ONLINE MUSIC STORE (OMS)

MOHD HAFIZ BIN ROSLAN (CB11022)

DR BALSAM ABDUL JABBAR MUSTAFA

TECHNICAL REPORT SUBMITTED TO IN FULLFILLMENT OF THE DEGREE OF COMPUTER SCIENCE IN SOFTWARE ENGINEERING

FACULTY OF COMPUTER SYSTEMS AND SOFTWARE ENGINEERING

2013/2014

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere thanks to my project supervisor, Dr. Balsam Abdul Jabbar Mustafa for her insightful comments, outstanding advice, and exceptional guidance. I would also like to express my heartiest appreciation for his patience in spending a lot of time to guide me in my project and provide a lot of valuable and practical suggestions during this period.

Also, I would like to express my appreciation to my friends for sharing their valuable idea and knowledge with me, in order to assist myself to succeed the project.

Moreover, I am grateful to express my thanks to my family for their understanding, supports, and encouragement towards the completion of this project.

ABSTRACT

In the music industry, to find a comprehensive system that meets all requirements are difficult. There are not too many of them in the world that produced a system like that. This project is done to overcome the lack of features in the music website. Online Music Store (OMS) is a web based system that design to build a well-defined website that acts as a music database. This system includes three categories of users which are unregistered user, users that classified registered user and musician, and admin. The methodology uses for the website development is Rapid Application Development (RAD). In this project, the enhancement of features increases the user's usability and creates a network among music enthusiastic.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	TITLE
OMS	Online Music Store/Online Music System
The WKND	The Wknd Website
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
I/O	Input and Output
PHP	Hypertext Preprocessor
PSM	Projek Sarjana Muda
SRS	Software Requirements Specifications
SDD	Software Design Description

PART 1: INTRODUCTION

1.1 Background

After year 1997 where Capitol Records announce their intention to offer the single album from the new Duran Duran album in downloadable form on the Internet, many companies saw it as a threat to their sales. However, in 2003, Apple Computer launches the most successful online music store to date. In its first year, Apple sells 70 million songs at $0.99 per song, creating nearly $70 million in legal Internet music sales. This raises awareness among the “other side” that Internet also can be an effective sales medium in addition to retail store.

Today, we can found as many as music website in the Internet that evolved alongside others internet business industry. But the question is which website is comprehensive enough to achieve higher usability, performance, maintainability, and survivability?

Malaysia`s music industry also thrive concurrently with growth of the Internet technologies. Many musicians express their ideas in the website columns and also sells theirs work here. The availability of the Internet also gives the musician’s potential to sell-out their works around the world and connects with others. With this in mind I tried to develop a system that can give benefits to all musicians.

As music industries become more competitive, and are required to increase their standards, I believe the Online Music Store will become very popular choice for a music enthusiasm. I strongly believe this will prove to be a very worthwhile and profitable investment in the future of music.

1.2 Main Aims and Objectives

i) To create music website for music enthusiasms aims to be as forum for all musicians and extend their knowledge with social network features in addition to promoting musicians works.

ii) The system will act as a database of music in Malaysia, organized around local and international musicians.

iii) To enhance the functionality of the typical music website with features such as social network, streaming, download, rating system, library system, profiling system and shopping cart system.

1.3 Problem Statement

There are many music websites can be found on the Internet. Even in Malaysia, there`s a lot of music websites either for commercial or non-commercial purposes. These websites or systems were integrated by all people around the world. Some of them had little knowledge about the website development and some have enough knowledge to integrate it. With these kind of variety developers involved in the website development, many problems that occurs involving situation such as reliability, usability, performance, functionality and security had been found.

Typical problems that always found in the previous music websites are lack of features. Typically, music website should contains the details about musician, their materials such as albums and artworks if available, a playlist for the streaming of their music or downloadable area, comment box for the listeners to give comments, up-to-date news about the musician, musician and albums rating, purchases section, and all of the reasonable music features that should be there. From what I`ve found, these websites not comprehensive enough to fulfil the meaning of music website.

Some problems were found in others music website in Malaysia, like The Wknd (http://www.the-wknd.com/) website such as poof features integrated. Users can browse for the music they like but the resources are limited. For example the unavailability of the user whole album.

With the growing of the blog technology, many musicians involved themselves and created their own blogs to promote their works. Here we found the second problem regarding blog-type website which as we know stored information only for single musician/single musician group per website. In this website we can find all information about the musician but lack of usability and wasted time. This situation can be a problem for users to navigate. They need to jump into one site to another site for viewing process instead stay at one site.

Website designs are one of the important parts in the website development. Well-designed website will attract more users to visit and use the system. From what I've found, some of the previous music websites are poor in designed. We can start with background which contains unsuitable colours. Music website should contain minimal and light colour in background because of its large contents. User should be able to read the contents and applied it. As we know, music website had many contents that should be organized properly. For example the playlist should be in appropriate size and located properly without disturb others contents. Yes, contents management also the typical problems that can be found at the previous system. Since design topics are huge, conclusion can be made that design is important to the website development and this problems should be avoided.

1.4 Review of Previous Work

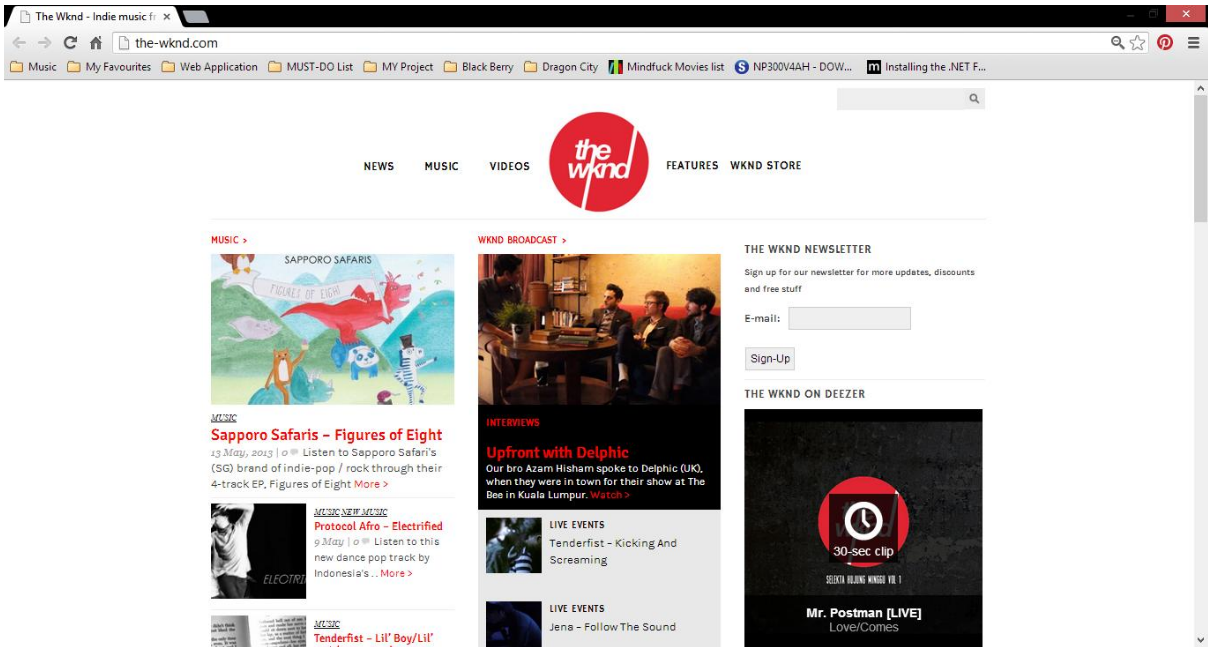
Currently, there is no related previous work that had relation with the same scope like this project. This project originally derived from my own idea. This idea started with my deep interest in music and would like to create my own music website. In the Internet, we can find variety type of music website. Those websites sometimes included the only up-to-date news and events features; others just have details and downloadable songs features. They are too many varieties of types to be classified here. With this idea, I`ve come up with my own idea to conclude all of these features into one website.

This type of music website that includes social network style is already implemented in the others country like U.S. As far as I`m doing my research, I don't see any occurrence in this type of website in our country (Malaysia). With this, I can’t recommend review about previous work regarding this type of website.

1.5 Review of Current System and Its Limitation

Currently, there is no system like this type of system which is music website with a social network features but I`ve found a similar system that a lot like this system. The system I meant about is the music website that was established in Malaysia. This website not implemented with the social network type but the other contents and features are lot likes this system (my system). With this I`ve think it`s worth to do review about this The Wknd website.

The Wknd (http://the-wknd.com/)



THE WKND NEWSLETTER

Sign up for our newsletter for more updates, discounts and free stuff

MUSIC

Sapporo Safaris – Figures of Eight

13 May, 2013 | 0 Listen to Sapporo Safari's (SG) brand of indie-pop / rock through their 4-track EP, Figures of Eight More >

MUSIC NEW MUSIC

Protocol Afro – Electrified

9 May | 0 Listen to this new dance pop track by Indonesia's . . More >

MUSIC

Tenderfist – Lil' Boy/Lil'

Upfront with Delphic

Our bro Azam Hisham spoke to Delphic (UK), when they were in town for their show at The Bee in Kuala Lumpur. Watch >

Tenderfist – Kicking And Screaming

Jena – Follow The Sound

THE WKND ON DEEZER

Figure 1.5-1: The Wknd

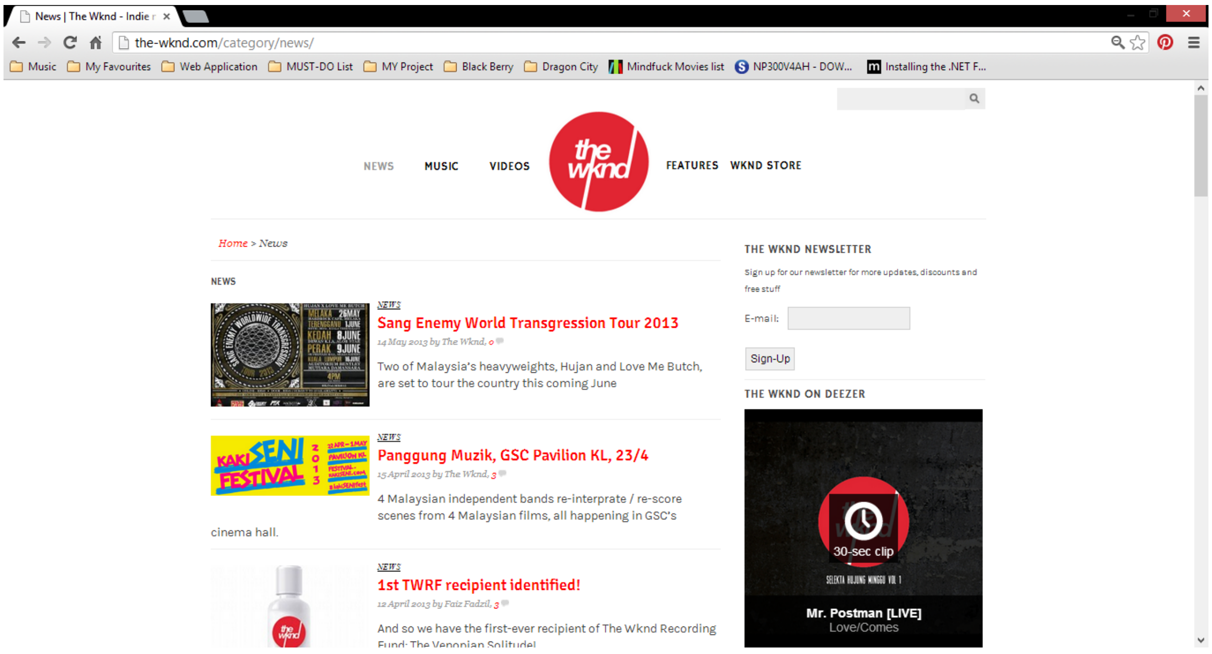
The Wknd is the music website that acts as a promoting platform for the musician to introduce their works and shared their ideas. This website contains video performances, music videos, mp3s, new music, events and the latest in Malaysian and South East Asian music industry. This website stated as the high ranking music website popularity nowadays. This website is design by Bright Light at Midnight (BLAM) which is an independent design studio that established to provide design and visual communications solutions to clients from a wide range of industries – from business and corporate entities, to social networking communities, to entertainment and arts groups.

I ought to say this website is well-designed and have a capability to send message to viewers but if this website enhance more of its functionality it would be perfect. This website can be categorised as a blog-type website. Like practically everything else on the Web, blogs are easy to start and hard to maintain. Writing coherently is one of the most difficult and time-consuming tasks for a human being to undertake. So, far from blogs being a cheap strategy, they are a very expensive one, in that they eat up time. As a result, many blogs are not updated, thus damaging rather than enhancing the reputation of the organization.

1.5.1 Features provided by The Wknd

There are several features that provided in this website. Each of them played uniquely roles to this website. They can be break down into 5 categories.

i) News Page



NEWS

NEWS
Sang Enemy World Transgression Tour 2013
14 May 2013 by The Wknd, 0
Two of Malaysia's heavyweights, Hujan and Love Me Butch, are set to tour the country this coming June

NEWS
Panggung Muzik, GSC Pavilion KL, 23/4
15 April 2013 by The Wknd, 3
4 Malaysian independent bands re-interprate / re-score scenes from 4 Malaysian films, all happening in GSC's cinema hall.

NEWS
1st TWRF recipient identified!
12 April 2013 by Faiz Fadzil, 3
And so we have the first-ever recipient of The Wknd Recording Fund: The Venonian Solitude!

THE WKND NEWSLETTER
Sign up for our newsletter for more updates, discounts and free stuff

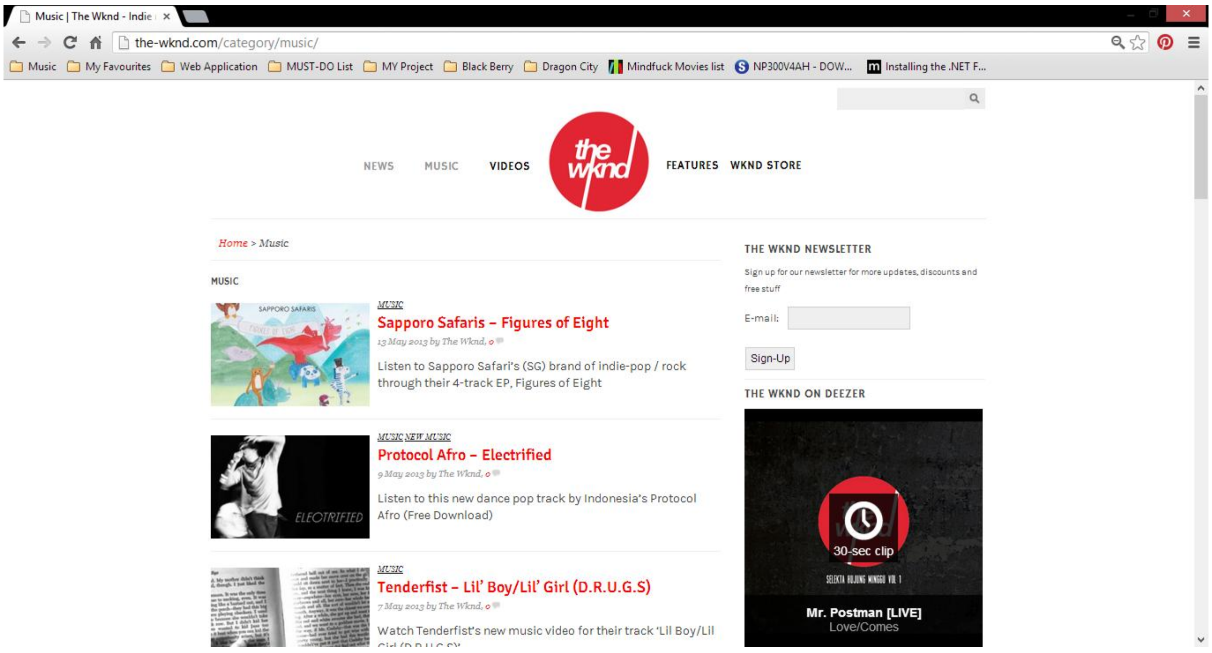
THE WKND ON DEEZER
Mr. Postman [LIVE]
Love/Comes

Figure 1.5.1-1: News Page

“News” is the first categories that had three sub categories under its. “News” is a front page that stored information such as news & update, events and photo gallery.

- News & Update – Stored the information about music industry upcoming, on-going, past news and update. In addition, in this section also had others information that unclassified such as reviews and interviews of the musicians and musician groups.
- Events – Stored the information about the upcoming, on-going and past events that happens in Malaysia.
- Photo Gallery – Stored the information about photo that taken on the selected events that occurs around Malaysia.

ii) Music Page



Music | The Wknd - Indie

the-wknd.com/category/music/

Music | My Favourites | Web Application | MUST-DO List | MY Project | Black Berry | Dragon City | Mindfuck Movies list | NP300V4AH - DOW... | Installing the .NET F...

NEWS MUSIC VIDEOS FEATURES WKND STORE

Home > Music

MUSIC

MUSIC

Sapporo Safaris – Figures of Eight

13 May 2013 by The Wknd, 0

Listen to Sapporo Safari’s (SG) brand of indie-pop / rock through their 4-track EP, Figures of Eight

MUSIC NEW MUSIC

Protocol Afro – Electrified

9 May 2013 by The Wknd, 0

Listen to this new dance pop track by Indonesia’s Protocol Afro (Free Download)

MUSIC

Tenderfist – Lil’ Boy/Lil’ Girl (D.R.U.G.S)

7 May 2013 by The Wknd, 0

Watch Tenderfist’s new music video for their track ‘Lil Boy/Lil

THE WKND NEWSLETTER

Sign up for our newsletter for more updates, discounts and free stuff

E-mail:

Sign-Up

THE WKND ON DEEZER

30-sec clip

Mr. Postman [LIVE]

Love/Comes

Figure 1.5.1-2: Music Page

“Music” is the second categories that had two sub categories under its. “Music” is a front page that stored information about the songs by the musicians. In this section, certain songs can be streaming and download as well.

- New Music – Stored the information about the up-to-date songs that available on the industry.

- #NOWPLAYING – Stored the information about the selected/favourite song by the selected/favourite musician. Mostly the songs that stored here are from the outside country (Malaysia) that gain achievement in music industry and needed to expose more to the current users on this website.

iii) Videos Page

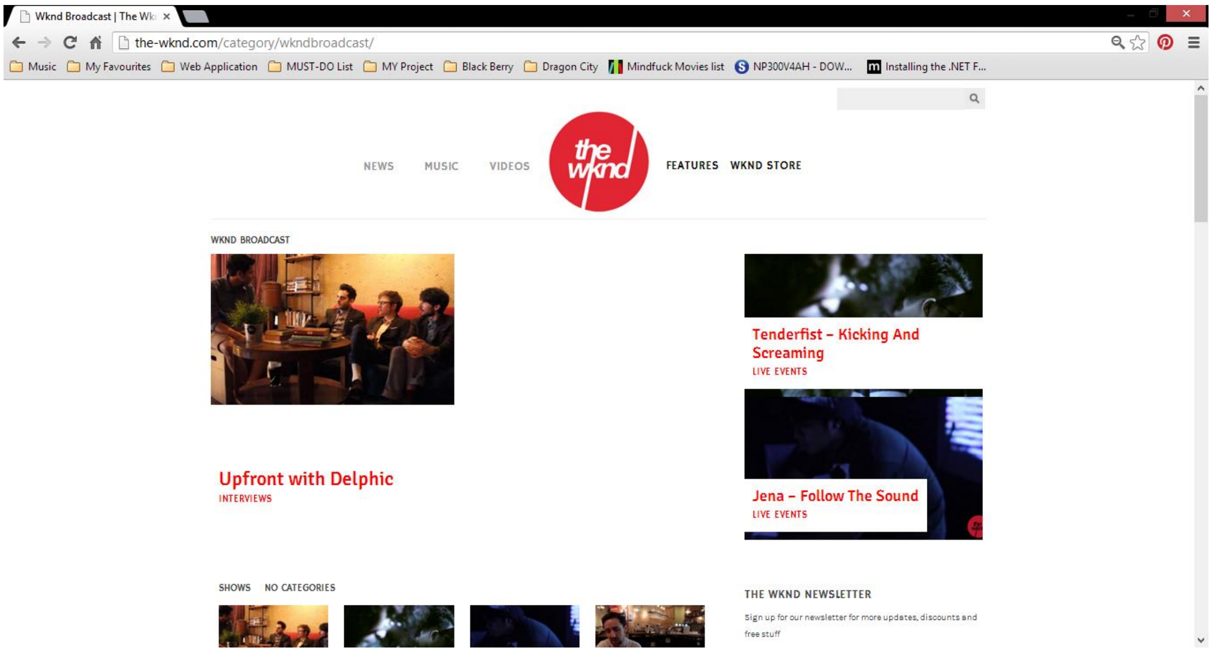
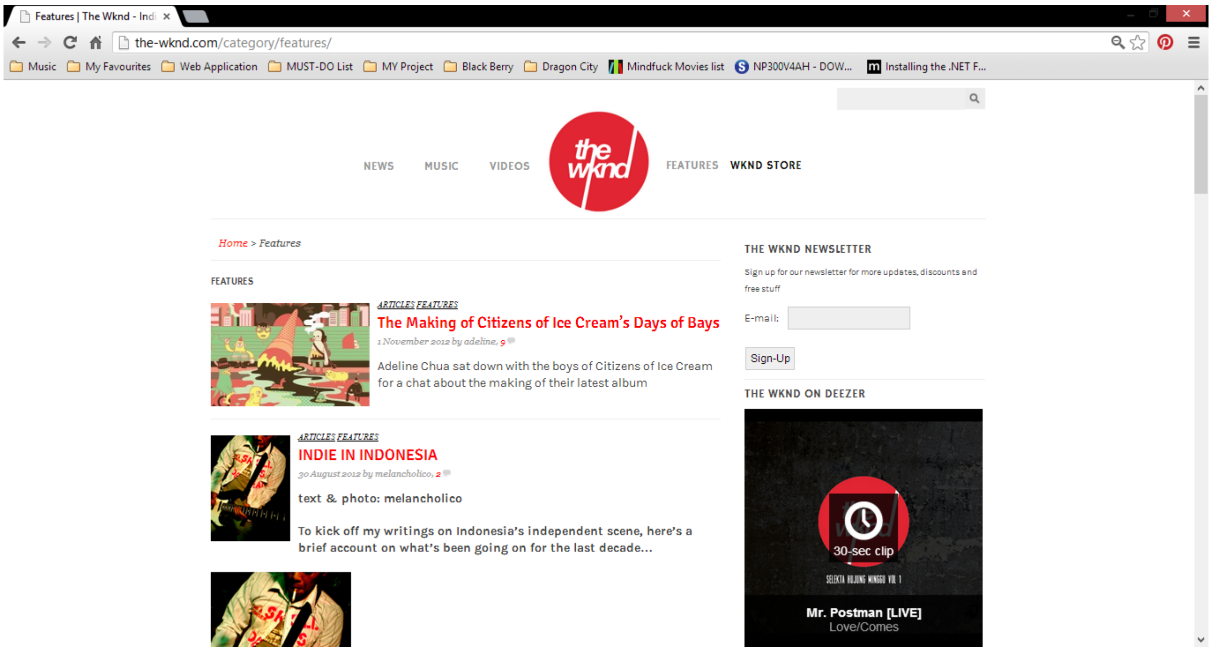


Figure 1.5.1-3: Videos Page

“Videos” is the third categories that had six sub categories under its. Same with others features; “Video” is a front page that stored information about the video matters. Mainly information that store in this features are live video that recorded in the selected events, interviews, and music videos.

iv) Features Page



ARTICLES FEATURES
The Making of Citizens of Ice Cream's Days of Bays
1 November 2012 by adeline, 9
Adeline Chua sat down with the boys of Citizens of Ice Cream for a chat about the making of their latest album

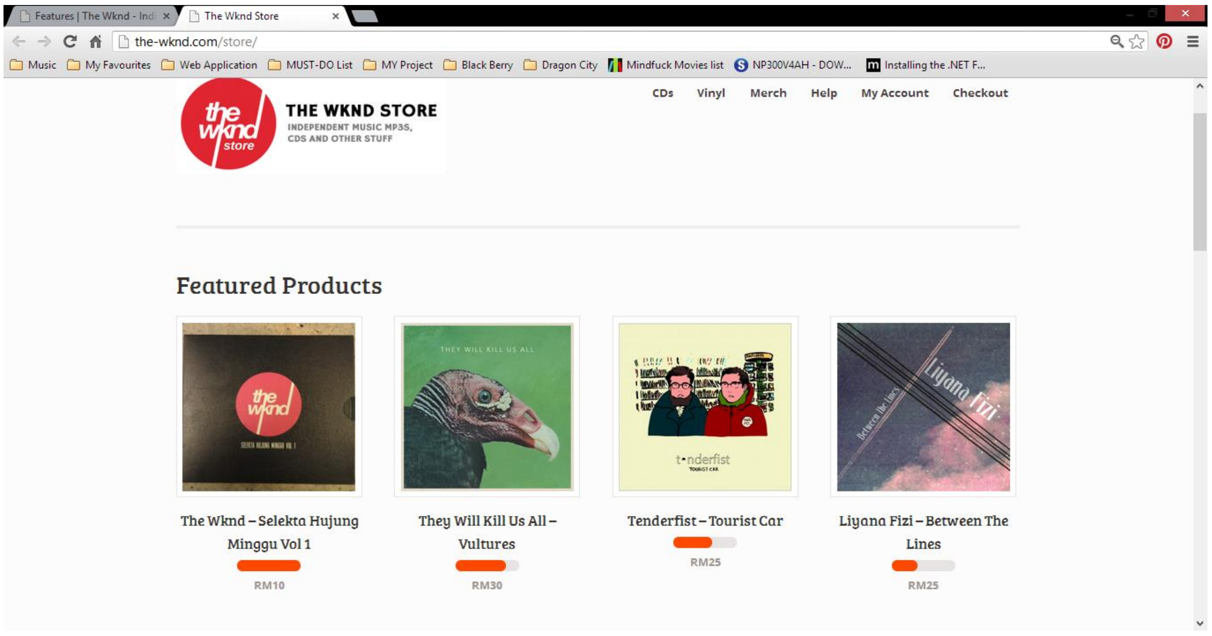
ARTICLES FEATURES
INDIE IN INDONESIA
30 August 2012 by melancholico, 2
text & photo: melancholico
To kick off my writings on Indonesia's independent scene, here's a brief account on what's been going on for the last decade...

Sign up for our newsletter for more updates, discounts and free stuff

Figure 1.5.1-4: Features Page

"Features" is the fourth categories that provided in this website. It contains the information such as articles and musician-talk. "Features" here can be categorized as topic that unclassified as others features that provided by this website.

v) WKND Store Page



Featured Products

The Wknd – Selekta Hujung Minggu Vol 1 RM10

They Will Kill Us All – Vultures RM30

Tenderfist – Tourist Car RM25

Liyana Fizi – Between The Lines RM25

Figure 1.5.1-5: Wknd Store Page

“WKND Store” is the page where business transaction is occurs. This page is about the website store that contains information such as albums and materials to be sold. User needed an account to purchase the materials here. Add to cart systems had been implemented here. Purchasing can be done via bank transfer, PayPal account and credit card transfer.

1.5.2 Limitation of The Wknd

Limitation described by the dictionary is restrictive weakness or lack of capacity. We as human can’t avoid this limitation. There are several points that found in this website regarding its limitation.

System Function

- The first limitation is about the playlist for the songs. In this website, the playlist is embedded is from the Soundcloud plugin. Soundcloud is an online audio distribution platform that allows collaboration, promotion and distribution of audio recordings by users. Limitation issues here are about the performance. Streaming for the songs will lead into poor performance. Internet connection will divided since neither The Wknd website nor Soundcloud used Internet.
- The Wknd website is using a blog type structured so several disadvantages can be recognized with this situation.
- The third limitation of this website can be found with the lack of contents and important features. For example, it contains no detail about musician and users require to look for this information in other site if they intend to do. This information is really important to development for music website.
- The fourth limitation of this website is the poor usability such as no sorting system for type of music for easy access to the musician and lack of management.

Admin

- This website is controlled by the admin who authorized to alter the website. Admin controlled all of the information that send via email by providers and present it at this website. Any unintended contents can be alter and delete if necessary. So, admin take part as connection between users and the system.

User

- User can created their own profiles to get special abilities such as gets more update, discounts and free stuffs. But a social network features such as interacting with each other's, get comments and others socialize are not provided by this website

1.5.3 Comparison between The Wknd and Online Music Store

	The Wknd	**Online Music Store**
Social network – Messaging – Comments – Discussion board (forum)	Not have	Have
Admin account	Have	Have
User account – User profiles – Photo album – Rating	Have (photo album only)	Have
Musician account – Musician profiles – Content management • Details • Albums • Pictures • Video • Songs	Not have Control by admin	Have Control by registered musician
Business transaction 1. Add to cart system	Have	Have

Table 1.5.3-1: Features Differences

1.6 Method of Approach

To develop a good system, we need to choose a suitable Software Development Life Cycle for this this system. To develop this system, first I need to clarify all the requirements needed for this system. After all requirements have been stated and analysed, I need to design a good User interface for this system. There will be a continuous interactive process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs. The third phase is the construction phases, which mean the system, will start to develop. In this phase user will be taking part in development and still can suggest for improvement. The last one is the Cutover Phases is the final task of the methodology. From above phases, the most suitable methodology for this kind of system is Rapid Application Development Methodology.

1.7 Scope and Limitation

The uses of software and hardware:

1. Software:
    - i) PHP languages
    - ii) XAMPP
        - Apache web server
        - MySQL Database
    - iii) Adobe Dreamweaver CS5
    - iv) Adobe Photoshop CS5
    - v) Microsoft Visio 2010

2. Hardware:
    - i) Laptop
    - ii) Desktop

Users or respondents:

1. Users
    i) Unregistered users
    ii) Registered users
    iii) Musician
2. Supervisor
3. PSM Coordinator

1.8 Outline of Material

The overall of this report consist of three (3) main parts.

Part 1 will discuss on the purpose of the project, current system and its limitation.

Part 2 will discuss on user requirement specification, design description and test planning on the system.

Part 3 will discuss on the conclusion obtain in the overall process of the development of the system.

PART 2: REPORT BODY

2.1 User Requirement

During designing a software product, the important and difficult process is to determine what the user needs is. Generally, customer not able to explain and discuss their needs, sometimes the information is not in a complete form, self-conflicting and less accurate. The project manager has the responsibility to understand their customer needs. Once the requirements is documented in URD, the software will spell out exactly as stated in the document because the contractual agreement had been sign off between both firms.

2.1.1 System Requirements

Online Music Store (OMS) will develop using web based application which is using Adobe Dreamweaver CS5 as an implementation platform and PHP scripting language, and interact with MySQL Server.

1. The web pages (HTML/PHP) will be used for client/users side interfaces
2. PHP language will be used for server side processing
3. Apache web server will be used for PHP and MySQL as a database server will use to store the information
4. Communication between client and server side is provided through HTTP/HTTPS protocol

i) System Interfaces

The system interfaces are as below:

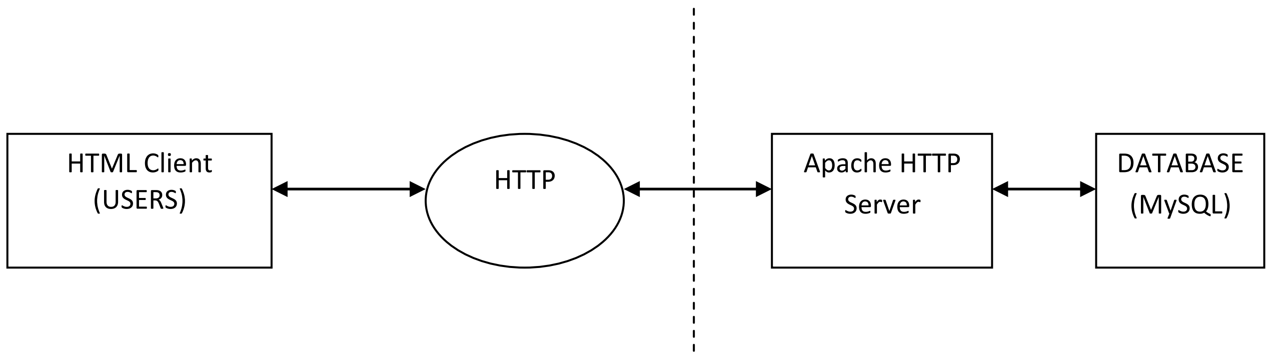


Figure 2.1.1-1: System Interfaces

ii) User Interfaces

There are three different category of users who will use the system; coordinator, student and lecturer. All users will access the system via web browser. The application should allow basic process such as insert, update, delete, and view for all of the users. The context diagram shows the user interfaces of the system:

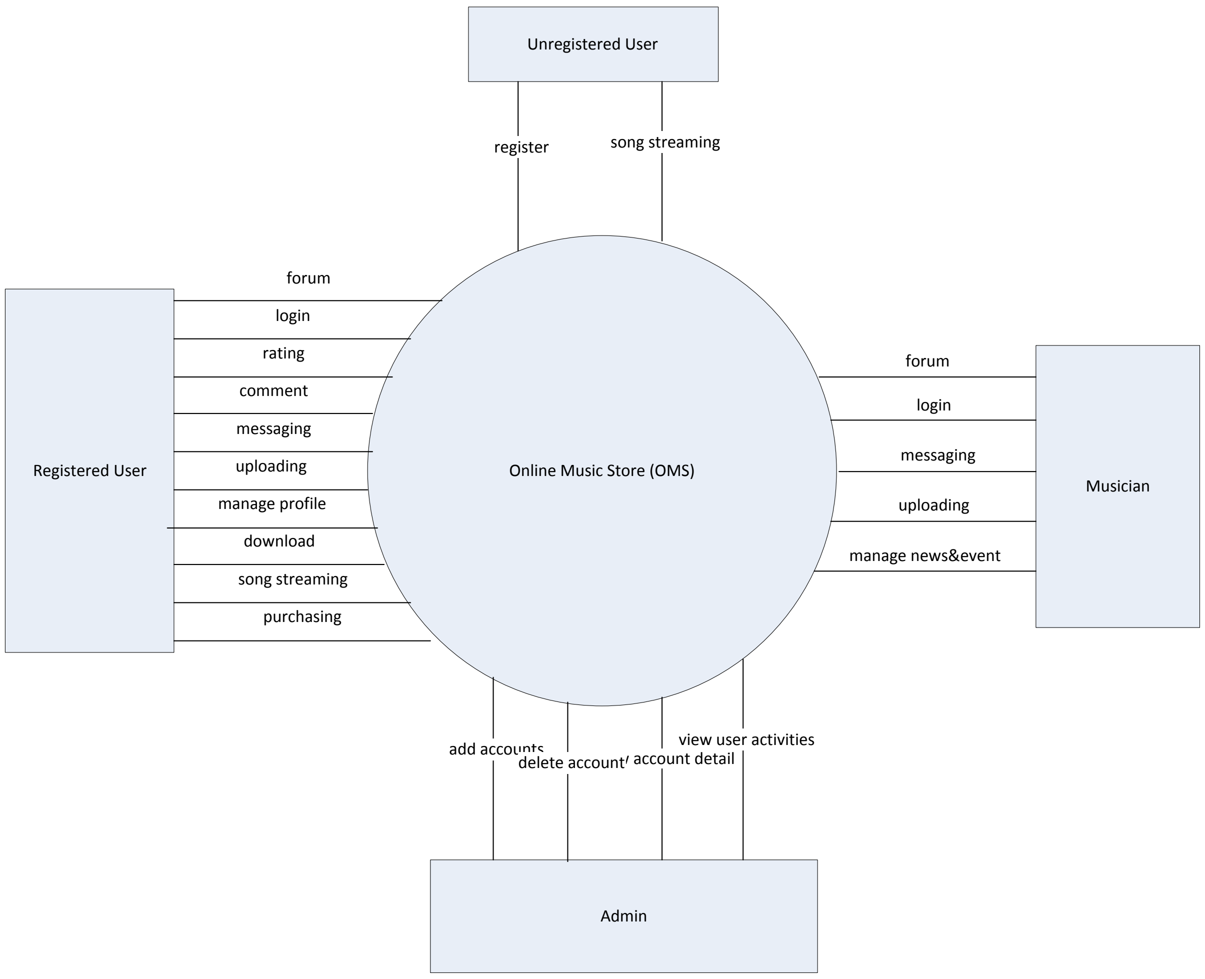


Figure 2.1.1-2: Context Diagram

1. **Registered User Interfaces**

   Registered user interfaces start with the user`s profile. At this user`s profile, it consist of inbox for message and playlist from the added song. There are menu on the above interfaces which consists music that will link to the musician profile, events, news, charts and forum. The Unregistered User interfaces can be said same with the registered user interfaces. The similarity between those interfaces are the menu above the interface and the differences between those interface are unregistered user interface don't start with user`s profile which mean there are no inbox for message and playlist from the added song.

