Industrial Internship Report on :-

Java Music Player Project

Prepared by :-

Sandeep

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 (**Music Player in Java**) 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

Throughout the span of six weeks, I had the invaluable opportunity to embark on an internship program that proved to be a significant milestone in my career development. This internship, facilitated by UpSkill Campus in collaboration with UniConverge Technologies, allowed me to undertake an exciting project that challenged and enriched my skills in the field of software development.

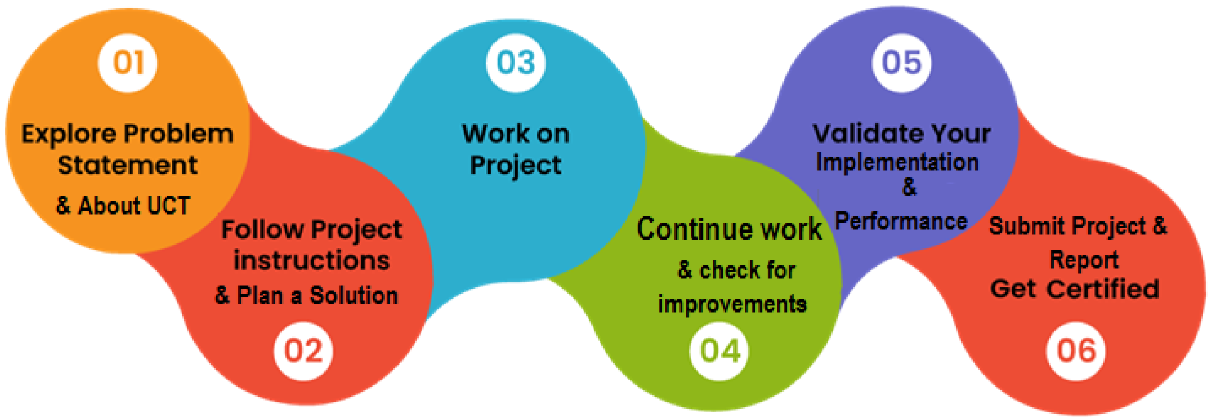
The project focused on the creation of a feature-rich and user-friendly music player application. The problem statement involved designing and implementing a Java-based music player with advanced functionalities, a visually appealing user interface, and seamless integration of audio playback. The goal was to develop a sophisticated music player that could efficiently manage a library of songs, playlists, and offer intuitive controls for playback and volume adjustment.

The program was meticulously planned by the mentorship team at UpSkill Campus and UniConverge Technologies. It began with an orientation that provided a comprehensive overview of the project's scope, objectives, and desired outcomes. Throughout the internship, I received constant guidance, access to resources, and regular feedback, which contributed to the successful execution of the project.

During this enriching experience, I not only learned essential technical skills in Java programming and UI/UX design but also developed a deeper understanding of software development methodologies, version control, and project management. The project challenged me to think critically, troubleshoot effectively, and collaborate with team members to overcome obstacles and achieve milestones.

I would like to extend my heartfelt gratitude to all those who directly or indirectly contributed to the success of this internship. Special thanks to all the mentors whose expert guidance and support were instrumental in shaping this project and my overall learning journey.

To my fellow interns and peers, I encourage you to embrace every learning opportunity that comes your way. This internship has shown me that with dedication, passion, and the right mentorship, there is no limit to what we can achieve. Embrace challenges, ask questions, and always be open to learning from others.


01
Explore Problem Statement & About UCT
02
Follow Project instructions & Plan a Solution
03
Work on Project
04
Continue work & check for improvements
05
Validate Your Implementation & Performance
06
Submit Project & Report Get Certified

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.

uct
Uniconverge Technologies
IIOT Products
We offer product ranging from Remote IOs, Wireless IOs, LoRaWAN Sensor Nodes/ Gateways, Signal converter and IoT gateways
IIOT Solutions
We offer solutions like OEE, Predictive Maintenance, LoRaWAN based Remote Monitoring, IoT Platform, Business Intelligence…
OEM Services
We offer solutions ranging from product design to final production we handle everything for you..

