THE MUSIC VAULT

Team 7
Steven Cao, Aye Swe, Geordi Reiner
SJSU
CS157A - Wu
Fall 17

TABLE OF CONTENTS

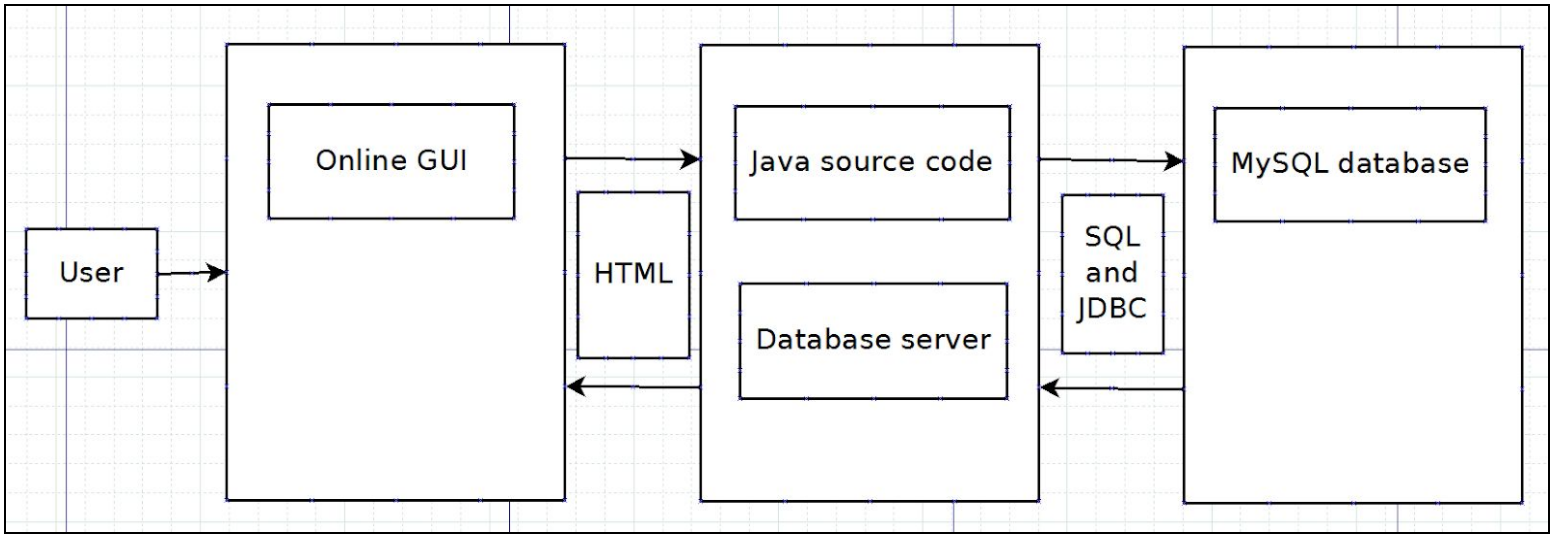
REQUIREMENTS

Project Description

The Music Vault is a web music database and rating application designed for entertainment and archival purposes. The goal is to build a database that allows for users to publish and edit music related data. This project is motivated by a love of music and a need for cleaner and more up-to-date entertainment databases. Stakeholders may include anyone with large physical music collections, people looking to discover new music, or anybody interested in rating and organization of personal tastes. The Music Vault's project domain resides purely in entertainment and archival interests. At the bare minimum, users will be able to archive and rate albums and search the database to find releases or artists. Additional functionality such as internet connected shopping options, list making, Spotify / Itunes integration, and global charting may be added as needed. The Music Vault is developed by three San Jose State University computer science and software engineering students: Steven Cao, Aye Swe, and Geordi Reiner.

System Environment

3-Tier Web Application Architecture:



Software Development Environment:

Windows 10

Relational Database Management System:

MySQL version 5.7.19

Application Languages:

Back-end: Java with JDBC

Database & Query: SQL

Front-end: HTML

Functional Requirements

1.1 Application must allow users to login and logout

Description: User enters username and password, both are validated. Successful login brings users to profile, failure presents the user with the opportunity to log-in again.

1.2 Application must allow users to register an account

Description: User clicks register option on “Login” prompt and enters a valid username, password, and email. User clicks register and will be prompted if registration was successful or not.

1.3 Application must display a profile page

Description: Profile displays a list of most recently rated albums and a list of favorite artists. Clicking on an artist or album name will take the user to the respective artist or album page. Profile also has a search bar for artist and album query.

1.4 Application must allow user to query by artist or release name.

Description: Users can enter queries in the search bar on their profiles. Users can search by artist name or by release name. Upon entering a search query, the user is presented a list of results sorted by relevance. Users can click on the presented artist or release name to be brought to each respective page.

1.5 Application must allow users to publish and edit data

Description:

For artist pages, users can do the following:

1. Add/edit formation location of artist
2. Add/edit current location of artist
3. Add/edit current members of band (with instruments played) if applicable
4. Add/edit also-known-as information of artist
5. Add/edit biography of artist

For album pages, users can do the following:

1. Add/edit release date
2. Add/edit release type
3. Add/edit genre information
4. Add/edit language information
5. Add/edit songs, which includes:
    a. Track number
    b. Song title
    c. Song duration (total length of album will be calculated and displayed)
    d. Disc number
6. Add/edit recording personnel and release credits
7. Add/edit recording label

Upon successfully publishing a new album, it will appear in the list of the artist's releases on its profile.

1.6 Application must allow users to delete data

Description: Users can also delete any aspect of artist or album data.

1.7 Application must allow users to rate albums

Description: Release pages will display an average rating and a total number of ratings. Users will be able to click on a star allocator to rate an album. The ratings will follow a 1 to 5 star system. The average rating will follow a usual mean calculation. Rated releases will then appear on the user's profile.

Non-Functional Requirements

2.1 Graphical User Interface (GUI)

Description:

1. Application uses customized HTML as GUI.
2. Application has a welcome screen presented to users before accessing login and registration features.

2.2 Security Features

Description:

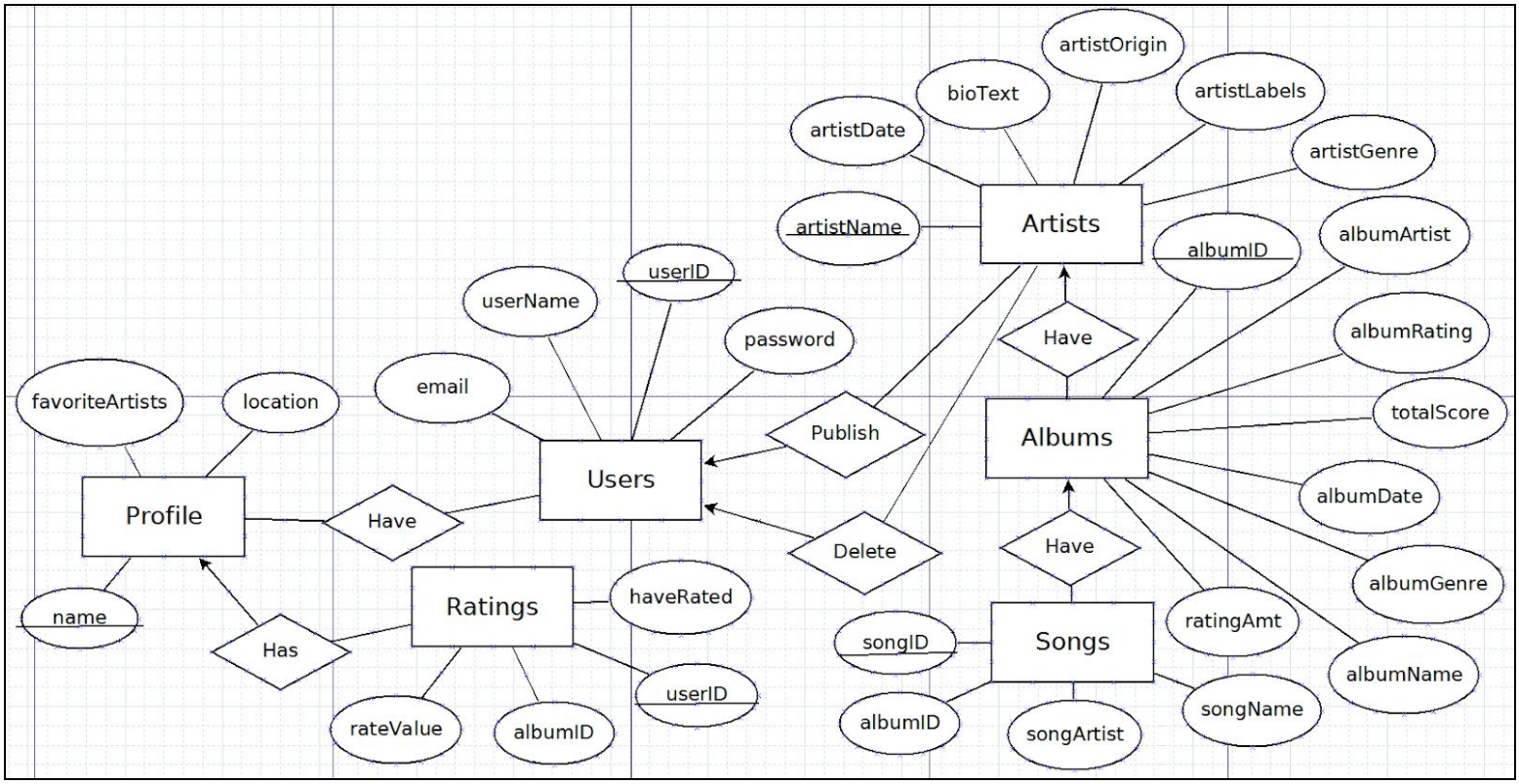
1. MySQL supports Transport Layer Security (TLS) which are encrypted connections between clients and server to ensure data over public network can be trusted, The Music Vault utilizes this.
2. MySQL is imbedded with MySQL Access Control System, an access control and privilege system that allows to create comprehensive access rules for handling client operations and effectively preventing unauthorized clients from accessing the database system. This allows for client

connection verification to check for valid users and request verification to check validation of the client's requests, The Music Vault utilizes this.

3. Hyper Text Transfer Protocol Secure (HTTPS) for protection of privacy and integrity of exchanged data to and from the web server is utilized.

DESIGN

Entity-Relationship Diagram



Non-Trivial Functional Dependencies

Ratings:

albumID ---> **rateValue**

Description: userID determines rateValue, not albumID. albumID is included to determine what Album the User has rated.

Schemas

Profile(name, location, favoriteArtists)

Has(name, userID)

Ratings(userID, albumID, haveRated, rateValue)

Have(name, userID)

Users(userID, userName, password, email)

Publish(userID, artistName)

Delete(userID, artistName)

Artists(artistName,artistDate, bioText, artistOrigin, artistLabels, artistGenre)

Have(artistName, albumID)

Albums(albumID, albumArtist, albumRating, totalScore, albumDate, albumGenre, albumName, ratingAmt)

Have(albumID, songID)

Songs(songID, albumID, songArtist, songName)

Schema Descriptions

Profile: Profile is an entity set because it elaborates on personal user preferences and is viewable by Users.

Has: Has is a many-to-one relationship connecting Ratings to Profile distinguishing that profiles have many ratings.

Ratings: Ratings is an entity set because the size of its tuple would be too large for an attribute and behaves differently than Albums (cannot publish ratings for example).

Have: (for Users connecting to Profile) is a one-to-one relationship displaying that Users are capable of having one viewable profile per account.

Users: Users is an entity set because we will need to distinguish them by unique userIDs, userNames, emails, and passwords if we are to maintain a fully deployed web application.

Publish: Publish is a many-to-one relationship between Users and Artists. This is many-to-one because each User can publish as much data as they want. The cascading relationship of Artists-Albums-Songs also allows User to publish Song and Album data.

Delete: The Delete relationship is applicable to the same data as Publish, but needs to be distinguished separately because it modifies the database with different behavior.

Artists, Albums, and Songs: These are entities because they embody the primary core of our data and the purpose of our archival music project.

Have: Lastly, both Have relationships between Artists and Albums, and Albums and Songs denote a many-to-one connection cascading down from Artists. Essentially this denotes: A singular Artist can have many Albums and a singular Album can have many Songs.

Example Tables

Artists(artistName,artistDate, bioText, artistOrigin, artistLabels, artistGenre)

artistName	artistDate	artistGenre	artistOrigin	artistLabels	bioText
Adele	2006-present	Soul, pop	London, England	Vocals	Adele Laurie Blue Adkins is an English singer and...
Blake Shelton	2001-present	Nashville Tennessee	Country	Giant	Blake Tollison Shelton (born June 18, 1976) is a...
Bon Jovi	1983-present	Hard rock	Savreville, New Jersey, U.S.	Island	Bon Jovi is an American rock band from Savrevill...
Bruno Mars	2004-present	R&B, funk	Honolulu, Hawaii, U.S.	Universal Motown	Peter Gene Hernandez (born October 8, 1985)....
Josh Groban	1997-present	Easy Listening	Los Angeles, California, U.S.	Reprise	Joshua Winslow Groban (born February 27, 198...
Justin Bieber	2007-present	Pop, R&B	London, Ontario, Canada	IslandTeen Island	Justin Drew Bieber is a Canadian Singer and son...
Katy Perry	2001-present	Pop,rock	Santa Barbara, California, U.S.	Red Hill, Java	Kathervn Elizabeth Hudson (born October 25, 1...
Kelly Clarkson	2002-present	Burleson Texas	soul	Atlantic	Kelly Brianne Clarkson (born April 24, 1982) is a...
Lady Gaga	2001-present	Pop, dance, electronic	Manhattan, New York, U.S.	Def Jam Cherry tree	Stefani Joanne Angelina Germanotta(born Marc...
Mariah Carey	1988-present	New York City	R&B, pop, Hip hop	Columbia, Virgin	Mariah Carey (born March 27, 1969 or 1970) is ...
Megan Trainor	2009-present	R&B, pop	Nantucket, Massachusetts,U.S.	Epic	Meghan Elizabeth Trainor(born December 22, 1...
Michael Buble	1996-present	Traditional pop	Burnaby, British Columbia	Reprise	Michael Steven Bublé is a Canadian singer, song...
Pink	1995-present	Philadelphia	Pop	LaFace	Alecia Beth Moore (born September 8, 1979), k...
Taylor Swift	2004-present	Pop, country	Reading, Pennsylvania	RCA Big machine	Taylor Alison Swift (born December 13, 1989) is...
Woodkid	2006-present	Pop, Chamber Pop, A...	Lyon, France	Green United Music	Yoann Lemoine (born 16 March 1983) is a Frenc...
Yanni	1980-present	Contemporary, instru...	Kalamata, Greece	Virgin,EMI	Yiannis Chryssomallis born November 14, 1954)...

