UPPSALA
UNIVERSITET

IT 18 040
Examensarbete 45 hp
September 2018

Design and Implementation of a Web-Based System for Photo Sharing and Photography Competition

Pan Zhang

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Design and Implementation of a Web-Based System for Photo Sharing and Photography Competition

Pan Zhnag

With the development of tourism and the improvement of people's life, photography is becoming more and more popular. Photography has also become a major branch of entertainment and hobbies. At present, there are a lot of photography enthusiasts in China. They desire to have a good and easy-to-use platform to share hight quality photos and to participate in photography competitions on the Internet. In order to meet the demands of students and teachers in Tongji University, this project was setup 3 years ago, whose aim was to design and develop such a system for photo sharing and photography competition management.
This paper describes in detail the design and implementation of the system, which is a web-based photo sharing, photo management and photography competition management system. Firstly, we made investigation of requirements and finished requirement analysis by using UML. Secondly, we designed the architecture of the system. Thirdly, we designed the processing logic and steps for each module, and drew sequence diagrams for all models. We also finished the design of the data tables of the system and the E-R diagrams are given in the paper. Fourthly, an underlying storage space management module is proposed and designed for more efficient and flexible storage management for the photos, and to enhance the efficiency of data synchronization and transmission. The algorithm is carefully designed and implemented. Finally, we implemented the whole system by programming and testing, and put the system into trial operation. Screen shots of the user interfaces of the system, the code lines and the test results are given in this paper.
The system is designed and implemented by myself, using MVC for analyzing and designing, J2EE for programming, and MySQL for database management.

Handledare: Chenxi Zhang
Ämnesgranskare: Lars Oestreicher
Examinator: Mats Daniels
IT 18 040
Tryckt av: Reprocentralen ITC

CONTENTS

Chapter 1 Introduction

1.1 Project Background

Along with the improvement of people's life quality in China, tourism has become more and more popular, and it has become one of the major entertainments for people spending their leisure time. Both domestic tourism and outbound tourism has become increasingly common in recent years. According to statistics of the China National Tourism Administration, about 4.12 billion people went on domestic travel or outbound travel in 2015 [1]. With the rapid development of the tourism, the number of people who use photos to record the beautiful moments in their life is also increasing, and many people have become very fond of photography [2].

Internet is rapidly developing and people's lives are constantly changed by it. The Internet not only allows people to get the information they want easily, but also builds a strong social network. Through the structure of social network platforms and hubs, people can quickly and easily communicate with others, share their life moments, and so on.

Nowadays, many photography enthusiasts expect to have a platform to share high-quality photos with friends and manage their online storage photos more efficiently and conveniently. Photography enthusiasts are also keen to participate in photography competitions to get encourage from others and find their own shortcomings. However, most of the online photography competition websites mainly focus on their own competitions, which have a lot of limitations. This kind of website has quite a big difficulty to meet various needs of photography enthusiasts at different levels of expertise, and the number of systems containing both photo sharing and general photography competition management are quite a few in China.

At the same time, with the development of technology and photographers' higher demand for photo resolution, the system needs a more efficient way to manage storage space. Therefore, it is imminent to design a web-based photo sharing and photography competition management system for photography enthusiasts, which has more efficient storage space management.

1.2 Research status at China and abroad

Most of photo sharing or photography competition platforms appeared around the early 2000, and have become increasingly popular in recent years.

Flickr and *Imgur* are well-known international photo-sharing websites. *Flickr* has social contact and photo management functions, allowing users to easily share photos with their friends, interact with other users, and effectively manage uploaded resources. *Imgur* is more inclined to resource storage and image search platform. So uploaded images can be searched and viewed by other users, and the system will record the viewing number of these images, and push images, which have a large click numbers, to the website's homepage. *WPO (World Photography Organization)*, *World Press Photo* and *POYi (Photos of the Year International)* are internationally famous photography competition websites, but they are mostly established for their own competitions.

In China, most famous photo-sharing websites belong to the kinds of forum website, resource website or community website, such as *POCO* and *Fengniao*. Both are community photo-sharing websites with forum and photo viewing functions. Their users can post text to communicate with other users, upload photos to a unified photo library for sharing, and simply manage uploaded photos. *Global Photography* and *Photo Inter* are prestigious photography competition websites in China. They both have photo viewing and photography competition management functions, so users can upload photos to a unified photo library and view them online, or participate in the designated competitions after payment. At present, there are relatively few platforms containing both photo sharing and photography competition management functions.

The aim of our project is to build a system for photographers with excellent photo sharing, photo management and photography competition management functions. Also, a good storage space management subsystem is to be built for efficient resource use and system expansion, which will be incorporated into the system.

1.3 The purpose and significance of this project

With vigorous development of tourism and the change of people's viewpoints,

recording and enjoying excellent moments with photos has become more and more popular. Therefore, a large number of photography enthusiasts appeared. Photography enthusiasts are eager to communicate, share high-quality photos with others, and participate in photography competition to find out their own shortcomings or get approval from others. However, the platforms, which contain both photo sharing and photography competition management, are relatively few at present. So the system is developed to meet these needs, which not only allows people to upload and share high-quality photos more easily, but also narrows the distance among photographers and photography enthusiast by interaction on the system, allowing enthusiasts to learn from each other. This system also allows all photography enthusiasts to participate in, and get more evaluation, encouragement and support from other people.