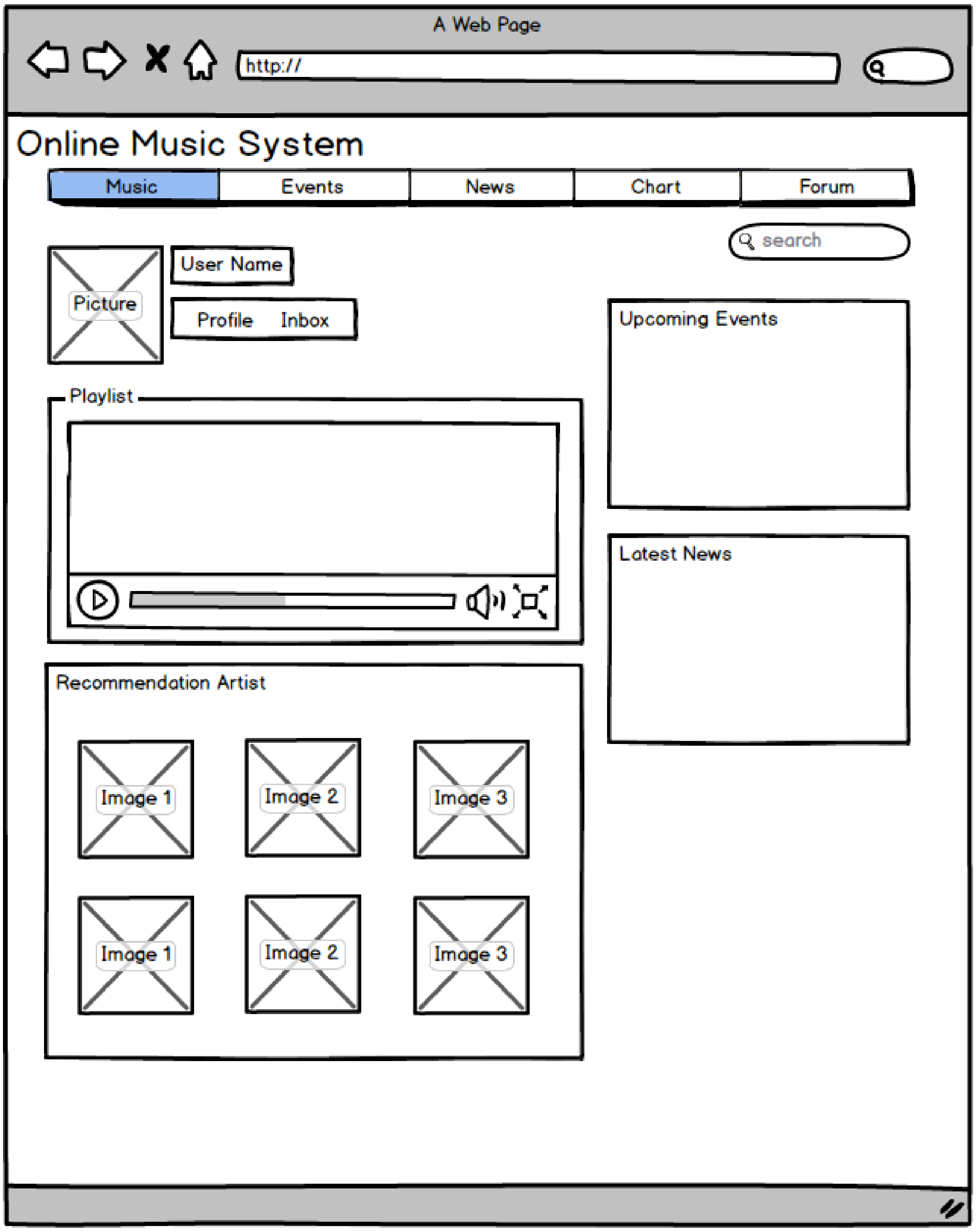


Figure 2.1.1-3: Registered User Interface

2. Musician Interfaces

Musician interfaces also start with the musician profile. At this profile, it consist of interfaces for manage the profile such as inbox for message, interface to upload items and manage news and events. The top menus of the musician interfaces are store and forum.

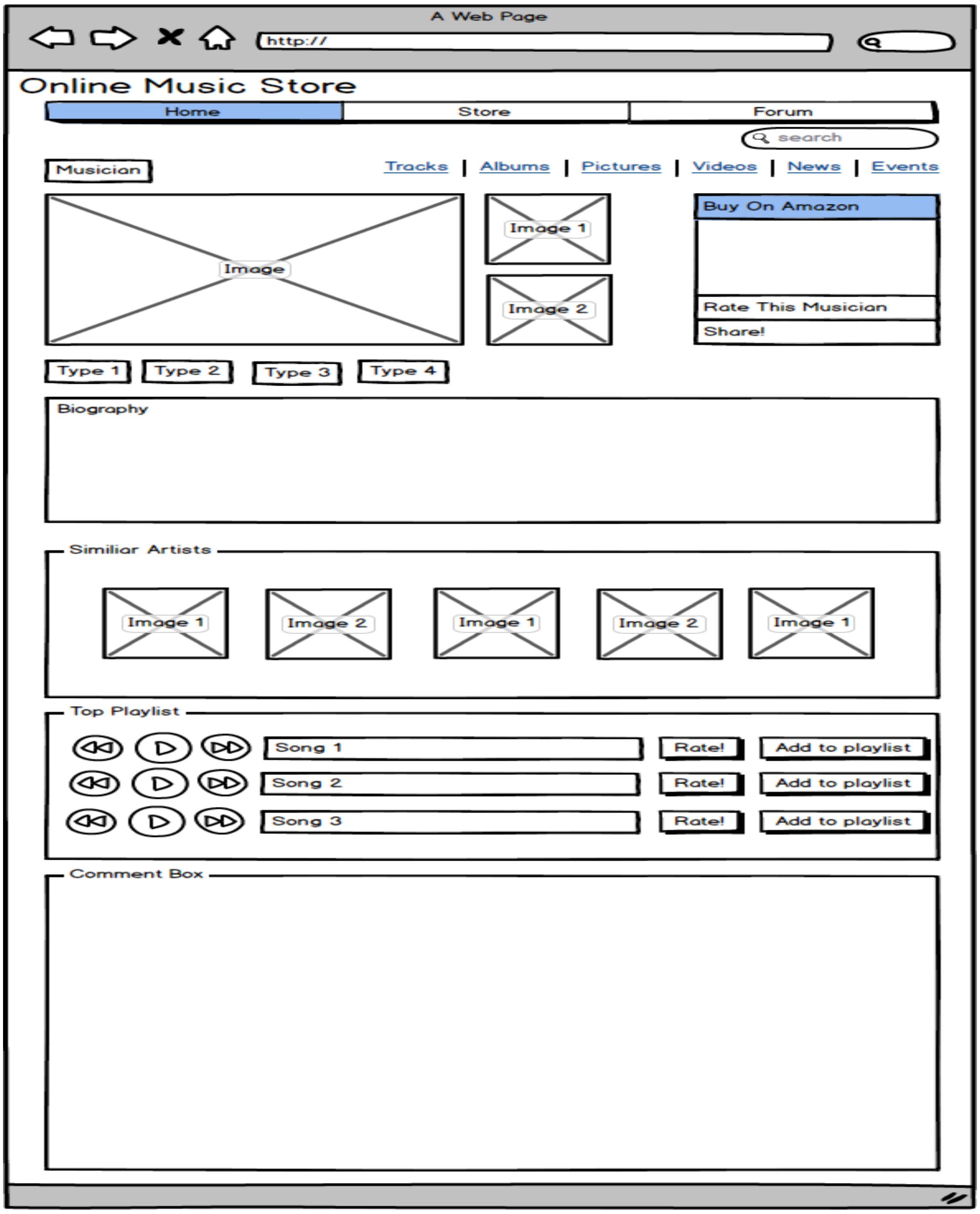


Figure 2.1.1-4: Musician Interface

3. Admin Interfaces

Admin interfaces consists of interfaces to add account, remove account, view user list, view details account and view user activities. The view user list account will have control at the side of list to perform an operation desired. In the view user activities account, the system user (registered user + musician) history will be display and can print the report. This interface consists of user activities such as song chart, top rated musician and top streamed song.

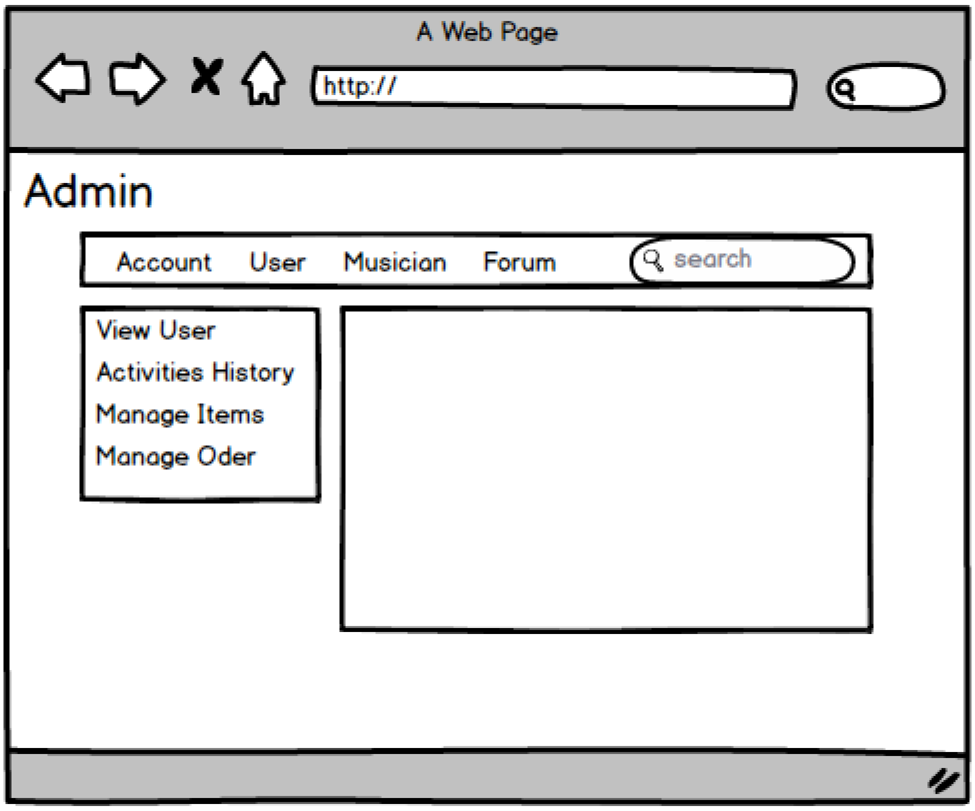


Figure 2.1.1-5: Admin Interface

iii) Hardware Interfaces

Minimum hardware requirements:

Client/Users Side			
	Processor	RAM	Disk Space
Internet Explorer (IE) 6.0 / Mozilla / Chrome & above	Pentium III at 500MHZ	512MB	60GB
Server Side			
	Processor	RAM	Disk Space
Apache HTTP Server V2.2.17	Pentium IV at 1.3GHz	512MB	200MB
MySQL V5.5.8	Pentium IV at 1.3GHZ	512MB	200MB (excluding data size)

Table 2.1.1-1: Hardware Interfaces

iv) Software Interfaces

This section identifies the type of software and their function that will be used to develop this system. The required software stated below:

Software	Function
Microsoft Windows Operating System • Window 7 Ultimate	• Operating system that will be used to develop the system • As a platform for a system to run
Microsoft Office 2010 • Microsoft Word 2010 • Microsoft PowerPoint 2010 • Microsoft Project 2010 • Microsoft Visio 2010	• Prepare reports and documentation • Prepare slide for presentation • Scheduling, planning and Gantt Chart development • Design, draw chart and diagram

Creatly.com	Open source website application for the chart designing and creating
Adobe Dreamweaver CS5	Design interfaces and generate codes
Adobe Photoshop CS5	Manage the pictures type work
Apache MySQL phpMyAdmin	Database for the system; generate database, database management and database platform
WinRAR	Compress project files

Table 2.1.1-2: Software Interfaces

v) Functional Requirement

Admin

- System shall allow admin to log in to the system and add, delete and alter the records of musician and their products
- System shall allow admin to add and delete user who are registered with the system
- System shall allow admin to monitor monthly sells and produce report to the admin
- System shall allow admin to view the statistic of the system such as how many user registered, frequently played song, most viewed musician and top rated musicians
- System shall allow admin to monitor the forum
- System shall allow admin to maintain user details and manage the music items in the inventory

Registered User

- System shall allow user to login and register to the system and stream, download and buy music
- System shall allow user to add item to shopping cart
- System shall allow user to reply or received private message
- System shall allow user to store their favourite music into their profile

- System shall allow user to interact with each other via comment box and forum
- System shall allow user to rate their favourite music
- System shall allow user to search for music and musician based on multiple keywords such as album name and music category

Unregistered User

- System shall allow user to stream and view the musician products
- System shall allow user to rate their favourite music
- System shall allow user to search for music and musician based on multiple keywords such as album name and music category

Musician

- System shall allow musician to login and register to the system
- System shall allow musician add, delete and alter their product
- System shall allow musician to interact with other user via private message or comment box
- System shall allow musician to take part in the forum too

vi) Constraints

There are several constraints that should be notified in this system.

- System is limited to HTTP/HTTPS protocols as the system is a web based application.
- Use of hardware and software by users should fulfil the minimum requirement of the system.
- Server must always be available.

vii) Types of files

Compressed files formatting:

- Audio format (.mp3, .mpeg-4, .wma)
- Image format (.jpeg, .png)
- Video format (.avi, mpeg-2, mpeg-4)

viii) Assumptions and Dependencies

1. The system will able to access by major of internet browser such as Internet Explorer, Google Chrome and Mozilla Firefox.
2. The speed of accessing the system depends on the network speed.
3. Higher RAM provides higher performance of the system.
4. Administrator was already created.
5. PayPal account or credit card demanded for the purchasing activity.

2.1.2 Specific Requirements

i) Function

- Use Case Diagram

Online Music Store (OMS) had four actors which are Unregistered Users, Registered Users, Musician and Admin. Registered Users and Musician combined together as Users since major of its cases similar to each other's. Below is the use case diagram that divided into three components which is the Unregistered User, Users (Musician + Registered User) and admin. Reason for the dividing is to make its easy to classify the system function since the combination of all actors produce a large and complex use case diagram.

Use Case Diagram #1

Actor : Unregistered User

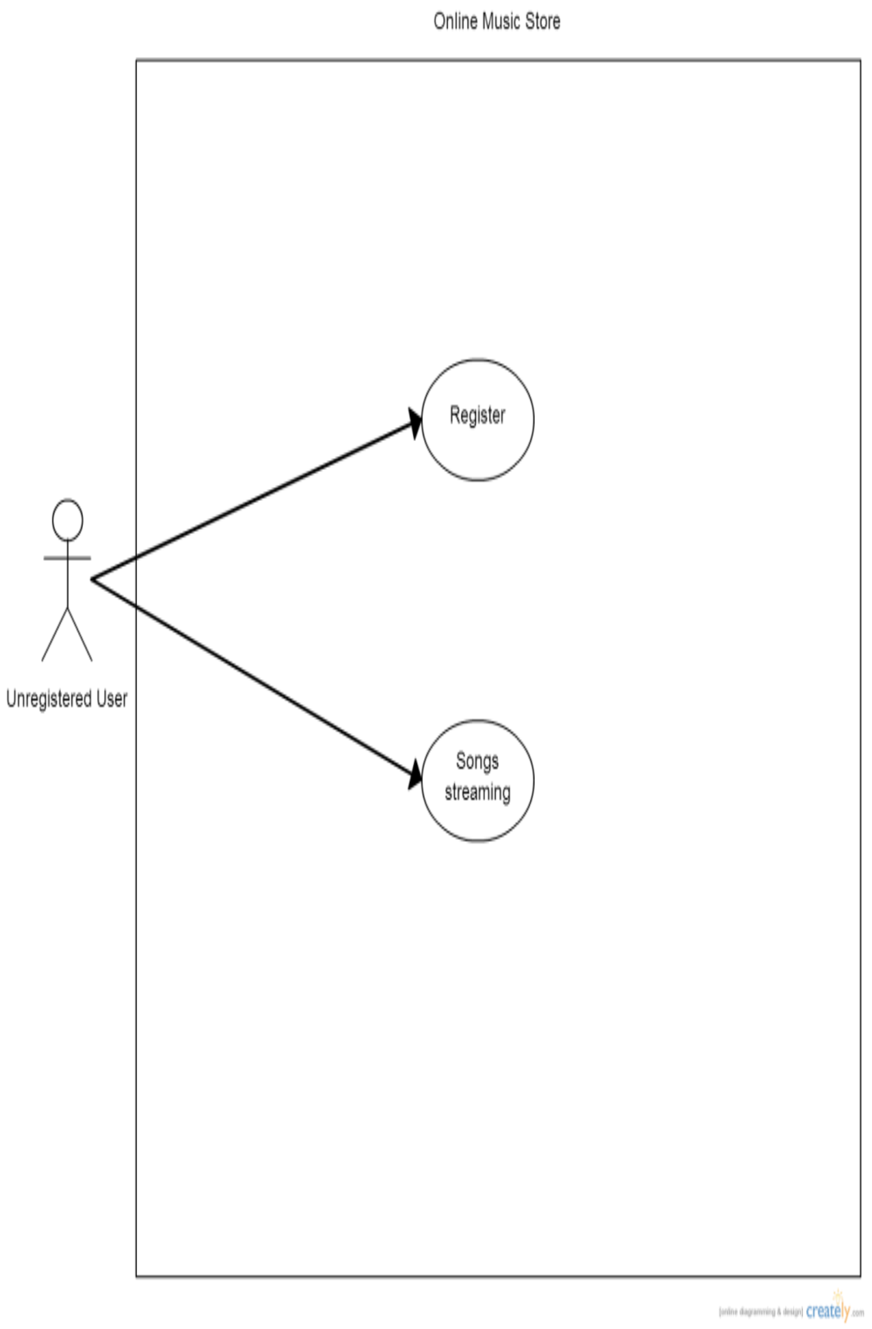


Figure 2.1.2-1: Use Case Diagram for Unregistered User

Use Case:	Register
ID:	US-001
Scope:	Register by unregistered users
Priority:	1/15
Summary:	This use case to register the users before they enter the system. There is an option given whether users want to register or not register. Registered users had additional abilities than unregistered users.
Primary Actor:	Unregistered User
Supporting Actors:	NA
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	User must not registered with the system
Trigger:	NA
Normal Flow:	1. Select Register 2. Enter require information **Loop**: Repeat 2 if the information is available in the database 3. Proceed to the system
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Registered User
Non-Behavioural Requirements:	NA
Open Issues:	
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-1: Register Use Case Description

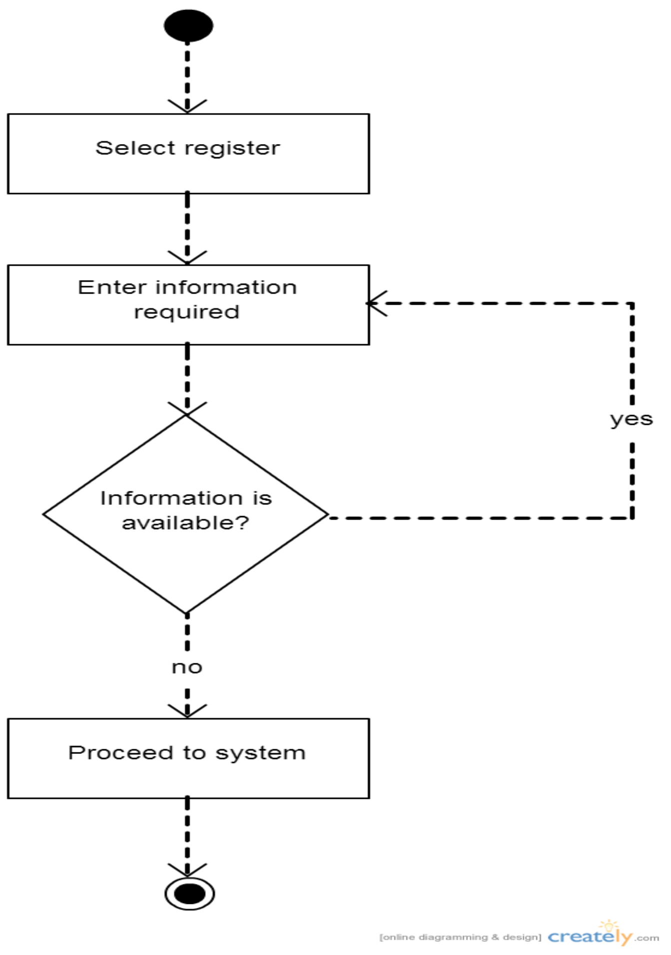

Select register
Enter information required
Information is available?
yes
no
Proceed to system
[online diagramming & design] creately.com

Figure 2.1.2-1: Register Activity Diagram

<u>Activity Diagram Flow Detail Description</u>

1. Users will select the Register link in the homepage
2. The Register pages will appear with several information need to input such as name and address.
3. After the information needed was input, next step is to click the Next button to process the information.
4. Information inserted will add to the database. If the information already available in the database, the system will send a message says duplication on the information and require users to input others information.
5. Activity will end with message says the success in registration and users now in the registered stated.

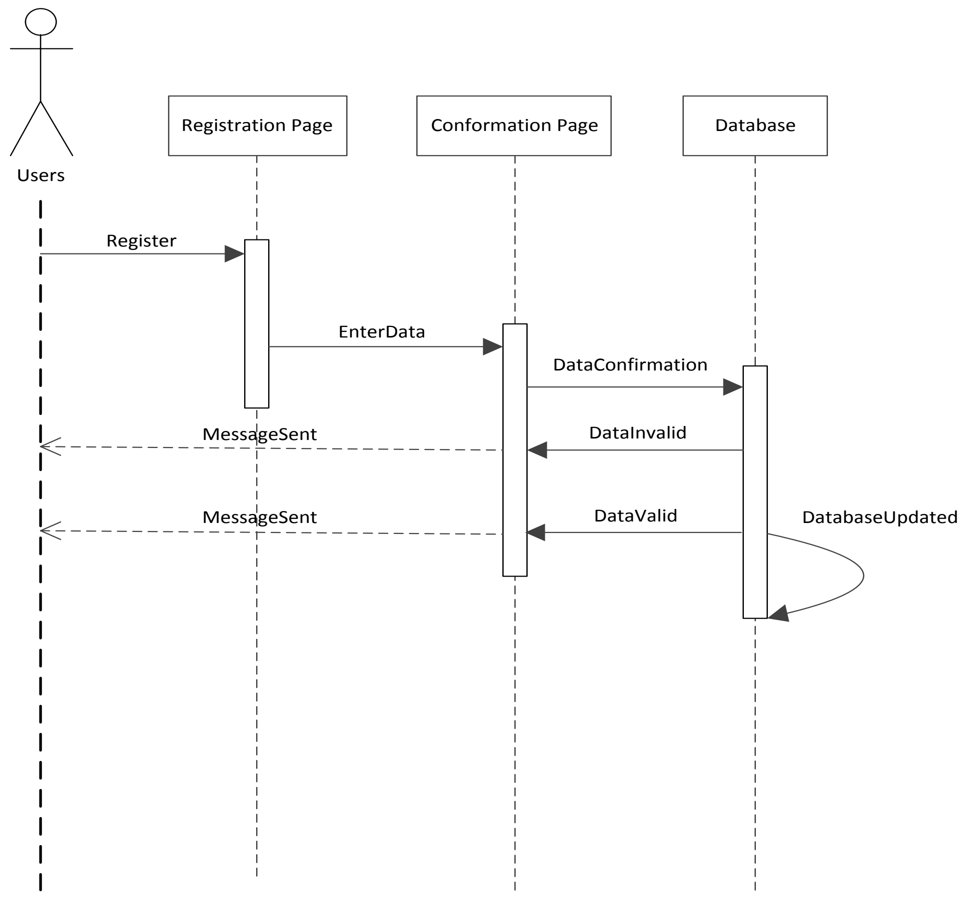


Figure 2.1.2-1: Register Sequence Diagram

Use Case:	Song streaming
ID:	US-002
Scope:	Streaming a song from a system
Priority:	2/15
Summary:	This use case to stream a song from a system. Either unregistered users or registered user can stream song from the system. A playlist will be embedded on the system to support the streaming activity.
Primary Actor:	1. Unregistered User 2. Registered User
Supporting Actors:	Musician
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	The songs streaming is applied from musician by the unregistered or registered users
Trigger:	NA
Normal Flow:	1. Users select their desired song to play **Loop:** Repeat 1 if the selected song is not right 2. A selected song will played. **Loop:** Repeat 2 if users decided to repeat the song
Sub-Flows:	NA
Alternate Flow/ Exceptions:	1a. Users can download the downloadable songs 2a. Stop the song and move on to others activities 2b. Pause the song in timestamp 1 second to 10 minutes
Post-Condition:	Streaming log for the Registered Users
Non-Behavioural Requirements:	NA
Open Issues:	
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-2: Song Streaming Use Case Description

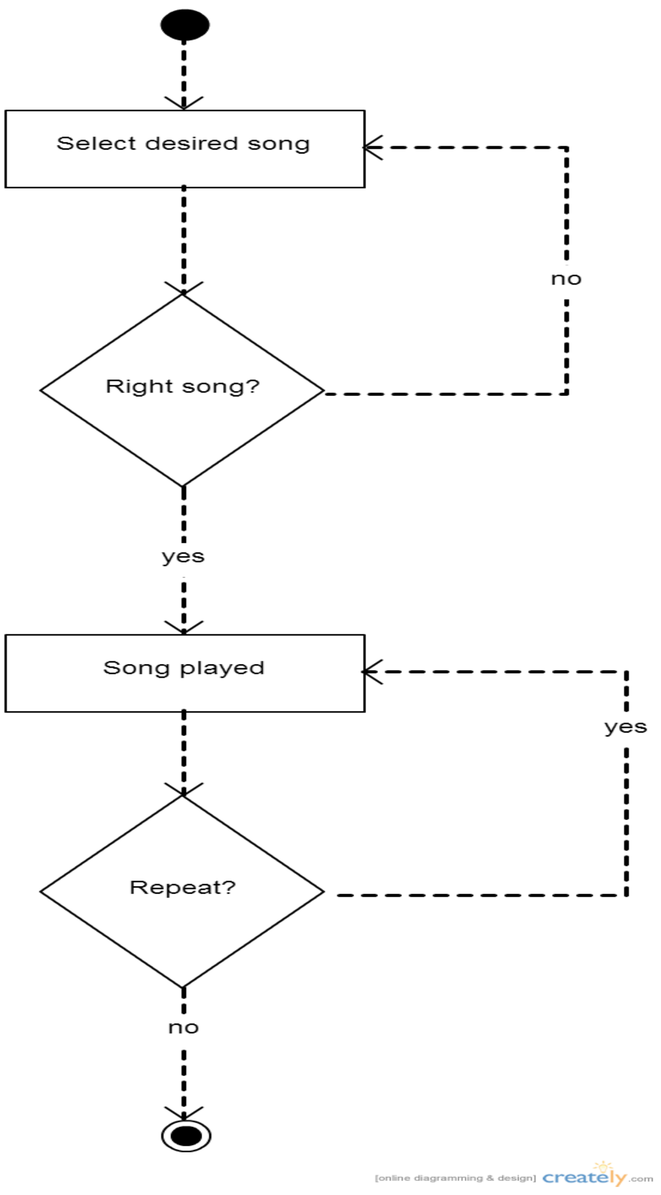


Figure 2.1.2-2: Song Streaming Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Users choose the song that like to play in the musician pages.
2. Begin the song play with the play button embedded in the playlist.
3. There are others buttons included such as stop button to stop the playing song and pause button for the pausing the song.
4. Users can choose other songs desired to continue the hearing.

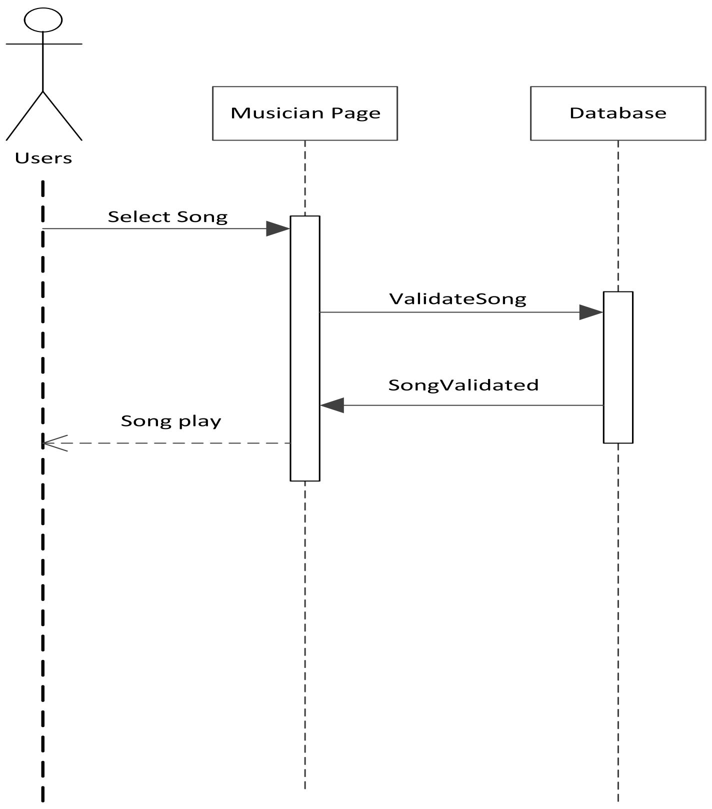


Figure 2.1.2-2: Song Streaming Sequence Diagram

Use Case Diagram #2

Actor: Users (Musician + Registered User)

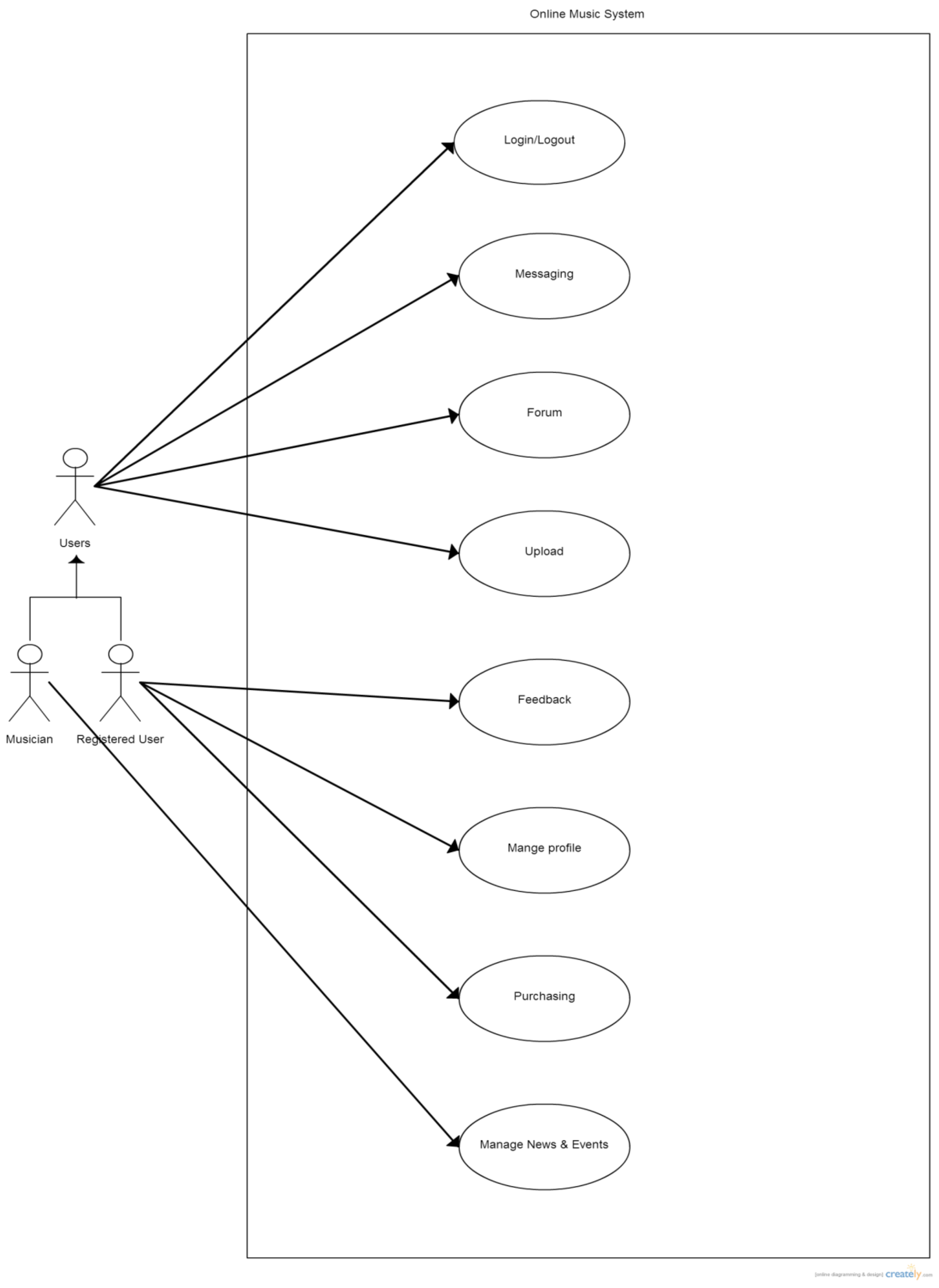


Figure 2.1.2-2: Use Case Diagram for User (Musician + Registered User)

Use Case:	**Login/Logout**
ID:	US-003
Scope:	Login/Logout to begin/end the users session
Priority:	3/15
Summary:	This use case to login or logout from the system. The need for this activity is to create a session to the system. A profile will be creating for the users that register into the system and the login/logout activity is like a gateway or identification to the users that owns that profile.
Primary Actor:	1. Musician 2. Registered User
Supporting Actors:	NA
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	User must be unregistered into the system
Trigger:	NA
Normal Flow:	Login: 1. Select login 2. Enter the required information (i.e. username, password) **Loop:** Repeat 2 if the data is invalid 3. Session begin Logout: 1. Select logout 2. Confirmation for the selection **Loop:** Repeat 1 if the selection is no which is not intended to logout 3. Session ended
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Login: Session begin Logout: Session ended
Non-Behavioural Requirements:	NA
Open Issues:	
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-3: Login/Logout Use Case Description

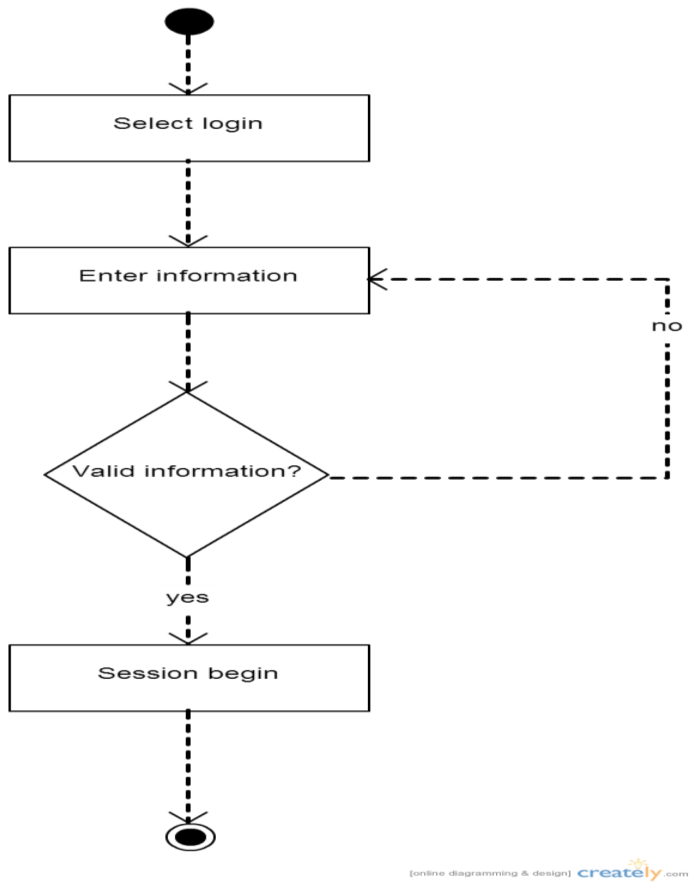


Figure 2.1.2-3: Login Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Users select the Login link from the homepage
2. The Login Page will appear with the box required information such as username and password to be entering.
3. The system will validate the entered information from the database. Valid information will give permission to the users to enter system while invalid information will sent a warning message that users can`t enter the system until valid information is recognized.
4. The session will begin if the users authorized by the system to enter system with several features such as download songs and make a purchase.

