1. INTRODUCTION

1.1 Project Overview

Rapid development of mobile devices and internet has made possible for us to access different music resources freely. The number of songs available exceeds the listening capacity of single individual. People sometimes feel difficult to choose from millions of songs. Moreover, music service providers need an efficient way to manage songs and help their costumers to discover music by giving quality recommendation. Thus, there is a strong need of a good recommendation system. Currently, there are many music streaming services, like Pandora, Spotify, etc. which are working on building high-precision commercial music recommendation systems. These companies generate revenue by helping their customers discover relevant music and charging them for the quality of their recommendation service. Thus, there is a strong thriving market for good music recommendation systems.

The explosive growth in the amount of available digital information and the number of visitors to the Internet have created a potential challenge of information overload which hinders timely access to items of interest on the internet. Information retrieval systems, such as Google have partially solved this problem, but prioritization and personalization of information were absent. This has increased the demand for recommender systems more than ever before. Recommender systems are information filtering systems that deal with the problem of information overload by filtering vital information fragment out of large amount of dynamically generated information according to user's preferences, interest, or observed behavior about item. Recommender system could predict whether a particular user would prefer an item or not based on the user's profile.

Music recommender system is a system which learns from the users past listening history and recommends them songs which they would probably like to hear in future. The project involves implementation of various algorithms to build an effective music recommender system.

• Firstly, it involves implementation of Non-Personalized popularity-based model which provides each song in descending order of popularity skipping those songs already consumed by the user.

• The recommender system also provides the user with the greatest hits of the artist that the user has already listened to.

• Collaborative filtering algorithms which predict (filtering) taste of a user by collecting preferences and tastes from many other users (collaborating) is also implemented.

• The project also involves experiments on content-based models, based on latent factors and metadata.

We used data provided by Million Song Data [6] Challenge hosted by Kaggle. It was released by Columbia University Laboratory for the Recognition and Organization of Speech and Audio. The data is open; meta-data, audio1 content analysis, etc. are available for all the songs. It is also very large and contains around 48 million (userid, songid, play count) triplets collected from histories of over one million users and metadata (280 GB) of millions of songs. But the users are anonymous here and thus information about their demography and timestamps of listening events is not available. The feedback is implicit as play-count is given instead of explicit ratings. The contest was to predict one half of the listening histories of 11,000 users by training their other half and full listening history of other one million users. Since, processing of such a large dataset is highly memory and CPU-intensive, we used validation set as our main data. It consists of 10,000,000 triplets of 10000 users. We used metadata of only 10,000 songs (around 3GB). From the huge amount of song metadata, we focus only on features that seem to be most relevant in characterizing a song. We decided that information like year, duration, hotness, danceability, etc. may distinguish a song most from other songs. To increase processing speed, we converted user and song ids from strings to integer numbers.

1.2 Need Analysis

With the rise of digital content distribution, people now have access to music collections on an unprecedented scale. Commercial music libraries easily exceed 15 million songs, which vastly exceeds the listening capability of any single person. With millions of songs to choose from, people sometimes feel overwhelmed. Thus, an efficient music recommender system is necessary in the interest of both music service providers and customers. Users will have no more pain to make decisions on what to listen while music companies can maintain their user group and attract new users by improving users' satisfaction.

In the academic field, the domain of user centric music recommendation has always been ignored due to the lack of publicly available, open and transparent data. Million Song Dataset [1] Challenge provides data which is open and large scale which facilitates academic research in user centric music recommender system which hasn't been studied a lot.

Using the music streaming services can represent an innovative and superior experience for the user. One important reason why more and more people choose to use music streaming services is that they thus can build up a massive music collection at low cost. However, this advantage also entails a problem: information overload. This problem becomes obvious on streaming websites.

Facing a massive collection of music, users are unable to make a decision and have no idea of what to listen to. The users sometimes have problems discovering new songs when using music streaming websites. They wish the streaming websites to provide recommendations for them. Even according to the CEO and founder of Spotify, Daniel Ek, users have frequently voiced their desire of finding new music to listen to. Obviously, a music recommender system is essential in music streaming websites. Users demand an effective music recommender system because music streaming websites offer numerous items to choose from within a limited period of time which is insufficient to evaluate all possible options. With the help of a recommender system, users can skip over the information overload and get customized recommendations from the system

1.3 Problem definition and Scope of project

The central issue is the recommendation of songs to a user. For a given user, we have their song history and play count for each song. From this, we want to produce a set of recommendations to the user. Then, we try to answer the question, "How do we use the song history of user to provide recommendations that they will like? "Generally, this is done by looking at the songs that are most similar to the user's songs, as well as the users who are most similar to the user according to their listening history. In order to discern whether songs are similar to those in a query user listening history, our methods use the metadata collected from a song to weigh its similarity with another song. From this, we can choose the songs with the highest weights as the recommended songs. Another problem that we considered is the cold-start and near-cold-start problem. The cold-start problem, in terms of song recommendations, refers to the situation in which the test user has no song history; since all our metrics use this for song prediction, a lack of song history is a difficult problem for our algorithms. It can be solved by simply recommending a subset of the most popular songs. Lastly, we have the issue of dealing with large quantities of data. The Million Song Data Set and accompanying datasets offer over 280 gigabytes of information about songs, their metadata, and user play counts. How do we deal with all the data? Often, we simply use a random sample of the data for training, validation, and testing. For example, we use the data in our user song history set with only the top one thousand songs in the song history.

1.4 Objectives

• Personalized music recommendation system with the goal of predicting the songs that a user is going to listen.

• Measure the precision and recall of different algorithms and decide which is more efficient to apply on the dataset.

• Integrate this algorithm into a webpage where user can login and search for their music and get personalized recommendations according to their taste.

• To improve overall efficiency of the recommendation system using more optimized algorithms.

1.5 Methodology

Data Set: Data is provided by Million Song Data Challenge hosted by Kaggle. It was released by Columbia University Laboratory for the Recognition and Organization of Speech and Audio.

Algorithms: Various algorithms to be implemented build an efficient recommendation system.

• Popularity based Model: It is the most basic and simple algorithm. We find the popularity of each song by considering the training set and calculating the number of users who had listened to this song. Songs are then sorted in the descending order of their popularity.

• Collaborative based Model: Collaborative filtering involves collecting information from many users and then making predictions based on some similarity measures between users and between items. This can be classified into user-based and item-based models.

• In item-based model, it is assumed that songs that are often listened together by some users tend to be similar and are more likely to be listened together in future also by some other user.

• According to user-based similarity model, users who have similar listening histories, i.e., have listened to the same songs in the past tend to have similar interests and will probably listen to the same songs in future too.

• Singular Value Decomposition Model: Listening histories are influenced by a set of factors specific to the domain (e.g. genre, artist). These factors are in general not at all obvious and we need to infer those so called latent factors from the data. Users and songs are characterized by latent factors.

• K Nearest Neighbors Mode: In this method, the available metadata is utilized. A space of songs is created according to their features from metadata and find out neighborhood of each song. Some of the available features are chosen (e.g., loudness, genre, mode, etc.) which are found to be most relevant to distinguish a song from others. After creating the feature space, to recommend songs to the users, each user's profile is looked and suggest songs which are neighbors to the songs present in his listening history. Evaluation Metrics: Precision and recall are used as evaluation metrics for the algorithms.

1.6 Assumptions

The Internet connection is a constraint for the application. Since the application fetches data from the server over the Internet, it is crucial that there is an Internet connection for the application to function. Execution time for the algorithm should take no longer than one second. Music Recommender will be a sub component of Music website. Users shall be required to log in to the website to get recommendations. It also needs to give unique recommendations for each user. Another assumption is that the user has a web browser and a capable hardware in order to launch the website. It is also assumed that system is able to provide necessary requirements for the recommender system to run.

1.7 Project Outcomes

• Working software prototype.

A web application would be available for user in which user can search for music and get recommendations based on its history.

• Accurate Recommendations.

Various algorithms are analyzed and compared based on precision and recall providing accurate recommendations to the user.

• User friendly web applications.

User will be able to easily search for his choice of music with help of user friendly interface which can be accessed by people from all age groups.

1.8 Project Schedule

Table 1: Project Schedule

Task Name	Start	End	Duration
Planning and SRS	4/02/2018	25/02/2018	21
Schedule and Task Management	03/03/2018	20/03/2018	17
Learning of tools	25/03/2018	10/04/2018	16
Study of algorithms	15/04/2018	15/05/2018	31
Implementation of algorithms	20/05/2018	30/06/2018	40
Software design & build	01/07/2018	30/07/2018	29
System Testing	01/08/2018	30/08/2018	29
Product Refinement	01/09/2018	16/09/2018	15
Documentation	17/09/2018	17/10/2018	30
Customer Survey and Feedback	18/10/2018	30/10/2018	12
Final Presentation	01/11/2018	01/12/2018	30

2. LITERATURE REVIEW

2.1 Background

It is obvious that music plays an important role in many people's daily life. No matter if one is a huge fan of music or just randomly listens to music for fun, it cannot be denied that music is a major entertainment factor. From vinyl records to cassette tapes, from CD to mp3, the ways of listening to music have changed. With the help of technology, music can be enjoyed in a more and more convenient way.

Nowadays, with the rapid development of the Internet, it is getting common to use music streaming services. Compared to other ways of providing music, streaming websites can provide more and better services. There are a lot of advantages of using music streaming websites: customers pay less to listen to music than with iTunes or real CDs; the number of music collections in streaming websites is huge; it is much more convenient to listen to music online etc.

According to Karp (2014) [3], in the first half of 2014, the number of downloads of singles and albums dropped by 11% and 14%, whereas the number of users of streaming services increased by 28%; these figures make it obvious that more and more people have changed their ways of listening to music.

