Vivekanand Education Society's Institute Of Technology
Department Of Information Technology
DSA mini Project
A.Y. 2025-26

Title : BeatLinked-Smart Music Playlist Management System using Data Structures

Sustanibilty Goal: Supports UN SDG 9(Industry, Innovation & Infrastructure) by promoting digital innovation through intelligent, data-driven playlist management.

Domain: **Data Structures & Algorithms**

Member: **Gourish Naik**

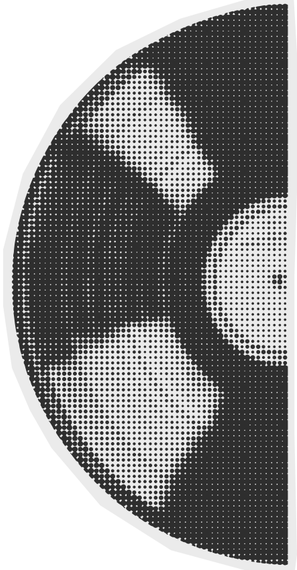
Mentor Name: **Kajal Jewani**


6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
THE GLOBAL GOALS
For Sustainable Development
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE AND JUSTICE STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS

Content

Introduction to Project

- The **Predictive Song Playlist System** is a console-based Java application that allows users to manage, play, skip, and predict songs in a music playlist.
- The system demonstrates the use of **Data Structures like LinkedList, Queue, and Stack** and simulates **intelligent behavior** by predicting the next song a user may want to play based on their recent activity.
- This project combines **Data Structure implementation** with a simple form of **automation/prediction**



Problem Statement

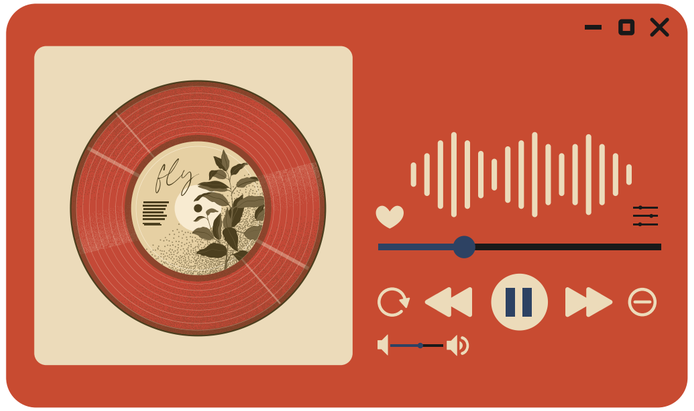
In modern music players, users often want **recommendations or automatic suggestions** for the next song based on their listening history.

Existing systems may require complex AI or database support.

- This project solves the problem by providing a **lightweight predictive playlist system** using only fundamental data structures, making it suitable for learning DSA concepts while offering simple prediction functionality.

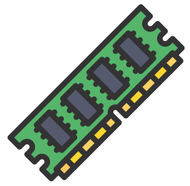
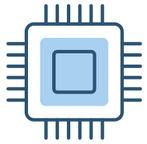
Objectives of the project

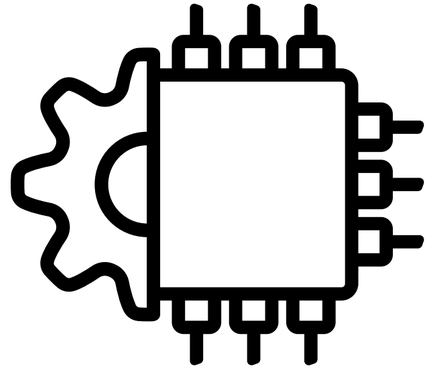
- Implement **play, skip, undo, and prediction features.**
- Demonstrate DSA operations like **insertion, deletion, traversal.**
- Maintain a **recently played list and undo stack.**
- Provide **intelligent next-song suggestions**.



Requirements of the system (Hardware, software)

Hardware:

