Report for Final Project

Title of the project: My Music Diary

Name of the students: Junyi (Jenny) Wei, Fangying Zhan (Group 26)

1. Introduction:

1.1. Motivation

Listening to music using music player applications has become a trend nowadays, and many interesting music players have appeared online for downloading. However, it is hard to find a personal music application that provides a fast and accurate query tool for accurate music information, as well as a private and hearty place for each song to keep a diary of your mood and thoughts when listening to it. So, we decide to design our application "My Music Diary" in order to meet these potential customer demand.

1.2. Application description

Our application has several highlighted functions. Firstly, as a personal diary application, we protect the privacy for every user by providing a login step using a user designed login name and password pair. Secondly, users can easily search for detailed information about each song, album or artist in our database by inputting partial name of the target, in case that the user forgets the full name. Thirdly, if the user finds any song, album or artist appeared in the application to be interesting, they can get all the information of it including the corresponding song list by a simple click on the name. Moreover, our application will provide a quick access list to the recent diaries and the user's favorite songs. It will also recommend new songs every day basing on the user's past preferences. Last and the most importantly, the user can keep a music diary for each song and access to them easily by the recent diary list or the song information page.

1.3.Task assignment for each team member

In order to design and implement this application, the tasks are roughly divided as following: Jenny designed the interface and conducted evaluation. Fangying designed all the queries. We implemented the b+ tree and interface together. For the final submission, we wrote the report together, while Fangying made the application demo video and Jenny made the slides for presentation. Jenny did the presentation.

1.4. Report organization

In this report we will first talk about the system architecture of our application, then we will describe our data set, including our ER diagram and relational model derived from our dataset. After that, we will come to detail of our implementation and describe the prototype. Then we will talk in detail about the testing methods we use to guarantee our application to work well. Finally, we will discuss the lessons we learned by doing this project.

2. Our Implementation

2.1. Description of the dataset

We used the We will use the Amazon Music dataset from The 784 Data Sets for EM[1]. It consists of 55923 records is relatively clean and sufficient for the implementation of our database management project. It provides data for important entities including songs, albums, artists, and music labels with various attributes including song prices, genres, and copyright information, which are necessary for the development of a music software. In addition to the song data, our creative diary part of the application will maintain contents from user data, which is another source for this database. The structure of the data set is shown in the following table:

Attribute	Atomic Domain	Description
Sno	Int	The label and ordering of the song
Album_Name	String	The name of the album that the song belongs to;
Artist_Name	String	The name of the artists who perform in this song
Price	Float	The price of the song in the market
Time	time	The duration of the song
Released	date	The released date of the song

Label	String	The company that produces the song
Copyright	String	The organization that the copyright of the song belongs to
Genre	String	The genre that the song belongs to

Table 1: Structure of the original Amazon music dataset.

This table shows the structure and content of the original Amazon music dataset.

