Industrial Internship Report on

"Music Player with Playlist"

Prepared by

Joel Abhishek

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. The project I chose to undertake in my Core Java Internship is **Advanced Project – Music Player with Playlist**. I had to develop a music playing application in Java that incorporates **JDBC API** for database connectivity. The application will allow users to import their music libraries and the user will be able to select one to play songs. The user can create playlists of songs from any library. The user can view all the songs in a particular library or playlist. The application also provides standard functionalities for a music player such as a pause/play button, buttons to go to the next and previous song, a shuffle and autoplay option. Another important feature is that all of the song's available metadata such as title, artist, album etc. will be displayed by the application. The application uses a database with many tables to store song metadata, library and playlist information and the search functionality relies on this database to return any results. This internship gave me a very good opportunity to get exposure to Industrial problems and design and implement a solution for the problem statement. It also gave me a chance to learn a lot about Java and about various libraries and how to implement them into a project to enhance its capabilities. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

Summary of every week's work

Week 1: In the first week I got to know more about Upskill Campus and UniConverge Technologies. I got to learn about UCT's domain- Digital Transformation, the products they offer relating to Internet of Things, Machine Learning, Cloud Computing and long range communication protocols such as 5G and LoRaWAN, and also some technologies they employ such as Predictive Maintenance.

During this week I also brushed up my knowledge of Java so that I can start working on my project. I also selected my project- **Music Player with Playlist** after reading all the problem statements provided by Upskill Campus.

Week 2: I spent this analyzing the requirements of my project and the expected functionalities. I searched online the best way to implement UI in Java and how database connectivity can be achieved. I found JavaFX to be the latest and most popular framework for designing user interfaces in Java. I spent some time learning all of it's features and how it can be used. JavaFX has many features such as- Platform Independence, FXML, Hardware Accelerated Graphics, many built-in UI Controls and CSS Styling. The main reason I selected JavaFX to create the frontend is because it has a 'MediaPlayer' functionality which will be the core feature of my project.

After learning more about JavaFX I created a basic application which could play a music file on my disk after pressing a button. Since JavaFX libraries are available as Java Modules, I also got to know more about Modules in Java.

Week 3: This week I researched about databases and how to implement a database that can work without a client-server connection. I learned about embedded database where an application will store a database for its use on the disk. Embedded databases are small and easy to use. I found that SQLite is the most popular embedded database which is used in millions of applications on every platform. I decided to use SQLite as the database since this application does not need complex database functionalities.

I also made more progress on the music player's frontend, I was able to show a list of songs from a folder on my disk and play each of them. I was also able to store the song's data into the database table. At the end of this week I had a clear outlook on how the rest of the project will be designed.

Week 4: I continued to make improvements in my application. I created a table to display song's title and album, I added a progress bar that allows the user to seek to a particular time in the song's duration. I also made a way to import multiple libraries and select any one of them to view all the songs contained in the selected library. I also added a shuffle function.

I was able to store all of a song's metadata into the database using mp3agic, a Java library that retrieves metadata from mp3 files.

Week 5: In this week I added multiple functionalities to my project. The application now displays all the available metadata on the left side including lyrics. I made an option to create custom tags for a song which can be searched later. There is a stop playback option to stop music playing and reset the player. I added a volume slider to control the volume and an auto play option to continue playing music when a song ends. I also formatted the progress bar to display the time in M:SS format where 'M' is minutes and 'SS' is seconds.

I also added full playlist functionalities. Playlists can be created, deleted. Songs can be added and removed from a playlist and songs can also be reordered within a playlist. Songs can be added to removed by right clicked on a song in the table and selecting an option in the context menu.

I also made a search function that searches the database for songs by the title, artist, album, genre or custom tags. The search uses the SQL 'LIKE' operator so it can search for patterns and does not need exact search text. I also implemented various Dialogs to display messages to the user or take their input.

Week 6: This week I made final adjustments to my project. I fixed any issues and bugs. I added a Image Viewer to load the album art of any song. I added maven to the project so that I can easily compile a fat Jar of the application so I can run it anywhere. It will also help anyone seeing my project on GitHub to easily run and compile it themselves.

I also implemented a way to create a folder on the user's home directory when the application is opened. Inside this folder the app will create the database file if it does not exist and create the database schema. This way the application can be run by itself on any windows computer. I used the Java logging Class to create logs in a file if any exceptions are thrown in my project's Class that connects to the database and performs operations with the database.

Need of relevant Internship in career development

I believe Internships are very important for a person's career. In my opinion colleges by themselves do not provide the practical knowledge that is needed by the industries. Companies will not want to hire freshers who do not have the necessary skills to perform the tasks at their company.

An internship is a good way to learn the necessary skills and get hands on experience with industrial projects and gain some practical knowledge. Internships allow students to apply the theoretical knowledge from classrooms in their work. The exposure to industry practices and technologies helps bridge the gap between academia and the professional world, enhancing their technical skills and problem-solving abilities. This is why Internships can greatly increase the chances of getting hired and so they are important for career development.

Brief about the project

The project that I have chosen is **Music Player with Playlist**. A music playing application is to be developed in the Java language that makes use of Java's JDBC API for connecting to any database to store information such as a song's metadata or playlist information. Moreover it should provide a user friendly interface for playing music, viewing the music library and creating, deleting or managing playlists. The key features as described by the project list are:

- Music Playback: Allow users to play, pause, stop, and skip tracks.
- Music Library Management: Utilize JDBC to store and retrieve song information from a database, including metadata like artist, album, and duration. Import and scan music files.
- Playlist Creation and Management: Enable users to create, modify, and delete playlists. Utilize JDBC to store playlists in the database and provide options to add, remove, and reorder songs within playlists.
- Advanced Search and Filtering: Implement search and filtering functionalities based on artist, genre, release year, or custom tags, using efficient SQL queries for database retrieval.
- Error Handling and Logging: Implement robust error handling mechanisms to handle exceptions, database connectivity issues, and log relevant information for troubleshooting.
- User-Friendly Interface: Design an intuitive graphical user interface using advanced Java UI frameworks for easy navigation and interactive controls.

The most important aspect of this project will be the database functionality as most of the required features will not work without a database.

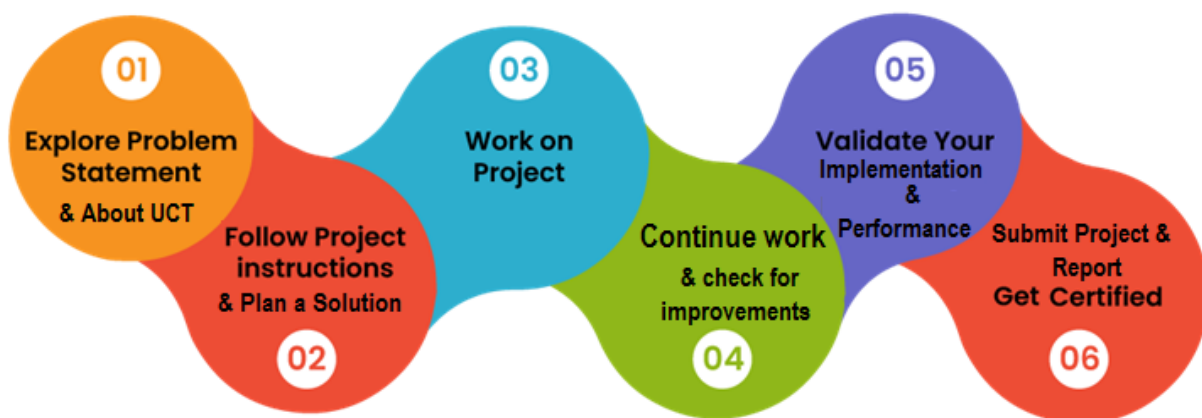## Opportunity given by USC/UCT

Upskill Campus and UniConverge Technologies offer many Internship programs aimed at providing students with the necessary skills to meet industry expectations. Internships are offered in domains like Internet of Things, Machine learning and data science, Python, Core Java. All these are domains which UniConverge Technologies is working in. UCT is providing a valuable opportunity to receive hands on experience with industry projects relating to these domains.

