Group Members	**KEVIN NYENDE OMYONGA - 669390**
	BILLY NELSON OMINGO - 668071
	BRIAN JOHN ONSATI - 668482
	LEONARD OCHIENG OMONDI - 653347
	WELLINGTON WEKESA LUVONGA - 671183
Course	**MIS6030 - Application Development (Spring 2024)**
Assignment	**Group Work - Bankai Jukebox Documentation**
Lecturer	**Lawrence Nderu, PhD**
Due Date	**Friday, 5 April 2024**

TABLE OF CONTENTS	

Assignment Parameters

FINAL DELIVERABLE C

To cover as much of stages 1 – 6 as you have completed:

A zip file containing working code in the form of .java and .class files or a zipped NetBeans project folder, together with any files or databases you have used for external storage. You are free to use NetBeans, but do not include the GUI code generated by NetBeans if you use the GUI builder!

A written report containing the evidence of all completed stages, which should include:

- Introduction
- Designs for the GUIs
- A commented version of CheckLibrary.java
- A description of how you designed and developed the final code
- Suitable screenshots of the program in operation
- Evidence of testing as detailed in the Testing section of the notes
- Details of any faults and failures
- Conclusions
- Your program listing.

The written part (excluding code listings) should be no more than 2,000 words and there should be no more than 5 screenshots. There will be a penalty for going over either of these limits.

This final deliverable is to be submitted to the coursework upload system BEFORE THE COURSEWORK DEADLINE.

Introduction

Bankai Jukebox is a Java application designed to provide users with a comprehensive music and media streaming experience, drawing inspiration from popular platforms like Apple Music and Spotify. This application allows users to enjoy music, videos, and radio channels seamlessly. One of its unique features is the ability to share currently playing music via a network link, enabling a collaborative listening experience among users.

The motivation behind developing Bankai Jukebox was to combine the elegance of modern music streaming interfaces with the flexibility of Java programming, making it accessible to a wide range of users across different platforms.

Design and Development

The development journey of Bankai Jukebox involved meticulous planning and iterative implementation:

1. **Requirements Gathering:** Detailed analysis of user expectations and feature requirements, including media playback, radio streaming, and networking capabilities.
2. **GUI Design Iterations:** Iterative design processes involving wire-framing, mockups, and user feedback to refine the interface.
3. **Backend Implementation:** Implementation of core functionalities such as media playback using Java's multimedia libraries, radio streaming using internet protocols, networking features for music sharing, and database integration with SQLite.
4. **Testing and Optimization:** Extensive testing phases encompassing unit tests and user acceptance tests to ensure robustness, performance, and user satisfaction.

The iterative development approach allowed for continuous improvement and adaptation based on feedback and evolving requirements.

Technologies Used

The app was developed using the following technologies:

- **Java:** The core programming language used for application development.
- **Java Swing:** The GUI toolkit used for creating the graphical user interface of the application.
- **SQLite:** The relational database management system used to store and manage music records and playlists.
- **IntelliJ IDEA IDE:** The integrated development environment used for coding, debugging, and testing the application.
- **GitHub:** The version control system utilized to store and manage the source code repository, facilitating collaboration among team members.

GUI Designs

Bankai Jukebox features a user-friendly graphical interface tailored to enhance the user experience. The design principles include:

- **Sleek Main Interface:** The main interface presents music categories such as playlists, artists, and albums in an organized layout, allowing users to navigate effortlessly.
- **Now Playing View:** This view showcases detailed information about the currently playing media, including album art, track metadata, and interactive playback controls for user convenience.

- **Radio Streaming:** Users can explore and tune into various radio stations with visually appealing thumbnails, enhancing discoverability and engagement.

The GUI design prioritizes simplicity, aesthetics, and functionality to ensure a delightful user interaction.

Database Integration

The Jukebox incorporates an SQLite database to efficiently manage records of music tracks and playlists. SQLite was chosen for its lightweight nature, ease of integration with Java applications, and support for relational database features.