2.2. ER Diagram

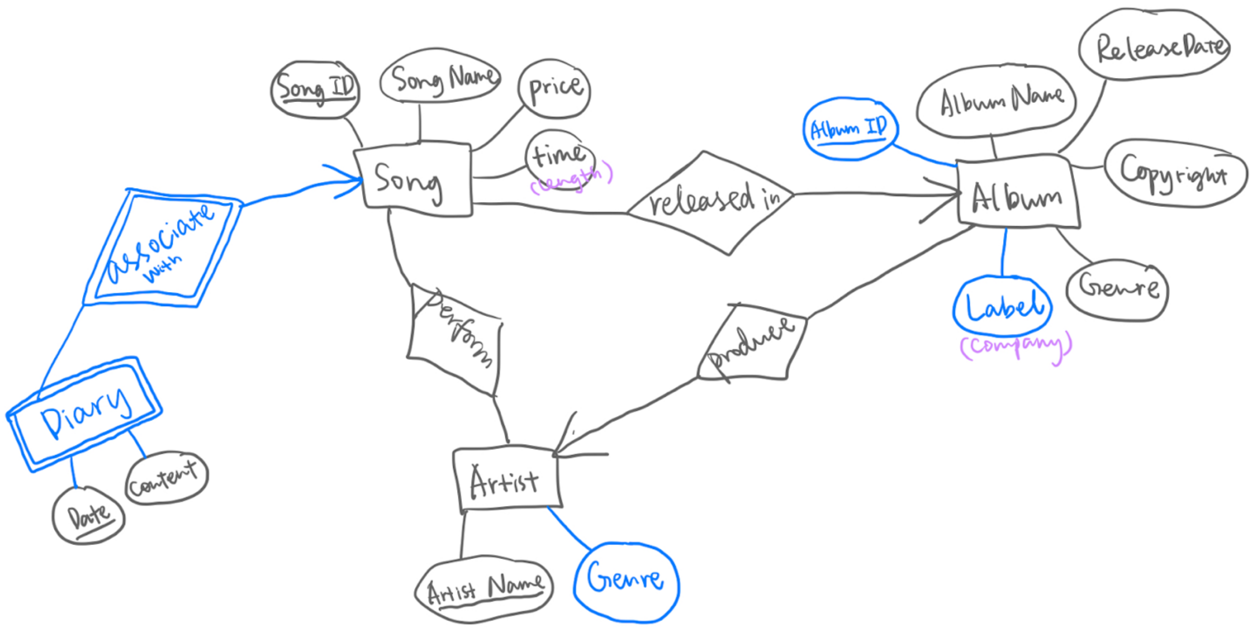


Figure 1: ER Diagram

This figure shows the ER diagram designed for Application implementation use.

Our ER diagram contains 4 entity sets: Artist, Song, Album, Artist and Diary, where Diary entity set is created by ourselves to record the diary inputs of the users. Diary serves as a weak entity because a diary has to belong to some song and it cannot exist alone. It has 4 relationships: each song belongs to one album; each album is produced by one Artist; each artist can perform in several songs and each song can have several artists performing together.

2.3. Relational Schema

When designing the relational schema, we always combine the many-to-one relationship table with the entity set table to simplify our structure. Since associate is a weak relationship, released in and produce are all many-to-one relationship, we only have 5 tables in total after the combination.

Relational schema	Primary key	Additional key
Song(song_id: String, song_name: String, price: Real, time: String, album_id: String)	song_id	album_id + song_name
Album(album_ID: String, album_name: String, copyright: String, release_date: String, album_genre: String, label: String, producer_name: String)	album_id	album_name + release date
Artist:(artist_name: String, artist_genre: String)	artist_name	<None>
Diary(song_id: String, date: String, content: String)	song_id + date	<None>
Perform(song_id: String, artist_name: String)	song_id + artist_name	<None>

Table 2: Relational Schema

This table shows the relational schema and pointed the primary key and other candidate keys for each relation in our model.

2.4. Normalization

Because of the relatively simple structure of our data set, all the non-trivial functional dependencies of all the relationships have their left-hand-side as a superkey, it is already in BCNF (so of course 3NF) and we don’t need to normalize it anymore.

Relation	Nontrivial FDs	Best Normal Form Achieved
Song	Song_ID → (Song_Name, Price, Time, Album_ID) (Song_Name, Album_ID) → (Song_ID, Price, Time)	BCNF: all the non-trivial FDs have their left-hand-side as a superkey
Album	Album_ID → (Album_Name, Copyright, Release_Date, Album_Genre, Label, Producer_Name) (Album_Name, Release_Date) → (Album_ID, Label, Producer_Name, Copyright, Album_Genre)	BCNF: all the non-trivial FDs have their left-hand-side as a superkey
Artist	Artist_Name → Artist_Genre	BCNF: all the non-trivial FDs have their left-hand-side as a superkey
Diary	(Song_ID, Date) → Content	BCNF: all the non-trivial FDs have their left-hand-side as a superkey
Perform	<None>	BCNF: all the non-trivial FDs have their left-hand-side as a superkey

Table 3: Functional Dependency and Normalization

This table shows the functional dependency and the best normal form achieved by each relation involved in our model

