**Core Java Internship Report on**

**Music Player Application**

**Prepared by**

**Bibhu padarabinda Behera**

|  |
| --- |
| *Executive Summary* |
| This report provides details of the Core Java 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 layer application that allows users to play, manage, and enjoy their music collection. The music player should provide essential functionalities for organizing and playing music files in various formats.  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 3](#_Toc139702806)

[2 Introduction 4](#_Toc139702807)

[2.1 About UniConverge Technologies Pvt Ltd 4](#_Toc139702808)

[2.2 About upskill Campus 8](#_Toc139702809)

[2.3 Objective 9](#_Toc139702810)

[2.4 Reference 9](#_Toc139702811)

[2.5 Glossary 10](#_Toc139702812)

[3 Problem Statement 11](#_Toc139702813)

[4 Existing and Proposed solution 12](#_Toc139702814)

[5 Proposed Design/ Model 13](#_Toc139702815)

[5.1 High Level Diagram (if applicable) 13](#_Toc139702816)

[5.2 Low Level Diagram (if applicable) 13](#_Toc139702817)

[5.3 Interfaces (if applicable) 13](#_Toc139702818)

[6 Performance Test 14](#_Toc139702819)

[6.1 Test Plan/ Test Cases 14](#_Toc139702820)

[6.2 Test Procedure 14](#_Toc139702821)

[6.3 Performance Outcome 14](#_Toc139702822)

[7 My learnings 15](#_Toc139702823)

[8 Future work scope 16](#_Toc139702824)

# Preface

Summary of the whole 6 weeks’ work. About need of relevant Internship in career development.

Brief about Your project/problem statement. Opportunity given by USC/UCT.

How Program was planned



Your Learnings and overall experience.

Thanks to all, who have helped you directly or indirectly.

Your message to your juniors and peers.

# Introduction

## 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.



1. UCT IoT Platform

**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.





1. **Smart Factory Platform**

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.

1.  based Solution

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

1. 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.



## 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/>



## 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.

## 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.

# Problem Statement

**Develop a music player application that allows users to play, manage, and enjoy their music collection. The music player should provide essential functionalities for organizing and playing music files in various formats.**

Required Functionalities:

1. **Music File Import:** Enable users to import music files from their local storage or specified directories into the music player library. The player should support common audio file formats, such as MP3, WAV, and FLAC.
2. **Music Playback:** Implement the ability to play, pause, resume, and stop music playback. The player should provide controls for adjusting the volume and seeking within the track.
3. **Playlist Management:** Allow users to create and manage playlists, enabling them to group their favorite songs or create custom collections. Users should be able to add and remove songs from playlists and organize the playlist order.
4. **Music Library Organization**: Provide features to organize and categorize music files within the player's library. Users should be able to create folders, assign tags or metadata to songs, and search for specific songs or artists.
5. **Audio Equalizer:** Implement an audio equalizer that allows users to adjust the sound output according to their preferences. The equalizer should provide pre-defined presets and allow users to customize the equalizer settings manually.
6. **Shuffle and Repeat:** Include options for shuffling the playlist order and repeating individual tracks or the entire playlist.
7. **Crossfade:** Implement a crossfade feature that smoothly transitions between songs, creating a seamless listening experience.
8. **User Interface:** Design an intuitive and user-friendly interface that provides easy navigation, displays album art and song information, and includes controls for playback, playlist management, and other functionalities.
9. **Metadata Display:** Retrieve and display metadata information, such as song title, artist, album, and duration, for each music file in the player's library.
10. **File Format Compatibility:** Ensure the music player supports a wide range of audio file formats to accommodate different user preferences and file types.

Minimum Features for a Music Player:

**1. Music Playback:** Users should be able to play, pause, and stop music playback.

Output: The music player starts playing the selected song and displays basic playback controls such as play, pause, and stop.

**2. Playlist Management:** Users should be able to create and manage playlists.

Output: Users can create a new playlist, add songs to the playlist, and view and modify existing playlists.

**3. Music Library Organization:** Users should be able to browse and select songs from their music library.

Output: The music player displays a list of available songs in the library. Users can select a song to play.