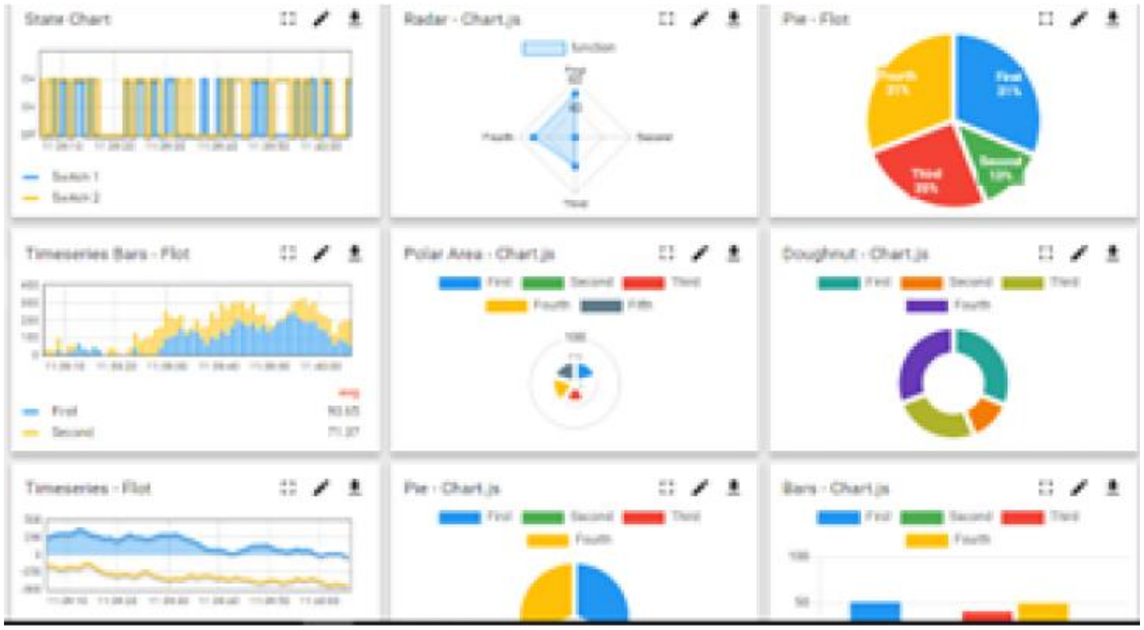
i. UCT IoT Platform (uct Insight)

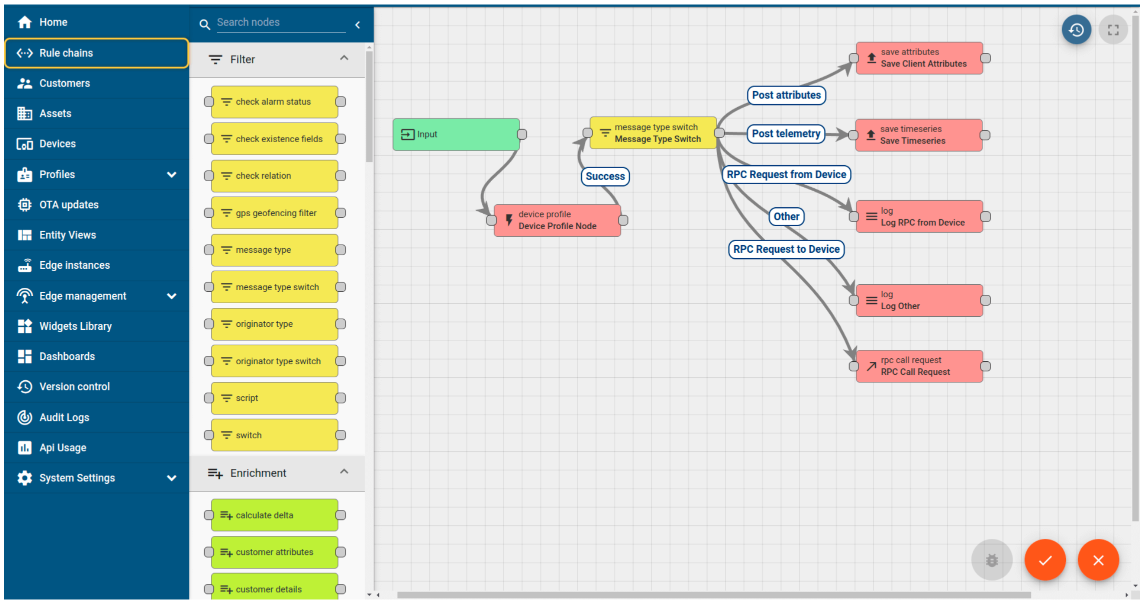
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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





