**Advanced Software Paradigms**

**Final Group Project**

**Report**

Team Name: Spring Flowers

Date: 04/13/2014

Team Member:

Alex Shen

Fei Yan

Junzhi Qian

Rong Zhou

Sunhao Shen

Xinyi Zhu

Yazhou Ye

Yuxin Duan

Zehui Wang

Zheshun Xie

1. [**Introduction 1**](#_Toc385191721)

[**2. Motivation and Goal: 2**](#_Toc385191722)

[**3. Problem Statement: 2**](#_Toc385191723)

[**4. Objective: 3**](#_Toc385191724)

[**5. Strategy: 7**](#_Toc385191725)

[**6. Tactics: 8**](#_Toc385191726)

[**6.1 Create the domain model 8**](#_Toc385191727)

[**6.1.1 Domain Dictionary 8**](#_Toc385191728)

[**6.1.2 Information model 8**](#_Toc385191729)

[**6.1.3 Feature model: 10**](#_Toc385191730)

[**6.1.4 Operational Model 11**](#_Toc385191731)

[**6.2 Requirements 11**](#_Toc385191732)

[**6.2 Reference architecture 11**](#_Toc385191733)

[**6.4 Component library 12**](#_Toc385191734)

[**6.5 Create the domain model 13**](#_Toc385191735)

[**6.5.1 Use Case Diagram: 13**](#_Toc385191736)

[**6.5.2 Control Flow: 15**](#_Toc385191737)

[**6.7 Create the operational model 15**](#_Toc385191738)

[**6.7.1 Data Flow diagram: 15**](#_Toc385191739)

[**6.7.2 State transfer diagram: 20**](#_Toc385191740)

[**6.7.3 Feature Relationship Diagram: 21**](#_Toc385191741)

[**7. Implementation 22**](#_Toc385191742)

[**8. Conclusion 27**](#_Toc385191743)

[**9. Group Assignment. 27**](#_Toc385191744)

[**10. Reference: 28**](#_Toc385191745)

## Introduction

User-generated content sharing has been popular for years along with the rising of the “we-media” concept. We believed that this trend would keep on developing and transforming exiting business models. Our project was focusing on this area – designing and implementation a user-generated music sharing system in the form of a website. This article described the designing and implementation process of this project. It started with introducing motivation and goals of this project, analyzed the problem and the objectives, and then it explained the strategy and tactics and how this system was implemented to solve problems. At last, it expanded on how to apply these paradigms to three other designs.

### **Motivation and Goal:**

As introduced above, we believed that the trend of user-generated content sharing would keep on developing and transforming exiting business models. We plan to develop the paradigms of developing this user-generated content sharing system by designing and implementation a user-generated music sharing system in the form of a website.

### **Problem Statement:**

A domain is an area of interest, usually representing a problem space that is a subset of one or more disciplines. For example, DNA research is a domain. When we construct an application, we are developing software for a problem space—a set of problems to be solved within the domain. Usually, the problem space is a subset of all the problems and their accompanying information within the discipline. Sometimes, a domain may represent an intersection of a number of disciplines, hence we speak of interdisciplinary problems.

A domain that is amenable to software architectures that promote successful reuse must meet the following criteria (Biggerstaff 1993):

* Is well understood.
* Is changing slowly.
* Has intercomponent standards.
* Provides economies of scale.
* Fits existing infrastructure.

The problem space domain of our project is a online user-generated music sharing website, while the solution space domain is that we map regions of the problem space (domains) into domain-specific software architectures (DSSAs). And the problems that our software paradigms solved:

* User registration
* User login in and login out
* Administrators upload music to database
* Management of music in database
* Management of user profiles in database
* Play music online

### **Objective:**

**4.1 Create the domain model**

A domain model is a representation of what happens in the domain (e.g., the domain’s “business operations”). Specifically, it describes the functions to be performed, the entities that perform them, and the data that flows between them and is used or produced by the functions. The objective of the domain model is to standardize the terminology and semantics of the problem space (e.g., the set of problems within the domain that a set of applications will support). These, in turn, provide the basis for standardized descriptions of the problem to be solved within the domain.

The elements of a domain model include:

* A customer needs statement, expressed in the customer’s words, which specifies a high-level set of functional requirements but is often incomplete and ambiguous.
* A set of scenarios that describe the functional sequences, data flows, and user interactions.
* A domain dictionary that defines the terms used in the domain and which is extracted from the previous elements.
* A context block diagram that shows the high-level of interaction between major elements of the system.
* Entity-relationship diagrams describing the aggregation and composition of data entities within the domain.
* Data flow models that show data transfers between active entities.
* State transition models that show how operations change the state of an attribute.
* Object models that capture the attributes and operations of an entity in the domain.

**4.2 Create the reference requirements**

Reference requirements are standard requirements that arise from different functional areas for all of the applications to be built in a domain. Reference requirements define the general functional structure of the problem space (e.g., the functional requirements). They also specify the constraints (e.g., the nonfunctional requirements) on the design and resulting applications. These may include performance, security, user interface styles, and hardware/software platform and programming language requirements. Some requirements may be parameterized so that specific applications will supply the parameters. Some requirements may be allocated directly to components in the reference architecture. For example, the capacity of a database may apply to all applications within the domain.

Application requirements are drawn from particular customer requirements for each specific application. Sometimes, these requirements are refinements of generic reference requirements. However, these requirements may really be design decisions mandated by the customer.