Streaming music services started to change people's habits of listening to music. There are already several music streaming websites, for instance: Spotify, Beasts music, Pandora. Some big companies have started their own music recommendation services, for example: Google play music, Sony music unlimited, X-box music etc. The user group of streaming services is gigantic in number.

Using the music streaming services can represent an innovative and superior experience for the user. One important reason why more and more people choose to use music streaming services is that they thus can build up a massive music collection. However, this advantage also entails a problem: information overload. This problem becomes obvious on streaming websites. Facing a massive collection of music, users are unable

to make a decision and have no idea of what to listen to. The users sometimes have problems discovering new songs when using music streaming websites. They wish the streaming websites to provide recommendations for them. Even according to the CEO and founder of Spotify [8], Daniel Ek, users have frequently voiced their desire of finding new music to listen to. Obviously, a music recommender system is essential in music streaming websites. Users demand an effective music recommender system because music streaming websites offer numerous items to choose from within a limited period of time which is insufficient to evaluate all possible options. With the help of a recommender system, users can skip over the information overload and get customized recommendations from the system.

2.2 Existing Systems

To meet users' demands for a recommender system, there are some music streaming websites already providing music recommendation services, for example: Spotify [9], Pandora, Beats music etc. The ways how they compile their recommendation lists varies between companies. Some websites make up recommendations based on users' listening records; some recommend the music that the "neighbor user" listens to, which means that the system assumes that they share a similar taste, and other websites recommend music based on user's mood. Although there are already lots of different ways to draw up recommendations, users are still not satisfied with the recommendation service.

2.3 Problem Identified

According to Karp (2014) [4] the music recommended by the recommender systems in music streaming websites does not match with the user's taste. Sometimes the music recommended was completely different from what they like. A music recommender system, however, is supposed to provide good recommendations for users to solve the information overload problem. However, it has become obvious that the music recommender systems do not meet the demands of the users. The question is: What causes this problem? [7] There must be some drawbacks existing in the current music recommender systems. A music recommender system consists of several different components, such as:

- The way of drawing up recommendations: whether the system compiles the recommendations based on data of users' behaviors or users' mood or the "neighbor user's" taste
- The interface design: whether it is easy for users to understand and apply.
- The feedback system: whether it can actually support the recommender system to get feedback from users and in this way to improve the service. Drawbacks in any part of the recommender system may lead to the "un-customized" problem: the recommendations provided by the system are not tailored to users' demand.

2.4 Methods and Tools

The current music recommendation systems use the following two algorithms:

2.4.1 Content Based Recommendation

Content-based [10] recommendation is an inheritance and development of information filtering technology. It is based on the content of user profiles to provide a recommendation service without the user's evaluation. Content-based recommendation uses machine language to acquire information on the user's interest by relating to the content of the user profile. Using characterization methods, content-based recommendation can offer some choices to the user and then get the user's feedback in content-based recommender systems, the items or objects are defined by characteristics and related attributes. Content-based recommendation approaches predict the user's interest by using text only, users' ratings are not involved during the process of the prediction. The user data model depends on the learning method. Decision trees, neural networks and vector-based representation methods are commonly used. The user data model perhaps varies from user to user. The advantages of the content-based recommendation approach are summarized as follows:

- Other user information is not required in the recommendation process, it is easier to provide the recommendation service at the initial stage of the system
- It can provide recommendation service to users with special interests
- It is able to recommend new or "not mainstream" items
- It can list the recommended items by content characteristics.

2.4.2 Collaborative Filtering

Collaborative filtering [5] recommendation is one of the earliest applied techniques and it has successfully spread and entered the recommender system field. It generally uses the “K-nearest neighbor (KNN)” technique which is based on the user’s historical records. The music taste of users calculates the distances between the different users. Collaborative filtering recommendation uses the target “user's nearest neighbor user” to weight and evaluate the value of the product. Collaborative filtering recommendation predicts the extent of user's preference for a specific target product.

Based on the usage data and the user’s interests, the system searches the “neighbor users” [2] who share similar interests with the user. Then the recommendation system recommends the contents that “neighbor users” are interested in to the user. The general idea of how collaborative recommender systems work is easy to understand. In other words, this approach is also commonly accepted in daily life, people refer to friends’ recommendations when making their own decisions. Collaborative filtering has been frequently used in e-commerce recommender systems during the past few years. Collaborative filtering provides recommendations for the target users based on other users’ evaluation of content.

3. Requirement Analysis

3.1 Software Requirements Specifications:

3.1.1 Functional Requirements:

a. Client: The client-side of the system will be an application with a user interface that is integrated into a music listening website or application. This application gathers the information from users, investigates some actions of the users, and provides the connection with the server. This application is the client-side interface of the Music Recommender, so it does not include the functionalities of the host music environment such as playing music etc.

● Requesting recommendations: The client-side application must allow user to request recommendations manually and interact with the server to receive recommendations.

● Display recommendations: The application must display the recommendations that are obtained from the server to the user in a proper way by providing a GUI.

b. Server: The server-side system will hold the entire data in a graph database and must include all functionality to perform operations on this database, receive requests from the clients, evaluate, create and send recommendations etc.

● Handle recommendation requests

The server application shall obtain and handle requests for recommendations.

● Store evaluations

The server application shall receive and store music evaluations

● Data updating

The server application shall be able to store the newly retrieved data to the database.

● Generate Recommendations

The server application can produce recommendations by interpreting the content and evaluations by actual user. Server will have a recommendation class which contains functions of algorithm.

3.1.2 Non-Functional Requirements:

a. Performance requirements:

● Accuracy

Since we will give the priority to the accuracy of the software, the performance of the Music Recommender will be based on its accuracy on recommendations.

● Speed

The system should generate and provide personalized recommendations to the users in a reasonable time.

b. Design constraints:

● Hardware Constraints

The system will be integrated with a website. To use recommendation system, user should enter from a personal computer, mobile device with internet connection, tablet etc.

● Software System Attributes

Usability: The software will be embedded in a website. It should be scalable designed to be easily adopted by a system.

Reliability: The system should have accurate results and a fast response to user's changing habits. It also should work properly for a reasonably long amount of time. The probability of self-induced failure must be low.

Security: User profile information will be used, so data security is one of the most important concerns of the system

3.1.3 Use case diagram

Figure 1: Use case diagram

3.1.4 Use Case Template

Table 2: SearchAlbum Template

Use Case Name	**SearchAlbum**
Super Use Case	NA
Actor	User
Brief Description	When user will search for an album, then albums in database corresponding query will be shown to the user.
Preconditions	The registered user must be logged in.
Post-Conditions	User will get his/her searched result on the website.

Table 3: ViewAlbumDetails Template

Use Case Name	**ViewAlbumDetails**
Super Use Case	SearchAlbum
Actor	User
Brief Description	Details corresponding to the typed query will be shown to the user.
Preconditions	User must have searched for the album.
Post-Conditions	Details will be shown about the corresponding album.

Table 4: Login Template

Use Case Name	**Login**
Super Use Case	NA
Actor	User
Brief Description	When the user logs in to the website; Music Recommendation System will be informed and a recommendation session for the user will start and generate recommendations.

Preconditions	User must be registered.
Post-Conditions	User will see home screen of his profile along with some recommended music.

Table 5: Register Template

Use Case Name	**Register**
Super Use Case	NA
Actor	User
Brief Description	When the user creates a new account on the website/application that the recommendation system runs on, also implicitly an account for the user inside the recommendation system will be created to store information about the user profile and statistics to provide better recommendations.
Preconditions	NA
Post-Conditions	User will become a registered member of the website.

Table 6: Logout Template

Use Case Name	**Logout**
Super Use Case	NA
Actor	RegisteredUser
Brief Description	When the user logs out from the website; Music Recommendation System will be informed and the associated recommendation session for the user will be closed.
Preconditions	User must have logged in.
Post-Conditions	NA

Table 7: DownloadAlbum Template

Use Case Name	**DownloadAlbum**
Super Use Case	ViewAlbumDetails
Actor	RegisteredUser
Brief Description	User can download his/her bought albums.
Preconditions	User must be registered and must have bought the album.
Post-Conditions	Album will be downloaded to the user's local machine.

Table 8: BuyAlbum Template

Use Case Name	**BuyAlbum**
Super Use Case	ViewAlbumDetails
Actor	RegisteredUser
Brief Description	User can buy his/her favorite album.
Preconditions	User must have enough credit to buy the album.
Post-Conditions	Album will be available for listening and download.

Table 9: Recharge Template

Use Case Name	**Recharge**
Super Use Case	BuyAlbum, ViewAccount
Actor	RegisteredUser
Brief Description	User can add credit to his account.
Preconditions	User must be a registered user and have valid credit card.
Post-Conditions	Account will have recharged with credited amount.

Table 10: ViewAccount Template

Use Case Name	**ViewAccount**
Super Use Case	NA
Actor	RegisteredUser
Brief Description	User can view his account details. User can check or edit his/her playlists.
Preconditions	User must be a registered user.
Post-Conditions	Users will be able to see his account details.

Table 11: ViewOwnedAlbumList Template

Use Case Name	**ViewOwnedAlbumList**
Super Use Case	ViewAlbumDetails
Actor	RegisteredUser
Brief Description	User can view his/her bought albums.
Preconditions	User must be a registered user.
Post-Conditions	Owned album of the user will be shown and will be available to download.

Table 12: ProvideFeedback Template

Use Case Name	**ProvideFeedback**
Super Use Case	NA
Actor	RegisteredUser
Brief Description	User can rate songs or albums.
Preconditions	User must be a registered user.
Post-Conditions	Ratings will be given to song or album.

