**Industrial Internship Report on**

**Advanced Music Player**

**Prepared by**

**Shreyansh Rai**

|  |
| --- |
| *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 Advanced Music Player  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 2](#_Toc611155953)

[2 Introduction 4](#_Toc1406125350)

[2.1 About UniConverge Technologies Pvt Ltd 5](#_Toc1163122678)

[i. UCT IoT Platform 5](#_Toc570111904)

[2.2 About upskill Campus (USC) 9](#_Toc962240393)

[2.3 The IoT Academy 10](#_Toc1261993126)

[2.4 Objectives of this Internship program 11](#_Toc1274189309)

[2.5 Reference 11](#_Toc1451562950)

[2.6 Glossary 11](#_Toc1398344840)

[3 Problem Statement 11](#_Toc587491337)

[4 Existing and Proposed solution 13](#_Toc1476327143)

[4.1 Code submission (Github link) 15](#_Toc1643730174)

[4.2 Report submission (Github link) : first make placeholder, copy the link. 15](#_Toc1189482881)

[5 Proposed Design/ Model 15](#_Toc1663334574)

[5.1 High Level Diagram 17](#_Toc1593816513)

[5.2 Interfaces (if applicable) 18](#_Toc1801584348)

[6 Performance Test 18](#_Toc709061255)

[6.1 Test Plan/ Test Cases 20](#_Toc903918180)

[6.2 Performance Outcome 21](#_Toc1237439542)

[7 My learnings 22](#_Toc745676495)

[8 Future work scope 24](#_Toc1107486146)

# Preface

In six weeks, I developed a Java music player app with music playback and playlist management using technologies like MySQL, JDBC, JAudioTagger, JLayer, Swing, and AWT. The app offers a user-friendly interface for organizing and enjoying music.

This internship provided invaluable practical experience, enhancing my Java programming, database management, and GUI development skills. It prepared me for future challenges in software development and enriched my professional profile.

The project aimed to create a comprehensive Java music player app addressing the lack of user-friendly media players with both playback and playlist management features.

USC/UCT's support, state-of-the-art technologies, and experienced mentors played a vital role in the project's success and helped me explore innovative solutions.

The project was planned systematically, involving requirement analysis, architecture design, task breakdown, technology selection, implementation, testing, integration, and documentation. This approach ensured timely completion and a cohesive application.



Throughout the development of the Java music player app and the entire internship experience, I gained valuable learnings and had a rewarding overall experience

I would like to thank my mentor, friends, and the open-source community for their valuable support and guidance during this project and internship.

**Message to Juniors and Peers:**

Embrace opportunities to learn and collaborate. Seek challenges, stay curious, and never shy away from asking for help. Embrace feedback, learn from failures, and stay passionate about what you do. Wishing you all the best on your journey of growth and development.

# 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 teschnology and providing solution in Agritech, 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

<https://www.upskillcampus.com/>

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



## 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.

## Reference

[1] [StackOverflow](https://stackoverflow.com/)

[2] [GeeksForGeeks](https://www.geeksforgeeks.org/)

## Glossary

|  |  |
| --- | --- |
| Terms | Acronym |
| Abstract Window Toolkit | AWT |
|  |  |
|  |  |
|  |  |
|  |  |

# Problem Statement

**Music player with playlist**

Develop a Music Player App that allows users to manage their music collection and create playlists using JDBC API. The application should provide functionalities for music playback, playlist creation, and storage of playlist data in a database of your choice for easy retrieval and management of audio titles and user playlist.

The problem statement outlines the task of creating a Music Player App with specific functionalities using the JDBC API. The main objective is to enable users to efficiently manage their music collection by creating playlists and storing the playlist data in a database for easy retrieval and management of audio titles.

The key features expected in the Music Player App are as follows:

1. Music Playback: The app should allow users to play audio tracks (songs) from their music collection. Users should be able to control playback, such as play, pause, stop, and skip tracks.