Albums(albumID, albumArtist, albumRating, totalScore, albumDate, albumGenre, albumName, ratingAmt)

albumID	albumArtist	albumName	albumRating	totalScore	ratingAmt	albumDate	albumGenre
1	Woodkid	The Golden Age	0	0	0	18 March 2013	Orchestral Pop
2	Adele	21	0	0	0	24 January 2011	Pop
3	Yanni	In My Time	0	0	0	6 April 1993	Instrumental
4	Justin Bieber	Purpose	0	0	0	13 November 2015	Dance
5	Taylor Swift	Reputation	0	0	0	24 August 2017	Pop
6	Meghan Trainor	All about That Bass	0	0	0	30 Jun 2014	Bubblegum
7	Lady Gaga	The Fame	0	0	0	19 August 2008	Electropop
8	Bruno Mars	Unorthodox Jukebox	0	0	0	7 December 2012	Pop
9	Katy Perry	Teenage Dream	0	0	0	24 August 2010	Pop
10	Pink	Beautiful Trauma	0	0	0	13 October 2017	Pop
11	Blake Shelton	All About Tonight	0	0	0	10 August 2010	Country
12	Kelly Clarkson	Breakaway	0	0	0	30 November 2004	Pop rock
13	Josh Groban	To Where You Are	0	0	0	12 November 2002	Adult Contem...
14	Mariah Carey	Butterfly	0	0	0	16 September 1997	R&B
15	Bon Jovi	Slippery When Wet	0	0	0	18 August 1986	Hard rock
16	Michael Buble	Its Time	0	0	0	8 February 2005	Vocal Jazz

Profile(name, location, favoriteArtists

name	location	favoriteArtists
Ave Swe	Sunny Value, CA	Adele, Bon Jovi, Bruno Mars, Yanni
Alec	Berkeley, CA	Danny Brown, Kanye Wst
Anna	San Francisco	The Doors, Jimi Hendrix, Pink Floyd
Bryan	Los Angeles, CA	Metallica, Slayer,Megadeth
Gary	Portland, OR	Aphex Twin, Boards of Canada, Squarepusher
Geodi	Los Angels, CA	Justin Bieber
Gordon	London, UK	The Rolling Stones, The Who, The Betales
Henry	Hemosa Beach, CA	Black Flag, Minor Threat, Dead Kennedys
John	Houston, TX	Hank Williams, Waylon Jennings, Willie Nelson
Mare	Berlin, DE	Ludwig van Beethoven
Natalie	Santa Rosa, CA	Lady Gaga
Oystein	Oslo, NO	Mayhem, Burzum, Darkthrone
Richard	Lake County, CA	Taylor Swift
Sarah	Twin Peaks, WA	Slowdive, The Smiths, The Cure
Steven	Fremont, CA	Ludwig van Beethoven

Songs(<u>songID</u>, albumID, songArtist, songName)

songID	songArtist	songName	albumID
1	Woodkid	The Golden Age	1
2	Woodkid	The Great Escape	1
3	Woodkid	Boat Song	1
4	Woodkid	I Love You	1
5	Woodkid	The Shore	1
6	Woodkid	Ghost Lights	1
7	Woodkid	Shadows	1
8	Woodkid	Stabat Mater	1
9	Woodkid	Conquest of Spaces	1
10	Woodkid	Falling	1
11	Woodkid	Where I Live	1
12	Woodkid	Iron	1
13	Woodkid	The Other Side	1
14	Adele	Rolling In The Deep	2
15	Adele	Rumour Has It	2
16	Adele	Turning Tables	2
17	Adele	Don't You Remember	2
18	Yanni	In The Morning Light	3
19	Yanni	One Man's Dream	3
20	Yanni	Before I Go	3
21	Yanni	Enchantment	3
22	Justin Bie...	Mark My Words	4
23	Justin Bie...	I'll Show You	4
24	Justin Bie...	What Do you Mean	4
25	Justin Bie...	Sorry	4
26	Taylor Swift	End Game	5
27	Taylor Swift	I Did Something Bad	5
28	Taylor Swift	Delicate	5
29	Taylor Swift	Gorgeous	5
30	Meghan ...	All About That Bass	6
31	Meghan ...	3am	6
32	Meghan ...	The Best Part	6
33	Meghan ...	Close Your Eyes	6
34	Lady Gaga	Just Dance	7
35	Lady Gaga	LoveGame	7
36	Lady Gaga	The Fame	7
37	Lady Gaga	Poker Face	7
38	Bruno Mars	Young Girls	8
39	Bruno Mars	Locked Out of Hea...	8
40	Bruno Mars	Moonshine	8
41	Bruno Mars	Treasure	8
42	Katy Perry	Teenage Dream	9
43	Katy Perry	California Gurls	9

Users(<u>userID</u>, userName, password, email)

userID	username	password	email
215	Emma	bvghtuv	em...
235	Ave Swe	thueveo	appl...
243	Kai	edwo	kai...
324	Scarlett	bgfuv34	scar...
342	John	iohnAtei...	iohn...
548	Harper	oueoto22	Har...
587	Zoe	eqeto22	Zoe...
665	Brandon	bvnbku	bra...
784	Christopher	oevte	chri...
841	Violet	loiqw	viol...
856	Jasper	nmvnha	iasp...
857	Logan	iouego	loga...
876	Luna	ovvqvw...	luna...
879	Lucas	hqvqi	luca...
887	Natalie	lkoiq	Nat...
NULL	NULL	NULL	NULL

Publish(<u>userID</u>, <u>artistName</u>)

userID	artistName
215	Adele
235	Blake Shel...
243	Bon Jovi
324	Bruno Mars
342	Josh Groban
548	Justin Bieber
587	Katy Perry
665	Kelly Clark...
784	Lady Gaga
841	Mariah Ca...
856	Megan Tr...
857	Michael B...
876	Pink
879	Taylor Swift
887	Woodkid

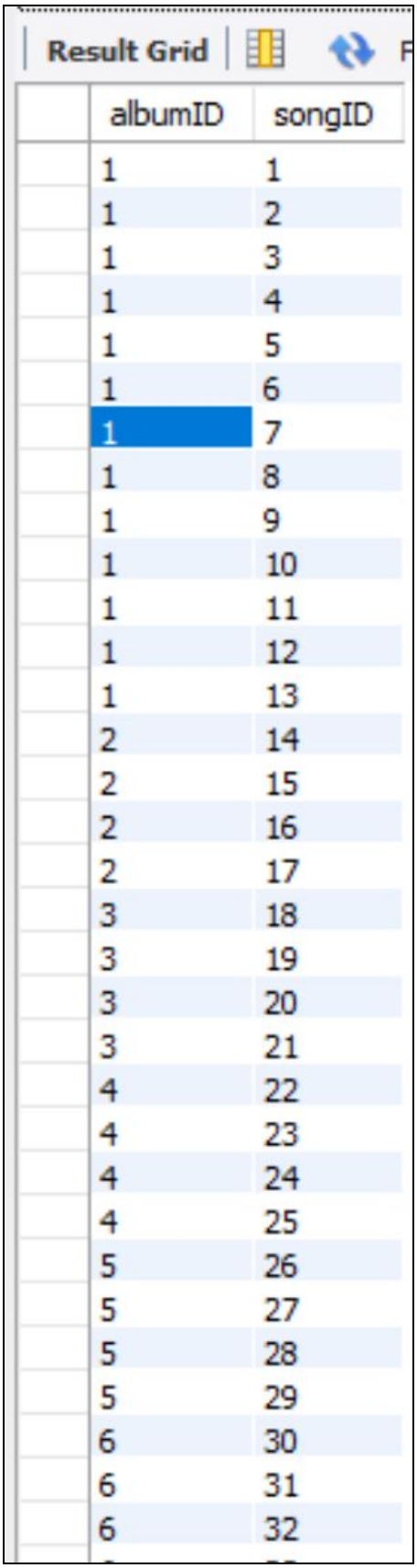
Have(<u>artistName</u>, <u>albumID</u>) (From Artists to Albums)

artistName	albumID
Adele	2
Blake Shelton	11
Bon Jovi	15
Bruno Mars	8
Josh Groban	13
Justin Bieber	4
Katy Perry	9
Kelly Clarkson	12
Lady Gaga	7
Mariah Carey	14
Megan Trainor	6
Michael Buble	16
Pink	10
Taylor Swift	5
Woodkid	1

