

**Spring, 2019-2020**

**SE2224 - Software System Analysis**

**Term Project Report**

|  |  |
| --- | --- |
| **Project Name:** | **SECOTIFY** |
| **Student Name:** | **Serkan Koçoğlu** |
| **Student No:** |  |
| **Department No:** |  |
| **Course Section No:** | **SE2224** |

**Table of Contents (Do not change the Section Names!)**

**[1.](#_Toc36757855)****[Introduction:](#_Toc36757855)** [2](#_Toc36757855)

**[2.](#_Toc36757856)****[System Request:](#_Toc36757856)** [3](#_Toc36757856)

**[3.](#_Toc36757857)****[Workplan:](#_Toc36757857)** [3](#_Toc36757857)

**[4.](#_Toc36757858)****[Feasibility Analysis:](#_Toc36757858)** [3](#_Toc36757858)

**[4.1.](#_Toc36757859)****[The Technical Feasibility:](#_Toc36757859)** [3](#_Toc36757859)

**[4.2.](#_Toc36757860)****[The Economic Feasibility:](#_Toc36757860)** [3](#_Toc36757860)

**[4.3.](#_Toc36757861)****[The Issues that Affect the Two Kinds of Feasibility:](#_Toc36757861)** [3](#_Toc36757861)

**[5.](#_Toc36757862)****[Requirements Definition:](#_Toc36757862)** [3](#_Toc36757862)

**[5.1.](#_Toc36757863)****[Functional Requirements:](#_Toc36757863)** [3](#_Toc36757863)

**[5.2.](#_Toc36757864)****[Nonfunctional Requirements:](#_Toc36757864)** [3](#_Toc36757864)

**[6.](#_Toc36757865)****[Functional Model:](#_Toc36757865)** [3](#_Toc36757865)

**[6.1.](#_Toc36757866)****[Activity Diagrams:](#_Toc36757866)** [4](#_Toc36757866)

**[6.2.](#_Toc36757867)****[Use Case Diagrams:](#_Toc36757867)** [4](#_Toc36757867)

**[7.](#_Toc36757868)****[Structural Models:](#_Toc36757868)** [4](#_Toc36757868)

**[7.1.](#_Toc36757869)****[Class Diagrams:](#_Toc36757869)** [4](#_Toc36757869)

**[8.](#_Toc36757870)****[Behavioral Models:](#_Toc36757870)** [4](#_Toc36757870)

**[8.1.](#_Toc36757871)****[Sequence Diagrams:](#_Toc36757871)** [4](#_Toc36757871)

**[8.2.](#_Toc36757872)****[Communication Diagrams:](#_Toc36757872)** [4](#_Toc36757872)

**[8.3.](#_Toc36757873)****[Final Version(s) of the Graphical User Interface(s) of the System:](#_Toc36757873)** [4](#_Toc36757873)

**[9.](#_Toc36757874)****[Process Modeling:](#_Toc36757874)** [4](#_Toc36757874)

**[9.1.](#_Toc36757875)****[Data Flow Diagram (DFD):](#_Toc36757875)** [4](#_Toc36757875)

**[10.](#_Toc36757876)****[Data Modeling:](#_Toc36757876)** [4](#_Toc36757876)

**[10.1.](#_Toc36757877)****[E/R Diagrams:](#_Toc36757877)** [4](#_Toc36757877)

**[11.](#_Toc36757878)****[Risk Assessments:](#_Toc36757878)** [4](#_Toc36757878)

**[12.](#_Toc36757879)****[Conclusion and Future Work:](#_Toc36757879)** [4](#_Toc36757879)

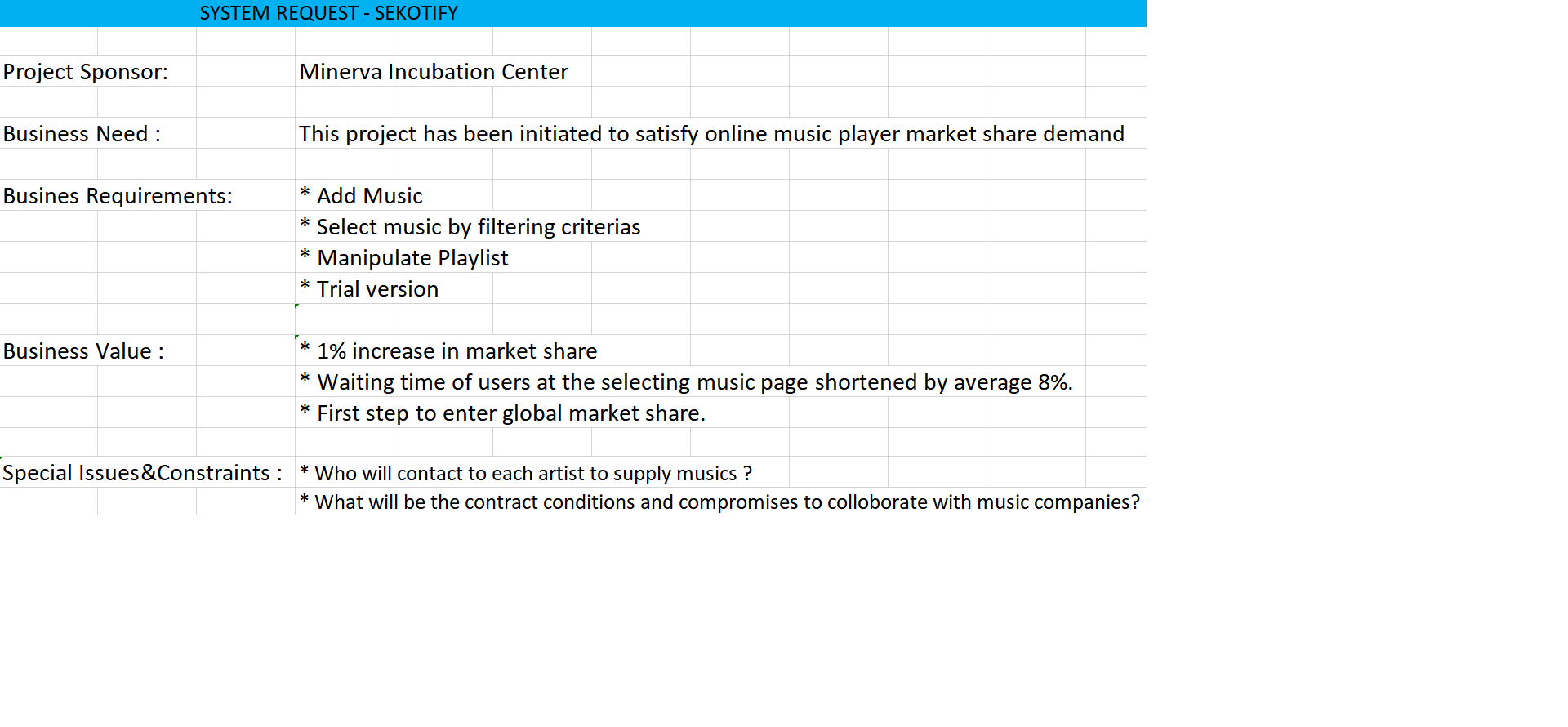
**[13.](#_Toc36757880)****[Appendix:](#_Toc36757880)** [4](#_Toc36757880)

# **Introduction:**

Introduce your project here.

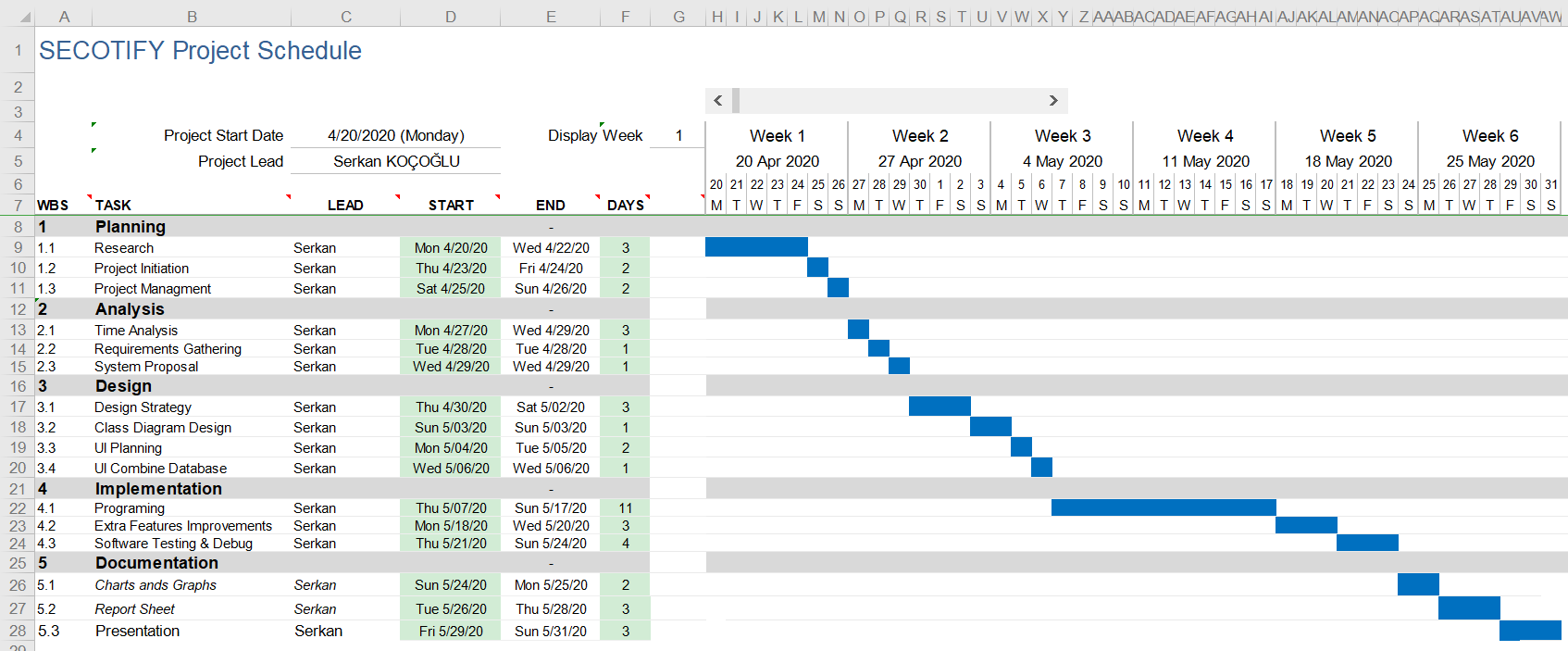
Sekotify is a user friendly and global music player application. This project is realized because of online song listening applications are not able to meet demand. Application requires membership. Trial version released with no advertisement. User able to find songs by filtering through song type and artist ! User can add, delete or display user’s personal playlist. First release is as a desktop application, other platforms are coming soon.