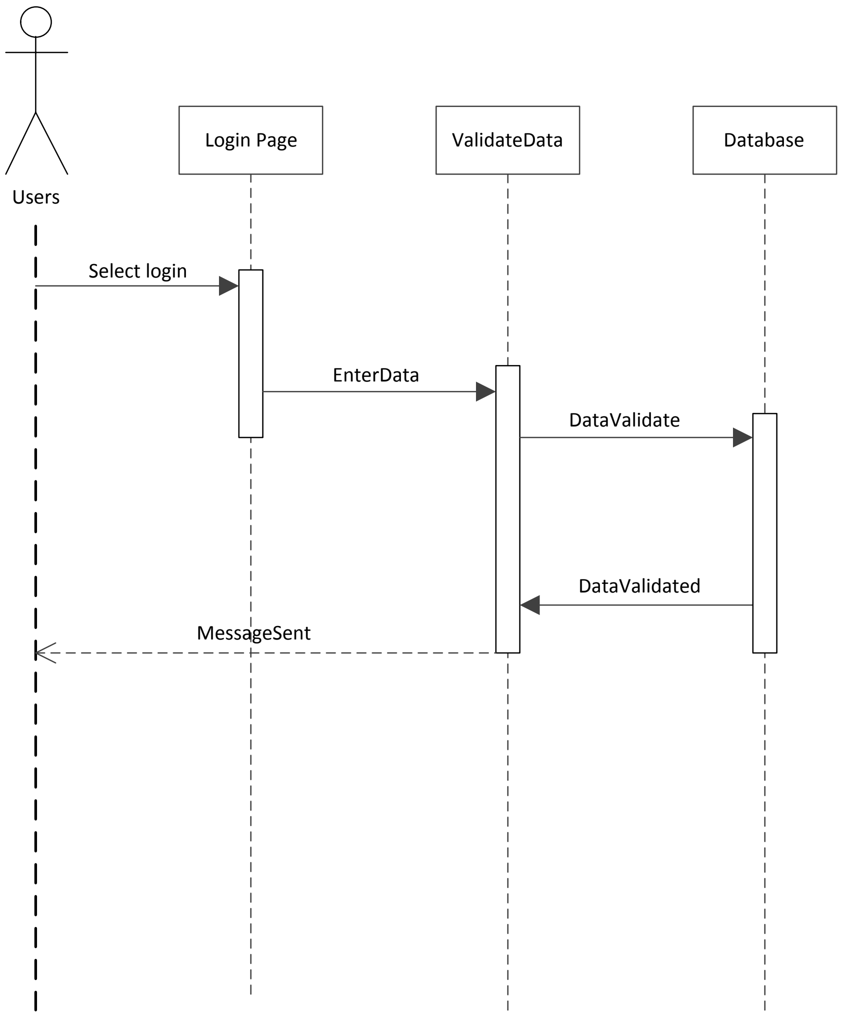


Figure 2.1.2-3: Login Sequence Diagram

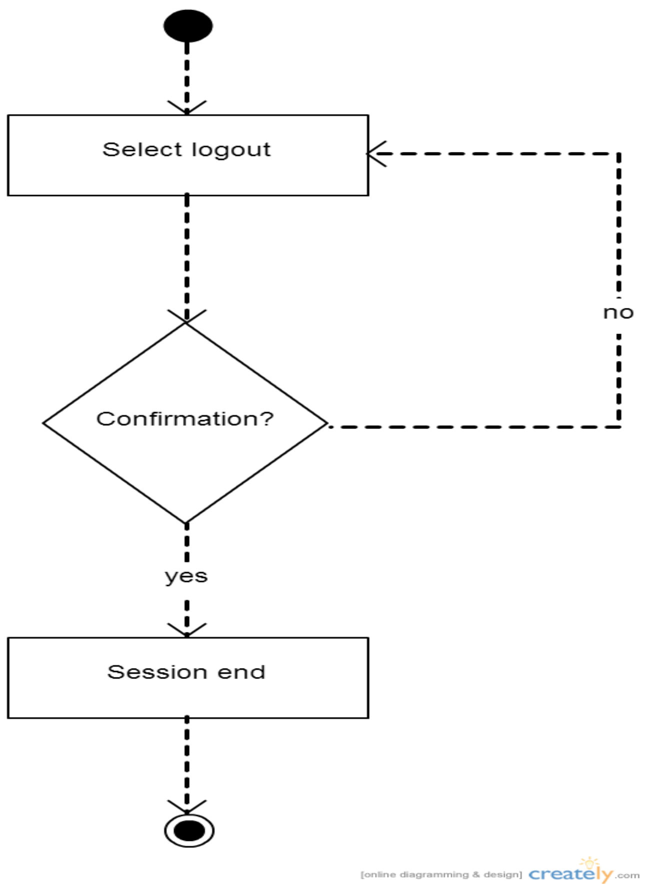


Figure 2.1.2-4: Logout Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Users will select the Logout link at the homepage.
2. The confirmation message will popped out asking for confirmation to logout or to stay.
3. If the users choose to stay, the system will revert back to the current page without anything happens.
4. If the users choose to logout, the system will terminated the users session and continue as a unregistered users.

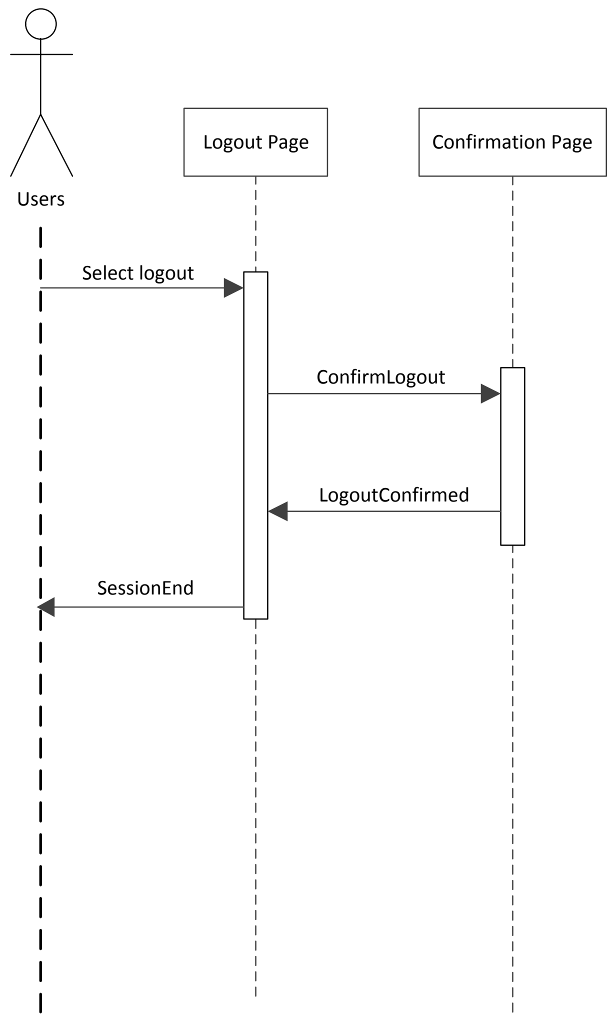


Figure 2.1.2-4: Logout Sequence Diagram

Use Case:	Messaging
ID:	US-004
Scope:	To send a message to other users
Priority:	4/15
Summary:	This use case to allow registered users to send message to others registered users. Sent messages can be read via inbox in the user profile. There are two situations to send message that will describe below.
Primary Actor:	1. Musician 2. Registered User
Supporting Actors:	NA
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	User must be registered into the system.
Trigger:	NA
Normal Flow:	1. Search for registered users or musician 2. Select message 3. Start to message 4. Message sent Loop 1: Repeat 3 if the delete button is selected Loop 2: Repeat 4 if the edit button is selected
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Message sent
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-4: Messaging Use Case Description

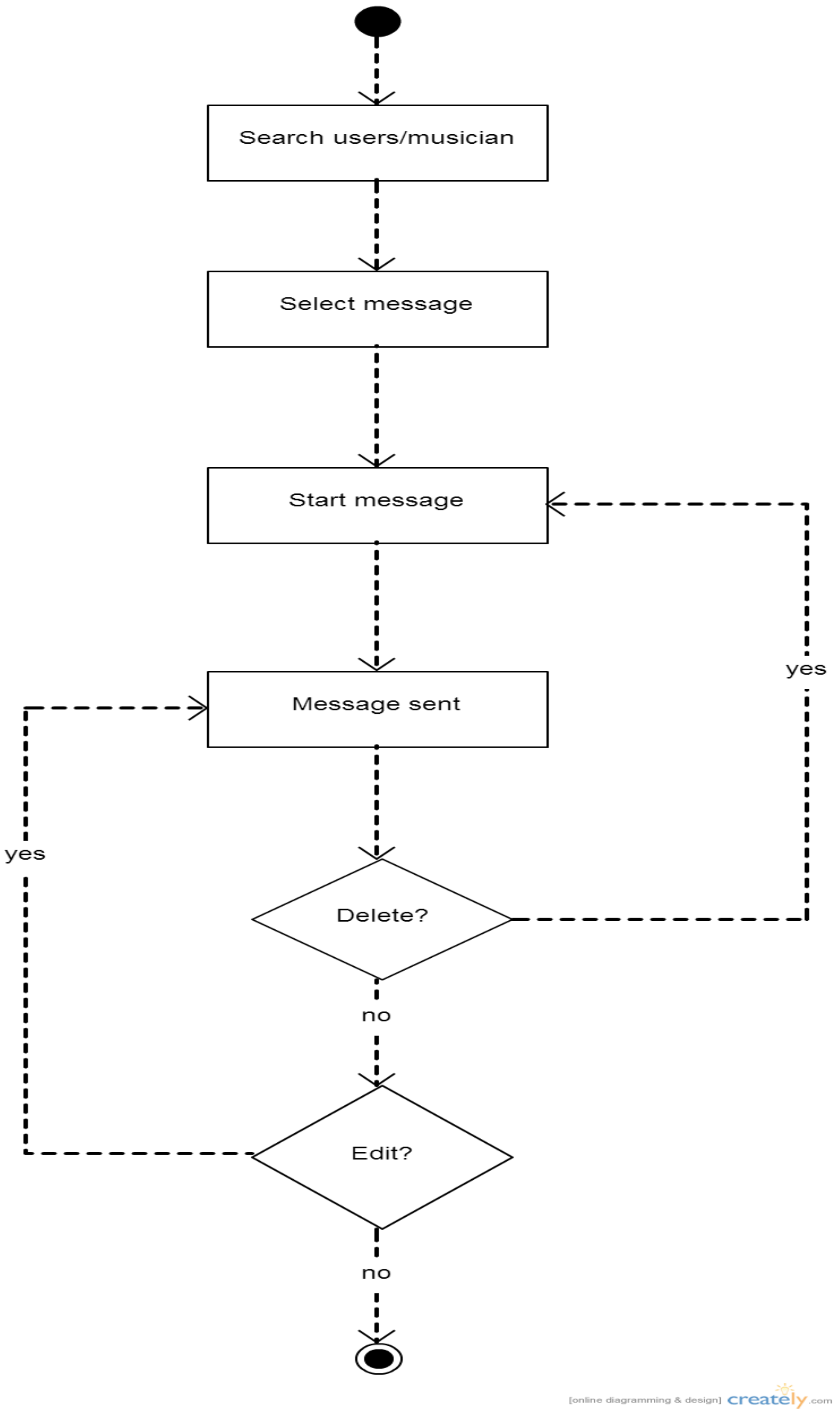


Figure 2.1.2-5: Messaging Activity Diagram

Activity Diagram Flow Details Description

1. Users search for the registered users or musician that his intended to send message.
2. Select the Message link at the above of the searched page.
3. The Message Page will appears and start a typing.
4. After the message sent, users can edit the message and delete it using the inbox in the user's profiles.
5. Inbox will store the user's message activities included the received message from other users.

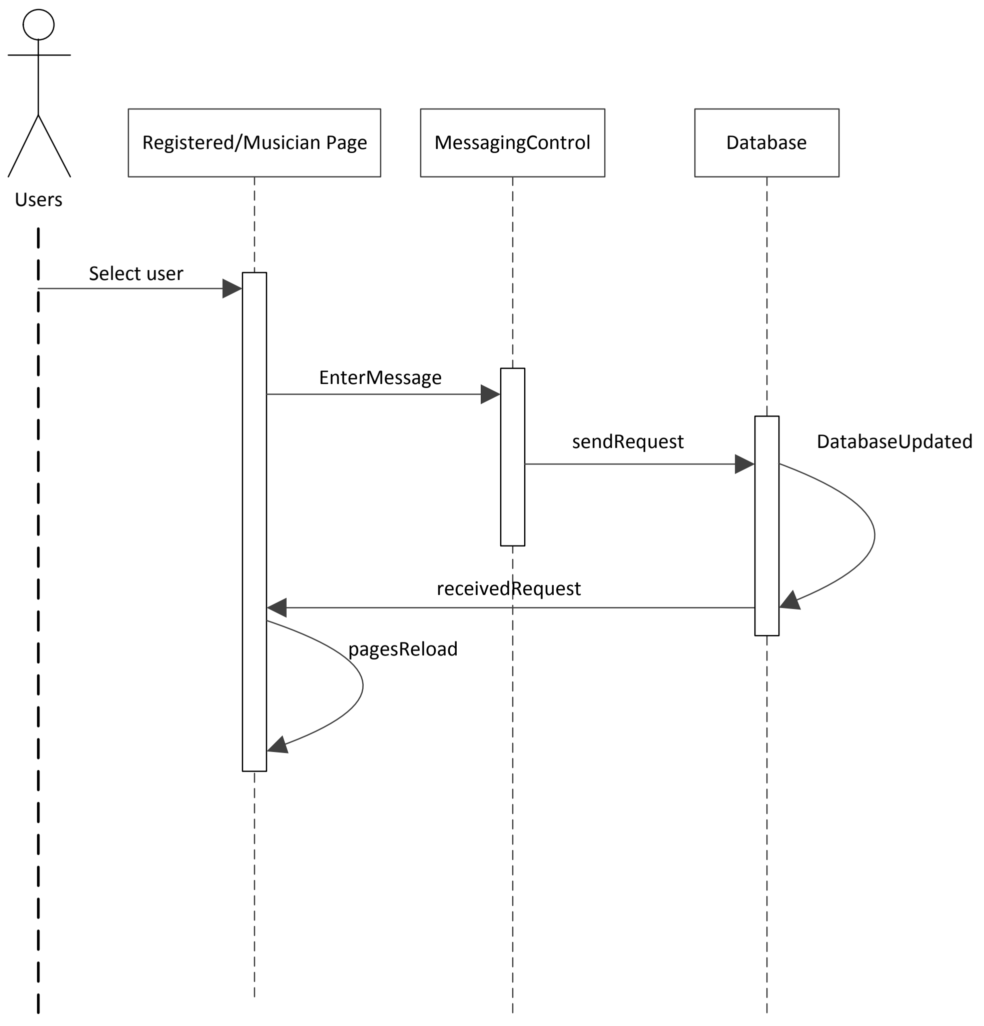


Figure 2.1.2-5: Messaging Sequence Diagram

Use Case:	Forum
ID:	US-005
Scope:	Give the users abilities to join into forum
Priority:	5/15
Summary:	This use case to allow registered users and musician to join into forum. This activity start with moderator opens the title such as “the best album in 2013” and participants (users) can discuss among the others regarding the topic.
Primary Actor:	1. Musician 2. Registered User
Supporting Actors:	Admin
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	User must be registered into the system.
Trigger:	NA
Normal Flow:	1. Users select forum link 2. Users select the topic to join 3. Start typing
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Comment sent
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-5: Forum Use Case Description

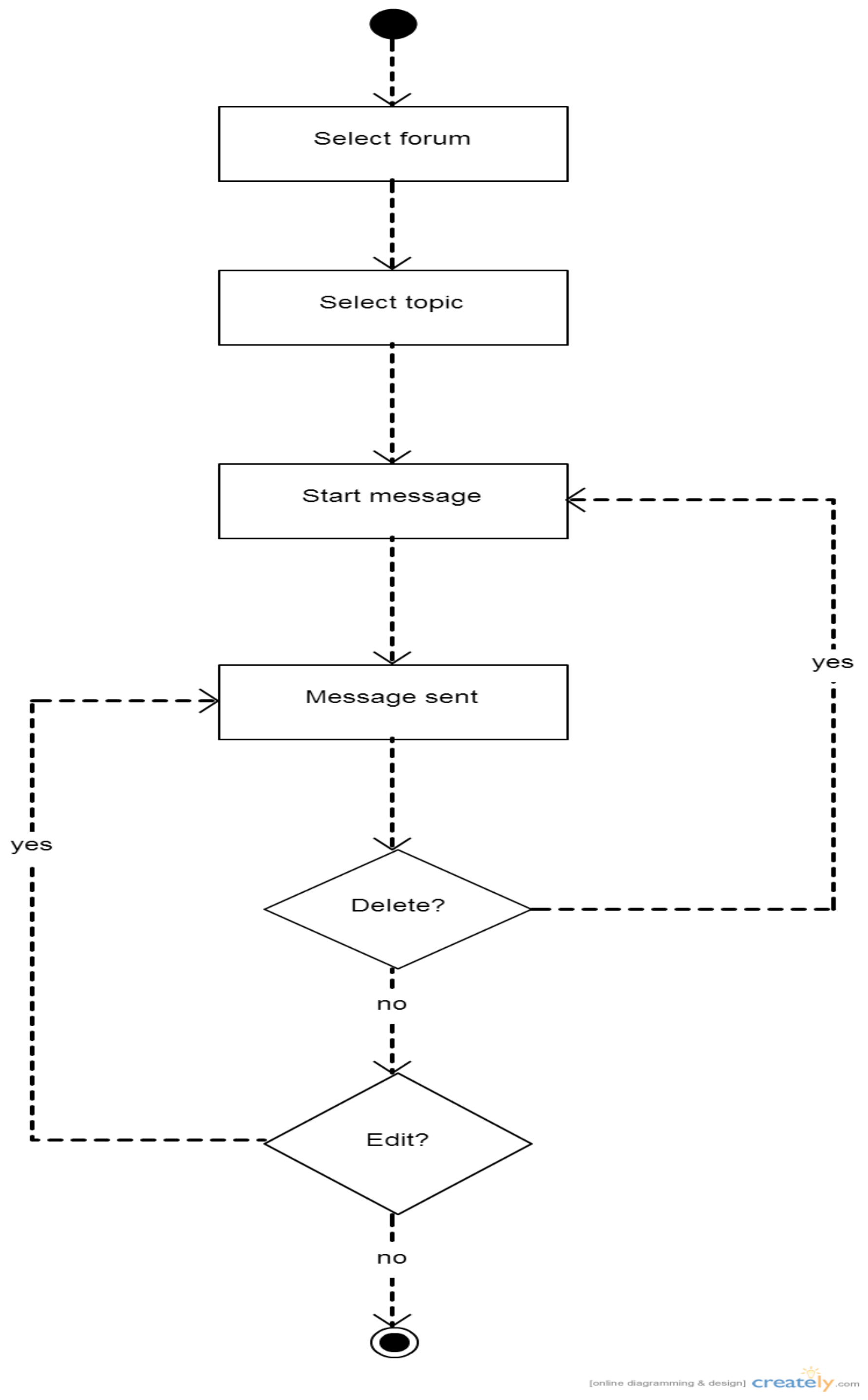


Figure 2.1.2-6: Forum Activity Diagram

Activity Diagram Flow Details Description

1. Users select the forum link at the homepage.
2. The Forum Page will come out with a lists of topic available created by moderator and users.
3. Users select the topic they desired to join.
4. Users begin the typing for comment, critique or give solution. That’s what forum for.
5. After comment sent, users can edit the comment if mistake happens or delete it from the system.
6. All forum activities will be stored in the database.

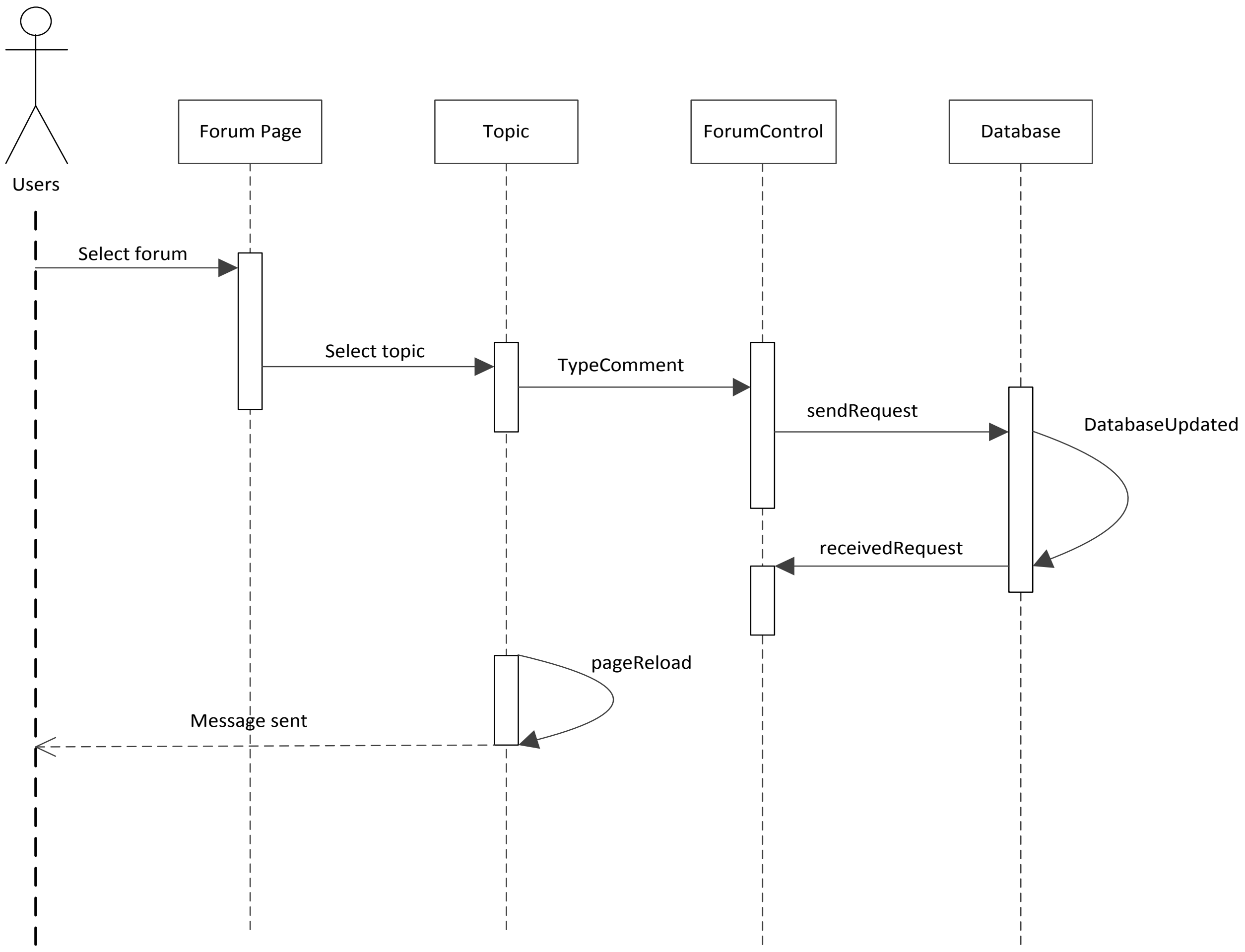


Figure 2.1.2-6: Forum Sequence Diagram

Use Case:	Upload
ID:	US-006
Scope:	To upload files into the system
Priority:	6/15
Summary:	This use case to allow registered users and musician to upload files into the system. There are differences between registered users and musician regarding types of files that they can upload. Registered users can upload files like picture only while musician can upload files like audio, video and pictures.
Primary Actor:	1. Musician 2. Registered User
Supporting Actors:	Admin
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	User must be registered into the system.
Trigger:	NA
Normal Flow:	1. Select file to upload Loop: Repeat 1 if the file is in invalid forms 2. File uploaded
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Files uploaded
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-6: Upload Use Case Description

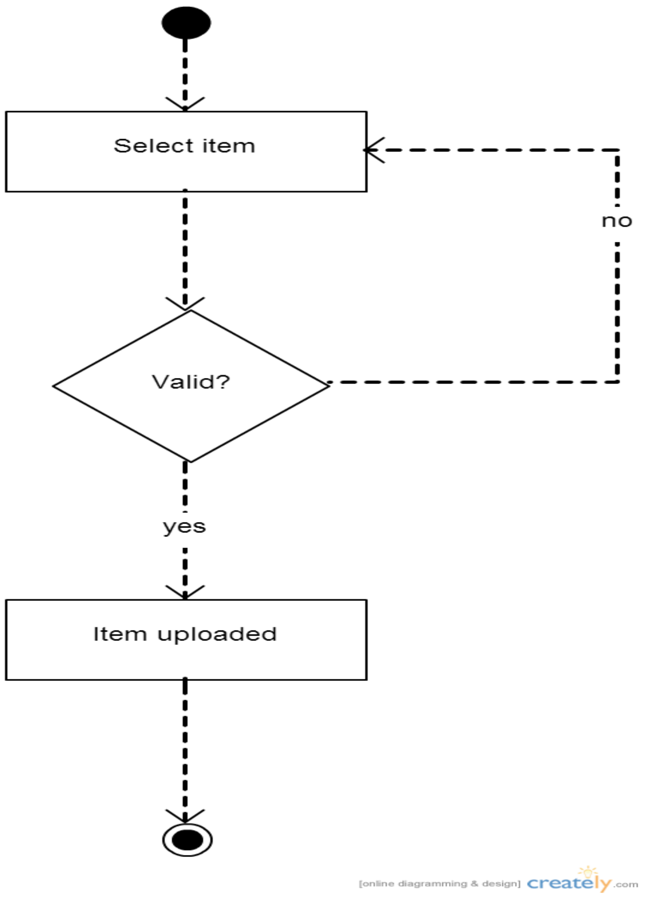


Figure 2.1.2-7: Upload Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Users select the file to be uploading.
2. Uploading box will popped out and starts the file selection.
3. The system will give warning if the file is in inappropriate format like described in the constraint topic.
4. The successful uploaded file will store in the database.

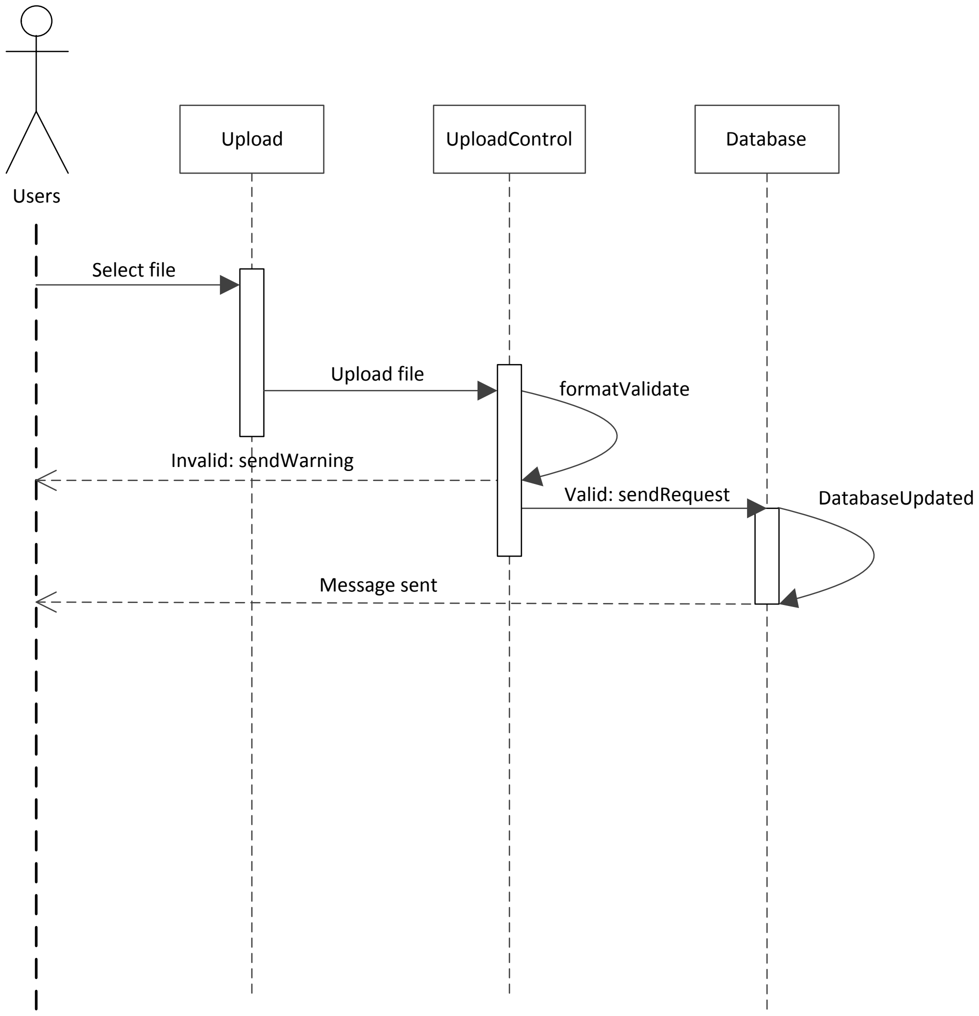


Figure 2.1.2-7: Upload Sequence Diagram

Use Case:	**Feedback**
ID:	US-007
Scope:	To give a feedback to other users
Priority:	7/15
Summary:	This use case allows registered users to give a feedback to others users such as give a rating and ability to comment. Mostly this activity is for the musician users.
Primary Actor:	1. Musician 2. Registered User
Supporting Actors:	Admin
Stakeholders:	NA
Generalization:	NA
Include:	1. Rating 2. Comment
Extend:	NA
Precondition:	User must be registered into the system and only applied to musician users.
Trigger:	NA
Normal Flow:	Rating: 1. Select users 2. Select rate button **Loop:** Repeat 1 if the undo button been selected 3. System updated with the addition rating to the users Comment: 1. Enter comment in the comment box **Loop:** Repeat 2 if the cancel button is selected 2. Comment sent **Loop 1:** Repeat 2 if the delete button is selected **Loop 2:** Repeat 3 if the edit button is selected 3. System updated with new comment
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	NA
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-7: Feedback Use Case Description

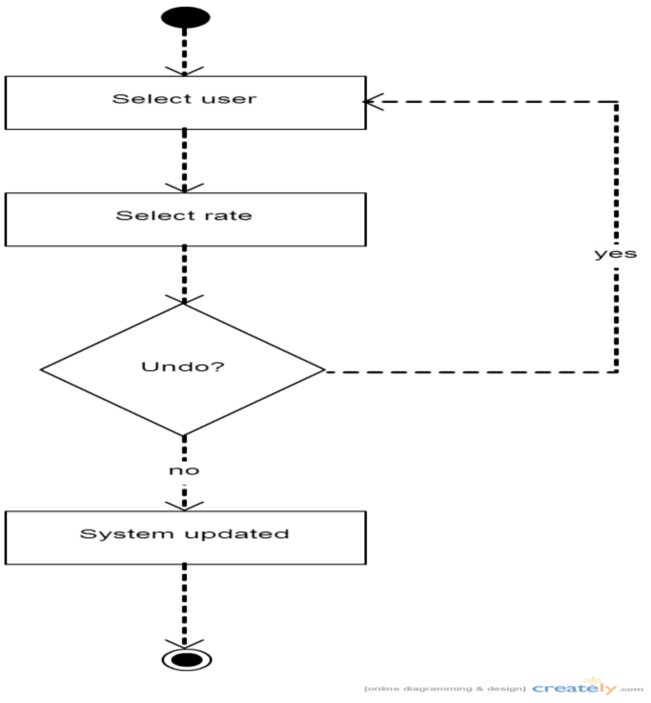


Figure 2.1.2-8: Rating Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Users will select the musician to be rating.
2. Process will continue after users click on the rating button at the musician pages.
3. In timestamp 1 second to 3 second, the musician pages will auto refresh and this is how users can know his rated take action with the increase number near the rating button.