**4. Basic Navigation:** Users should be able to navigate through the music library and playlists.

Output: Users can browse through their music library, view songs by artist, album, or genre, and switch between different playlists.

**5. Audio Control:** Users should be able to adjust the volume of the music playback.

Output: Users can increase or decrease the volume of the music player, and the output audio volume changes accordingly.

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.

# Existing and Proposed solution

Proposed Solution: Java Music Player Application

The proposed solution is to develop a versatile and interactive Java Music Player Application that offers a seamless listening experience to users across multiple platforms. The application will be designed using Java programming language, incorporating the JavaFX library for a user-friendly graphical interface. The main objectives of the proposed solution are as follows:

**1. Cross-Platform Support:**

The Java Music Player Application will be developed to run on various platforms, including desktop (Windows, macOS, Linux), mobile (Android, iOS), and web browsers. This cross-platform compatibility will ensure that users can access their music library and enjoy their favorite tracks regardless of their device.

**2. User-Centric Design:**

The application will focus on user-centric design principles to create an intuitive and visually appealing user interface. The design will emphasize smooth navigation, easy access to core functionalities, and minimal learning curves for users to quickly adapt to the application.

**3. Music Playback and Core Functionality:**

The music player will be equipped with essential playback controls, including play, pause, stop, skip, and volume control. Users will be able to browse their music library, create playlists, and manage their music collection effortlessly.

**4. Cloud Integration:**

The proposed solution will integrate cloud storage services to enable seamless access to music collections across devices. This feature will allow users to sync their music and playlists, ensuring a consistent experience regardless of the device they use.

**5. AI-Powered Music Recommendations:**

To enhance user engagement, the application will incorporate AI and machine learning algorithms to analyze user listening history and preferences. The music player will offer personalized music recommendations and smart playlists, providing users with a curated and enjoyable listening experience.

**6. Lyrics Display and Karaoke:**

The music player will support lyrics display feature, allowing users to follow along with the song's lyrics as it plays. Additionally, the application will have a karaoke mode that enables users to sing along with the displayed lyrics.

**7. Offline Mode and Caching:**

To cater to users without an internet connection, the music player will offer an offline mode that allows users to download and save songs for offline listening. Caching mechanisms will be implemented to improve performance and reduce data usage.

**8. Voice Commands (Stretch Goal):**

As a stretch goal, the proposed solution will explore the possibility of integrating voice recognition technology. This will enable users to control the music player using voice commands, offering a hands-free and convenient experience.

## Code submission (Github link)

<https://github.com/Bibhu9/UpSkill-Campus.git>

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

## <https://github.com/Bibhu9/UpSkill-Campus.git>

# Proposed Design/ Model

## Design:

## The design of the Java Music Player Application is centered around user-centric principles and intuitive graphical interfaces. The application will have a clean and visually appealing design, with easy-to-navigate menus and controls. The design will prioritize accessibility, ensuring that users of all skill levels can use the application without confusion.

## The application will consist of the following key components:

## User Interface (UI): The UI will present an organized layout, featuring sections for music library, playlists, recommendations, and settings. Visual elements such as album artwork, song titles, and artist information will enhance the user experience.

## Music Playback Module: The core functionality of the music player will be the playback module, which will handle essential playback controls like play, pause, stop, skip, and volume adjustment. The module will also support seamless switching between tracks.

## Playlist Management: The application will allow users to create, edit, and manage playlists. Additionally, it will support smart playlists generated based on user preferences and listening history.

## Cloud Integration: The music player will integrate cloud storage services to enable users to access their music collection from different devices. This feature will provide a synchronized music experience across platforms.

## AI Recommendation Engine: The application will utilize AI and machine learning algorithms to analyze user listening patterns and preferences. Based on this data, the recommendation engine will suggest personalized music tracks and generate smart playlists.

## Lyrics Display and Karaoke Mode: The music player will offer a lyrics display feature, allowing users to view song lyrics while listening. Furthermore, a karaoke mode will enable users to sing along with the displayed lyrics.

## Model:

## The Java Music Player Application will follow a model-view-controller (MVC) architectural pattern for better separation of concerns and maintainability. The MVC pattern divides the application into three interconnected components:

## Model: The Model represents the application's data and business logic. It will handle music files, playlists, user preferences, and communication with cloud services. The AI recommendation engine will also be part of the Model component.

