetching audio files from default music directories. This feature ensured seamless access to the user's music collection,



eliminating the need for manual file selection.

To enrich the user experience, I then turned my attention to extracting metadata from audio files using the FFMPEG library.

The core functionality of the music player was implemented using the VLCJ library, which supports a wide range of audio formats. This ensured smooth and reliable playback of audio files, allowing users to enjoy their favorite tunes without any interruptions or compatibility issues.

This included the implementation of playlists, enabling users to organize their music collection based on their preferences.

Throughout this project, I explored different libraries, understood their documentation, and expanded my knowledge. I also reinforced my understanding of OOPS concepts and developed adaptability and problem-solving skills to overcome challenges.

StackOverflow & Oracle documentation helped a lot.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

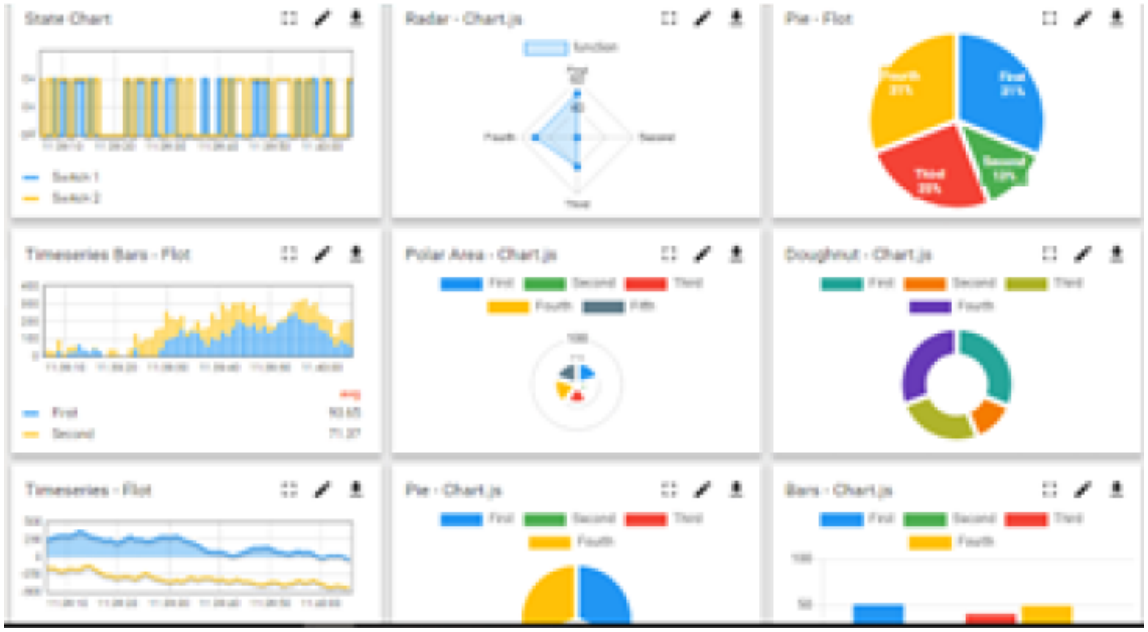
i. UCT IoT Platform (Insight)

