

# **JAVA PROJECT REPORT**

(Project Term January-May 2023)

## **MEDIA PLAYBACK PROGRAM FOR AUDIO FILES**

Submitted By:

VAINAVI SRIVASTAVA

12111359

CSE-310

Under the guidance of

MR. RANJITH A. KUMAR

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



## **DECLARATION**

We hereby declare that the project work entitled MEDIA PLAYBACK PROGRAM FOR AUDIO FILES is an authentic record of our own work carried out as requirements of JAVA Project for the award of BTech degree in COMPUTER SCIENCE AND ENGINEERING from Lovely Professional University, Phagwara, under the guidance of MR. RANJITH A. KUMAR during January to May term 2023. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

**Name:** VAINAVI SRIVASTAVA

**Registration Number:** 12111359

(Vainavi Srivastava)

Date: 20 April,2023

## **TABLE OF CONTENTS**

### **1. INTRODUCTION**

### **2. PROPOSED TECHNIQUE**

- i)     MODULE 1
- ii)    MODULE 2
- iii)   MODULE 3

### **3. CODE**

### **4. CONCLUSION**

## **INTRODUCTION**

Music is one of the best ways to relieve pressure in stressful modern society life. The purpose of this project is to develop a player which can play the mainstream file format. To browse and query the storage space as well as operation of playing can be realised. Meanwhile, this software can play, pause and select songs with latest button and next button according to sets requirement as well as set up songs.

Java music player is a simple classic mp3 player which has features like playing selected mp3 music files, pausing the music, resuming the music, and stopping the music.

The music player is used daily by all types of users. Music helps users to create a fresh mind, inspire life, and also boost the mind of the user.

The music player allows a user to play various media file formats. It can be used to play audio as well as video files. The music player is a software project supporting all known media files and has the ability to play them with ease.

The project features are as follows:

- Interactive GUI
- Consists of Pause/Play/Stop Features
- Consists of a Volume controller.
- As the song plays you can see the lyrics as well.
- A dancing GIF is installed in the Background.

## **MERITS:**

- Easier to explain or put things in perspective.
- Helps keep mistakes at a minimum

## **PROPOSED TECHNIQUE**

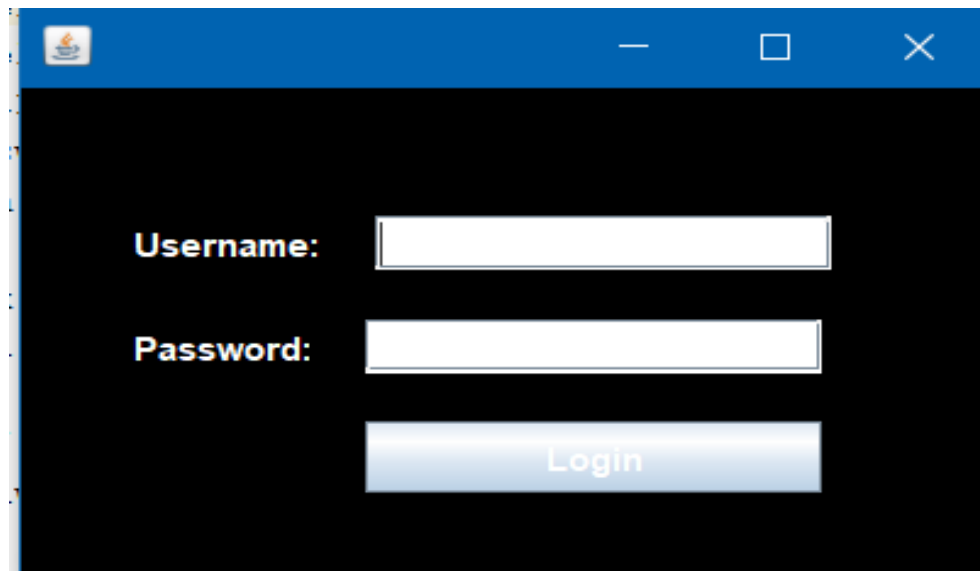
1. **J-SWING and J-Frames:** J-SWING is a UI toolkit for building GUI applications in JAVA, while J-FRAMES is a class in the SWING toolkit that provides a top-level container for creating GUI application in JAVA.