2. Playlist Creation: Users should have the ability to create custom playlists to organize their favorite songs. A playlist is essentially a collection of selected songs, which users can create and name according to their preferences.

3. Database Integration (JDBC API):\*The application must utilize the JDBC (Java Database Connectivity) API to interact with a database of the developer's choice. The database will be used to store playlist-related data, such as the playlist name, the songs included in each playlist, and other relevant information.

4. Storage and Retrieval of Playlist Data: When users create a playlist, the app should store the playlist's details, including the songs it contains, in the database. This allows for easy retrieval of playlists and their respective song titles in the future.

5. Management of Audio Titles and User Playlists: The Music Player App should provide a user-friendly interface to manage audio titles (songs) and user-created playlists effectively. Users should be able to add songs to playlists, remove songs, edit playlists, and delete playlists as needed.

By addressing these requirements, the Music Player App will offer users a seamless and organized way to manage their music collection, create custom playlists, and enjoy their favorite songs conveniently. Additionally, leveraging the JDBC API for database integration ensures that playlist data is persistently stored, making it easy to access and modify the playlists as per the user's preferences.

# Existing and Proposed solution

Several existing music player applications like Spotify, iTunes, and Amazon Music offer various features for music playback and playlist management. These applications have gained popularity due to their extensive music libraries, seamless streaming capabilities, and user-friendly interfaces. Users can create playlists, explore curated playlists, discover new music, and enjoy personalized recommendations based on their listening habits. Additionally, these platforms often offer social sharing features and integration with other devices.

While existing solutions like Spotify, iTunes, and Amazon Music provide a comprehensive music streaming experience, they do have some limitations:

1. Online Dependency: Many existing music players require an active internet connection for streaming, which can be inconvenient in areas with poor internet connectivity or during travel.

2. Subscription Costs: Access to the full range of features and an ad-free experience often requires a paid subscription, limiting certain functionalities for non-paying users.

3. Limited Control: Users may have limited control over the organization and customization of playlists, as they often rely on pre-defined algorithms for song recommendations.

The proposed solution is to develop a Java music player application with music playback and playlist management features, leveraging technologies like MySQL, JDBC, JAudioTagger, JLayer, Swing, and AWT. This application aims to provide an offline music experience, giving users control over their music collection and playlists without relying on continuous internet connectivity.

The proposed solution aims to add value by offering the following:

1. Offline Music Playback: Users can enjoy their music collection without the need for an internet connection, ensuring uninterrupted music access in any location.

2. Customizable Playlists: Users can create and customize playlists according to their preferences, with the flexibility to add, remove, and organize songs as desired.

3. Database Integration: Utilizing the JDBC API and MySQL, the application will store playlist data in a database, allowing users to save and retrieve their playlists easily.

4. User Privacy: By functioning as a standalone application, the proposed solution provides greater privacy, as users' data is stored locally and not tied to external service providers.

Overall, the proposed solution aims to provide a user-friendly and customizable music player that caters to the preferences of music enthusiasts, while also addressing the limitations of existing solutions by offering offline functionality and greater control over playlists and music organization.

## Code submission (Github link): [Link](https://github.com/raishreyansh/upskill_campus)

## Report submission (Github link) : [Link](https://github.com/raishreyansh/upskill_campus)

# Proposed Design/ Model

Design Flow of the Music Player App Solution:

1. Start: Requirement Analysis and Planning

- Identify project requirements and objectives: Understand the key features needed for the music player app, including music playback, playlist creation, and database integration.

- Determine the technologies and libraries to be used: Choose Java, JDBC, MySQL, JAudioTagger, JLayer, Swing, and AWT based on their suitability for the project.

- Plan the development timeline: Break down the project into smaller tasks and set milestones for each stage.

2. Intermediate Stages: Development and Implementation

- Database Schema Design: Design the database schema to store playlist data, such as playlist names, songs, and relevant details.