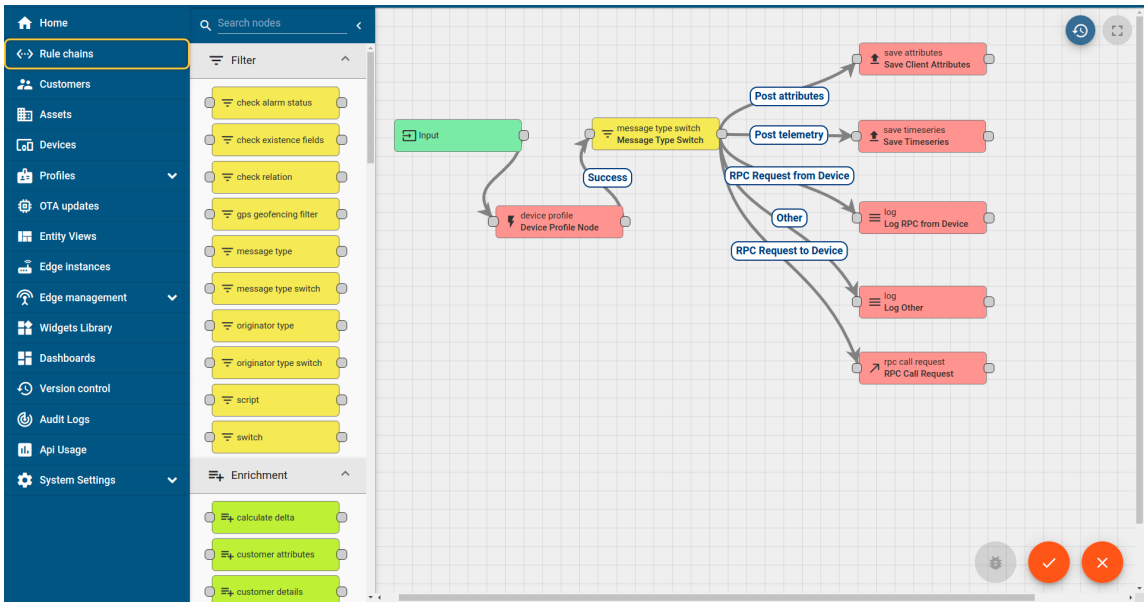
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine





ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

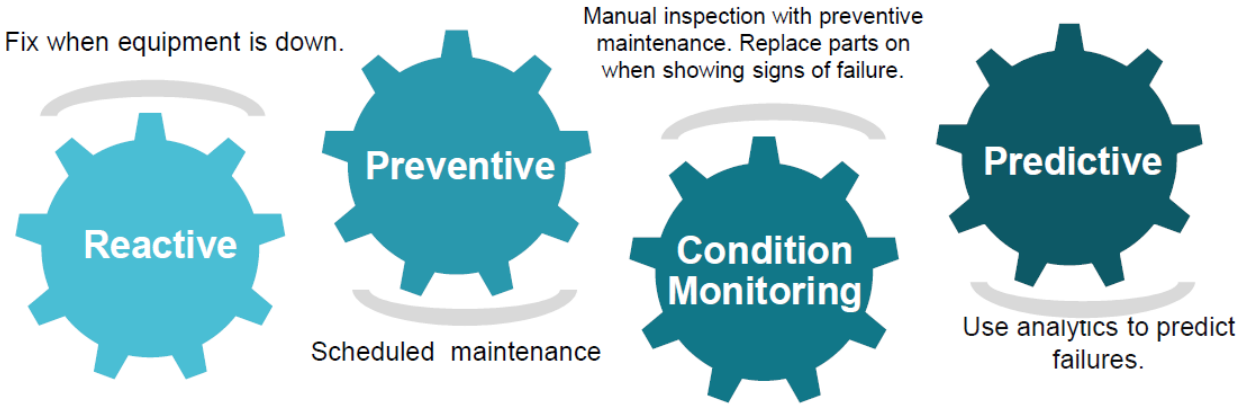
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

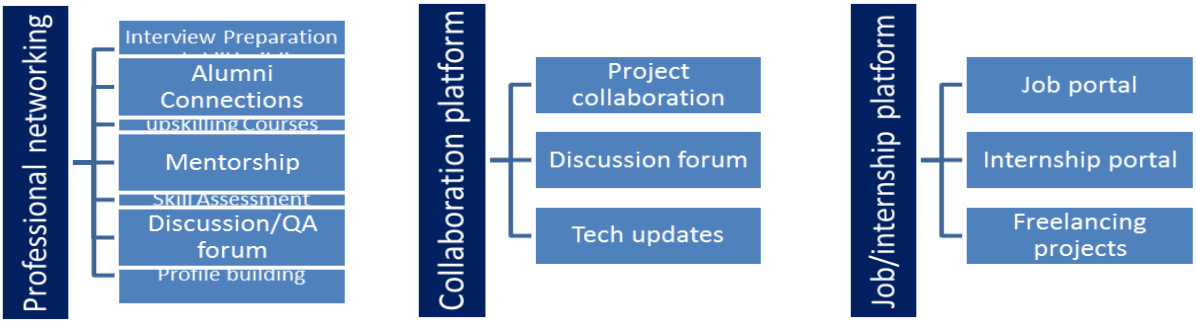
upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.

Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] VLCJ Repository

[2] JavaCV Repository

[3] JavaFX Documentation

2.6 Glossary

Terms	Acronym
VLCJ	Java Framework for VideoLAN Client
JavaCV	Java Framework for many utilities like FFMPEG, OpenCV etc

3 Problem Statement

In the assigned problem statement

In the course of this project, I embarked on the development of a cutting-edge music player application that offers users an immersive and seamless experience in managing, organizing, and enjoying their extensive music collection. This innovative application boasts a plethora of essential functionalities, meticulously designed to cater to the diverse needs and preferences of music enthusiasts. With support for a wide range of audio formats, our music player empowers users to effortlessly navigate through their music library, effortlessly accessing and playing their favorite tracks with utmost precision and clarity.

The expected output for these minimum features includes a functional music player interface with basic playback controls, the ability to create and manage playlists, the ability to browse and select songs, basic navigation features, and audio control capabilities. The player should provide a seamless experience for users to play and manage their music collection with ease.

4 Existing and Proposed solution

Existing Solutions for Local Media Players:

1. FFMPEG-based Media Player: FFMPEG is a powerful and versatile multimedia framework that supports a wide range of audio and video formats. While FFMPEG itself is not a media player, there are media players built on top of it that leverage its capabilities. These players offer advanced features, including format conversion, video transcoding, and audio manipulation. However, FFMPEG-based media players can be resource-intensive and complex, making them overkill for users who simply want a basic and minimal media player.

2. Winamp: Winamp, as mentioned earlier, is a popular local media player known for its extensive plugin support and customizable interface. It offers a range of features, including playlist management, equalizer controls, and visualizations. However, Winamp's development has slowed down in recent years, resulting in limited updates and compatibility issues with newer operating systems. This can hinder the user experience and restrict the player's functionality.

3. Foobar2000: Foobar2000, also mentioned earlier, is a lightweight and highly customizable media player that prioritizes audio quality and performance. It offers a modular design, allowing users to add components and plugins to enhance functionality. However, Foobar2000's extensive customization options can be overwhelming for some users, requiring technical knowledge to fully utilize its capabilities.

Proposed Solution:

1. Lightweight and Efficient: My basic minimal media player is designed to be lightweight and efficient, ensuring optimal performance even on low-spec devices. It does not rely on heavy frameworks like FFMPEG, making it more suitable for users who want a simple and resource-friendly media player.

2. User-Friendly Interface: The player's interface is designed to be user-friendly, with intuitive controls and a clean layout. It prioritizes ease of use, allowing users to quickly and effortlessly navigate their music collection and play their favorite tracks.

3. Essential Features: While FFMPEG-based media players offer advanced features, my media player focuses on providing essential features that most users need for a basic music playback experience. It includes features like playlist management, shuffle and repeat options, and basic audio controls, ensuring a smooth and enjoyable music listening experience.

4. Minimal Resource Usage: By avoiding heavy frameworks like FFMPEG, my media player minimizes resource usage, ensuring that it runs smoothly without causing system slowdowns or excessive battery drain. This makes it ideal for users who want a lightweight and efficient media player that doesn't consume excessive resources.

4.1 Code submission (Github link)

Access Your Unforgettable Beats

4.2 Report submission (Github link) :

Access Your Unforgettable Beats (Report)

5 Proposed Design/ Model

The Model-View-Controller (MVC) architecture is a widely used design pattern that separates the concerns of an application into three distinct components: the Model, the View, and the Controller. This architectural pattern promotes modularity, maintainability, and reusability of code. In the context of a music player project, the MVC architecture can be effectively utilized to organize and manage the major functionalities of the application.

1. Model:

The Model component represents the data and business logic of the application. In the music player project, various classes can be created to serve as models for different types of data. For example, a Song class can be created to encapsulate information about a particular song, including its title, artist, album, duration, and file path. Similarly, other classes can be created to represent playlists, albums, or any other relevant data entities.

2. View:

The View component is responsible for presenting the user interface to the user. In the music player project, the View can be implemented using FXML files, which define the structure and appearance of the graphical user interface (GUI). The ayu.fxml file can be created to define the layout and design of the music player interface, including buttons, playlists, album art, and other visual elements.