The Internship in Core Java is an excellent opportunity to get practical knowledge in developing a industry level project. As Java is widely used in enterprise software the relevant experience gained throughout this internship while developing a Java application can be a huge boost to the employability of any candidate. This is why I chose the Core Java Internship.

How the Program was planned

The Upskill Campus Internships have a 6 week duration. During this time the interns are expected to learn by themselves with the provided material in the Upskill Campus portal and work on any one project in the given project list. At the end of every week a report has to be submitted that contains all the work done during that week. This report is then checked and reviewed by Upskill Campus. Every two weeks there is a quiz relating to the Java language that has to be answered by the interns. At the end of the internship a final project report has to be made and submitted on GitHub along with the code for the project.

The diagram below shows the structure of the internship program.

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

Your Learnings and overall experience

I learned many things during the course of this Internship relating to Java and app development. I learned how to use JavaFX a modern Java library for making user interfaces, and other libraries like mp3agic for metadata and sqlite for database creation. I learned how to incorporate external libraries into my project in order to provide extra functionality. After many hours of tweaking I managed to create a good UI for the application that is easy to use and understand. This project taught me a lot about good UI design. I also learned how to connect to a database and use SQL queries to efficiently store and retrieve information from a database. I also learned how to use a MVC design where the database operations, the user interface and a controller connecting both are kept separate so modifying code in one doesn't affect the other.

In general I learned a lot about Java programming and Object Oriented design and gained practical experience in them. I learned how to read documentation of any library so that I can better understand how to implement them into my project. I also learned many problem solving methods and how to use the debugger so that my application can work without causing any major problems.

Overall it was a great experience to take part in this internship. I will not forget the skills that I learned during these 6 weeks.

I would like to thank Upskill Campus and UniConverge Technologies for providing this internship opportunity. I also thank my friends and family for supporting me during this internship. I also thank my friend Chris Daniel who is enrolled in the Python Internship for all his advice and help.

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 (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
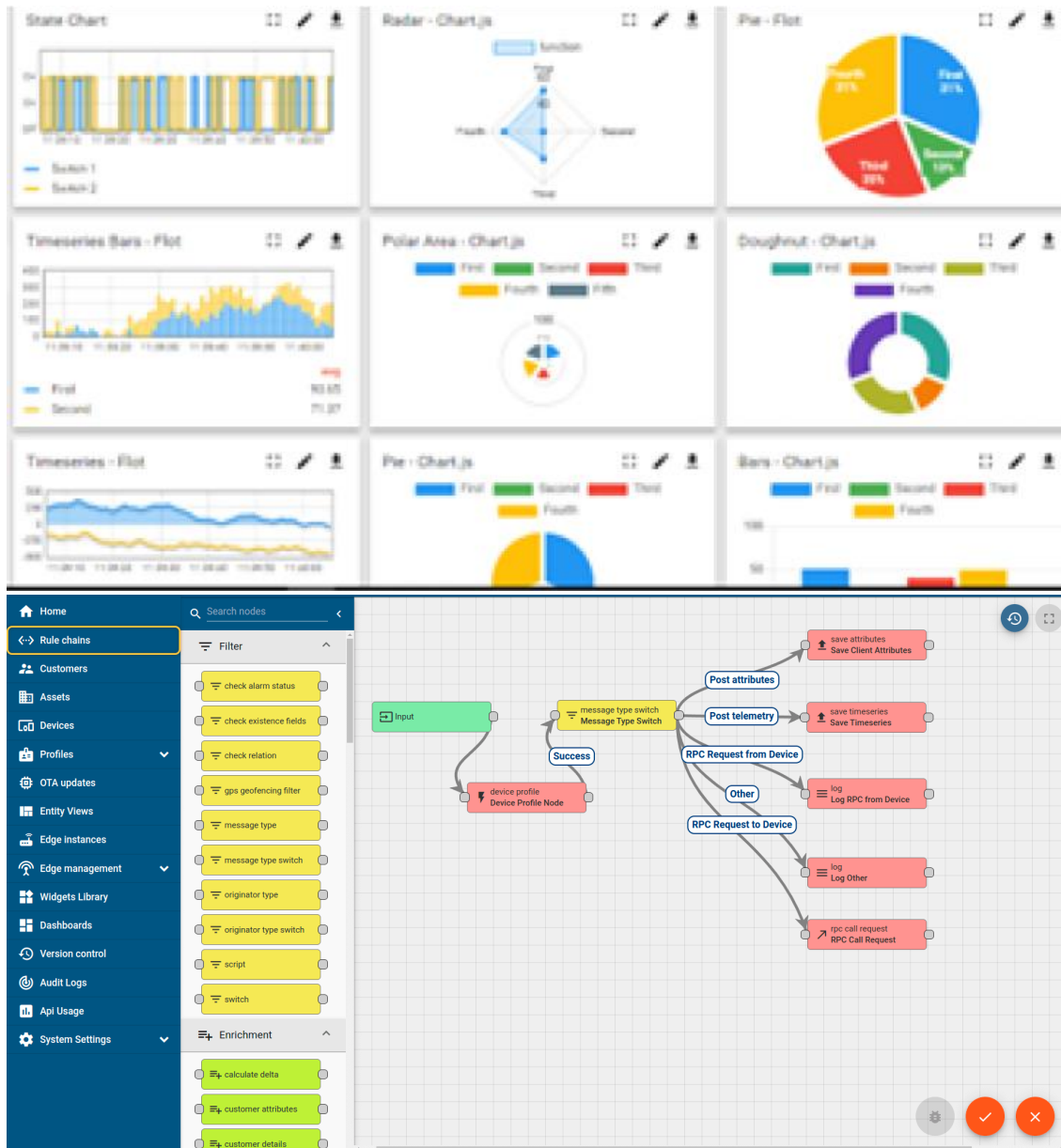
ii. **Smart Factory Platform (** FACTORY WATCH **)**