2.5. Description of the system architecture

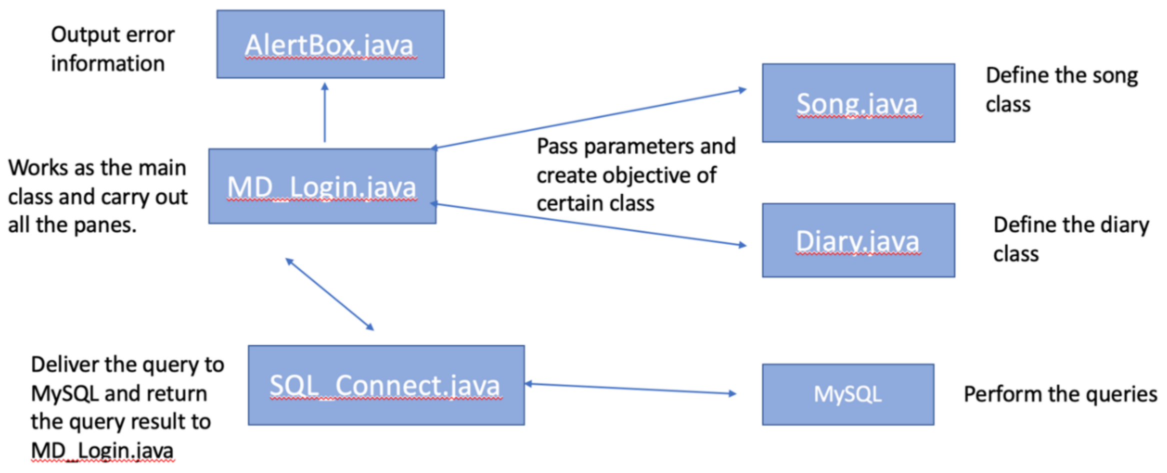

Output error information
AlertBox.java
Song.java
Define the song class
Works as the main class and carry out all the panes.
MD_Login.java
Pass parameters and create objective of certain class
Diary.java
Define the diary class
Deliver the query to MySQL and return the query result to MD_Login.java
SQL_Connect.java
MySQL
Perform the queries

Figure 2: System Architecture

This table shows the architecture of our implementation, including the main functionality of each part and how they communicate with each other

Our implementation can be roughly separated into 3 parts: user interface, SQL connector and database. The user interface creates and display the panes which can interact with user and receive query requirements. The SQL connector passes queries received from the interface to the DBMS. The DBMS performs queries in the dataset and return the query result to the connector again. The connector passes the result to the interface and the interface will create pane and display the query result.

There are 5 java files in total in our application:

Song.java is a class that can receive parameters from MD_Login.java and creates objectives that contains all the information of a song.

Diary.java is a class that can receive parameters from MD_Login.java and creates objectives and contains all the information of a Diary.

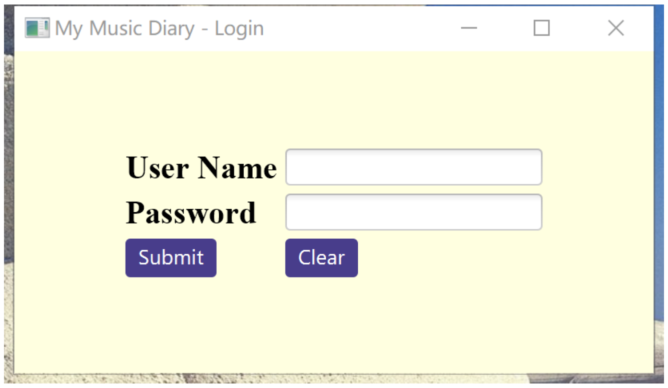
MD_Login.java serves as a main function and the interface. It creates and displays most of the panes and receive interactions from the users. It will pass the query information to SQL_Connect.java and receive the query result from there. It can display the query result and display error message through AlertBox.java if any error happens.

Alertbox.java creates an alert box when there is an error in the program.

SQL_Connect.java connects MD_Login.java and DBMS. It helps login to DBMS, passes queries to DBMS and return the query result to MD_Login.java.