Have(<u>name</u>, <u>userID</u>) (From Users to Profile)

name	userID
Zoe	587
Violet	841
Natalie	887
Luna	876
Lucas	879
Logan	857
Kai	243
John	342
Jasper	856
Harper	548
Emma	215
Christopher	784
Brandon	665
Ave Swe	235

Have(albumID, songID) (From Albums to Songs)



Result Grid

albumID	songID
1	1
1	2
1	3
1	4
1	5
1	6
1	7
1	8
1	9
1	10
1	11
1	12
1	13
2	14
2	15
2	16
2	17
3	18
3	19
3	20
3	21
4	22
4	23
4	24
4	25
5	26
5	27
5	28
5	29
6	30
6	31
6	32

Ratings(<u>userID</u>, albumID, haveRated, rateValue)

	userID	albumID	haveRated	rateValue
	215	7	1	5
	235	10	1	2
	243	1	0	NULL
	324	15	1	5
	342	3	0	NULL
	548	16	1	4
	587	2	0	NULL
	665	6	0	NULL
	784	8	1	5
	841	4	0	NULL
	856	12	1	3
	857	5	1	1
	876	13	0	NULL
	879	17	1	2
	887	14	0	NULL
	NULL	NULL	NULL	NULL

IMPLEMENTATION

Implementation Description

Our program's structure follows a MVC (Model, View, and Controller) pattern. Java files labeled as "DAO" serves as the connection to the database or the "Model" where it will retrieve the data and store them into objects for use. Files labeled as "Servlets" handles the flow of data by retrieving and sending data back and forth from the database and to the viewer. This serves as our "Controller". The JSP files are implemented as our "Viewer". It contains HTML code and uses CSS to display our web pages for the users. The images and data are stored locally on your machine.

Design Implementation

The requirement was to implement a 3-tier architecture which involves a viewer that the user interacts with, a web server where it would communicate between the viewer and the database, and a database to store and retrieve data from. We decided to use a web browser for our viewer, WildFly(JBoss) as our web server, and MySQL as our database.

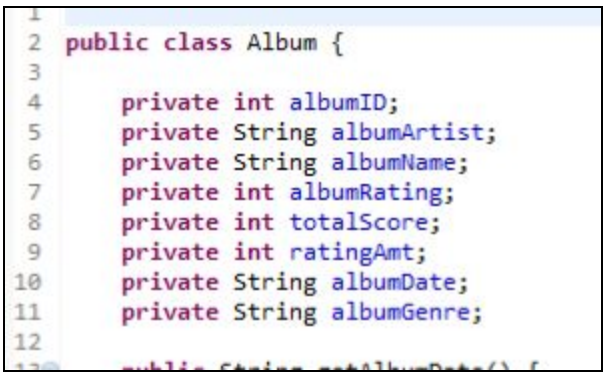
Function Descriptions

File Name	Functions	Description
albumPageDAO.java	public List<Songs> list(String query)	Retrieves and creates a list of songs that is associated to an album
albumPageDAO.java	public void deleteAlbum(int albumID)	Searches and deletes the inputted albumID.
albumPageServlet.java	protected void doGet(HttpServletRequest request, HttpServletResponse response)	Retrieves the song list and forwards the data to albumProfile.jsp
albumPageServlet.java	protected void doPost(HttpServletRequest request, HttpServletResponse response)	Retrieves input action and calls deleteAlbum function
artistPageDAO.java	Artist grabArtist(String artistName)	Retrieves an artist's information and store in an Artist object

artistPageServlet.java	protected void doGet(HttpServletRequest request, HttpServletResponse response)	Retrieves artist information, their albums information, and their following ratings. Forwards the information to artistProfile.jsp for display
RatingListDAO.java	public List<Album> list(String query,boolean mainPage)	Retrieves entire album database and their following ratings.
RatingListDAO.java	public void ratingUpdate(int rating, int totalScore, int ratingAmt, int albumIDKey, String action, String userID)	Updates the selected album's ratings with user's rating inputs.
RatingListDAO.java	public boolean CheckHaveRatedFalse(String userID, int albumID,int rateValue)	Verifies whether if user has not rated the selected album
RatingListDAO.java	public boolean CheckHaveRatedTrue(String userID, int albumID,int rateValue)	Verifies whether if user has already rated the selected album
RatingListDAO.java	public Album grabAlbum(String albumID)	Retrieves an album's information.
RatingListDAO.java	public void search(String albumName)	Used for search bar. Searches the database for matching substrings.
RatingServlet.java	protected void doGet(HttpServletRequest request, HttpServletResponse response)	Retrieves entire album database and their following ratings and forwards it to RatingList.jsp
RatingServlet.java	protected void doPost(HttpServletRequest request, HttpServletResponse response)	Retrieves user's rating input and updates the rating database

Entities that are used to store MySQL Database information with their following attributes:

Albums(albumID, albumArtist, albumRating, totalScore, albumDate, albumGenre, albumName, ratingAmt)



```
1
2  public class Album {
3
4      private int albumID;
5      private String albumArtist;
6      private String albumName;
7      private int albumRating;
8      private int totalScore;
9      private int ratingAmt;
10     private String albumDate;
11     private String albumGenre;
12
```

Artists(artistName,artistDate, bioText, artistOrigin, artistLabels, artistGenre)

```
public class Artist {


    private String artistName;
    private String artistDate;
    private String artistGenre;
    private String artistOrigin;
    private String artistLabels;
    private String bioText;
```

Songs(songID, albumID, songArtist, songName)

```
public class Songs {
    private int songID;
    private int albumID;
    private String songName;
    private String songArtist;
```

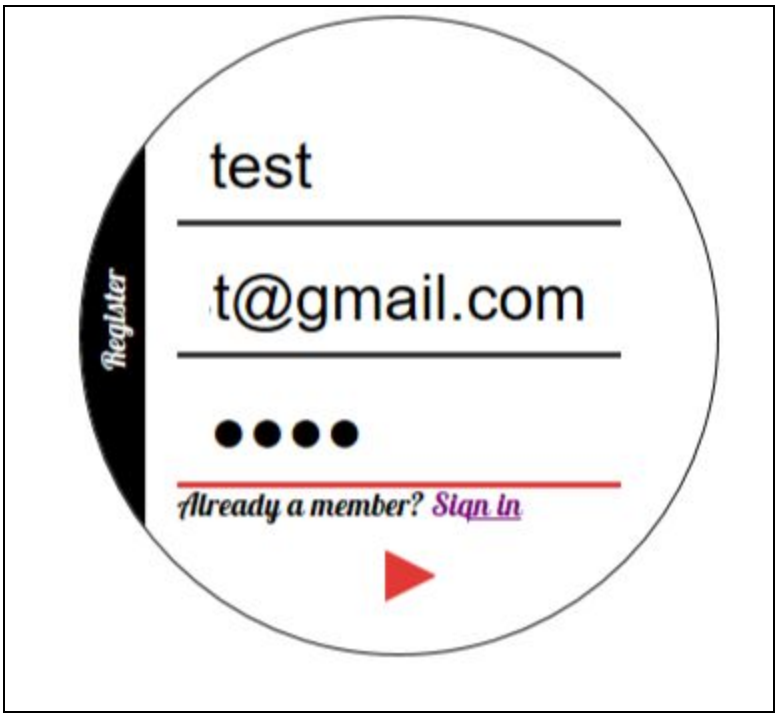
haveRated(userID, albumID, haveRated)

```
public class haveRated {

    private int userID;
    private int albumID;
    private String haveRated;
```