# **System Request:**



# **Workplan:**

Put your Project Work Plan here. You can design it as a Gantt Chart.



# **Feasibility Analysis:**

## **The Technical Feasibility:**

* List three things that can influence the technical feasibility of the system.
* Not a big concern because does not require a foreign technology to realize.
* Same projects experienced before, this lowers risk.
* Easy to crash because of more users than expected.

## **The Economic Feasibility:**

* List three things that can influence the economic feasibility of the system.
* Artists sell or hire their songs through contracting is a basic expense.
* Free for small time interval can beneficial for advertising.
* Subscription price must be set through that country’s currency and purchasing power increases profit rate.

## **The Issues that Affect the Two Kinds of Feasibility:**

* How can you learn more about the issues that affect these two kinds of feasibility?

For technical feasibility : Searching documentations about relevant technologies and finding person who had worked at same project before. Furthermore , getting counseling from companies which have already managed to use that techniques .

For Economic feasibility : Searching market share and tax shares through our rival company’s country tax announcements .For example people can make economical feasibility research through TÜİK in Turkey to find market shares by statistically.Researchers must prevent from unrealistic data.

# **Requirements Definition:**

Requirements specify a set of features that the system must have. **A functional requirement** is a specification of a function that the system must support, whereas a **nonfunctional requirement** is a constraint on the operation of the system that is not related directly to a function of the system.

Write the Functional and Nonfunctional Requirements of the project.

## **Functional Requirements:**

+ The system will update song list automatically after buying or hiring a new song.

+ The system will allow to create and manipulate playlist.

+ The system will provide selecting song by filtering through artist , genre and title.

+ The system will allow to see most listened songs list.

## **Nonfunctional Requirements:**

+ System should not keep waiting user over 3 seconds(Responsiveness).

+ User should reach every property with minimum clicks.(User friendly interface).

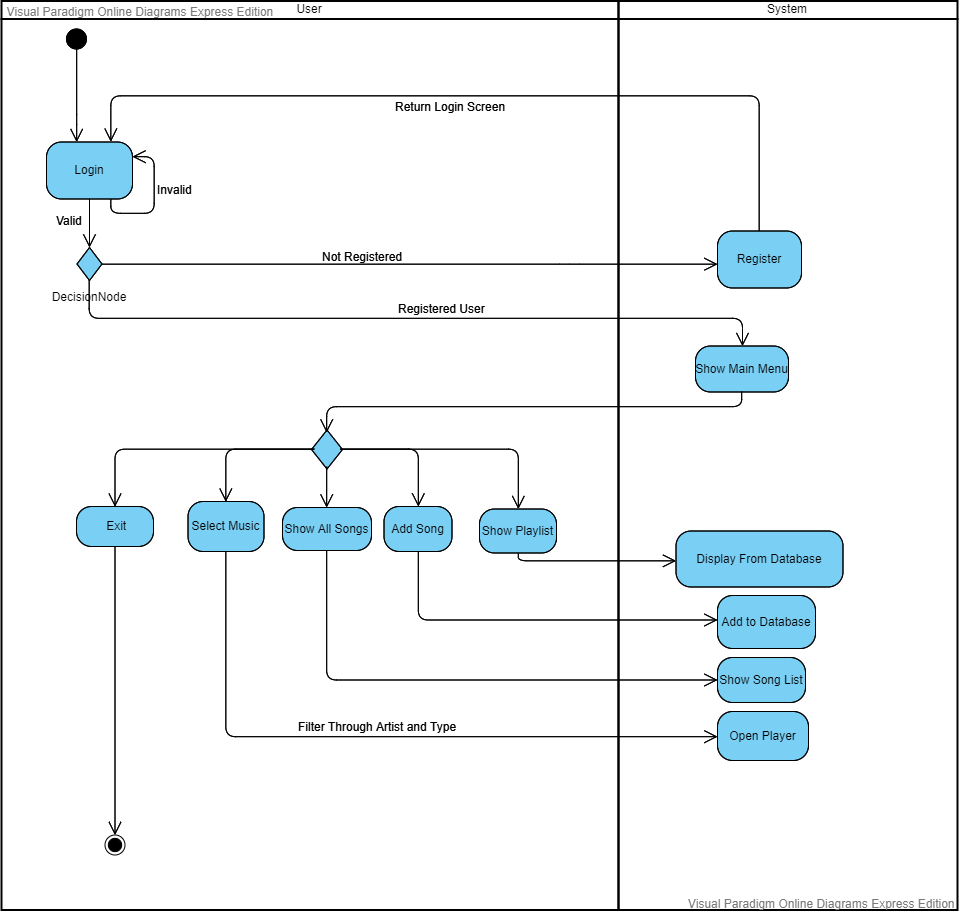
+ User logs will be encrypted and secured.

+System supports the users who have account on that mobile devices or personal computers where the app is to be installed.

# **Functional Model:**

## **Activity Diagrams:**

* Write its explanation as well.



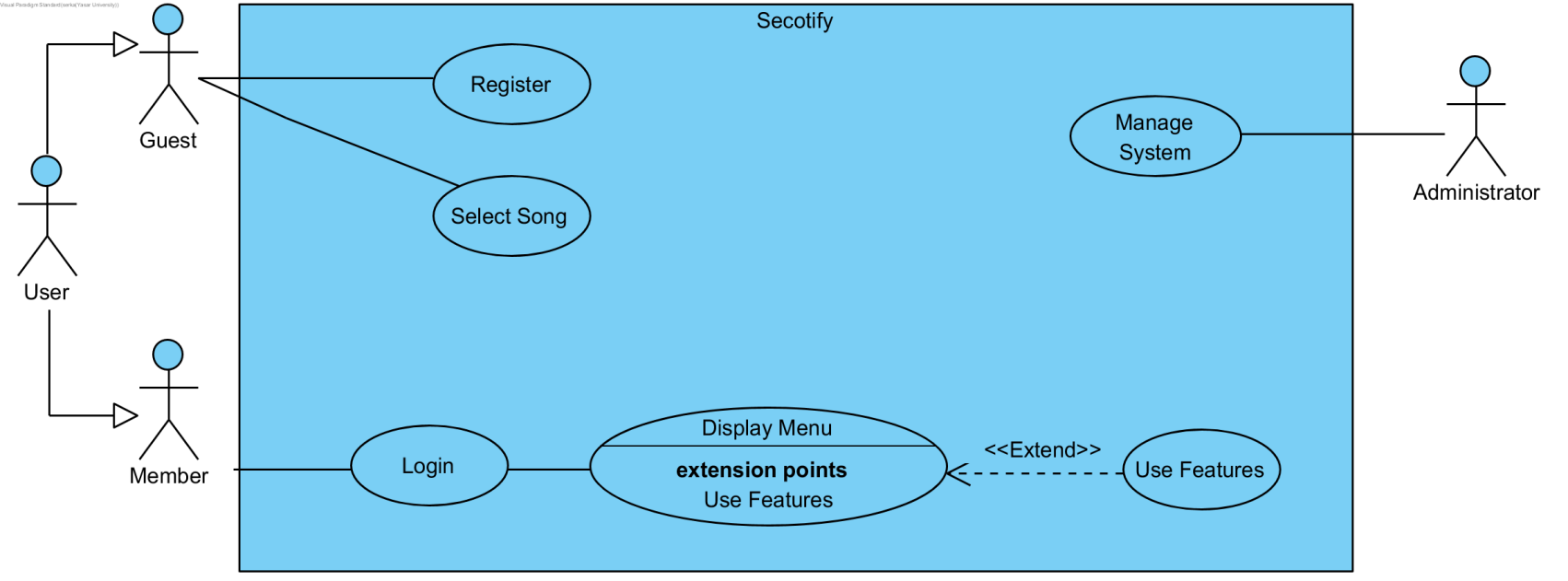
Activity Diagram Explanation

Activity starts with login if user already registered. Application shows main menu if entry is valid. User is able to select music by filtering through artist and genre thus plays selected music on player. User may show all songs and personal playlist. By user’s request system adds song to database. Whenever user wants able to exit from application.