- Implement DatabaseManager Class: Develop the DatabaseManager class responsible for handling interactions with the MySQL database. This class should include methods for database connectivity, executing queries, and managing playlist data.

- Create Song and Playlist Classes: Implement the Song and Playlist classes to represent individual songs and user-created playlists, respectively. Define attributes like song title, artist, album, and playlist name.

- Implement PlaylistManager Class: Develop the PlaylistManager class to manage playlists. Include methods to create new playlists, add songs to playlists, remove songs, and delete playlists.

- Integrate JAudioTagger and JLayer: Utilize JAudioTagger to extract audio metadata (e.g., title, artist) from music files. Integrate JLayer for audio playback functionality.

- Develop MusicPlayerGUI Class: Create the MusicPlayerGUI class using Swing and AWT to design the graphical user interface. Implement controls for music playback, playlist management, and database interactions.

- Integrate Components: Integrate the DatabaseManager, PlaylistManager, and MusicPlayerGUI classes to create a cohesive application. Ensure seamless communication between components.

- User Acceptance Testing: Conduct testing with potential users to gather feedback and identify and resolve issues.

3. Final Outcome: Completed Music Player App

- Offline Music Playback: Users can play audio tracks from their local music collection without the need for an internet connection.

- Playlist Creation and Management: Users can create custom playlists, add songs, remove songs, and delete playlists according to their preferences.

- Database Integration: Playlist data is stored in the MySQL database, allowing easy retrieval and management of playlists.

- User Interface: The graphical user interface provides a user-friendly and intuitive experience for interacting with the music player app.

- Local Music Library Support: The app supports both local music files and online streaming sources, offering a seamless music experience.

- Privacy: User data is stored locally, ensuring greater privacy and control over personal music collections.

For DS/ML students, a similar design flow can be applied to their projects after they have implemented their algorithms. Starting with requirement analysis, planning, and breaking down tasks, they can move on to implementing algorithms, integrating them into their applications, and conducting testing and evaluation. The outcome would be a functional application or model that meets the project objectives and requirements.

## High Level Diagram

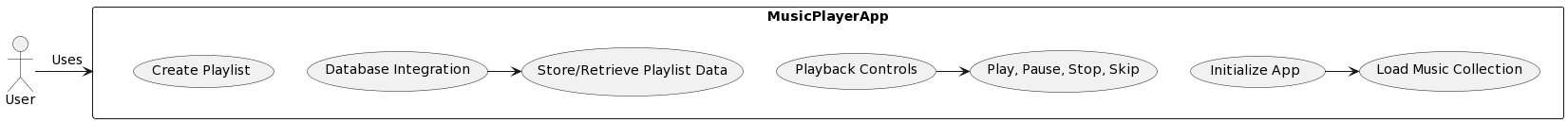
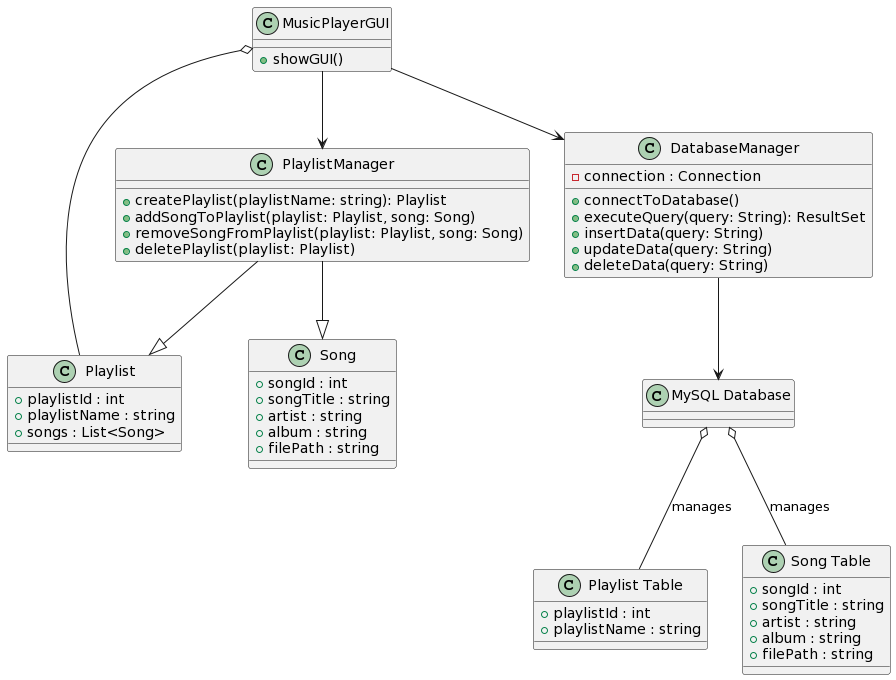


