KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering (CSE)

A Report on "Java Audio Player" project

Conducted under the course CSE 2200 (Advanced Programming)

Submitted to:

H. M. Abdul Fattah **&** Sunanda Das

Lecturer

Department of Computer Science and Engineering (CSE)

Khulna University of Engineering & Technology

Submitted by:

Md. Tajmilur Rahman

Roll: 1807100

Year: Second

Semester: Second

Department of Computer Science and Engineering

Khulna University of Engineering & Technology

INDEX

1. Objectives ---------- page 3
2. Introduction ---------- page 4
3. Implementation ---------- page 6
   i) Designing Log in & Register form ---------- page (6-11)
   ii) Design player frame ---------- page (12-15)
   iii) Functionalities ---------- page (16-18)
   iv) Debugging and testing ---------- page 18
4. Target vs. Actual Accomplishment ---------- page 19
5. Risks and Issues ---------- page 19
6. Discussion and Conclusion ---------- page 20
7. References ---------- page 20

1. OBJECTIVES

"Java Audio Player" is a project which is of a great use at present. The main goal of this project is to play back an audio file. The objectives of this project are:

1. Playing an audio file.
2. Loading an audio file from the server.
3. Providing a way to find files from the user storage.
4. Creating a user account by register form fill up.
5. Providing background data storing facility for every single user using database.
6. Providing user security log in using their user name and password.
7. Providing a history based playlist for every user which can store the information of recent six played audio.
8. Providing a direct access from the play list to the user without opening any file from the directory.
9. Providing auto pop out facility from the playlist when it is full to store the data of recent files.
10. Providing basic audio functionalities like:
    i) Play and Stop audio
    ii) Pause and Resume audio
    iii) Volume controls: Mute and unmute and a volume control slider.
11. To show the currently playing file name on the top and also the playing time and total time of the file and duration visualizer time slider.

2. INTRODUCTION

"Java Audio Player" is a basic media package developed by using the high level programming language **JAVA** on **WINDOWS** platform. But the result of the project is platform independent. The basic versions of **JAVA** programming language were used to implement the control functions. The user interfaces were designed using the latest version of **JAVA scene builder** in **NETBEANS IDE.**

Every user of the project has separate playlist saved in user directory. The user registration information was saved in database table. The database was created using **MySQL** database management system. Then it was connected to the main project using a jar file of java named **JDBC (mysql-connector-java-8.0.25)** which provided the functionalities to connect the database to the program data. Thus the registration data were stored. While logging in the provided log in data from the user were checked & examined the validity of the data from the database. Then the other information was retrieved from the data table user information. The API used in the project to use the SQL was **JAVA SQL API**.

The User Interfaces were designed using the **JAVA SWING API** and NETBEANS scene builder. Java swing is the most popular and mostly used API for design purpose. The designing codes were auto generated while using the NETBEANS scene builder. To give the UI a look and feel a jar file named **JTATOO** was used. The other icons, images and UI components were collected from different resources. YouTube and other internet platforms helped the most to fix the problems I faced in GUI design.

The main part of the project, playing audio was performed by using the **JAVA SOUND API**. But the drawback of this API is it can play audio of only wav format so a conversion form other format into wav format was planned to be implemented. So that I can play wav format audio as well as other formats like MP3. However the java sound API takes the audio file as a clip and process it to play back. It uses a line listener to detect the completion of a clip playing back. The other functions were interrupted in the thread where the play back function was called.

JAVA UTIL API provided the time showing facilities of the audio clip while playing. It gives the total time and current time of a clip.

The Float Control API which is a part of java sound was used to control the volume of the sound passing the Decibel gain value by a volume slider.

3. IMPLEMENTATION

At the first stage of implementation of the project the following steps were followed as the strategy:

1. Design Login, Registration form and functionalities including database connection
2. Designing the player UI
3. Implementing functional parts, player and timer classes
4. Testing the output for all the input cases and fixing bugs

Step1:

The first step was to design a login and register form which was accomplished by using the Java swing API and scene builder of NETBEANS IDE. So the code for the design section was auto generated. The components of the log in form are:

i) A user name text field for user input
ii) A password field for password input
iii) A login button and a cancel button
iv) A Label to take the user to the registration page if user wants to create a new account.

The functional parts were added as the action of the 2 buttons and as the action of the 'Click here to register' label. First I will describe the click action of the label.