## **Use Case Diagrams:**

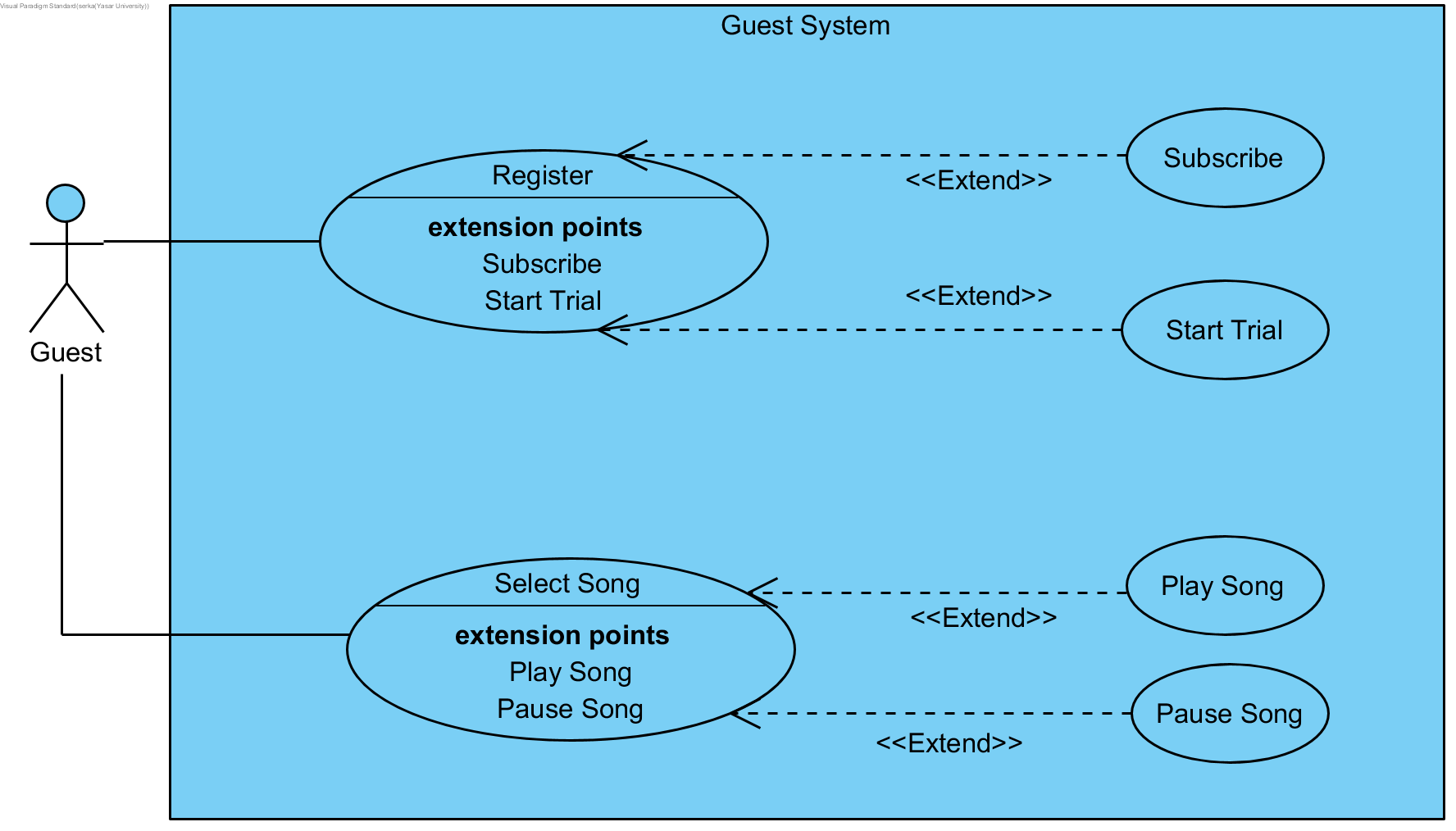
* Write all actors with their descriptions.
* Write all use cases with their descriptions.
* Put the main use case diagram and its sub use case diagrams.