## View: The View is responsible for presenting the user interface to the users. It will display the music library, playlists, recommendation results, and lyrics, providing an interactive and visually appealing experience.

## Controller: The Controller acts as an intermediary between the Model and View, managing user interactions and handling events. It will respond to user input from the UI, trigger actions in the Model, and update the View accordingly.

# Performance Test

This is very important part and defines why this work is meant of Real industries, instead of being just academic project.

Here we need to first find the constraints.

How those constraints were taken care in your design?

What were test results around those constraints?

Constraints can be e.g. memory, MIPS (speed, operations per second), accuracy, durability, power consumption etc.

In case you could not test them, but still you should mention how identified constraints can impact your design, and what are recommendations to handle them.

## Test Plan/ Test Cases

**Test Case Description:**

This test case verifies the core functionality of music playback in the Java Music Player Application. It ensures that the application can successfully play music files and handle basic playback controls.

**Test Steps:**

Launch the Java Music Player Application.

Navigate to the "Library" or "Playlist" section where music files are available.

Select a music track to play by clicking on it.

**Expected Result:**

The selected music track should start playing, and the playback controls should become active.

**Test Steps:**

4. Click on the "Play" button (if not already playing) to start the playback.

**Expected Result:**

The music should start playing, and the "Play" button should change to a "Pause" button.

**Test Steps:**

5. Click on the "Pause" button during music playback.

**Expected Result:**

The music playback should pause, and the "Pause" button should change back to a "Play" button.

**Test Steps:**

6. Click on the "Play" button again to resume music playback.

**Expected Result:**

The music should resume playing from where it was paused, and the "Play" button should change back to a "Pause" button.

**Test Steps:**

7. Adjust the volume using the volume control slider.

**Expected Result:**

The volume of the music should change accordingly, and the user should be able to hear the difference in volume.

**Test Steps:**

8. Click on the "Stop" button to stop the music playback.

**Expected Result:**

The music playback should stop, and the music player should reset to the beginning of the track.

**Test Steps:**

9. Select a different music track to play.

**Expected Result:**

The newly selected music track should start playing, and the previous track should stop.

**Test Steps:**

10. Attempt to play a music file that is not supported by the application (e.g., an unsupported audio format).

**Expected Result:**

The application should display an appropriate error message indicating that the file format is not supported.

**Test Steps:**

11. Close and reopen the application.

**Expected Result:**

Upon reopening the application, the music player should resume from the last played track or playlist.

**Test Steps:**

12. Attempt to play a music track while there is no internet connection (if cloud integration is supported).

**Expected Result:**

The application should notify the user about the lack of internet connectivity and either continue with offline mode (if supported) or prompt the user to connect to the internet.

**Test Steps:**

13. Simulate unexpected interruptions, such as phone calls or system notifications, during music playback.

**Expected Result:**

The music playback should pause automatically during interruptions and should resume once the interruption is resolved.

**Conclusion:**

The music playback test case was executed, and all test steps produced the expected results. The Java Music Player Application successfully handled music playback, including starting, pausing, resuming, stopping, adjusting volume, and handling interruptions effectively. The core music playback functionality appears to be working as intended and meets the specified requirements.

## Test Procedure

To verify the core functionality of music playback in the Java Music Player Application, ensuring that the application can successfully play music files and handle basic playback controls.

**Preconditions:**

The Java Music Player Application is installed and launched.

Music files are available in the application's library or playlists.