2.6. Description of the prototype

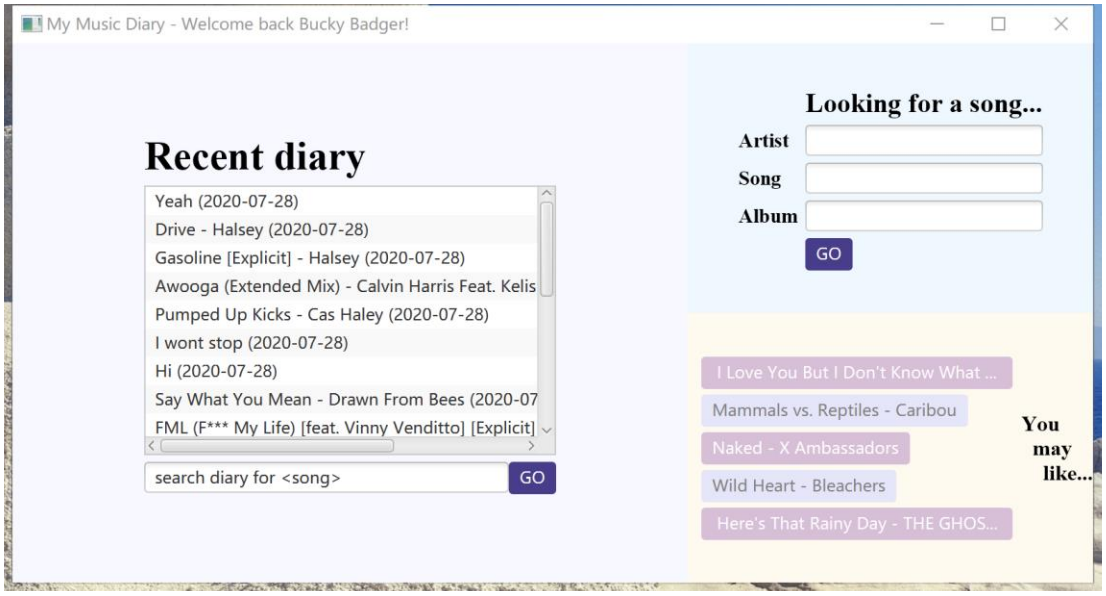
Pane 1: login page. Here you can enter the username and password pair designed by you to login to the main page.


My Music Diary - Login

User Name

Password

Pane 2: main page. Here you can see three parts: Recent diary shows all the recent diaries you have created. Under the list there is a searching engine that you can search for a diary by entering the song name. On the right-hand-side there is also a searching engine. You can search for any song, album or artist information by inputting their names. “You May Like” part will recommend 5 songs that are close to your taste.


My Music Diary - Welcome back Bucky Badger!

Recent diary

Yeah (2020-07-28)
Drive - Halsey (2020-07-28)
Gasoline [Explicit] - Halsey (2020-07-28)
Awooga (Extended Mix) - Calvin Harris Feat. Kelis
Pumped Up Kicks - Cas Haley (2020-07-28)
I wont stop (2020-07-28)
Hi (2020-07-28)
Say What You Mean - Drawn From Bees (2020-07
FML (F*** My Life) [feat. Vinny Venditto] [Explicit]

search diary for <song>

Looking for a song...