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.
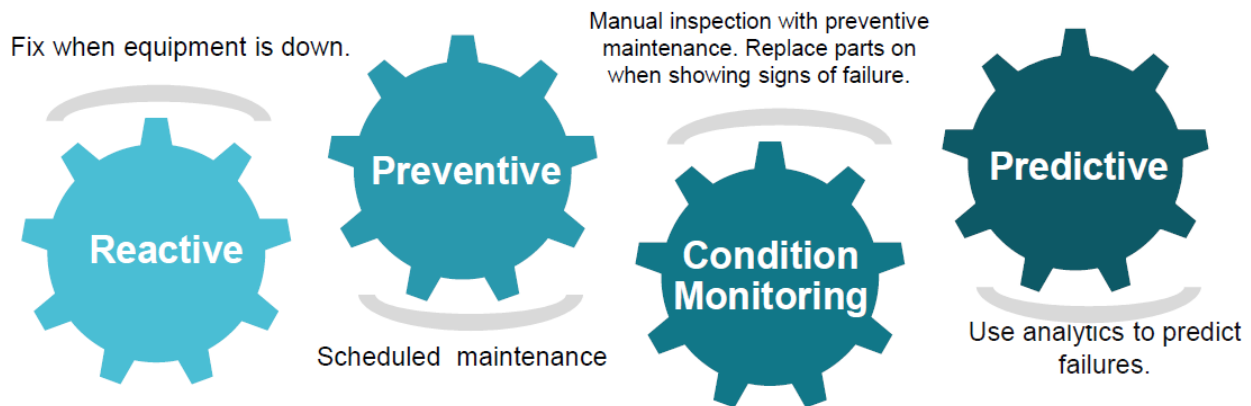
2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.
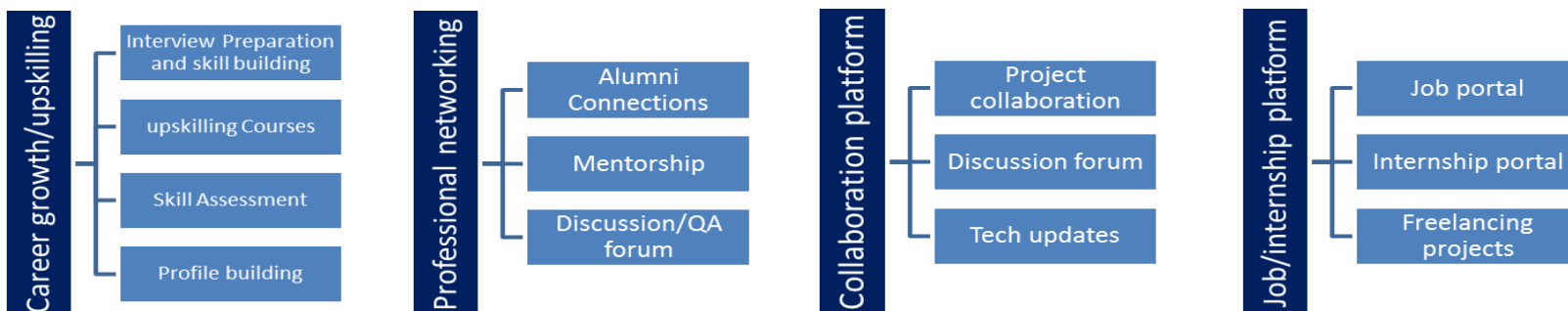
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.
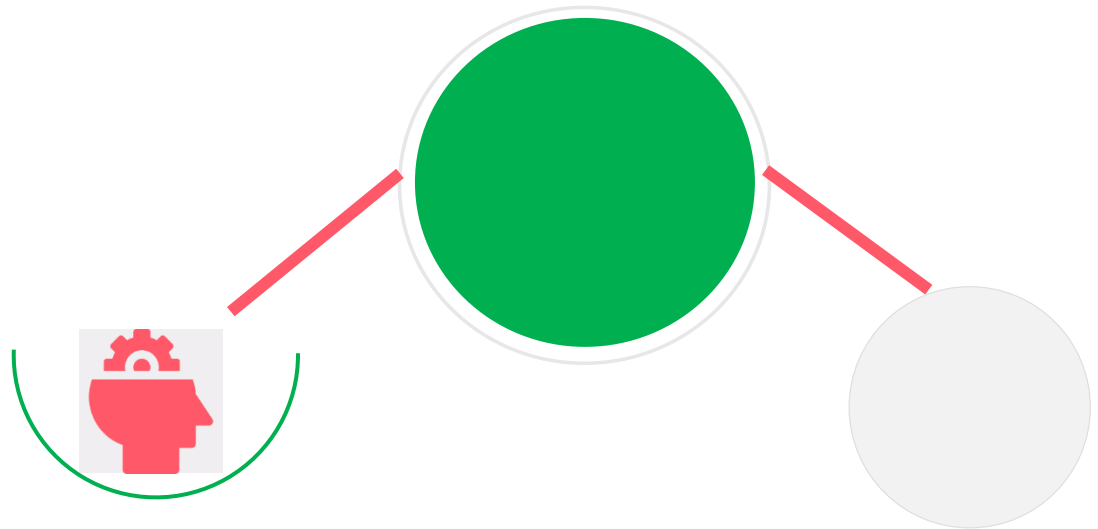
Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

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

https://www.upskillcampus.com/

Career growth/upskilling
- Interview Preparation and skill building
- upskilling Courses
- Skill Assessment
- Profile building

Professional networking
- Alumni Connections
- Mentorship
- Discussion/QA forum

Collaboration platform
- Project collaboration
- Discussion forum
- Tech updates

Job/internship platform
- Job portal
- Internship portal
- Freelancing projects

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. https://openjfx.io/
2. https://github.com/mpatric/mp3agic
3. https://www.sqlite.org/index.html
4. https://maven.apache.org/

2.6 Glossary

Terms	Explanation
JDBC	Stands for Java Database Connectivity. It is an API that allows to connect to any database with Java code provided a suitable driver for JDBC is available.
Metadata	It is information about a piece of software. In this context it is the attributes relating to a music file like the title, artist, genre, duration etc.
SQL	Structured Query Language, is a domain-specific language used in programming and designed for managing data held in a relational database management system
API	An **API** is a way for two or more computer programs to communicate with each other
GUI	A graphical user interface (GUI) is a digital interface in which a user interacts with graphical components such as icons, buttons, and menus.
MVC	MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). This pattern helps to achieve separation of concerns.

3   Problem Statement

The problem statement of my project is to create a music player application. The music player should have a good user interface and provide features that make use of JDBC to connect to a database. Some of these features are Libraries, Playlists management, Searching, Metadata management. Moreover there should be all the standard music playback functions like play/pause, previous song, next song, autoplay and shuffle. The required functionalities are mentioned below.

1. **Music Playback:** Allow users to play, pause, stop, and skip tracks, with advanced audio capabilities for uninterrupted playback.

   I implemented standard music playback functionality by using various JavaFX controls specifically the Button, CheckBox and Slider. I used 3 buttons for playing/pausing the music, going to the next song and previous song. I used two checkboxes, one for a shuffle function where a random song from the selected library or playlist can be played and another for autoplay where after one song has ended another will play automatically.
   I used one slider for volume control, users can set any volume between 0-100.
   Another slider was used as a progress bar for the song that is playing, the progress bar will automatically move during playback and the user can go forward or back in the song by moving the slider.

2. **Music Library Management:** Utilize JDBC to store and retrieve song information from a database, including metadata like artist, album, and duration. Implement features for importing and scanning music files.

   I used SQLite, a popular embedded database for my application to store and retrieve information, being an embedded database there is no need for a client-server connection. As a music player is a stand alone app it makes sense to go with an embedded database. Since SQLite is lightweight and simple to use I decided to implement it.
   I downloaded a driver for JDBC that allows me to manage a SQLite database with JDBC API. My project follows the MVC design pattern so the logic for database operations is mostly contained within a single class and is separate from the UI controls. I used various JDBC APIs like DriverManager, Connection, PreparedStatement, Statement, executeUpdate(), ResultSet and some others. PreparedStatements are the most important and commonly used feature as they allow me to write a SQL query and later input certain values into the query so that the queries work dynamically with provided data.
   I created many tables in my database to store song metadata, library and playlist information and even custom tags.

3. **Playlist Creation and Management:** Enable users to create, modify, and delete playlists. Utilize JDBC to store playlists in the database and provide options to add, remove, and reorder songs within playlists.

   I made a MenuBar where an option is there to create and delete playlists. A Dialog will appear where the user can enter a playlist name or a dialog with a combo box to select the playlist to delete. To add and remove songs a Context Menu was added to the music table where all the songs are listed. So if a user right clicks on a song they can add or remove it from a playlist by selecting the popup options. All of these functions have their database logic in the class Connector where database operations are done. Users can also reorder songs in a playlist by entering a new order number in a textfield.

4. **Advanced Search and Filtering:** Implement search and filtering functionalities based on artist, genre, release year, or custom tags, using efficient SQL queries for database retrieval.

   In SQL the LIKE operator is used to match patterns. I used it in combination with the % operator which matches any characters. This way the user can search for a title, artist, genre, or custom tags by entering a partial search string. For example searching "fav" will return all songs with tag "favourites".