We created the player window which can be reduced in size. Addition of buttons in the window and text fields were added using that.

2. **JACo MP3 Library:** It is a JAVA based library for working with MP3 files. It provides a set of classes and methods for decoding, playing, and manipulating MP3 audio data. The library is open-source and freely available for download and use.

Support for creating and playing playlists of MP3 files. The JACo MP3 library is well-documented and includes a number of examples to help developers get started. It can be used in a variety of applications, including music players, audio editors, and more.

## MODULE 1: LOGIN MODULE



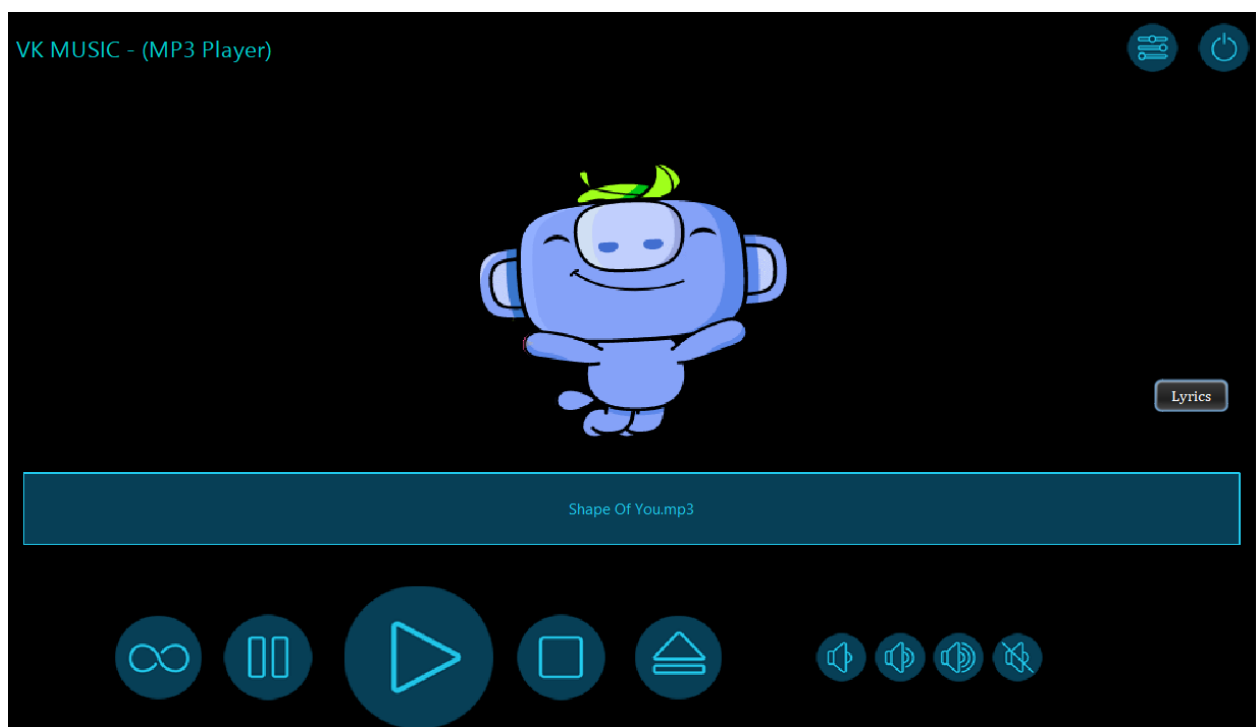
A screenshot of a login window with a blue title bar and a black background. It features two white input fields for 'Username' and 'Password', and a blue 'Login' button.