Figure (3.1): Log in Form

If the user doesn't have a registered account there will be no data for him in the database. So the user must go to the register form and fill the required fields with his valid information. The user name must be unique which will be checked in the log in stage from the database. After clicking the register button the data will be saved in database if all fields are filled and the user name is unique. Then the user will be able to log in.

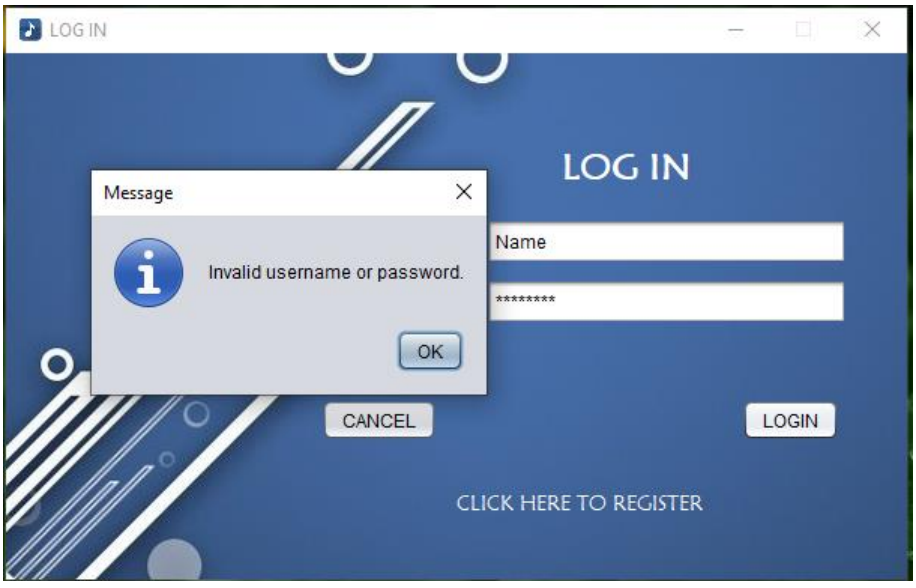

Figure (3.2): Log in Failed message

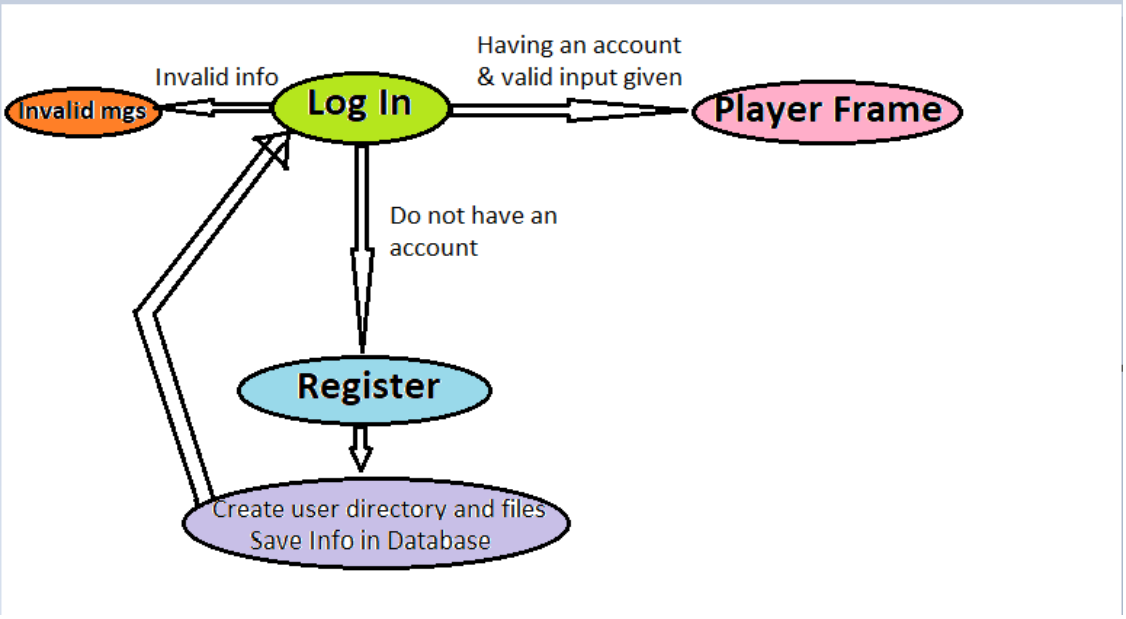

Figure (3.3): Log in and Registration cycle

In the register form there are 5 fields for taking user information: **name, user name, email, phone number** and **password**. When all the required fields are filled by a new user with a unique user name he will be able to register by clicking the register button. He also can cancel the process before clicking register button. After the registration he can go to the log in page again by clicking on the '**go to log in**' label.