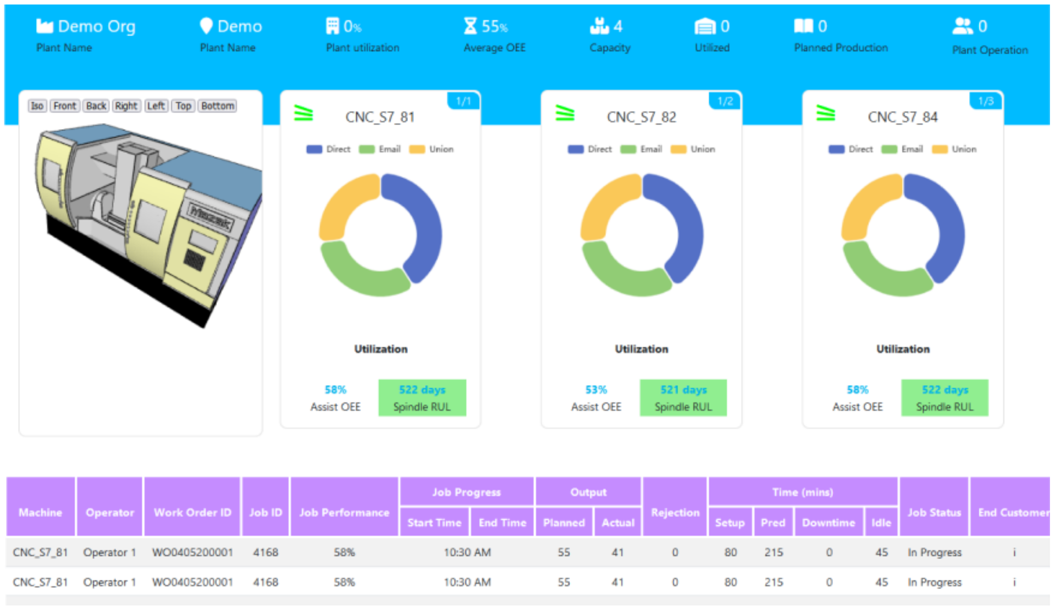
ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