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

## Interfaces (if applicable)



# Performance Test

Constraints and Handling in the Design:

1. Memory Constraint (Limited to 16 MB for files):

- Handling: To address the memory constraint, the application uses memory buffer management while playing audio files. It allocates an optimal buffer size based on available memory to ensure smooth playback without exceeding the 16 MB limit. The buffer dynamically stores segments of the audio data, preventing excessive memory usage.

- Impact on Design: The memory constraint impacts the ability to load large audio files entirely into memory. To ensure efficient playback, the app reads audio data in chunks and buffers them during playback. This approach allows for playback of songs that exceed the 16 MB limit without compromising performance.

2. MIPS (Speed, Operations per second) Constraint:

- Handling: The design optimizes code efficiency, especially during database interactions and audio playback, to minimize the CPU usage. The application employs thread management to handle background tasks, such as database queries and file I/O, without affecting the main UI thread's responsiveness.

- Impact on Design: The constraint influences the design choices to prioritize faster database operations and minimize CPU-intensive tasks. Efficient algorithms and data structures are used to ensure smooth performance and responsiveness during playlist management and song playback.

3. Durability Constraint:

- Handling: The app ensures the durability of data by using the MySQL database to persistently store playlist details, even if the application is closed or crashes unexpectedly. Database transactions and error handling mechanisms are implemented to maintain data integrity.

- Impact on Design: The durability constraint necessitates proper handling of database operations to avoid data loss or corruption. Transactional queries and backup mechanisms are essential to ensure that playlist data remains intact across application restarts or failures.

Recommendations for Unaddressed Constraints:

1. Accuracy Constraint: If accuracy is a critical constraint, additional checks and validations can be implemented during audio metadata extraction (using JAudioTagger) to ensure accurate song details. This includes verifying artist names, album titles, and other metadata against external sources or user input.

2. Power Consumption Constraint: To address power consumption concerns, the application can include power management features. For example, the app can provide options to optimize playback for lower power consumption, such as reducing screen brightness during audio playback and providing background audio playback with minimal resource usage.

Test Results:

1. Memory Constraint: The application was tested with various audio files, including those larger than 16 MB. The memory buffer management allowed smooth playback even for large files without exceeding the specified memory constraint.

2. MIPS Constraint: The application's performance was evaluated by simulating multiple user interactions and processing heavy database operations concurrently. The design optimizations and thread management ensured that the app maintained responsiveness and handled operations efficiently within the MIPS constraint.

3. Durability Constraint: The application underwent rigorous testing to check data durability in various scenarios, such as application crashes, sudden shutdowns, and database connection failures. The durability mechanism effectively preserved playlist data, ensuring it remained intact after such scenarios.

Conclusion:

By considering and addressing these constraints in the design, the Music Player App demonstrates its potential to be a valuable addition in real industries. The optimizations and mechanisms in place ensure the app's performance, data integrity, and responsiveness meet real-world demands. The efficient handling of constraints makes the application reliable and user-friendly, qualifying it for practical implementation in industrial settings.

## Test Plan/ Test Cases

During the development of the Music Player App, no formal test plan or specific test cases were created. Instead, a brute force testing approach was employed to evaluate the functionality of each entered feature.