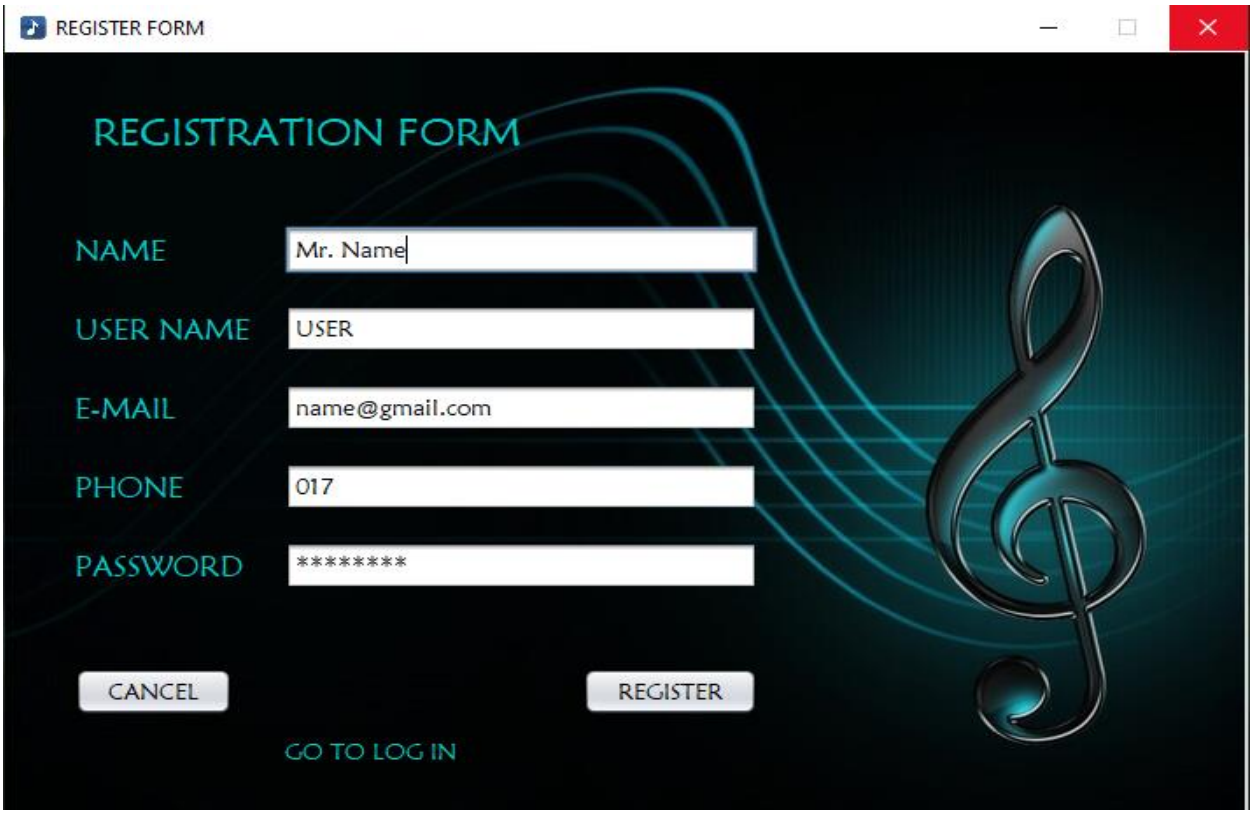

Figure (3.4): Registration Form

When register button is clicked it will create a user directory and two text file in the directory. The name of the directory will be the (user name + phone number) string. As the user name is unique each and every directory will be unique in this way. The texts files are named as paths and names which contain the playlist information of the user, the name of the song played and the location of that audio file. Then the total seven data will be stored in the database in the table named ‘table data’.