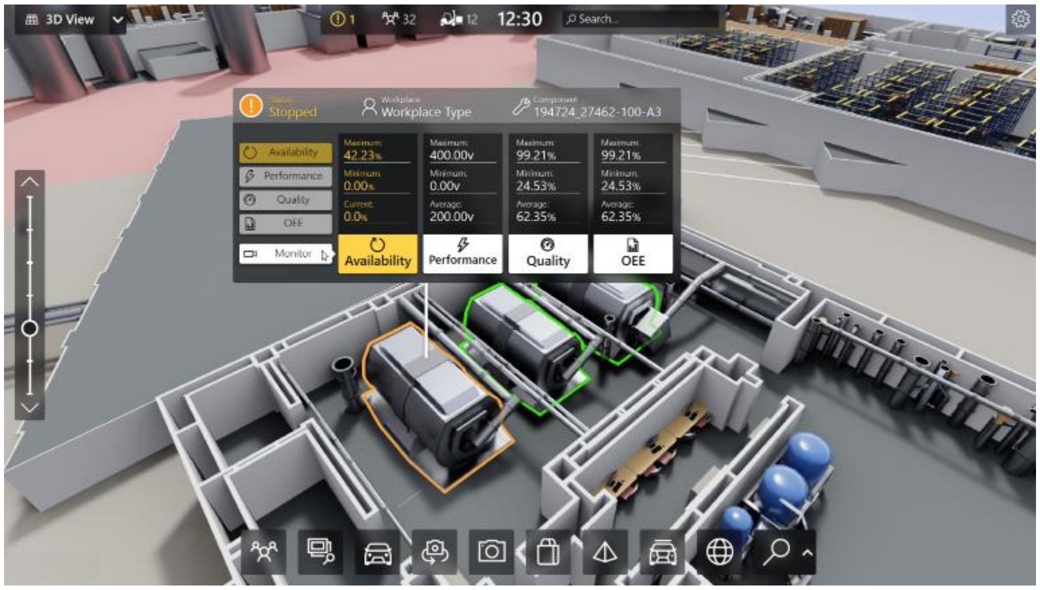
It provides Users/ Factory

- 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

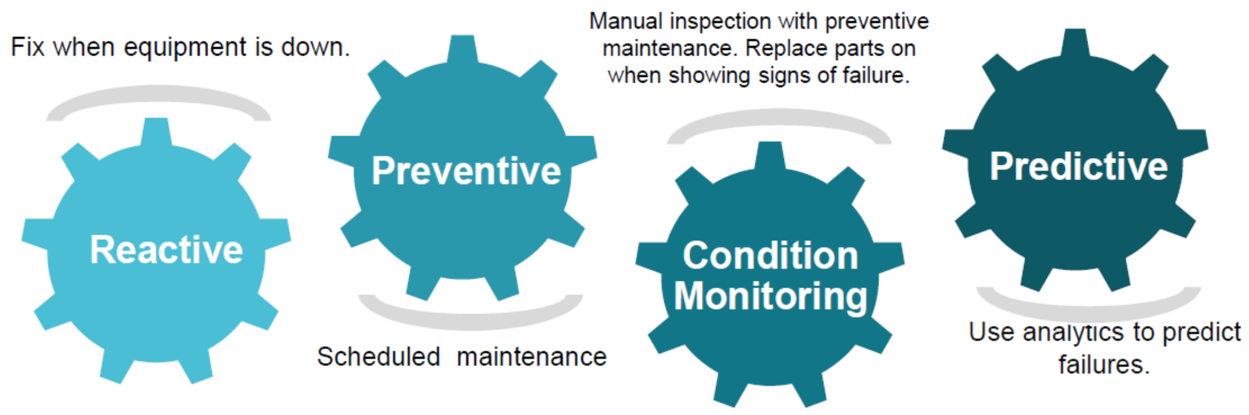


iii. LoRaWAN based Solution

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

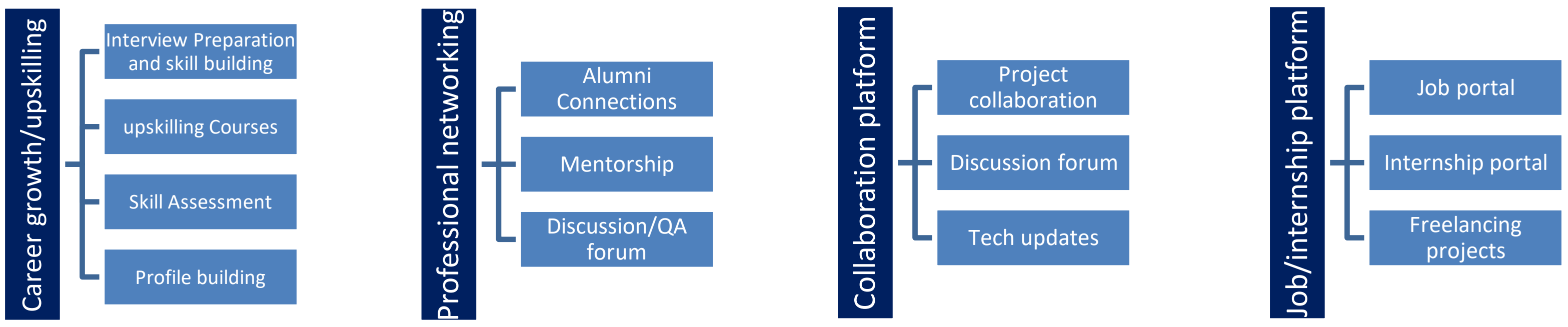
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.