Artist

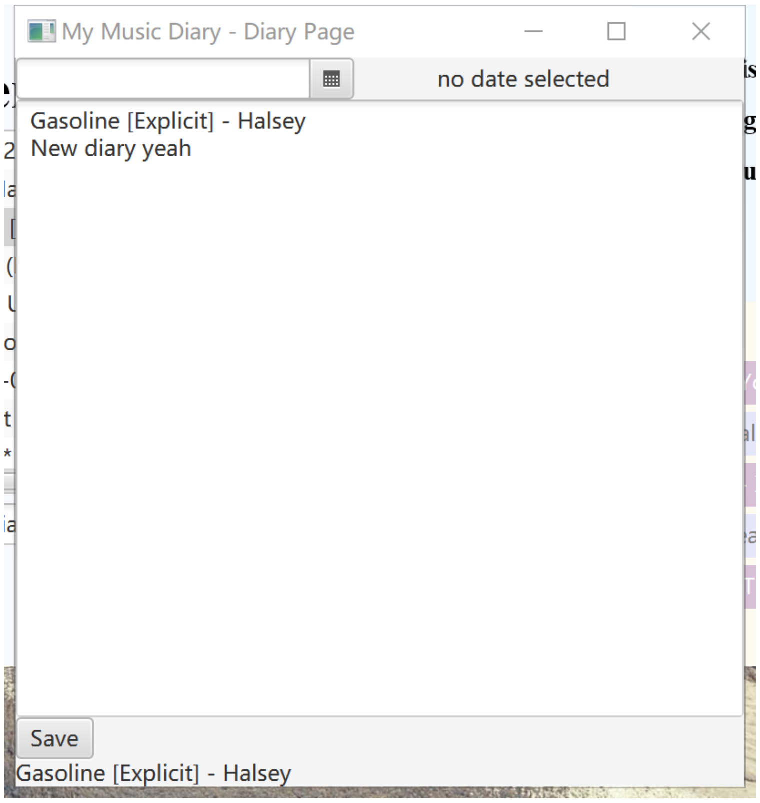
Song

Album

I Love You But I Don't Know What ...
Mammals vs. Reptiles - Caribou
Naked - X Ambassadors
Wild Heart - Bleachers
Here's That Rainy Day - THE GHOS...

You may like...

Pane 3: when you click any diary in the diary list, you will go to the diary page. Here you can select the date of the diary and edit the content. If you clear the content and click save, the diary will be deleted.


My Music Diary - Diary Page

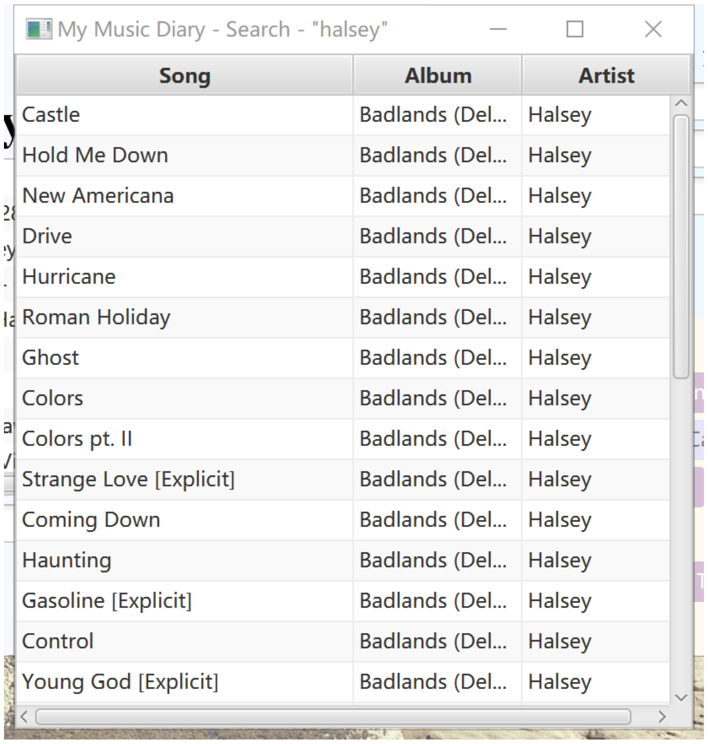
no date selected

Gasoline [Explicit] - Halsey
New diary yeah

Save

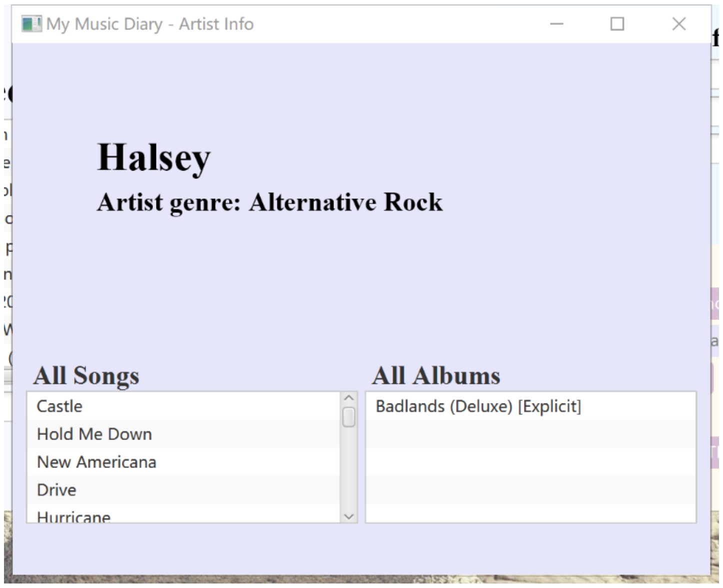
Gasoline [Explicit] - Halsey

Pane 4: If you input any information in the searching engine "search for a song", it will show a list of all the song-Album-Artist tuple results that matches the searching criteria.