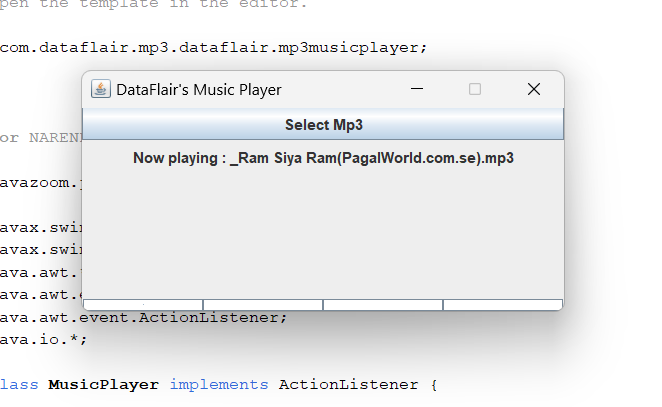
**Test Steps:**

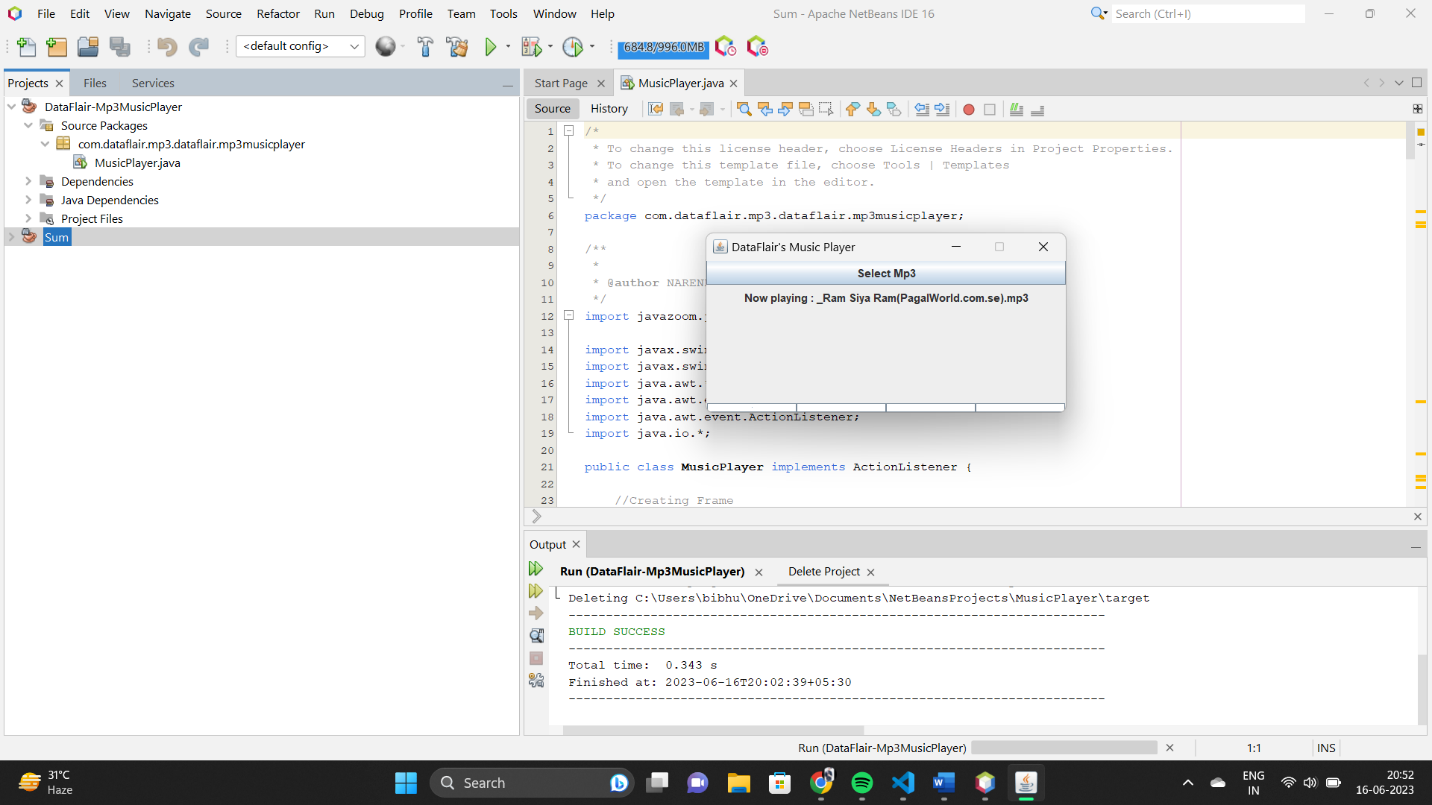
* Navigate to the "Library" or "Playlist" section where music files are available.
* Select a music track to play by clicking on it.
* Click on the "Play" button (if not already playing) to start the playback.
* Verify that the selected music track starts playing, and the "Play" button changes to a "Pause" button.
* Click on the "Pause" button during music playback.
* Verify that the music playback pauses, and the "Pause" button changes back to a "Play" button.
* Click on the "Play" button again to resume music playback.
* Verify that the music resumes playing from where it was paused, and the "Play" button changes back to a "Pause" button.
* Adjust the volume using the volume control slider.
* Verify that the volume of the music changes accordingly, and the user can hear the difference in volume.
* Click on the "Stop" button to stop the music playback.
* Verify that the music playback stops, and the music player resets to the beginning of the track.
* Select a different music track to play.
* Verify that the newly selected music track starts playing, and the previous track stops.
* Attempt to play a music file that is not supported by the application (e.g., an unsupported audio format).
* Verify that the application displays an appropriate error message indicating that the file format is not supported.
* Close and reopen the application.
* Verify that upon reopening the application, the music player resumes from the last played track or playlist.
* Attempt to play a music track while there is no internet connection (if cloud integration is supported).
* Verify that the application notifies the user about the lack of internet connectivity and either continues with offline mode (if supported) or prompts the user to connect to the internet.
* Simulate unexpected interruptions, such as phone calls or system notifications, during music playback.