Database Schema

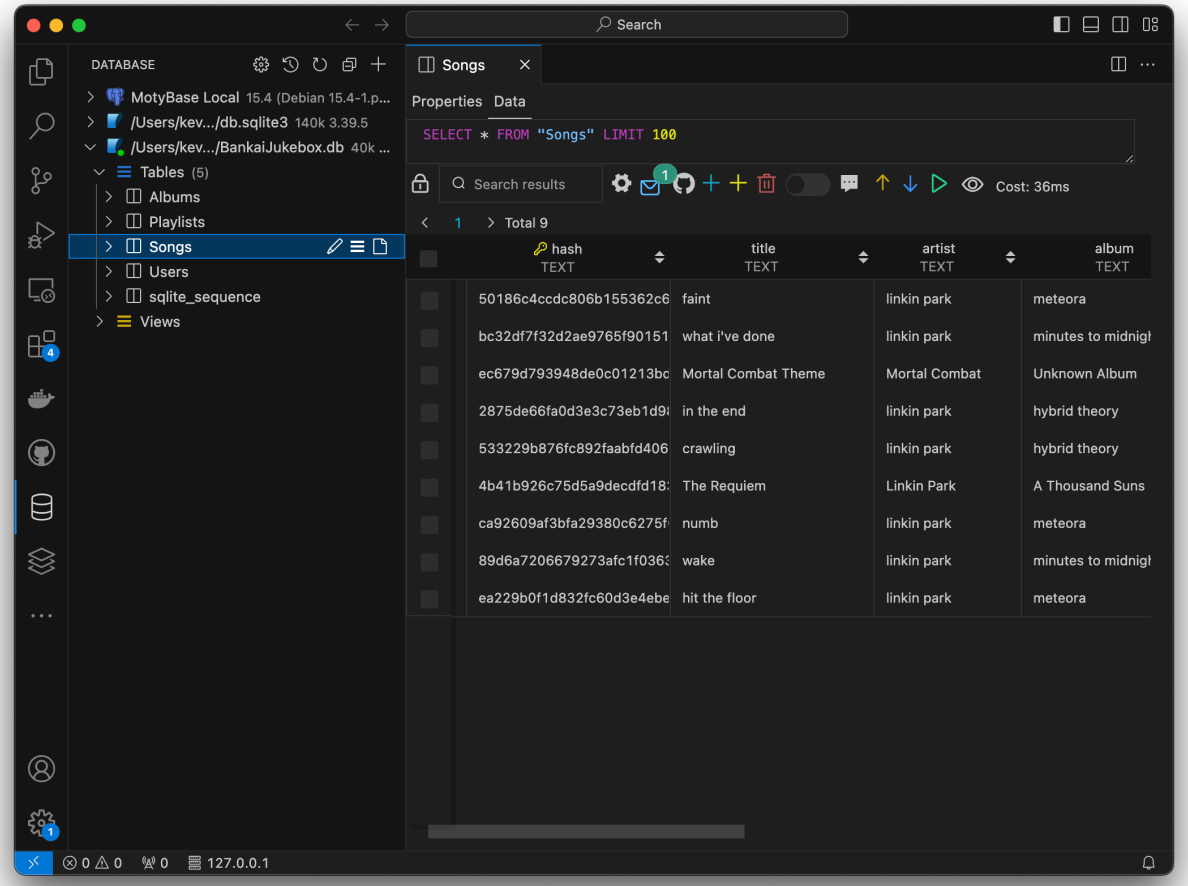
The SQLite database used by Bankai Jukebox includes the following tables:

1. **Songs Table:** Stores information about individual music tracks, including title, artist, album, duration, and file path. The 'songs' table schema in the Bankai Jukebox database.
2. **Playlists Table:** Manages playlists created by users, associating them with specific songs through a many-to-many relationship.

Database Interaction

The Bankai Group utilized JDBC (Java Database Connectivity) to establish a connection between the Java application and the SQLite database. This allowed seamless querying and manipulation of data within the application, enabling functionalities such as browsing songs, creating playlists, and retrieving metadata.

The integration of SQLite within Bankai Jukebox enhances its functionality by enabling robust data management and organization, contributing to a more dynamic and user-friendly music streaming experience.



Screenshot showing the structure of the 'songs' table within the Bankai Jukebox database, highlighting key fields used to store information about music tracks.