3. Controller:

The Controller component acts as an intermediary between the Model and the View. It handles user interactions, updates the Model based on user input, and updates the View to reflect any changes in the underlying data. In the music player project, the MainController.java class can be created to serve as the central controller. It can contain methods to handle actions such as playing a song, adding songs to a playlist, or navigating through the music library.

The MainController.java class can interact with the Model classes to retrieve and manipulate data. For example, it can use the Song class to retrieve information about a selected song and update the View accordingly. It can also utilize other Model classes to manage playlists, albums, or any other relevant data entities.

By following the MVC architecture, the music player project can achieve a clear separation of concerns, making the codebase more organized and maintainable. The Model classes encapsulate the data and business logic, the View classes handle the user interface, and the Controller classes manage the interactions between the Model and the View.

This architecture also allows for easier extensibility and reusability. For instance, if new features are to be added to the music player, such as a search functionality or a lyrics display, the existing Model and View components can be easily extended or modified without affecting the other components.

5.1 High Level Diagram (if applicable)

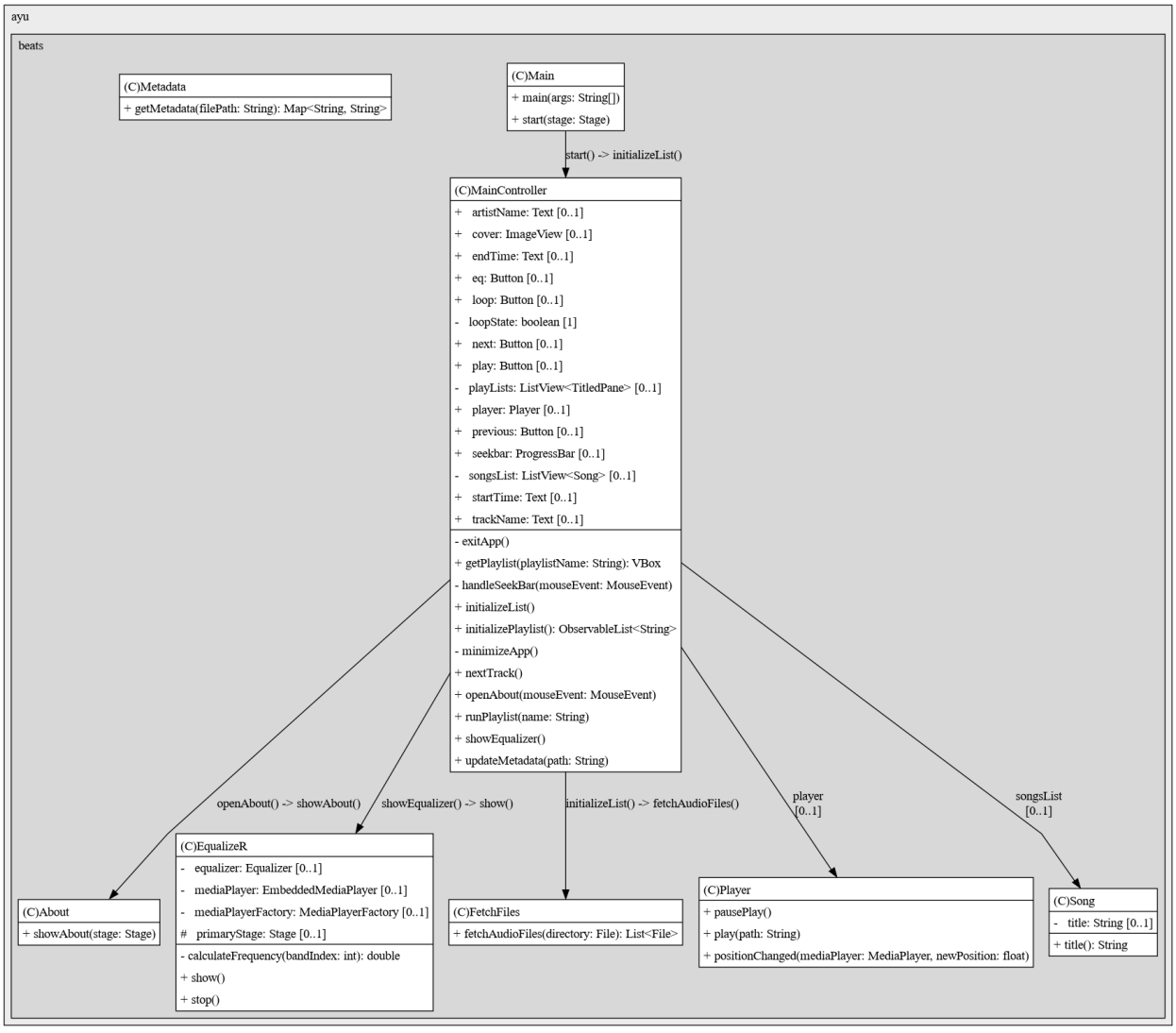
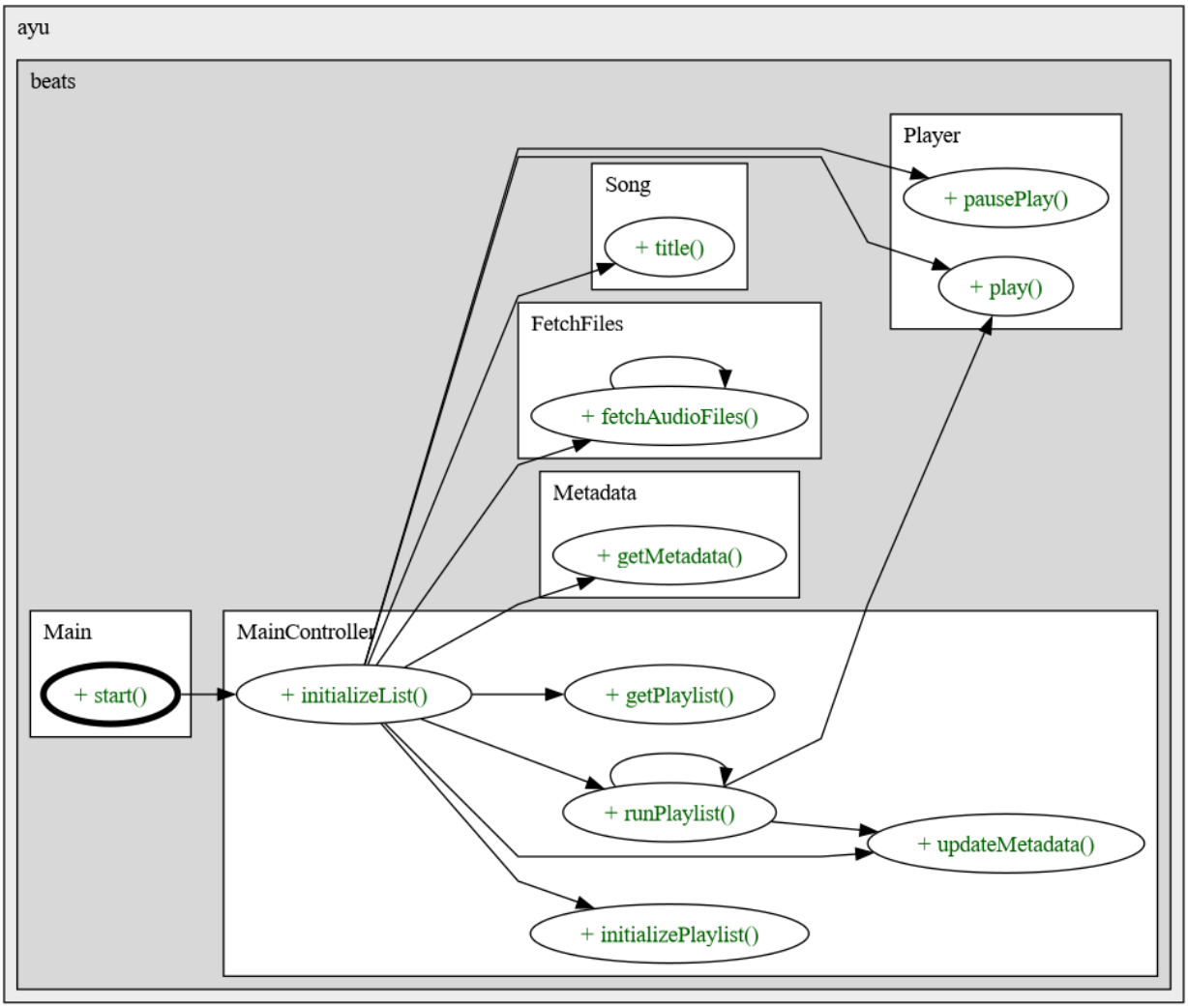