**Main Diagram:**

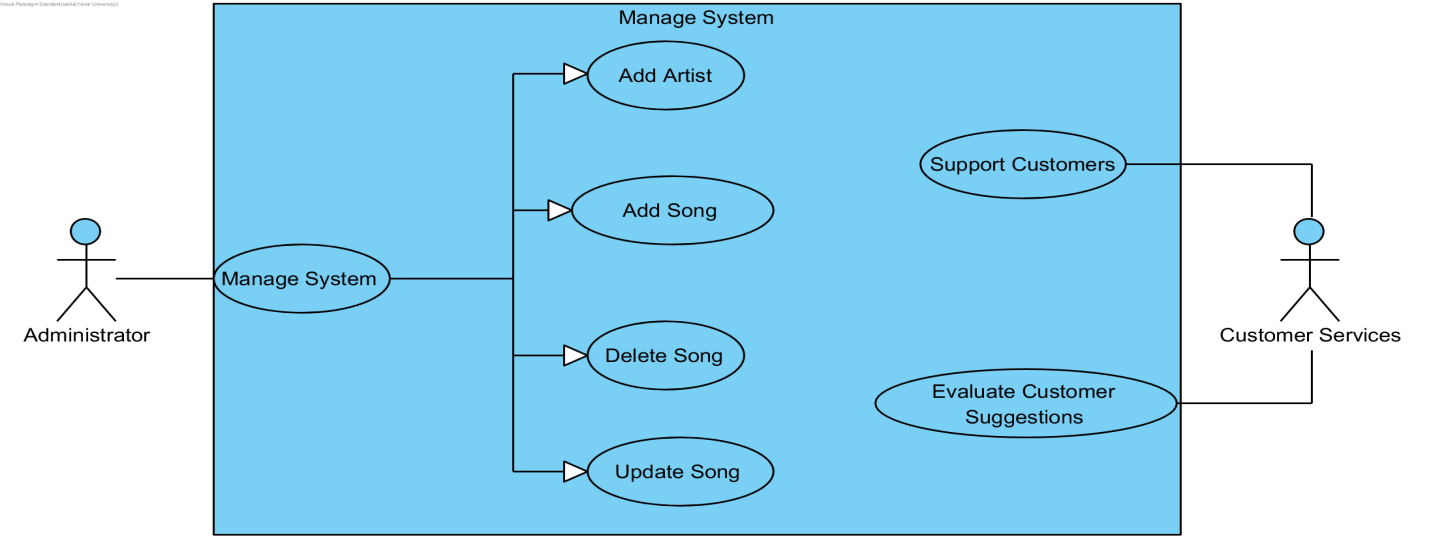


**Sub use cases:**

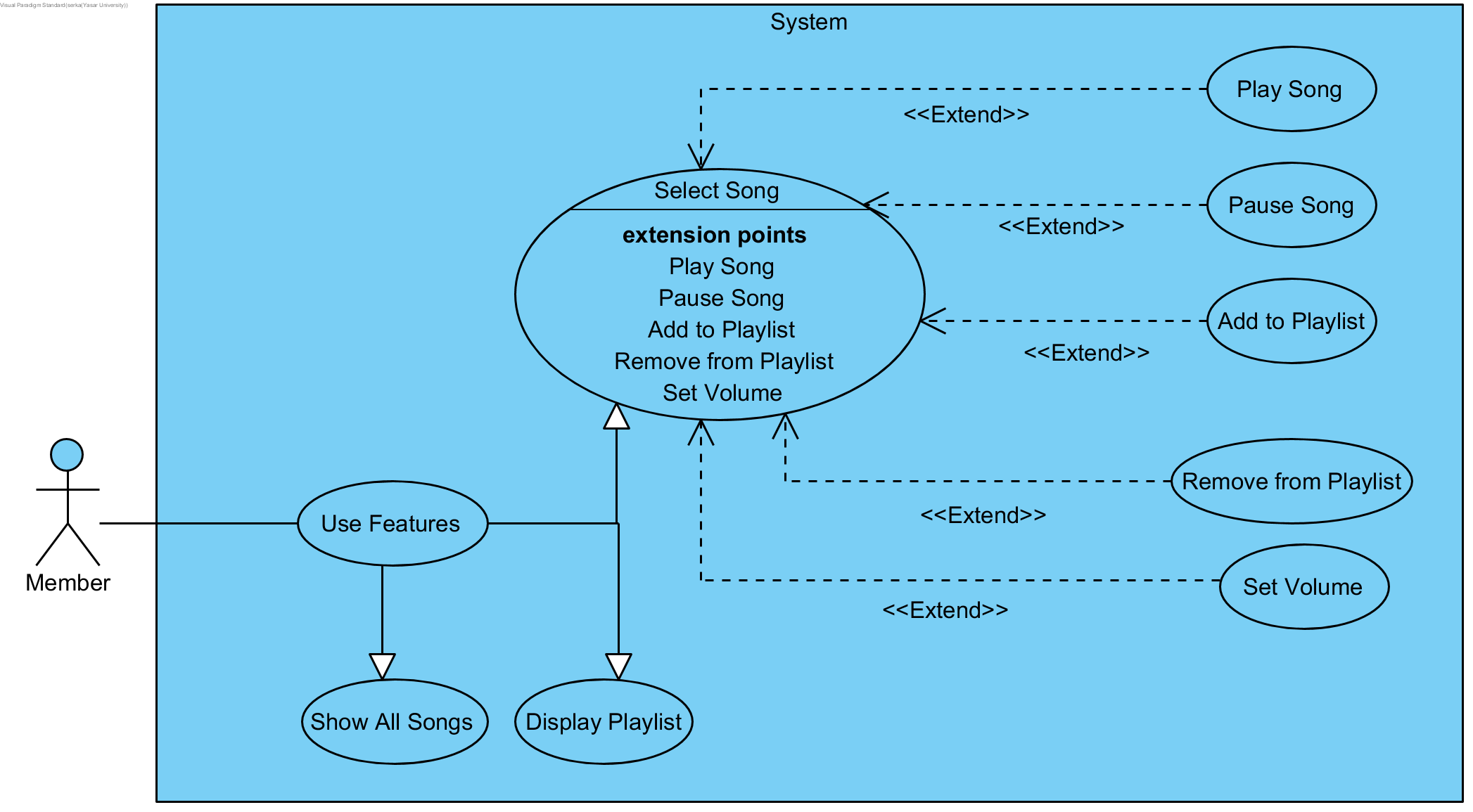
Trial



Administrator:



Features:



Use Case Diagram Description:

**Actors**

1: Guest ; Customer who is using trial version currently.

2: Member ; Subscribed customer.

3: Administrator ; System developer.

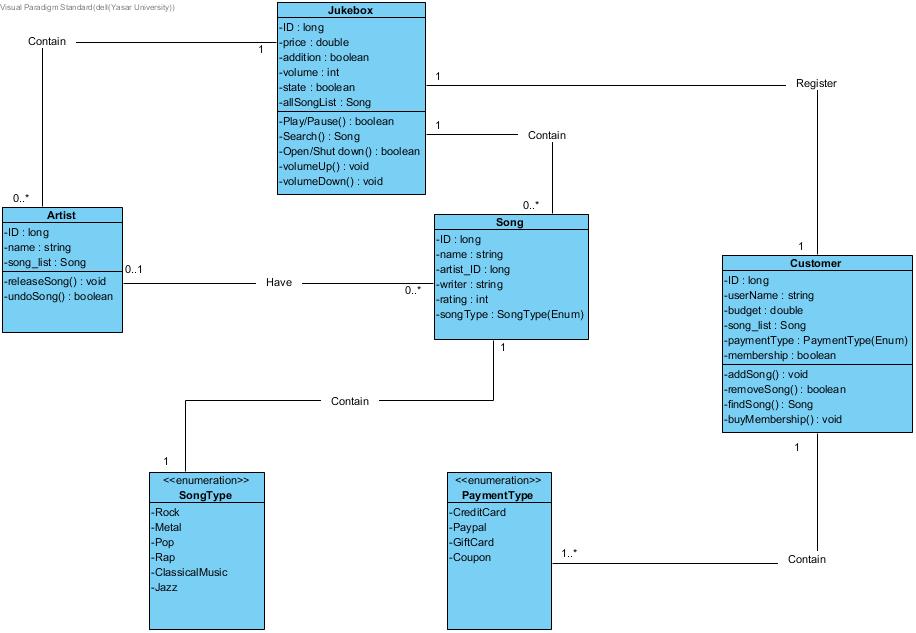
4: Customer Services ; Customer supporter.

SECOTIFY is an application that provides online music.People who wants to listen online music can use trial version or subscribe directly.Although trial version allows to select song and play music , premium version allows to use playlist and reaching all songs additionally.Supplied songs by artists are added to database. The system is maintained as follows, administrator manipulates songs and customer service manages customer relations .

# **Structural Models:**

## **Class Diagrams:**

A class diagram describes the structural aspects of the to-be system. Put your class diagram with the descriptions of all classes.



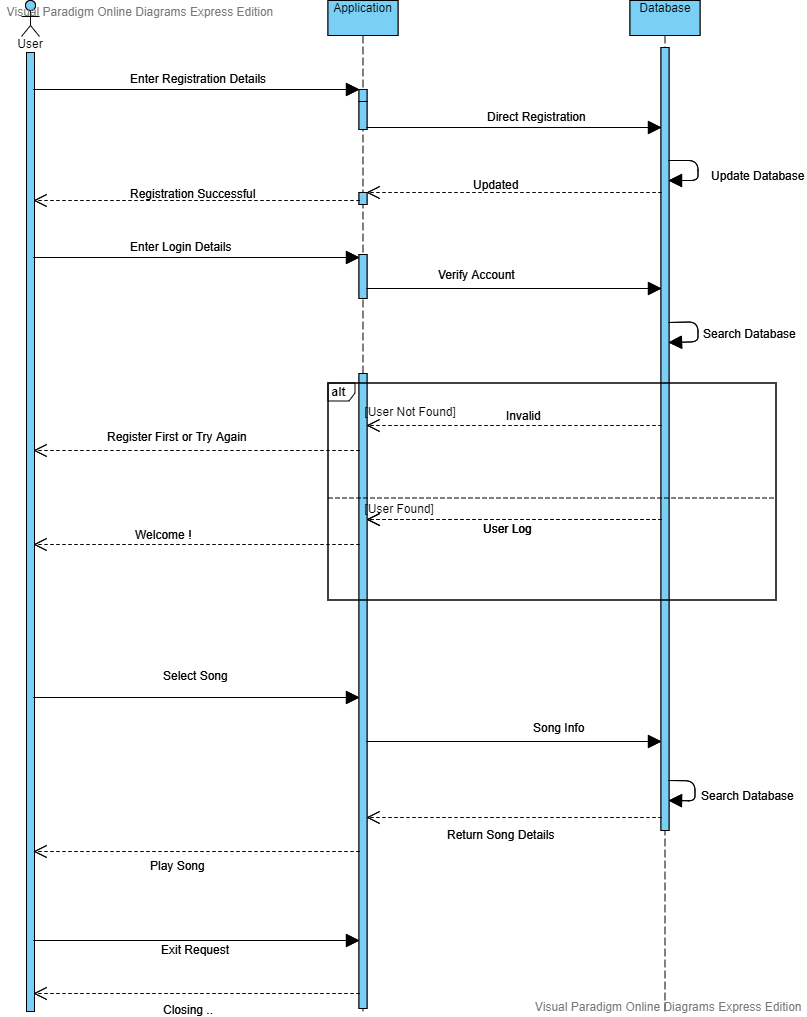
**Class Diagram Description**

This class diagram describes structure of an online music listening application.Jukebox is the main class which has critical attributes and operations such has play pause and search song.Song and artist classes provide Song’s specific information thus Jukebox reach the necessary information by those sub classes. Customer class holds all customer information.Song Type and Payment Type classes are enumerations to provide song types and payments types to their ancestors.

# **Behavioral Models:**

## **Sequence Diagrams:**

* Write its explanation as well.



**Sequence Diagram Explanation:**

Firstly user must register or login to enter the application.

Application directs to database to compare user entry.

If user information is not matched an error message is showed.

If user information matches user enters the system.

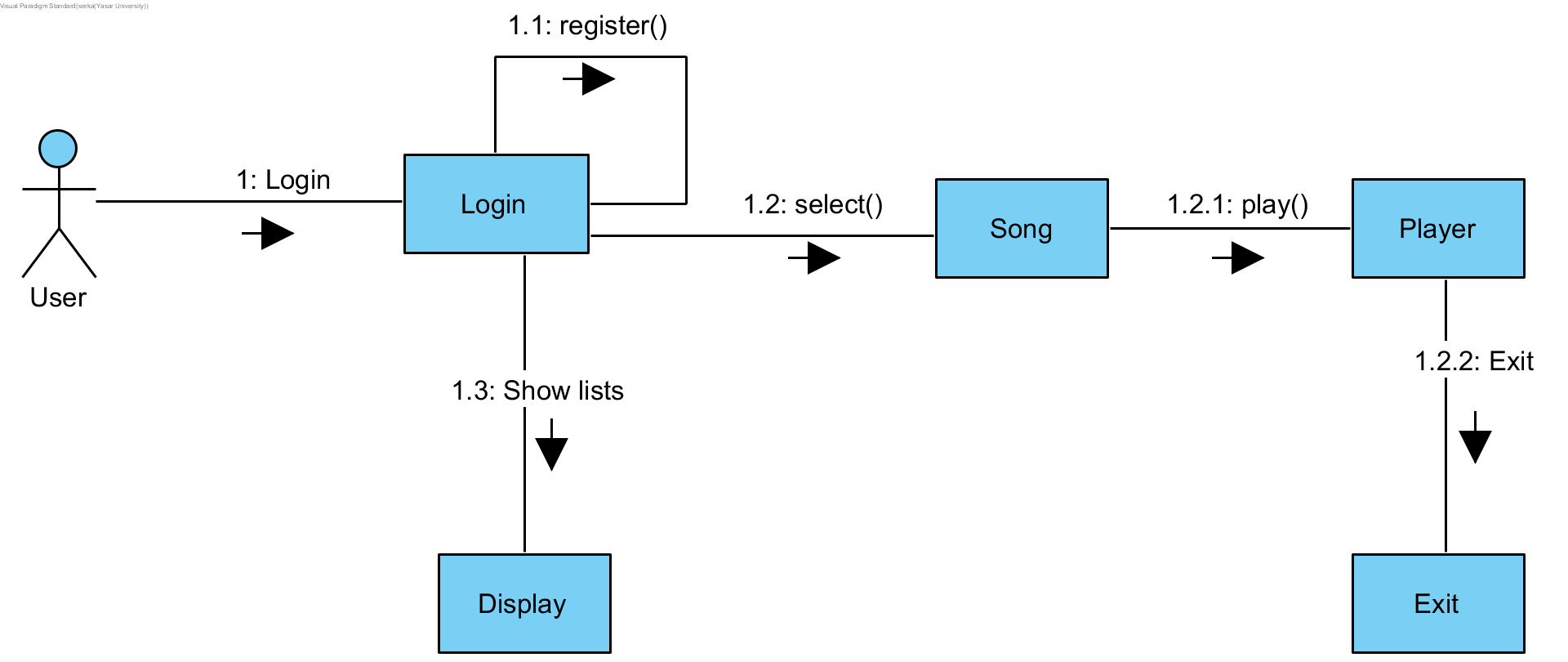
User selects song.