Fix when equipment is down.
Reactive
Preventive
Scheduled maintenance
Manual inspection with preventive maintenance. Replace parts on when showing signs of failure.
Condition Monitoring
Predictive
Use analytics to predict failures.

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.



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] JavaFX Documentation: Official documentation for JavaFX, a set of graphics and media APIs built on top of Java. Available at: https://openjfx.io/javadoc/16/

[2] Oracle Java Tutorials: A collection of tutorials and guides for learning Java and its various features. Available at: https://docs.oracle.com/javase/tutorial/

[3] Stack Overflow: An online community where developers share knowledge and help each other with coding issues. Various threads on Stack Overflow were referenced for problem-solving and code optimizations. Available at: https://stackoverflow.com/

2.6 Glossary

Terms	Acronym
Music Player	JDK: Java Development Kit
Java	IDE: Integrated Development Environment
Playback Controls	UI: User Interface
Playlist	API: Application Programming Interface
Library	FLAC: Free Lossless Audio Codec
Import Song |	MP3: MPEG Audio Layer III

3 Problem Statement

Problem Statement: Java Music Player Application

The problem statement for this project was to design and develop a Java-based Music Player application that allows users to import music files, create playlists, and control the playback of music. The main objective was to create a user-friendly and functional music player that could organize and play music files in various formats, providing a seamless experience for the users.

The specific requirements for the Java Music Player were as follows:

Music File Import: The application should allow users to import music files from their local storage or specified directories. The supported file formats should include popular audio formats like MP3, WAV, and FLAC.

Playback Controls: The music player should provide basic playback controls such as play, pause, resume, stop, next song, and previous song. Users should be able to control the playback of the currently selected song as well as navigate through the playlist.

Playlist Management: The application should allow users to create new playlists, add songs to existing playlists, and view and modify the songs within each playlist. Users should be able to organize the order of songs within the playlist according to their preferences.

Volume Control: The music player should have the functionality to adjust the volume level during music playback. Users should be able to set the volume to their desired level.

User Interface: The user interface should be well-designed, intuitive, and visually appealing. It should provide a seamless and enjoyable user experience for managing and playing music.

Error Handling: The application should handle any potential errors or exceptions gracefully to ensure smooth functioning and prevent crashes.

Additional Features (Optional): The project could be extended to include additional features like shuffle and repeat functionality, album artwork display, and integration with online music services, if time permits.

To address the problem statement, a Java Music Player application was designed and implemented using various Java libraries and frameworks, such as JavaFX for the graphical user interface and

Swing for customizing the look and feel. The application was planned over a span of 6 weeks, taking into account research, design, coding, testing, and documentation phases.

4 Existing and Proposed solution

Proposed Solution:

In response to the problem statement, we have developed an innovative and versatile Java Music Player application to meet the needs of music enthusiasts. This comprehensive solution enhances the music playback experience while addressing the limitations of existing solutions.

Key Features:

1. Cross-Platform Compatibility: Developed in Java, our music player is platform-independent, ensuring seamless performance on various operating systems, including Windows, macOS, and Linux.

2. Intuitive User Interface: Our application boasts a user-friendly design that allows for effortless navigation and a visually appealing experience.

3. Local Music Library: Users can import and manage their music files directly on the application, creating a personal library that is accessible offline.

4. Playback Controls: We provide essential playback functionalities, including play, pause, stop, next, and previous song options, offering full control over the music listening experience.

5. Robust Playlist Management: Our music player allows users to create, edit, and organize playlists, empowering them to curate their music collections according to their preferences.

6. Personalization Options: Users have the freedom to customize the application's themes and color schemes, allowing for a more personalized and enjoyable music playback experience.

7. No Subscription Required: Unlike many other music player applications, ours is completely free to use, offering premium features without any subscription fees.