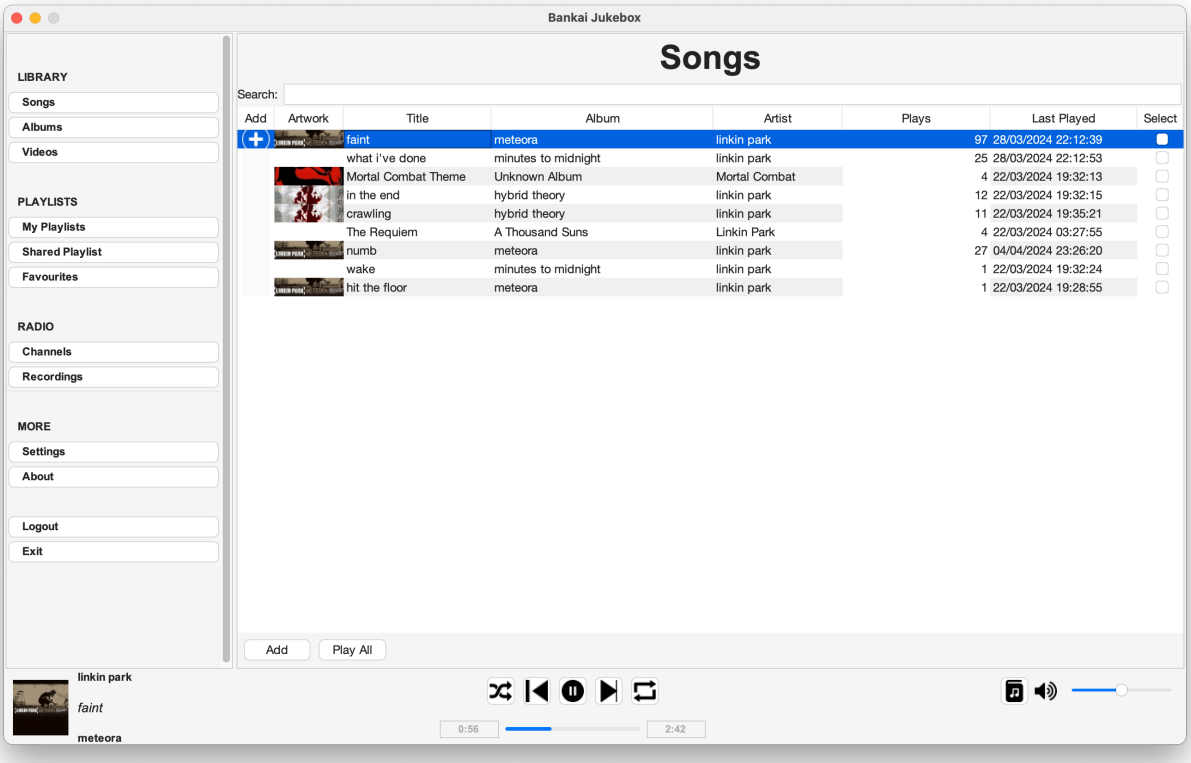
CheckLibrary.java

```
package com.bankai.jukebox.gui;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

public class CheckLibrary {
    private JPanel rootPanel; // Panel to hold all GUI components
    private JTextField txtTrackNumber; // Text field to input track number
    private JButton btnCheckTrack; // Button to check a specific track
    private JButton btnListAllTracks; // Button to list all tracks
    private JList list1; // List to display tracks

    public CheckLibrary() {
        // ActionListener for text field to handle user input
        txtTrackNumber.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Add functionality to handle user input in the text field
            }
        });