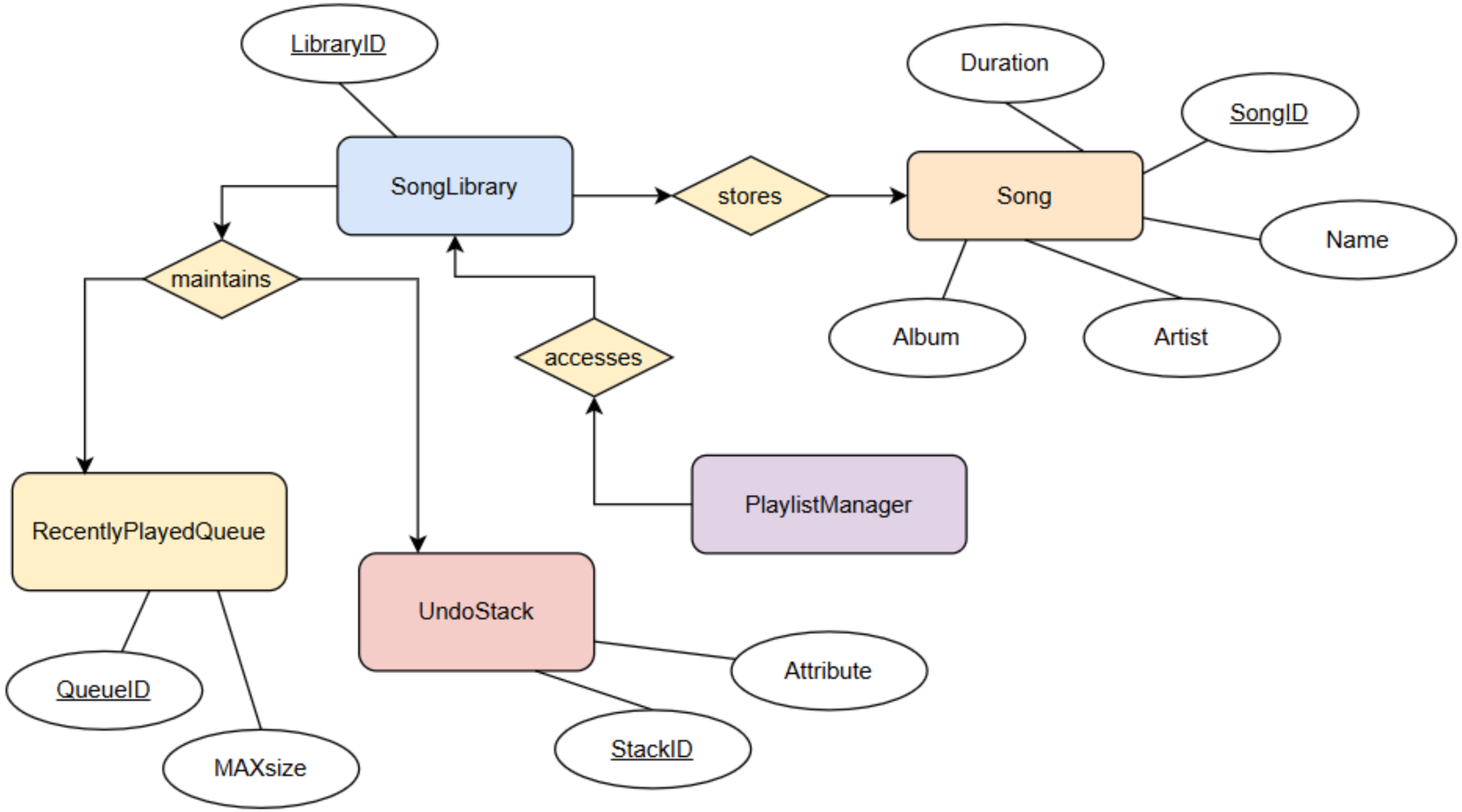
- 4GB RAM minimum 
- 500MB free storage
- Any standard processor 



Software:

- Java JDK 11 or higher 
- Any Java IDE (Eclipse, VS Code, IntelliJ) 
- Terminal / Command Prompt 

ER diagram of the proposed system



Front End

```
--- Predictive Song Playlist ---
1. Display All Songs
2. Play a Song
3. Skip a Song
4. Undo Last Action
5. Show Recently Played
6. Predict Next Song
7. Exit
Enter your choice:
```