Table 13: ModifyRecommendedAlgorithms

Use Case Name	**ModifyRecommendedAlgorithms**
Super Use Case	NA
Actor	Admin
Brief Description	Admin can change the algorithms according precision, recall and accuracy
Preconditions	Not applicable
Post-Conditions	Better accuracy can be obtained.

Table 14: AnalyzeSystemPerformance Template

Use Case Name	**AnalyzeSystemPerformance**
Super Use Case	NA
Actor	Admin
Brief Description	Precision recall and accuracy can be monitored.
Preconditions	NA.
Post-Conditions	Whole system performance is well monitored.

Table 15: ManageDataSources Template

Use Case Name	**ManageDataSources**
Super Use Case	NA
Actor	Admin
Brief Description	Database and dataset is managed.
Preconditions	NA.
Post-Conditions	New entries are added in database making it more versatile.

Table 16: MonitorUserActions/Feedback Template

Use Case Name	**MonitorUserActions/Feedback**
Super Use Case	NA
Actor	Admin
Brief Description	Feedback of user is monitored and applied using algorithms.
Preconditions	NA.
Post-Conditions	These ratings can be applied through algorithms to improve recommendation system.

Table 17: GetCustomerProfileandStatistics Template

Use Case Name	**GetCustomerProfileandStatistics**
Super Use Case	NA
Actor	Admin
Brief Description	Admin can check customer profiles and can obtain statistics.
Preconditions	NA
Post-Conditions	Admin can get insight of what user listens to most.

Table 18: AcquireandIndexNewAlbums Template

Use Case Name	**AcquireandIndexNewAlbums.**
Super Use Case	NA
Actor	Admin
Brief Description	Admin can add more albums into database.
Preconditions	NA
Post-Conditions	Database is expanded, and more songs are added to database.

3.1.5 Class Diagram

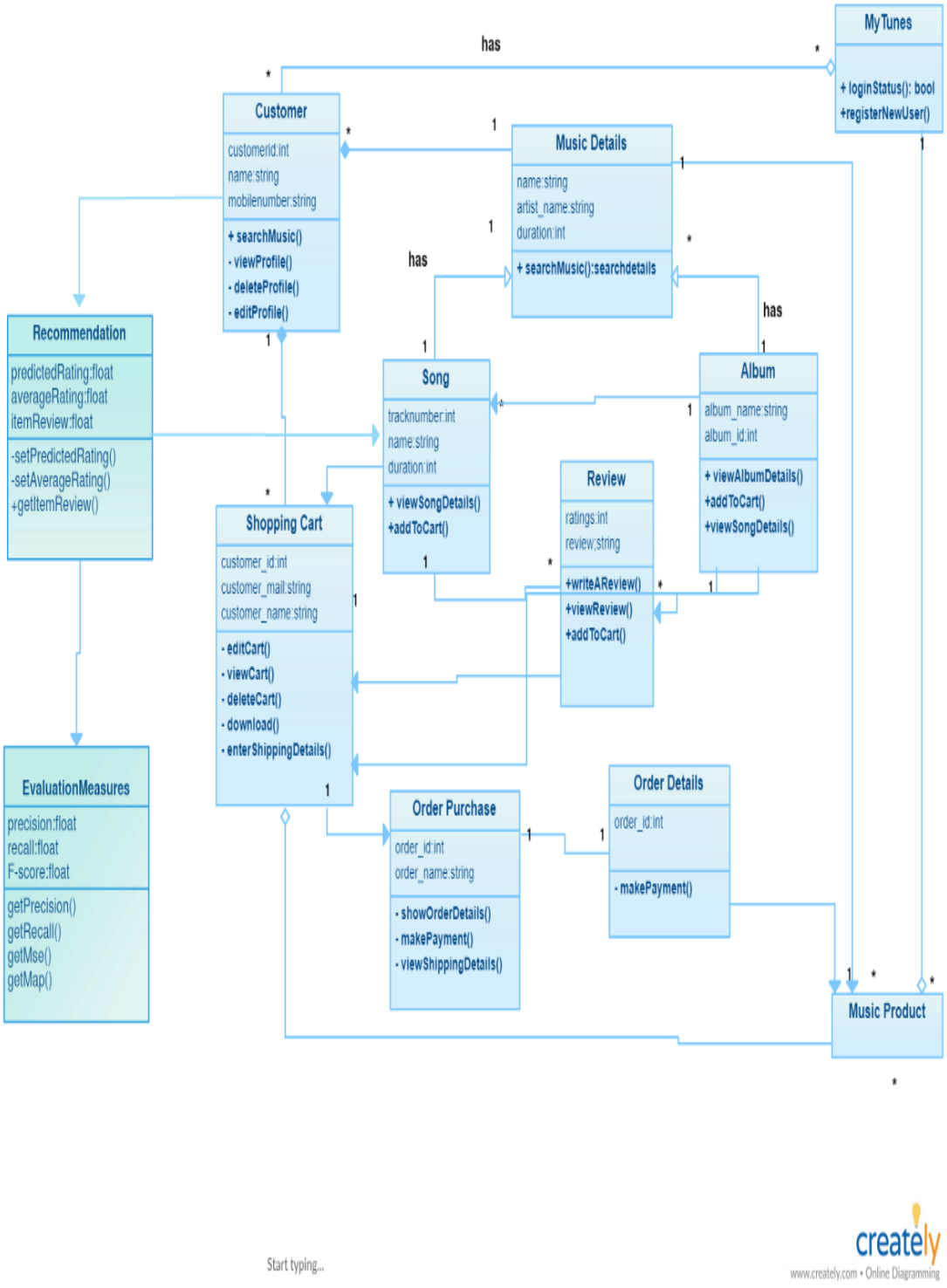

Figure 2: Class Diagram

3.2 Cost Analysis

The development tools that we will use for this project are open source. We will use Python, PHP, JavaScript, MYSQL, MongoDB and Apache Spark for our project. Since all these are open source, we will not incur any cost for the development of this project. In addition to this if we want to make our project live, there will be cost of servers and domain names. Domain names cost around 1000-2000 rupees per year. Servers will cost around 700-800 rupees per year.