Reference requirements are composed of:

* Functional requirements: the essential operations/processes, their numbers, etc.
* Non-functional requirements: all other functionality of the domain, i.e., security, extendibility, etc.
* Design requirements
* Implementation requirements

Reference requirements may be of:

* Mandatory features: applicable to all systems in the domain
* Optional features: applicable to only subsets of systems in the domain
* Variable features: known to differ in nature by specific application in the domain

**4.3 Create the reference architecture**

A reference architecture is a standard, generic architecture describing all systems in the domain. It may specify both hardware and software platforms and components. Typically, it is represented in a hierarchical, compositional form but may contain multiple trees if it is representing a distributed set of systems for a complex domain. The functional requirements are allocated to the components. The constraints are used to specify some of the characteristics of the platforms and the components. For example, a performance requirement may, upon detailed analysis, require that the hardware platform speed be at least 2 GHz. The reference architecture is instantiated.

A reference architecture contains the following elements:

* A reference architecture model that describes a configuration of the hardware and software components based on an architectural style.
* An architectural schema that describes the high-level structure and constraints of each of the components, including the major data entities.
* A component dependency diagram describing the interactions among the major components.
* A set of component interface descriptions describing the API of each component.
* A set of constraints organized by parameters, value ranges, default values, and effect on the architecture.
* A rationale (prose description) for the particular description.

**4.4 Create the component library**

* Software Components
* Database Schemas
* Ontologies – classes, sets, attribute , relations

**4.5 Create the domain dictionary**

It is composed of domain specific terms.

**4.6 Create the information model**

It is a collection of models built from context analysis and information analysis (domain analysis).

* Context information diagram
* Entity-Relation diagram
* Semantic Network
* Class (object)diagram

**4.7 Create the feature model**

It describes the user/customers expectations of the capabilities of the application in the given domain.

* Feature Relation diagram
* Use Case diagram
* Representation diagram

**4.8 Create the operational model**

It identifies how the applications within the domain work.

* Data-flowdiagram
* Control-flowdiagram
* State Transitiondiagram

### **Strategy:**

**Paradigms:**

A problem paradigm is a model for solving a class of problems. An algorithm represents a solution to a problem. Specifically, an algorithm is a complete, unambiguous procedure for solving a specified problem in a finite number of steps. Algorithms may be deterministic or stochastic. The former always return the same value, while the latter do not necessarily do so.

**Patterns:**

Patterns have their origin in object-oriented programming, where they began as collections of objects organized to solve a problem. There isn’t any fundamental relationship between patterns and objects; it just happens they began there. Patterns may have arisen because objects seem so elemental, but the problems we were trying to solve with them were so complex.

Brad Appleton (Appleton 1999) notes that others have made distinctions in conceptual levels of patterns, such as Buschmann et al. (1996) who defines three types of patterns:

Architectural Patterns: An architectural pattern expresses a fundamental structural organization or schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.

Design Patterns: A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes commonly recurring structure of communicating components that solves a general design problem within a particular context.

Idioms: An idiom is a low-level pattern specific to a particular programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language.

**Frameworks:**

One concept closely related to design patterns and components is a framework. We’ll define it as follows: a software framework is a reusable mini-architecture that provides the generic structure and behavior for a family of software abstractions, along with a context of metaphors that specifies their collaboration and use within a given domain. Software frameworks are also referred to as application frameworks.

**Components:**

We see component software as an object-based software model aimed at efficient and incremental software development. The main idea is to break monolithic applications into reusable, binary components that can be developed, distributed, and upgraded independently. To accomplish this, we need to provide standard mechanisms for interoperability between applications and components. If components can interoperate, they can be combined to build larger applications in a flexible and incremental way.

### **Tactics:**

* 1. **Create the domain model**

The Domain Model of this project consists of four main components: Domain Dictionary, Information Model, Feature Model and Operational Model. The figure below explains the elements of Domain Model.

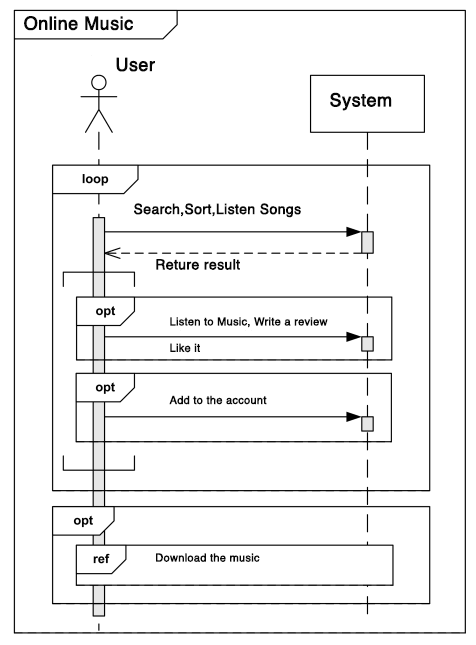
**6.1.1 Domain Dictionary**

In this case, the Domain Dictionary includes the structure .NET Framework, the functionality, a Recommendation System and two terminologies, object-oriented language and information filtering system.

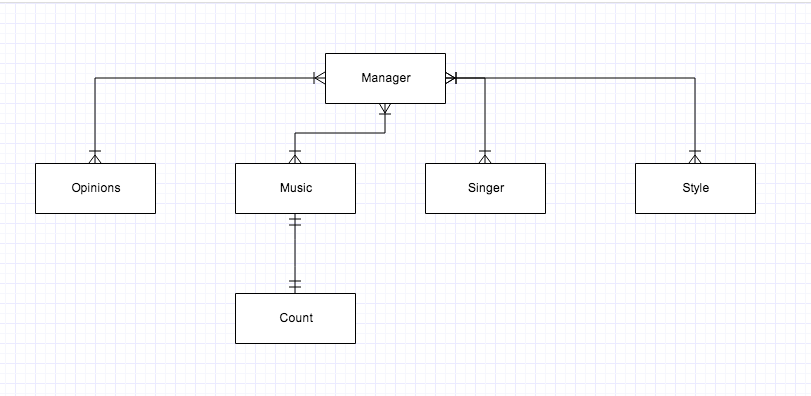
* + 1. **Information model**

By different models, the information model is able to describe the context of the software. The models used in this project are Context-Information diagram, Semantic Network, Feature-relation diagram and Representation diagram.