Selected song’s details resend to application from database.

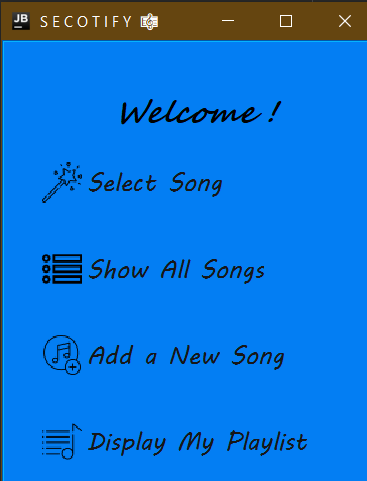
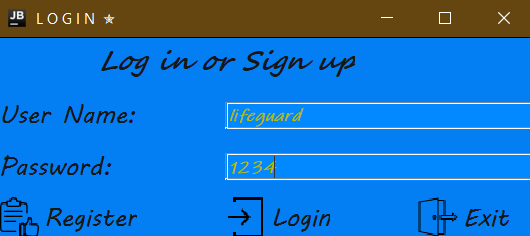
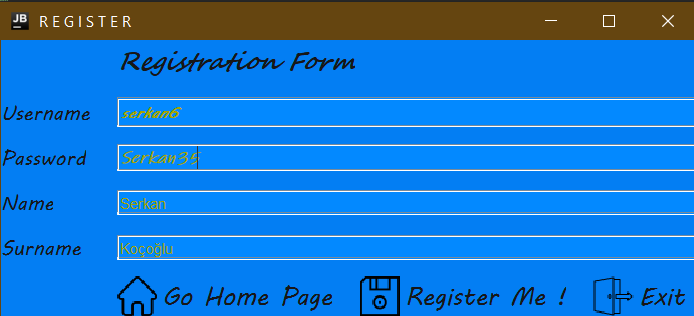
Application plays song.

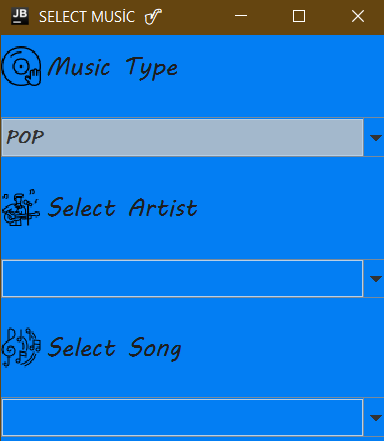
Application closes with user request.