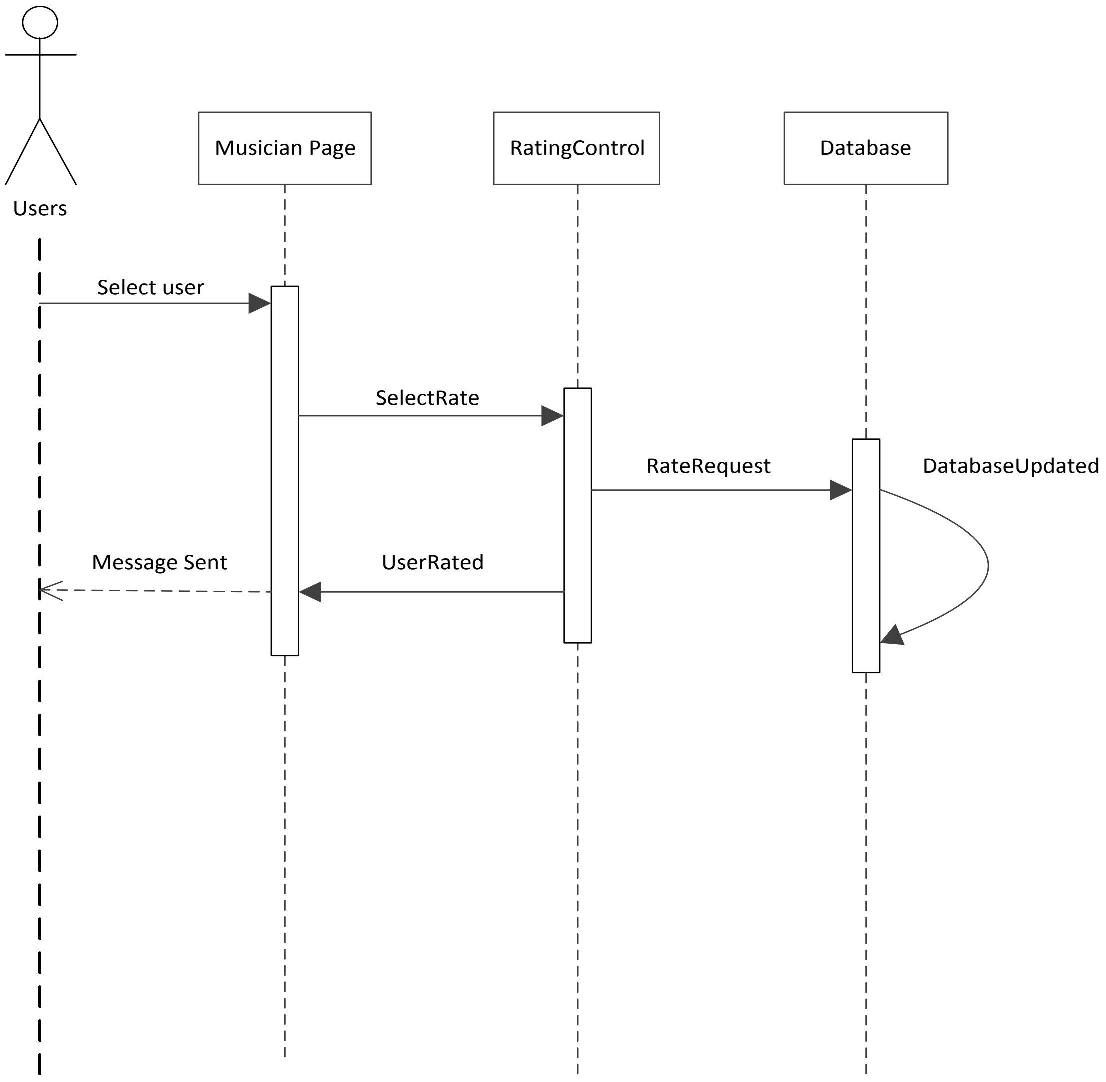


Figure 2.1.2-8: Rating Sequence Diagram

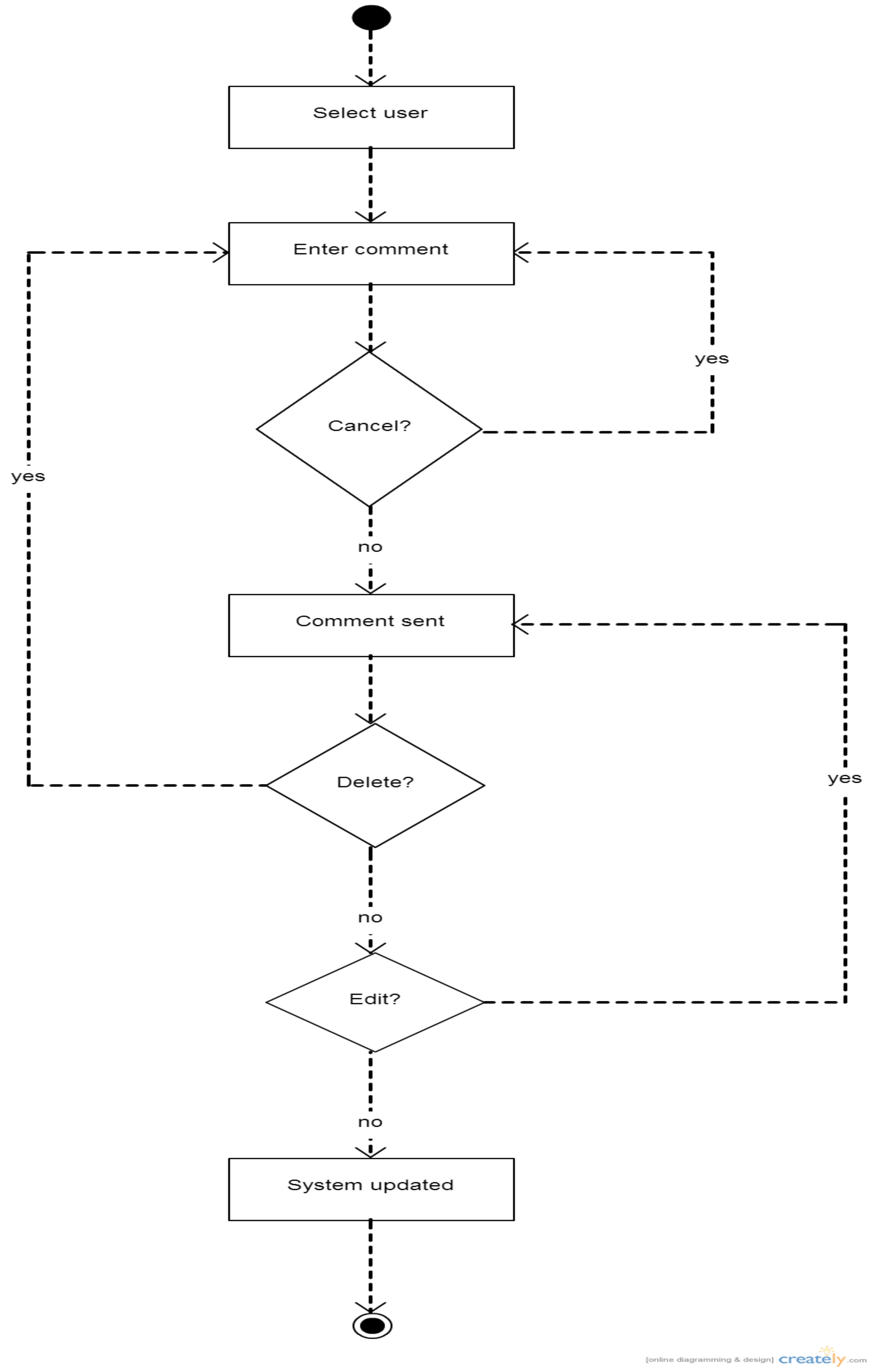


Figure 2.1.2-9: Comment Activity Diagram

Activity Diagram Flow Details Description

1. Users will select the musician to be comment.
2. Begin the typing to comment.
3. After the typing, there are two buttons provided which are cancel and send button. Cancel button will act as a cancelation for the words typed in the comment box while send button will act as an agreeable for the words typed in the comment box.
4. The system will auto refresh so users can see their comments after 1-3seconds.
5. There are control included in this comment system such as edit and cancellation to the comments.
6. All of the comments will be added into database.

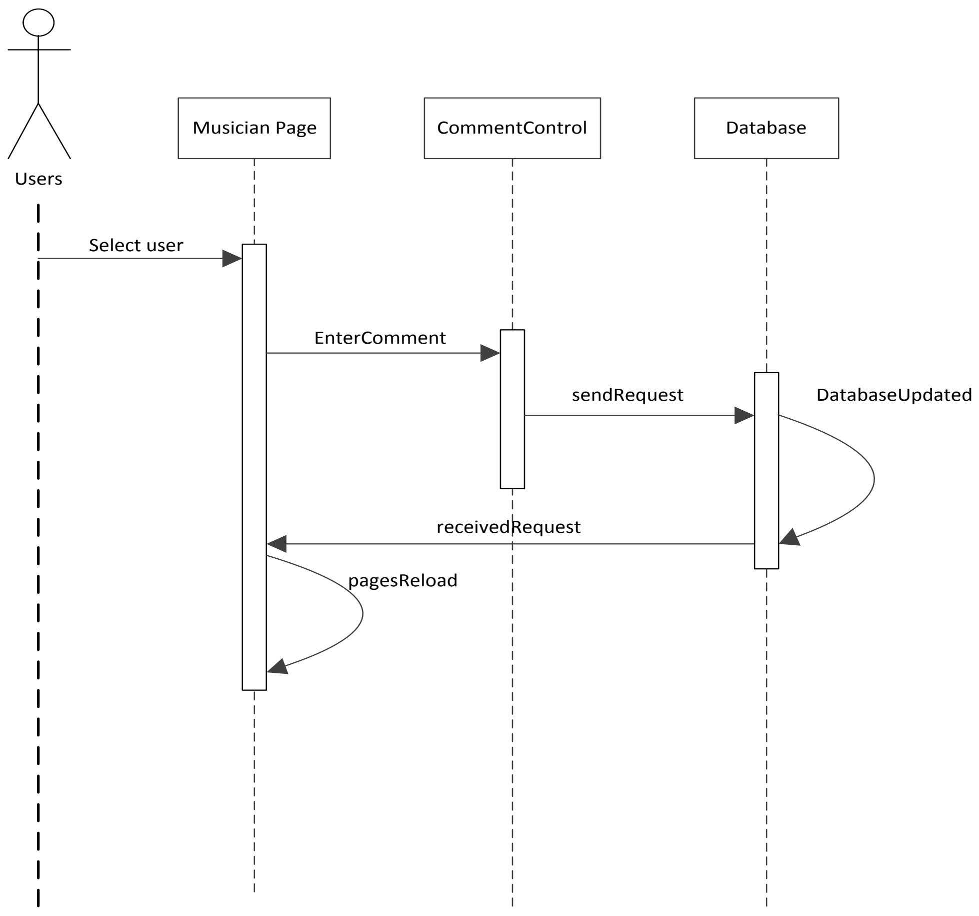


Figure 2.1.2-9: Comment Sequence Diagram

Use Case:	Manage Profile
ID:	US-010
Scope:	To manage the registered users profile
Priority:	10/15
Summary:	This use case to allow registered users to manage their own profile such as add and remove songs from profile
Primary Actor:	Registered User
Supporting Actors:	Admin
Stakeholders:	NA
Generalization:	NA
Include:	1. Add songs to profile 2. Remove songs from profile
Extend:	NA
Precondition:	User must be registered into the system and the songs added only can applied with the users add it from musician pages
Trigger:	NA
Normal Flow:	1. Select the musician 2. Select the add button near the chosen song Loop: Repeat 2 if the delete button is chosen 3. Confirmation box appear Loop: Repeat 1 if choose to continue 4. Proceed to system
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Songs added to profile
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-8: Manage Profile Use Case Description

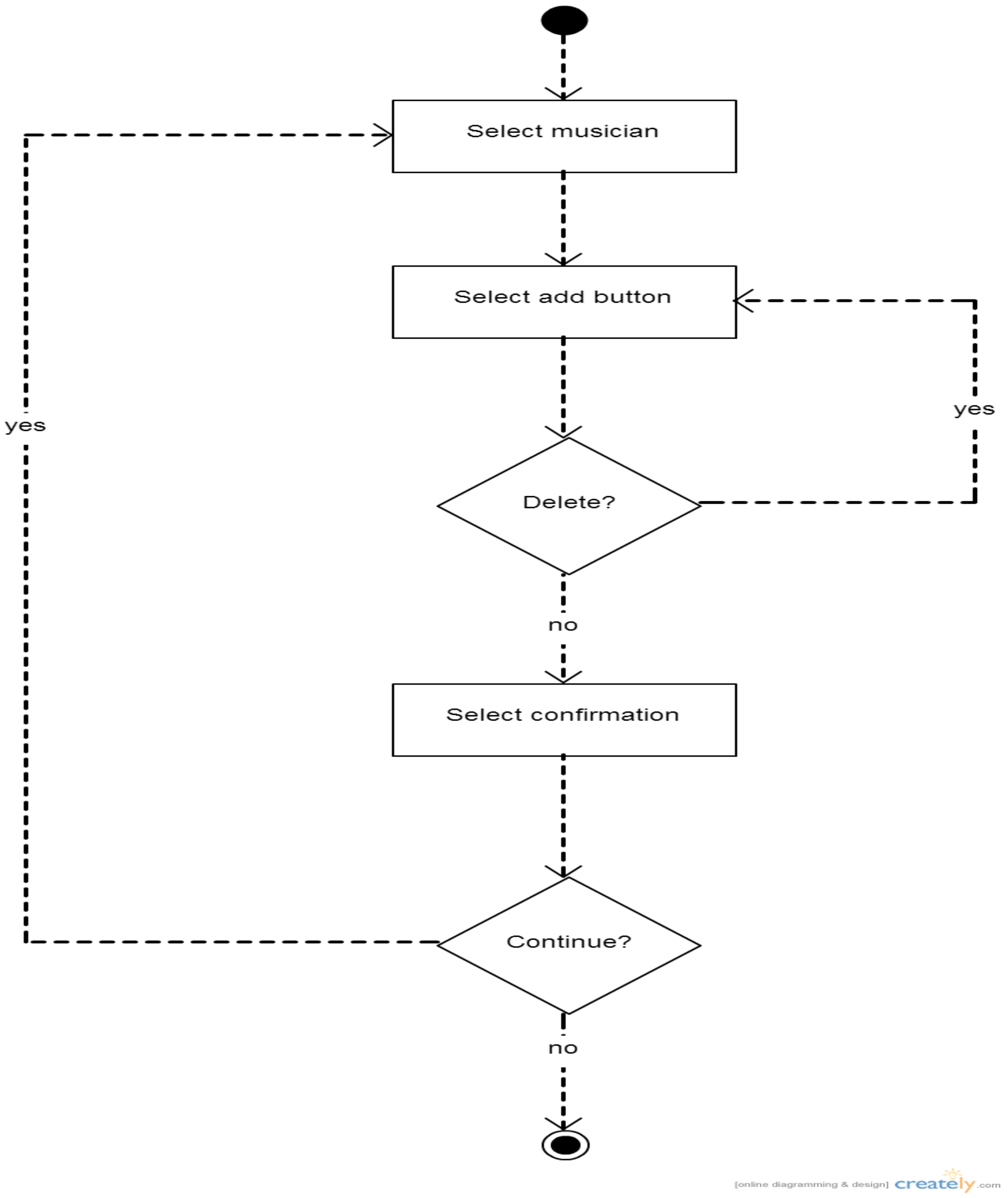


Figure 2.1.2-10: Manage Profile Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Users select the musician pages that they like.
2. The musician pages appear with the playlist of their songs.
3. Users select the songs they like to be add to their profiles with the button included near of the song selected.

4. The selected song also included delete button near it functioning to remove the songs from user’s profile.
5. The confirmation box asks for continue the activity will popped out.
6. If the selection is to continue, the pages will auto reload to the musician selection pages.
7. If the selection is not to continue, the pages will auto reload to the homepage of the user`s profile. They can see the playlist with the selected added songs there.

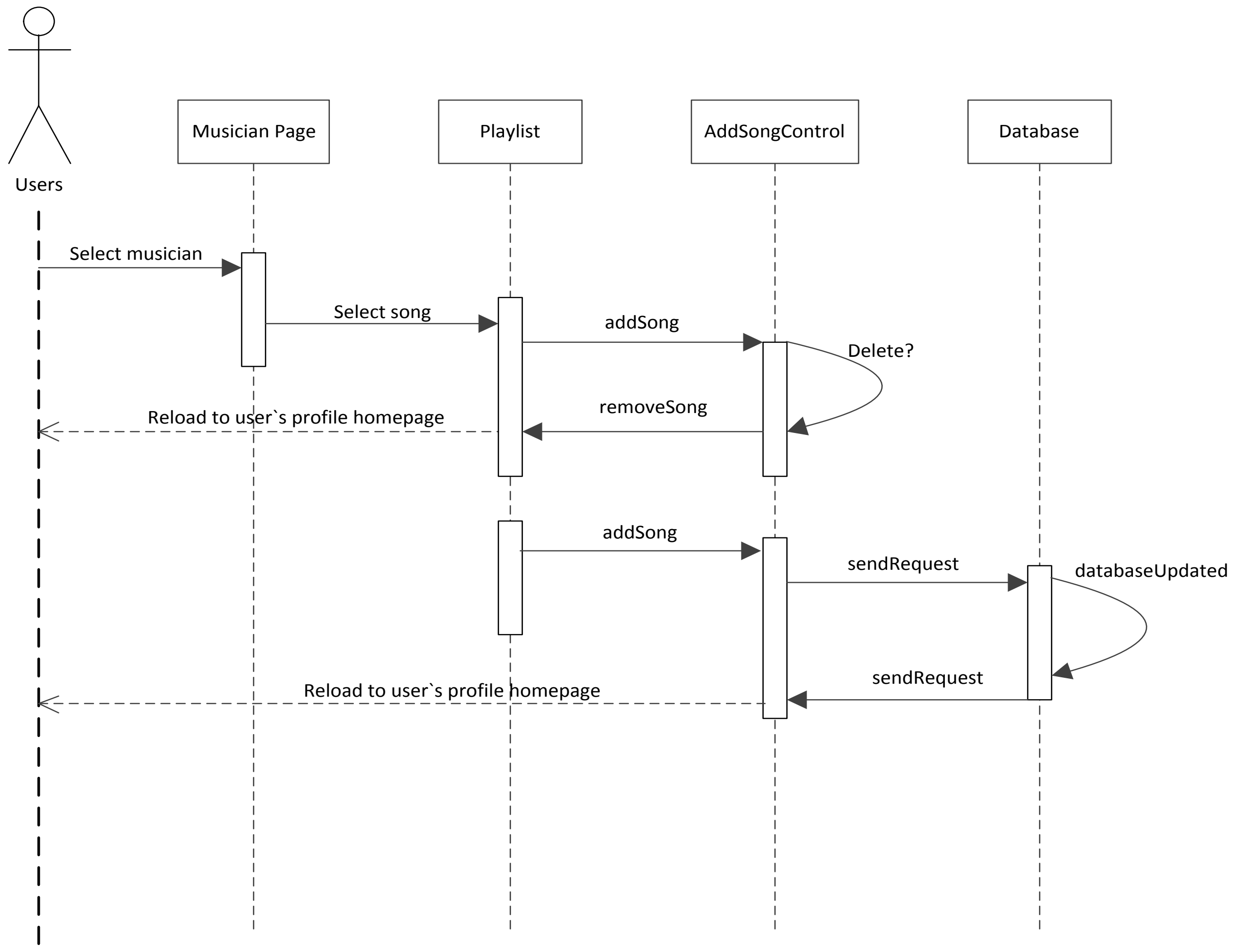


Figure 2.1.2-10: Manage Profile Sequence Diagram

Use Case:	Purchasing
ID:	US-011
Scope:	To purchase materials from the system
Priority:	11/15
Summary:	This use case to allow registered users to purchase materials such as albums, songs and artworks provided by musician. Add to cart system had been implemented in this activity for flexibility of purchasing.
Primary Actor:	Registered User
Supporting Actors:	Admin
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	User must be registered into the system and purchase the material from the musician pages
Trigger:	NA
Normal Flow:	1. Select musician 2. Select store link in the musician page 3. Select items to purchase 4. Item added to cart Loop: Repeat 3 for continue purchase 5. View item Loop: Repeat 5 if agree to remove item from cart
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Purchase succeed
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-9: Purchasing Use Case Description

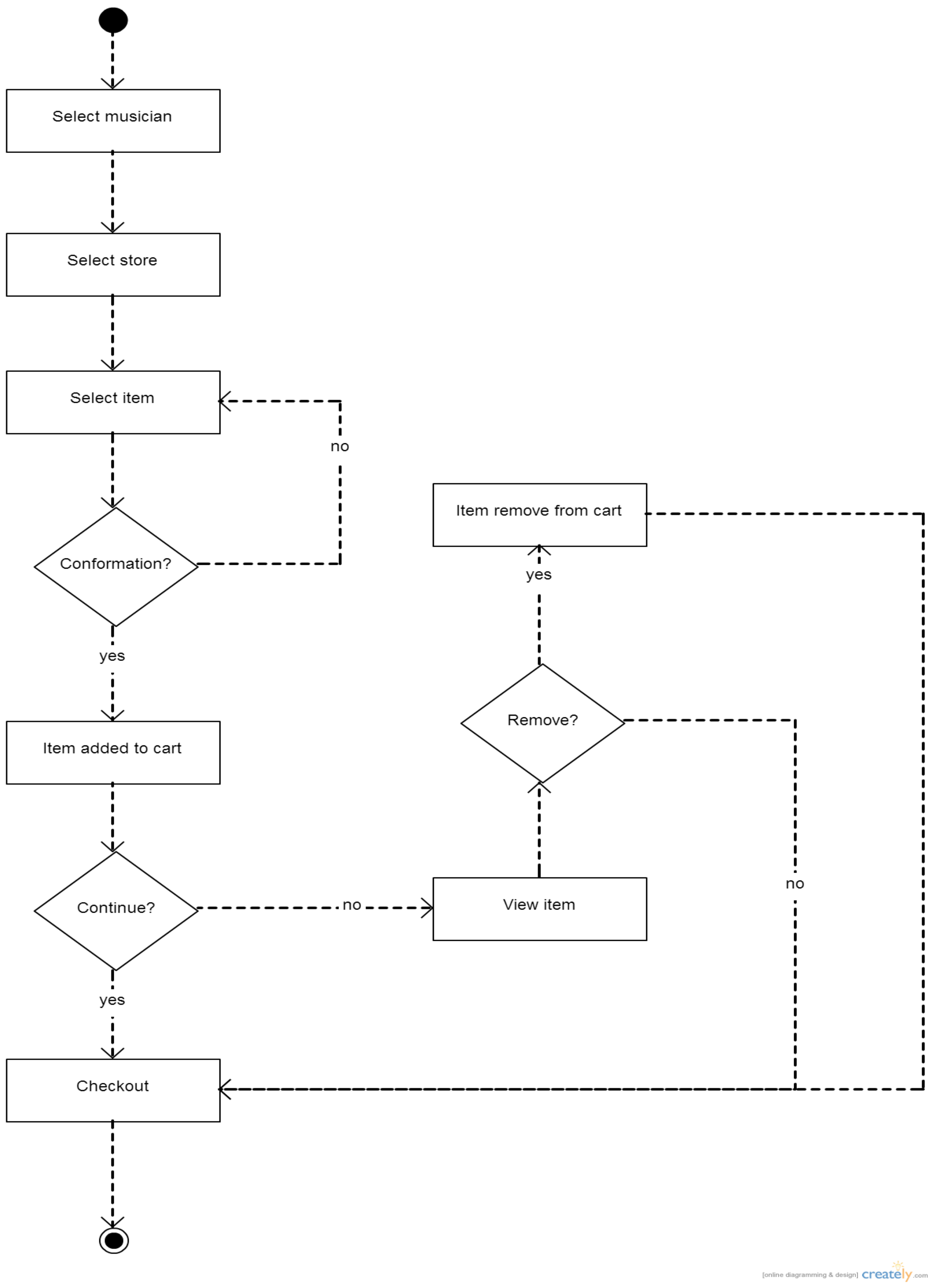


Figure 2.1.2-11: Purchasing Activity Diagram

Activity Diagram Flow Details Description

1. Users select the musician.
2. The link to store will show if that musician items is purchasable. Go to the link to continue purchasing.
3. Select the items to be purchase.
4. Add to cart system implemented here so the desired purchased item will be added to the cart. There are button for add to cart at the items to click.
5. To end the transaction, click on the checkout button. The list of the items added into cart will show after that.
6. To remove the item from the cart, simply click the delete button available at the side of each item on the view list.
7. Page will auto reload to summarize latest purchase.
8. Click the checkout button to continue and here come the payment page which required transaction using PayPal account and credit card like said before in constraints topic.

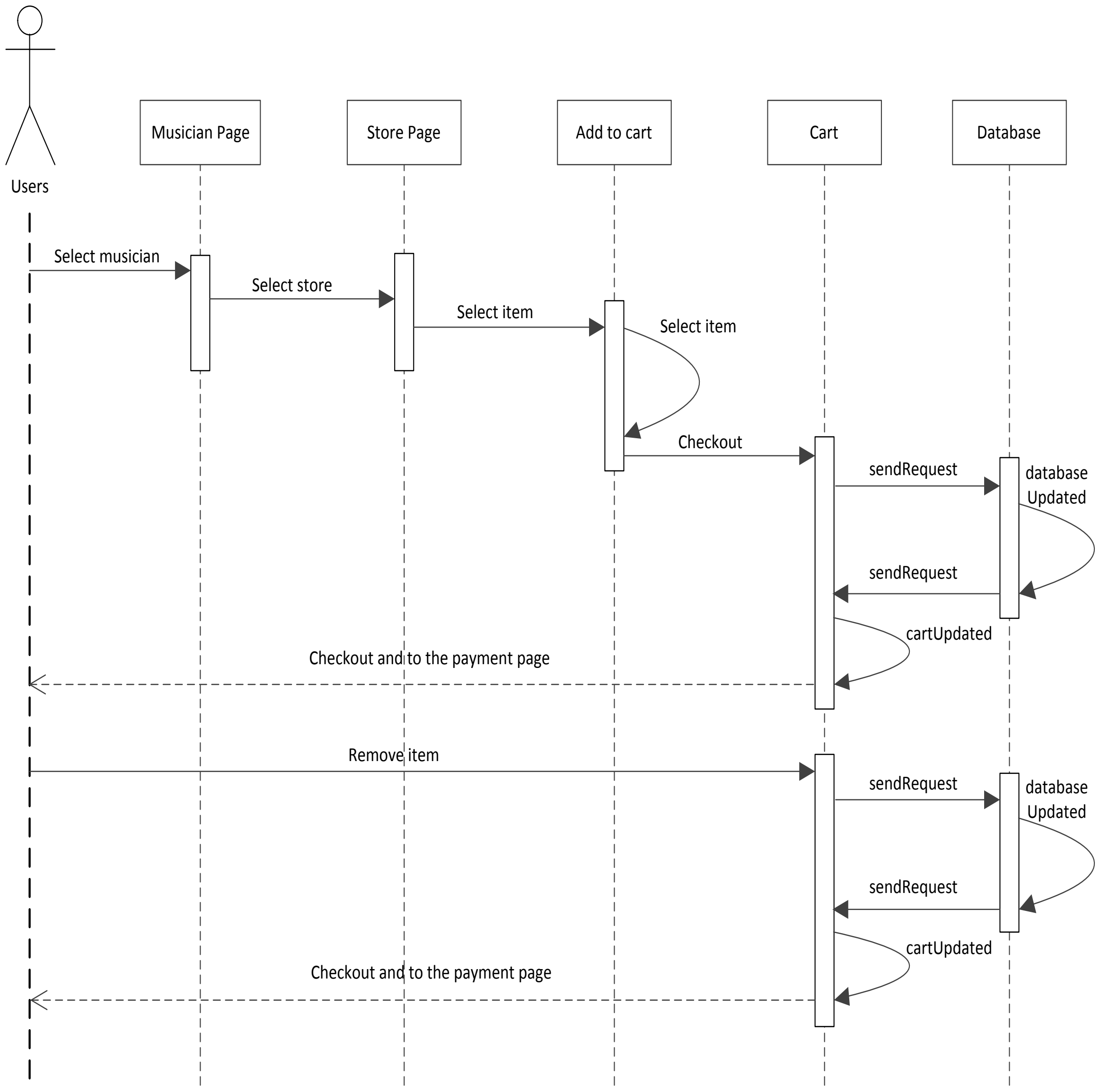


Figure 2.1.2-11: Purchasing Sequence Diagram

Use Case:	News & Events
ID:	US-011
Scope:	To manage news and events activities
Priority:	11/15
Summary:	This use case to allow musician to manage their past and upcoming news and events. It included add, delete and edit the contents.
Primary Actor:	Musician
Supporting Actors:	Registered User
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	User must be musician and registered into the system
Trigger:	NA
Normal Flow:	1. Select news and events 2. Select add new entry 3. Insert required information (date, place, news, events) 4. Select view Loop 1: Repeat 2 if select delete Loop 2: Repeat 3 if select edit
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	News and events added
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-10: News & Events Use Case Description

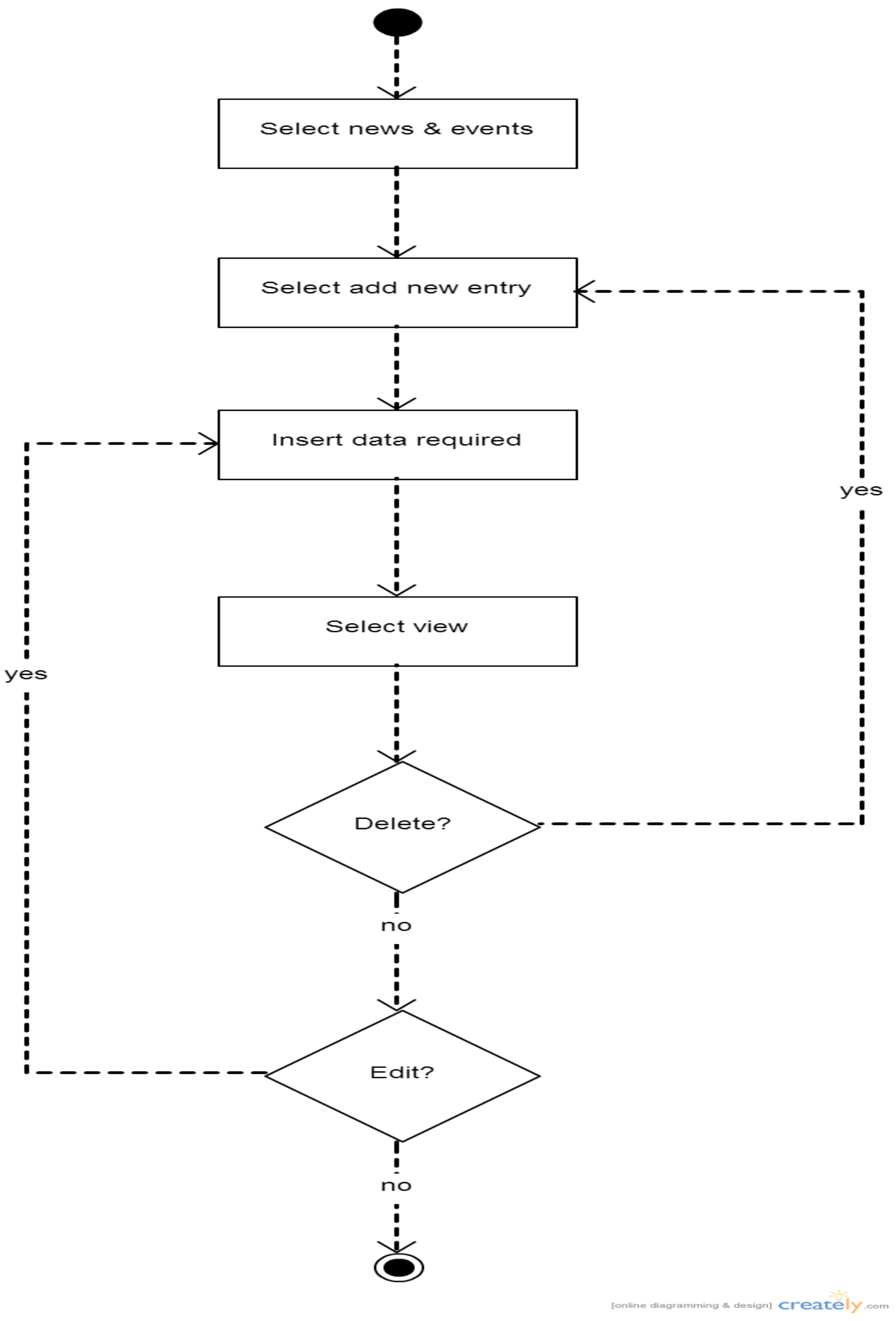


Figure 2.1.2-12: News & Events Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Users select the News & Events.
2. Select add new entry at the News & Events page.
3. Begin typing the information required such as date, place, events description, news description and picture if available.

4. Proceed with button ok at the pages.
5. Select view for the whole news and events picture posted by users.
6. Each of the new and events had two button for delete and edit the contents.

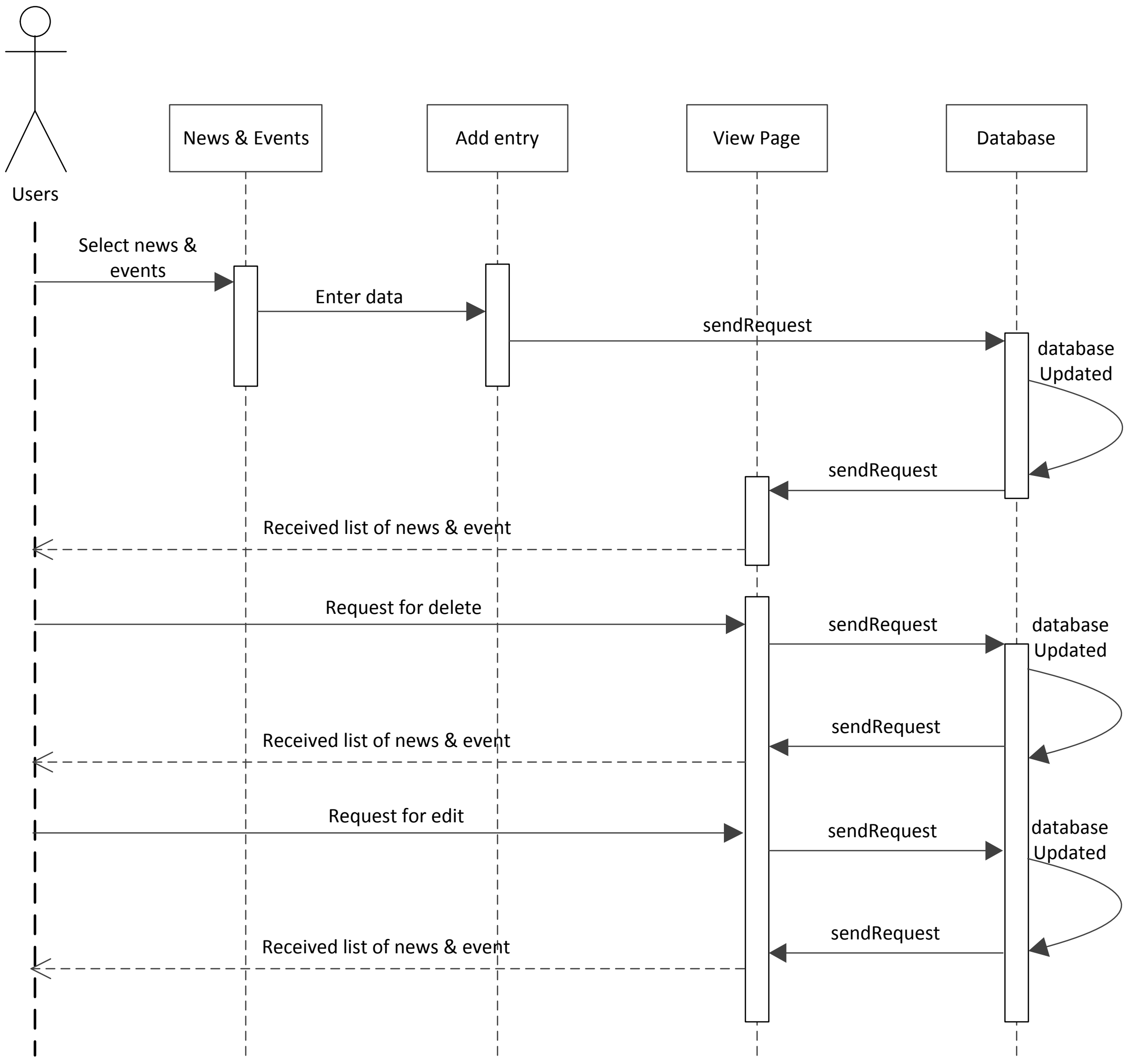


Figure 2.1.2-12: News & Events Sequence Diagram

Use Case Diagram #3

Actor: Admin

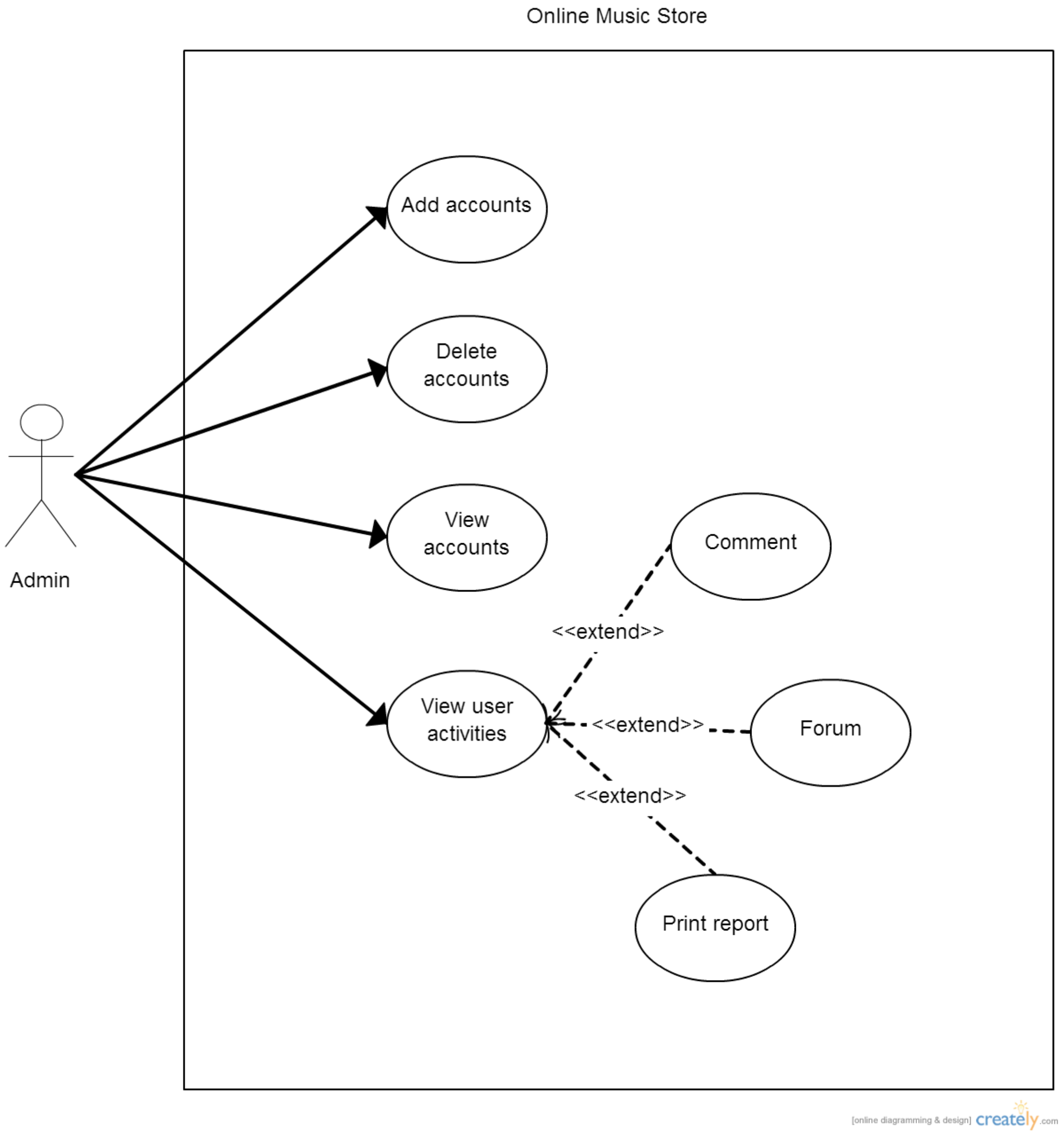


Figure 2.1.2-3: Use Case Diagram for Admin

Use Case:	Add Accounts
ID:	US-012
Scope:	To add accounts into the system
Priority:	12/15
Summary:	This use case to add accounts for the users for using this system.
Primary Actor:	Admin
Supporting Actors:	Users (Registered User + Musician)
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	NA
Trigger:	NA
Normal Flow:	1. Select add new accounts 2. Insert required information 3. Account created
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Account created
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-11: Add Accounts Use Case Description

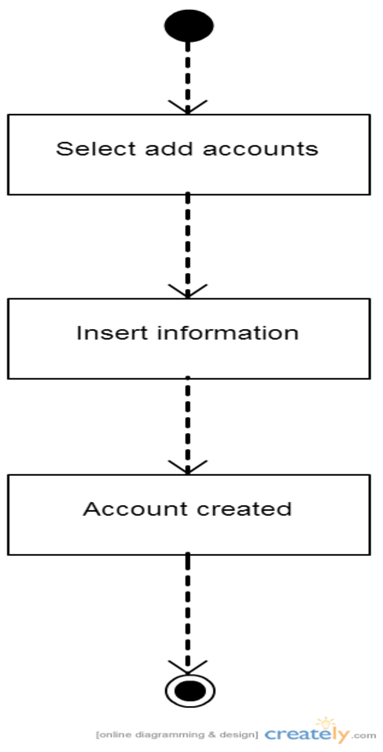


Figure 2.1.2-13: Add Accounts Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Admin select add new accounts.
2. The new accounts page will appeared with the information required to fill in such as name, address, username and password.
3. Click done button for proceed. Information will store in the database.