1. **Context Diagram**



1. **Semantic Network**



1. **Object Diagram**

|  |  |  |
| --- | --- | --- |
| Object | Attributes | Operations |
| Manager | Username  Password  Permission | Manage the platform |
| Opinion | Review Content  Username  Attitude | User review: save  Display,  Review,  Analyze |
| Music | SongName,  Style | Listen |
| Singer | Singer Name | Rank |
| Style | Style Name |  |
| Count | Listen Sum, Download Sum | User Likes it.  Display  Save |
|  |  |  |

* + 1. **Feature model:**

Feature model is designed to describe the capabilities of the application that will fulfill the user’s expectation. In this project, the model leverages Feature Relation diagram, Use case diagram and Representation diagram.

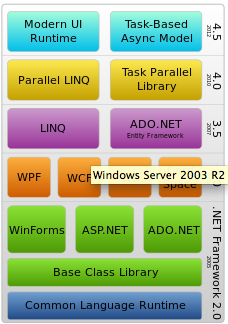
User expectation of the capabilities of the application in the given domain

**6.1.4 Operational Model**

Operational Model explains the application’s data flow, control flow and state transition. It is the guide for software engineers to design their framework and software architecture.

**6.2 Requirements**

* Users can listen, watch, like, collect and download songs, albums and music videos.
* Users are able to sort the music list by categories, and rate the songs, albums, singers and artists
* Users can know what songs their friends are listening throughout the social media.
* The music recommendation system is designed by both collaborative filtering and content-based filtering.
  1. **Reference architecture**
* The reference architecture in this project is .NET framework. The .NET Framework provides user interface, data access, database connectivity and web application development and network communications. The project is built in the Visual Studio, which is an integrated development environment produced by Microsoft. It helps the team to develop a modern web application with cutting edge technologies, unparalleled cross-browser compatibility and an advanced Model-Control-View architecture.



The Framework Stack

**6.4 Component library**

The component is a combination of elements that creates a purposeful, reusable, and independent structure. This application is built by Web From, which is the heart and soul of ASP.NET. (msdn.microsoft.com/en-us/library/ms973868.aspx#introwebforms\_topic1)

Two components are applied in this application: the visual portion and the model behind the form, which resides in a separate class file.

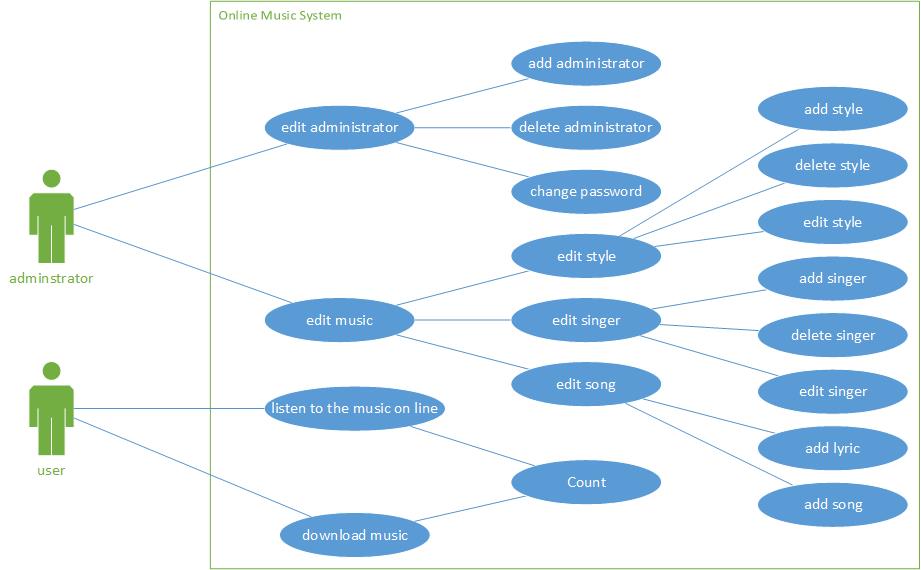
The visual portion includes two types of controls for creating the user interface: HTML controls and Web Form controls

HTML controls include HTML elements and the client-side events. The Web From controls contains a Server-Side control that is one to one correspondence with their HTML elements.