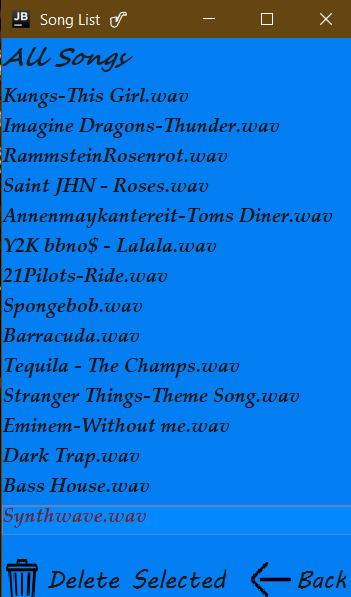
**Communication Diagrams:**

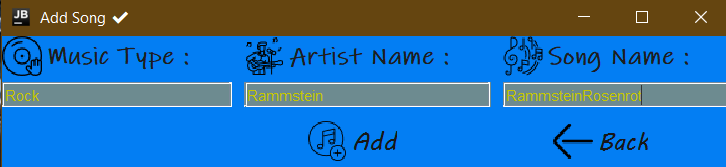
****

## **Final Version(s) of the Graphical User Interface(s) of the System:**

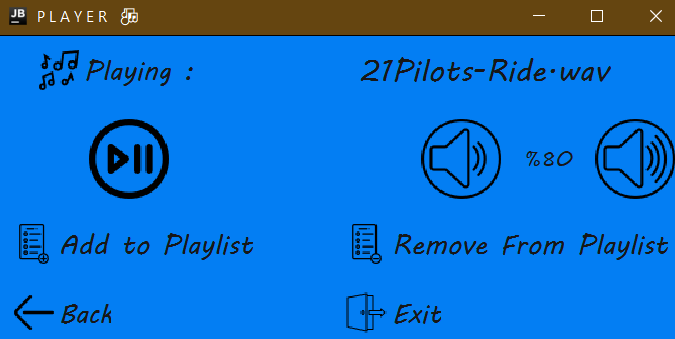
****

****

****

****

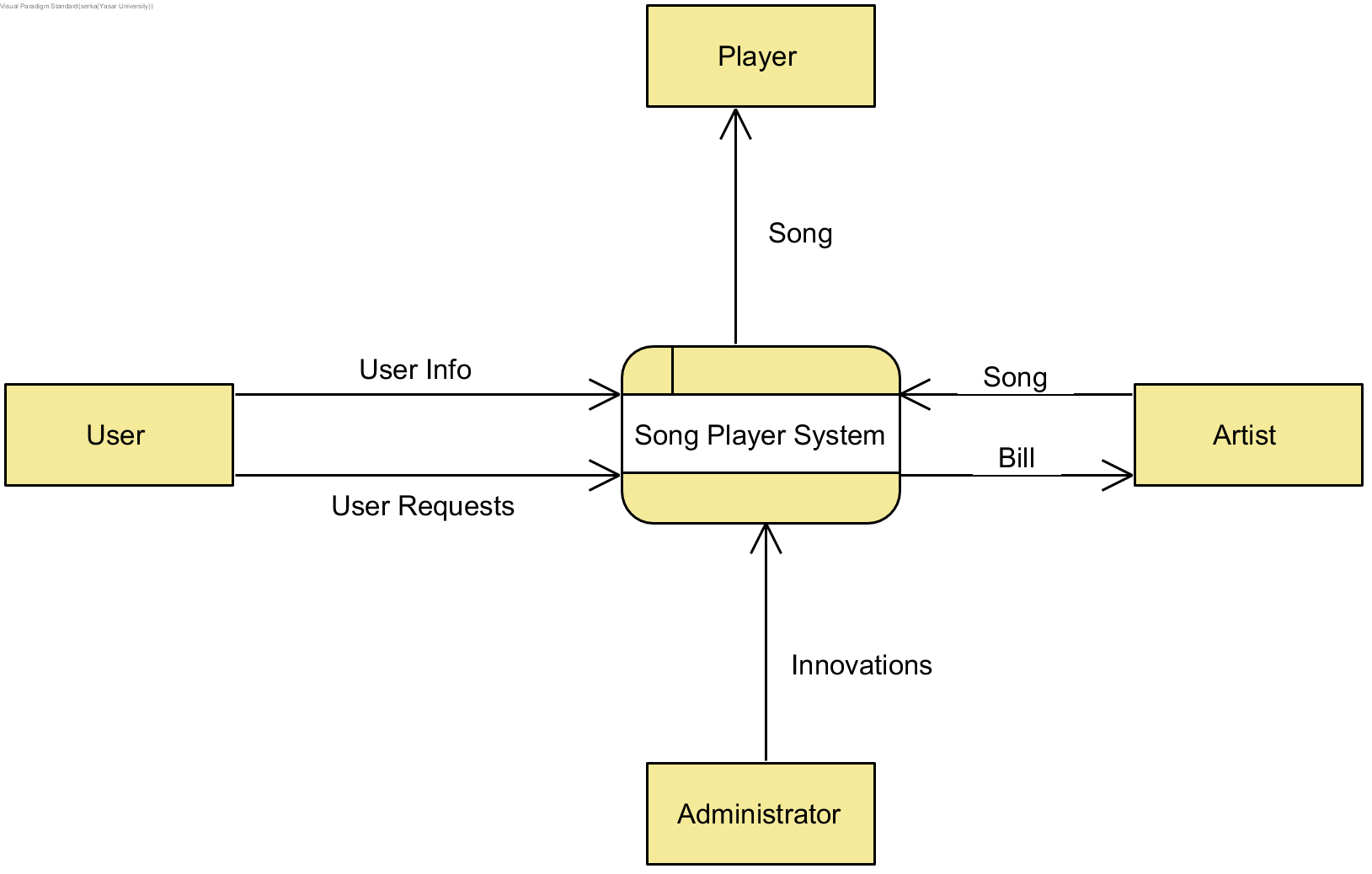
****

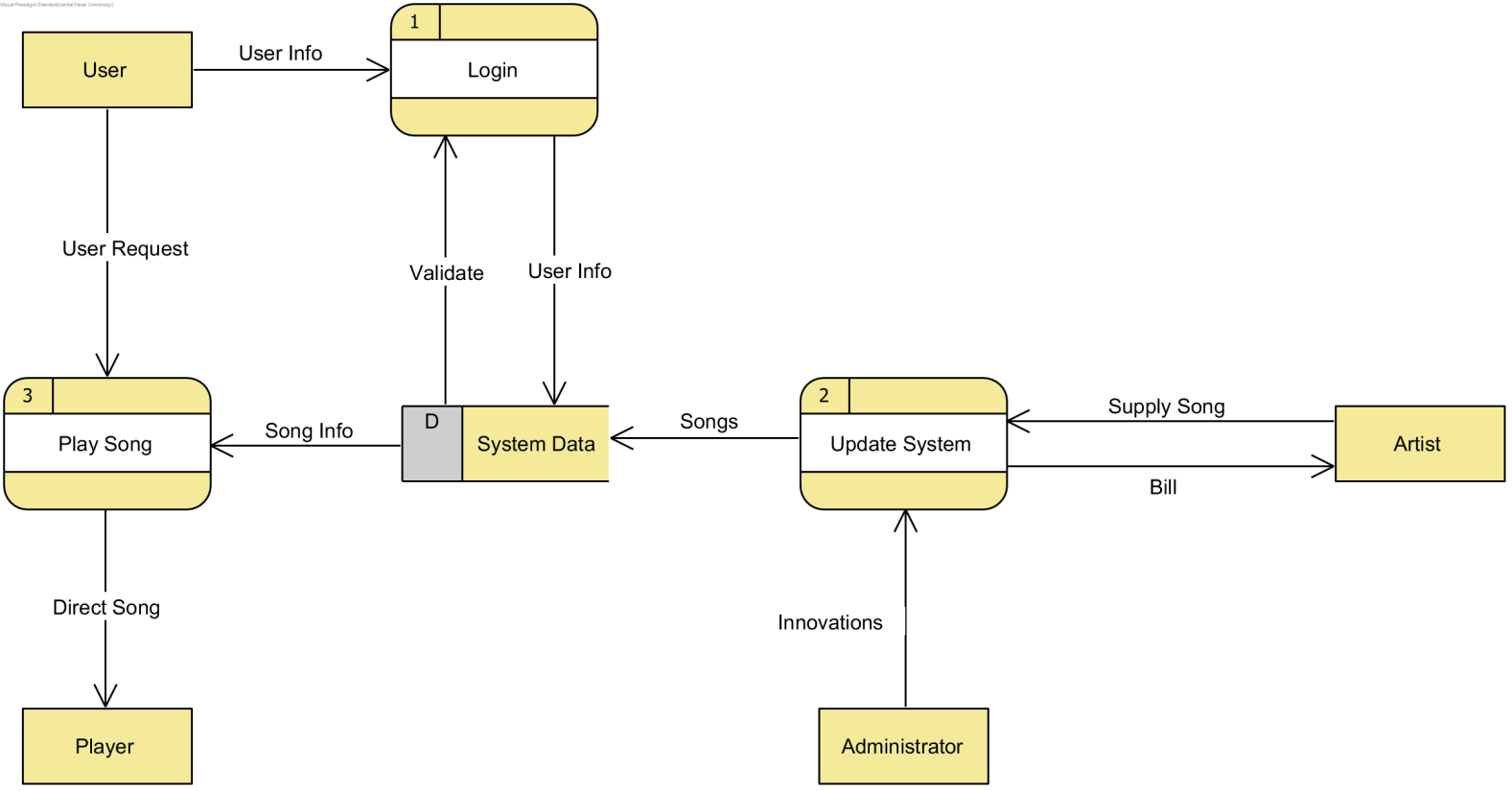


# **Process Modeling:**

## **Data Flow Diagram (DFD):**

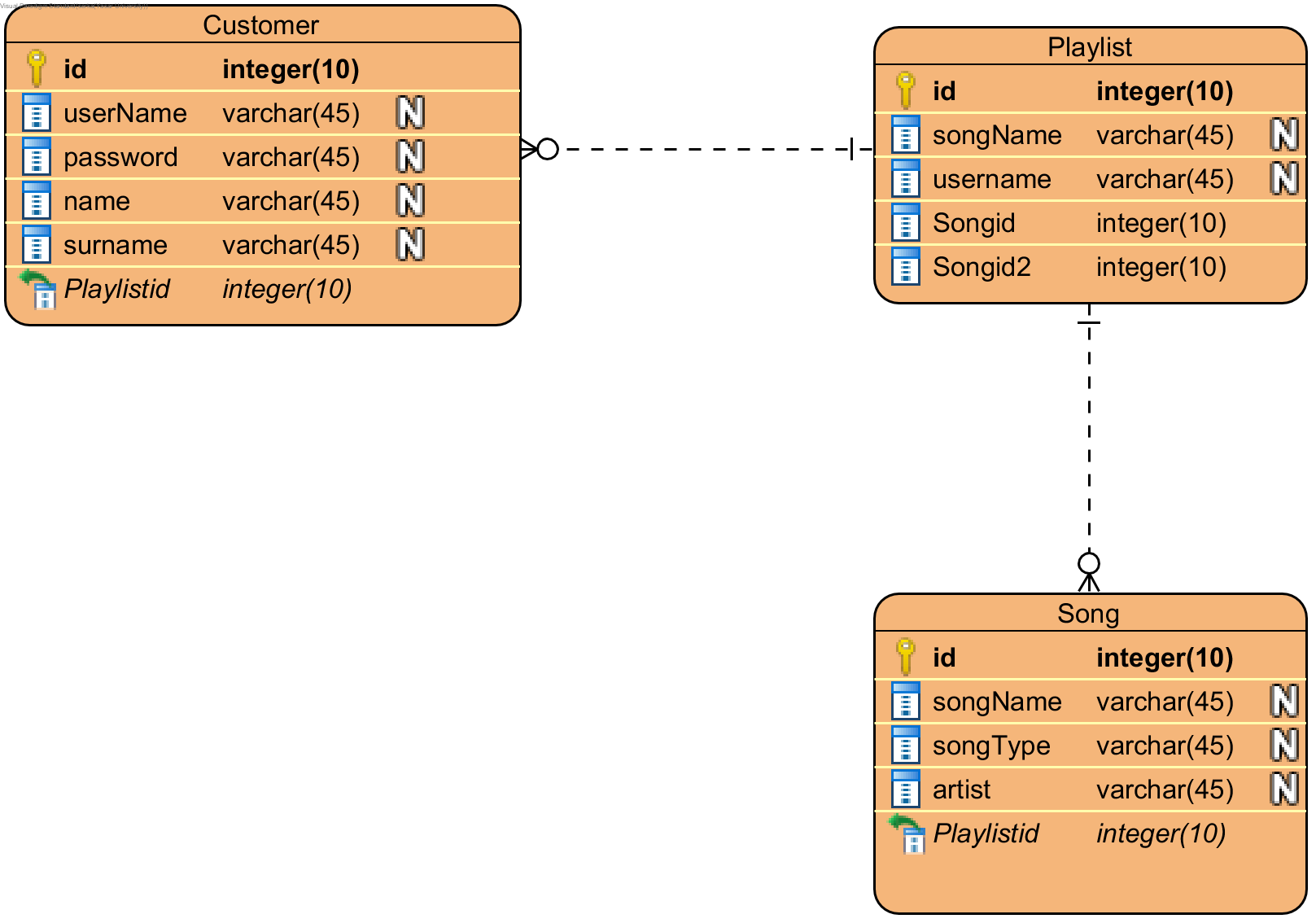
You do not have to go further to level 2 DFD. Level 1 is more than enough.





# **Data Modeling:**

## **E/R Diagrams:**

****

# **Risk Assessments:**

What kind of technical problems did you actually face in this project? Write only the REASONABLE ones. How did you solve/address them? In other words, how did you manage/eliminate these risks?

RISK #1 :

Project team members had no experience about using GUI and database at the same time.

It was very imported to learn fast to reach project deadline so programmers took online helpful tutorials

and team made prototype of the project which is called ’lab works’ thus risk is lowered.

RISK #2:

Biggest project of all team members had realized before.

Complicated systems increase risk rate polynomial so this causes high probability of risk.

Rate of planning phase doubled and strict rules followed.Work hours increased with reward method.

Parts which are not very important was discarded.

# **Conclusion and Future Work:**

Write a conclusion for your document and summarize your work. Also write what kind of additional features/ functionality can be added to your current project.