Implementation

```java
import java.util.LinkedList;
import java.util.Scanner;

// Song class
class Song {
    String name;
    String artist;
    String album;
    int duration; // in seconds

    public Song(String name, String artist, String album, int duration) {
        this.name = name;
        this.artist = artist;
        this.album = album;
        this.duration = duration;
    }

    @Override
    public String toString() {
        return name + " by " + artist + " (" + album + ") [" + duration + "s]";
    }
}

// Song Library using LinkedList
class SongLibrary {
    LinkedList<Song> songs;

    public SongLibrary() {
        songs = new LinkedList<>();
    }

    public void addSong(Song song) {
        songs.add(song);
    }

    public void displayAllSongs() {
```

```java
class SongLibrary {
    public void displayAllSongs() {
            System.out.println("No songs in the library.");
            return;
        }
        System.out.println("Song Library:");
        int i = 1;
        for (Song s : songs) {
            System.out.println(i + ". " + s);
            i++;
        }
    }

    public Song getSong(int index) {
        if (index >= 0 && index < songs.size()) {
            return songs.get(index);
        } else {
            return null;
        }
    }

    public int size() {
        return songs.size();
    }
}

// Playlist Manager with Queue and Stack
class PlaylistManager {
    LinkedList<Song> recentlyPlayed; // Queue
    LinkedList<Song> undoStack;      // Stack
    int maxRecent;

    public PlaylistManager(int maxRecent) {
        recentlyPlayed = new LinkedList<>();
        undoStack = new LinkedList<>();
        this.maxRecent = maxRecent;
    }

    public void playSong(Song song) {
        System.out.println("Playing: " + song);
        // Add to recently played queue
        recentlyPlayed.addLast(song);
        if (recentlyPlayed.size() > maxRecent) {
            recentlyPlayed.removeFirst(); // maintain max size
        }
        // Push to undo stack
        undoStack.push(song);
    }

    public void skipSong(Song song) {
        System.out.println("Skipped: " + song);
        // Push skipped song to undo stack
        undoStack.push(song);
    }

    public void undoLastAction() {
        if (undoStack.isEmpty()) {
            System.out.println("Nothing to undo.");
            return;
        }
        Song last = undoStack.pop();
        System.out.println("Undoing last action. You can replay: " + last);
        playSong(last);
    }

    public void showRecentlyPlayed() {
        if (recentlyPlayed.isEmpty()) {
```

```java
    public void showRecentlyPlayed() {
        if (recentlyPlayed.isEmpty()) {
            System.out.println("No recently played songs.");
            return;
        }
        System.out.println("Recently Played Songs:");
        for (Song s : recentlyPlayed) {
            System.out.println("- " + s);
        }
    }

    // Simple prediction: picks the 1st song in library that is not in recently played
    public void predictNextSong(SongLibrary library) {
        for (Song s : library.songs) {
            if (!recentlyPlayed.contains(s)) {
                System.out.println("Predicted Next Song: " + s);
                return;
            }
        }
        System.out.println("No prediction available (all songs recently played).");
    }
}

// Main Program
public class EchoQueueSystem {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        SongLibrary library = new SongLibrary();
        PlaylistManager manager = new PlaylistManager(maxRecent:5); // Keep last 5 songs

        // Sample songs
        library.addSong(new Song(name:"Shape of You", artist:"Ed Sheeran", album:"Divide", duration:240));
        library.addSong(new Song(name:"Blinding Lights", artist:"The Weeknd", album:"After Hours", duration:200));
        library.addSong(new Song(name:"Levitating", artist:"Dua Lipa", album:"Future Nostalgia", duration:203));
        library.addSong(new Song(name:"Believer", artist:"Imagine Dragons", album:"Evolve", duration:210));
        library.addSong(new Song(name:"Perfect", artist:"Ed Sheeran", album:"Divide", duration:265));

        int choice;
        do {
            System.out.println("\n--- Predictive Song Playlist ---");
            System.out.println("1. Display All Songs");
            System.out.println("2. Play a Song");
            System.out.println("3. Skip a Song");
            System.out.println("4. Undo Last Action");
            System.out.println("5. Show Recently Played");
            System.out.println("6. Predict Next Song");
            System.out.println("7. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            sc.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    library.displayAllSongs();
                    break;
                case 2:
                    library.displayAllSongs();
                    System.out.print("Enter song number to play: ");
                    int playIndex = sc.nextInt() - 1;
                    sc.nextLine();
                    Song playSong = library.getSong(playIndex);
                    if (playSong != null) {
                        manager.playSong(playSong);
                    } else {
                        System.out.println("Invalid song selection.");
                    }
                    break;
                case 3:
                    library.displayAllSongs();
                    System.out.print("Enter song number to skip: ");
                    int skipIndex = sc.nextInt() - 1;
                    sc.nextLine();
                    Song skipSong = library.getSong(skipIndex);
                    if (skipSong != null) {
                        manager.skipSong(skipSong);
                    } else {
                        System.out.println("Invalid song selection.");
                    }
                    break;
                case 4:
                    manager.undoLastAction();
                    break;
                case 5:
                    manager.showRecentlyPlayed();
                    break;
                case 6:
                    manager.predictNextSong(library);
                    break;
                case 7:
                    System.out.println("Exiting... Goodbye!");
                    break;
                default:
                    System.out.println("Invalid choice. Try again.");
            }

        } while (choice != 7);

        sc.close();
    }
}
```