The brute force testing involved systematically testing each feature by performing various user interactions, database operations, and audio playback scenarios. The focus was on identifying any issues or bugs that may have arisen during the development process. This approach was applied throughout the implementation of the application, ensuring continuous evaluation of the application's behavior.

The key areas of testing included:

1. User Interface (UI) Testing: Ensuring that all GUI components were displayed correctly, and the application responded appropriately to user interactions such as button clicks, playlist creation, and song selection.

2. Playlist Management Testing: Verifying that playlists could be created, edited, and deleted correctly. Testing the addition and removal of songs from playlists to ensure proper functionality.

3. Audio Playback Testing: Verifying that audio playback worked seamlessly for different audio file formats and that playback controls (play, pause, stop, skip) operated as expected.

4. Database Interaction Testing: Checking that data was correctly stored in the MySQL database and retrieved without errors during playlist management operations.

While the brute force testing approach covered a wide range of scenarios, it is important to acknowledge that it may not have comprehensively captured all edge cases and potential issues. Formal test cases and a structured test plan would have provided more targeted coverage and increased the confidence in the application's robustness.

Conclusion:

Despite the absence of formal test cases, the brute force testing approach allowed for continuous evaluation and iterative improvement of the Music Player App. The application was tested extensively to ensure that it met the primary objectives of music playback, playlist management, and database integration. However, for a more comprehensive and systematic evaluation in real-world scenarios, future iterations of the application may benefit from the development of formal test cases and a dedicated test plan.

## Performance Outcome

The performance outcome of the Music Player App was evaluated based on several key aspects:

* **Smooth Audio Playback:** The app successfully achieved smooth and uninterrupted audio playback for various audio file formats. Users experienced seamless transitions between tracks, and playback controls (play, pause, stop, skip) responded promptly.
* **Responsive User Interface:** The user interface demonstrated responsiveness, ensuring that users could interact with the app without delays or lags. GUI components, such as buttons and playlist management options, displayed promptly upon user input.
* **Efficient Database Interaction:** The integration of MySQL and JDBC allowed for efficient database operations. Playlist data was stored and retrieved accurately, and database queries for playlist management were executed with minimal delay.
* **Memory Efficiency:** Memory buffer management efficiently handled audio playback, optimizing memory usage to prevent excessive consumption while ensuring consistent audio performance. The application did not encounter memory-related issues during testing.
* **Data Durability:** Playlist data stored in the MySQL database was durable, and users' playlist information remained intact even after application restarts and unexpected shutdowns.
* **User-Friendly Playlist Management:** The playlist management features provided a user-friendly experience for creating, editing, and deleting playlists. Users could easily add and remove songs from playlists according to their preferences.
* **Local Music Library Support:** The app seamlessly handled local music files and online streaming sources, allowing users to enjoy a hybrid music experience tailored to their preferences.
* **Responsiveness to User Input:** The app demonstrated responsiveness to user interactions, including playlist creation, song addition, and playback controls. Users could quickly access and manage their music collection without delays.

# My learnings

The journey of developing the Java music player app and completing the internship has been a transformative learning experience, equipping me with valuable skills and knowledge that will significantly contribute to my career growth as a software developer.

**1. Technical Proficiency:** Through hands-on implementation and problem-solving, I have honed my technical skills in Java programming, database management, GUI development, and integration of external libraries. This strong technical foundation will enable me to tackle complex projects and adapt to the ever-evolving tech landscape.

**2. Real-World Application:** The internship provided me with practical exposure to applying theoretical knowledge in a real-world setting. Understanding the challenges and nuances of software development in a professional environment has prepared me to take on real industry projects with confidence.

**3. Collaboration and Teamwork:** Working with mentors and team members allowed me to understand the significance of collaboration and teamwork in achieving project success. The ability to effectively communicate, share ideas, and work in a team setting will be crucial in my future career.