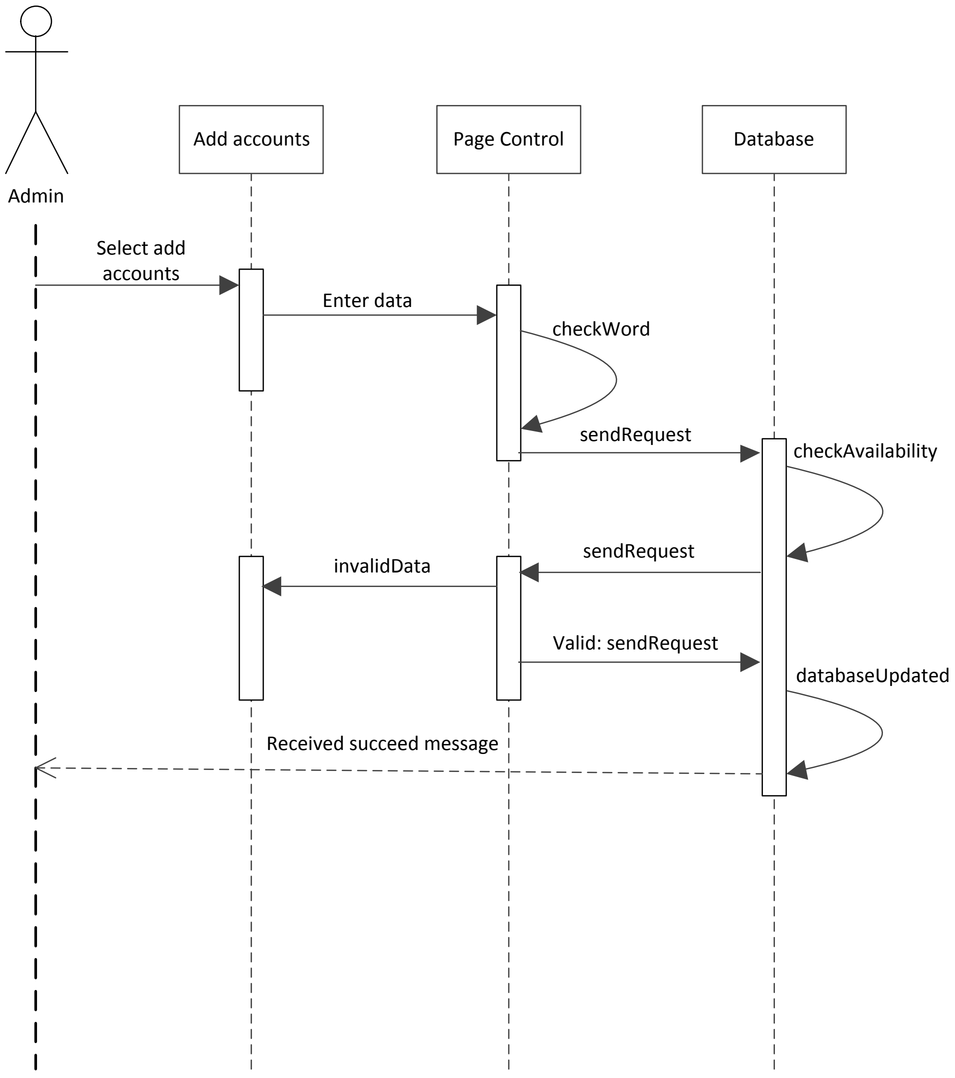


Figure 2.1.2-13: Add Accounts Sequence Diagram

Use Case:	Delete Accounts
ID:	US-013
Scope:	To delete accounts from the system
Priority:	13/15
Summary:	This use case to delete accounts for the users for using this system.
Primary Actor:	Admin
Supporting Actors:	Users (Registered User + Musician)
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	NA
Trigger:	NA
Normal Flow:	1. Select delete account 2. Select view users 3. Account deleted
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Account deleted
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-12: Delete Accounts Use Case Description

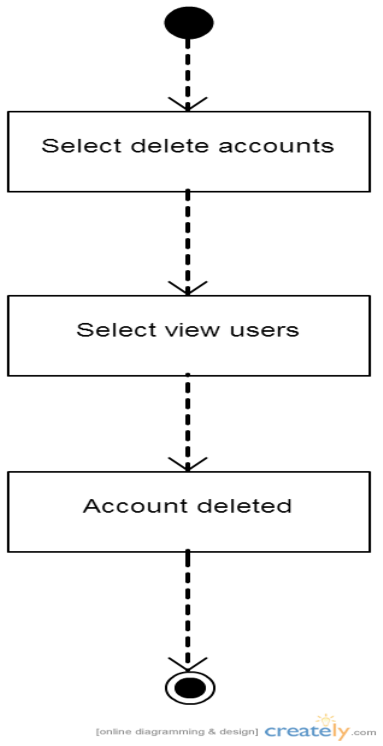


Figure 2.1.2-14: Delete Accounts Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Admin select delete accounts.
2. Select view users.
3. View users will provide list of all users registered in the database. Side of their name included with delete button.
4. Click the delete button then users will delete permanently from database.

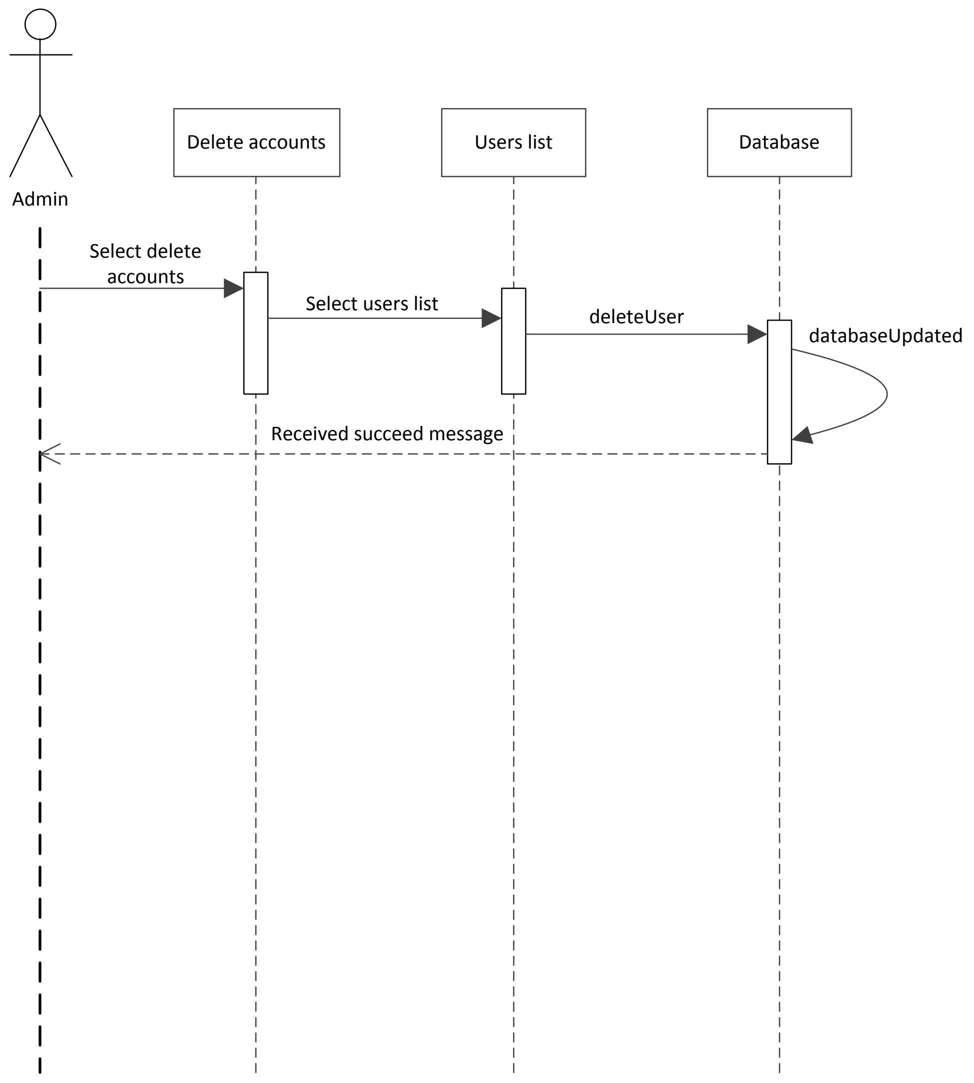


Figure 2.1.2-14: Delete Accounts Sequence Diagram

Use Case:	View Accounts
ID:	US-014
Scope:	To view users accounts in details registered in the system
Priority:	14/15
Summary:	This use case to view the registered user`s account in details
Primary Actor:	Admin
Supporting Actors:	Users (Registered User + Musician)
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	NA
Precondition:	NA
Trigger:	NA
Normal Flow:	1. Select view users 2. Select account details 3. Viewed
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	Account viewed
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-13: View Accounts Use Case Description

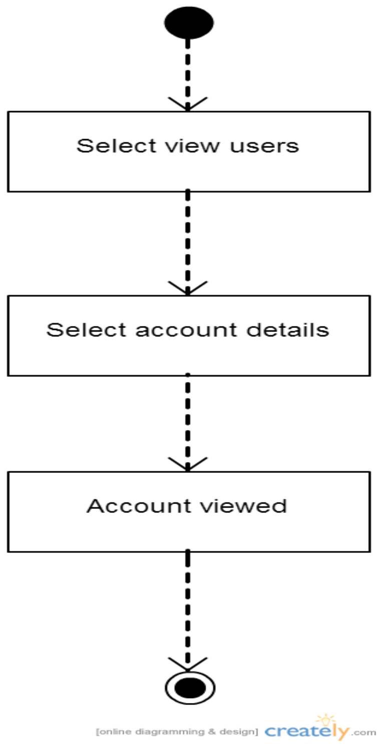


Figure 2.1.2-15: View Accounts Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Admin select view users.
2. View users will appear with the list of all registered users.
3. Each of the users will include account details button.
4. Click for the account details and new page will pop out.
5. The popped out pages shows the details information about the users.

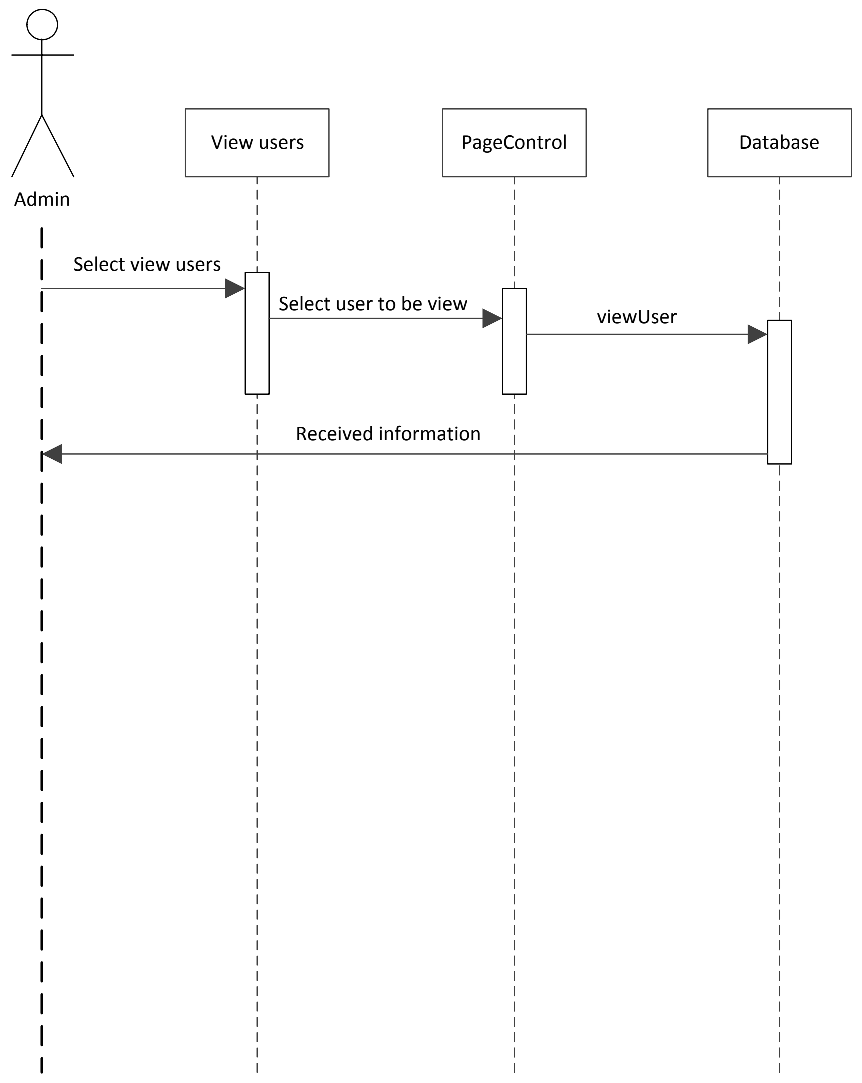


Figure 2.1.2-15: View Accounts Sequence Diagram

Use Case:	View User`s Activities
ID:	US-015
Scope:	To view users activities in the system
Priority:	15/15
Summary:	This use case to view the registered user`s activities through using the system
Primary Actor:	Admin
Supporting Actors:	Users (Registered User + Musician)
Stakeholders:	NA
Generalization:	NA
Include:	NA
Extend:	1. Comment 2. Forum 3. Print Report
Precondition:	NA
Trigger:	NA
Normal Flow:	1. Select view users 2. Select view user`s activities 3. User`s activities viewed
Sub-Flows:	NA
Alternate Flow/ Exceptions:	NA
Post-Condition:	User`s activities viewed
Non-Behavioural Requirements:	NA
Open Issues:	NA
Source:	Requirement Statement
Author:	System Analyst
Revision & Date	Revision 01 24/4/2013

Table 2.1.2-14: View User Activities Use Case Description

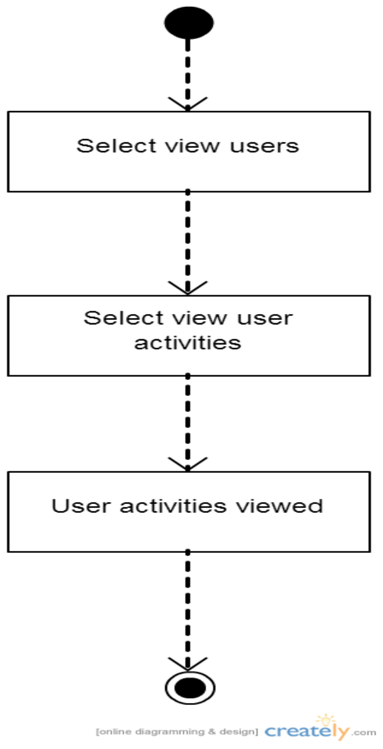

Select view users
Select view user activities
User activities viewed
[online diagramming & design] creately.com

Figure 2.1.2-16: View User Activities Activity Diagram

<u>Activity Diagram Flow Details Description</u>

1. Admin select view users.
2. View users will appear with the list of all registered users.
3. Each of the users will include view user activities button.
4. Click for the view user activities and new page will pop out.
5. The popped out pages shows the details information about the user’s activities.

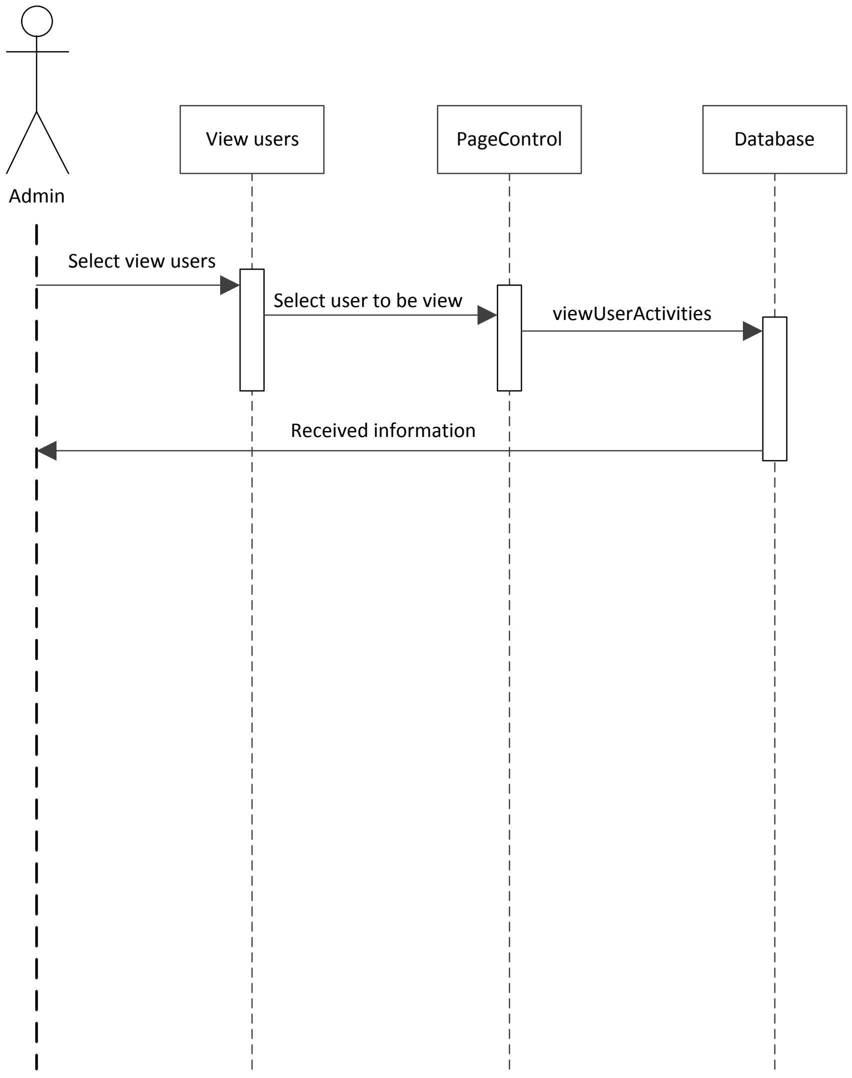


Figure 2.1.2-16: View User Activities Sequence Diagram

ii) Performance requirements

- **Capacity**

  System can communicate with 200 users at the same time. Users do not need to wait for a long time for the transaction of information, since the system is web based system and more than one user can use the system simultaneously.

- **Response Time**

  The system should be able to react and perform well. The respond time of data saving should within 5 seconds. All data will be saved after the confirmation is made, time for saving will not exceed 30 seconds.

- **Error Handling**

  Error handling should be implementing and the application should be able to handle all run time errors. If an error condition occurs, the system should output helpful error message and if not recovery is not possible, it should exit gracefully.

iii) Logical Database Requirements

An entity-relationship (ER) diagram is a specialized graphic that illustrates the relationships between entities in a database. Since the system will be using the database to connect one form to another, the entities must be defined within it characteristics. Every single entity has it Primary Key (PK) which is unique character among all characters in single entity. There is also the entity that has Foreign Key (FK) which is the primary key from other entity. The cardinality must be stated in order to show their relationship between each other.

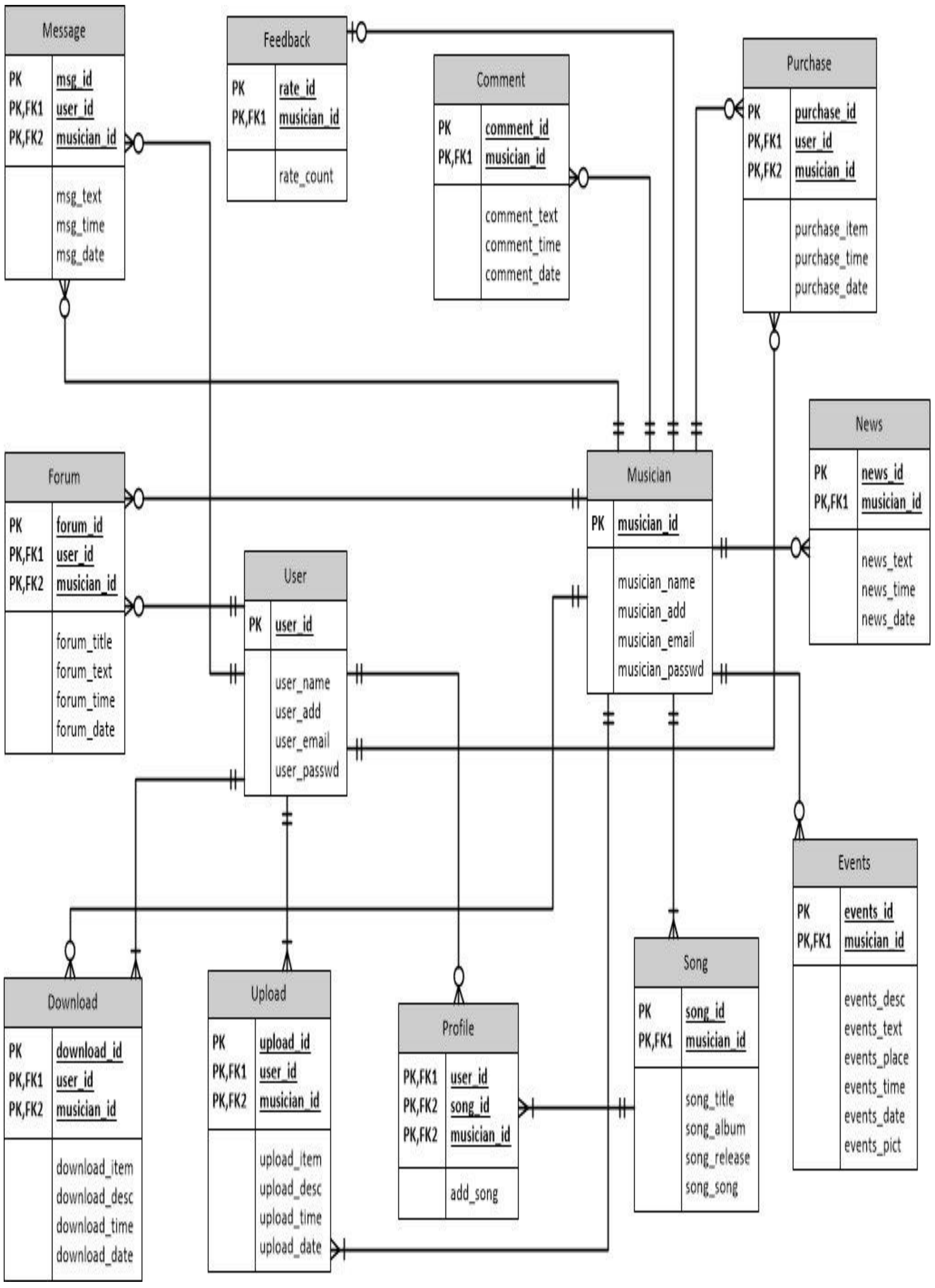


Figure 2.1.2-17: ER diagram

The data descriptions of each of these data entities are as follows:

Table 2.1.2-16: User Entity

Attributes	Definition	Data Type	Details
user_id	Defines user id as auto incremental	INT(200)	Used as unique identifier for each user
user_name	Defines user name	VARCHAR(100)	The name of the user
user_add	Defines user address	VARCHAR(200)	The address of the user
user_email	Defines user email (XXXX@XXXX.XXX)	VARCHAR(50)	The email of the user and used as a username to log in into the system
user_passwd	Defines user password	VARCHAR(20)	Used by a user to log in into the system

Table 2.1.2-17: Musician Entity

Attributes	Definition	Data Type	Details
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each musician
musician_name	Defines musician name	VARCHAR(100)	The name of the musician
muscisian_add	Defines musician address	VARCHAR(200)	The address of the musician
musician_email	Defines musician email (XXXX@XXXX.XXX)	VARCHAR(50)	The email of the musician and

			used as a username to log in into the system
musician_passwd	Defines musician password	VARCHAR(20)	Used by a musician to log in into the system

Table 2.1.2-18: Message Entity

Attributes	Definition	Data Type	Details
msg_id	Defines message id as auto incremental	INT(200)	Used as unique identifier for each message
user_id	Defines user id as auto incremental	INT(200)	Used as unique identifier for each message. Also a foreign keys from user entity
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each message. Also a foreign key from musician entity
msg_text	Defines sent message	VARCHAR(500)	The message that was sent recorded here
msg_time	Defines message time (HH:MM:SS)	TIME()	The time where message that was sent recorded here

msg_date	Defines message date (YYYY-MM-DD)	DATE()	The date where message that was sent recorded here

Table 2.1.2-19: Feedback Entity

Attributes	Definition	Data Type	Details
rate_id	Defines rate id as auto incremental	INT(200)	Used as unique identifier for each rate
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each rate. Also a foreign key from musician entity
rate_count	Define rate counting	INT(200)	To count the rating from user that rated the musician

Table 2.1.2-20: Comment Entity

Attributes	Definition	Data Type	Details
comment_id	Defines comment id as auto incremental	INT(200)	Used as unique identifier for each comment
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each comment. Also a foreign key from

			musician entity
comment_text	Defines sent comment	VARCHAR(500)	The comment that was sent recorded here
comment_time	Defines comment time (HH:MM:SS)	TIME()	The time where comment that was sent recorded here
comment_date	Defines comment date (YYYY-MM-DD)	DATE()	The date where comment that was sent recorded here

Table 2.1.2-21: Purchase Entity

Attributes	Definition	Data Type	Details
purchase_id	Defines purchase id as auto incremental	INT(200)	Used as unique identifier for each purchase
user_id	Defines user id as auto incremental	INT(200)	Used as unique identifier for each purchase. Also a foreign keys from user entity
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each purchase. Also a foreign key from musician entity
purchase_item	Defines purchased item	VARCHAR(500)	The purchases item that was purchased recorded here
purchase_time	Defines purchases	TIME()	The time where

	time (HH:MM:SS)		purchases item that was purchased recorded here
purchase_date	Defines purchases date (YYYY-MM-DD)	DATE()	The date where purchases item that was purchased recorded here

Table 2.1.2-22: Forum Entity

Attributes	Definition	Data Type	Details
forum_id	Defines forum id as auto incremental	INT(200)	Used as unique identifier for each forum
user_id	Defines user id as auto incremental	INT(200)	Used as unique identifier for each forum. Also a foreign keys from user entity
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each forum. Also a foreign key from musician entity
forum_title	Defines forum title	VARCHAR(100)	The title of the forum
forum_text	Defines sent message/comment	VARCHAR(500)	The message/comment that was sent recorded here
forum_time	Defines	TIME()	The time where

	message/comment time (HH:MM:SS)		message/comment that was sent recorded here
forum_date	Defines message/comment date (YYYY-MM-DD)	DATE()	The date where message/comment that was sent recorded here

Table 2.1.2-23: Download Entity

Attributes	Definition	Data Type	Details
download_id	Defines download id as auto incremental	INT(200)	Used as unique identifier for each download
user_id	Defines user id as auto incremental	INT(200)	Used as unique identifier for each download. Also a foreign keys from user entity
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each download. Also a foreign key from musician entity
download_item	Defines downloaded item	MEDIUMBLOB	The downloaded item stored here for future references
download_desc	Defines downloaded item description	VARCHAR(500)	The downloaded item foot notes or description for future references

download_time	Defines downloaded time (HH:MM:SS)	TIME()	The time where downloaded item was download recorded here
download_date	Defines downloaded date (YYYY-MM-DD)	DATE()	The date where downloaded item was download recorded here

Table 2.1.2-24: Upload Entity

Attributes	Definition	Data Type	Details
upload_id	Defines upload id as auto incremental	INT(200)	Used as unique identifier for each upload
user_id	Defines user id as auto incremental	INT(200)	Used as unique identifier for each upload. Also a foreign keys from user entity
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each upload. Also a foreign key from musician entity
upload_item	Defines uploaded item	MEDIUMBLOB	The uploaded item stored here for future references
upload_desc	Defines uploaded item description	VARCHAR(500)	The uploaded item foot notes or description for

			future references
upload_time	Defines uploaded time (HH:MM:SS)	TIME()	The time where uploaded item was upload recorded here
upload_date	Defines uploaded date (YYYY-MM-DD)	DATE()	The date where uploaded item was upload recorded here

Table 2.1.2-25: Song Entity

Attributes	Definition	Data Type	Details
song_id	Defines song id as auto incremental	INT(200)	Used as unique identifier for each song
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each song. Also a foreign key from musician entity
song_title	Defines song title	VARCHAR(100)	The title for the song
song_album	Defines song album	VARCHAR(100)	The album for the song
song_release	Define song release date (YYYY-MM-DD)	DATE ()	The released date for the song
song_song	Define uploaded song	MEDIUMBLOB	The uploaded song stored here

Table 2.1.2-26: Profile Entity

Attributes	Definition	Data Type	Details
user_id	Defines user id as auto incremental	INT(200)	Used as unique identifier for each profile. Also a foreign keys from user entity
song_id	Defines song id as auto incremental	INT(200)	Used as unique identifier for each profile. Also a foreign keys form the song entity
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each profile. Also a foreign key from musician entity
add_song	Defines added song	MEDIUMBLOB	The added song stored here

Table 2.1.2-27: Events Entity

Attributes	Definition	Data Type	Details
events_id	Defines event id as auto incremental	INT(200)	Used as unique identifier for each event
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each event. Also a

			foreign key from musician entity
events_desc	Defines event description	VARCHAR(500)	The event foot notes or description for future references
events_text	Defines event message/comment	VARCHAR(500)	The event message/comment recorded here
events_time	Defines event time (HH:MM:SS)	TIME()	The time where event held recorded here
events_date	Defines event date (YYYY-MM-DD)	DATE()	The date where event held recorded here
events_pict	Defines event picture	MEDIUMBLOB	The picture of the event stored here if available

Table 2.1.2-28: News Entity

Attributes	Definition	Data Type	Details
news_id	Defines news id as auto incremental	INT(200)	Used as unique identifier for each news
musician_id	Defines musician id as auto incremental	INT(200)	Used as unique identifier for each news. Also a foreign key from musician entity
news_text	Defines news message/comment	VARCHAR(500)	The news message/comment

			recorded here
news_time	Defines news time (HH:MM:SS)	TIME()	The time where news held recorded here
news_date	Defines news date (YYYY-MM-DD)	DATE()	The date where news held recorded here

2.1.3 Non-functional Requirements

Attributes	Description
Reliability	• Mean Time Between Failures: 30 days. • Mean Time To Repair: 1 hour. • Accuracy: 100%.
Availability	• This system has to be available 24 hours / days. Every week typically on Monday morning around 3am to 5am, the system will be shut down for maintenance and updating • The system will recover as soon as possible while it has problem ▪ Approximate bugs recover for critical – 2 days. ▪ Approximate bugs recover for half-critical – 1 day. ▪ Approximate bugs recover for non-critical – 4 hour.
Security	• All network transactions that involve financial information or personally identifiable information shall be encrypted. • Any errors in purchasing transaction can be claimed only if users have proof regarding the transaction. • The system shall not allow confidential data stored in the

Attributes	Description
	system's database to be accessed, whether directly or indirectly.
Maintainability	• This system shall permit the swapping and upgrade of hardware without down time. • This system shall permit the upgrade of software without down time. • The Mean Time To Fix shall not exceed one person day. • The user will be able to reset all options and all stored user variables to default settings.
Portability	• This system compatible with Window operating system such as Window XP, Window Vista and Window 7. In addition, this system not compatible with Macintosh operating system (Mac OS) • This system can comply with personal computer (Desktop and Laptop) and not comply with mobile technology • This system only can only be running on the Mozilla Firefox, Opera, Google Chrome and Internet Explorer browser

Table 2.1.3-1: Software System Attributes

2.2 Preliminary Design

2.2.1 System Overview

Three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. At first glance, the three tiers may seem similar to the model-view-controller (MVC) concept; however, topologically they are different. A fundamental rule in three tier architecture is the client tier never communicates directly with the data tier; in a three-tier model all communication must pass through the middle tier. Conceptually the three-tier architecture is linear. However, the MVC architecture is triangular: the view sends updates to the controller, the controller updates the model, and the view gets updated directly from the model

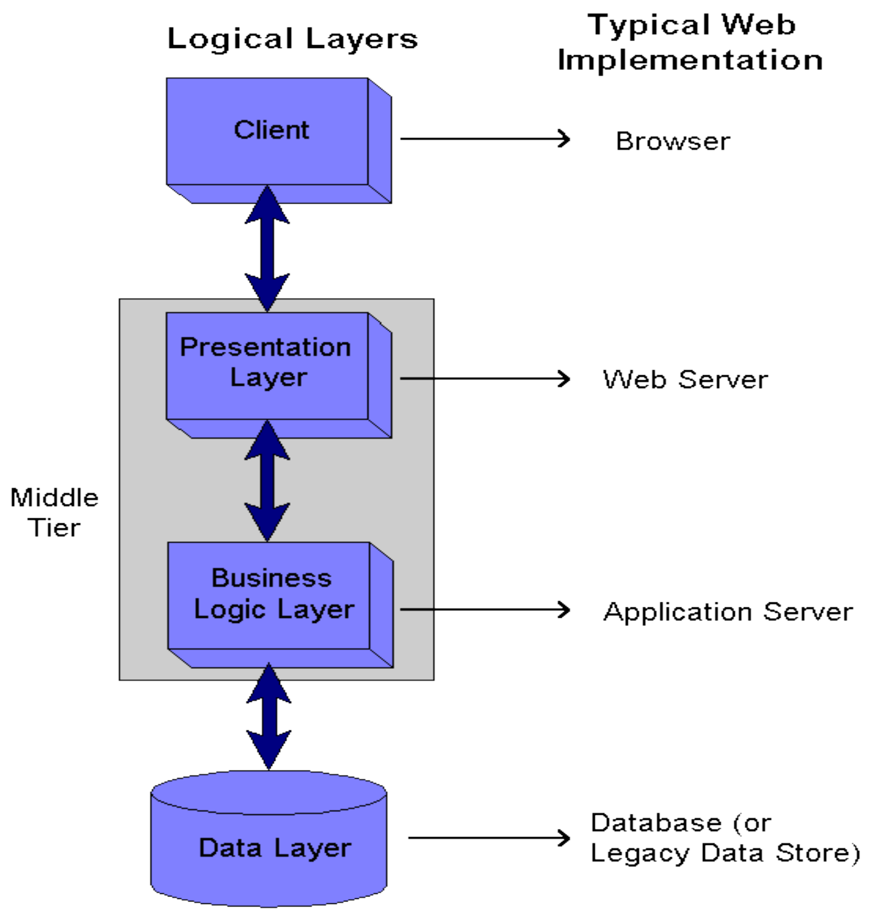


Figure 2.2.1-1: Three Tier Architecture

2.2.2 System Architecture

This paragraph identifies the internal organizational structure of the system. The relationship among system subsystem will be described.

i) Static Organization

In this section, it'll describe all the packages available in the system. There are five packages in this system which is Admin, Unregistered User, Global User, Registered User and Musician.