```

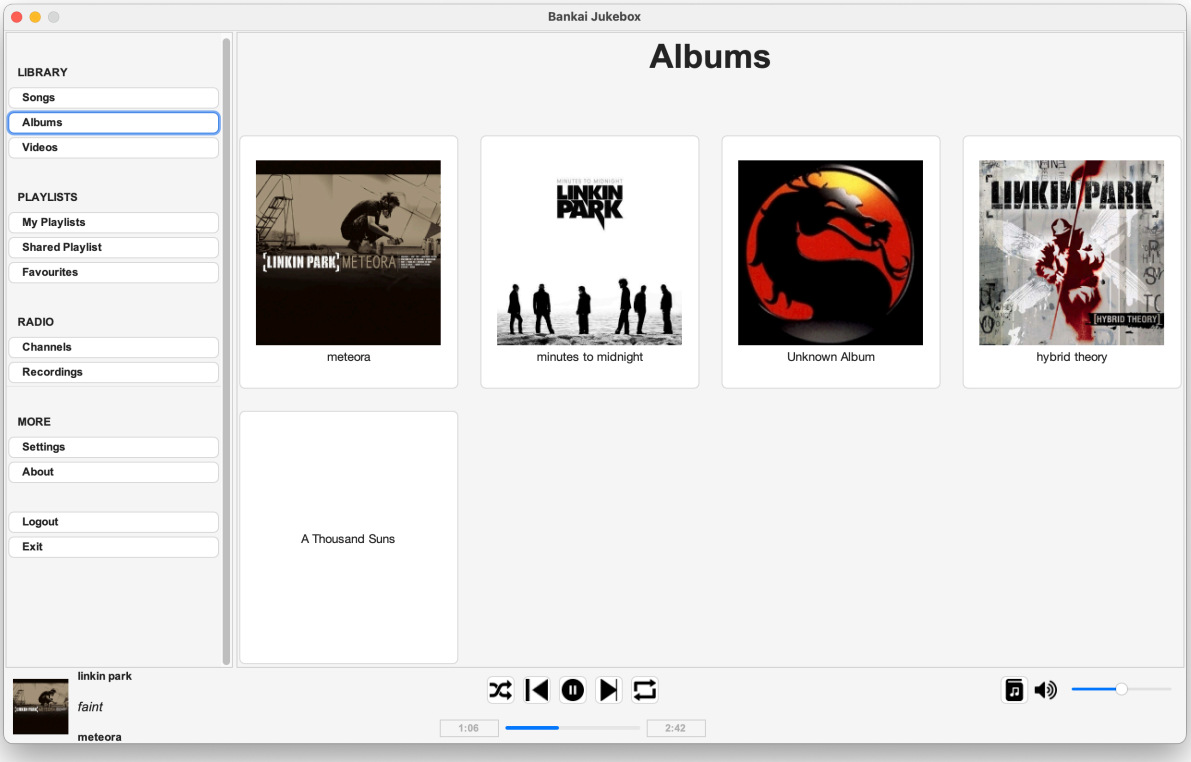
```
        // ActionListener for check track button
        btnCheckTrack.addActionListener(new ActionListener()
{
            @Override
            public void actionPerformed(ActionEvent e) {
                // Add functionality to check a specific
track
            }
        });

        // ActionListener for list all tracks button
        btnListAllTracks.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Add functionality to list all tracks
            }
        });
    }

    // Method to launch the GUI
    public void guiLaunch() {
        JFrame frame = new JFrame("Check Library"); //
Create a new frame with title
        frame.setContentPane(new
CheckLibrary().rootPanel); // Set the content pane to the
root panel

frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); //
Set close operation
        frame.setLocationRelativeTo(null); // Set frame to
appear in the center of the screen
        frame.pack(); // Pack components
```

```
        frame.setVisible(true); // Make frame visible
        frame.setResizable(false); // Disable frame resizing
    }

    public static void main(String[] args) {
        new CheckLibrary().guiLaunch(); // Create an
instance of CheckLibrary and launch the GUI
    }
}