3.3 Work Breakdown Structure

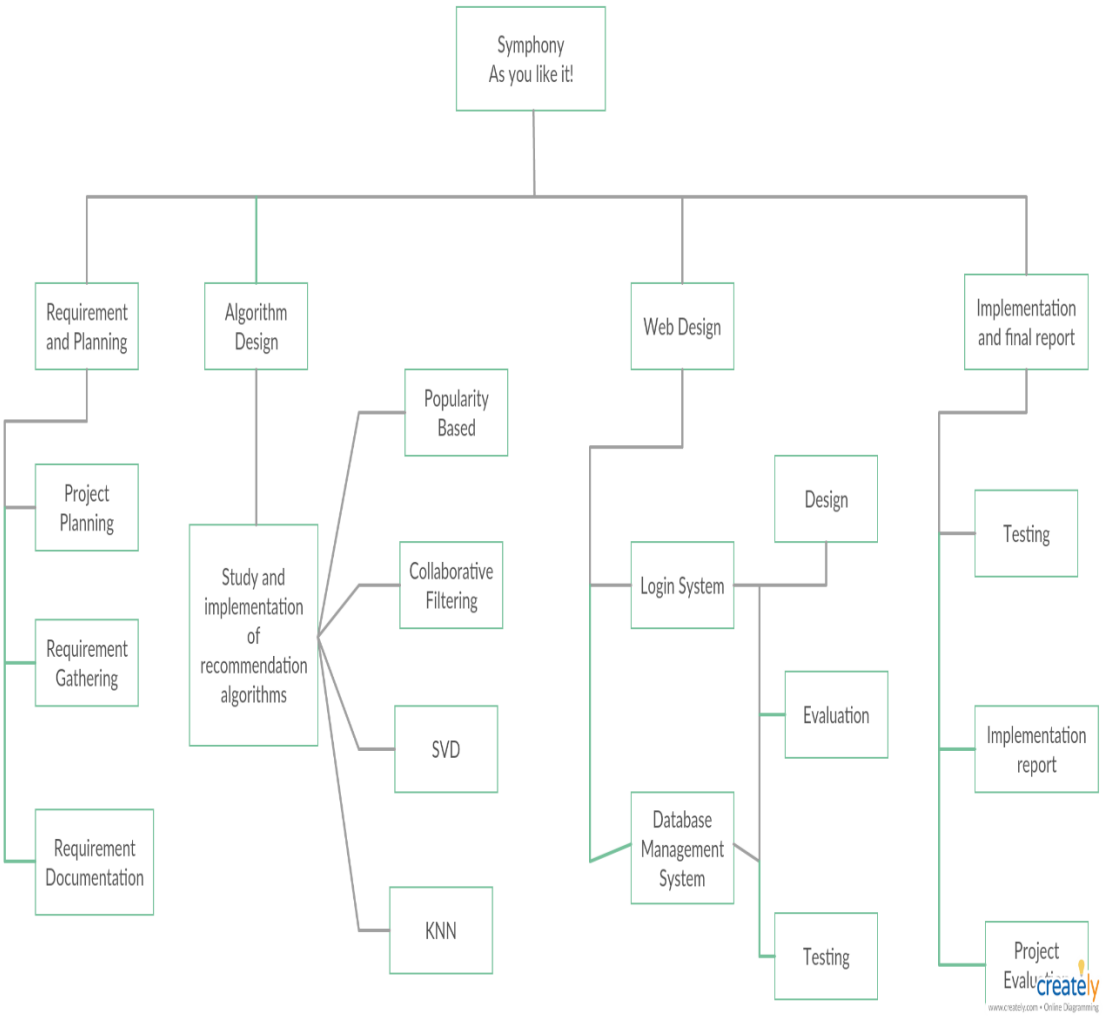

Figure 3: WBS

Work Breakdown Structure using GANTT Chart

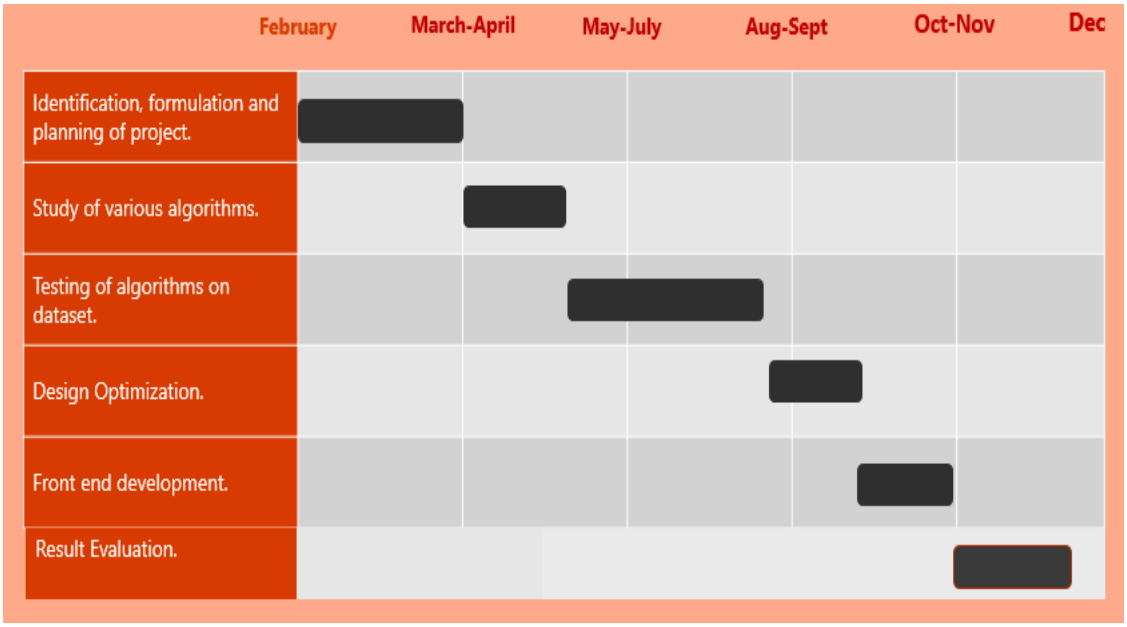

Figure 4: WBS using GANTT chart

4.1 Data Flow Diagram

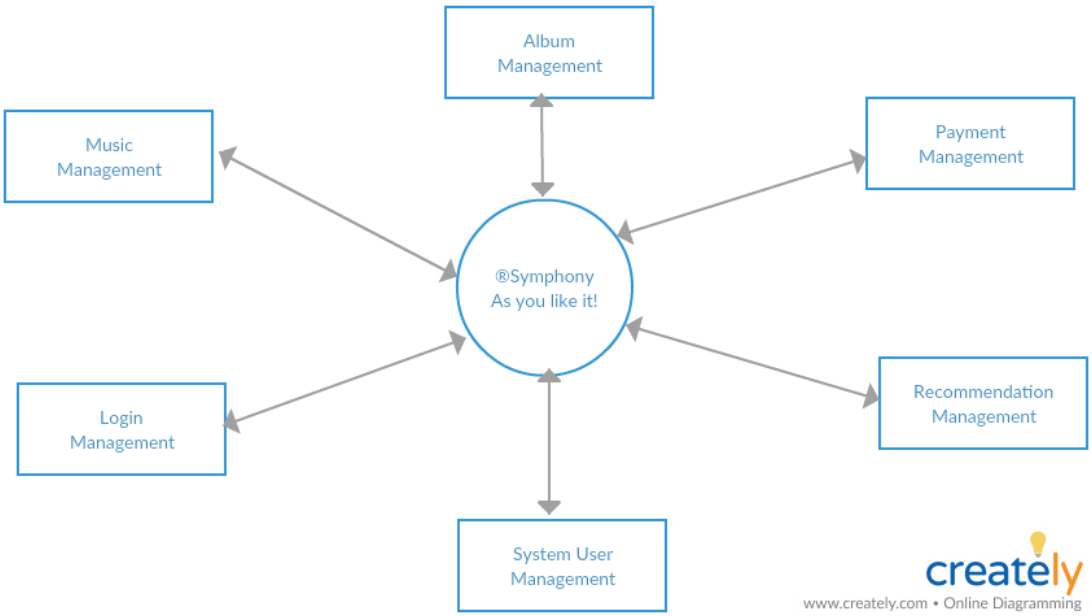

Figure 5: *Level 0*

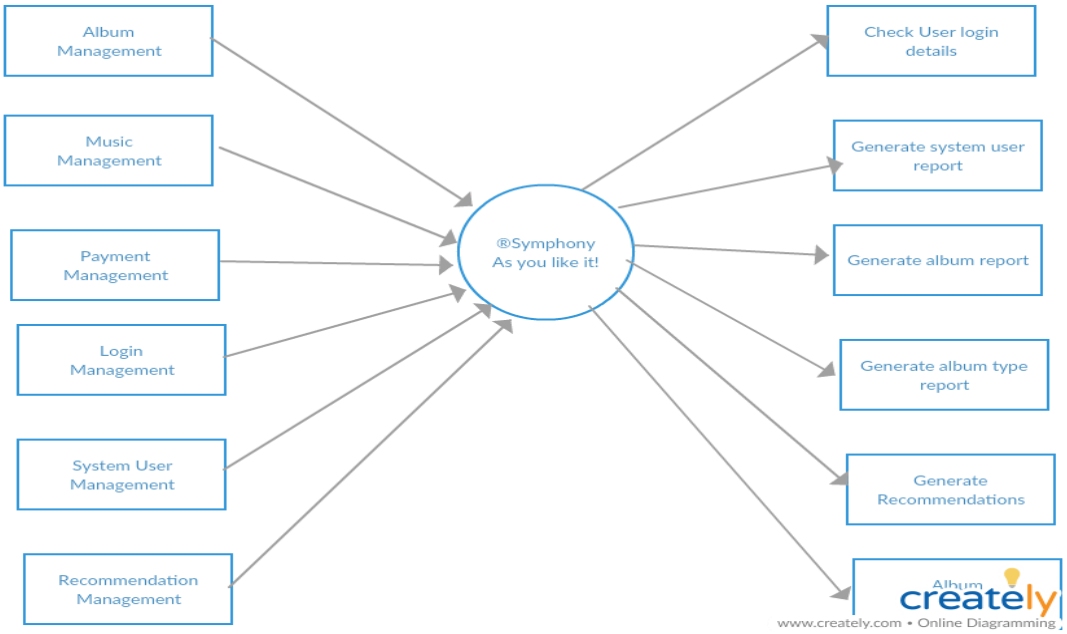