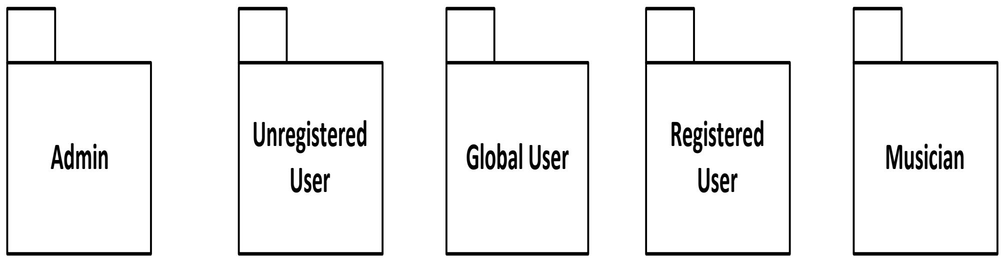


Figure 2.2.2-1: Online Music Store Packages

This section describes the details description for each subsystem/package

1. **Admin**
   Admin package determine the system admin function such as control the process, manage and its interface stored in one package. This package consists of the following classes/unit.

   a) addaccount_page class
   b) addaccount_process class
   c) deleteaccount_page class
   d) deleteaccount_process class
   e) viewuser_page class
   f) viewuser_process class
   g) viewaccount_page class
   h) viewaccount_process class

i) viewuseractivity_page class
j) viewuseractivity_process class

2. **Unregistered User**

   Unregistered User package determine the unregistered user function such as control the process, manage and its interface stored in one package. This package consists of the following classes/unit.

   a) register_page class
   b) register_process class

3. **Global User**

   Registered User package determine the global user function such as control the process, manage and its interface stored in one package. To be fully understand about global user package, we can say with the generalization of the users into two parts which is Registered User and Musician, somehow they have several function that shared together. So with the global user package classifying, the function do not have to repeat, just call it from this package. This package consists of the following classes/unit.

   a) login_page class
   b) login_ process class
   c) logout_process class
   d) messaging_page class
   e) messaging_process class
   f) forum_page class
   g) forum_process class
   h) upload_page class
   i) upload_process class

4. **Registered User**

   Registered User package determine the registered user function such as control the process, manage and its interface stored in one package. This package consists of the following classes/unit.

   a) addsong_process class
   b) removesong_process class
   c) rating_process class
   d) comment_process class
   e) download_process class
   f) songstreaming_process class
   g) purchasing_page class
   h) purchasing_process class

5. **Musician**

   Musician package determine the musician function such as control the process, manage and its interface stored in one package. This package consists of the following classes/unit.

   a) newsevents_page class
   b) newsevents_process class

ii) External Interfaces

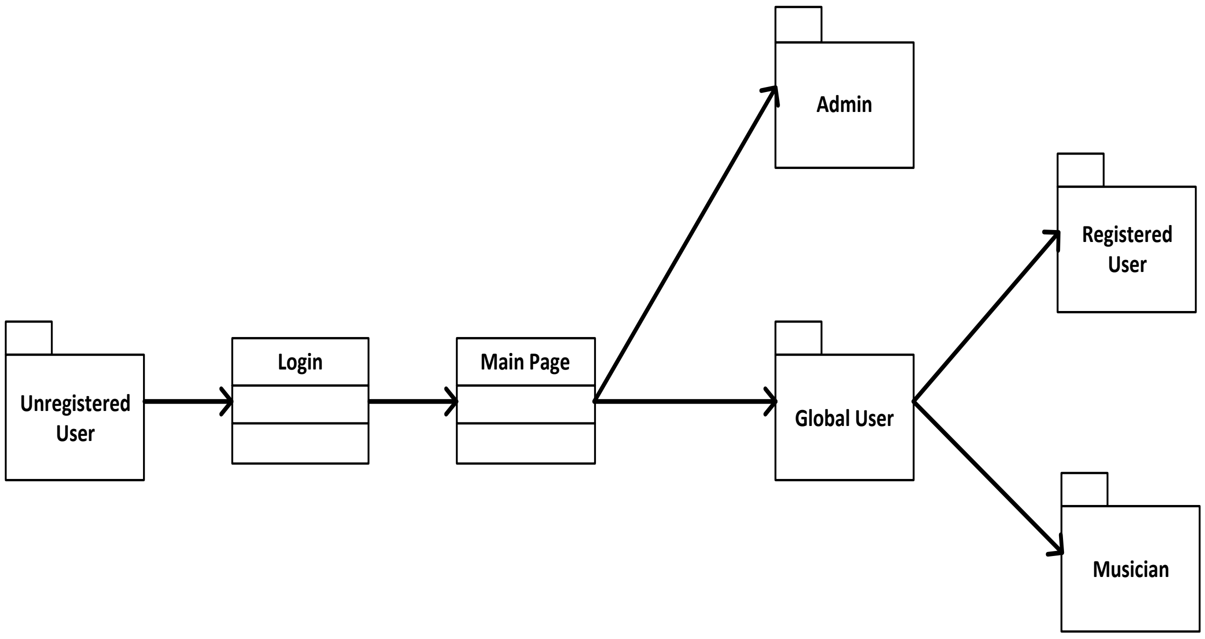


Figure 2.2.2-2: External Interfaces

2.2.3 System States and Modes

This section describes states diagrams for Online Music Store (OMS). Figure below show the state diagram for Unregistered User subsystem.

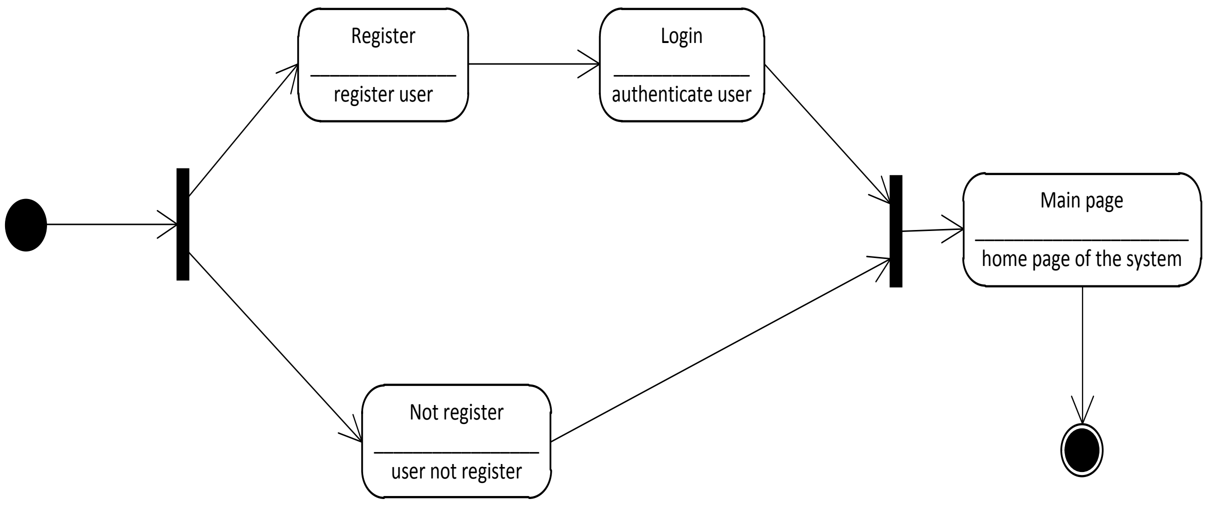
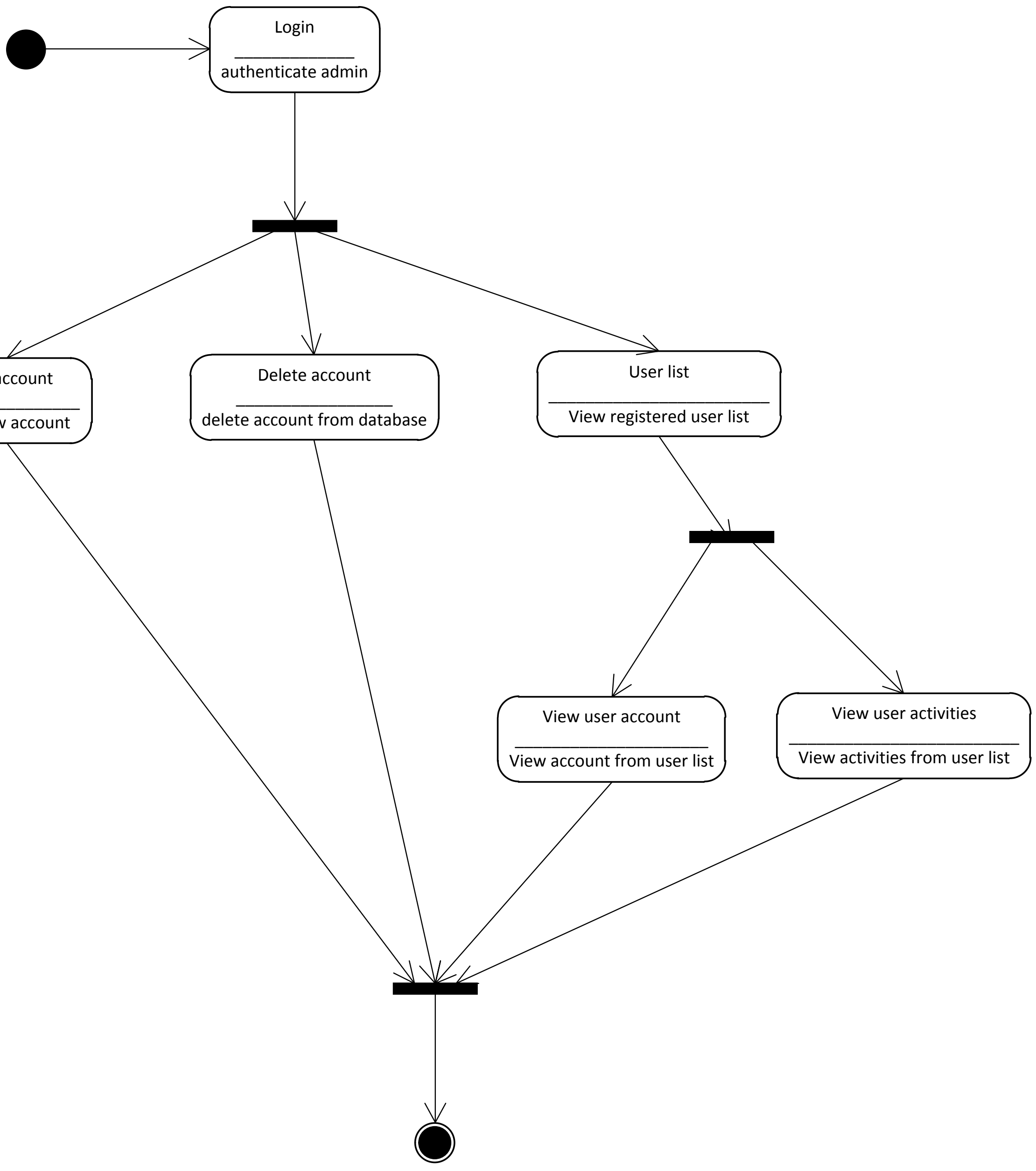


Figure 2.2.3-1: Unregistered User Subsystem

Figure below show the state diagram for Admin subsystem.



Login
authenticate admin
Add account
add new account
Delete account
delete account from database
User list
View registered user list
View user account
View account from user list
View user activities
View activities from user list

Figure 2.2.3-2: Admin Subsystem

Figure below show the state diagram for Global User combined with Registered User and Musician subsystem.

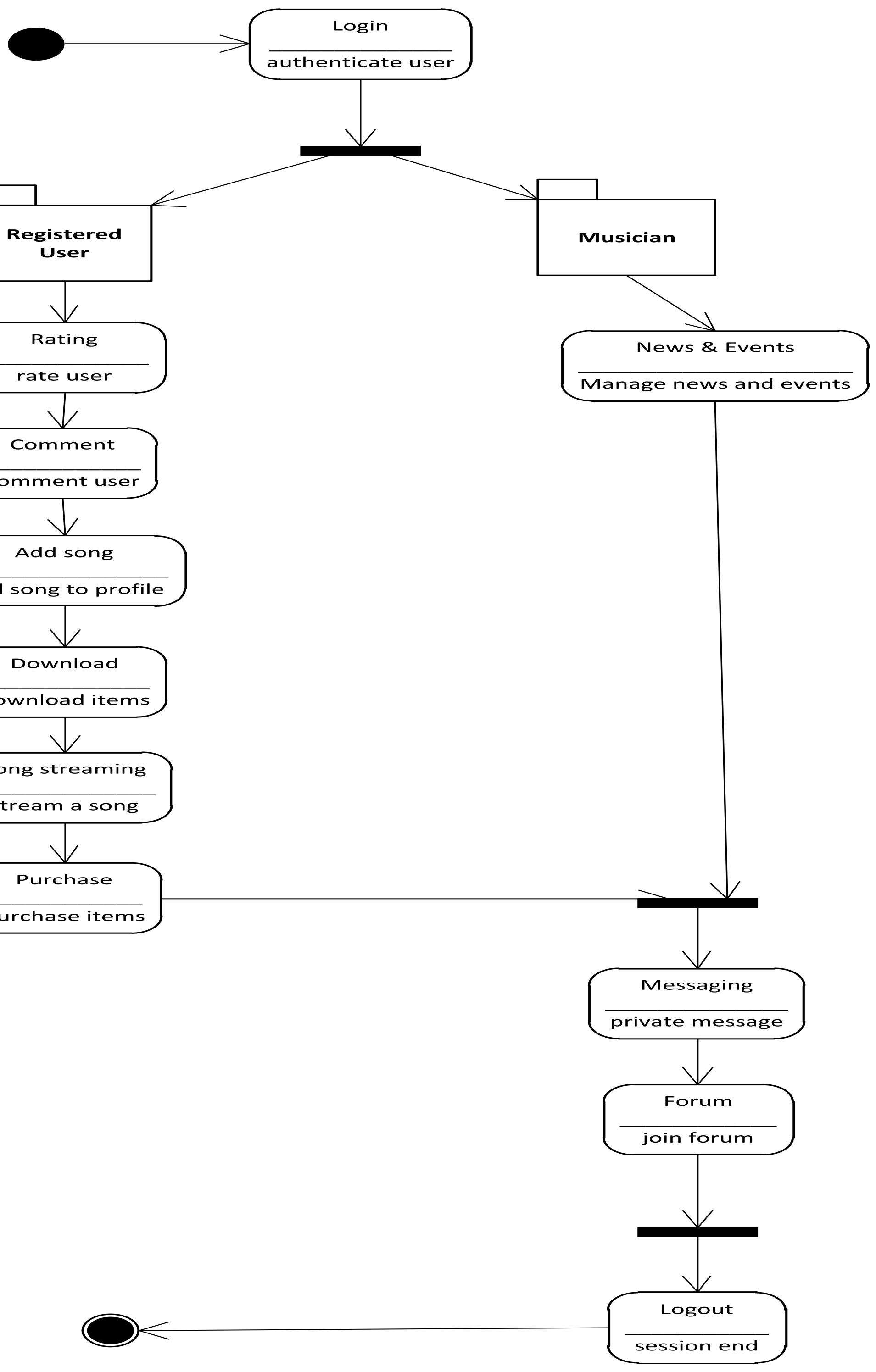


Figure 2.2.3-3: Combined Global User Subsystem

2.3 System Design Description

Online Music Store (OMS) consists of three modules which are Users, Musician/Artists and Admin. Currently, OMS not using any framework, and resulting from this, the classes of the three modules combine together in one subsystem. Categorization of the modules is the best way to elaborate this project`s system design descriptions.

2.3.1 Global User Subsystem

Figure below shows the classes which are associated with Global User Subsystem via class diagram.

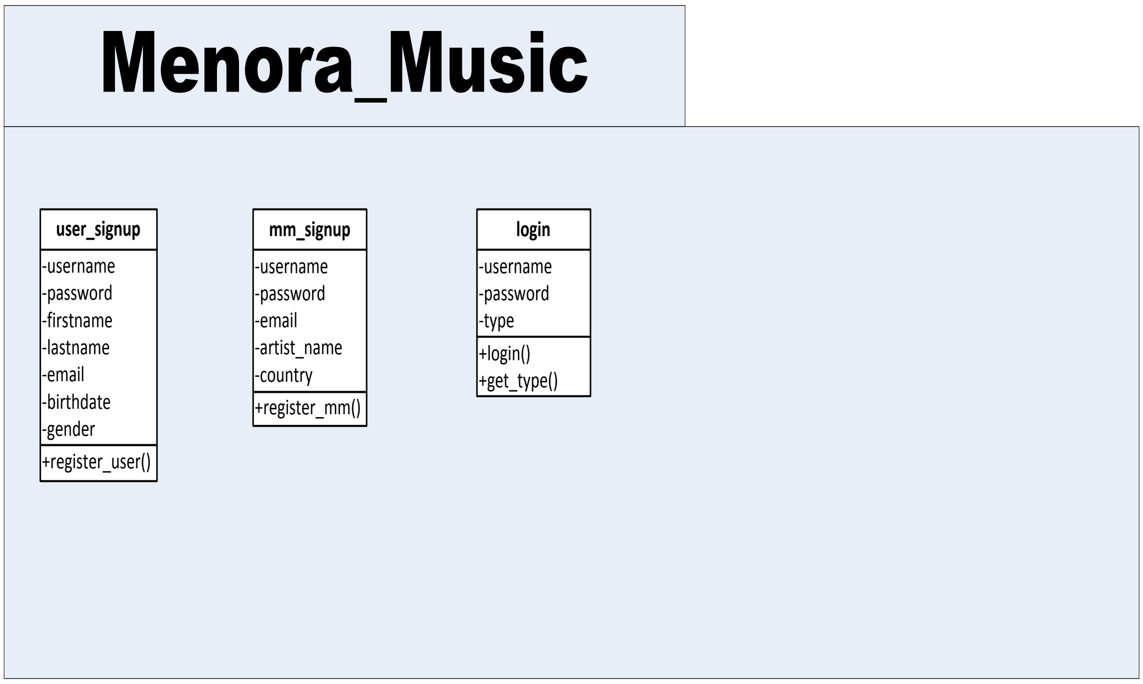
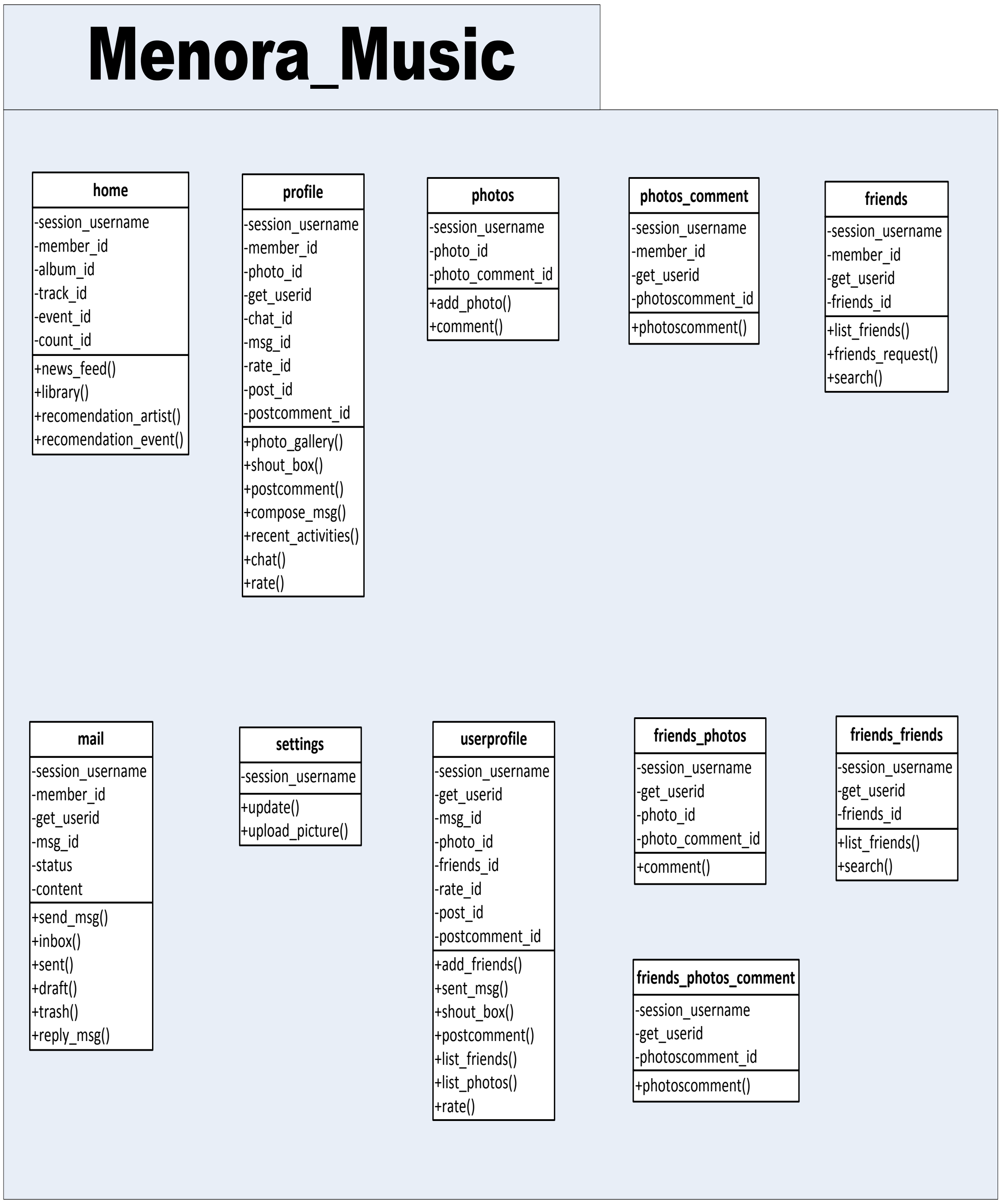


Figure 2.3.1-1: Global Users Subsystem Class Diagram

2.3.2 Users Module

Figure below shows the classes which are associated with Users Module via class diagram.



2.3.2-1: Users Module Class Diagram

2.3.3 Musician/Artists Module

Figure below shows the classes which are associated with Musician/Artists Module via class diagram.

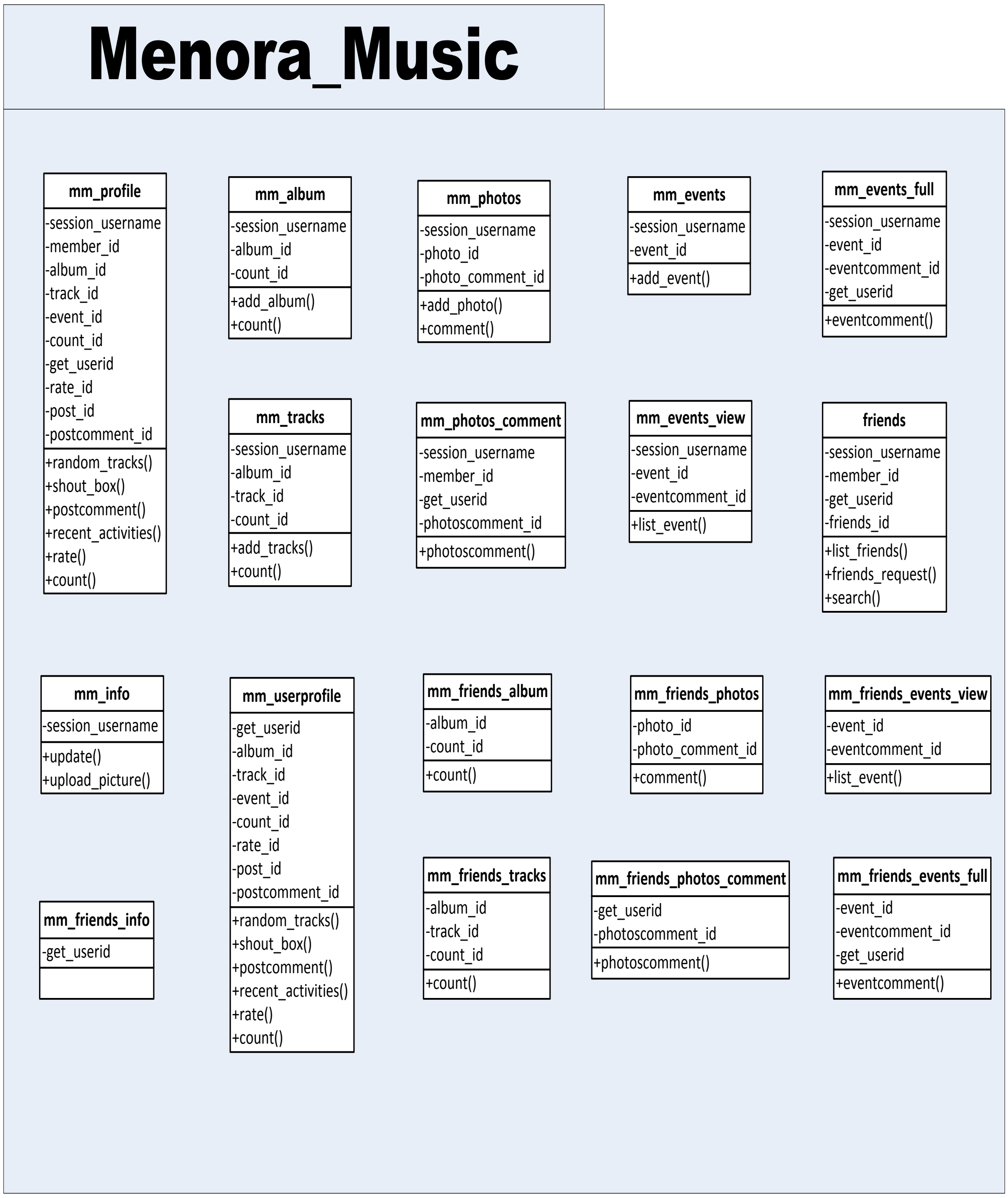


Figure 2.3.3-1: Musician/Artists Module Class Diagram

2.3.4 Admin Module

Figure below shows the classes which are associated with Admin Module via class diagram.

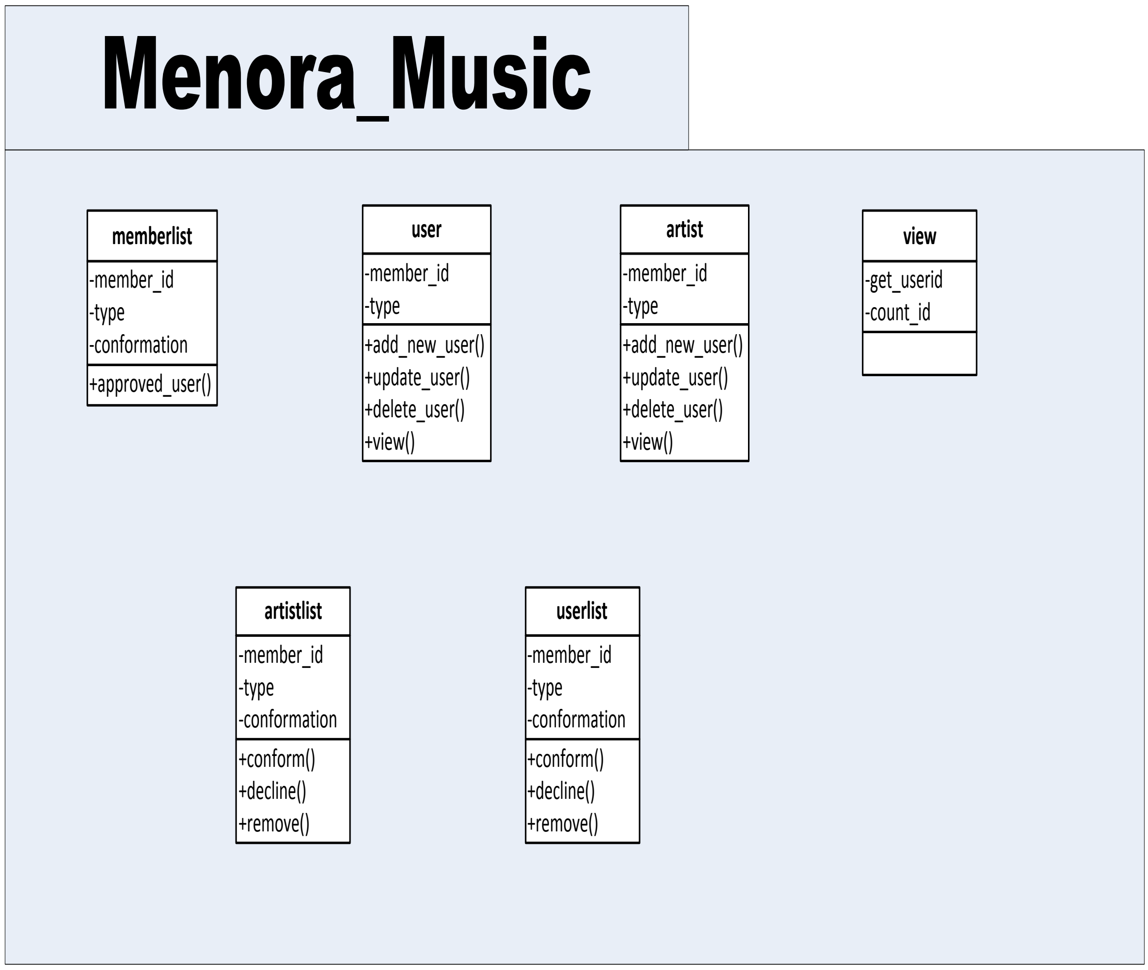


Figure 2.3.4-1: Admin Module Class Diagram

2.4 Detailed Design

This section divided into the following paragraphs and subparagraphs to describe the detailed design.

2.4.1 Global User Subsystem

Figure below shows global user subsystem with its relationship among the other subsystem classes.

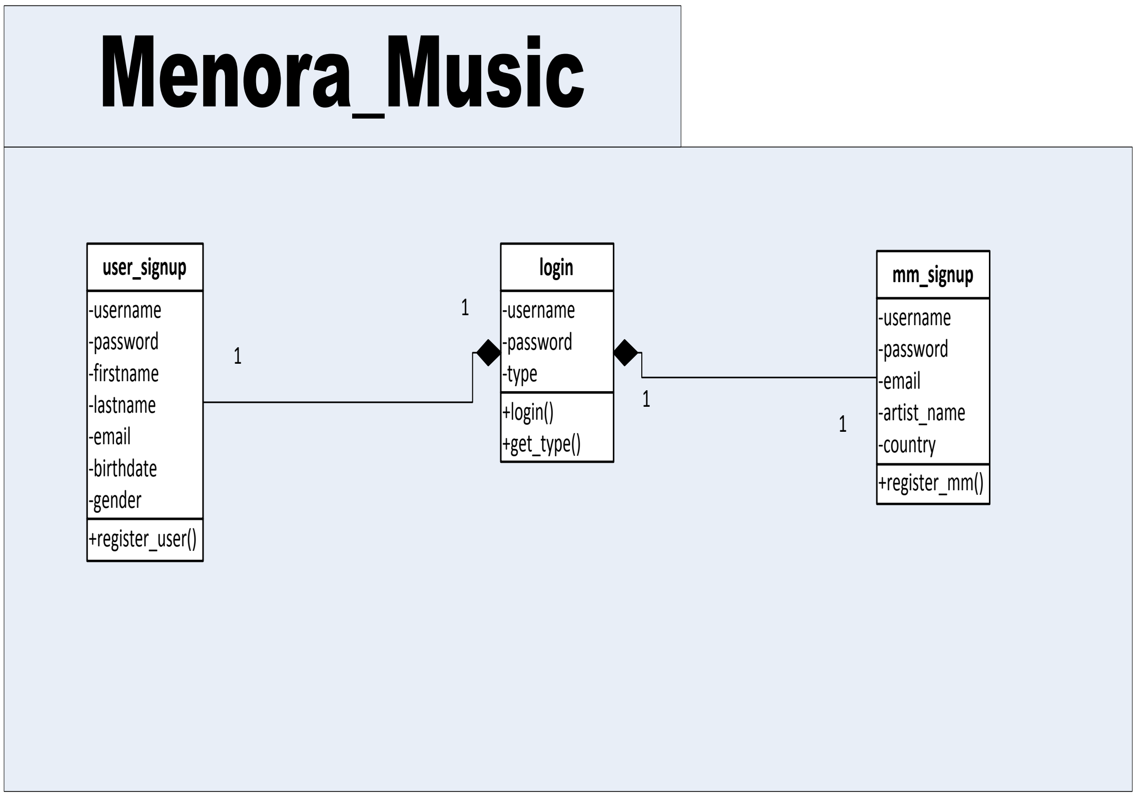


Figure 2.4.1-1: Global User Subsystem Relationship

- **Global User Subsystem Class**

Below is the details explanation about the classes available in the Global User subsystem.

i) user_signup Design

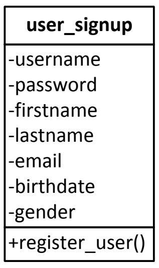


Figure 2.4.1-2: user_signup Class

a) Input/Output Data Elements

Input: username, password, firstname, lastname, email, birthdate, gender
Output: User registered

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	To add a new account into the database so they can login into the system.		
Attributes	:	username	:	Varchar
	:	password	:	Varchar
	:	firstname	:	Varchar
	:	lastname	:	Varchar
	:	email	:	Varchar
	:	birthdate	:	Varchar
	:	gender	:	Varchar
Methods	:	register_user()	:	Insert the input into the database

Table 2.4.1-1: user_signup Details

c) Algorithm

Method: register_user()

```
BEGIN

    Get input from button ($_POST)
    Set type = user for categorization
    Check if username if available
    If username not available
    Then insert the data into the database

END
```

ii) login Design

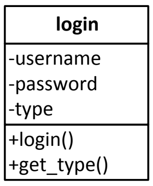


Figure 2.4.1-3: login Class

a) Input/Output Data Elements

Input: username, password, type

Output: User enter the system

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	Select the requested input user from database and if available the user can enter the system		
Attributes	:	username	:	Varchar

	:	password	:	Varchar
	:	type	:	Varchar
Methods	:	login()	:	Select the database with requested input
	:	get_type()	:	Get the requested type

Table 2.4.1-2: login Details

c) Algorithm

Method: login()

```
BEGIN

    Get input from button ($_POST)
    If type is equal to User (get_type())
    Select the requested input then direct user to User Homepage
    If type is equal to Artist (get_type())
    Select the requested input then direct user to Artist Homepage
    If type is equal to Admin (get_type())
    Select the requested input then direct user to Admin Homepage

END
```

iii) mm_signup Design

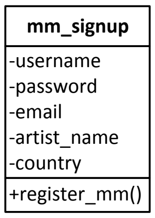


Figure 2.4.1-4: mm_signup Class

a) Input/Output Data Elements

Input: username, password, email, artist_name, country

Output: User registered

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	To add a new account into the database so they can login into the system.		
Attributes	:	username	:	Varchar
	:	password	:	Varchar
	:	email	:	Varchar
	:	artist_name	:	Varchar
	:	country	:	Varchar
Methods	:	register_mm()	:	Insert the input into the database

Table 2.4.1-3: mm_signup Details

c) Algorithm

Method: register_mm()

```
BEGIN

    Get input from button ($_POST)
    Set type = user for categorization
    Check if username if available
    If username not available
    Then insert the data into the database

END
```

2.4.2 Users Module

Figure below shows users module with its relationship among the other subsystem classes.