5. **Error Handling and Logging:** Implement robust error handling mechanisms to handle exceptions, database connectivity issues, and log relevant information for troubleshooting.

   In my class 'Connector' I used the Java logging class to log any exceptions that are thrown while accessing the database into a log file. This file is automatically created if it does not exist.

6. **User-Friendly Interface:** Design an intuitive graphical user interface using advanced Java UI frameworks for easy navigation and interactive controls.

   I used JavaFX to create a simple and intuitive graphical user interface that anyone can understand. All the buttons and options have some text that identifies them and even at first glance it is possible to understand their purpose.

4 Existing and Proposed solution

Since this project is being developed in Java for the Core Java internship course I decided to look for music player applications that were made using Java. I found two projects on GitHub that use JavaFX like my project.

The first one is called "music-player" by connoryork. It comes with features for playing, pausing, rewinding, skipping songs, volume changing and playing from playlists. However it does not seem to have any database functionality or viewing the metadata. It's UI and functionality is very basic. It also does not have any executable jar for easy usage.

Another project is called "jMusic" by jihedkdiss, it has a more modern looking UI with features for library selection, playing/pausing, going to next and previous songs and adjusting the volume. But it also does not have any database functionality so only a single music library is stored in memory at a time and it also does not show any metadata.

The limitations of these projects are that they only provide very basic functionality of a music player. Without a database they cannot provide proper playlist functionality and metadata management. The UI is not appealing and there is no easy way to download and use the applications.

My proposed solution is to create a music playing application that provides a good user interface, many audio playback functions and a database for various data related functionalities. The key components of the proposed solution are-

- Database- A robust SQLite database that facilitates the creation of playlists and addition/removal of songs from a playlist. The database can also store metadata and multiple music libraries. This way more advanced functionality can be added to the project, even searching through the database can be possible.
- User Interface- UI built with JavaFX that is simple and easy to use but still provides a good look and feel to the application. The UI must be easily understandable to any user.
- Audio Playback- Include all standard audio playing functions such as play/pause, seek, next and previous song, shuffle and auto play. The JavaFX Media and MediaPlayer Classes can be used to play audio.

Looking beyond Java, there are many popular music playing applications on every platform, such as "Windows Media Player Legacy", "Potplayer", "VLC" and some music streaming apps like "Spotify" and "iTunes". These applications are made and supported by large corporations or are large open source projects supported by many people as is the case with VLC. Even though my

project has only been in development for 6 weeks I can say it has some value added features such as:

- Open Source- This project will be available as an open source project on GitHub where anyone can access the source code. The advantage of this is you do not need to rely on proprietary software like the windows player, iTunes etc. where you may need to agree with the terms and conditions of software companies and allow them to gather your data. Open source software also provides more security and privacy as the code can be analyzed by anyone for any vulnerabilities.
  Finally, since the project is open source it is available for anyone to use for free without any advertisements.
- Lightweight- The size of the executable jar for the music player is only 25 MB making it portable.
- Cross-Platform- Since Java is a platform independent language the music player can be compiled for most desktop operating systems like Windows, Linux and MacOS by using the respective platform specific JavaFX libraries.
- Comparing with Spotify, my music player can display multiple libraries of local music files whereas Spotify can only display local files in a single library. Spotify also cannot show the metadata of local files but my application can display metadata and lyrics.

4.1 Code submission (Github link):

https://github.com/Joel0423/upskill_campus/

Link to each .java file:

1) https://github.com/Joel0423/upskill_campus/blob/master/src/main/java/MusicPlayerFX/ApplicationController.java
2) https://github.com/Joel0423/upskill_campus/blob/master/src/main/java/MusicPlayerFX/Connector.java
3) https://github.com/Joel0423/upskill_campus/blob/master/src/main/java/MusicPlayerFX/Launcher.java
4) https://github.com/Joel0423/upskill_campus/blob/master/src/main/java/MusicPlayerFX/Libraries.java
5) https://github.com/Joel0423/upskill_campus/blob/master/src/main/java/MusicPlayerFX/Main.java
6) https://github.com/Joel0423/upskill_campus/blob/master/src/main/java/MusicPlayerFX/Playlist.java

7) https://github.com/Joel0423/upskill_campus/blob/master/src/main/java/MusicPlayerFX/SearchCategory.java

8) https://github.com/Joel0423/upskill_campus/blob/master/src/main/java/MusicPlayerFX/Song.java

4.2   Report submission (Github link) :

https://github.com/Joel0423/upskill_campus/blob/master/Internship_Report_Core_Java_JOEL_ABHISHEK_USC_UCT.pdf

5 Proposed Design/ Model

The diagram below is a proposed design of the music player, I created the UI following this layout. But the final outcome has a few differences that were made for accommodating all the required functionality.
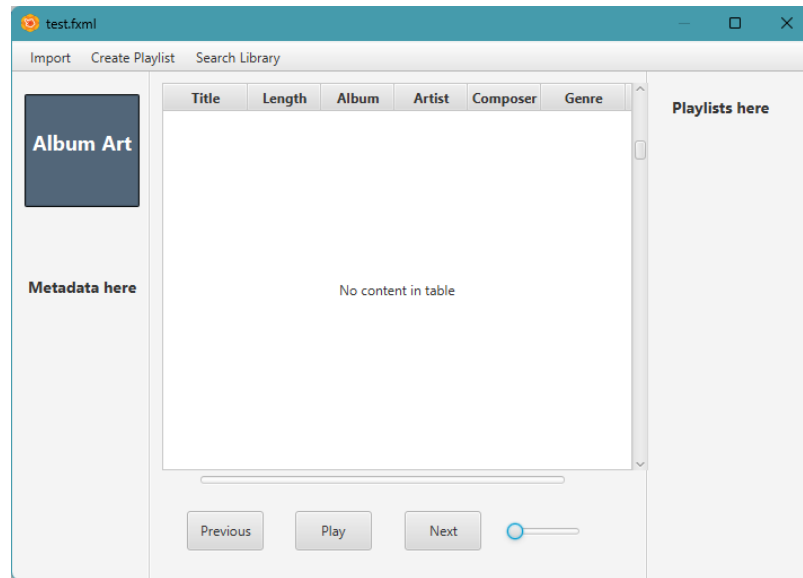
Figure 1: Proposed Design

Next is the intermediary stage
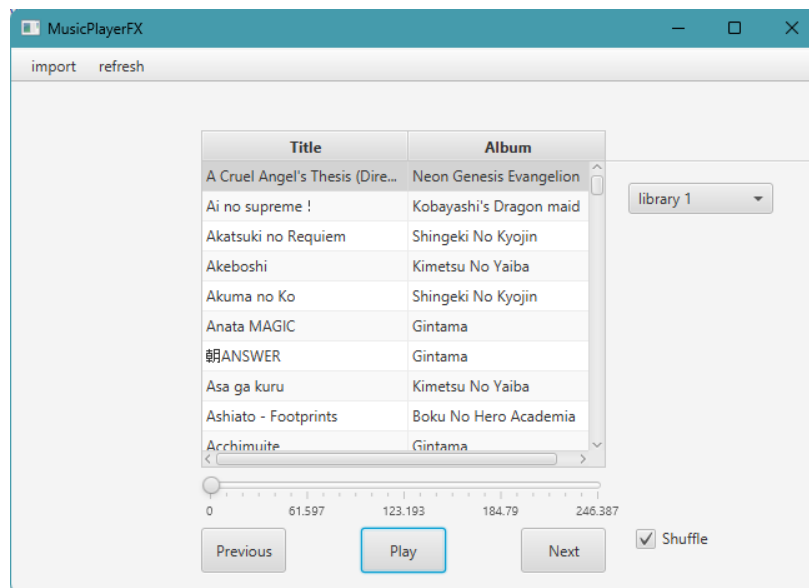
Figure 2: Intermediate Stage