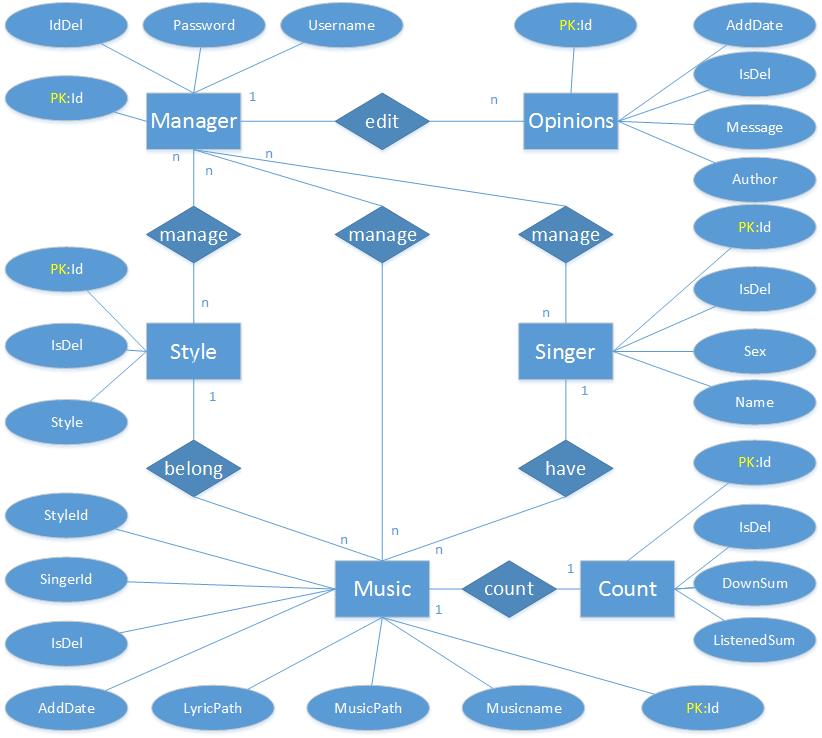
The model .NET components run on the web server and deliver the data and other services to the Web Service operating in the server. Each model is a class implemented the one or several functions requested by database or the user interface.

* 1. **Create the domain model**
     1. **Use Case Diagram:**

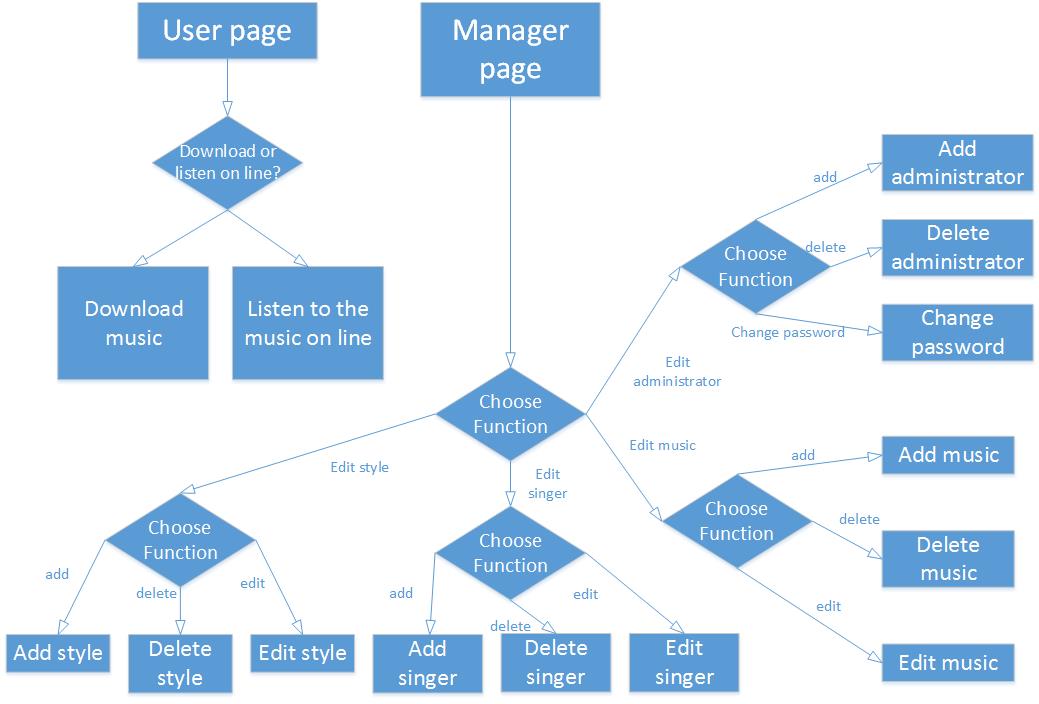
**Use Case:**



ER:



* + 1. **Control Flow:**

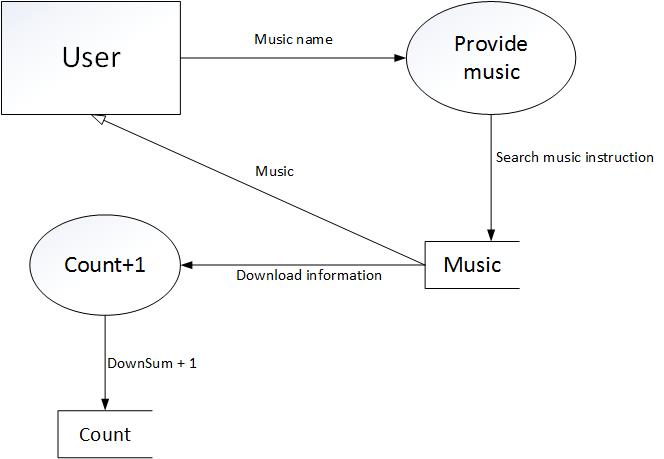


**6.7 Create the operational model**

**6.7.1 Data Flow diagram:**

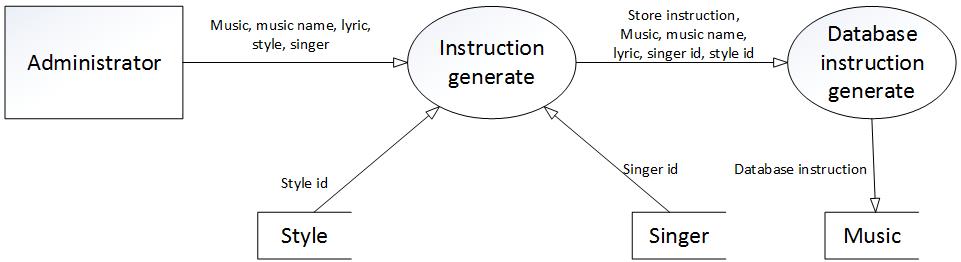
a: Download music: (the same as the listen to the music on line)

User choose the music they want to download. Then the user page will submit a “music name” data to the Logic Layer. Then download music function will proceed this data and result a search music instruction data. The search music instruction data will transfer to the Data Layer. According to this instruction, the music database will provide the music to the user and invoke the count function to calculate the download time.