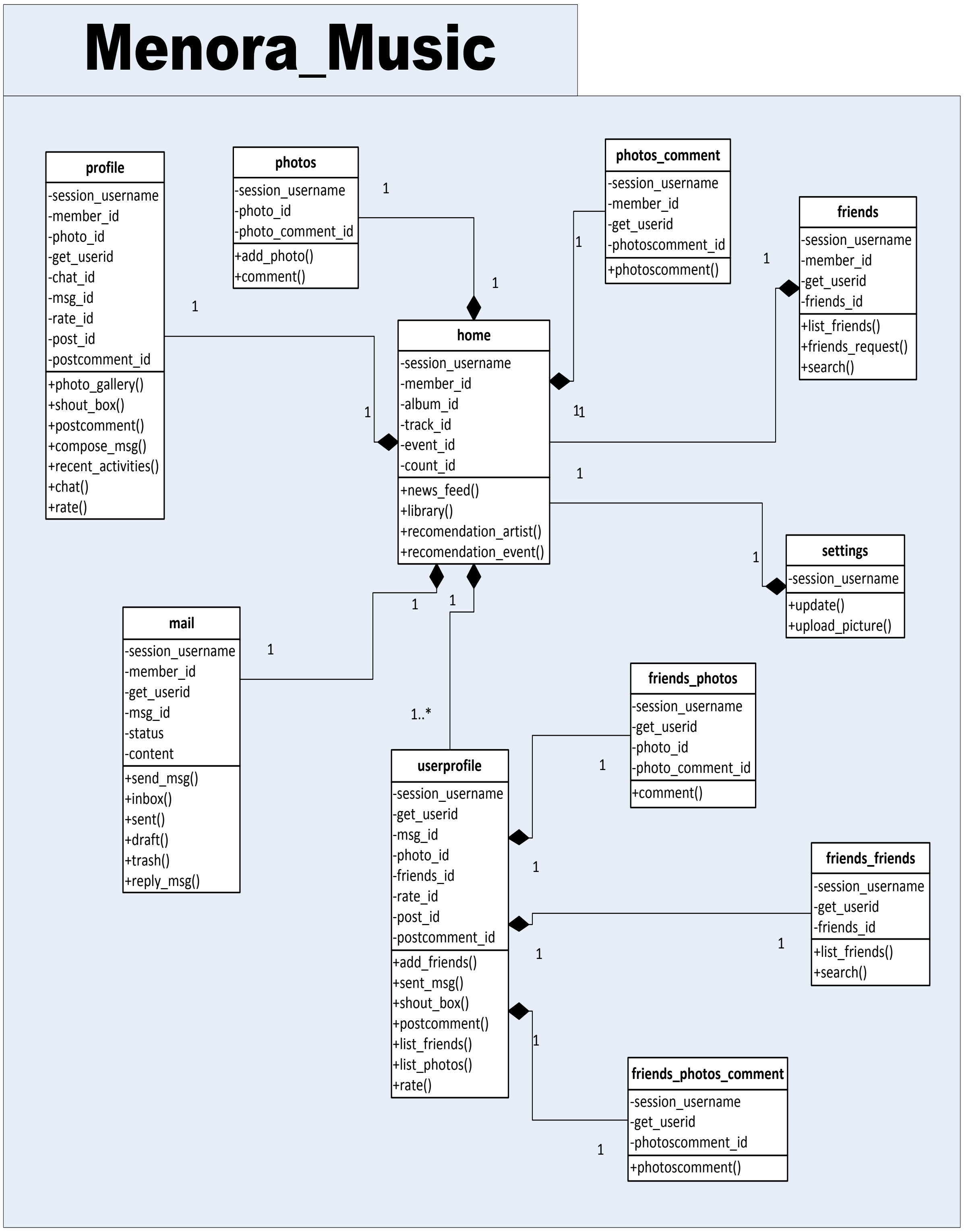


Figure 2.4.2-1: Users Module Relationship

- **Users Module Class**

  Below is the details explanation about the classes available in the Users Module.

i) home Design

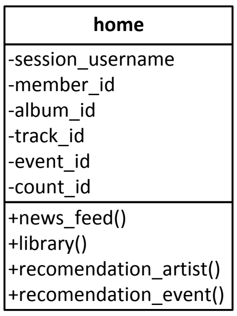


Figure 2.4.2-2: home Class

a) Input/Output Data Elements

Input: session_username, member_id, album_id, track_id, event_id, count_id
Output: N/A

b) Details

Class Type	:	View Class		
Responsibility	:	The homepage of the register user (type = User) after succeed with login.		
Attributes	:	session_username	:	Varchar
	:	member_id	:	Integer
	:	album_id	:	Integer
	:	track_id	:	Integer

	:	event_id	:	Integer
	:	count_id	:	Integer
Methods	:	news_feed()	:	View the most update activities from the artists
	:	library()	:	Total views and tracks played of the session user
	:	recommended_artist()	:	View the random list of artists available in the system
	:	recommended_event()	:	View the events sorted by upcoming date

Table 2.4.2-1: home Details

c) Algorithm

Method: news_feed ()

```
BEGIN

    Get member_id, album_id, track_id and event_id from database
    Display the result

END
```

Method: library ()

```
BEGIN

    Get count_id from database
    Display the result

END
```

Method: recommended_artist ()

```
BEGIN

    Get member_id which type = Artist from database
    Display the result

END
```

Method: recommended_event ()

```
BEGIN

    Get event_id from database sort by date descending (upcoming date)
    Display the result

END
```

ii) profile Design

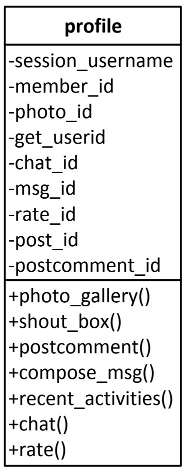


Figure 2.4.2-3: profile Class

a) Input/Output Data Elements

Input: session_username, member_id, photo_id, get_userid, chat_id, msg_id, rate_id, post_id, postcomment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This page contains the profile of the user include quick button to compose message, random view of the photos uploaded, shout where interaction medium are, and list of activities that`s going on in the system		
Attributes	:	session_username	:	Varchar
	:	member_id	:	Integer
	:	photo_id	:	Integer
	:	get_userid	:	Integer
	:	chat_id	:	Integer
	:	msg_id	:	Integer
	:	rate_id	:	Integer
	:	post_id	:	Integer
	:	postcomment_id	:	Integer
Methods	:	photo_gallery()	:	Random uploaded photo that can be view on the page
	:	shout_box()	:	Enables user to post their comment (interaction medium)
	:	postcomment()	:	Enables user to post the comment under the posted comment

	:	compose_msg()	:	Quick link to compose message to a friends
	:	recent_activities()	:	View the activities that's going on in the system
	:	chat()	:	Enables online user (same session) to chat among them
	:	rate()	:	Like and dislike button function that placed below the comment`s post

Table 2.4.2-2: profile Details

c) Algorithm

Method: photo_gallery ()

```
BEGIN

    Get photo_id from database where uploaded by session_username
    Display the result with limitation being set to 4 (4 random photos only)

END
```

Method: shout_box ()

```
BEGIN

    Get session_username from database
    View post if session_username posted something on own wall
    Get userid and friend_id from database
    If userid is friends with session_username
    Display his post

END
```

Method: postcomment ()

```
BEGIN

    Get member_id and userid from the database
    Check friends with session_username
    If friends then allow him to post a comment
    Display the comment below the posted comment

END
```

Method: compose_msg ()

```
BEGIN

    Select friends list
    Type content and send
    Send msg_id to database to save the id
    Message send is success

END
```

Method: recent_activities ()

```
BEGIN

    Get userid and updated content
    Insert the updated items into database
    Display

END
```

Method: rate ()

```
BEGIN

    Select post_id to be rate
    Click on the rate button to rate (like or dislike)
    Insert the result into database
    Display the inserted result with total count of rate

END
```

iii) photos Design

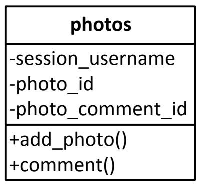


Figure 2.4.2-4: photos Class

a) Input/Output Data Elements

Input: session_username, photo_id, photo_comment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	Photo management page that enables user to upload the image types and organize the uploaded image		
Attributes	:	session_username	:	Varchar
	:	photo_id	:	Integer
	:	photo_comment_id	:	Integer
Methods	:	add_photo()	:	Add a photo into database

Table 2.4.2-3: photos Details

c) Algorithm

Method: add_photo ()

```
BEGIN

    Select a photo to be uploads with the name of the photo
    Insert chosen photo into database
    Get photo_id to display the result
    Get photo_comment_id to display total comments

END
```

iv) photos_comment Design

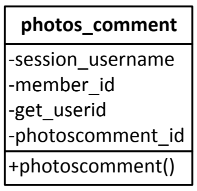


Figure 2.4.2-5: photos_comment Class

a) Input/Output Data Elements

Input: session_username, member_id, get_userid, photo_comment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	Photo comment page that enables user to post a comment regarding the photo and received the comment from others		
Attributes	:	session_username	:	Varchar

	:	member_id	:	Integer
	:	get_userid	:	Integer
	:	photo_comment_id	:	Integer
Methods	:	photoscomment()	:	Control the function of the photo comment including post and received comment

Table 2.4.2-4: photos_comment Details

c) Algorithm

Method: photoscomment ()

```
BEGIN

    Post comment on the available text box
    Insert content into database
    Select userid that comment on the photo too
    Get photo_comment_id to display total comments

END
```

v) friends Design

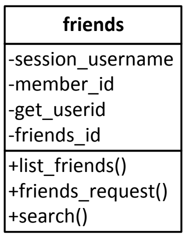


Figure 2.4.2-6: friends Class

a) Input/Output Data Elements

Input: session_username, member_id, get_userid, friends_id

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	User`s friend management pages include the list of the user`s friends, friends request and the function to accept or delete the friends request		
Attributes	:	session_username	:	Varchar
	:	member_id	:	Integer
	:	get_userid	:	Integer
	:	friends_id	:	Integer
Methods	:	listfriends ()	:	List of the user`s friends
	:	friends_request()	:	Enables the user to perform a function on friends request whether to accept it or decline
	:	search()	:	Search function for the friends who already friended with the user

Table 2.4.2-5: friends Details

c) Algorithm

Method: listfriends ()

```
BEGIN

    Select friends_id where friends_id is friend with session_username
    Get member_id from database and display the result

END
```

Method: friends_request ()

```
BEGIN

    Select friends_id where friends_id is requested friend with session_username
    Perform action
    If action = accept then add the friends_id in the database
    Else do nothing
    Get friends_id and display the result

END
```

Method: search ()

```
BEGIN

    Enter a character
    If character = member username then display the result

END
```

vi) mail Design

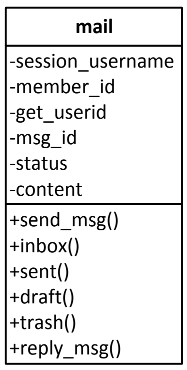


Figure 2.4.2-7: mail Class

a) Input/Output Data Elements

Input: session_username, member_id, get_userid, msg_id, status, content
Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	Mail page enables user to send mail, save and delete the mail whether it was received from others or its own sent mail		
Attributes	:	session_username	:	Varchar
	:	member_id	:	Integer
	:	get_userid	:	Integer
	:	msg_id	:	Integer
	:	status	:	Varchar
	:	content	:	Text
Methods	:	send_msg()	:	Send the message to chosen friends
	:	inbox()	:	Control the page that received message from others
	:	sent()	:	Control the page that store the sent message

	:	draft()	:	Control the page that store the message that being save by user
	:	trash()	:	Control the page that store the deleted message that perform by the user
	:	reply_msg()	:	Enables user to reply message

Table 2.4.2-6: mail Details

c) Algorithm

Method: mail ()

```
BEGIN

    Select friends_id to send message
    Call send_msg()
    Insert sent message into database
    Call sent()
    Display sent message into sent()
    If perfom an action to the message in the sent()
    If perform = save then status = draft (call draft())
    If perform = delete then status = trash (call trash())

    Get userid and message_id where userid sent message to session_username
    Display content and call inbox()
    If reply = yes then call reply_msg()

END
```

vii) settings Design

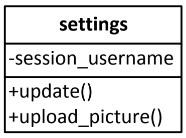


Figure 2.4.2-8: settings Class

a) Input/Output Data Elements

Input: session_username

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	The setting page to let user to update his profile		
Attributes	:	session_username	:	Varchar
Methods	:	update()	:	Update the profile
	:	upload_picture()	:	Upload the default picture

Table 2.4.2-7: settings Details

c) Algorithm

Method: update ()

```
BEGIN

    Get input from button posted
    Update database and set the value = input
    Display updated result

END
```

Method: upload_picture ()

```
BEGIN

    Get session_username and current photo
    Update database and set the photo = latest photo
    Display updated result

END
```

viii) userprofile Design

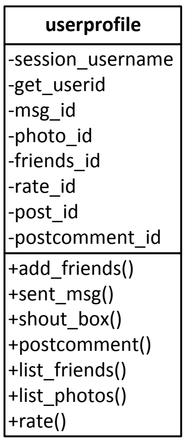


Figure 2.4.2-9: userprofile Class

a) Input/Output Data Elements

Input: session_username, get_userid, photo_id , friends_id, rate_id, post_id, postcomment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class
Responsibility	:	Userprofile page act as a view pages for the not login user. Example, when a login user wants to view others page including

		his friends, this page will come out. Several functionalities implement here such as add to friend function, send message, shout box, list of user`s friends and random view of user`s photos		
Attributes	:	session_username	:	Varchar
	:	get_userid	:	Integer
	:	photo_id	:	Integer
	:	friends_id	:	Integer
	:	rate_id	:	Integer
	:	post_id	:	Integer
	:	postcomment_id	:	Integer
Methods	:	add_friends()	:	Function for adding a user to the list of friends
	:	send_msg()	:	A link to compose message to the user
	:	shout_box()	:	Enables user to post their comment (interaction medium)
	:	postcomment()	:	Enables user to post the comment under the posted comment
	:	list_friends()	:	Display a user`s list of friends

	:	list_photos()		Display a user`s uploaded of friends with random displaying
	:	rate()	:	Like and dislike button function that placed below the comment`s post

Table 2.4.2-8: userprofile Details

c) Algorithm

Method: add_friends ()

```
BEGIN

    Get userid from database where userid is no friend with session_username
    If userid = friend then the add to friend button will disappear
    If userid != friend then get userid and display the add to friend button appear
    Insert userid in the friends table
    Get friends_id and display friends list

END
```

Method: send_msg ()

```
BEGIN

    Select friends_id
    Type content and send
    Send friends_id to database
    Get msg_id
    Message send is success

END
```

Method: shout_box ()

```
BEGIN

    Get session_username from database
    View post if session_username posted something on own wall
    Get userid and friend_id from database
    If userid is friends with session_username
    Display his post

END
```

Method: postcomment ()

```
BEGIN

    Get member_id and userid from the database
    Check friends with session_username
    If friends then allow him to post a comment
    Display the comment below the posted comment

END
```

Method: list_photos()

```
BEGIN

    Get photo_id from database where uploaded by session_username
    Display the result with limitation being set to 4 (4 random photos only)

END
```

Method: rate ()

```
BEGIN

    Select post_id to be rate
    Click on the rate button to rate (like or dislike)
    Insert the result into database
    Display the inserted result with total count of rate

END
```

ix) friends_photos, friends_photos_comment and friends_friends Design

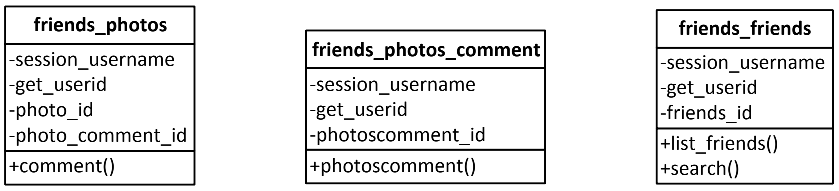


Figure 2.2-3: friends_photos, friends_photos_comment and friends_friends Class

a) Input/Output Data Elements

For these three classes, their input and output attriburte same as above design.

1. friends_photos (add get_userid) = photos
2. friends_photos_comment = photos_comment (exclude member_id)
3. friends_friends = friends (exclude member_id)

b) Details

Detail of these three classes same as above class.

Class type: View class

Responsibility: These classes are used for viewing the profile. So the details of these classes are the same from its controller class which is all defined in the above.

c) Algorithm

Method of these three classes same as above class.

2.4.3 Musician/Artists Module

Figure below shows musician/artists module with its relationship among the other subsystem classes.

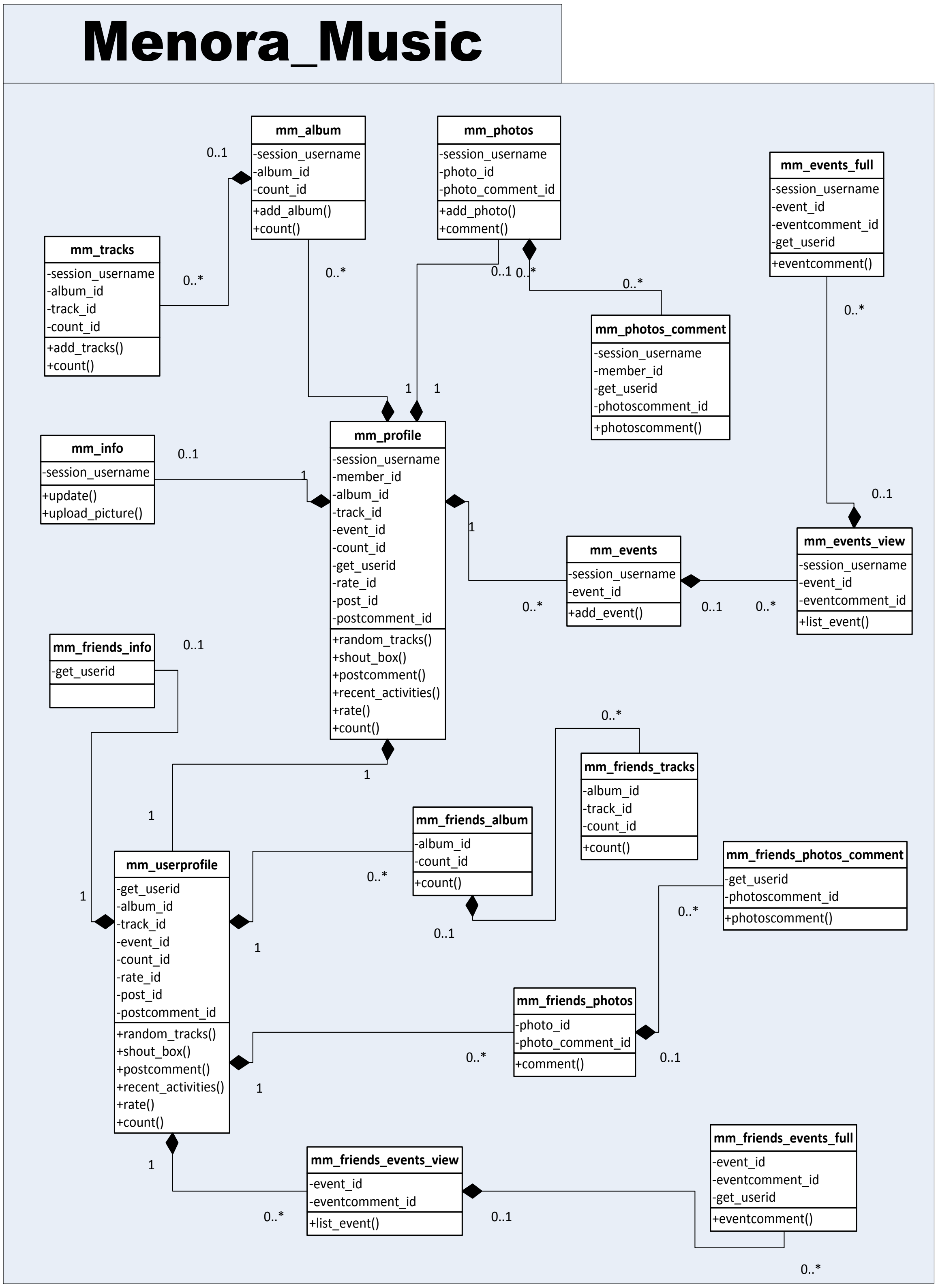


Figure 2.4.3-1: Musician/Artists Module Relationship

➢ **Musician/Artists Module Class**

Below is the details explanation about the classes available in the Musician/Artists Module.

i) mm_profile Design

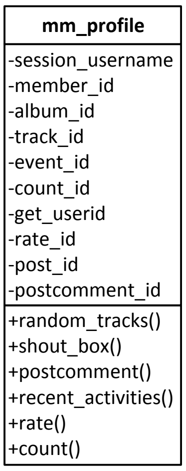


Figure 2.4.3-2: mm_profile Class

a) Input/Output Data Elements

Input: session_username, member_id, album_id, track_id, count_id, get_userid, rate_id, post_id, postcomment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class
Responsibility	:	This pages act as a home page to the user type='Artist'. It enables the artists to post/received comment via shout box, organize their content such as biography and see the random

		display of the uploaded albums and events.		
Attributes	:	session_username	:	Varchar
	:	member_id	:	Integer
	:	album_id	:	Integer
	:	track_id	:	Integer
	:	event_id	:	Integer
	:	count_id	:	Integer
	:	get_userid	:	Integer
	:	rate_id	:	Integer
	:	post_id	:	Integer
	:	postcomment_id	:	Integer
Methods	:	random_tracks()	:	Generate the random tracks from the uploaded tracks
	:	shout_box()	:	Enables user to post their comment (interaction medium)
	:	postcomment()	:	Enables user to post the comment under the posted comment
	:	recent_activities()	:	View the activities that's going on in the system
	:	rate()	:	Like and dislike button function that placed below the

				comment`s post
	:	count()	:	Function to calculate how many user`s pages visited and total play of the user`s tracks

Table 2.4.3-1: mm_profile Details

c) Algorithm

Method: random_tracks ()

```
BEGIN

    Get album_id, tracks_id from the database
    Compare album_id and tracks_id with session_username
    If album_id && tracks_id associated with session_username
    Display random tracks

END
```

Method: shout_box ()

```
BEGIN

    Get session_username from database
    View post if session_username posted something on own wall
    Get userid and friend_id from database
    If userid is friends with session_username
    Display his post

END
```

Method: postcomment ()

```
BEGIN

    Get member_id and userid from the database
    Check friends with session_username
    If friends then allow him to post a comment
    Display the comment below the posted comment
```

```
END

Method: recent_activities ()

BEGIN

    Get userid and updated content
    Insert the updated items into database
    Display

END

Method: rate ()

BEGIN

    Select post_id to be rate
    Click on the rate button to rate (like or dislike)
    Insert the result into database
    Display the inserted result with total count of rate

END

Method: count ()

BEGIN

    Select rate_id, member_id from the database
    Compare rate_id with member_id if those two associated
    If associated then get the rate_id with totalcount
    Else then do nothing

END
```

ii) mm_album Design

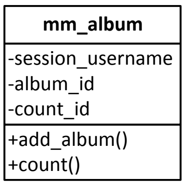


Figure 2.4.3-3: mm_album Class

a) Input/Output Data Elements

Input: session_username, album_id, count_id

Output: New album add

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This pages enable user to add the new album and view the list of the added albums		
Attributes	:	session_username	:	Varchar
	:	album_id	:	Integer
	:	count_id	:	Integer
Methods	:	add_album ()	:	Add a new album with picture
	:	count()	:	Function for counting the total played of the tracks in the each album

Table 2.4.3-2: mm_album Details

c) Algorithm

Method: add_album ()

```
BEGIN

    Click on the “Add album” link and insert the particular form
    Get input from inserted and save it into the database
    Retrieve album_id from database and display the list of uploaded album

END
```

iii) mm_tracks Design

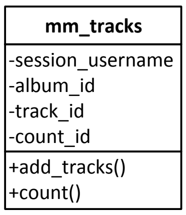


Figure 2.4.3-4: mm_tracks Class

a) Input/Output Data Elements

Input: session_username, album_id, track_id, count_id
Output: New track add

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This pages enable user to add the new track in each of the uploaded album		
Attributes	:	session_username	:	Varchar
	:	album_id	:	Integer
	:	track_id	:	Integer
	:	count_id	:	Integer
Methods	:	add_tracks ()	:	Add a new tracks
	:	count()	:	Function for counting the total played of the tracks in the each album

Table 2.4.3-3: mm_tracks Details

c) Algorithm

Method: add_tracks ()

```
BEGIN

    Click on the "Add tracks" link and insert the particular form
    Get input from inserted and save it into the database
    Retrieve tracks_id and album_id from database and display the list of uploaded
    tracks

END
```

iv) mm_events Design

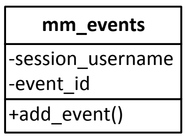


Figure 2.4.3-5: mm_events Class

a) Input/Output Data Elements

Input: session_username event_id

Output: New event add

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This pages enable user to add the new event		
Attributes	:	session_username	:	Varchar
	:	album_id	:	Integer

Methods	:	add_event ()	:	Add a new event

Table 2.4.3-4: mm_events Details

c) Algorithm

Method: add_event ()

```
BEGIN

    Click on the "Add events" link and insert the particular form
    Get input from inserted and save it into the database
    Retrieve event_id from database and display the list of uploaded events

END
```

v) mm_events_view Design

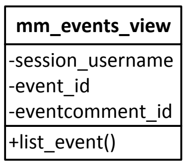


Figure 2.4.3-6: mm_events_view Class

a) Input/Output Data Elements

Input: session_username, event_id, eventcomment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class

Responsibility	:	This page show the list of the created event		
Attributes	:	session_username	:	Varchar
	:	event_id	:	Integer
	:	eventcomment_id	:	Integer
Methods	:	list_event ()	:	Display the list of the created events

Table 2.4.3-5: mm_events_view Details

c) Algorithm

Method: list_event ()

```
BEGIN

    Get event_id
    If event_id associated with session_username
    Then display the list of events with link to the event`s comment
    (eventcomment_id)

END
```

vi) mm_events_full Design

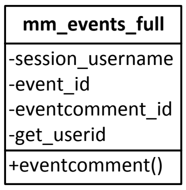


Figure 2.4.3-7: mm_events_full Class

a) Input/Output Data Elements

Input: session_username, event_id, eventcomment_id, get_userid

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This page show the event content and comments when clicked on the previous page (mm_event_view class)		
Attributes	:	session_username	:	Varchar
	:	event_id	:	Integer
	:	eventcomment_id	:	Integer
	:	get_userid	:	Integer
Methods	:	event_comment ()	:	Display the list of the created events

Table 2.4.3-6: mm_events_full Details

c) Algorithm

Method: event_comment ()

```
BEGIN

    Get event_id
    If event_id associated with session_username
    Then get content with get_userid
    If get_userid associated with evencomment_id
    Then display comment
    Else then add new comment and display result

END
```

vii) mm_photos Design

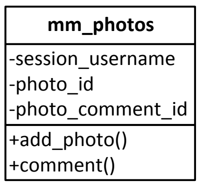


Figure 2.4.3-8: mm_photos Class

a) Input/Output Data Elements

Input: session_username, photo_id, photo_comment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	Photo management page that enables user to upload the image types and organize the uploaded image		
Attributes	:	session_username	:	Varchar
	:	photo_id	:	Integer
	:	photo_comment_id	:	Integer
Methods	:	add_photo()	:	Add a photo into database

Table 2.4.3-7: mm_photos Details

c) Algorithm

Method: add_photo ()

```
BEGIN

    Select a photo to be uploads with the name of the photo
    Insert chosen photo into database
    Get photo_id to display the result
    Get photo_comment_id to display total comments

END
```

viii) mm_photos_comment Design

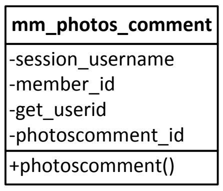


Figure 2.4.3-9: mm_photos_comment Class

a) Input/Output Data Elements

Input: session_username, member_id, get_userid, photo_comment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	Photo comment page that enables user to post a comment regarding the photo and received the comment from others		
Attributes	:	session_username	:	Varchar
	:	member_id	:	Integer
	:	get_userid	:	Integer

	:	photo_comment_id	:	Integer
Methods	:	photoscomment()	:	Control the function of the photo comment including post and received comment

Table 2.4.3-8: mm_photos_comment Details

c) Algorithm

Method: photoscomment ()

```
BEGIN

    Post comment on the available text box
    Insert content into database
    Select userid that comment on the photo too
    Get photo_comment_id to display total comments

END
```

ix) mm_info Design

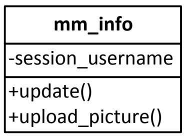


Figure 2.4.3-10: mm_info Class

a) Input/Output Data Elements

Input: session_username

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	The setting page to let user to update his profile		
Attributes	:	session_username	:	Varchar
Methods	:	update()	:	Update the profile
	:	upload_picture()	:	Upload the default picture

Table 2.4.3-9: mm_info Details

c) Algorithm

Method: update ()

```
BEGIN

    Get input from button posted
    Update database and set the value = input
    Display updated result

END
```

Method: upload_picture ()

```
BEGIN

    Get session_username and current photo
    Update database and set the photo = latest photo
    Display updated result

END
```

x) mm_userprofile Design

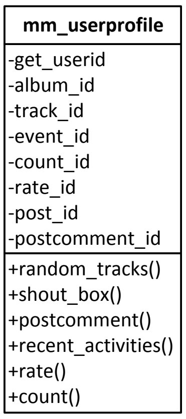


Figure 2.4.3-11: mm_userprofile Class

a) Input/Output Data Elements

Input: get_userid, album_id , track_id, event_id, count_id, rate_id, post_id, postcomment_id

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	mm_userprofile page act as a view pages for the not login user. Example, when a login user wants to view others page including his friends, this page will come out. Several functionalities implement here such as info, random tracks display, list of random album and events, shout box and total viewed page and played.		
Attributes	:	get_userid	:	Integer
	:	album_id	:	Integer

	:	track_id	:	Integer
	:	event_id	:	Integer
	:	count_id	:	Integer
	:	rate_id	:	Integer
	:	post_id	:	Integer
	:	postcomment_id	:	Integer
Methods	:	random_tracks()	:	Generate the random tracks from the uploaded tracks
	:	shout_box()	:	Enables user to post their comment (interaction medium)
	:	postcomment()	:	Enables user to post the comment under the posted comment
	:	recent_activities()	:	View the activities that’s going on in the system
	:	rate()	:	Like and dislike button function that placed below the comment`s post
	:	count()	:	Function to calculate how many user`s pages visited and total play of the user`s tracks

Table 2.4.3-10: mm_userprofile Details

c) Algorithm

Method: random_tracks ()

```
BEGIN

    Get album_id, tracks_id from the database
    Compare album_id and tracks_id with session_username
    If album_id && tracks_id associated with session_username
    Display random tracks

END
```

Method: shout_box ()

```
BEGIN

    Get session_username from database
    View post if session_username posted something on own wall
    Get userid and friend_id from database
    If userid is friends with session_username
    Display his post

END
```

Method: postcomment ()

```
BEGIN

    Get member_id and userid from the database
    Check friends with session_username
    If friends then allow him to post a comment
    Display the comment below the posted comment

END
```

Method: recent_activities ()

```
BEGIN

    Get userid and updated content
    Insert the updated items into database
    Display

END
```

Method: rate ()

```
BEGIN

    Select post_id to be rate
    Click on the rate button to rate (like or dislike)
    Insert the result into database
    Display the inserted result with total count of rate

END
```

Method: count ()

```
BEGIN

    Select rate_id, member_id from the database
    Compare rate_id with member_id if those two associated
    If associated then get the rate_id with totalcount
    Else then do nothing

END
```

xi) mm_friends_album, mm_friends_tracks, mm_friends_events_view, mm_friends_events_full, mm_friends_photos, mm_friends_photos_comment and mm_friends_info Design

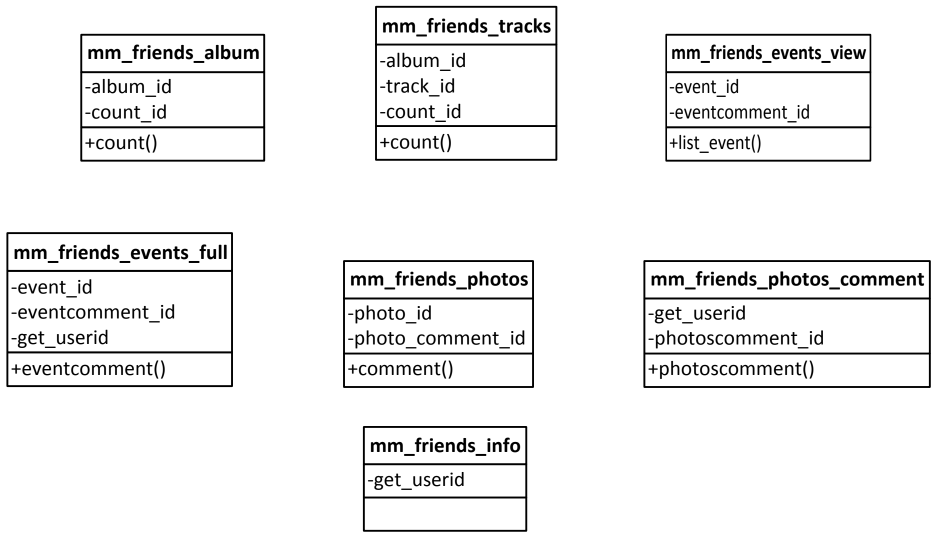


Figure 2.4.3-12: mm_friends_album, mm_friends_tracks, mm_friends_events_view, mm_friends_events_full, mm_friends_photos, mm_friends_photos_comment and mm_friends_info Class

a) Input/Output Data Elements

For these three classes, their input and output attriburte same as above design.

1. mm_friends_album = mm_album (exclude session_username)
2. mm_friends_tracks = mm_tracks (exclude session_username)
3. mm_friends_events_view = mm_event_view (exclude session_username)
4. mm_friends_events_full = mm_events_full (exclude session_username)
5. mm_friends_photos = mm_photos (exclude session_username)
6. mm_friends_photos_comment = mm_photos_comment (exclude session_username)
7. mm_friends_info (add get_userid) = mm_info (exclude session_username)

b) Details

Detail of these three classes same as above class.

Class type: View class

Responsibility: These classes are used for viewing the profile. So the details of these classes are the same from its controller class which is all defined in the above.

c) Algorithm

Method of these three classes same as above class.

2.4.4 Admin Module

Figure below shows admin module with its relationship among the other subsystem classes.