This is the final design of the music player:
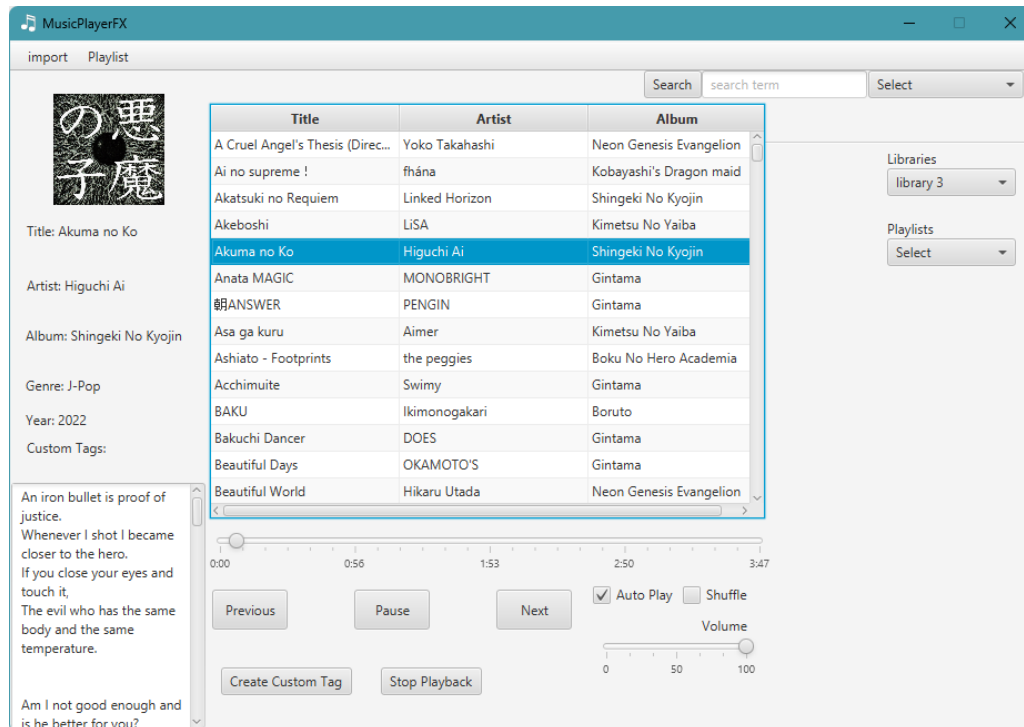
Figure 3: Final Design

The User Interface was designed in such a way so that the user can easily understand the GUI components and their function at the top an import and playlist options are shown so the user knows those options are important. On the right side there are combo boxes for playlists, libraries and a search box. The text on the combo boxes are set to "Select" so the user knows they have to select some option there. The music playback controls are also clearly labeled so the user can understand their purpose.

5.1 High Level Diagram
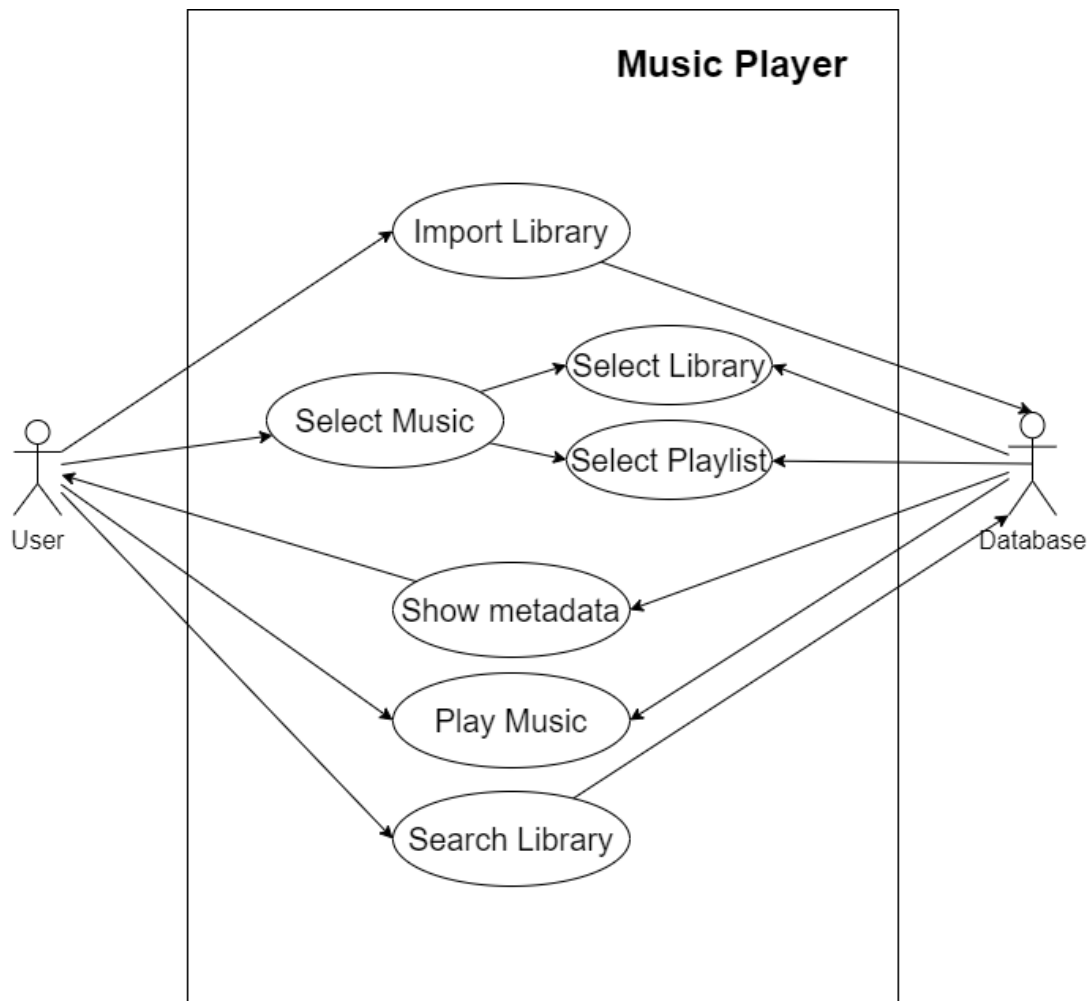
Figure 4: High Level Use Case Diagram

A use case diagram showing the important functionalities of the music player app is shown above. There are two actors, the user and the database.