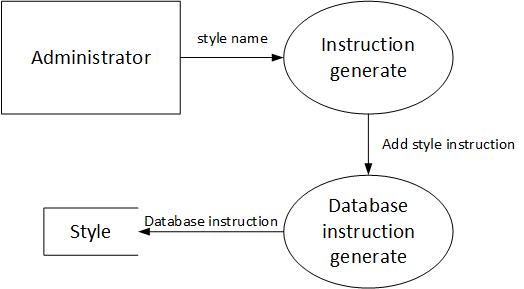


Edit function only can invoke by the administrators. After they select a mange function, the Logic Layer will receive correlate data from administrators’ page (UI Layer) and provide a corresponding instruction to the Data Layer. In the same time, according to the function they choose, Data Layer will also provide the data to assist the instruction generate function (Logic Layer). Then the Database instruction generate function (Data Layer) will manage the data base according to the instructions.

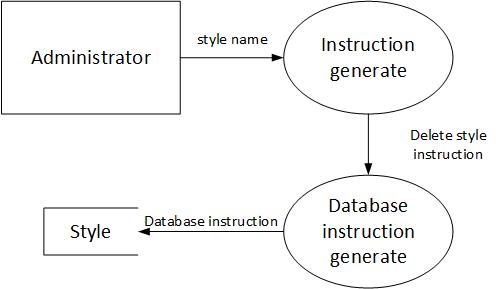
b: Store Music:



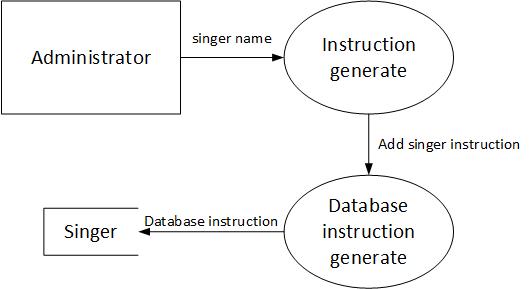
c: Add Style:



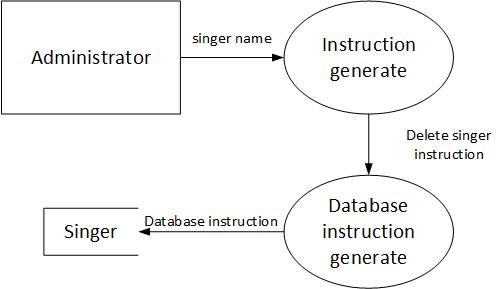
d: Delete Style:



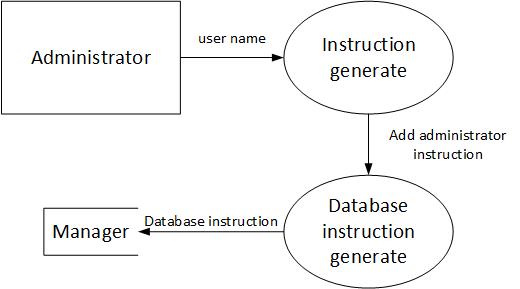
e: Add Singer



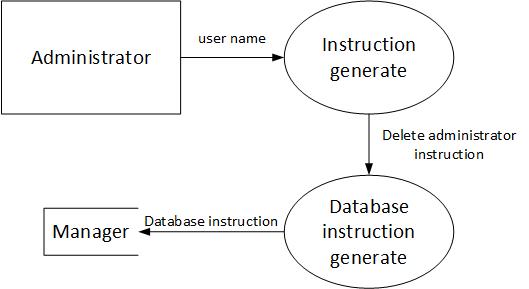
f: Delete Singer:



g: Add administrator:



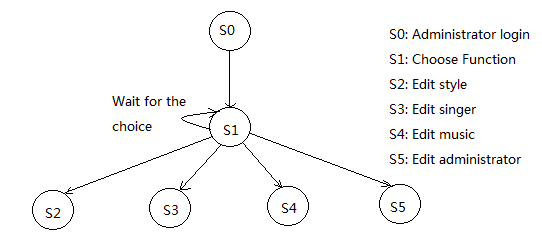
h: Delete administrator:



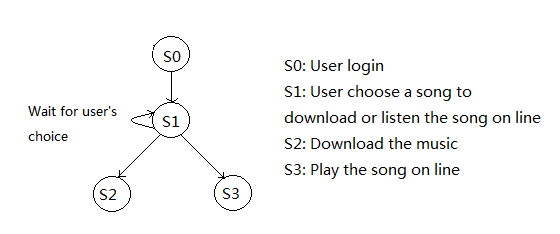
**6.7.2 State transfer diagram:**

a.Admin:

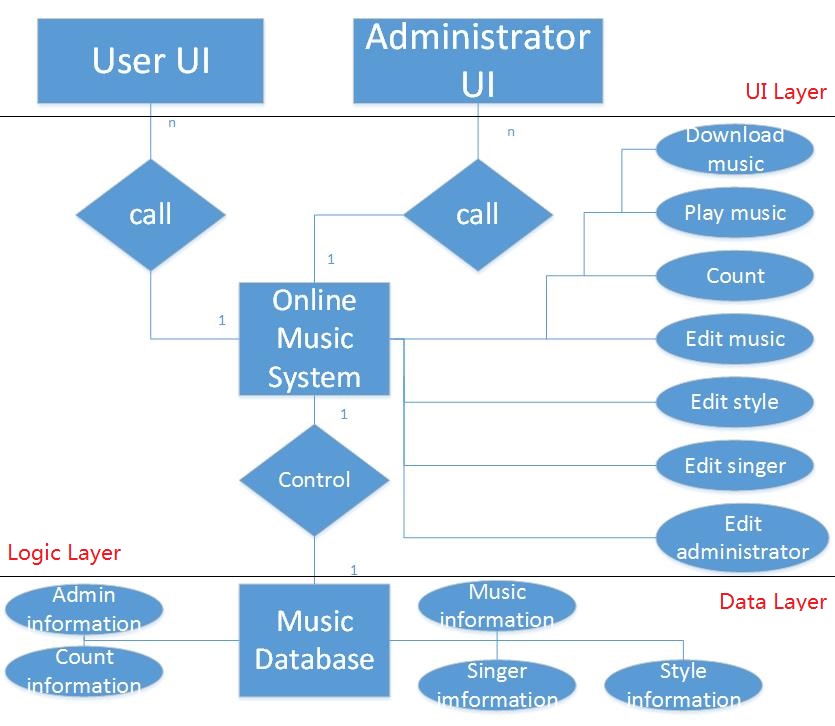
The administrator has six operation statement. They represent six different condition of the administrator operation. The statement will start at administrator login.



b.User:



**6.7.3 Feature Relationship Diagram:**



1. **Describes overall mission operations of a system**

User click on the download button or listen on line button to choose the function they want to invoke.

Administrator must input their user name and password to login. Then they can access to the management page. Clicking on different button to choose the edit function.

1. **Describes major features and decomposition:**

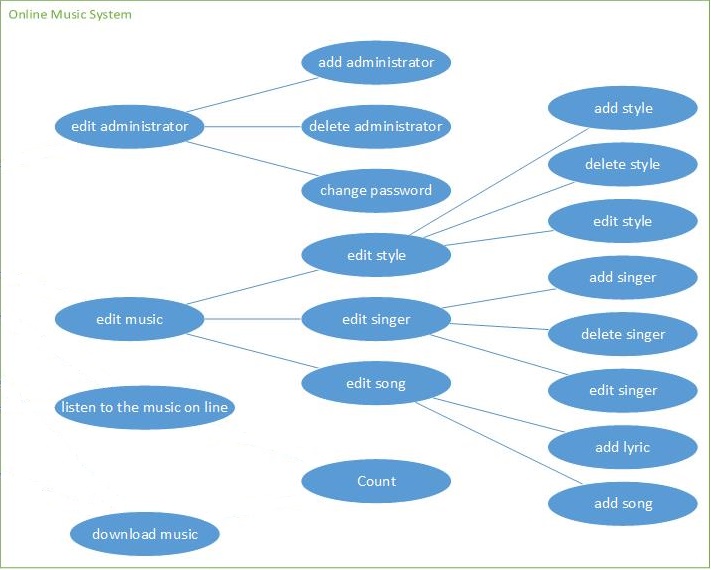
User don’t need to login. They can invoke the download music and play music function.

Administrator login in to manage the Edit function.

Edit function include edit music, edit style, edit singer, and edit administrator.

Count function is invoke after user using the download music and play music function.

The Logic Layer is correlation with the space domain.