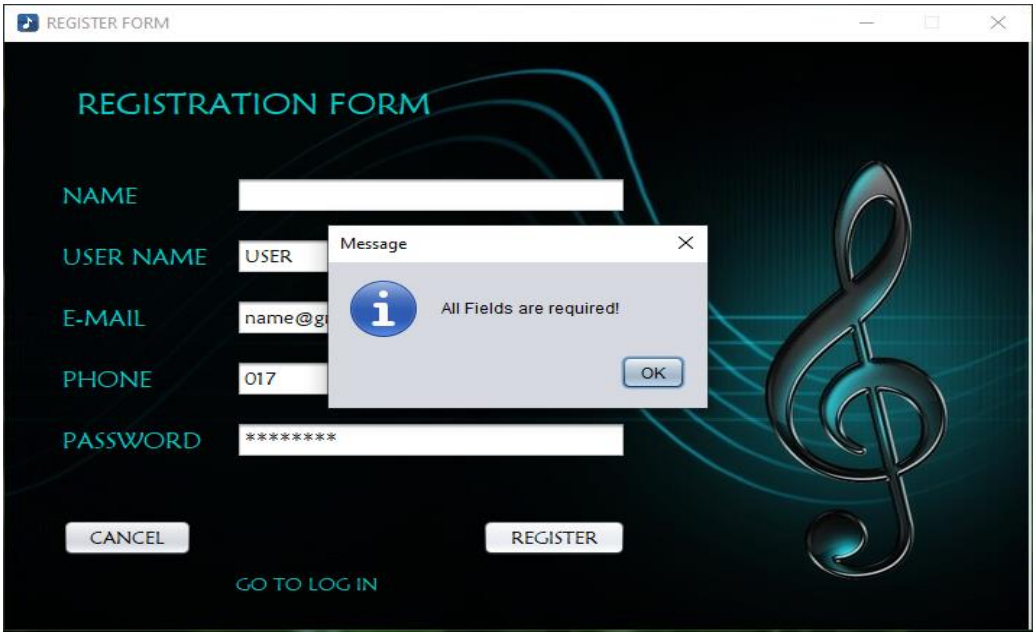

Figure (3.5): Blank Field Error of registration

If the user provides any existing user name or any of the required fields are not complete then the registration will not be done rather a message will be shown according to the error.

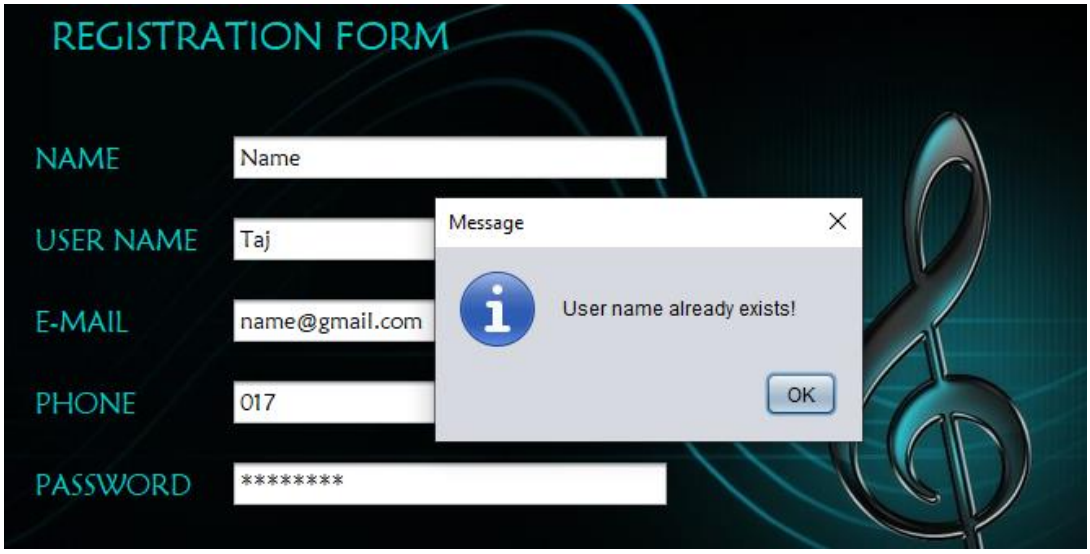

Figure (3.6): Existing username error

In the background of the log in and registration mechanism there are two vital classes, one for the database connection and another one is for making table in that database where the user data will be stored.