Insert, Delete, Update, and Query Examples

User Registration Insertion

GUI:



Code:

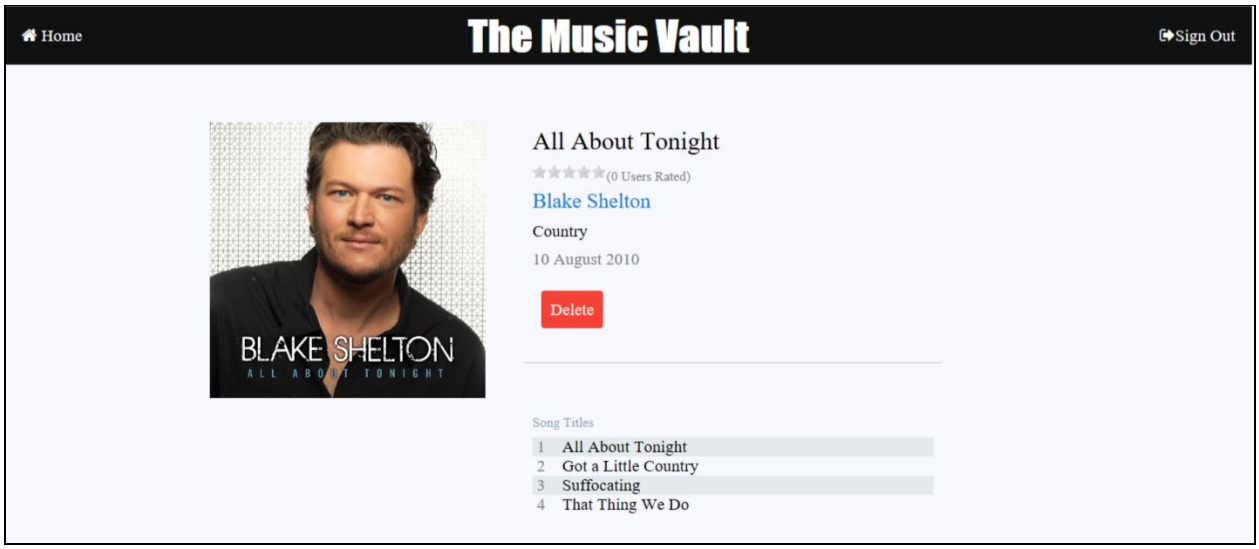
```
<%@ page import= javax.sql.* %>
<%
String username = request.getParameter("username");
session.putValue("username",username);
String password = request.getParameter("password");
String email = request.getParameter("email");
Class.forName("com.mysql.jdbc.Driver");
Connection connect = DriverManager.getConnection("jdbc:mysql://localhost:3306/TheMusicVault?autoReconnect=true&useSSL=false","root","root");
Statement stmt = connect.createStatement();

ResultSet rs;

int i = stmt.executeUpdate("insert into users values (NULL,'"+username+"','"+password+"','"+email+"')");

response.sendRedirect("login2.html");
%>
```

Result:

userID	username	password	email
1	test	test	test@gmail.com
2	Steven	53177	stevencao@gmail.com
3	Alex	password	alex@gmail.com
4	l33t	superman	cringe@gmail.com
NULL	NULL	NULL	NULL

Album Deletion

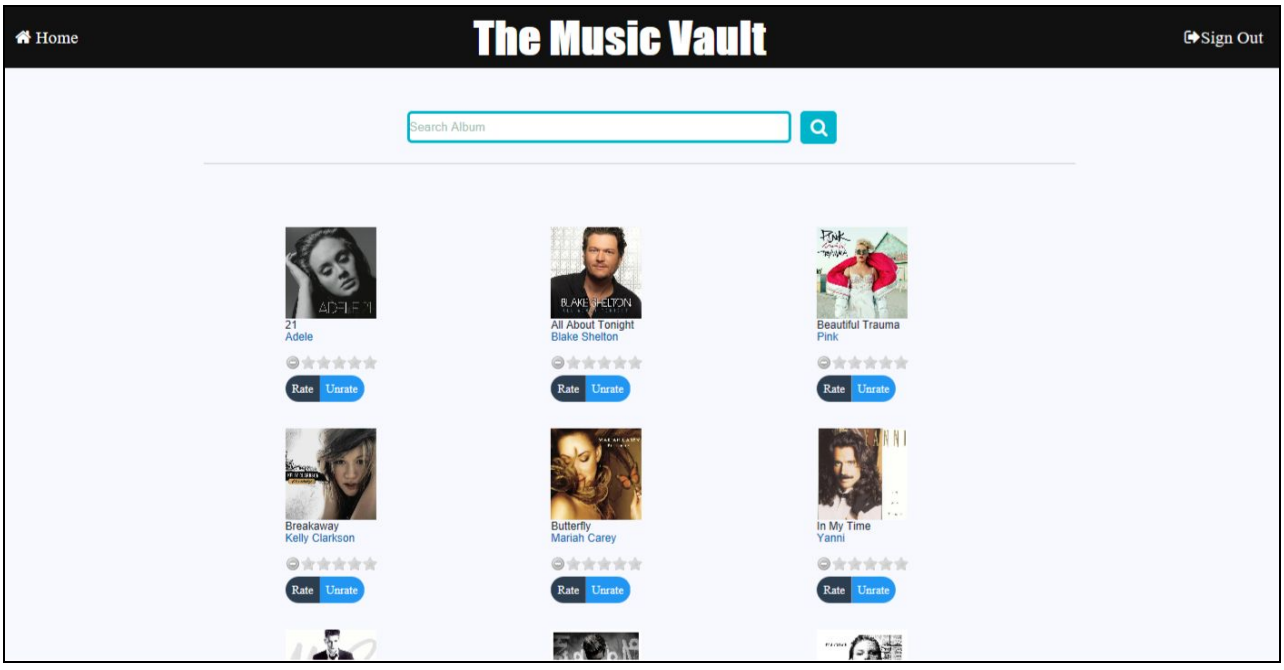
GUI:



Code:

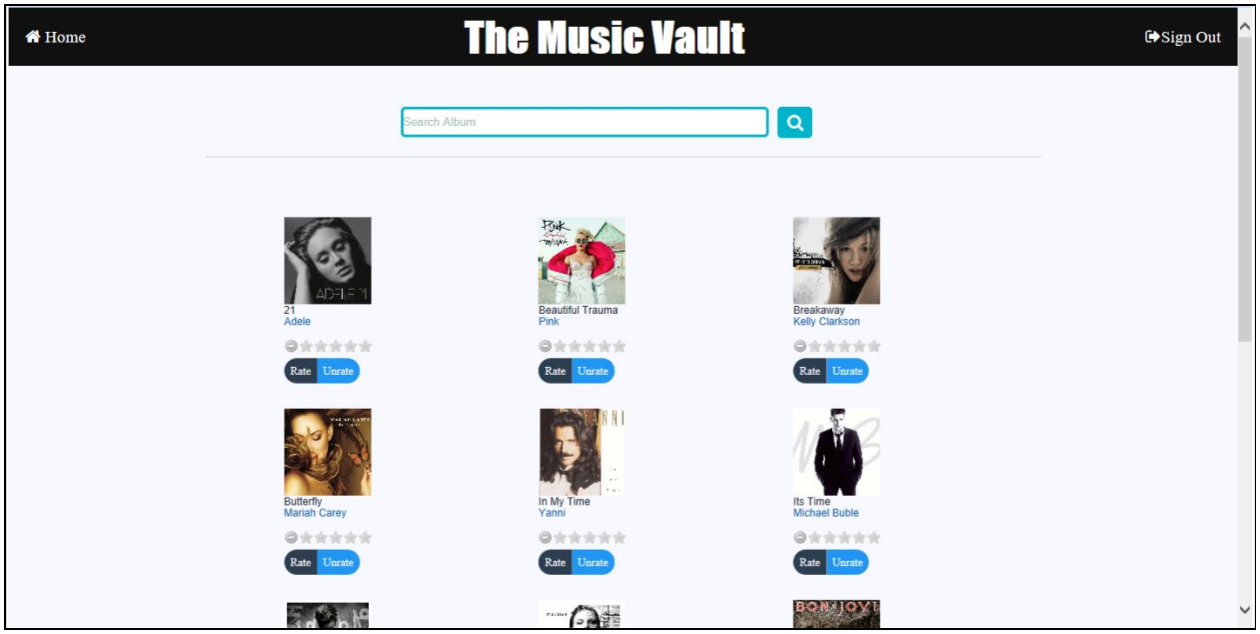
```
    public void deleteAlbum(int albumID)
    {
        try {
            Class.forName("com.mysql.jdbc.Driver");
               Connection connect = DriverManager.getConnection("jdbc:mysql://localhost:3306/TheMusicVault?autoReconnect=true&useSSL=false","root","root");
               Statement stmt = connect.createStatement();
               int i = stmt.executeUpdate("DELETE FROM albums WHERE albumID='"+albumID+"'");

        }catch (Exception e)
        {
            System.out.println("Error at deleteAlbum");
        }
    }

}
```