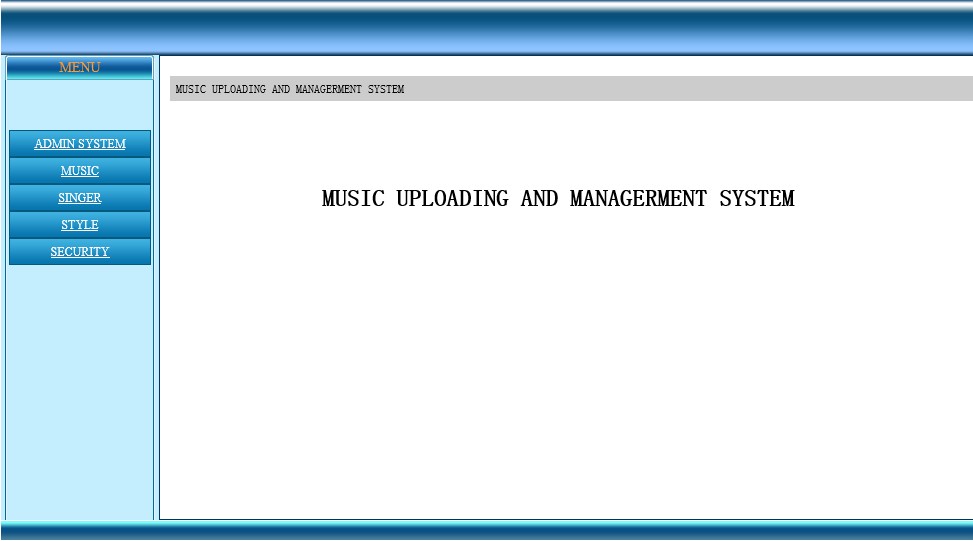


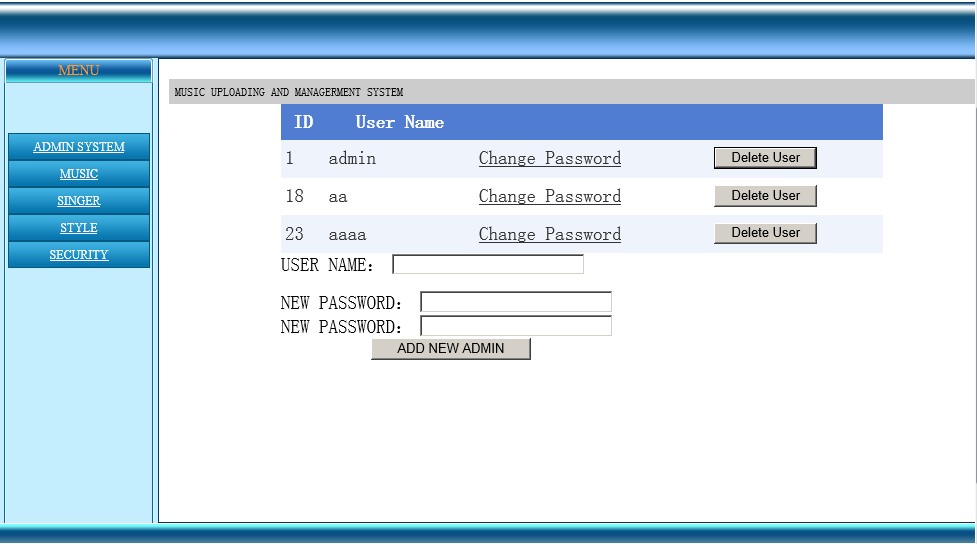
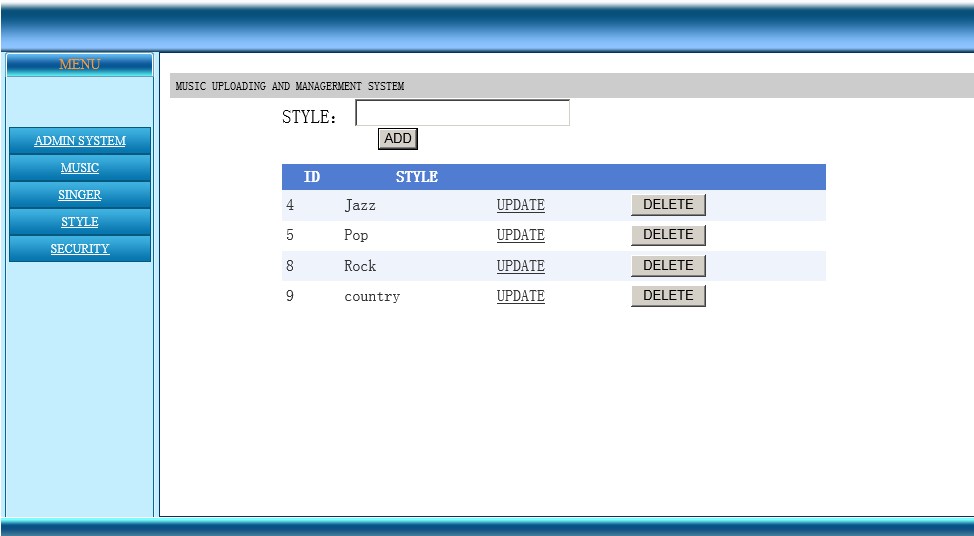
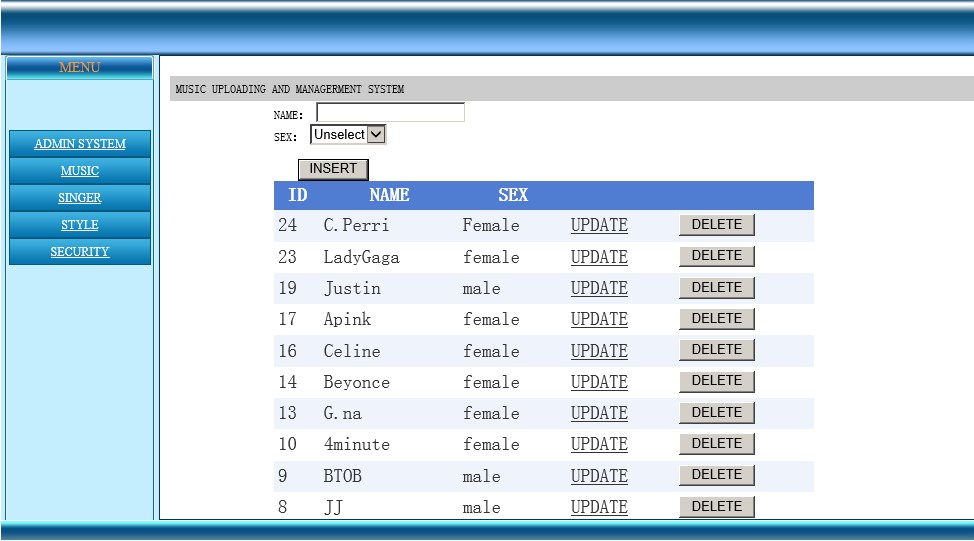
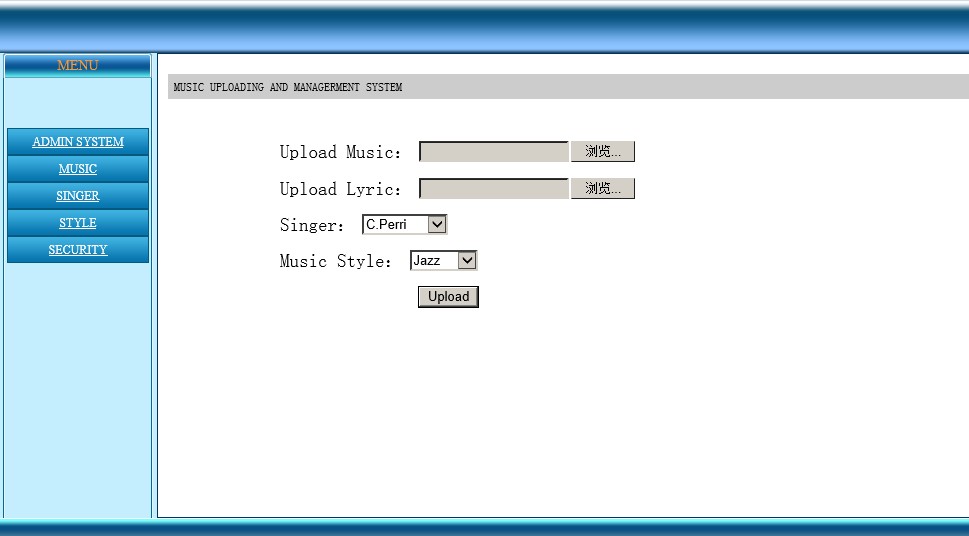
## Implementation

In this part, this report will show us the picture of the systems. Including the Admin System page, the homepage , the play and the download page.