My Music Diary - Search - "halsey"

Song	Album	Artist
Castle	Badlands (Del...	Halsey
Hold Me Down	Badlands (Del...	Halsey
New Americana	Badlands (Del...	Halsey
Drive	Badlands (Del...	Halsey
Hurricane	Badlands (Del...	Halsey
Roman Holiday	Badlands (Del...	Halsey
Ghost	Badlands (Del...	Halsey
Colors	Badlands (Del...	Halsey
Colors pt. II	Badlands (Del...	Halsey
Strange Love [Explicit]	Badlands (Del...	Halsey
Coming Down	Badlands (Del...	Halsey
Haunting	Badlands (Del...	Halsey
Gasoline [Explicit]	Badlands (Del...	Halsey
Control	Badlands (Del...	Halsey
Young God [Explicit]	Badlands (Del...	Halsey

Pane 5: if you click any artist name, this pane will appear and show the detailed information the song list and the album list of this artist.


My Music Diary - Artist Info

Halsey

Artist genre: Alternative Rock

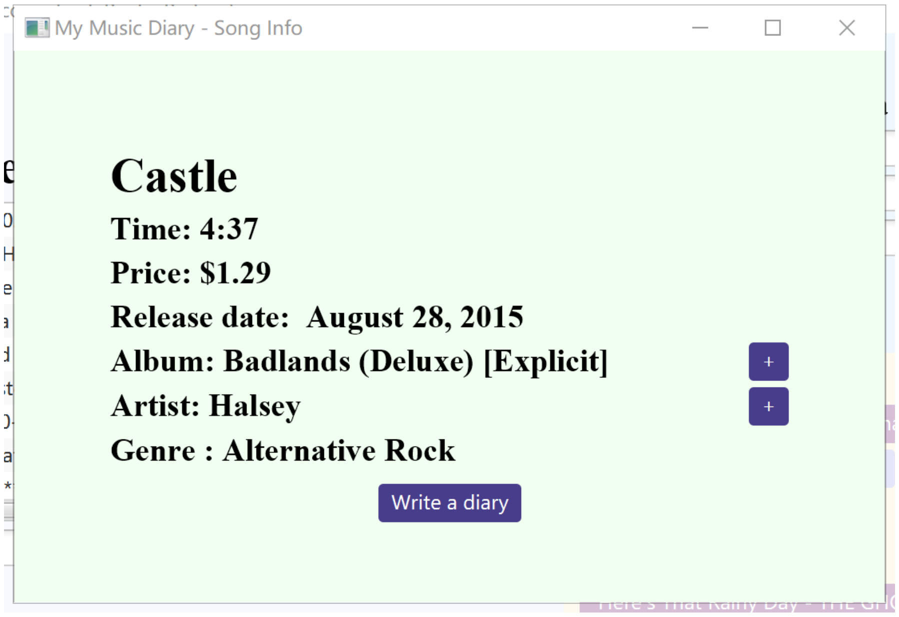
All Songs

Castle
Hold Me Down
New Americana
Drive
Hurricane

All Albums

Badlands (Deluxe) [Explicit]

Pane 6: If you click any song name, this pane will appear and show the detailed information of the song. Clicking “write a diary” here will direct you to the diary page again.