5.2 Low Level Diagram
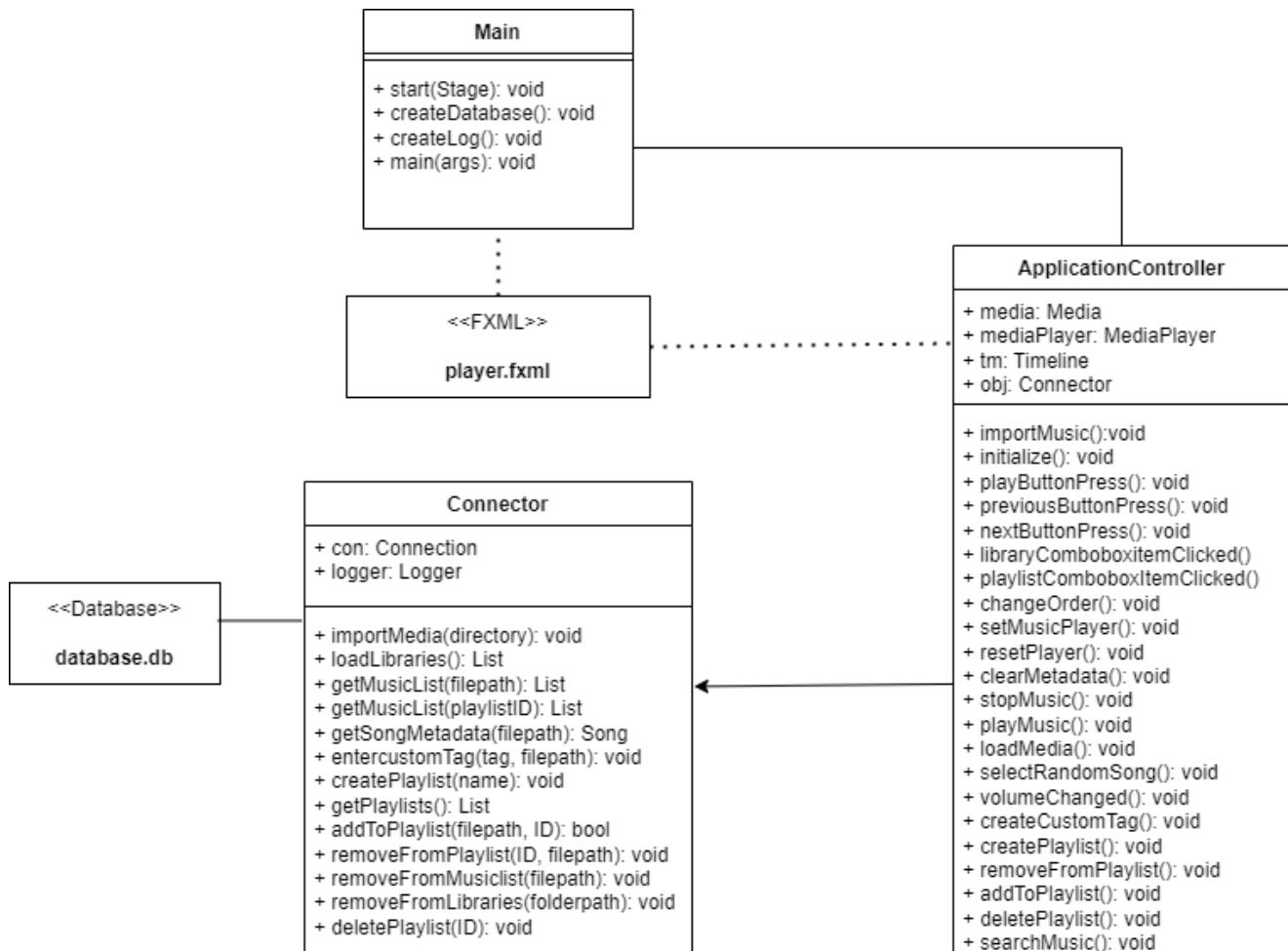
Figure 5: Class Diagram

The important Classes of this application are shown in the above class diagram.

5.3 Interfaces
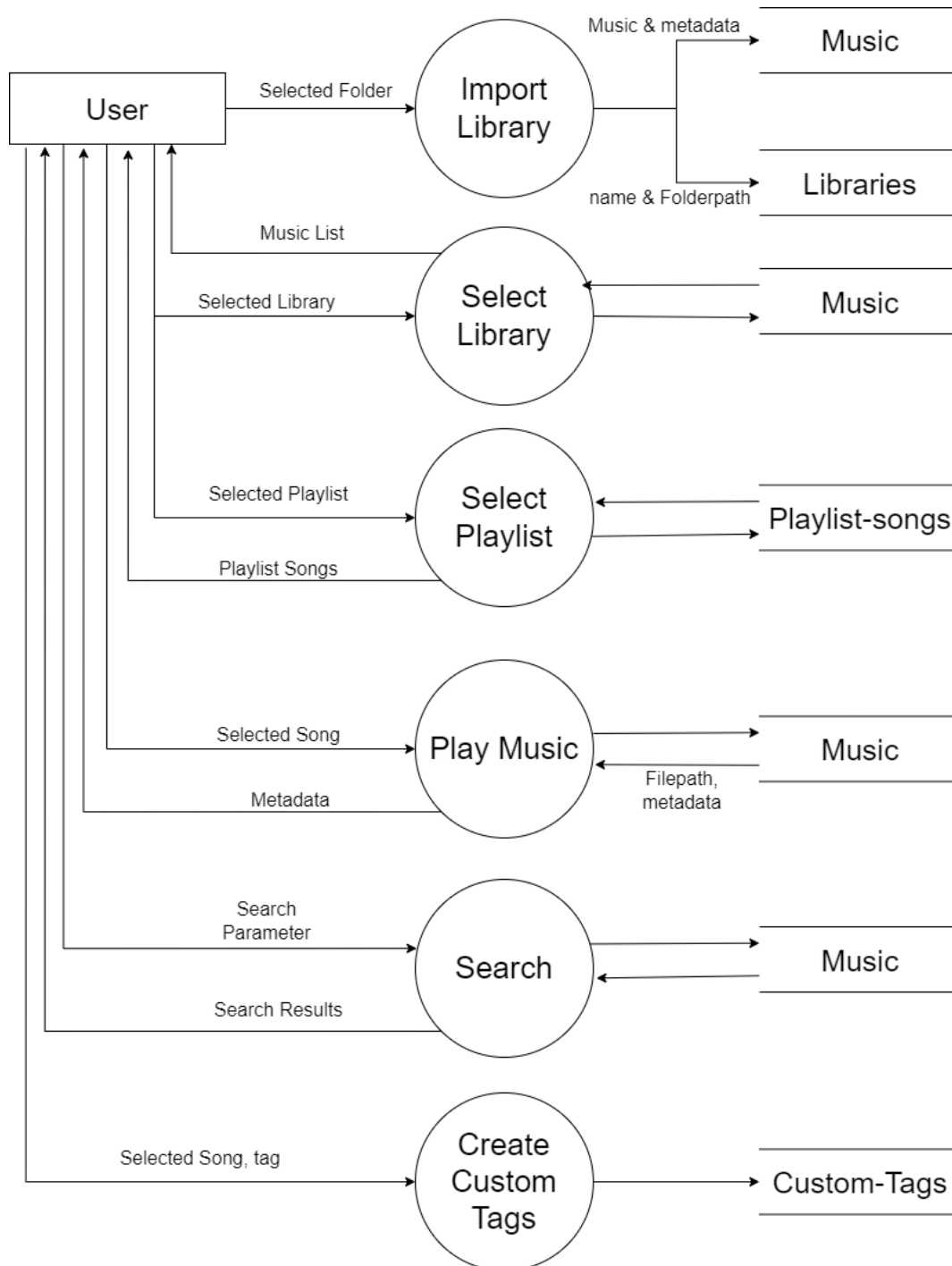
Figure 6: Level 1 Data Flow Diagram
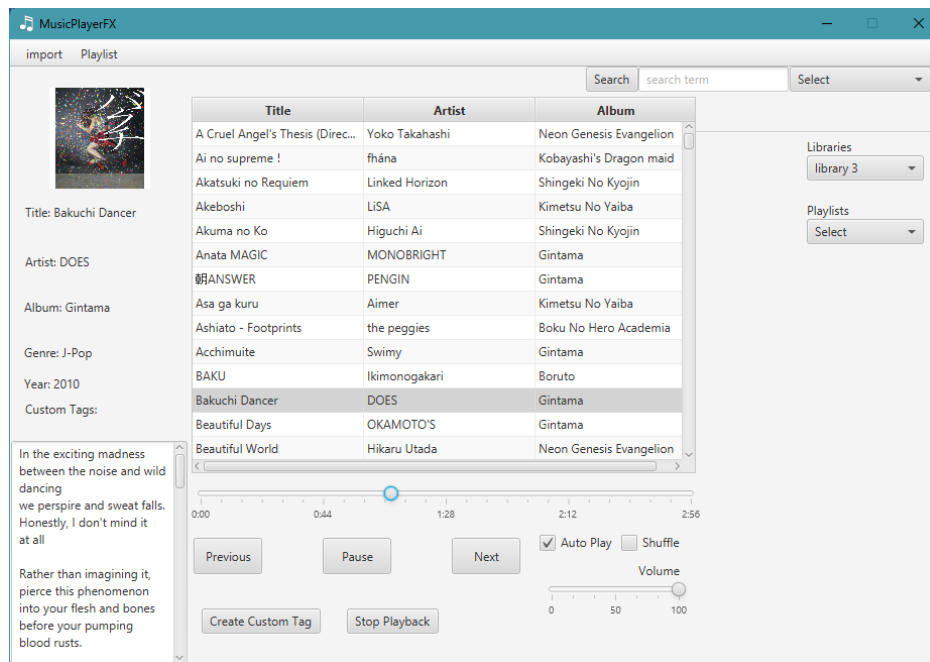
User Interfaces
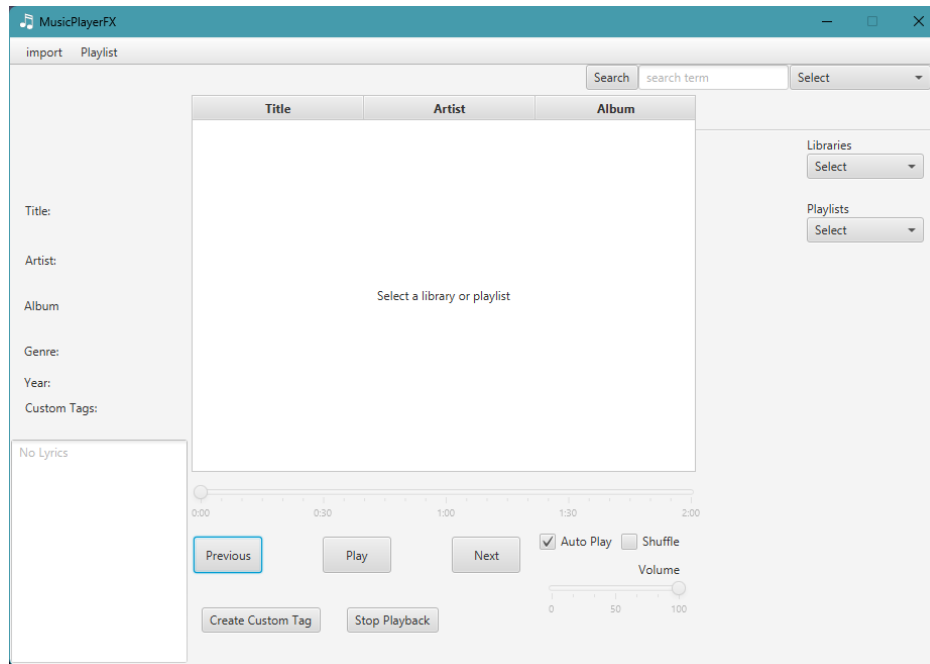
Figure 7: Player at startup