Before Delete:



albumID	albumArtist	albumName	albumRating	totalScore	ratingAmt	albumDate	albumGenre
1	Woodkid	The Golden Age	0	0	0	18 March 2013	Orchestral Pop
2	Adele	21	0	0	0	24 January 2011	Pop
3	Yanni	In My Time	0	0	0	6 April 1993	Instrumental
4	Justin Bieber	Purpose	0	0	0	13 November 2015	Dance
5	Taylor Swift	Reputation	0	0	0	24 August 2017	Pop
7	Lady Gaga	The Fame	0	0	0	19 August 2008	Electropop
8	Bruno Mars	Unorthodox Jukebox	0	0	0	7 December 2012	Pop
9	Katy Perry	Teenage Dream	0	0	0	24 August 2010	Pop
10	Pink	Beautiful Trauma	0	0	0	13 October 2017	Pop
11	Blake Shelton	All About Tonight	0	0	0	10 August 2010	Country
12	Kelly Clarkson	Breakaway	0	0	0	30 November 2004	Pop rock
13	Josh Groban	To Where You Are	0	0	0	12 November 2002	Adult Contem...
14	Mariah Carey	Butterfly	0	0	0	16 September 1997	R&B
15	Bon Jovi	Slippery When Wet	0	0	0	18 August 1986	Hard rock
16	Michael Buble	Its Time	0	0	0	8 February 2005	Vocal Jazz
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

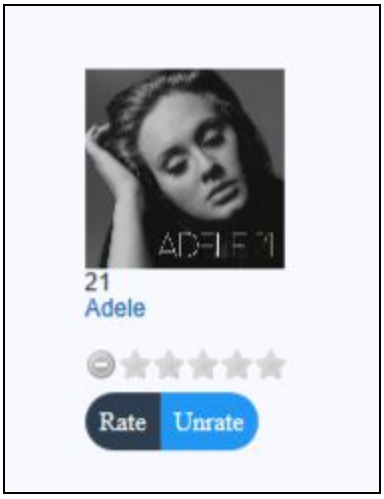
After Delete:



albumID	albumArtist	albumName	albumRating	totalScore	ratingAmt	albumDate	albumGenre
1	Woodkid	The Golden Age	0	0	0	18 March 2013	Orchestral Pop
2	Adele	21	0	0	0	24 January 2011	Pop
3	Yanni	In My Time	0	0	0	6 April 1993	Instrumental
4	Justin Bieber	Purpose	0	0	0	13 November 2015	Dance
5	Taylor Swift	Reputation	0	0	0	24 August 2017	Pop
7	Lady Gaga	The Fame	0	0	0	19 August 2008	Electropop
8	Bruno Mars	Unorthodox Jukebox	0	0	0	7 December 2012	Pop
9	Katy Perry	Teenage Dream	0	0	0	24 August 2010	Pop
10	Pink	Beautiful Trauma	0	0	0	13 October 2017	Pop
12	Kelly Clarkson	Breakaway	0	0	0	30 November 2004	Pop rock
13	Josh Groban	To Where You Are	0	0	0	12 November 2002	Adult Contem...
14	Mariah Carey	Butterfly	0	0	0	16 September 1997	R&B
15	Bon Jovi	Slippery When Wet	0	0	0	18 August 1986	Hard rock
16	Michael Buble	Its Time	0	0	0	8 February 2005	Vocal Jazz
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Rating Update

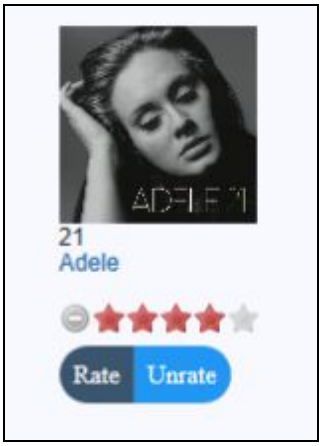
GUI:



Code:

```
lic void ratingUpdate(int rating, int totalScore, int ratingAmt, int albumIDKey, String action, String userID)

try
{
      Class.forName("com.mysql.jdbc.Driver");
      Connection connect = DriverManager.getConnection("jdbc:mysql://localhost:3306/TheMusicVault?autoReconnect=true&useSSL=false","root","root");
      Statement stmt = connect.createStatement();
      if(action.equals("submitRating"))
      {
          int newTotalScore = totalScore + rating;
          int newRatingAmt = ratingAmt + 1;
          int averageRating = (newTotalScore / newRatingAmt) + ((newTotalScore % newRatingAmt == 0) ? 0 : 1); //rounds up the average Rating
          int i = stmt.executeUpdate("Update albums SET albumRating = '"+(averageRating)+"',totalScore='"+(newTotalScore)+"',ratingAmt='"+(newRatingAmt)+"' where albumID = '"+albumIDKey+"' ");
      }
```

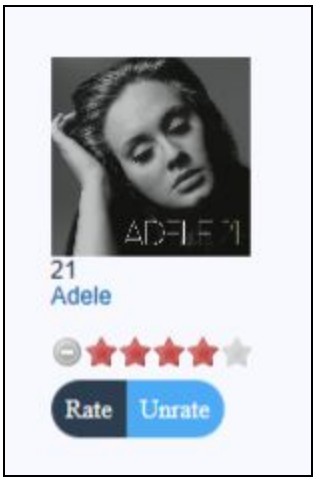
Result:



albumID	albumArtist	albumName	albumRating	totalScore	ratingAmt	albumDate	albumGenre
1	Woodkid	The Golden Age	0	0	0	18 March 2013	Orchestral Pop
2	Adele	21	4	4	1	24 January 2011	Pop
3	Yanni	In My Time	0	0	0	6 April 1993	Instrumental
4	Justin Bieber	Purpose	0	0	0	13 November 2015	Dance
5	Taylor Swift	Reputation	0	0	0	24 August 2017	Pop
7	Lady Gaga	The Fame	0	0	0	19 August 2008	Electropop
8	Bruno Mars	Unorthodox Jukebox	0	0	0	7 December 2012	Pop
9	Katy Perry	Teenage Dream	0	0	0	24 August 2010	Pop
10	Pink	Beautiful Trauma	0	0	0	13 October 2017	Pop
12	Kelly Clarkson	Breakaway	0	0	0	30 November 2004	Pop rock
13	Josh Groban	To Where You Are	0	0	0	12 November 2002	Adult Contem...
14	Mariah Carey	Butterfly	0	0	0	16 September 1997	R&B
15	Bon Jovi	Slippery When Wet	0	0	0	18 August 1986	Hard rock
16	Michael Buble	Its Time	0	0	0	8 February 2005	Vocal jazz
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Rating Update (Unrating)