My Music Diary - Song Info

Castle

Time: 4:37

Price: $1.29

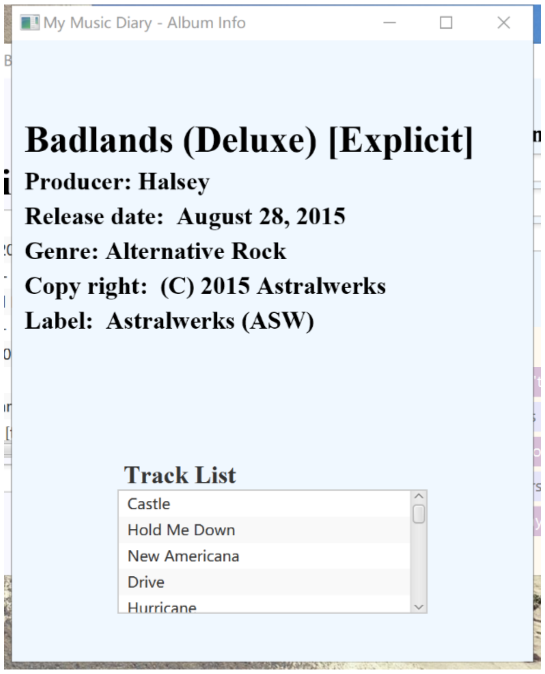
Release date: August 28, 2015

Album: Badlands (Deluxe) [Explicit]

Artist: Halsey

Genre : Alternative Rock

Pane 7: If you click any album name, this pane will appear and show the detailed information of this album, including its song list.



Badlands (Deluxe) [Explicit]

Producer: Halsey

Release date: August 28, 2015

Genre: Alternative Rock

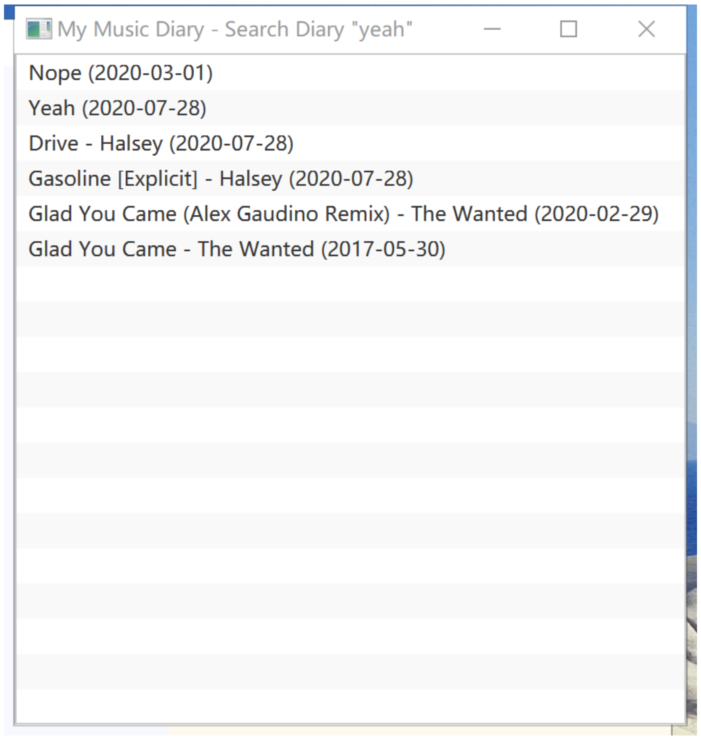
Copy right: (C) 2015 Astralwerks

Label: Astralwerks (ASW)

Track List

Castle
Hold Me Down
New Americana
Drive
Hurricane

Pane 8: If you use the "searching diary" search engine in the main page, this pane will appear to show all the diaries whose song name contains the words that you input.



Nope (2020-03-01)
Yeah (2020-07-28)
Drive - Halsey (2020-07-28)
Gasoline [Explicit] - Halsey (2020-07-28)
Glad You Came (Alex Gaudino Remix) - The Wanted (2020-02-29)
Glad You Came - The Wanted (2017-05-30)