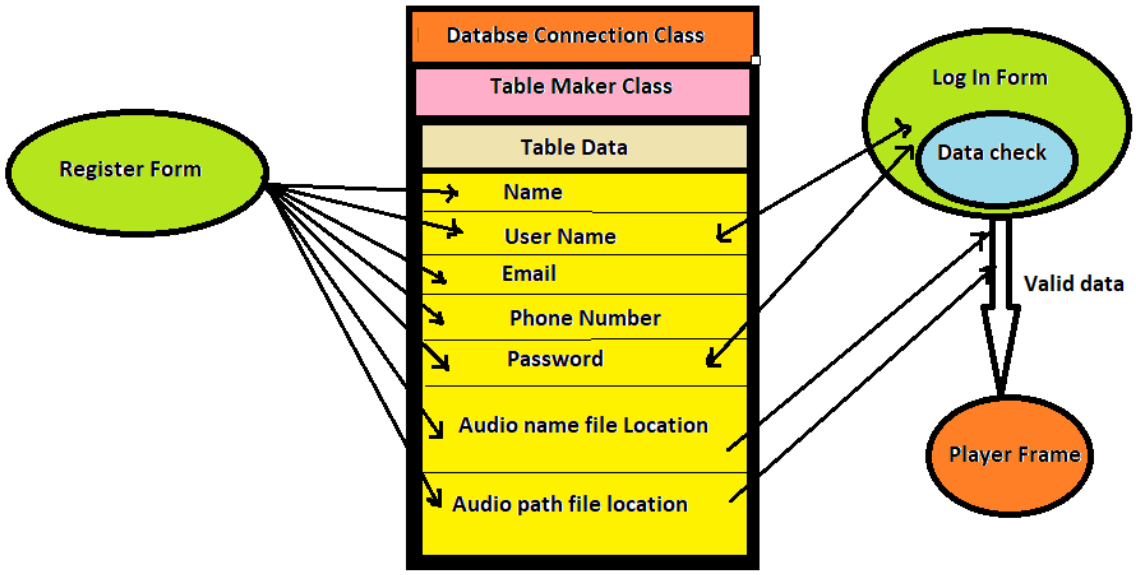

Figure (3.7): The data communication process of database table with Register Form and Log in form.

Now the user can log in with his unique user name and password. When the log in button is clicked the provided user name and the password will be checked with the data of the user database. If there is a match and validity the file paths of the particular user playlist will be retrieved from the user information database table. Then the user name, name file location, path file location will be sent to the player form to initialize the player.

Step2:

The second step was to design the player user interface. The components of the player frame are:

i) A label to show the audio name currently playing
ii) A label to show different images in different mode
iii) Two label for the total playing time and current time
iv) A time duration slider
v) A volume control slider
vi) A set of buttons: Open button, Play and Stop button, Pause and Resume Button, Mute or Unmute button and Playlist view button.

The Play/Stop button and the Pause/Resume buttons are initially disabled as there is no audio loaded. Similarly the sliders will be kept at an initial position and the time labels show '**00:00:00'** time.

Figure (3.8): Initial state of the player frame

⇨ The **view playlist button** will show the previously added audio file names. The user can directly play an audio by clicking one of the items from the list. If user plays an audio from the list the Stop and Pause buttons will be enabled. If the view playlist button is clicked again the list will disappear.

Figure (3.9): Playing audio from play list

⇨ The **Open button** will enable the user to choose an audio file from the directory. It also checks the validity of the file format. If the user click of an audio file from the directory and open it the play back will start and the audio is listed in the play list as the last element.

In case the list is full the first one of the list will be popped out and the recent one will be added.

If the user again clicks on the open button it will open the last opened path from where the user chose an audio and played.