\*Future study could investigate the most liked features of the app and new requests which are captured by surveys.Changeable theme,setting to most used platforms for future, dark mode, just one click to play personal playlist after entering the system may be likely to appreciated.

\*In summary, this online music listening project realized successfully.Plan phase was much detailed so our team did not mix structure of the code , kept clear and sustainable thus deadline risk lowered successfully.Finally , we would like to mention that online music applications are becoming more and more widespread and we consider it appropriate to work in this field at least in next decade.

1. **Appendix:**

Put all the source codes of your system by organizing them neatly (separate them by file names).

Class Start

public class Start {  
 public static void main(String[] args) {  
 Login l = new Login();  
 }  
}

Class Utility

import javax.sound.sampled.\*;  
import java.io.File;  
import java.io.IOException;  
  
public class Utility {  
 public int count = 0;  
 long Position = -1;  
 public boolean Started = false;  
 float volume=0;  
  
 public Utility() throws LineUnavailableException {  
 }  
  
 Clip clip = AudioSystem.*getClip*();  
 public void VolumeUp(){  
  
 if(volume <1) {  
 setVolume(volume+5f);  
 }  
 FloatControl gainControl = (FloatControl) clip.getControl(FloatControl.Type.*MASTER\_GAIN*);  
 gainControl.setValue(volume);  
 }  
  
 public void VolumeDown(){  
 FloatControl gainControl = (FloatControl) clip.getControl(FloatControl.Type.*MASTER\_GAIN*);  
 setVolume(volume-5f);  
 gainControl.setValue(volume);  
 }  
  
 public void Play(String s) throws IOException, LineUnavailableException, UnsupportedAudioFileException {  
 Started = true;  
 String filePath = s;  
 File musicPath = new File(String.*valueOf*(filePath));  
 AudioInputStream audioInput = AudioSystem.*getAudioInputStream*(musicPath);  
  
 if (count < 1) {  
 clip.open(audioInput);  
 }  
  
 clip.start();  
 if (count >= 1) {  
 clip.setMicrosecondPosition(Position);  
 clip.start();  
 }  
 count++;  
 }  
  
 public void Pause() throws LineUnavailableException, IOException, UnsupportedAudioFileException {  
 Position = clip.getMicrosecondPosition();  
 clip.stop();  
 Started = true;  
 }  
  
 public void setVolume(float volume) {  
 this.volume = volume;  
 }  
  
 public int getVolume() {  
 int result = 2\*(int) (volume+45);  
 return result;  
 }  
}

Class DbService

import javax.swing.\*;  
import java.sql.\*;  
import java.util.ArrayList;  
  
public class DbService {  
 static String *CurrentCustomerName* = "";  
  
 String url = "jdbc:mysql://localhost:3306/sekotify?useUnicode=true&useLegacyDatetimeCode=false&serverTimezone=Turkey";  
 String user = "root";  
 String password = "123456";  
 ArrayList<String> TypeList = new ArrayList();  
  
 public Connection connect() {  
 Connection connection;  
 try {  
 connection = DriverManager.*getConnection*(url, user, password);  
 } catch (SQLException e) {  
 connection = null;  
 }  
 return connection;  
 }  
  
 public void Register(JTextField textField1, JTextField textField2, JTextField textField3, JTextField textField4) {  
 JFrame f = new JFrame();  
 try {  
 Connection connection = connect();  
 String query = "insert into sekotify.customer (username,password,name,surname) values (?,?,?,?)";  
 PreparedStatement statement = connection.prepareStatement(query);  
  
 if (!textField1.getText().equals("") && !textField2.getText().equals("") && !textField3.getText().equals("") && !textField4.getText().equals("")) {  
 statement.setString(1, textField1.getText());  
 statement.setString(2, textField2.getText());  
 statement.setString(3, textField3.getText());  
 statement.setString(4, textField4.getText());  
 statement.execute();  
 JOptionPane.*showMessageDialog*(f, "Registered Succesfully");  
 } else {  
 JOptionPane.*showMessageDialog*(f, "Fill all fields !");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public boolean Login(JTextField textField1, JTextField textField2) {  
 String username = textField1.getText();  
 String password = textField2.getText();  
 Connection connection = connect();  
 try {  
 String query = "select userName,password from sekotify.customer where userName='" + username + "' and password='" + password + "' ";  
 Statement st = connection.createStatement();  
 ResultSet rs = st.executeQuery(query);  
  
 if (rs.next()) {  
 setCurrentCustomerName(username);  
 return true;  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
 public void getListForTypeCombo() throws SQLException {  
 Connection connection = connect();  
 Statement st = connection.createStatement();  
 String type = "";  
  
 String query = "select distinct songType from sekotify.song ;";  
 ResultSet rs = st.executeQuery(query);  
 while (rs.next()) {  
 type = rs.getString(1);  
 TypeList.add( type);  
 }  
 }  
  
 public void Display(DefaultListModel modelList) throws SQLException {  
 Connection connection = connect();  
 PreparedStatement find = connection.prepareStatement("select songName from song"); // songType artist de ekle  
 ResultSet rs = find.executeQuery();  
  
 while (rs.next()) {  
 String name = rs.getString("songName");  
 name.split(".");  
  
 modelList.addElement("" + name);  
 }  
 }  
  
 public void DeletefromDB(String value) throws SQLException {  
 Connection connection = connect();  
 Statement st2 = connection.createStatement();  
 String query = "DELETE FROM sekotify.song WHERE songName = '" + value + "';";  
 st2.executeUpdate(query);  
 }  
  
 public void AddSongDB(String textField1Text, String textField2Text, String textField3Text) throws SQLException {  
 Connection connection = connect();  
 Statement st = connection.createStatement();  
 String query = "INSERT INTO `sekotify`.`song` (`songName`, `songType`, `artist`) VALUES ("+ "'" + textField3Text + ".wav'" + "," + "'" + textField1Text + "'" + "," + "'" + textField2Text + "'" + ");";  
 st.execute(query);  
 }  
  
 public void addtoPlayList(String sarkiadi) throws SQLException {  
 String CurrUserName = getCurrentCustomerName();  
 Connection connection = connect();  
 Statement st = connection.createStatement();  
 String query = "INSERT INTO `sekotify`.`playlist` (`username`, `songName`) VALUES (" + "'" + CurrUserName + "'" + "," + "'" + sarkiadi + "'" + ");";  
 st.execute(query);  
 }  
  
 public void RemovefromPlayList(String sarkiadi) throws SQLException {  
 String CurrUserName = getCurrentCustomerName();  
 Connection connection = connect();  
 Statement st2 = connection.createStatement();  
 String query = "DELETE FROM sekotify.playlist WHERE songName = '" + sarkiadi + "' " + "and username = " + "'" + CurrUserName + "';";  
 st2.executeUpdate(query);  
 }  
  
 public void DisPlaylist(DefaultListModel modelList) throws SQLException {  
 String CurrUserName = getCurrentCustomerName();  
 System.*out*.println(CurrUserName);  
 Connection connection = connect();  
 PreparedStatement find = connection.prepareStatement("select distinct songName from playlist where username = " + "'" + CurrUserName + "';");  
 ResultSet rs = find.executeQuery();  
  
 while (rs.next()) {  
 String name = rs.getString("songName");  
 name.split(".");  
  
 modelList.addElement("" + name);  
 }  
 }  
 public String getCurrentCustomerName() {  
 return *CurrentCustomerName*;  
 }  
 public void setCurrentCustomerName(String currentCustomerName) {  
 *CurrentCustomerName* = currentCustomerName;  
 }  
}

Class AddSong

import javax.swing.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.sql.SQLException;  
  
public class AddSong {  
 private JTextField textField1;  
 private JTextField textField2;  
 private JTextField textField3;  
 private JPanel AddSongPanel;  
 private JLabel add;  
 private JLabel back;  
  
  
 public AddSong(JFrame frame) {  
 DbService db = new DbService();  
 add.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 try {  
 db.AddSongDB(getTextField1Text(), getTextField2Text(), getTextField3Text());  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
 back.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 frame.setContentPane(new Menu( frame).getMenuPanel());  
 frame.setVisible(true);  
 frame.pack();  
 }  
 });  
 }  
  
 public String getTextField1Text() {  
 return textField1.getText();  
 }  
  
 public String getTextField2Text() {  
 return textField2.getText();  
 }  
  
 public String getTextField3Text() {  
 return textField3.getText();  
 }  
  
 public JPanel getAddSongPanel() {  
 return AddSongPanel;  
 }  
}

Class Display

import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.sql.SQLException;  
  
public class Display {  
 private JPanel DisplayPanel;  
 private JList list;  
 private JLabel del;  
 private JLabel back;  
  
 DbService db = new DbService();  
 String value;  
  
 public Display(JFrame frame) throws SQLException {  
 loadList();  
 list.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 if (e.getClickCount() == 1) {  
 value = (String) list.getSelectedValue();  
 }  
 }  
 });  
  
 del.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 try {  
 db.DeletefromDB(value);  
 loadList();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
  