Username:

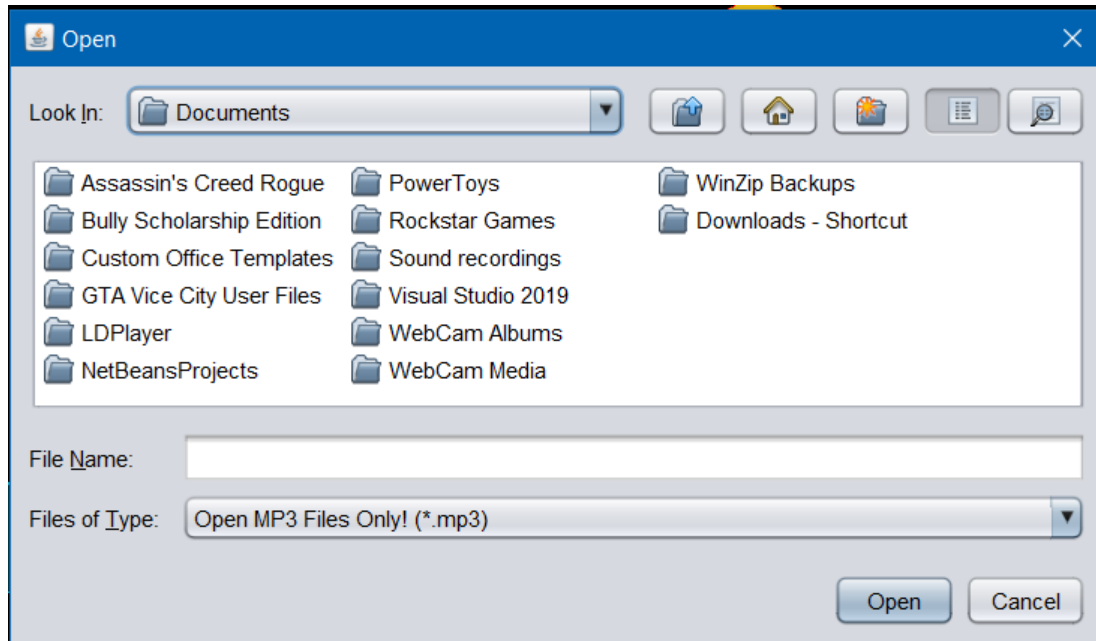
Password:

Login

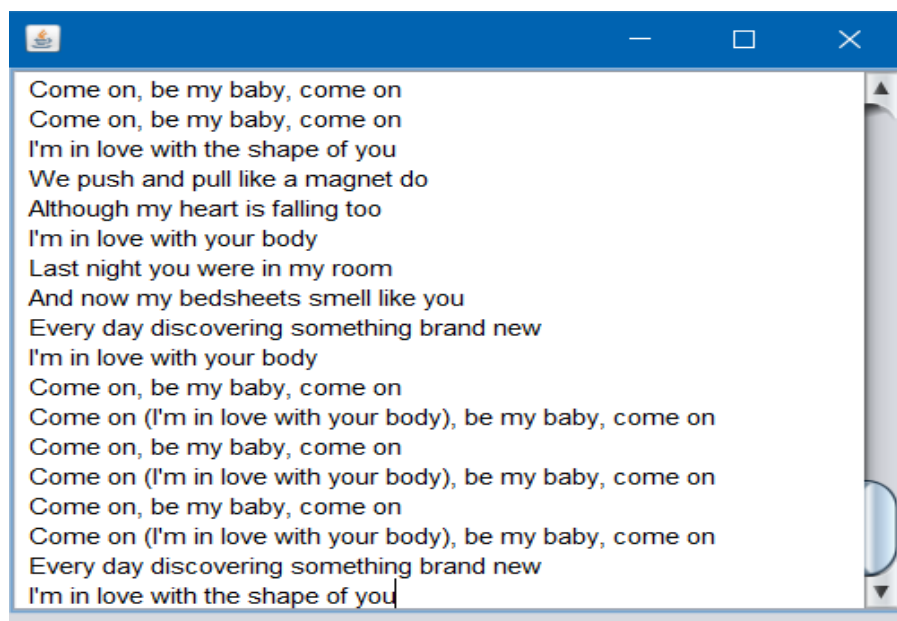
## MODULE 2: PLAYER MODULE



## MODULE 3 : FILE ACCESS MODULE



## MODULE 4: LYRICS FRAME



## PLAYER FRAME CODE SNIPPET

```
private void volumeControl(Double valueToPlusMinus){
    // Get Mixer Information From AudioSystem
    Mixer.Info[] mixers = AudioSystem.getMixerInfo();
    // Now use a for loop to list all mixers
    for(Mixer.Info mixerInfo : mixers){
        // Get Mixer
        Mixer mixer = AudioSystem.getMixer(info: mixerInfo);
        // Now Get Target Line
        Line.Info[] lineInfos = mixer.getTargetLineInfo();
        // Here again use for loop to list lines
        for(Line.Info lineInfo : lineInfos){
            // Make a null line
            Line line = null;
            // Make a boolean as opened
            boolean opened = true;
            // Now use try exception for opening a line
            try{
                line = mixer.getLine(info: lineInfo);
                opened = line.isOpen() || line instanceof Clip;
                // Now Check If Line Is not Opened
                if(!opened){
                    // Open Line
                    line.open();
                }
                // Make a float control
                FloatControl volControl = (FloatControl) line.getControl(control: FloatControl.Type.VOLUME);
                // Get Current Volume Now
                float currentVolume = volControl.getValue();
                // Make a temp double variable and store valuePlusMinus
                Double volumeToCut = valueToPlusMinus;
                // Make a float and calculate the addition or subtraction in volume
                float changedCalc = (float) ((double)volumeToCut);
                // Now Set This Changed Value Into Volume Line.
                volControl.setValue(newValue: changedCalc);
            }
```



## LYRICS CODE SNIPPET:

```
public class LyricFrame extends javax.swing.JFrame {
```

```
    /**
```

```
     * Creates new form NewJFrame
```

```
     */
```

```
    public LyricFrame() {
```

```
        initComponents();
```

```
    }
```

```
    /**
```

```
     * This method is called from within the constructor to initialize the form.
```

```
     * WARNING: Do NOT modify this code. The content of this method is always
```

```
     * regenerated by the Form Editor.
```

```
     */
```

```
    @SuppressWarnings("unchecked")
```

```
    Generated Code
```

```
    /**
```

```
     * @param args the command line arguments
```

```
     */
```

```
    public static void main(String args[]) {
```

```
        /* Set the Nimbus look and feel */
```

```
        Look and feel setting code (optional)
```

```
        //</editor-fold>
```

```
        /* Create and display the form */
```

```
        java.awt.EventQueue.invokeLater(new Runnable() {
```

```
            public void run() {
```

```
                new LyricFrame().setVisible(true);
```

```
            }
```

## FILE ACCESS FRAME:

```
public class FileTypeFilter extends FileFilter{

    // File Extensions String
    private final String extension;
    // File Extension Description
    private final String description;

    // Constructor Method
    public FileTypeFilter(String extension, String description){
        // Set Constructor Values
        this.extension = extension;
        this.description = description;
    }

    @Override
    public boolean accept(File file) {
        // If File Is Returning Directory
        if(file.isDirectory()){
            return true;
        }
        // Return File Name with Extension
        return file.getName().endsWith(suffix:extension);
    }

    @Override
    public String getDescription() {
        // Return To Display File Type and Description
        return description + String.format(format:" (%s)", args: extension);
    }
}
```

## LOGIN PAGE FRAME

```
private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    PlayerFrame obj1=new PlayerFrame();  
  
    String username = usernameField.getText();  
    char[] password = passwordField.getPassword();  
  
    // Sample authentication logic  
    boolean authenticated = false;  
    if (username.equals(anObject: "admin") && String.valueOf(data: password).equals(anObject: "password")) {  
        authenticated = true;  
    }  
  
    if (authenticated) {  
        JOptionPane.showMessageDialog(parentComponent: this, message: "Login successful!");  
        obj1.setVisible(b: true);  
    } else {  
        JOptionPane.showMessageDialog(parentComponent: this, message: "Incorrect username or password.");  
    }  
}  
  
public static void main(String args[]) {  
    java.awt.EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            new LoginFrame().setVisible(b: true);  
        }  
    });  
}
```

## CONCLUSION

# CONCLUSION

The Web-Based Music Player is used to automate and give a better music player experience for the end user. The application solves the basic needs of music listeners without troubling them as existing applications do: it uses technology to increase the interaction of the system with the user in many ways.

It eases the work of the end-user by capturing the image using a camera, determining their emotion, and suggesting a customized play-list through a more advanced and interactive system. The user will also be notified of songs that are not being played, to help them free up storage space.

The proposed MP3 music player will focus on improving the experience of users of the music player experience.