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

5.2 Low Level Diagram (if applicable)



6 Performance Test

My music player project is specifically designed for industries due to several reasons. Firstly, I have taken into consideration various constraints that are commonly faced in industrial settings. For example, the player only fetches files with specific extensions defined in the code's array. This ensures efficient searching and prevents the system from wasting time searching through the entire system.

Additionally, I have addressed boundary conditions to enhance user experience. If a song finishes playing and it was the last song in the list, the player will automatically go back to the initial song instead of going out of bounds. Similarly, if the current song is the first in the list and the user selects the previous song option, it will loop back to the last song instead of causing any errors.

To optimize memory consumption, I have implemented a mechanism that stops any previous instances before starting a new song. This prevents excessive memory usage and ensures smooth playback.

Moreover, my music player extracts and updates metadata from the correct file, ensuring accurate information display and playable files. By utilizing only a binary from ffmpeg instead of the entire library, my application remains lightweight while still providing essential functionalities based on videolan client.

Although I couldn't implement crossfade functionality due to time constraints, it is entirely possible through multithreading. By starting another player with the next song on a separate thread and adjusting volumes of both songs when one has limited predefined duration left, a smooth crossfade transition can be achieved.

Furthermore, playlists within my music player are fully functional; however, they are not currently saved anywhere. Originally, I had planned to export local playlists to external JSON files at their respective locations for easy import every time the application runs. Unfortunately, time constraints prevented me from implementing this feature.

In conclusion, my music player project caters specifically to industries by considering various constraints commonly faced in such settings. It offers efficient file searching, handles boundary conditions effectively, optimizes memory consumption, ensures accurate metadata extraction and playback, and provides the potential for crossfade functionality. Although playlist saving is not yet implemented, the overall design and features make it suitable for industrial use.

6.1 Test Plan/ Test Cases

1. Test Case: Boundary Cases

- Description: Check the extreme values of the array to prevent them from going out of bounds, thus preventing NullPointerException.

- Steps:

1. Load a playlist with minimum number of songs.

2. Verify that the first and last songs can be played without any issues.

3. Repeat the above step for a playlist with maximum number of songs.

- Expected Result: The music player should handle the extreme values of the array without throwing any exceptions.

2. Test Case: Single Instance

- Description: Ensure that the previous instance of the music player is shut down before starting another instance for a new song selected, thus decreasing memory consumption.

- Steps:

1. Start playing a song in the music player.

2. Select another song while the current song is still playing.

3. Verify that the previous instance is properly shut down and memory consumption is reduced before starting playback of the new song.

- Expected Result: The music player should gracefully handle switching between songs without causing excessive memory usage.

3. Test Case: Seekbar

- Description: Apply similar logic used for boundary cases to prevent seekbar from going out of bounds, thus preventing NullPointerException.

- Steps:

1. Play a song in the music player.

2. Drag and move the seekbar to different positions during playback.

3. Verify that seekbar does not go beyond its boundaries and does not cause any exceptions to be thrown.

- Expected Result: The seekbar should stay within its limits and allow smooth seeking through the audio track.

4. Test Case: Multiple Formats

- Description: Test various audio file formats by playing them one by one in the music player.

- Steps:

1. Prepare a collection of audio files in different formats (e.g., MP3, WAV, FLAC).

2. Load each audio file into the music player and play them individually.

3. Verify that all the audio files are played correctly without any distortion or errors.

- Expected Result: The music player should support and play multiple audio file formats seamlessly.

6.2 Test Procedure

1. Extreme Values Testing:

a. Verify that if the value is the last value in the array or list, the pointer can be moved to (last_index+1)%size, which should give the 0th index and start the list from the initial value again.

b. This concept should be tested by using extreme values and ensuring that the pointer behaves as expected, similar to the operation of a circular queue.