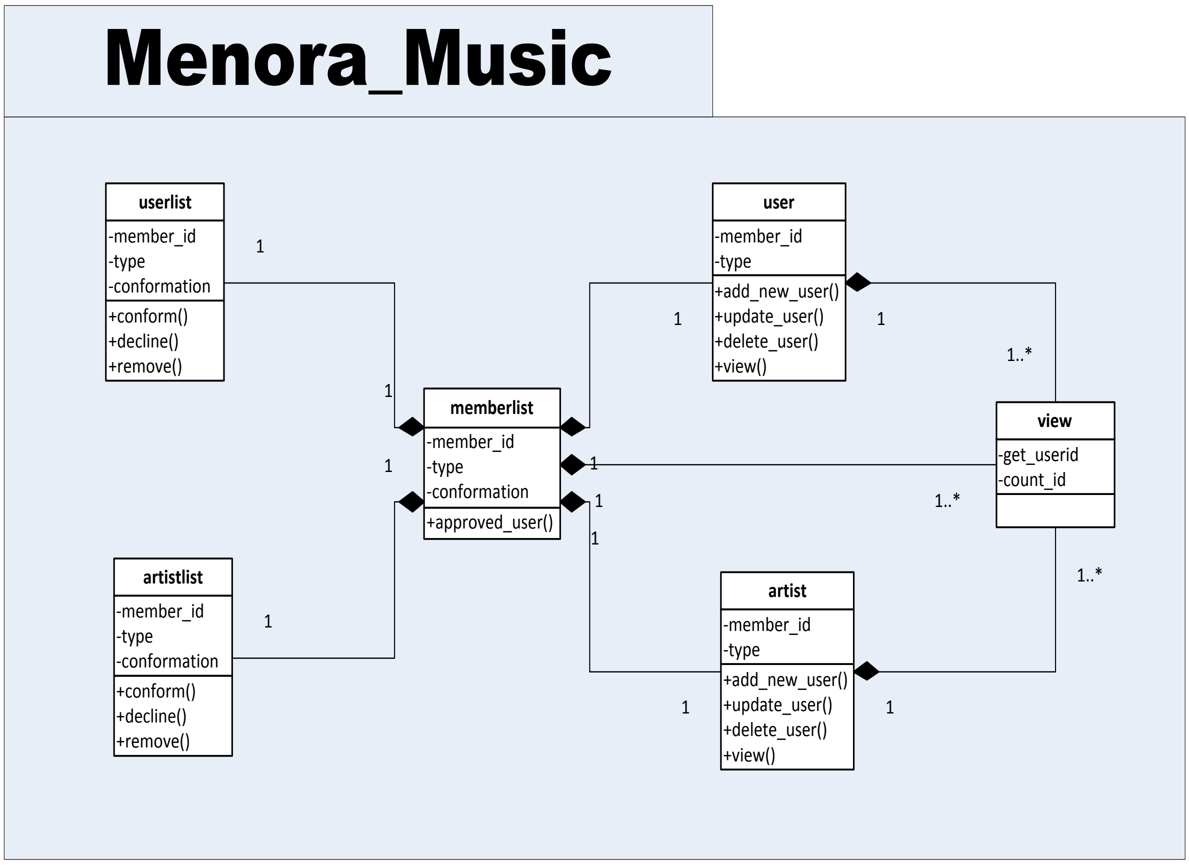


Figure 2.4.4-1: Admin Module Relationship

➢ **Admin Module Class**

Below is the details explanation about the classes available in the Admin Module.

i) memberlist Design

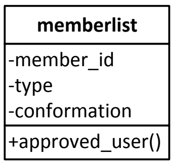


Figure 2.4.4-2: memberlist Class

a) Input/Output Data Elements

Input: member_id, type, conformation

Output: List of the user where comformation = 1 (authorized)

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This page display the list of users registered with the system where the conformation is approved (conformation = 1)		
Attributes	:	member_id	:	Integer
	:	type	:	Varchar
	:	comformation	:	Integer
Methods	:	approve_user()	:	List of the approved users

Table 2.4.4-1: memberlist Details

c) Algorithm

Method: approve_user ()

```
BEGIN

    Get member_id, type and comformation from database
    If comformation = 1
    Then display the result

END
```

ii) userlist Design

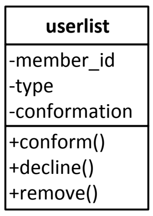


Figure 2.4.4-3: userlist Class

a) Input/Output Data Elements

Input: member_id, type, conformation

Output: List of the type = user where comformation = 0 (unauthorized)

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This page display the list of users whose type = User and conformation = 0 which is not approved yet by the admin		
Attributes	:	member_id	:	Integer
	:	type	:	Varchar
	:	comformation	:	Integer
Methods	:	conform()	:	Approve the user
	:	decline()	:	Decline the user

	:	remove()	:	Remove the user

Table 2.4.4-2: userlist Details

c) Algorithm

Method: comform ()

```
BEGIN

    Get member_id, type and comformation from database
    If type = User and comformation = 0
    Then display the list of users
    If user action = comform()
    Then set the comformation = 1

END
```

Method: decline ()

```
BEGIN

    Get member_id, type and comformation from database
    If type = User and comformation = 0
    Then display the list of users
    If user action = decline()
    Then set the comformation = 0 (do nothing)

END
```

Method: remove ()

```
BEGIN

    Get member_id, type and comformation from database
    If type = User and comformation = 0
    Then display the list of users
    If user action = remove()
    Then remove the user from database

END
```

iii) artistlist Design

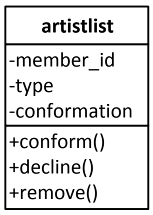


Figure 2.4.4-4: artistlist Class

a) Input/Output Data Elements

Input: member_id, type, conformation

Output: List of the type = artist where comformation = 0 (unauthorized)

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This page display the list of users whose type = User and conformation = 0 which is not approved yet by the admin		
Attributes	:	member_id	:	Integer
	:	type	:	Varchar
	:	comformation	:	Integer
Methods	:	conform()	:	Approve the user
	:	decline()	:	Decline the user

	:	remove()	:	Remove the user

Table 2.4.4-3: artistlist Details

c) Algorithm

Method: comform ()

```
BEGIN

    Get member_id, type and comformation from database
    If type = Artist and comformation = 0
    Then display the list of users
    If user action = comform()
    Then set the comformation = 1

END
```

Method: decline ()

```
BEGIN

    Get member_id, type and comformation from database
    If type = Artist and comformation = 0
    Then display the list of users
    If user action = decline()
    Then set the comformation = 0 (do nothing)

END
```

Method: remove ()

```
BEGIN

    Get member_id, type and comformation from database
    If type = Artist and comformation = 0
    Then display the list of users
    If user action = remove()
    Then remove the user from database

END
```

iv) user Design

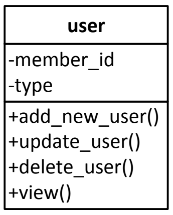


Figure 2.4.4-5: user Class

a) Input/Output Data Elements

Input: member_id, type

Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This page enables admin to manage the profile where type = user		
Attributes	:	member_id	:	Integer
	:	type	:	Varchar
Methods	:	add_new_user()	:	Add new user to the system
	:	update_user()	:	Update existing user profile
	:	delete_user()	:	Delete user from the system

	:	view()	:	View the existing user

Table 2.4.4-4: user Details

c) Algorithm

Method: add_new_user ()

```
BEGIN

    Click on the add new user button and fill in particular form
    If button = Add
    Then add the particular input into the database

END
```

Method: update ()

```
BEGIN

    Select the user wants to update and click on the update button
    Adjust the desire text line to update
    If button = Update
    Then update the user`s profile in database

END
```

Method: delete_user ()

```
BEGIN

    Select the user wants to delete and click on the delete button
    If button = Delete
    Then remove the user from database

END
```

v) artist Design

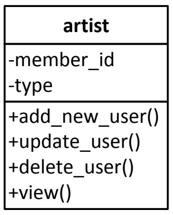


Figure 2.4.4-6: artist Class

d) Input/Output Data Elements

Input: member_id, type

Output: N/A

a) Details

Class Type	:	View and Controller Class		
Responsibility	:	This page enables admin to manage the profile where type = user		
Attributes	:	member_id	:	Integer
	:	type	:	Varchar
Methods	:	add_new_user()	:	Add new user to the system
	:	update_user()	:	Update existing user profile
	:	delete_user()	:	Delete user from the system

	:	view()	:	View the existing user

Table 2.4.4-5: artist Details

b) Algorithm

Method: add_new_user ()

```
BEGIN

    Click on the add new user button and fill in particular form
    If button = Add
    Then add the particular input into the database

END
```

Method: update ()

```
BEGIN

    Select the user wants to update and click on the update button
    Adjust the desire text line to update
    If button = Update
    Then update the user`s profile in database

END
```

Method: delete_user ()

```
BEGIN

    Select the user wants to delete and click on the delete button
    If button = Delete
    Then remove the user from database

END
```

2.4.5 Others Subsystem (Navigation Bar)

➢ **Music Class**

Below is the details explanation about the Music Class.

i) music Design

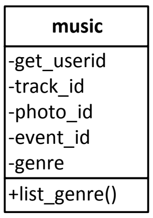


Figure 2.4.5-1: music Class

a) Input/Output Data Elements

Input: get_userid, track_id, photo_id, event_id, genre

Output: Sorting the artist via its genre (type)

b) Details

Class Type	:	View Class		
Responsibility	:	This page display the artist via its genre (type)		
Attributes	:	get_userid	:	Integer
	:	track_id	:	Integer
	:	photo_id	:	Integer
	:	event_id		Integer
	:	genre		Integer

Methods	:	list_genre ()	:	List of artist via its genre

Table 2.4.5-1: music Details

c) Algorithm

Method: list_genre ()

```
BEGIN

    Select all of the input from database
    Sort the artist via its type of genre

END
```

- **Event Class**

  Below is the details explanation about the Event Class.

i) event Design

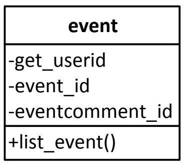


Figure 2.4.5-2: event Class

a) Input/Output Data Elements

Input: get_userid, event_id, eventcomment_id

Output: Sorting the events

b) Details

Class Type	:	View Class		
Responsibility	:	This page display the list of available events		
Attributes	:	get_userid	:	Integer
	:	event_id	:	Integer
	:	eventcomment_id	:	Integer
Methods	:	list_event ()	:	List of available events

Table 2.4.5-2: event Details

c) Algorithm

Method: list_event ()

```
BEGIN

    Select all of the input from database
    Display the result by sort the list of available events

END
```

➢ **Forum Class**

Below is the details explanation about the Forum Class.

i) forum Design

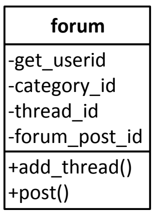


Figure 2.4.5-3: forum Class

a) Input/Output Data Elements

Input: get_userid, category_id, thread_id, forum_post_id
Output: N/A

b) Details

Class Type	:	View and Controller Class		
Responsibility	:	This page enables participants to join the forum by creating new threads on the existing categories. They also can reply to the available threads via posting a post/comment		
Attributes	:	get_userid	:	Integer
	:	category_id	:	Integer
	:	thread_id	:	Integer
	:	forum_post_id		Integer
Methods	:	add_thread ()	:	Add a new thread under selected categories
	:	post()	:	Post a comment into selected threads

Table 2.4.5-3: forum Details

c) Algorithm

Method: add_thread ()

```
BEGIN

    Click on the add thread button
    Select the categories available and fill in the required blanks
    If button = Add
    Then get category_id and insert into the database with the thread created
    (thread_id)

END
```

Method: post ()

```
BEGIN

    Select threads that want to participate
    Click on the reply to post button
    If button = Reply
    Then get category_id,thread_id and insert into the database with the post`s
    contents
    Get forum_post_id to display the posted content in the targeted thread

END
```

2.5 Method and Material

The best methodology for this type of system is Rapid Application Development which is called as RAD.

2.5.1 Project Methodology

Rapid application development is a software development methodology that uses minimal planning in favour of rapid prototyping. In rapid application development, to define user's requirements and design the final system, structured techniques and prototyping are expressly used. During the development process, initially the development of preliminary data models and business process models using structured techniques will starts. Next, requirements are verified using prototyping, eventually to refine the data and process models.

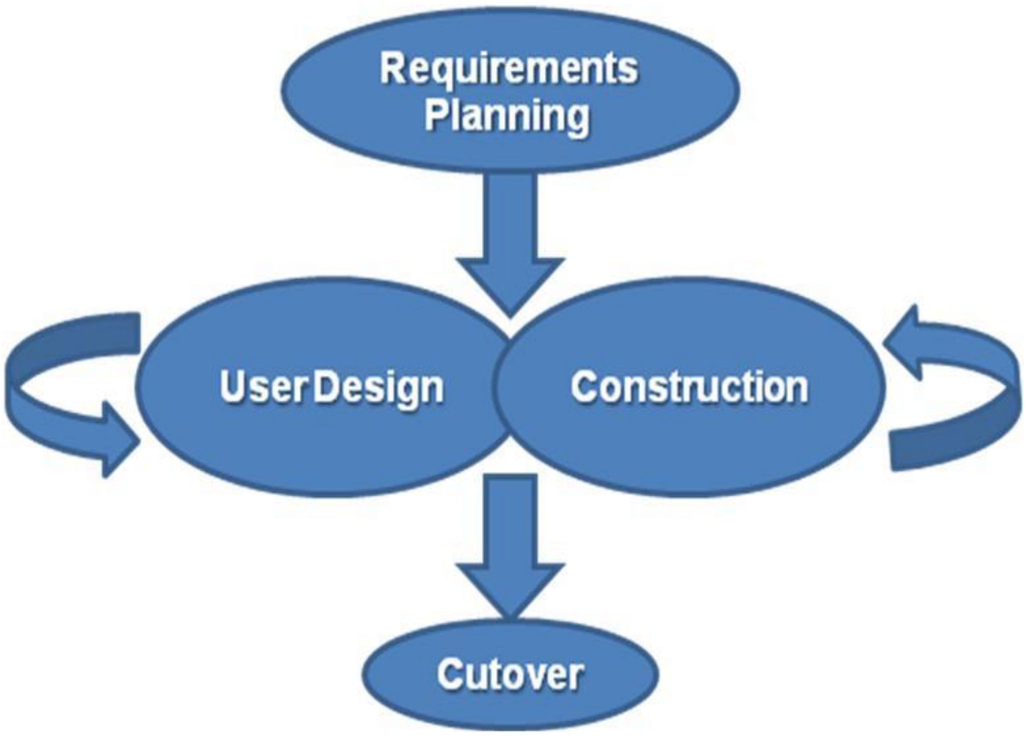


Figure 2.5.1-1: Rapid Application Development Methodology

i) Requirements Planning

Combine elements of the system planning and systems analysis phases of the System Development Life Cycle (SDLC). Users, managers, and IT staff members discuss and agree on business needs, project scope, constraints, and system requirements. It ends when the team agrees on the key issues and obtains management authorization to continue.

In this phase, user requirement are collected in order to plan the development process of OMS. Several processes had been done in this phase, understanding the current manual process by interview the selected users and study on existing system. After collecting data, meeting will be conduct with customer and the user requirement will sign off before development process start.

ii) User Design

During this phase, all system processes, inputs and outputs will develop into models and prototypes with involve of users interact with systems analysts. User Design is a constant cooperative process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs.

The design phase should be systematic and specific. In this phase, user requirement will transform into diagram using tools such as IBM Rational Rose and Microsoft Office Visio. Use case diagram, context diagram, activity diagram and ER diagram will design as logical description for the process flow of the system.

iii) Construction

Focus on program and application development task similar to the SDLC. In RAD, users continue to participate and can still suggest changes or improvements as actual screens or reports are developed. Its tasks are programming and application development, coding, unit-integration and system testing.

In this phase, the design diagram will transform into detail design which include coding execution. Construction phase also known as development phase that begin with make a coding or engine in each module by using selected language and tools. The language used is PHP because it provides high performance and it is open source.

iv) Cutover

Resemble the final tasks in the SDLC implementation phase; the entire process is compressed compared with traditional methods. These assist the process of building, delivering, and placing the new system quicker. The process tasks include data conversion, full-scale testing, system changeover, user training.

2.6 Testing Plan

2.6.1 Objectives

1) Purpose

This document describes the plan for testing web-based of the Online Music System. This Test Plan document supports the following objectives:

- Identify the project information and the module that should be tested.
- List the recommended test requirements.
- Recommend and describe the testing strategies to be employed.
- Identify the required resources.
- List the deliverable elements of the test activities.

2) Scope

This Test Plan describes the integration and system tests that will be conducted on this system following by integration of the subsystems and components. It is assumed that unit testing already provided thorough black box testing, extensive coverage of source code, and testing of all module interfaces.

The purpose of assembling the architectural prototype was to test feasibility and performance of the selected architecture. It is critical that all system and subsystem interfaces be tested as well as system performance at this early stage. Testing of system functionality and features will not be conducted on this system.

The interfaces between the following modules will be tested:

a) Users Module
    a. Home (home.php)
    b. Profile (profile.php)
    c. Photos (photos.php)
    d. Friends (friends.php)
    e. Messages (mail.php)
    f. Settings (info.php)
    g. User`s view (userprofile.php)

b) Musician/Artists Module
    a. Profile (mm_profile.php)
    b. Album & tracks (mm_album.php)
    c. Photos (mm_photos.php)

d. Events (mm_events.php)
e. Settings (mm_info.php)
f. User`s view (mm_userprofile.php)

The external interfaces to the following devices will be tested:

1. Local PCs
2. Remote PCs

The most critical performance measures to test are:

1. Response time for remote login to the user module (home) and musician/artists module (profile).
2. Response time to access classes in both of the user and musician/artists module.
3. User response time when system loaded with multiple logged in users or musician.
4. Musician response time when system loaded with multiple logged in users or musician.
5. Whole users response time when simultaneous accesses to the system`s database.

2.6.2 Requirement for Test

The listing below identifies those items (use cases, functional requirements, non-functional requirements) that have been identified as targets for testing. This list represents what will be tested.

1) Data and Database Integrity Testing

a. Verify access to OMS Database.
b. Verify simultaneous record read accesses.
c. Verify lockout during OMS Database updates.
d. Verify correct retrieval of update of database data.

2) Function Testing

a. The system shall interface with the OMS Database and shall support the data format as defined in the use cases. (ER Diagram)

b. The server component of the system shall operate on the Apache Web Server and shall run under the Window Operating System.
c. The users of the system shall operate on any personal computer with at least Pentium III Microprocessor or better.

3) Business Cycle Testing

None.

4) User Interface Testing

a. Verify ease of navigation while confront the interface.
b. The system shall be easy-to-use and shall be appropriate for the target users which are fans and musician.

5) Performance Testing

a. Verify response time to access external both user and musician/artists module.
b. Verify response time to access external OMS others subsystem.
c. Verify response time for remote login.
d. Verify response time for submittal of the registration function.

6) Load Testing

Verify system response when loaded with multiple logged on users.

7) Stress Testing

None.

8) Volume Testing

None.

9) Security and Access Control Testing

a. Verify Logon from a local PC.
b. Verify Logon from a remote PC.
c. Verify Logon security through user name and password.

10) Failover / Recovery Testing

None.

11) Configuration Testing

a. The users of the system shall run on Windows XP or better.
b. The web-based interface of this system shall run in Internet Explorer (IE) 6.0, Mozilla, Chrome browsers or better.
c. The web-based interface shall be compatible with Adobe Flash Player and Shockwave Flash environment.

12) Installation Testing

None.

2.6.3 Test Strategy

The Test Strategy presents the recommended approach to the testing of the software applications. The previous section on Test Requirements described what will be tested and this section describes how it will be tested.

The main considerations for the test strategy are the techniques to be used and the criterion for knowing when the testing is completed.

2.6.3.1 Testing Types

Below are the types of testing that will be used to test this system.

1) Data and Database Integrity Testing

The databases should be tested as separate systems. These systems should be tested without the applications (as the interface to the data). Additional research into the DBMS needs to be performed to identify the tools/techniques that may exist to support the testing identified below.

Test Objective:	Ensure database access methods and processes function properly and without data corruption.
Technique:	• Invoke each database access method and process, seeding each with valid and invalid data. • Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved.
Completion Criteria:	All database access methods and processes function as designed and without any data corruption.
Special Considerations:	• Testing may require a DBMS development environment or drivers to enter or modify data directly in the databases. • Processes should be invoked manually. • Small or minimally sized databases (limited number of records) should be used to increase the visibility of any non-acceptable events.

Table 2.6.3-1: Data and Database Integrity Testing

2) Function Testing

Testing of the application should focus on any target requirements that can be traced directly to use cases and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques, that is, verifying the application (and its internal processes) by interacting with the application via the GUI and analysing the output (results). Below is an outline of the testing recommended for each application:

Test Objective:	Ensure proper application navigation, data entry, processing, and retrieval.
Technique:	• Execute each use case, use case flow, or function, using valid and invalid data, to verify the following: • The expected results occur when valid data is used. • The appropriate error/warning messages are displayed when invalid data is used. • Each business rule is properly applied.
Completion Criteria:	• All planned tests have been executed. • All identified defects have been addressed.

Table 2.6.3-2: Function Testing

3) Business Cycle Testing

This section is not applicable to test of this system.

4) User Interface Testing

User Interface testing verifies a user's interaction with the software. The goal of UI Testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the applications. In addition, UI Testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

Test Objective:	Verify the following: • Navigation through the application properly reflects business functions and requirements, including pages to pages in browser, and use of access methods (tab keys, mouse movements, accelerator keys) • Pages objects and characteristics, such as menus, size, position, buttons, links and others conform to standards.
Technique:	Create/modify tests for each page in browser to verify proper navigation and object states for each web-based application and objects.
Completion Criteria:	Each pages in browser successfully verified to remain consistent with benchmark version or within acceptable

	standard
Special Considerations:	Not all properties for custom and third party objects can be accessed.

Table 2.6.3-3: User Interface Testing

5) Performance Profiling

Performance testing measures response times, transaction rates, and other time sensitive requirements. The goal of Performance Testing is to verify and validate the performance requirements have been achieved. Performance Testing is usually executed several times, each using a different "background load" on the system. The initial test should be performed with a "nominal" load, similar to the normal load experienced (or anticipated) on the target system. A second performance test is run using a peak load.

Additionally, Performance tests can be used to profile and tune a system’s performance as a function of conditions such as workload or hardware configurations. Transactions below refer to "logical business transactions." These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract.

Test Objective:	Validate system response time for designated transactions or business functions under a the following two conditions: • normal anticipated volume • anticipated worse case volume
Technique:	• Use Test Scripts developed for Business Model Testing (System Testing). • Modify data files (to increase the number of transactions) or modify scripts to increase the number of iterations each transaction occurs. • Scripts should be run on one machine (best case to benchmark single user, single transaction) and be repeated with multiple clients (virtual or actual, *see special considerations below*).
Completion Criteria:	• Single Transaction/single user: Successful completion of the test scripts without any failures and within the expected/required time allocation (per transaction) • Multiple transactions/multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation.

Special considerations:	• Comprehensive performance testing includes having a "background" load on the server. There are several methods that can be used to perform this, including: o "Drive transactions" directly to the server, usually in the form of SQL calls. o Create "virtual" user load to simulate many (usually several hundred) clients. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with "traffic." o Use multiple physical clients, each running test scripts to place a load on the system. • Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. • The databases used for Performance Testing should be either actual size, or scaled equally.

Table 2.6.3-4: Performance Testing

6) Load Testing

Load testing measures subjects the system-under-test to varying workloads to evaluate the system's ability to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics (response times, transaction rates, and other time sensitive issues).

Transactions below refer to "logical business transactions." These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract.

Test Objective:	Verify System Response time for designated transactions or business cases under varying workload conditions.
Technique:	• Use tests developed for Business Cycle Testing. • Modify data files (to increase the number of transactions) or the tests to increase the number of times each transaction occurs.
Completion Criteria:	• Multiple transactions / multiple users: Successful completion of the tests without any failures and within acceptable time allocation.

Special Considerations:	• Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. • The databases used for load testing should be either actual size, or scaled equally.

Table 2.6.3-5: Load Testing

7) Stress Testing

This section is not applicable to test of this system.

8) Volume Testing

This section is not applicable to test of this system.

9) Security and Access Control Testing

Security and Access Control Testing focus on two key areas of security:

a) Application security, including access to the Data or Business Functions, and
b) System Security, including logging into / remote access to the system.

Application security ensures that, based upon the desired security, users are restricted to specific functions or are limited in the data that is available to them. For example, everyone may be permitted to enter data and create new accounts, but only managers can delete them. If there is security at the data level, testing ensures that user "type" one can see all customer information, including financial data, however, user two only sees the demographic data for the same client.

System security ensures that only those users granted access to the system are capable of accessing the applications and only through the appropriate gateways.

Test Objective:	• Function/Data Security: Verify that user can access only those functions/data for which their user type is provided permissions. • System Security: Verify that only those users with access to the system and application(s) are permitted to access them.
Technique:	• Function/Data Security: Identify and list each user type and the functions/data each type has permissions for. • Create tests for each user type and verify permission by creating transactions specific to each user type. • Modify user type and re-run tests for same users. In each case verify those additional functions/data are correctly available or denied. • System Access (*see special considerations below*)
Completion Criteria:	For each known user type the appropriate function/data are available and all transactions function as expected and run in prior Application Function tests
Special Considerations:	• Access to the system must be reviewed/discussed with the appropriate network or systems administrator. This testing may not be required as it maybe a function of network or systems administration.

Table 2.6.3-6: Security and Access Control Testing

10) Failover and Recovery Testing

This section is not applicable to test of this system.

11) Configuration Testing

Configuration testing verifies operation of the software on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections and database servers vary. Client workstations may have different software loaded (e.g. applications, drivers, etc.) and at any one time many different combinations may be active and using different resources.

Test Objective:	Validate and verify that the client Applications function properly on the prescribed client workstations.
Technique:	• Use Integration and System Test scripts • Open/close various PC applications, either as part of the test or prior to the start of the test. • Execute selected transactions to simulate user activities into and out of various PC applications. • Repeat the above process, minimizing the available conventional memory on the client.
Completion Criteria:	For each pages from the browser, transactions are successfully completed without failure.
Special Considerations:	• What data are the applications running (i.e. large spreadsheet opened in Excel, 100 page document in Word). • The entire systems, network servers, databases, etc. should also be documented as part of this test.

Table 2.6.3-7: Configuration Testing

12) Installation Testing

This section is not applicable to test of this system.

2.6.3.2 Tools

The following tools will be employed for testing of this system:

Testing type	Tool
Test Management	o Rational RequisitePro o Rational Unified Process
Test Design	o Rational Rose
Defect Tracking	o Rational ClearQuest
Functional Testing	o Rational Robot

Performance Testing	o Rational Visual Quantify
Test Coverage Monitor or Profiler	o Rational Visual PureCoverage
Other Test Tools	o Rational Purify o Rational TestFactory
Project Management	o Microsoft Project o Microsoft Word o Microsoft Excel
DBMS tools	o TBD

Table 2.6.3-8: Tools

2.6.4 Deliverables

1) Test Suite

The Test Suite will define all the test cases and the test scripts which are associated with each test case.

2) Test Logs

It is planned to use RequisitePro to identify the test cases and to track the status of each test case. The test results will be summarized in RequisitePro as untested, passed, conditional pass, or failed. In summary, RequisitePro will be setup to support the following attributes for each test case, as defined in the Software Requirements Specification:

- Test status
- Build Number
- Tested By
- Date Tested
- Test Notes

3) Defect Reports

Rational ClearQuest will be used for logging and tracking individual defects.

PART 3: CONCLUSION AND FUTURE WORK

For the conclusion, OMS is developed to enhance the typical music website with additional features such as messaging, forum, comments, added song to user profile and purchase items. These features are useful and make it the comprehensive music website. With the purchases store included, the level of availability also increases. Users can make purchases with intended product from musician online instead need to go to music shop. The availability of the products makes music enthusiastic satisfied with the system. There are more enhancements that can be done to produce a better system. Further researches have to be done to ensure a good system created and fulfilled all users' requirement.

REFERENCES

Blogs – Advantages and Disadvantages. (2009). Retrieved 5 April, 2013, from Online EDU Blog: http://www.onlineedublog.com/blogs/

Social Networking for Musicians – Quality vs Quantity. (29 March, 2011). Retrieved 15 May, 2013, from Make It In Music: http://www.makeitinmusic.com/social-networking-for-musicians/

Social Media: What are the advantages and disadvantages of social networking sites? What should we include in a policy? (6 January, 2012). Retrieved 27 March, 2013, from SHRM: http://www.shrm.org/TemplatesTools/hrqa/Pages/socialnetworkingsitespolicy.aspx

Blog Type Websites. (n.d.). Retrieved 4 April, 2013, from Build Your Own Business: http://www.byobwebsite.com/lesson-subjects/blog-type-websites/

COMPARE MUSIC SOCIAL NETWORK WEBSITES. (n.d.). Retrieved 7 March, 2013, from FindTheBest: http://social-networking.findthebest.com/d/b/Music

Dahud, H. (26 July, 2012). *Fandalism: Finally A Social Network That Works For Musicians*. Retrieved 5 May, 2013, from Hypebot: http://www.hypebot.com/hypebot/2012/07/fandalism-finally-a-social-network-that-works-for-musicians-exclusive.html

Danah M. Boyd, N. B. (17 December, 2007). *Social Network Sites: Definition, History, and Scholarship*. Retrieved 2013 March, 23, from Wiley Online Library: http://onlinelibrary.wiley.com/doi/10.1111/j.1083-6101.2007.00393.x/full

Features And Benefits of Social Networking Sites. (n.d.). Retrieved 27 March, 2013, from Slideshare: http://www.slideshare.net/OnlineMarketing10/features-and-benefits-of-social-networking-sites

Heng, C. (2007). *The Pros and Cons of Using an Online Blog Software or a Content Management System (CMS)*. Retrieved 4 April, 2013, from thesisterwizard: http://www.thesitewizard.com/general/blogging-pros-and-cons.shtml

Music Library Projects, Proposals. (n.d.). Retrieved 20 March, 2013, from Boston University Music Library: http://www.bu.edu/library/music/about/mk12/

Rai, M. S. (5 April, 2011). *A Case Study on Social Networking Website Development for Music, Arts, Dance and Entertainment*. Retrieved 20 April, 2013, from Ezine Articles: http://ezinearticles.com/?A-Case-Study-on-Social-Networking-Website-Development-for-Music,-Arts,-Dance-and-Entertainment&id=6204034

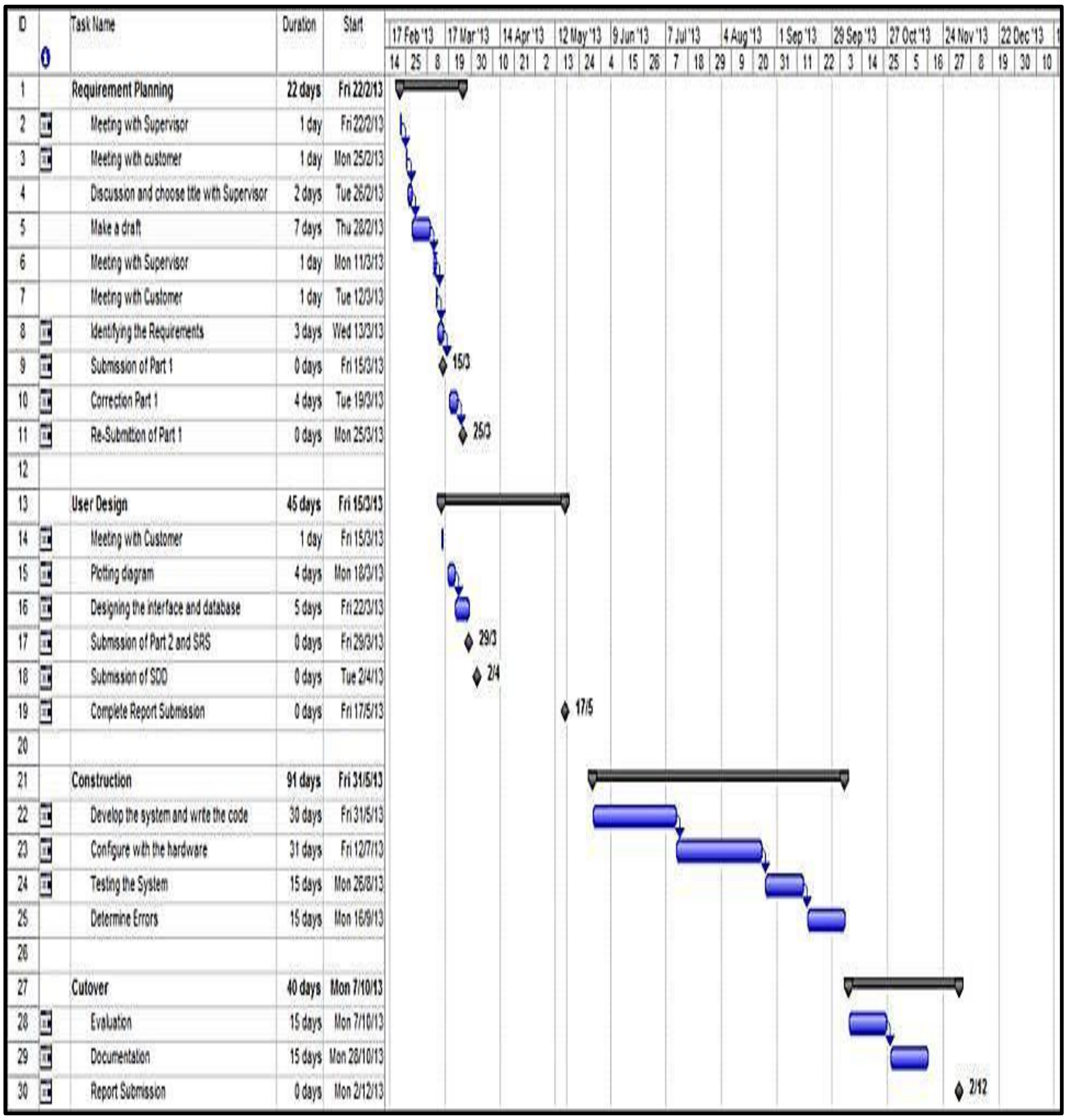
Snell, S. (5 March, 2009). *35 Inspirational Website Designs from the Music Industry*. Retrieved 1 April, 2013, from Vandelay Design: http://vandelaydesign.com/blog/galleries/music-websites/

What pages should be included on a musician's website? (n.d.). Retrieved 7 March, 2013, from The Website Design Studio: http://www.thewebsitedesignstudio.co.uk/pages_for_musicians_website.html

Wikipedia. (n.d.). *Blog*. Retrieved 4 April, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Blog

Wikipedia. (n.d.). *iTunes Ping*. Retrieved 11 April , 2013, from Wikipedia: http://en.wikipedia.org/wiki/ITunes_Ping

APPENDIX A: GANTT CHART



ID		Task Name	Duration	Start
1		**Requirement Planning**	**22 days**	**Fri 22/2/13**
2		Meeting with Supervisor	1 day	Fri 22/2/13
3		Meeting with customer	1 day	Mon 25/2/13
4		Discussion and choose title with Supervisor	2 days	Tue 26/2/13
5		Make a draft	7 days	Thu 28/2/13
6		Meeting with Supervisor	1 day	Mon 11/3/13
7		Meeting with Customer	1 day	Tue 12/3/13
8		Identifying the Requirements	3 days	Wed 13/3/13
9		Submission of Part 1	0 days	Fri 15/3/13
10		Correction Part 1	4 days	Tue 19/3/13
11		Re-Submition of Part 1	0 days	Mon 25/3/13
12				
13		**User Design**	**45 days**	**Fri 15/3/13**
14		Meeting with Customer	1 day	Fri 15/3/13
15		Plotting diagram	4 days	Mon 18/3/13
16		Designing the interface and database	5 days	Fri 22/3/13
17		Submission of Part 2 and SRS	0 days	Fri 29/3/13
18		Submission of SDD	0 days	Tue 2/4/13
19		Complete Report Submission	0 days	Fri 17/5/13
20				
21		**Construction**	**91 days**	**Fri 31/5/13**
22		Develop the system and write the code	30 days	Fri 31/5/13
23		Configure with the hardware	31 days	Fri 12/7/13
24		Testing the System	15 days	Mon 26/8/13
25		Determine Errors	15 days	Mon 16/9/13
26				
27		**Cutover**	**40 days**	**Mon 7/10/13**
28		Evaluation	15 days	Mon 7/10/13
29		Documentation	15 days	Mon 28/10/13
30		Report Submission	0 days	Mon 2/12/13