Figure 8: Playing from a library
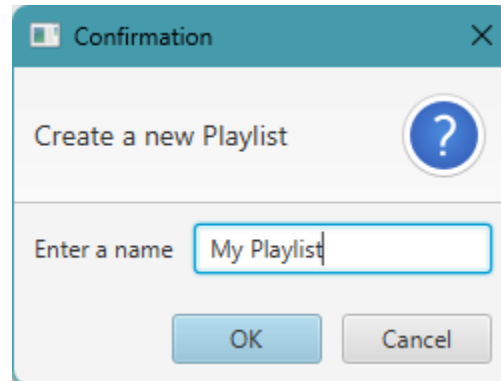
Figure 9: creating playlist
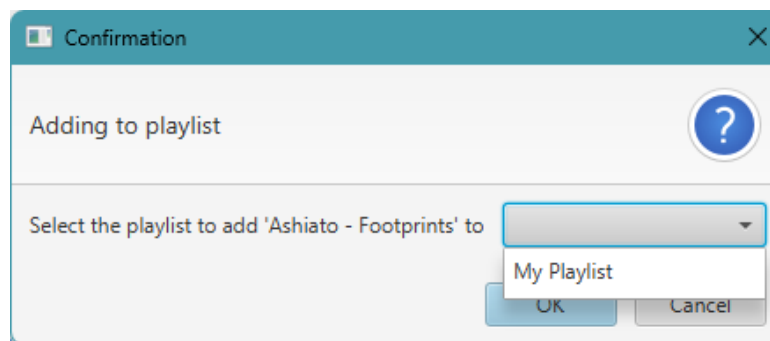
Figure 10: Adding song to playlist
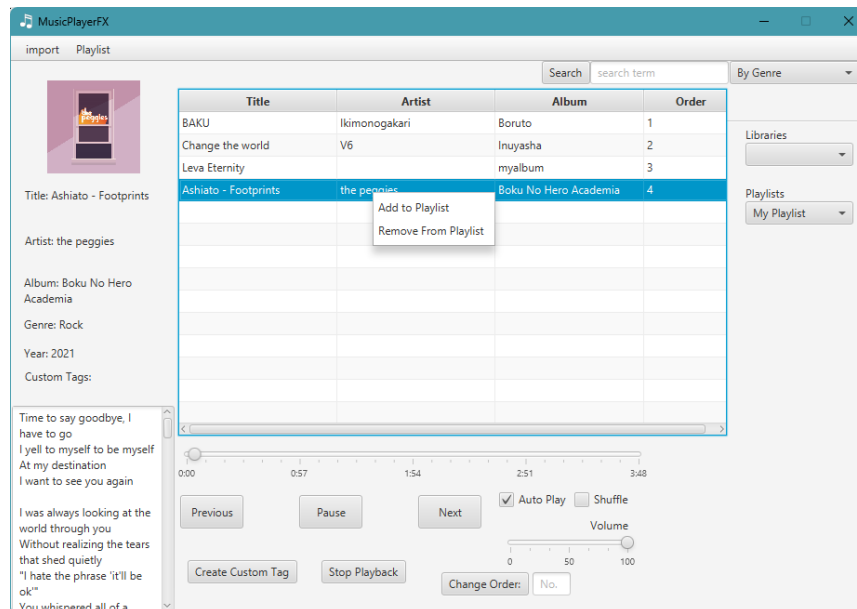
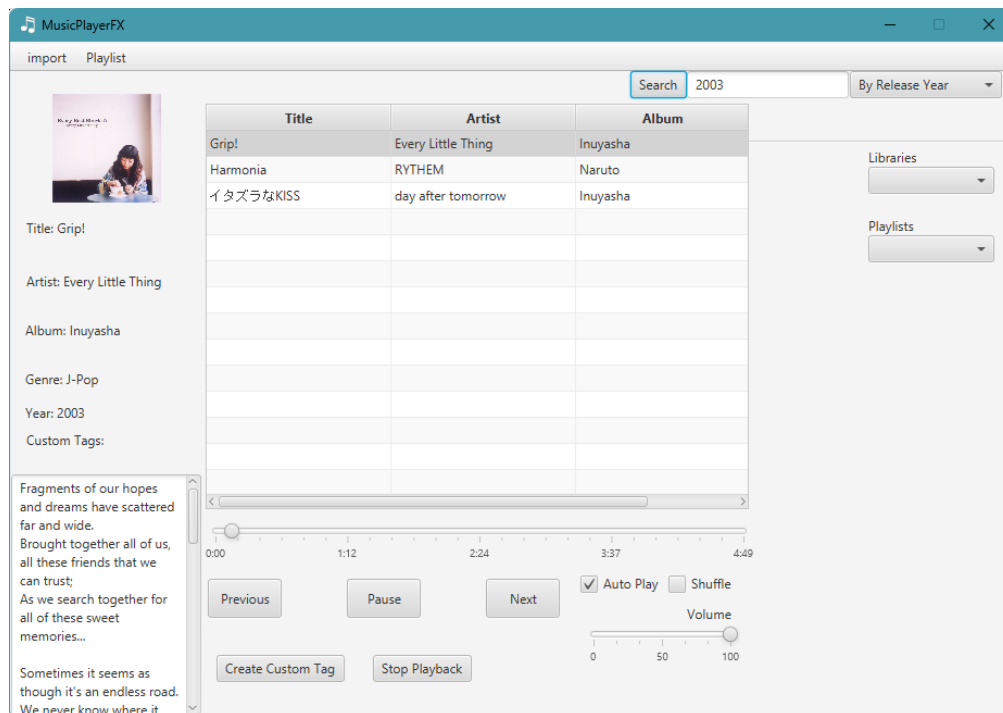Figure 11: Viewing a playlist