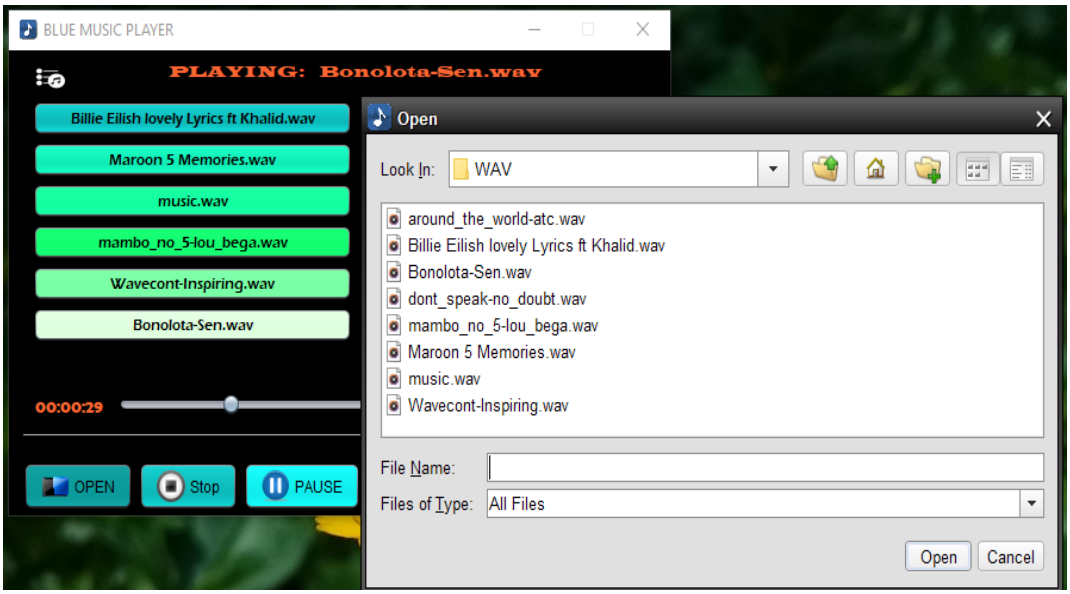

Figure (3.10): Open button action file chooser opened and new audio added

To the list

⇨ While an audio file is playing the play button shows '**Stop**' icon and text and the pause button shows '**Pause**' icon and text. The time labels shows the total time and current time and the slider starts moving. If the '**Pause**' button is clicked the play will be paused and the slider and current time labels will also be paused.

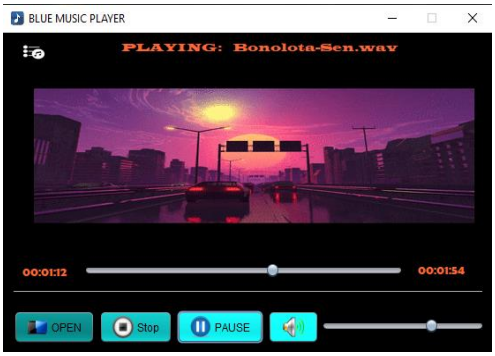

Figure (3.11): Audio is playing

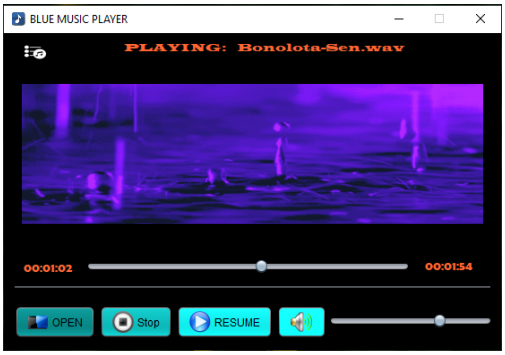

Figure (3.12): Audio is paused

⇨ If the **Stop button** is clicked the playing audio will be stopped but still loaded in the player. The Stop button will show **'Play'** text and icon. If the audio playing reaches at end the player will go to the state same as the stop method do to it.

Figure (3.13): Audio play back is stopped but the audio is still loaded

⇨ The **volume button** will mute the sound and unmute it and the volume slider will control the volume.

Figure (3.14): Audio is playing but muted

Step 3:

In this section the **functional parts** of the audio player were implemented. The player frame holds the controlling functions but there are two more vital classes in the background of the player frame. They are:

i) The Audio Player class
ii) The Playing Timer class

The audio player class is used in a thread in the play back method of the player frame. The playing timer class also runs along with the play back thread.