```

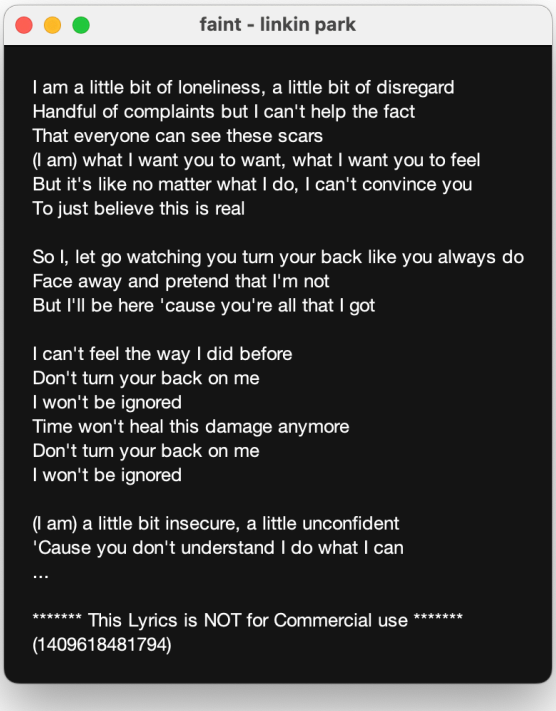
Screenshots



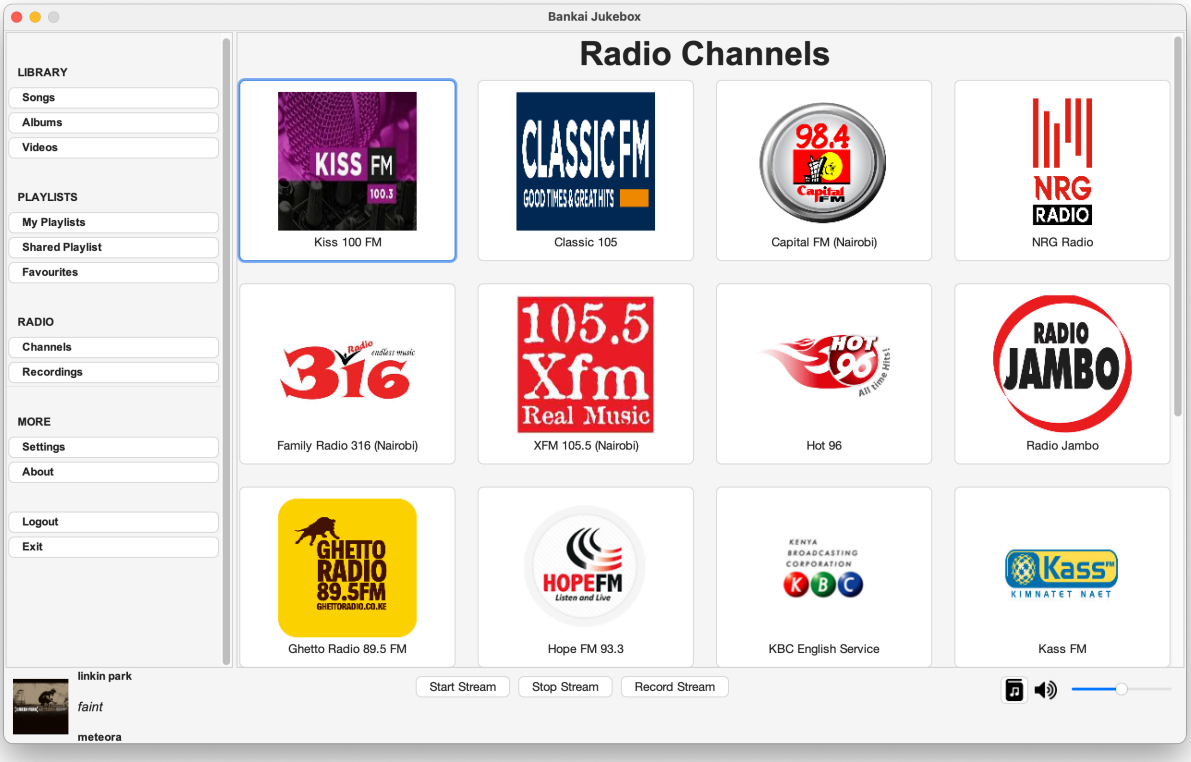
Screenshot showing the library with the first song on the queue selected



Screenshot showing the albums the songs imported into the app have been grouped into.



Lyrics fetched from an online API for the song that is currently playing.