Figure 12: Searching for songs from 2003

6 Performance Test

Performance testing is important to make sure the newly developed system behaves and performs as expected. I identified the following constraints that might affect my application-

1. **Memory:** Any application that needs to run on a computer needs to be loaded in main memory. If there is not enough memory available then the software might slow down and become difficult to use. In some cases the program will even crash.
   Since my application is made in Java, the JVM will automatically remove objects from memory with garbage collection if they are not being used so memory is not wasted. I also tried to design my code in such a way that system resources are not being used unnecessarily.

2. **Accuracy:** Accuracy in software means that output produced by the software is what is expected by the user. Software must be made to handle any errors it might come across while it is running so that the errors do not disrupt the system and there can be a smooth continuation of its operations.
   Since my project involves databases I needed to use the Java SQL package which allows me to connect to my database. All the methods relating to databases throw SQL exception so all of the database functionality had to be surrounded with try/catch blocks to handle the exceptions. The logging class was used to log any exceptions to a log file so that any problems can be analyzed later.

3. **Disk Usage:** The most resource intensive operation in my music player is importing music files. All the files in a selected folder are checked if they are .mp3 files, then in a loop the metadata is retrieved from one mp3 file and then stored into the database then the loop iterates to the next file. When importing a folder with a large number of music files it takes a long time to save it all in the database. Initially I used a new thread to do this importing operation so the whole application does not freeze while importing, but the time taken for importing is still the same. The test cases below will analyze this constraint.

6.1 Test Plan/ Test Cases

I have many music files on my computer, with them I will test the disk usage and time taken to import all music files.

Sl. No.	Test Cases	Importing Time (sec.)	Max Disk Usage (MB/s)
1	Import 7 files	-	-
2	Import 324 files	-	-
3	Import 554 files	-	-
4	Import 1014 files	-	-

6.2 Test Procedure

I have many mp3 files on my computer all of which have metadata. By copying and pasting the mp3 files I can create different folders which contain varying number of mp3 files.

Using the Java logging API I can output a message in the log file before the importing starts and after it is finished. Using the LocalDateTime object in the java.time package I can save the start and end time. However it was not necessary in test cases 2,3 and 4 since the logging function already outputs the time of the log entry. The disk usage will be monitored with Windows task manager.
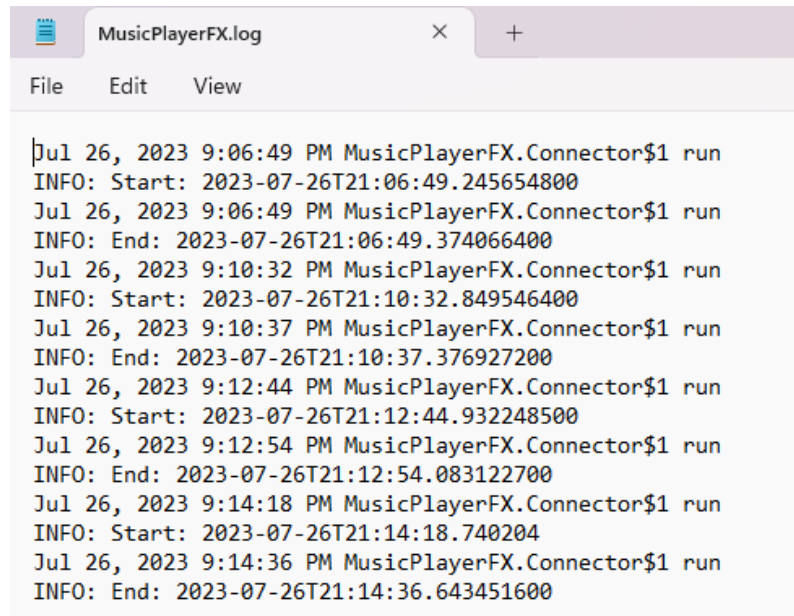
6.3 Performance Outcome

Sl. No.	Test Cases	Importing Time (sec.)	Max Disk Usage (MB/s)
1	Import 7 files	0.129	0.1
2	Import 324 files	4.53	~340
3	Import 554 files	9.15	499
4	Import 1014 files	17.9	505

Analysing the test case outcomes, we can see that as the number of files being imported increases the importing time also increases linearly. The disk usage reaches around 500 MB/s then stabilizes.

The disk reading speed was able to reach this speed because my laptop has a NVMe M.2 SSD. If someone runs this application on a computer with a hard drive it might take even longer to import the files. This is why I used a Dialog box to tell the user after selecting a folder to import that large folders will take some time.

Logs from the test cases:

Figure 13: Logs

7 My learnings

During the course of my internship, I had the opportunity to learn about various aspects of Java and app development, which proved to be an enriching experience. One of the key skills I acquired was using JavaFX, a modern Java library that is used for creating user interfaces. I also gained proficiency in working with external libraries, such as mp3agic for metadata handling and sqlite for database management enabling me to add extra functionality to my project.

After many hours of fine-tuning, I successfully created an intuitive and user friendly UI for the application, providing me with invaluable experience of good UI design practices. I learned how to connect and interact with databases effectively, employing SQL queries to store and retrieve information efficiently.

A significant aspect of my learning was the improvement of my Java programming skills and understanding of Object-Oriented design principles and adoption of the Model-View-Controller (MVC) design pattern. This design principle ensures that the database connectivity, the user interface and the logic controlling both are kept separate, this way changing any one does not affect the other.

I also gained the ability to make use of library documentation, so I can get a deeper understanding of the code written by others and know how to properly implement them into my projects. I enhanced my problem-solving skills and became more proficient in using debuggers to ensure the smooth functioning of my application.

All of these different aspects of my technical learning is part of the hands-on practical experience with industry projects that Upskill Campus offers with their internship program.

Other than my project, I also had the opportunity to attend a webinar held by Upskill Campus and IoT Academy where they shared important information regarding interviews and how to successfully market yourself in interviews.

Overall, my six-week internship was a fantastic experience. The skills and knowledge I acquired during this time will definitely have a big impact on my career growth as a software developer.

8 Future work scope

There are many features that can be introduced to make this project an even better music player.

1. Multiple Audio Formats- Right now only mp3 files are supported by the app. It will be better if there is support for all popular audio formats and their metadata. It might be necessary to use many different libraries for different metadata formats.
2. Better UI – The UI can be made better by making full use of JavaFX. Since I only started to learn about JavaFX 6 weeks ago I could not make full use of its advanced capabilities like CSS Styling
3. Music Playback- There are many more playback options that could be added like repeating music, crossfade and equalizers.
4. Database redesign- The database could be normalized better, right now the database uses a music file's filepath as its primary key but it might be better to use a auto generated ID instead.
5. Controller redesign- The way the controller logic is right now there is a possibility that using one UI element affects other UI elements on the page. There must be better ways of implementing UI control.