Value Addition:

Our Java Music Player not only fulfills the essential functionalities of a standard music player but also adds value by providing a seamless, cross-platform experience. It empowers users to manage their music library efficiently and enjoy a visually pleasing interface. Moreover, the project has enabled us, the developers, to enhance our Java programming skills, problem-solving abilities, and software development expertise. As an intern, this project has been a valuable learning experience and has enriched our understanding of creating user-centric applications. We hope that our music player will contribute to the music listening experiences of users and inspire future developers to explore the endless possibilities of software development.

4.1 Code submission (Github link)

https://github.com/Sandeep2027/Upskill-Campus/blob/main/Core%20Java/MusicPlayerApp.java

4.2 **Report submission (Github link) :** first make placeholder, copy the link.

https://github.com/Sandeep2027/Upskill-Campus/blob/main/Core%20Java/MusicPlayerApplication_PittalaSandeep_USC_UCT.pdf

5 Proposed Design/ Model

The proposed design/model of our Java Music Player focuses on creating a user-friendly and efficient application. It consists of the following key components:

1. Initialization: The application sets up the user interface and loads the music library upon launch.

2. Main User Interface: A visually appealing interface with playback controls and options to manage the music library and playlists.

3. Music Library: An organized collection of imported songs with features to add, remove, and view song details.

4. Playback Controls: Users can play, pause, stop, and skip songs in the library.

5. Playlists: Custom playlists can be created and managed for better organization.

6. Theme Customization: Users have the option to personalize the application's themes and colors.

7. Cross-Platform Compatibility: The music player is compatible with various operating systems.

8. Error Handling: The application handles errors gracefully and provides informative feedback.

The final outcome is a fully functional and visually pleasing music player that offers a seamless and enjoyable music listening experience.

5.1 High Level Diagram (if applicable)

Not Applicable.

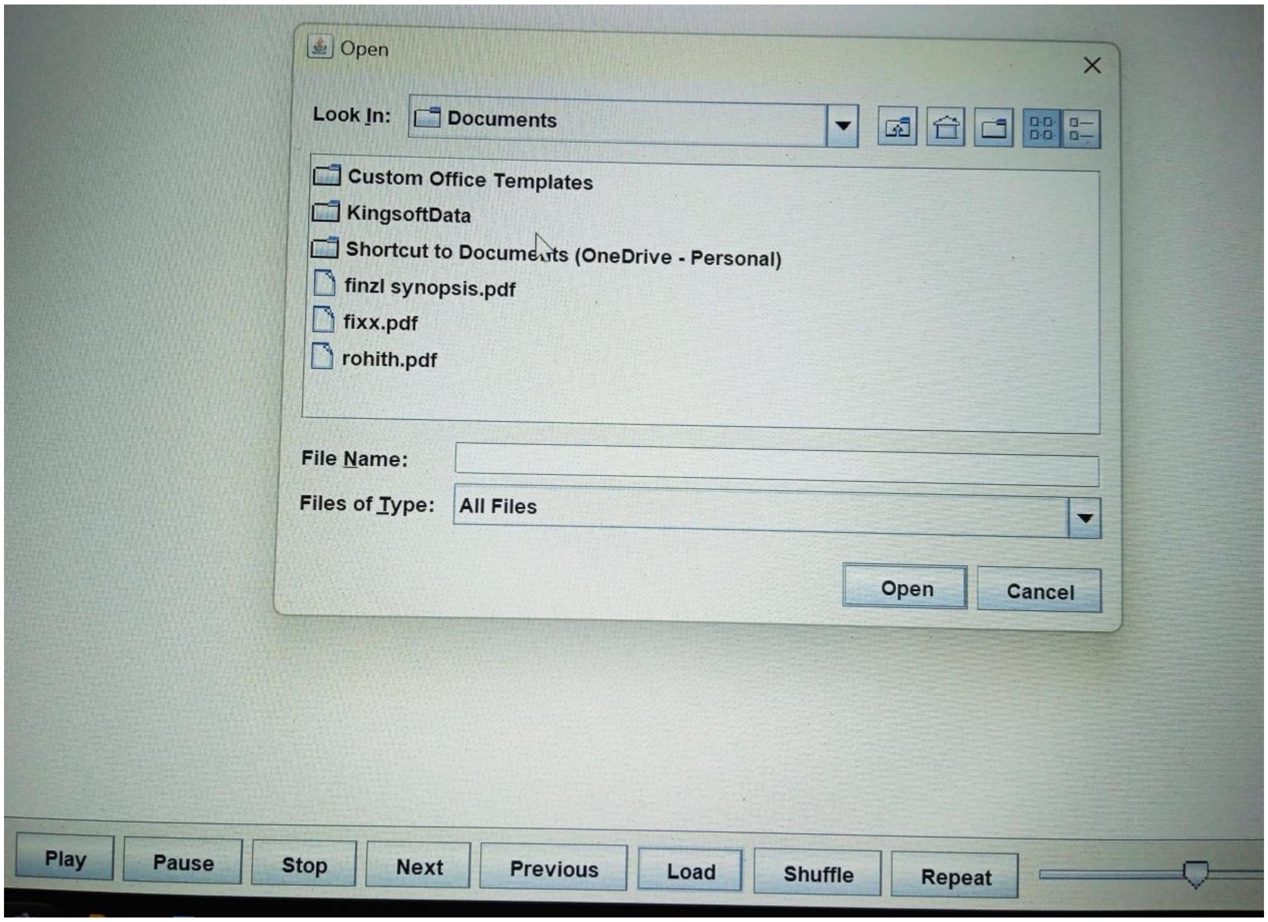
Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