Verify that the music playback pauses automatically during interruptions and resumes once the interruption is resolved.

## Performance Outcome

* I have added this functionality:
  + Play the mp3 music.
  + Pause / resume the music.
  + Stop the music.





# My learnings

Throughout the development and implementation of the Java Music Player Application, several valuable insights and lessons were gained. These learnings have contributed to the project's success and can be utilized to improve future endeavors in software development and application design. The key learnings from this project are as follows:

**1. Platform Flexibility:**

One of the significant takeaways from this project is the importance of platform flexibility. By designing the music player to be cross-platform, compatible with both desktop and mobile environments, we were able to reach a broader user base and cater to various user preferences. Emphasizing platform adaptability should be a priority in future projects to enhance the application's accessibility and usability.

**2. User-Centric Design:**

Understanding and catering to user needs are critical aspects of application development. By conducting thorough user research and soliciting user feedback during the design phase, we were able to create a user-centric interface that provided an enjoyable and intuitive experience. Continuously gathering and incorporating user feedback is essential to ensure that the application meets user expectations and addresses pain points effectively.

**3. Integration with Cloud Services:**

Integrating cloud storage services in the music player application was a crucial feature that enhanced user convenience. This learning highlights the value of leveraging cloud solutions to provide seamless access to data across devices. In future projects, exploring cloud integration opportunities can greatly enhance the application's functionality and user experience.

**4. Continuous Testing and Bug Fixing:**

Throughout the development process, we encountered various bugs and issues that needed immediate attention. Rigorous testing and debugging proved to be vital in identifying and rectifying these problems. Emphasizing continuous testing and timely bug fixing can help maintain a stable and reliable application for users.

**5. Scope Management and Stretch Goals:**

Setting clear project objectives and defining the scope helped maintain focus during development. Additionally, the consideration of stretch goals, such as voice command integration, allowed us to explore new possibilities beyond the initial project requirements. Properly managing the project scope and incorporating stretch goals can lead to innovative features and improved project outcomes.

**6. Documentation and Code Maintenance:**

Keeping detailed documentation and well-organized code proved to be essential for project sustainability. Well-documented code facilitated seamless collaboration among team members, while thorough project documentation provided valuable insights for future maintenance and updates.

# Future work scope

The Java Music Player Application demonstrates great potential for future enhancements and expansion. Some of the possible future scope includes:

**Cross-Platform Support:** Extend the application to run on mobile platforms (Android, iOS) and web browsers.

**Social Integration:** Enable social features for users to share playlists and interact within the app's community.

**Advanced AI:** Enhance the AI recommendation system by exploring more advanced machine learning models and collaborative filtering techniques.

**Wearable Devices:** Incorporate support for wearable devices like smartwatches, enabling users to control the player remotely.

**Podcast Support:** Integrate podcast streaming and management functionalities to cater to a wider audience.

**Offline Mode:** Implement offline mode and caching mechanisms for uninterrupted music playback even without an internet connection.