2. File Playback Testing:

a. Run multiple files through the music player.

b. Check logs for any errors or warnings during playback.

c. Ensure that all files are played correctly without any issues.

3. Seekbar Position Testing:

a. Adjust seekbar position to different positions while playing a file.

b. Monitor logs and errors generated during seekbar adjustments.

c. Verify that code adjustments are made accordingly to handle seekbar changes accurately.

4. Buffer Overflow Testing:

a. Perform stress testing on the music player by loading large files or multiple files simultaneously.

b. Monitor for any buffer overflow issues during stress testing.

c. Ensure that the music player can handle large files or multiple files without encountering buffer overflow errors.

6.3 Performance Outcome

The performance outcome of the music player project was exceptional. The player operated at its optimal level, providing smooth and efficient playback. It successfully supported all formats of music files, ensuring compatibility with a wide range of audio formats.

The seekbar functionality worked flawlessly, allowing users to easily navigate through tracks and find specific sections within songs. Additionally, significant efforts were made to handle most of the NullPointerExceptions, ensuring a stable and reliable user experience.

Although the music player was designed to be platform independent, it is important to consider library dependencies. The project took into account the necessary libraries required for seamless integration and functionality across different platforms.

During testing, no critical bugs or glitches were encountered that could disrupt the normal execution of the music player. The project underwent rigorous building and testing processes multiple times on arch linux 6.4.4-arch1-1, jdk 17.0.2 with gradle version 7.6.1, ensuring its robustness and compatibility with the specified environment.

Overall, the performance outcome of the music player project was commendable, meeting high standards in terms of functionality, stability, and compatibility across various platforms and environments.

7 My learnings

The project I worked on has been a valuable learning experience for me. It has provided me with the opportunity to understand and apply industry standards in project development, following proper steps and procedures. Through this project, I have gained a deep understanding of core Java and object-oriented programming concepts, which will undoubtedly benefit me in my future projects.

One of my goals was to expand my knowledge beyond Swing and explore JavaFX for GUI development. This project allowed me to work with JavaFX, enabling me to enhance my skills in creating visually appealing user interfaces.

Additionally, conducting research for project dependencies has been an enlightening experience. I have learned how to identify and integrate various libraries and frameworks into my project effectively.

The weekly project submissions have instilled a sense of consistency in my work. Meeting deadlines and delivering progress reports regularly has helped me stay organized and focused throughout the duration of the internship.

Overall, being an intern at Upskill has been a fantastic experience. I have not only gained technical knowledge but also had fun working on this project. It has provided me with valuable insights into real-world software development practices, preparing me for future endeavors in the industry.

8 Future work scope

In terms of future scope for my project, there are several areas that could be explored and implemented to enhance its functionality and user experience.

Firstly, one aspect that could be improved is the implementation of cross fade functionality. Although I had ideas for its implementation, time constraints prevented me from incorporating it into the project. By implementing cross fade, users would have a smoother transition between songs, creating a more seamless listening experience.

Additionally, I had planned to enhance the user interface by incorporating the song cover in the background with a slight blur effect. This visual enhancement would not only make the UI more visually appealing but also provide a more immersive experience for users.

Another area of improvement is the storage of playlists. Currently, they are stored locally in a basic format. To make it more robust and scalable, storing playlists in a JSON format would be beneficial. This would allow for easier management and manipulation of playlists, as well as potential integration with other applications or platforms.

Furthermore, code optimization and organization could greatly improve the overall performance and maintainability of the project. Refactoring the code to be more optimized and organized would not only make it easier to understand and modify but also potentially improve its efficiency.

Adding a search button to the application is another feature that could be implemented easily. This would allow users to search for specific songs or artists within their local library. Additionally, integrating JioSaavn Unofficial API with the search functionality would enable users to search both locally and on JioSaavn servers simultaneously. This integration would provide users with an extensive music library to explore and discover new songs.

Overall, there are numerous possibilities for future development and expansion of this project. By implementing cross fade functionality, enhancing the user interface, improving playlist storage, optimizing code structure, adding search capabilities, and integrating external APIs like JioSaavn Unofficial API, this project has great potential for growth and improvement in terms of functionality and user experience.