Figure 6: *Level 1*

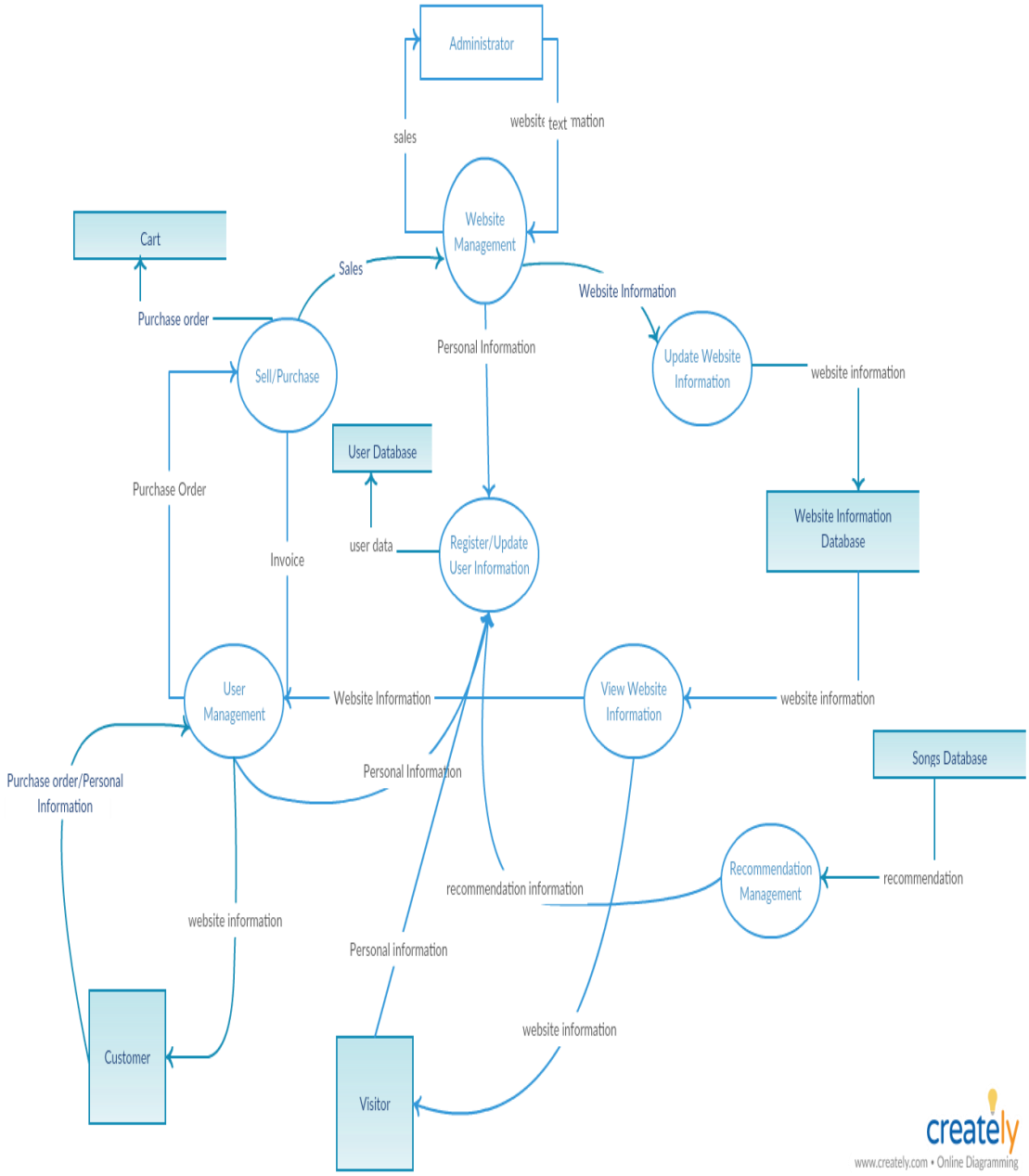

Figure 7: *Level 2*

4.2 Activity Diagram

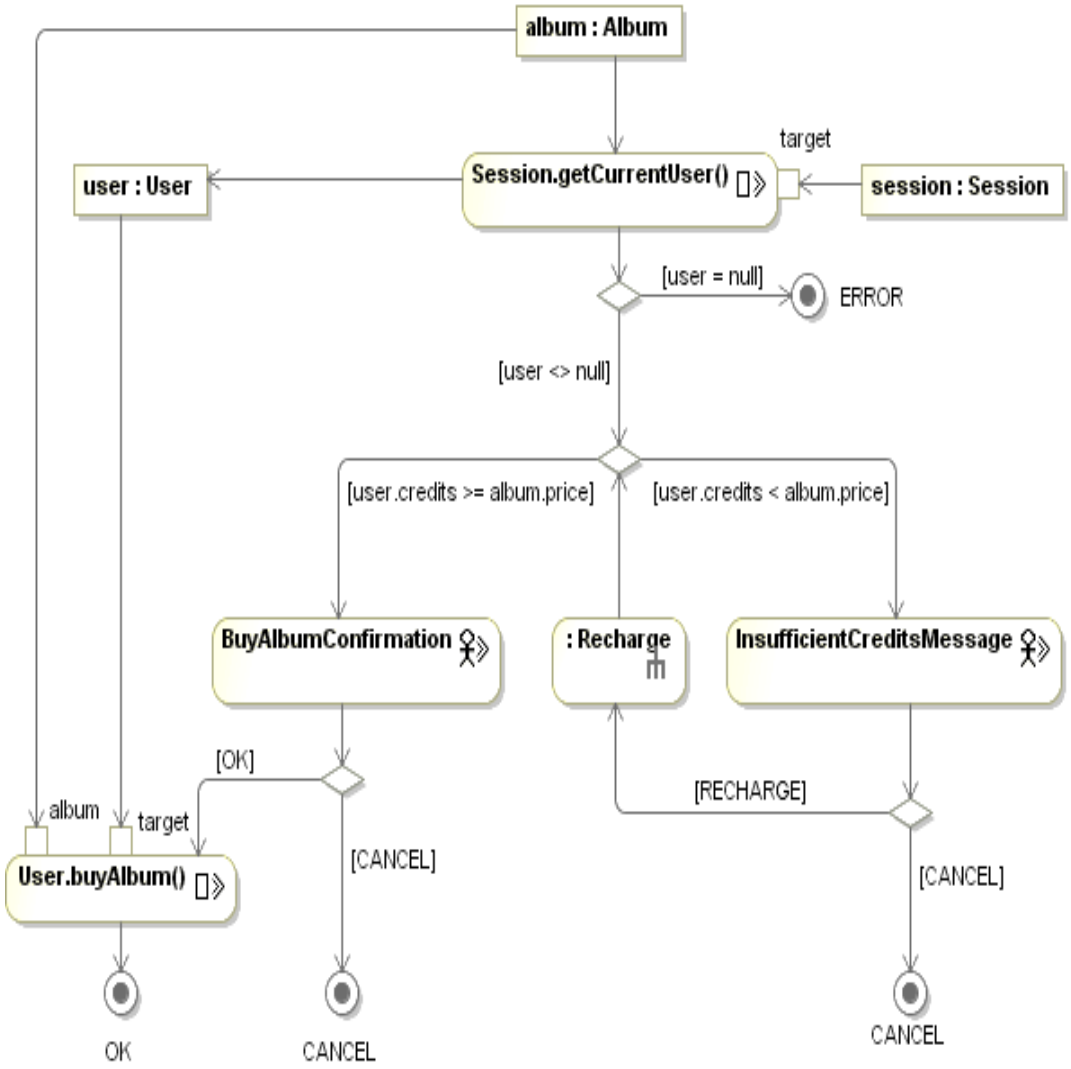

Figure 8: Buy Album

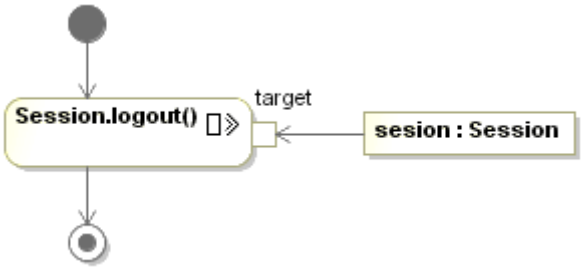

Figure 9: Logout:

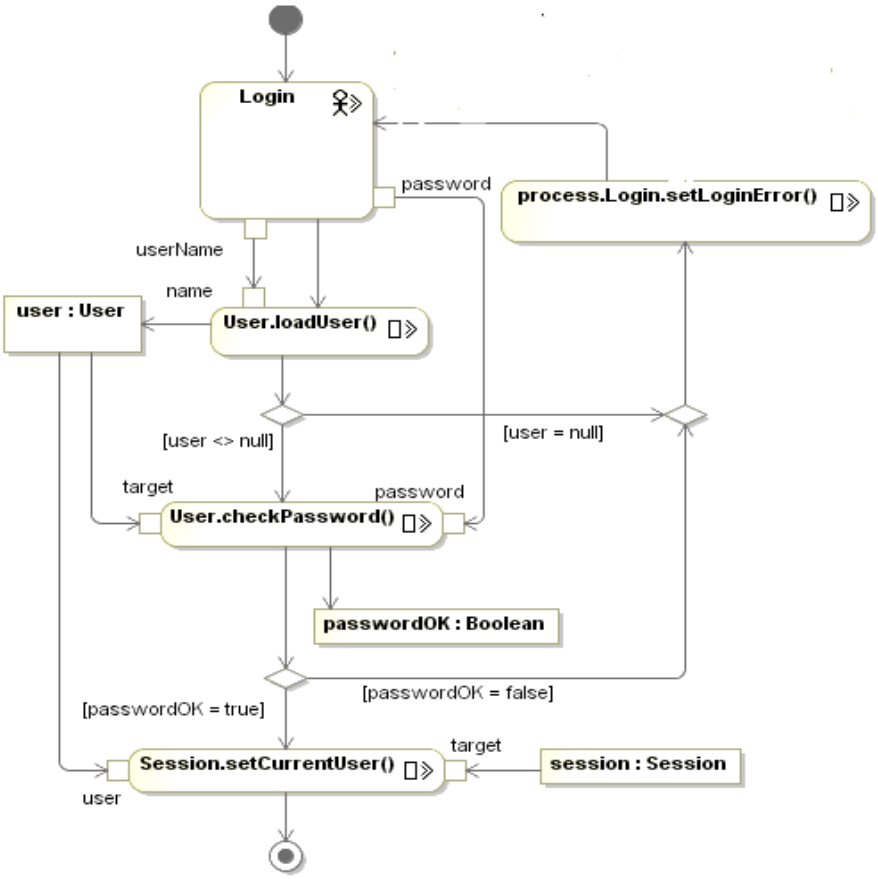

Figure 10: Login

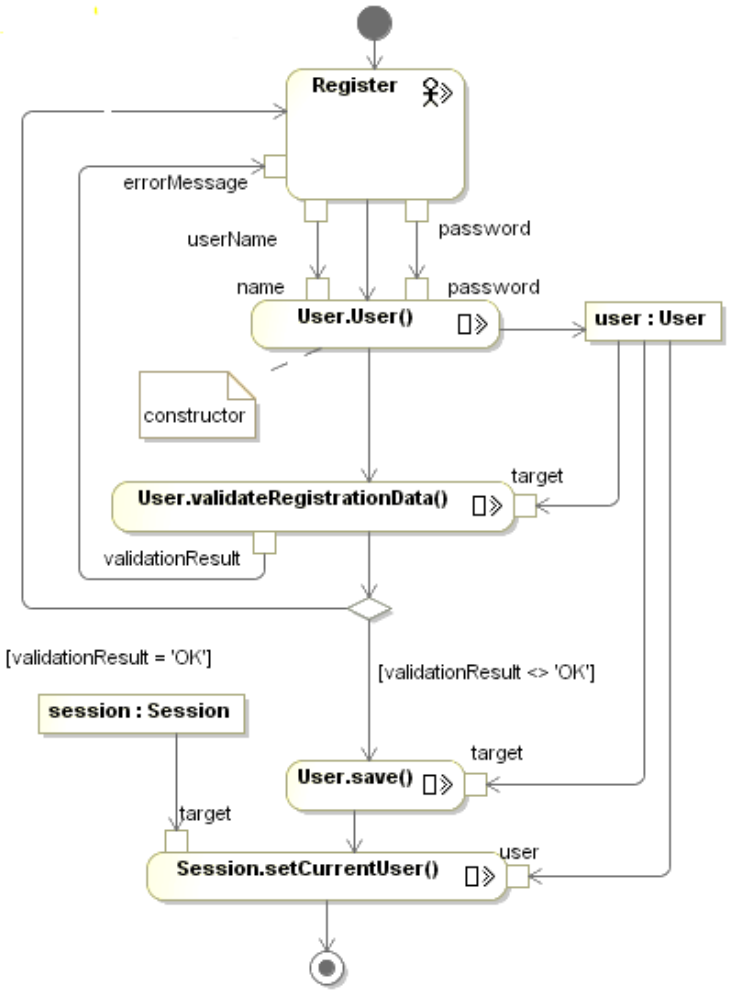

Figure 11: Register:

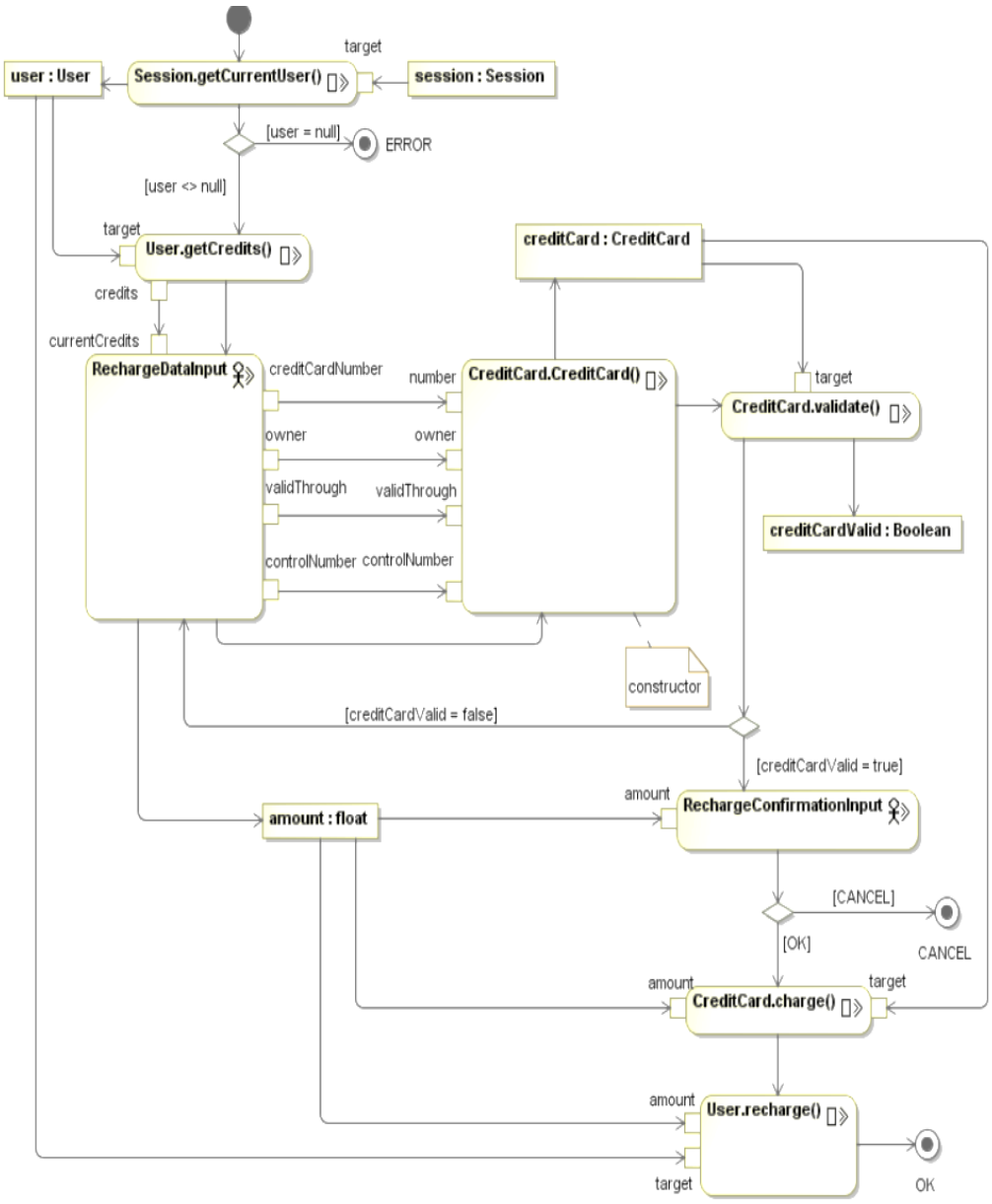

Figure 12: Recharge:

4.3 Sequence diagram:

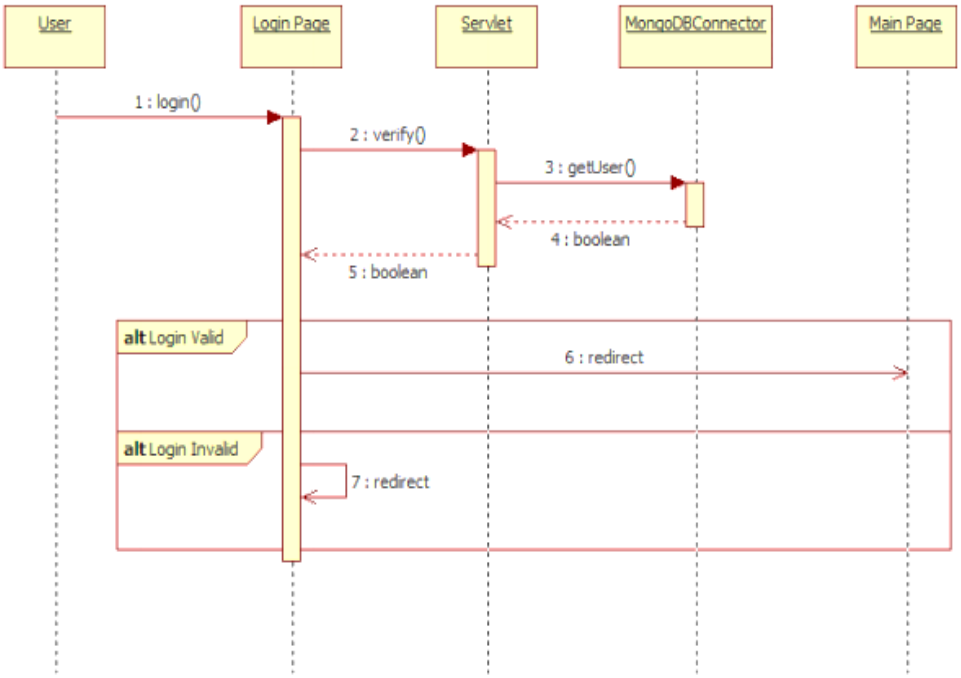

Figure 13: Login Sequence Diagram:

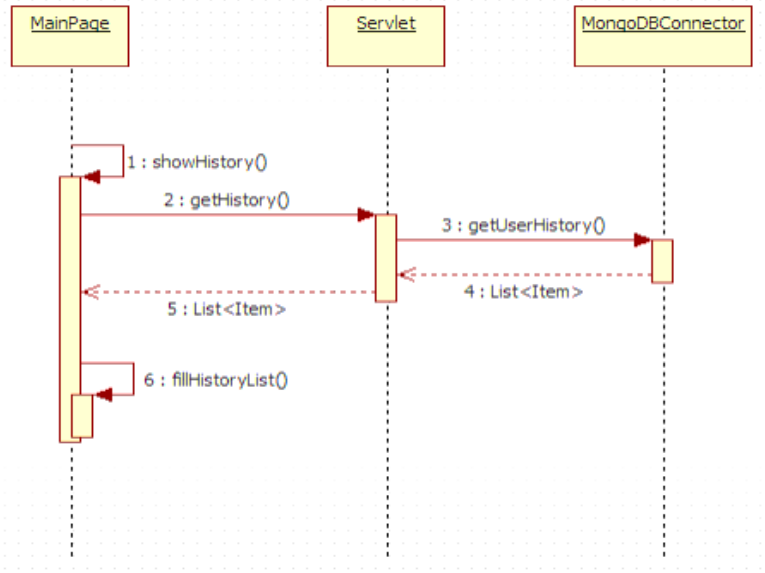

Figure 14: View History Sequence Diagram:

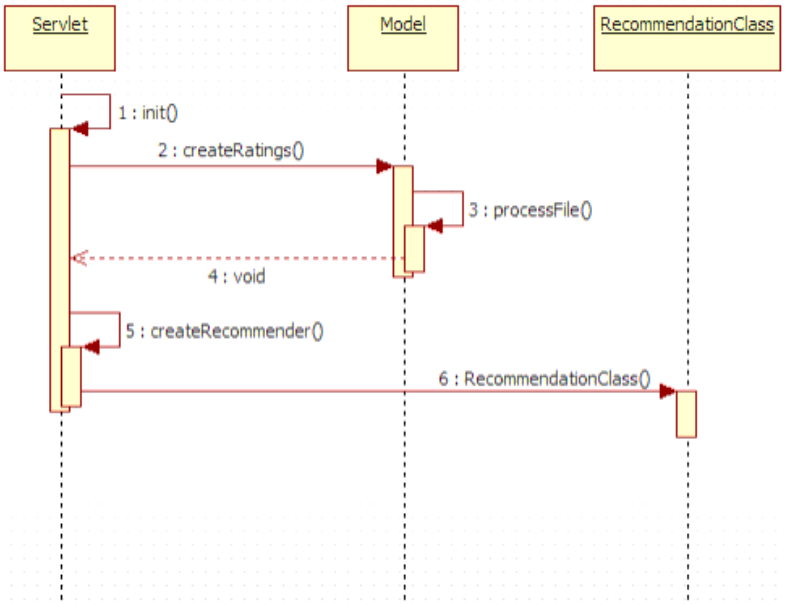

Figure 15: Generate Recommendations Sequence Diagram:

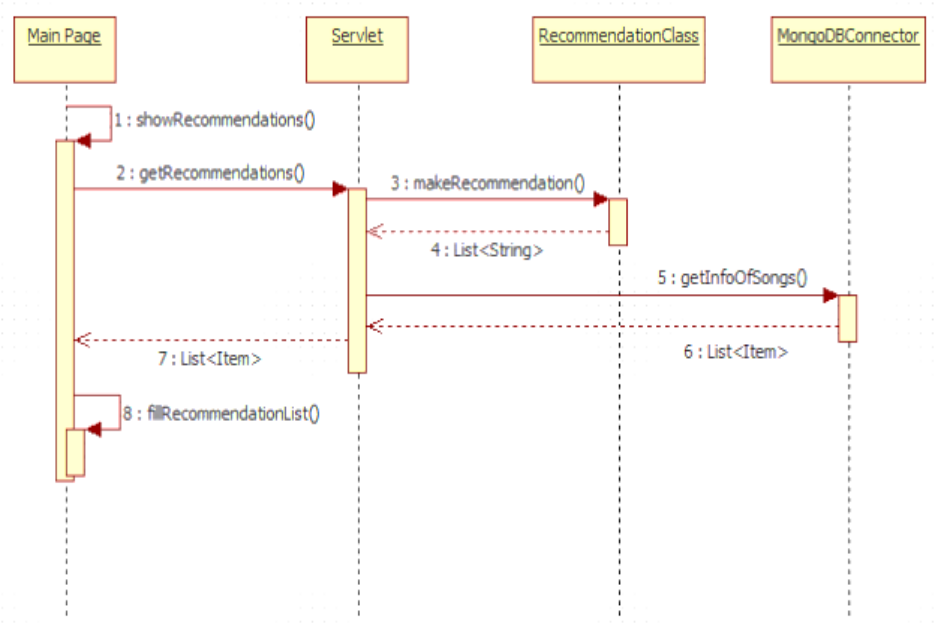

Figure 16: View Recommendations Sequence Diagram:

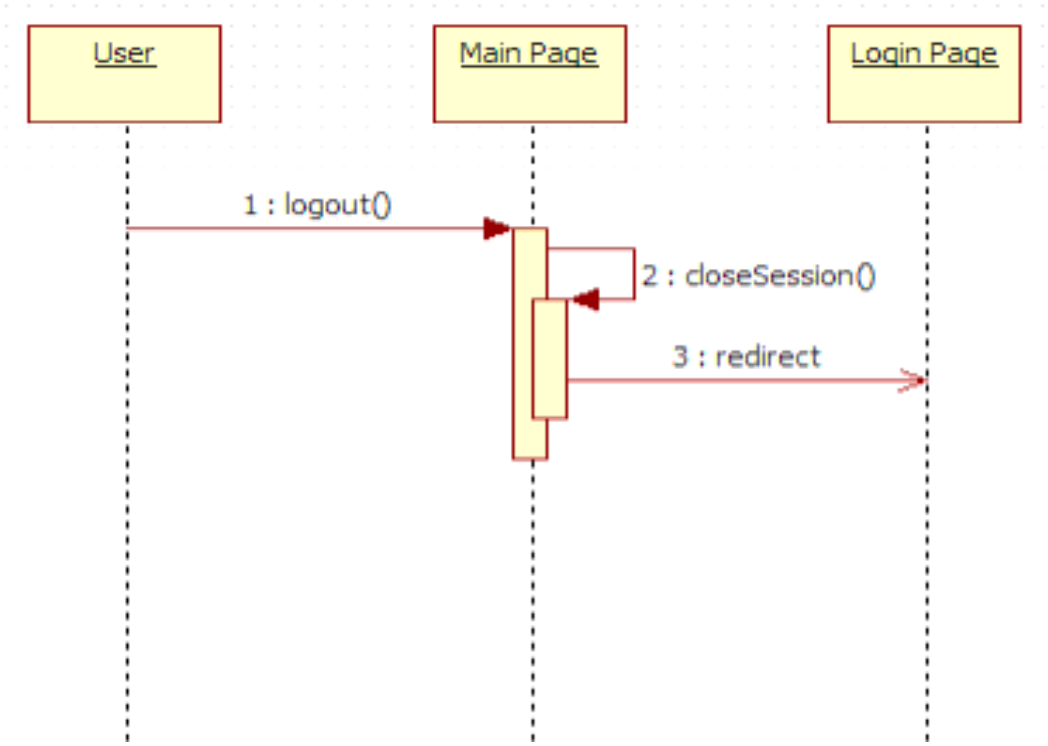

Figure 17: Logout Sequence Diagram:

4.4 Design Model

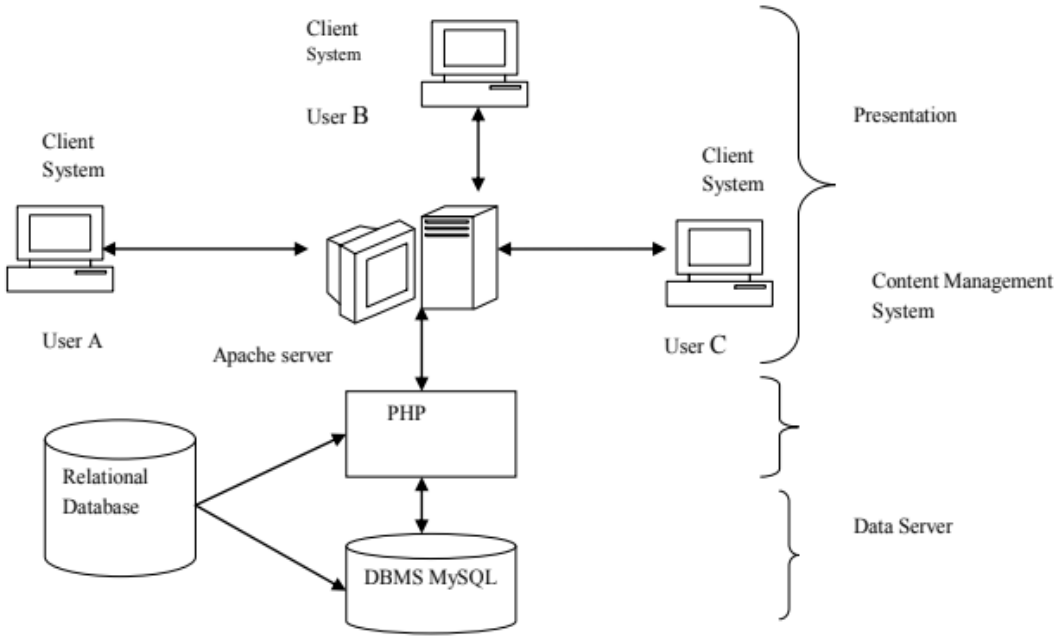

Figure 18: Design Model

4.5 Database design

Table 19: Users

Field	Type
userid	int (25)
first_name	varchar (25)
last_name	varchar (25)
email_address	varchar (25)
username	varchar (25)
mobile_no	varchar (25)
sex	varchar (25)
marital_status	varchar (25)
nationality	varchar (25)
password	varchar (25)

Table 20: Track table

Field	Type
id	bigint (25)
performer_id	int (11)
album_id	int (11)
track_no	smallint (6)
name	varchar (20)
duration	varchar (6)
last_played	varchar (20)
time_played	int (11)
year	varchar (4)

Table 21: Performer Table

Field	**Type**
pid	int (11)
pname	varchar (20)
year	varchar (4)

Table 22: Favorites table

Field	**Type**
id	int (11)
track_id	int (11)
performer_id	int (11)
album_id	int (11)
name	varchar (50)
duration	varchar (6)
last_played	varchar (20)
time_played	int (11)
year	varchar (4)
user_id	int (11)

Table 23: Album table

Field	**Type**
a_id	int (11)
performer_id	int (11)
name	varchar (50)

Table 24: Cart table

Field	Type
ordernumber	int (11)
order_id	int (11)
userid	int (25)
payment	int (8)

4.6 Snapshots

.