Screenshot showing local Kenyan radio stations that a user can stream music from

Testing

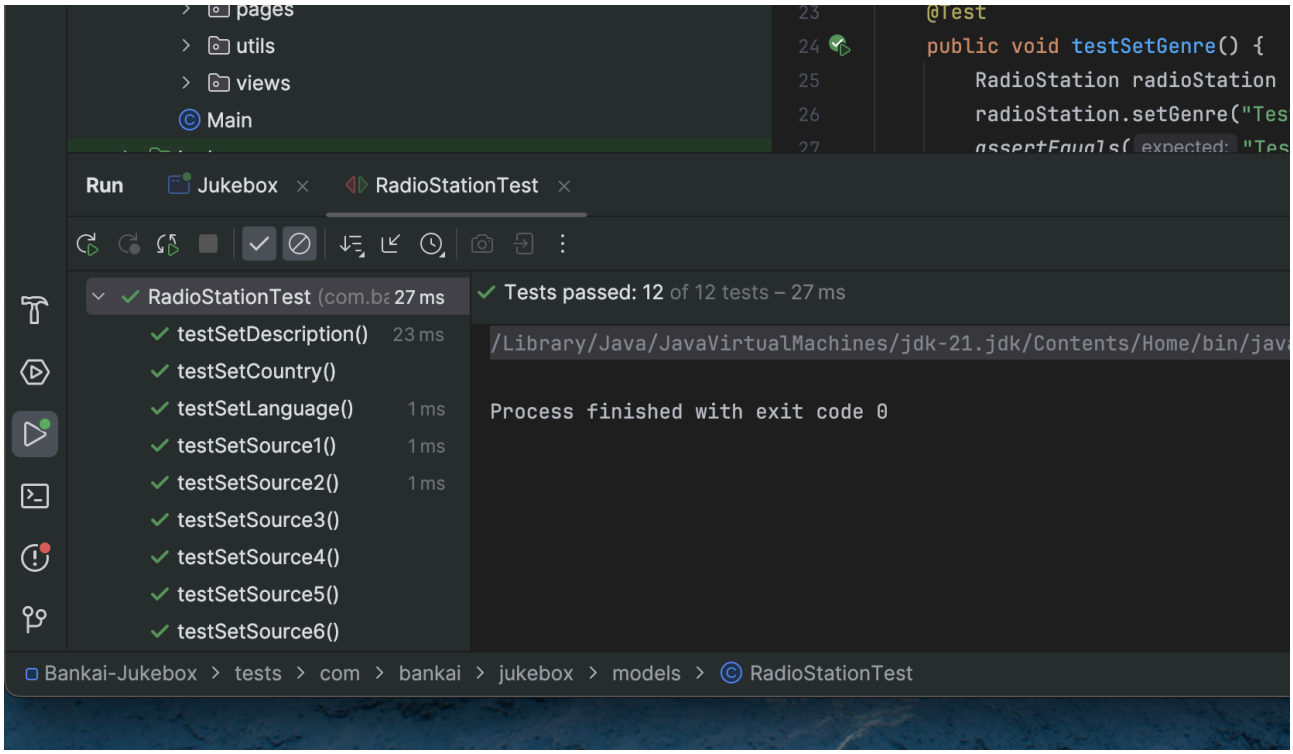
Comprehensive testing was conducted throughout the development lifecycle to validate Bankai Jukebox:

- **Unit Testing:** Individual components such as media player, networking modules, database interactions, and data management were thoroughly tested to verify correctness.
- **User Acceptance Testing:** Beta testing involved real users to gather feedback on usability, performance, and feature completeness.

Testing efforts were aimed at identifying and resolving issues promptly to deliver a stable and reliable application.

```
new *
public class RadioStationTest {

    new *
    @Test
    public void testSetTitle() {
        RadioStation radioStation = new RadioStation();
        radioStation.setTitle("Test Title");
        assertEquals( expected: "Test Title", radioStation.getTitle());
    }

    new *
    @Test
    public void testSetDescription() {
        RadioStation radioStation = new RadioStation();
        radioStation.setDescription("Test Description");
        assertEquals( expected: "Test Description", radioStation.getDescription());
    }

    new *
    @Test
    public void testSetGenre() {
```

Unit Test To Check The Radio Station Model



Results of running the test cases on the RadioStationModel

Faults and Failures

During testing and user feedback, several areas for improvement and challenges were identified:

1. **Network Streaming Latency:** Occasional delays in streaming media over slower network connections.
2. **UI Responsiveness:** Optimizing the user interface for smoother transitions and faster updates during media playback.

Addressing these challenges required focused efforts in performance optimization and user experience refinement.

Conclusions

The Bankai Jukebox application represents a significant achievement in Java application development, combining technical prowess with user-centered design principles. Despite encountering challenges, the application successfully delivers on its promise of providing an engaging and feature-rich music streaming experience.