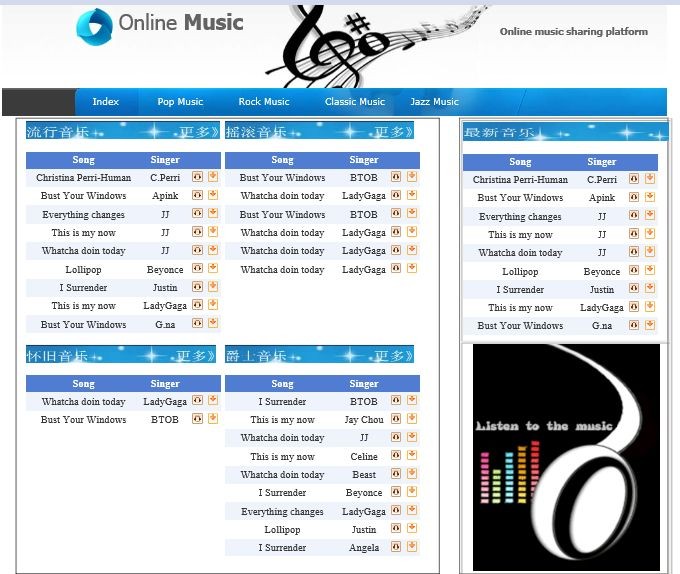
**Below is our Admin System:**







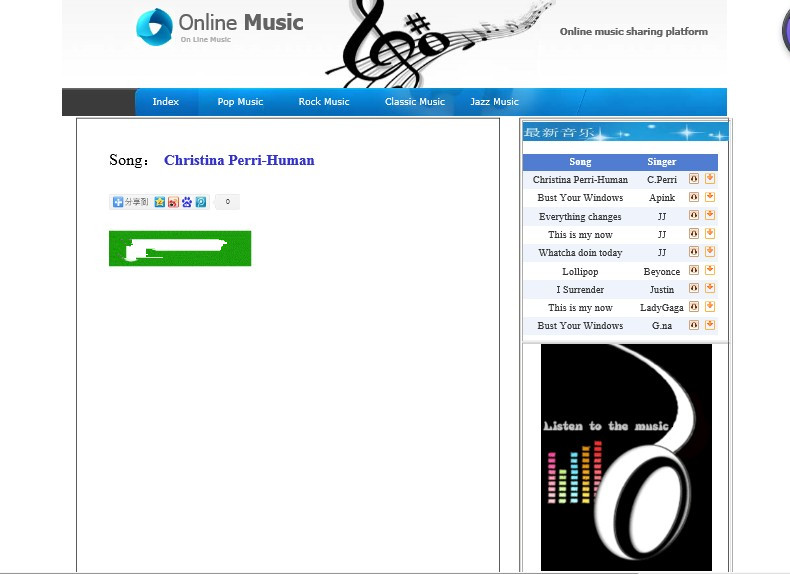
And below is the system home page:



This is our player:



And This is the download page:



## Conclusion

The designed user-generated music sharing system provided several paradigms for user-generated content sharing system, and even could be extended to the field of social websites. The reuse of different modules would greatly shorten the developing time of other systems. These modules combined together to produce other systems, and also the single one of them could be take out for composing other not-that-similar system. This is the beauty of software paradigms.

## Group Assignment.

**Project Part:**

**We have three groups:**

a. First group is design the web page, design this online music system. Using the c#, and asp to design the basic page. (FEI YAN,YUXING DUAN,RONG ZHOU, XINYI ZHU)

b. Second group is the database collection. (ALEX ,ZEHUI WANG,JUNZHI QIAN)

c. Third group is building the network platform to let the system work.

(SHUNZHE XIE, SUNHAO SHEN,YAZHOU YE)

**Report part:**

**We have five groups works together.**

a. Introduction, Conclusion, Combine PPT. (FEI YAN, YUXING DUAN)

b. Motivation and Goal, Problem Statement, Objective, Strategy(RONG ZHOU, XINYI ZHU))

c. Tactics Part I (ALEX ,ZEHUI WANG)

d. Tactics Part II: (JUNZHI QIAN,SHUNZHE XIE)

f. Implementation, combine all the parts together, Check Appendix(SUNHAO SHEN, YAZHOU YE)

## Reference:

Varma, M. C. A Reconnaissance on Design Patterns 2013

Khwaja, Salman. Towards design pattern definition language 2013

Meyers, S. (1995). More Effective C++: 35 new ways to improve your programs and designs. Pearson Education.

Sarwar, B., Karypis, G., Konstan, J., &Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). Introduction to algorithms (Vol. 2, pp. 531-549). Cambridge: MIT press.

(msdn.microsoft.com/en-us/library/ms973868.aspx#introwebforms\_topic1)Introduction to ASP.NET and Web Forms: Paul D. Sheriff DSA.Inc November 2001