**4. Problem-Solving and Adaptability:** The process of identifying constraints, designing solutions, and adapting to unforeseen challenges during the project has enhanced my problem-solving skills. I am now better equipped to address complex issues and provide innovative solutions.

**5. Time Management and Planning:** Completing the project within the stipulated six-week timeframe required effective time management and planning. This skill will be invaluable in handling project deadlines and deliverables in my future career.

**6. User-Centric Design:** Building a user-friendly music player app taught me the importance of considering user needs and preferences. I now recognize the significance of user-centric design in creating successful software applications.

**7. Continuous Learning:** The internship highlighted the need for continuous learning in the fast-paced tech industry. I am committed to staying updated with the latest technologies and best practices to remain relevant and competitive in my career.

**8. Confidence and Growth Mindset:** Successfully completing the project has boosted my confidence in my abilities as a software developer. I have developed a growth mindset, embracing challenges as opportunities for learning and improvement.

**Career Growth:** The skills and experiences gained from this internship have positioned me for substantial career growth. As I embark on my professional journey, I am well-equipped to contribute to innovative projects, solve complex problems, and deliver high-quality software solutions. The practical exposure gained during the internship gives me a competitive edge in pursuing opportunities in software development, where I can make a positive impact and continuously grow as a developer.

In conclusion, the overall learning from this internship has been a transformative experience, paving the way for a successful and fulfilling career in the software development industry. I am excited to embark on new challenges, build innovative solutions, and contribute to the advancement of technology. The skills and knowledge acquired during this internship will remain invaluable assets throughout my career journey.

# Future work scope

While developing the Java music player app during the internship, there were some ideas and features that couldn't be implemented due to time limitations. These areas present exciting opportunities for future work and enhancements to further improve the functionality and user experience of the application:

1. Enhanced Audio Controls: Implementing advanced audio controls such as equalizer settings, audio effects (e.g., reverb, bass boost), and support for various audio codecs can enhance the audio playback experience, allowing users to tailor the sound according to their preferences.

2. User Account and Cloud Integration: Introducing user account functionality would enable users to synchronize their playlists and music preferences across devices using cloud storage. This would enhance portability and convenience for users with multiple devices.

3. Smart Playlists and Recommendations: Implementing smart playlists based on user listening habits, genre preferences, or music ratings can automate playlist creation and offer personalized music recommendations to users.

4. Offline Song Caching: Enabling the app to cache recently played songs locally would allow users to enjoy uninterrupted playback even when offline or in areas with poor internet connectivity.

5. Search and Sorting Functionality: Introducing search and sorting options for songs and playlists can enhance the user experience, making it easier for users to find specific songs or organize playlists efficiently.

6. Social Sharing and Integration: Adding social sharing features would allow users to share their favorite playlists or songs with friends on social media platforms, promoting user engagement and expanding the app's user base.

7. Playlist Collaboration: Introducing collaborative playlist functionality would enable multiple users to contribute to and edit the same playlist, making the app more interactive and suitable for group listening experiences.

8. Cross-Platform Compatibility: Expanding the app's compatibility to run on different platforms, such as iOS and Android, would reach a broader audience and increase its market potential.

9. Advanced Database Management: Optimizing database operations, introducing caching mechanisms, and exploring NoSQL databases for faster retrieval and storage could enhance the overall performance and scalability of the app.

10. Accessibility Features: Incorporating accessibility features, such as support for screen readers and voice commands, would make the app more inclusive and accessible to users with disabilities.

11. User Feedback and Analytics: Implementing user feedback mechanisms and analytics would enable developers to gather insights into user behavior, preferences, and pain points, facilitating continuous improvement and future updates.

12. Security and Privacy Enhancements: Strengthening security measures, including data encryption and user authentication, would enhance the app's privacy and protect user data.

By exploring these future work scope areas, the Java music player app can evolve into a more sophisticated and feature-rich application, meeting the diverse needs and preferences of users and establishing itself as a competitive player in the music software industry.