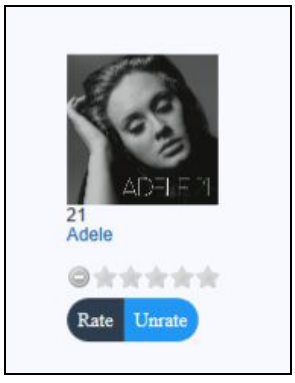
GUI:



Code:

```
if(action.equals("deleteRating"))
{
    PreparedStatement ps = connect.prepareStatement( "Select * FROM ratingCheck WHERE userID =? AND albumID =?");
    // Using prepared statements allows us to use inputs as parameters instead of allows user input directly inputing into the mysql command.
    ps.setString(1, userID);
    ps.setInt(2, albumIDKey);

    ResultSet rs = ps.executeQuery();


    if( rs.next())
    {
        rating  = rs.getInt("rateValue");
    }
    int newTotalScore = totalScore - rating;

    if(newTotalScore < 0)
    {
        newTotalScore = 0;
    }

    int newRatingAmt = ratingAmt - 1;

    if(newRatingAmt < 0)
        newRatingAmt = 0;

    int averageRating;
    if(newTotalScore == 0 && newRatingAmt == 0)
        averageRating = 0;
    else
         averageRating = (newTotalScore / newRatingAmt) + ((newTotalScore % newRatingAmt == 0) ? 0 : 1); //rounds up the average Rating
    int i = stmt.executeUpdate("Update albums SET albumRating = '"+(averageRating)+"',totalScore='"+(newTotalScore)+"',ratingAmt='"+(newRatingAmt)+"' where albumID = '"+albumIDKey+"' "

}
```

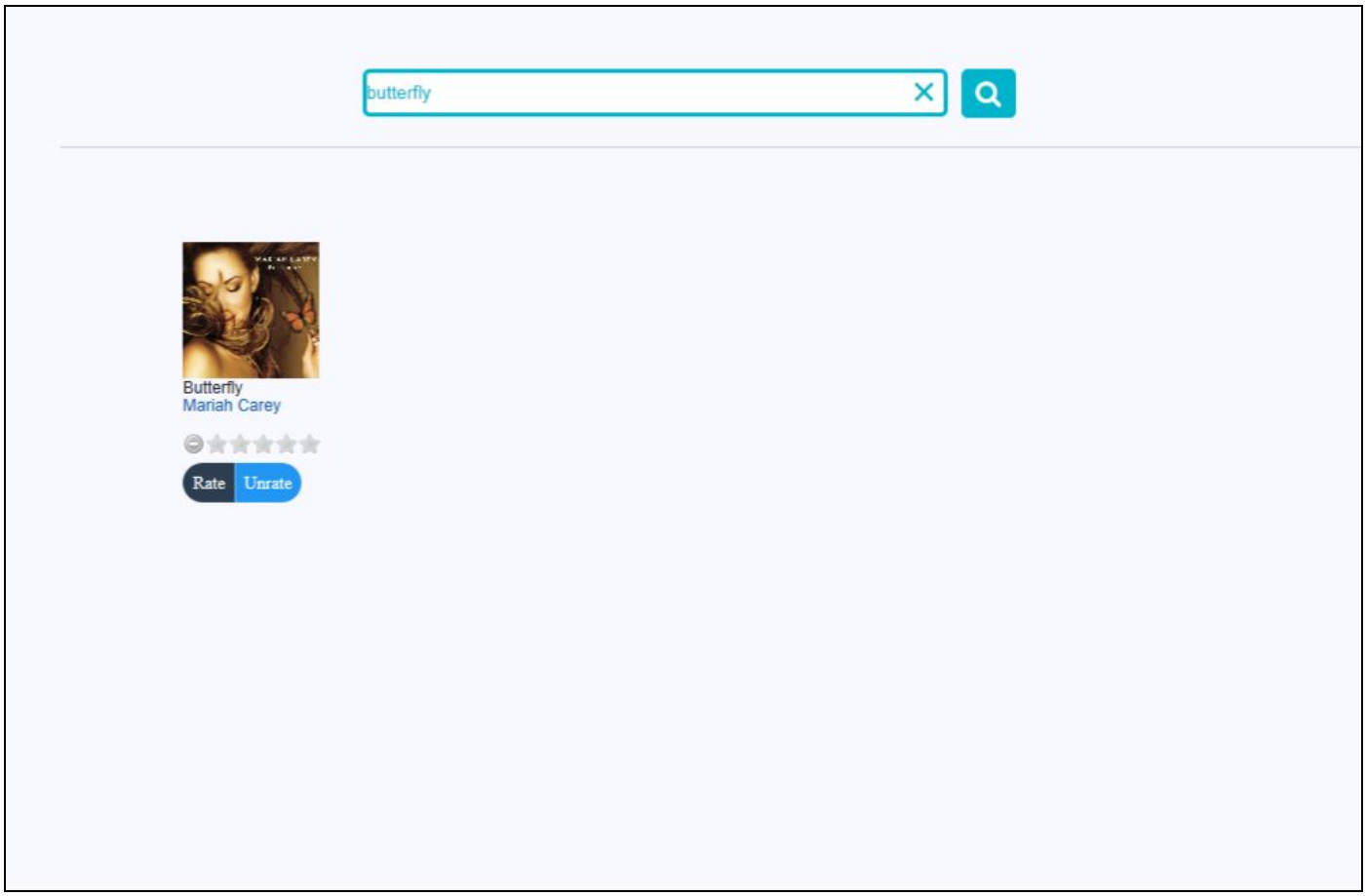
Result:



albumID	albumArtist	albumName	albumRating	totalScore	ratingAmt	albumDate	albumGenre
1	Woodkid	The Golden Age	0	0	0	18 March 2013	Orchestral Pop
2	Adele	21	0	0	0	24 January 2011	Pop
3	Yanni	In My Time	0	0	0	6 April 1993	Instrumental
4	Justin Bieber	Purpose	0	0	0	13 November 2015	Dance
5	Taylor Swift	Reputation	0	0	0	24 August 2017	Pop
7	Lady Gaga	The Fame	0	0	0	19 August 2008	Electropop
8	Bruno Mars	Unorthodox Jukebox	0	0	0	7 December 2012	Pop
9	Katy Perry	Teenage Dream	0	0	0	24 August 2010	Pop
10	Pink	Beautiful Trauma	0	0	0	13 October 2017	Pop
12	Kelly Clarkson	Breakaway	0	0	0	30 November 2004	Pop rock
13	Josh Groban	To Where You Are	0	0	0	12 November 2002	Adult Contem...
14	Mariah Carey	Butterfly	0	0	0	16 September 1997	R&B
15	Bon Jovi	Slippery When Wet	0	0	0	18 August 1986	Hard rock
16	Michael Buble	Its Time	0	0	0	8 February 2005	Vocal jazz

Search Query

GUI and result:



Code:

```
public void search(String albumName)
{
    try {

        Class.forName("com.mysql.jdbc.Driver");
        Connection connect = DriverManager.getConnection("jdbc:mysql://localhost:3306/TheMusicVault?autoReconnect=true&useSSL=false","root","root");
        PreparedStatement stmt = connect.prepareStatement("SELECT * FROM albums WHERE albumName like '%?%' ORDER BY albumName ASC;");

        stmt.setString(1,albumName); // first ?, username =?
        ResultSet rs= stmt.executeQuery(); // executes the query/statement stmt to SQL
    }catch(Exception e)
    {
        System.out.println("Error at RatingList search");
    }
}
```

Constraints

The Music Vault has only been developed and tested on the following:

- Windows 10 OS
- Eclipse IDE for Java EE Oxygen Package
- Wildfly 11
- Java 1.8
- Internet Explorer/Microsoft Edge Browser
- Google Chrome Browser

Library Dependencies:

- JSTL 1.2.jar
- Mysql Connector Java 5.1.444-bin.jar

LESSONS LEARNED

Steven:

This course and project is my first time interacting with a database and implementing a functional web page. Through this project, I've learned how jsp or web pages communicate with the database through servlets, how to connect to the database, and retrieve data. ER Diagrams and table documentations really helped organize and directly reflect how our database will look like and function. This contributed a lot to our project planning and it undergo massive changes as we continued to developed the program as we had gotten better idea and understanding on what we were trying to create and what our limitations were such as time and skill level. I've also learned how to design web pages through css and html and both which I did found frustrating to work with.