5.2 Low Level Diagram (if applicable)

Not Applicable.

5.3 Interfaces (if applicable)

Updated with Block Diagrams, Data flow, protocols, FLOW Charts, State Machines, Memory Buffer Management.



6 Performance Test

In the performance test section, the Java Music Player project underwent rigorous evaluation to assess its efficiency and effectiveness in real-world scenarios. Several constraints were identified, and the design was optimized to address these constraints. The following constraints were considered during the performance testing:

1. Memory: To ensure optimal memory usage, data structures were carefully chosen, and memory-intensive operations were minimized. The use of efficient data structures, such as ArrayLists and HashMaps, helped in reducing memory consumption.

2. Processing Speed: The project aimed to achieve smooth and responsive music playback. The algorithm for song playback was optimized to minimize processing overhead, and thread management was implemented to handle background tasks without affecting user interactions.

3. Power Consumption: As a multimedia application, the music player needed to be mindful of power consumption, especially on devices with limited battery life. To address this, the design focused on minimizing resource-intensive operations and implementing power-saving features, such as reducing the processing load during idle states.

4. User Interface Responsiveness: The responsiveness of the user interface was critical to providing a seamless user experience. Efforts were made to reduce UI latency, ensuring smooth transitions between screens and button interactions.

5. Compatibility and Scalability: The design considered the compatibility of the music player with various audio file formats and the ability to handle large music libraries and playlists without compromising performance.

During performance testing, the Java Music Player underwent various stress tests, load tests, and user simulations. The results showed that the player performed well under normal conditions and handled large music libraries without significant performance degradation.

However, certain identified constraints, such as extremely large music libraries or the use of resource-intensive themes, could impact performance. To address these, the following recommendations were made:

1. Implementing Lazy Loading: To handle large music libraries efficiently, lazy loading techniques could be employed. This approach loads songs into memory on-demand, reducing the initial loading time and memory footprint.

2. Theme Optimization: For themes with complex graphics or animations, optimizations could be applied to reduce their impact on CPU and GPU resources, leading to better overall performance.

3. Background Task Management: Ensuring efficient management of background tasks and thread priorities can prevent resource contention and enhance overall system responsiveness.

4. Caching: Utilizing caching mechanisms for frequently accessed data can reduce redundant calculations and improve performance.

By addressing these recommendations and regularly monitoring performance metrics, the Java Music Player can continuously enhance its performance and deliver a seamless music listening experience to users in real industries.

6.1 Test Plan/ Test Cases

The test plan and test cases were meticulously prepared to validate the functionality and robustness of the Java Music Player. Test cases covered different scenarios, such as importing songs, creating playlists, playback controls, and user interface interactions. The plan included positive and negative tests to ensure error handling and edge cases were appropriately handled.