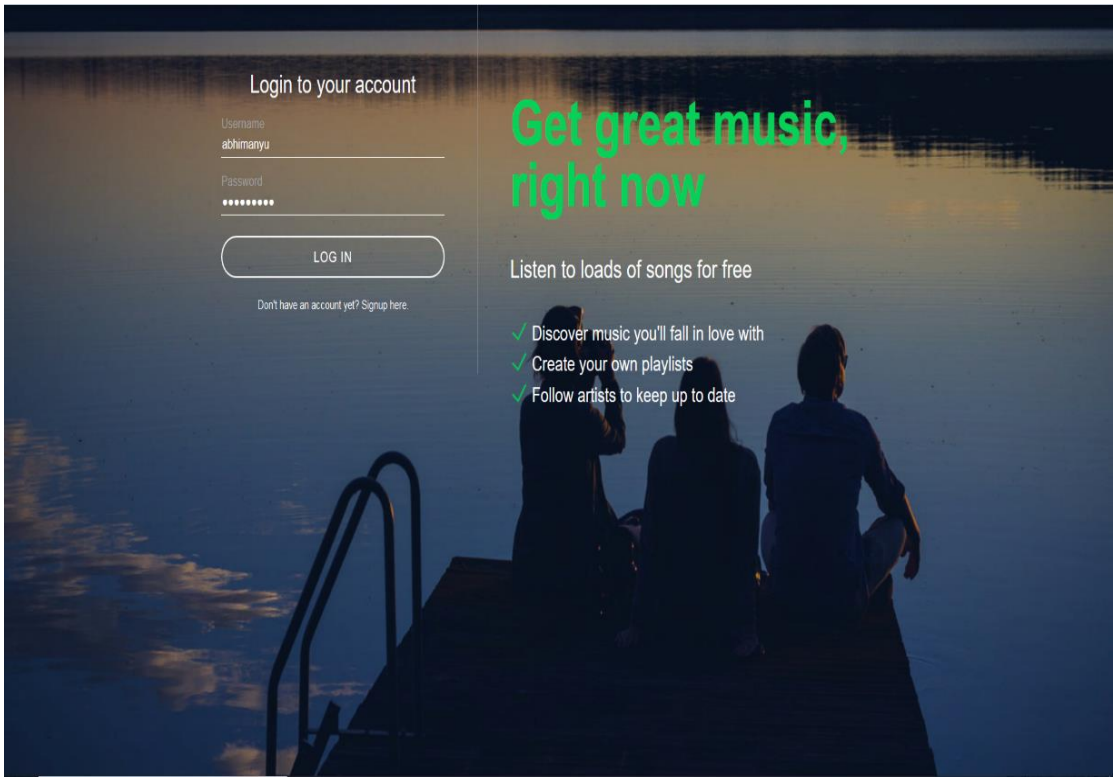

Figure 19: Main Page

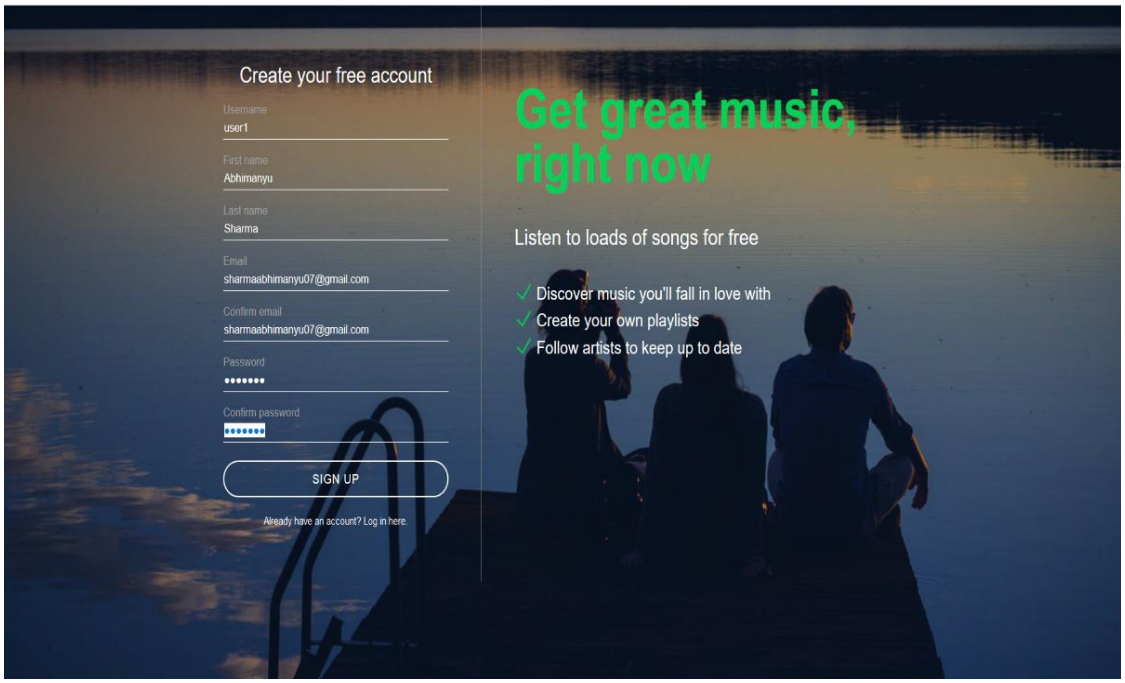

Figure 20: Register Page

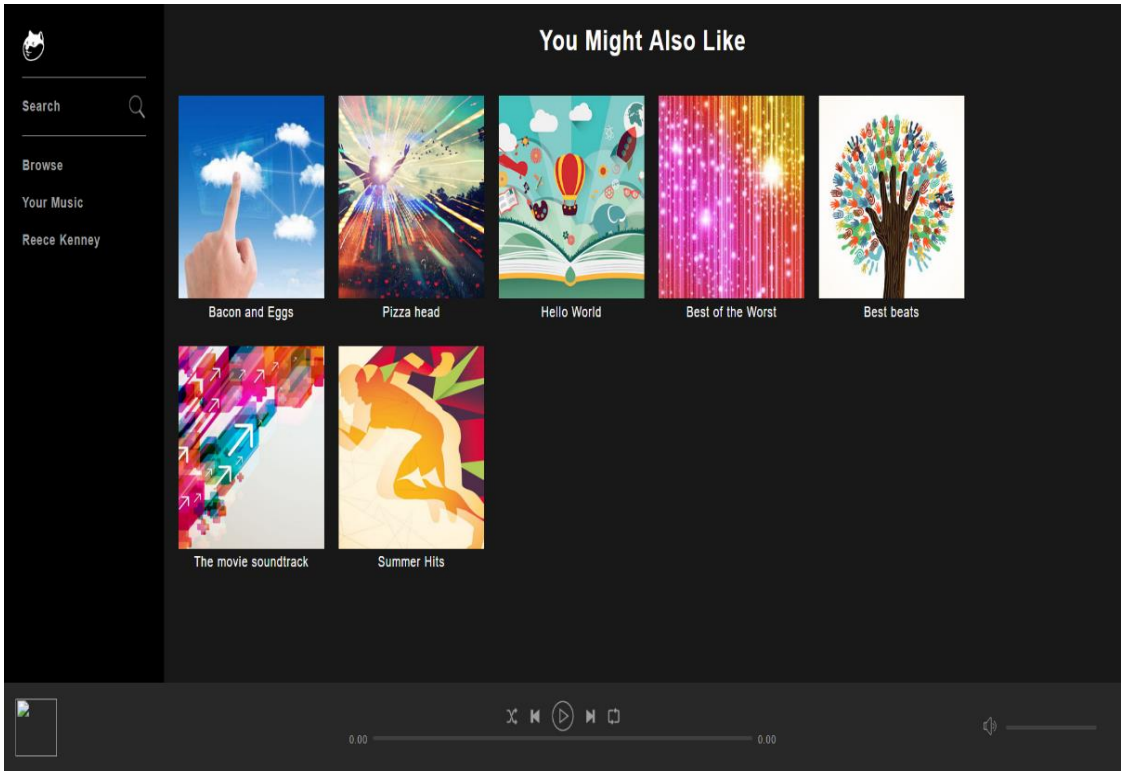

Figure 21: Home Page

5. CONCLUSION AND FUTURE DIRECTIONS

5.1 Work Accomplished

- Implemented popularity-based algorithms for non-personalized music recommendations.
- Implemented the basic front-end user interface.

5.2 Conclusions

This project is great example of recommendation algorithms which are prevalent in every field these days like e-commerce, video streaming websites etc. We aim to implement recommendation feature in the music domain.

5.3 Reflections

- We got to analyze various recommendations algorithms.
- Acquired skills to develop python-based Django Web apps.
- Implementing Web Application with php and JavaScript.
- Grasped the knowledge of Google Cloud Services for fast implementation of recommendation algorithms.
- Become proficient in working as an enthusiastic team

5.4 Future Work Plan

- Implementing collaborative filtering and content-based filtering algorithms for personalized music recommendations**.**
- Implementation of fully functional web application to show the recommendation results to users.
- Integrate both front end and back end using Django.

6. REFERENCES

[1] F. Aiolli. “A preliminary study on a recommender system for the million songs dataset challenge”. ECAI Workshop on Preference Learning: Problems and Application in AI, 2005.

[2] M.D.Ekstrand, J.T Riedl, J.A. Konstanz. “Collaborative filtering recommender systems”. Foundations and Trends in Human-Computer Interaction, vol. 4, no. 2, pp. 81-173, 2011.

[3] H. Karp. “Apple iTunes sees big drop in music sales” The Wall Street Journal. Internet:https://www.wsj.com/articles/itunes-music-sales-down-more-than-13-this-year-1414166672, Oct. 24, 2014 [May 25, 2018].

[4] H. Karp. “Scores of Music Services Stream into Crowded Field” The Wall Street Journal.Internet:https://www.wsj.com/articles/scores-of-music-services-into-crowded-field-231578667, Dec. 24, 2013 [May 25, 2018].

[5] Y.Koren. "Recommender system utilizing collaborative filtering combining explicit and implicit feedback with both neighborhood and latent factor models." AT&T Corp, AT&T Intellectual Property II LP, July 30, 2008.

[6] B. McFee, T. BertinMahieux, D.P. Ellis, G.R. Lanckriet. “The million-song dataset challenge”. The 21st international conference companion on World Wide Web (pp.909916).ACM, April 2012.

[7] Y. Ding, C. Liu. “Exploring drawbacks in music recommender systems–the Spotify case”.Internet:https://www.divaportal.org/smash/get/diva2:896794/FULLTEXT01.pdf, 2015 [May 26, 2018].

[8] M.Sutherland. Spotify improves discovery functions”. Internet: http://www.musicweek.com/publishing/read/spotify-improves-discovery-function /072645, June 14, 2014 [May 27, 2018].

[9] M.Sutherland. “Spotify press: Information about Spotify”. Internet: http://www.musicweek.com/publishing/read/spotify-press-information-about-spotify/182375 , April 06, 2014 [May 27, 2018].

[10] R.V.Meteren, M.V. Someren. “Using Content-Based Filtering for Recommendation”. NetlinQ Group, Amsterdam, 2000.