Aye:

This project give me a second chance to really understand in the web based languages such as HTML, CSS, JDBC, java server page and Javascript even though I had learned in other classes. In addition to those languages, I have learned how to design an application through ER diagram and used the knowledge in the implementation of database schema in the MySQl workbench, as well as the extensive use of sql language. I also have learned how to communicate with team members and work together as a team member.

Geordi:

The Music Vault was my first foray in databases, SQL programming, and 3-tiered web app design. I can walk away knowing more about the stack interactions of a fully-functioning web application. Creating documentation iteratively at all stages of design and implementation was also a new experience. This iterative process brought a greater understanding of proper software development practices and their importance. Overall, I have gained greater confidence in my abilities as a technical writer and a web programmer. I can say that I have learned more from this project than any other project I have had in my undergraduate studies.

INSTRUCTIONS FOR SETUP

Installing/Setting up Eclipse

1. Install Eclipse IDE for Java EE Developers Oxygen Version
2. Open Eclipse and on the top left, go to Help→Eclipse Marketplace. Search for Jboss and install “JBoss Tools 4.5.1 Final”.
3. If Server window is not in view, Go to Windows→Show View→Servers.
4. Right Click anywhere in the Server window space and then New→Server.
5. Look under Jboss Community and Select “Wildfly 11”. Go to “Next”.
6. If you do not have any runtime installed, step 6.1 may be skippable as it will take you straight to the “JBoss Runtime” page.
    6.1 Under “selected profile requires a runtime”, if you do not have a Wildfly runtime, click on the drop down menu and select “Create new runtime (next page)”.
7. Download and Extract our project files anywhere you like.
8. In JBoss Runtime window, under Home Directory, click on “Browse”. Search for “wildfly-11.0.0.CR1” and select it. This folder can be found within our project files.
9. Hit “Finish”.
10. Now go to File→New→Dynamic Web Project.
11. Project Name should be “CS157A” (This is important), and Target Runtime should be “WildFly 11.0 Runtime”.
12. Hit “Finish”.
13. Navigate to your project’s directory and start moving the project’s files in. All java files should be in CS157A\src folder and all jar files (mysql-connector-java-5.1.44-bin.jar and jstl-1.2.jar) should be in CS157A\WEB-INF\lib. The rest of the files should be in CS157A\WebContent.

Setting up MySQL Database

14. Assuming you’ve already have MySQL installed, and already have the knowledge on using it, we’ll quickly go over on how the database should be set up.
15. Create a database called “themusicvault”. (CREATE DATABASE themusicvault;)
16. Select themusicvault database and make sure it’s highlighted bold.
17. Go to File→Open SQL Script.
18. Navigate to the sql files extracted from our project files.
19. Execute sql files and repeat for all 6 sql files.

20. There should be 6 tables (profile, users, albums, artists, ratingCheck, and songs).

Database Connection

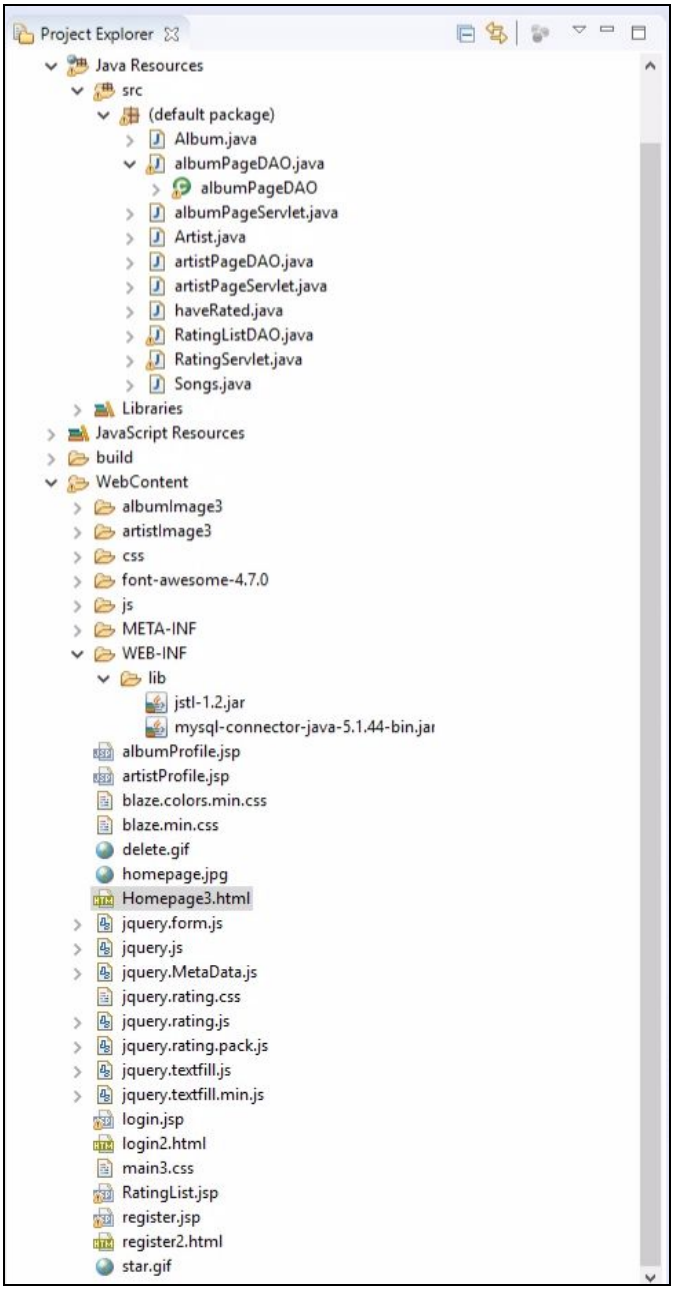
This project connects to MySQL database with the following: **(jdbc:mysql://localhost:3306/TheMusicVault?autoReconnect=true&useSSL=false","root","root")**

We are assuming your account login name and password for mySQL is **root** and **root**.

The database name is **themusicvault**.

Local Host **3306**.

Note: If you are having 'port already in use' issues. It is possible that the port is already being used by 'NVIDIA Network Service' if you have Nvidia. You can temporarily disable this through Task Manager→Services. Search for 'NVIDIA Network Service', right click and select 'Stop'.



Running The Project

1. The overall locations of the files should look like the directories on the right.
2. Start by running Homepage3.html (Run→Run) and have Wildfly 11 Server selected.
3. Optional: If you like, you can run this project within your web browser. (Window → Web Browser) and Select your favorite web browser. When running the file, it will automatically run on your browser.
4. Note: All the coding we've done are in the Java, JSP, and HTML files. The other files are mostly for the GUI.