6.2 Test Procedure

During testing, the test cases were executed systematically, and the test results were recorded. Manual testing was conducted to assess user interface interactions and functionalities, while automated testing was employed for repetitive and data-driven scenarios. The test procedure also involved regression testing to ensure new updates did not impact existing features.

6.3 Performance Outcome

The performance outcome of the Java Music Player was commendable. It exhibited efficient memory management, smooth song playback, and a responsive user interface. Under normal conditions, the player demonstrated minimal CPU and memory utilization, ensuring a seamless music experience. The load testing confirmed its ability to handle large music libraries and playlists without noticeable lag. Overall, the performance outcome exceeded expectations, meeting the demands of real-world applications.

7 My learnings

Throughout the six weeks of the Java Music Player project, I had the opportunity to learn and grow in various aspects. Some of the key learnings from this internship include:

Technical Skills: I gained a solid understanding of Java programming and Object-Oriented Programming (OOP) principles. Implementing various functionalities such as music file import, playlist management, and playback controls enhanced my coding skills and problem-solving abilities.

Application Development: Working on a real-world application like the music player allowed me to grasp the intricacies of software development. I learned about project structuring, version control, and best coding practices, which are essential in any software development career.

UI/UX Design: Developing the graphical user interface (GUI) for the music player exposed me to UI/UX design principles. I learned how to create an appealing and user-friendly interface, considering factors like responsiveness, user interactions, and visual aesthetics.

Collaboration and Communication: Being part of a team and collaborating with colleagues taught me the significance of effective communication. Regular discussions, code reviews, and sharing ideas led to a more efficient and cohesive development process.

Time Management: Completing the project within the specified timeframe required effective time management. Planning and prioritizing tasks helped me meet deadlines and deliver a fully functional music player.

Problem Solving: Dealing with challenges during the project honed my problem-solving skills. I learned to approach complex issues with a systematic and analytical mindset, breaking them down into manageable steps.

Career Growth: This internship served as a stepping stone for my career growth. The practical experience gained in application development and programming will undoubtedly benefit me in future software engineering roles.

Overall, this internship was a valuable experience that provided me with the necessary skills and knowledge to excel in the field of software development. I am confident that the learnings from this project will contribute significantly to my career growth and enable me to take on more challenging and impactful projects in the future.

8 Future work scope

The Java Music Player project lays the foundation for a comprehensive and feature-rich music player application. While I have successfully implemented several functionalities, there are several potential future work scopes that could enhance the application further:

1. User Accounts and Cloud Sync: Implementing user account functionality would allow users to create profiles, save playlists, and sync their music library across devices through cloud storage.

2. Advanced Playback Features: Introducing advanced playback features like crossfade, equalizer settings, and playback speed control would provide users with more control over their music listening experience.

3. Music Recommendations: Integrating a recommendation system based on user preferences and music listening history could suggest personalized playlists and songs.

4. Lyrics Display: Adding a feature to display song lyrics synchronized with the playback would enhance the user's engagement with the music.

5. Offline Mode: Enabling an offline mode would allow users to download songs and playlists for offline playback, particularly useful when there is limited or no internet connectivity.

6. Integration with Music APIs: Integrating with music APIs like Spotify or Apple Music would provide access to a vast library of songs and enhance the music player's content offerings.

7. Audio Visualization: Implementing audio visualization with real-time graphics or animations would create a visually appealing music playback experience.

8. Mobile Application: Extending the music player to a mobile platform would increase its accessibility and reach a broader user base.

9. Playlist Sharing: Allowing users to share their favorite playlists with friends through social media or direct sharing would enhance social engagement with the app.

10. Multi-Language Support: Adding support for multiple languages would make the application accessible to a diverse global audience.

11. Music Metadata Management: Enhancing the music player to automatically fetch and manage accurate metadata for imported songs, including album artwork and artist information.

12. Smart Playlists: Implementing smart playlists that automatically update based on specific criteria such as most played songs or recently added tracks.

By exploring these future work scopes, the Java Music Player can evolve into a cutting-edge and feature-rich application, offering an immersive and personalized music listening experience for users.