Outputs

```
--- Predictive Song Playlist ---
1. Display All Songs
2. Play a Song
3. Skip a Song
4. Undo Last Action
5. Show Recently Played
6. Predict Next Song
7. Exit
Enter your choice:
```

```
--- Predictive Song Playlist ---
1. Display All Songs
2. Play a Song
3. Skip a Song
4. Undo Last Action
5. Show Recently Played
6. Predict Next Song
7. Exit
Enter your choice: 6
Predicted Next Song: Shape of You by Ed Sheeran (Divide) [240s]
```

```
--- Predictive Song Playlist ---
1. Display All Songs
2. Play a Song
3. Skip a Song
4. Undo Last Action
5. Show Recently Played
6. Predict Next Song
7. Exit
Enter your choice: 2
Song Library:
1. Shape of You by Ed Sheeran (Divide) [240s]
2. Blinding Lights by The Weeknd (After Hours) [200s]
3. Levitating by Dua Lipa (Future Nostalgia) [203s]
4. Believer by Imagine Dragons (Evolve) [210s]
5. Perfect by Ed Sheeran (Divide) [265s]
Enter song number to play: 4
Playing: Believer by Imagine Dragons (Evolve) [210s]
```

```
--- Predictive Song Playlist ---
1. Display All Songs
2. Play a Song
3. Skip a Song
4. Undo Last Action
5. Show Recently Played
6. Predict Next Song
7. Exit
Enter your choice: 1
Song Library:
1. Shape of You by Ed Sheeran (Divide) [240s]
2. Blinding Lights by The Weeknd (After Hours) [200s]
3. Levitating by Dua Lipa (Future Nostalgia) [203s]
4. Believer by Imagine Dragons (Evolve) [210s]
5. Perfect by Ed Sheeran (Divide) [265s]
```

```
--- Predictive Song Playlist ---
1. Display All Songs
2. Play a Song
3. Skip a Song
4. Undo Last Action
5. Show Recently Played
6. Predict Next Song
7. Exit
Enter your choice: 4
Undoing last action. You can replay: Perfect by Ed Sheeran (Divide) [265s]
Playing: Perfect by Ed Sheeran (Divide) [265s]
```

```
Song Library:
1. Shape of You by Ed Sheeran (Divide) [240s]
2. Blinding Lights by The Weeknd (After Hours) [200s]
3. Levitating by Dua Lipa (Future Nostalgia) [203s]
4. Believer by Imagine Dragons (Evolve) [210s]
5. Perfect by Ed Sheeran (Divide) [265s]
Enter song number to skip: 5
Skipped: Perfect by Ed Sheeran (Divide) [265s]
```

```
--- Predictive Song Playlist ---
1. Display All Songs
2. Play a Song
3. Skip a Song
4. Undo Last Action
5. Show Recently Played
6. Predict Next Song
7. Exit
Enter your choice: 5
Recently Played Songs:
- Blinding Lights by The Weeknd (After Hours) [200s]
- Blinding Lights by The Weeknd (After Hours) [200s]
- Blinding Lights by The Weeknd (After Hours) [200s]
- Believer by Imagine Dragons (Evolve) [210s]
- Shape of You by Ed Sheeran (Divide) [240s]
```

Gantt Chart

WEEK	TASK
1	**Requirement Analysis & DSA Selection**
2	Design Classes and Data Structures
3	Implement SongLibrary & PlaylistManager
4	Implement Play, Skip, Undo Features
5	Implement Prediction & Recently Played Features
6	Testing, Debugging, and Final Report Preparation

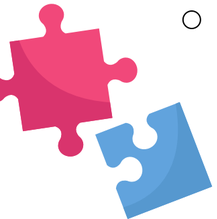
TEST CASE	INPUT	EXPECTED OUTPUT	Result
Play a song	Select song 1	“Playing: Shape of You ...”	**PASS**
Skip a song	Select song 2	“Skipped: Blinding Lights ...”	PASS
Undo last	Last action = play song	Last song plays again	PASS
Predict next	Library contains unplayed songs	First unplayed song is predicted	PASS
Show recently played	Queue not empty	Lists last 5 songs	PASS

Conclusion

The Predictive Song Playlist System successfully demonstrates how fundamental data structures—LinkedList, Queue, and Stack—can be utilized to manage and manipulate a music playlist efficiently. The system allows users to play, skip, undo, and view recently played songs, while also providing a simple prediction of the next song based on listening history.

CHALLENGES AND SOLUTIONS

- **Challenge**: Maintaining prediction without using HashMap
  - **Solution**: Traverse LinkedList of library and compare with recently played queue.
- **Challenge**: Undo feature for both play and skip
  - **Solution**: Use Stack to store last action for LIFO undo



Future Scope

- Add GUI-based interface.
- Include genre and artist-based predictions.
- Integrate real music API for live song data.
- Expand prediction algorithm using simple AI/ML.

References

1. R. Lafore, Data Structures and Algorithms in Java, 2nd ed., Sams Publishing, 2002.
2. Oracle, Java™ Platform Standard Edition Documentation, [Online]. Available: https://docs.oracle.com/javase/8/docs/
3. Y. Daniel Liang, Introduction to Java Programming, 10th ed., Pearson, 2015.
4. GeeksforGeeks, “Java LinkedList, Stack, and Queue”, [Online]. Available: **Implement** play, skip, undo, and prediction features.
5. **Demonstrate DSA operations like** insertion, deletion, traversal.
6. **Maintain a** recently played list and undo stack.
7. **Provide** intelligent next-song suggestions.