 }  
 });  
 back.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 frame.setContentPane(new Menu( frame).getMenuPanel());  
 frame.setVisible(true);  
 frame.pack();  
 }  
 });  
 }  
  
 private void loadList() throws SQLException {  
 DefaultListModel modelList = new DefaultListModel();  
 list.setModel(modelList);  
 db.Display(modelList);  
 }  
  
 public JPanel getDisplayPanel() {  
 return DisplayPanel;  
 }  
}

Class Login

import javax.swing.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
  
public class Login extends javax.swing.JFrame {  
 DbService db = new DbService();  
 boolean succesful;  
 private JTextField textField1;  
 private JPanel loginPanel;  
 private JTextField textField2;  
 private JLabel register;  
 private JLabel login;  
 private JLabel exit;  
  
 public Login() {  
 JFrame frame = new JFrame();  
 frame.setContentPane(getLoginPanel());  
 frame.setTitle(" L O G I N ✮" );  
  
 frame.setVisible(true);  
 frame.pack();  
  
 register.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 frame.dispose();  
 Register r = new Register();  
 }  
 });  
  
 login.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 succesful = db.Login(textField1, textField2);  
 if (succesful) {  
 frame.setTitle( " S E C O T I F Y "+"" );  
 frame.setContentPane(new Menu(frame).getMenuPanel());  
 frame.setVisible(true);  
 frame.pack();  
 } else {  
 JOptionPane.*showMessageDialog*(loginPanel, "Username and Password did not match,try again or register ! ");  
 }  
 }  
 });  
  
 exit.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 System.*exit*(0);  
 }  
 });  
 }  
  
 public JPanel getLoginPanel() {  
 return loginPanel;  
 }  
}

Class Menu

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.sql.SQLException;  
  
public class Menu extends Container {  
  
 private JPanel menuPanel;  
 private JLabel select;  
 private JLabel showAll;  
 private JLabel addSong;  
 private JLabel displayPlaylist;  
  
 public Menu(JFrame frame) {  
  
 select.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 try {  
 frame.setContentPane(new SelectMusic(frame).getSelectMusicPanel());  
 frame.setTitle(" SELECT MUSİC ");  
 frame.setVisible(true);  
 frame.pack();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
 showAll.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 try {  
 frame.setContentPane(new Display(frame).getDisplayPanel());  
 frame.setTitle(" Song List "+ " ");  
 frame.pack();  
 frame.setVisible(true);  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
 addSong.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 frame.setContentPane(new AddSong(frame).getAddSongPanel());  
 frame.setTitle(" Add Song ✔");  
 frame.pack();  
 frame.setVisible(true);  
 }  
 });  
 displayPlaylist.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 try {  
 frame.setContentPane(new Playlist(frame).getDisPanel());  
 frame.setTitle(" My Playlist ♪");  
 frame.setVisible(true);  
 frame.pack();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
 }  
 public JPanel getMenuPanel() {  
 return menuPanel;  
 }  
  
  
}

Class Playlist

import javax.swing.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.sql.SQLException;  
  
public class Playlist {  
 private JList list1;  
 private JPanel DisPanel;  
 private JLabel back;  
 private JButton backButton;  
 DbService db = new DbService();  
  
 public Playlist(JFrame frame) throws SQLException {  
 loadList();  
  
 back.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 frame.setContentPane(new Menu(frame).getMenuPanel());  
 frame.setVisible(true);  
 frame.pack();  
 }  
 });  
 }  
  
 private void loadList() throws SQLException {  
 DefaultListModel modelList = new DefaultListModel();  
 list1.setModel(modelList); //list 1 is a model list anymore  
 db.DisPlaylist(modelList);  
 }  
  
 public JPanel getDisPanel() {  
 return DisPanel;  
 }  
}

Class PlaySong

import javax.sound.sampled.LineUnavailableException;  
import javax.sound.sampled.UnsupportedAudioFileException;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.io.IOException;  
import java.sql.SQLException;  
  
public class PlaySong extends Container {  
  
 private JPanel playPanel;  
  
 private JLabel addToPlaylistLabel;  
 private JLabel music;  
 private JLabel play\_pause;  
 private JLabel lowerSound;  
 private JLabel IncreaseSound;  
 private JLabel removeFromPlaylist;  
 private JLabel back;  
 private JLabel exit;  
 private JLabel vol;  
  
 DbService db = new DbService();  
 boolean Play;  
  
 public PlaySong(String para, JFrame otherFrame) throws LineUnavailableException {  
 Utility u = new Utility();  
 play\_pause.setEnabled(false);  
  
 otherFrame.setContentPane(getPlayPanel());  
 otherFrame.setVisible(true);  
 otherFrame.setTitle(" P L A Y E R " + "");  
 otherFrame.pack();  
  
 music.setText(para);  
 Play = false;  
  
 play\_pause.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
  
 if (Play) {  
 try {  
 u.Pause();  
 } catch (LineUnavailableException ex) {  
 ex.printStackTrace();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 } catch (UnsupportedAudioFileException ex) {  
 ex.printStackTrace();  
 }  
 play\_pause.setEnabled(false);  
 Play = false;  
  
 } else {  
  
 try {  
 u.Play(para);  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 } catch (LineUnavailableException ex) {  
 ex.printStackTrace();  
 } catch (UnsupportedAudioFileException ex) {  
 ex.printStackTrace();  
 }  
 play\_pause.setEnabled(true);  
 Play = true;  
 }  
 }  
 });  
 lowerSound.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 lowerSound.requestFocus();  
 u.VolumeDown();  
 vol.setText("%"+u.getVolume());  
 lowerSound.setVisible(true);  
 }  
 });  
 IncreaseSound.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 IncreaseSound.setEnabled(false);  
 u.VolumeUp();  
 vol.setText("%"+u.getVolume());  
 IncreaseSound.setEnabled(true);  
 }  
 });  
 addToPlaylistLabel.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
  
 try {  
 db.addtoPlayList(para);  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
 removeFromPlaylist.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 try {  
 db.RemovefromPlayList(para);  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
 back.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 otherFrame.setContentPane(new Menu(otherFrame).getMenuPanel());  
 otherFrame.setVisible(true);  
 otherFrame.pack();  
 }  
 });  
 exit.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 System.*exit*(0);  
 }  
 });  
 }  
 public JPanel getPlayPanel() {  
 return playPanel;  
 }  
}

Class Register

import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
  
public class Register extends JFrame {  
 private JPanel RegisterPanel;  
 private JTextField textField1;  
 private JTextField textField2;  
 private JTextField textField3;  
 private JTextField textField4;  
 private JLabel home;  
 private JLabel register;  
 private JLabel cancel;  
  
 DbService db = new DbService();  
  
 public Register() {  
 JFrame frame = new JFrame();  
 frame.setContentPane(getRegisterPanel());  
 frame.setVisible(true);  
 frame.setTitle(" R E G I S T E R ");  
 frame.pack();  
  
 home.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 frame.dispose();  
  
 Login l =new Login();  
 }  
 });  
 register.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 db.Register(textField1, textField2, textField3, textField4);  
 }  
 });  
 cancel.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 frame.dispose();  
 }  
 });  
 }  
 public JPanel getRegisterPanel() {  
 return RegisterPanel;  
 }  
}

Class SelectMusic

import javax.sound.sampled.\*;  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.ArrayList;  
  
public class SelectMusic {  
  
 private JPanel SelectMusicPanel;  
 private JComboBox MusicType;  
 private JComboBox chooseArtist;  
 private JComboBox Song;  
 ArrayList<String> ArtistList = new ArrayList();  
 ArrayList<String> SongList = new ArrayList();  
 DbService db = new DbService();  
  
 public SelectMusic(JFrame otherFrame) throws SQLException {  
  
 db.getListForTypeCombo(); //TypeList filled  
 MusicType.setModel(new DefaultComboBoxModel<>(db.TypeList.toArray(new String[db.TypeList.size()]))); // MusicType combo filled  
  
 MusicType.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 try {  
 MusicTypebyCombo();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 chooseArtist.setModel(new DefaultComboBoxModel<>(ArtistList.toArray(new String[ArtistList.size()]))); //Artist combo filled  
 }  
 });  
 chooseArtist.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 try {  
 ArtistbyCombo();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 Song.setModel(new DefaultComboBoxModel<>(SongList.toArray(new String[SongList.size()]))); // Song combo filled  
 }  
 });  
 Song.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 String songNameFromCombo = (String) Song.getSelectedItem();  
  
 try {  
 new PlaySong(songNameFromCombo, otherFrame);  
 } catch (LineUnavailableException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
  
 public void MusicTypebyCombo() throws SQLException {  
 String namecombo = (String) MusicType.getSelectedItem();  
 Connection connection = db.connect();  
 Statement st = connection.createStatement();  
  
 String Query = "select distinct artist from sekotify.song where songType = '" + namecombo + "';";  
 ResultSet rs = st.executeQuery(Query);  
  
 String type = "";  
 while (rs.next()) {  
 type = rs.getString(1);  
 ArtistList.add(type);  
 }  
 }  
  
 public void ArtistbyCombo() throws SQLException { // Provides songName  
 String namecombo = (String) chooseArtist.getSelectedItem();  
 String Query = "select songName from sekotify.song where artist = " + "'" + namecombo + "'" + ";";  
 Connection connection = db.connect();  
 Statement st = connection.createStatement();  
 ResultSet rs = st.executeQuery(Query);  
 String songName = "";  
 while (rs.next()) {  
 songName = rs.getString(1);  
 SongList.add(songName);  
 }  
 }  
  
 public JPanel getSelectMusicPanel() {  
 return SelectMusicPanel;  
 }  
}

Best Regards ...