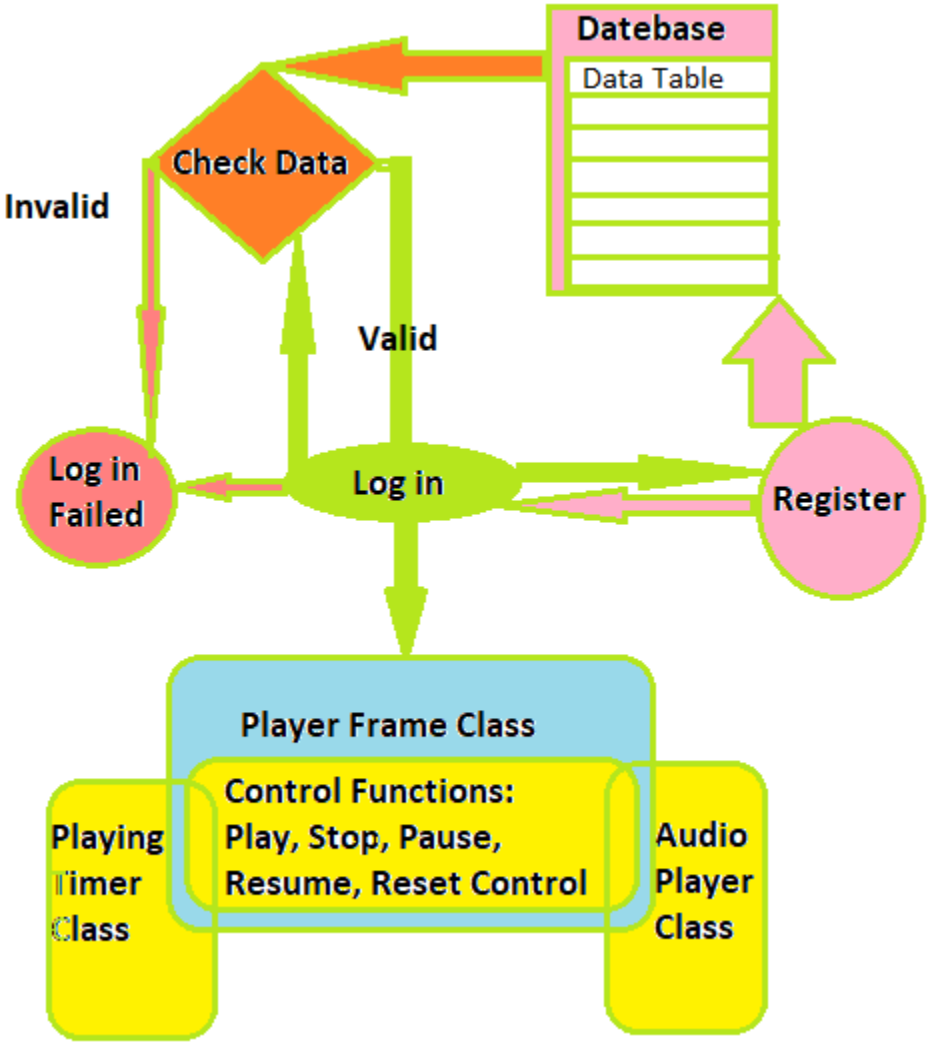

Figure (3.15): Relational flowchart of the functional part

The functionality of the Log in and Register part have already been discussed. Now the control functions of the player will be discussed in detail:

- ⇨ **The Open Method**: The open method simply brings a file chooser in the user interface. When the user chooses a file it also checks the format of the audio file. If the file is a wav file it continues and goes to play back method. Otherwise it shows Unsupported file Exception.

- ⇨ **The Play Back Method**: In this method a runnable thread named Play back thread is executed. The path of the file chosen by the user is sent to the Audio player class and the audio starts playing.

- ⇨ **The Pause and Resume Method:** If the player is in the resume state the pause button will show Pause text. If it is paused the button will show Resume text and icon. If clicked on the pause button the audio will stop playing but not break the thread. The timer and the slider will also be paused.
  Then if the Resume button is clicked the audio, timer and slider will start again from the position where it was stopped.

- ⇨ **Stop Method:** The stop method closes the player but the file is still loaded. The timer and the slider are reset.

- ⇨ **Mute-Unmute and Volume Control Method:** The slider value set the Decibel value of the gain of the Float control mechanism. If the slider value is 0 then the audio is mute otherwise unmute. The volume button does the same. It simply mutes and unmutes the audio.

Al the functionalities are visualized previously in the design section. Now the total process is depicting by a flow chart below:

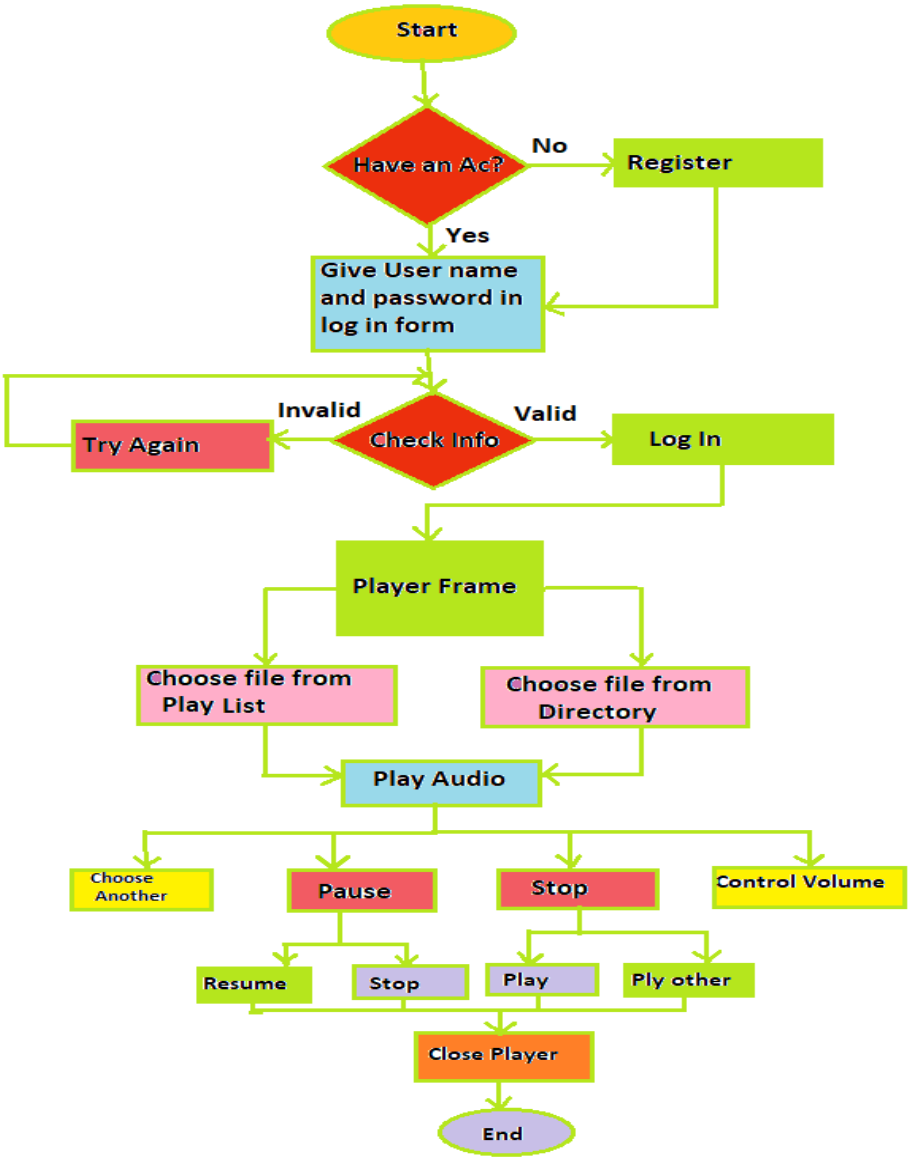

Figure (3.16): Flow chart of the whole process

Step 4: After completing all the 3 sections the outputs were checked for all the states. There were some bugs like: Double loading of audio, Play without stopping the loaded one, Button state unchanged. All the bugs that were identified were fixed. Still there could have some unidentified bugs.

4. Target vs. Actual Accomplishment

The main target of the project was to develop a platform where an audio file can be played and controlled which was achieved. But the drawback is it can only play audio file of wav format as I used the basic sound API of Java. So I wanted to add an audio format converter with the main player which can convert an mp3 file into a wav format before playing it. Thus it could be able to play both wav and mp3 format file.

Secondly there was some optimization needed in the playlist feature. For example, one user would have more than one play list and he could arrange it as his choice and wish. There was a target to add auto play option based on the play list which was not accomplished.

However the main goal of the project is accomplished providing the basic features and functionalities of an audio player, also the data security and usability at individual user mode.

5. Risks and Issues

There are some risks and issues of the audio player considering the data security system and the performance of the device on which it will be played.

Firstly, though the data security system of the MySQL database management system is strong enough it could be manipulated in this project. Because the data checking and hiding system used in this project was not at professional level.

Secondly the play back method works in a runnable thread. If the device works very slow it cannot match the time of the thread with the sequence of commands. Thus it could give some unexpected output like: double audio loading, unstable timer or some other errors. Sometimes the program can get crashed.

6. Discussion and Conclusion

This project is the very first advanced programming project for me. So at the first stage I faced a lot of difficulties. There were some little obstacles in every step from installing the ide to testing the final outcome. Luckily I could overcome them with the help and direction of our course teachers and other resources on the internet. The basic API of java for audio playing and controlling is java sound API. It has a set of functionalities to control the audio and the float gain. But the resources for java swing and for some other basic API like java sound was very rare on the internet. Though there were many resources for the latest java libraries but I think the basic versions are better than them for learning purpose. At the time of implementing new features there were some new problems arise every time. So I checked the consequence of the newly added feature on the whole program and that was a time consuming task. But finally the whole project worked quiet perfectly with all its functionalities and features.

7. References

1. https://docs.oracle.com/javase/tutorial/
2. https://www.javatips.net/
3. https://www.javatpoint.com/
4. https://stackoverflow.com/
5. https://www.geeksforgeeks.org/
6. https://www.youtube.com/watch?v=i5UG6ACtnEg&list=WL&index=2