2.7. Evaluation

Query Functionality	SQL Executable Statement	Input	Output	Status
Create a new diary entry with given song id, date (default is current date), and diary content	insert into diary values (<song_id>, <date>, <content>);	song clicked' 'date specified' 'input content' 'song clicked' 'no date specified' 'input content' 'song clicked' 'no date specified' 'no input content'	(1st line content with specified date in diary list) (1st line content with current date on top of diary list) (song name - artist name with current date on top of diary list)	Tested Correct
Delete a diary entry	delete from diary where song_id = <song_id> and date = <date>;	diary with song clicked' 'empty content'	(diary entry no longer appear in diary list or diary search list; new diary entry associated with the same song and date available for creating)	Tested Correct
Modify and save a diary entry on the recent/search diary list	update diary set content = <content>, date = <new_date> where song_id = <song_id> and date= <old_date>;	diary with song clicked' 'modified content'	(new keyword of diary content can be searched; correct in search table; content is saved)	Tested Correct
Display recent diary list on the main page	select song_id, date, content from diary order by date desc;	no input' 'modified date for a diary entry'	(ordered recent list on main page) (updated recent list on main page for the modified diary entry)	Tested Correct
Search for all diary entries containing the search key word or has the song name	select d.song_id, d.date, d.content from song s, diary d where s.song_id = d.song_id and (s.song_name like '%<searchKey>%' or d.content like '%<searchKey>%');	search key in content' 'search key as song name' 'search key as both song name and content'	(correct search list of diary entries display)	Tested Correct
Search for all songs containing the keywords in song name/artist name/album name	select s.song_name, al.album_name, al.prod_name, s.song_id from song s, album al where s.album_id = al.album_id and s.song_name like '%<song>%' and	song name' 'artist name' 'album name' (any combinations)	(correct search table resutls of songs display)	Tested Correct

	al.album_name like '<album>%' and al.prod_name like '<artist>%';			
Random song recommendations display on main page	select s.song_name, al.album_name, al.prod_name, s.song_id from song s, album al where s.album_id = al.album_id and album_genre like '%<genre>%' order by rand() limit 5;	last new diary entry assiciated song'	(recommendations in the same first genre as the song)	Tested Correct
Album list on an artist info page	select album_name, album_id from album where prod_name like '<artistName>';	artist on selected song page'	(correct album list of this artist)	Tested Correct
Song list on an artist info page	select song_name, song_id from song s, album al where s.album_id = al.album_id and prod_name like '<artistName>';	artist on selected song page'	(correct song list of this artist)	Tested Correct
Song list of an album on album info page	select song_name, song_id from song where album_id =<albumid>";	album on selected song page'	(correct song list of this album)	Tested Correct
Artist info on an artist page	select artist_name, artist_genre from artist ar, song s, album al where s.album_id = al.album_id and al.prod_name = ar.artist_name and s.song_id = <songid>;	artist on selected song page'	(correct artist information)	Tested Correct
Album info on an album page	select album_name, prod_name, release_date, album_genre, copyright, label from album where album_id = <albumid>;	album on selected song page'	(correct album information)	Tested Correct

Song info on a song page	select s.song_name, s.time, s.price, al.album_name, al.prod_name, al.release_date, al.album_genre from song s, album al where s.album_id = al.album_id and s.song_id = < songid>;	selected song on song search list' 'selected song from recommendations'	(correct song information)	Tested Correct

3. Conclusion

We learned a lot of useful materials from this project. First of all, this is the first time we use java to connect to another software use a connector, and this skill can be generalized to many different software. Java is no longer limited as java itself and it can be the controller of many useful software. We also learned about how to efficiently design the evaluation for our application. This is an important debugging skill and will help us a lot in the future.

The knowledge of database is very useful. We have entered the era of big data and there is increasing chance for us to deal with huge amount of data. For example, when we are doing machine learning projects, we always read and write the entire dataset at once and I/O cost becomes the most expensive cost in the learning procedure. However, we learned from this course that there are many mature ways of reducing the I/O cost: we can split the data into several sub tables, and always do selection and projection first if we only use partial information.

Last but not least, we learned about how to cooperate with our teammates, efficiently distribute the workload, create time plan, and achieve the result step by step. This is a very valuable experience and we will appreciate it a lot.

4. References

[1] Das, SanjibDoan, AnHai and G. C., Paul Suganthan and Gokhale, Chaitanya and Konda, Pradap and Govind, Yash and Paulsen, Derek, *The Magellan Data Repository,* retrieved from \url{https://sites.google.com/site/anhaidgroup/projects/data.