A good photo sharing and photography competition management system must be intuitive, convenient and easy to use. Users with different backgrounds can use its functions efficiently. At the same time, a better solution is needed to make the storage space management subsystem more efficient, stable and expansible, because of the storage of large number of high-quality photos and a lot of potential users. The project aims to develop a photo sharing and photography competition management system for photographic enthusiasts by using J2EE, MVC and other related technologies. We also realized the system's easy using, security, efficiently and expansibility. At the same time, the realization of the system not only has great potential value of commerce, but also promotes spread and development of photography.

1.4 Main work of this paper

The main contribution of our project is the design and implementation of the photo sharing and photography competition management system based on the J2EE and MVC, which adopt MySQL as its relationship database management system. We finished the following work:

1. Research was conducted on mainstream photo-sharing websites and photography competition management systems existed on the Internet. And we made an in-depth investigation on requirements from photography enthusiast groups, which are very helpful in requirement analysis.
2. Requirements analysis was done for the photo sharing and photography competition management system by using UML and object-oriented method. The use case diagrams under various cases were drawn.

3. The overall design of the system was carried out. The system was divided into two subsystems: photo sharing and photography competition management. The former subsystem consists of 5 parts, which are the *Personal Homepage*, *Photo Management module*, *Album Management module*, *Collection Management module*, and *User and Authorization Management module*. The latter consists of 6 modules, which are *User and Authorization Management module*, *Competition Management module*, *Review module*, *Contribution module*, *Display of Competition module*, and *Display of Competition Results module*.

4. All modules were designed in detail in their functions and processing logic, and the sequence charts for them were drawn, also giving detailed processing steps.

5. The database's design was accomplished, and E-R charts and database tables were given. We used a tree structure in the data classification. The tree structure can build good relationship and structure among resources and records in the database [3].

6. A unique storage space management module was proposed and designed, it not only provides efficient and flexible management of storage on the server, but can also fulfill efficient file synchronization and file transmission.

7. The UI of the system was designed and implemented. Each module of the system was coded and debugged by using Java and MyEclipse, and then system integration and testing are carried out.

1.5 The structure of this paper

Chapter 1, as the introduction, mainly discusses the project background, current status at China and abroad, and the purpose and significance of this project. It is stated that the main work and main research content of the project is the design and implementation of a web-based photo sharing and photography competition management system.

Chapter 2 is the requirement analysis of the photo sharing and photography competition management system, which analyzes functional and nonfunctional requirements of the system. Use case diagrams under various cases were drawn.

Chapter 3 describes the design of the photo sharing and photography competition management system, including overall design and detail design. Firstly, the system is divided into 2 subsystems, which contains 5 and 6 modules respectively. The function

of each module is described. Then, the processing logic and steps of each module is given by using sequence diagrams. Later on, the user management mechanism, directory tree and storage space management subsystem are described. Finally, database design of the system is described, and detail information of database tables and E-R charts are given.

Chapter 4 is the realization of the system, which describes the details of realization of each module. User interface screenshots and part codes are given.

Chapter 5 describes the test of the system. Test cases and test results are given.

Chapter 6 summarizes the work done in this paper, and describes future work, including possible improvement and extension of this system.

Chapter 2 System Requirement Analysis

2.1 Functional requirement analysis

First of all, requirements of this system are analyzed. After investigation and analysis of related websites at China and abroad, we confirm that the system should consist of two subsystems: the photo sharing subsystem and the photography competition management subsystem, which play different roles, but are closed related with each other.

2.1.1 Photo sharing subsystem

The photo sharing subsystem should provide a variety of functions for users to share photos easily and flexibly, allowing them to upload photos, to manage the uploaded photos, and to share photos with other users. It should also provide a variety of operations to easily create and manage their own albums.

Based on this analysis, we determine that the subsystem contains five parts: *Personal Homepage*, *Photo Management*, *Album Management*, *Collection Management*, and *User and Authorization Management*.

1. Personal Homepage

The *Personal Homepage* is the “Home” of a user, for others to visit over Internet. When visitor visit a user's homepage, the page will automatically show the user’s personal information, representative photos, selected album and user customize, which also functions as a guide for browsing all his photos and information. Visitors can also comment on his photos and albums.

2. Photo Management

Users can upload, delete, move and copy photos. When uploading photos, they need fill in metadata of the photos and can later modify the metadata. The purpose of the metadata is to provide for better sharing and searching of the photos, through key words, for example. At the same time, the user can also choose specified photos to participate in the ongoing photography competitions.

3. Album Management

Album Management allows users to create and manage album, the operations include

creating, deleting, removing and modifying albums. It also allows users to manage photos inside an album.

4. Collection Management

When viewing other people's photos or albums, a user can add his interested objects into a collection for future access. He can manage his collection using the functions provided by *Collection Management*. Operations should include: adding collection items, deleting collection items and viewing collection items.

5. User and Authorization Management

As an important part of the system, *User and Authorization Management* manages users information and access permission, which is jointly shared by the photo sharing subsystem and the photography competition management subsystem. It provides users with the functions of registration, login and logout. It also allows users to modify his password and personal information, which mainly contains name, gender, avatar, contact information.

2.1.2 Photography competition management subsystem

This subsystem mainly realizes the functions related to photography competition management.

Through this subsystem, super administrators can easily set up and manage all photography competitions. He or she can assign some users as competition administrators for each competition. The competition administrator can manage a designated competition. Operations that can be performed include: modifying competition information, modifying review results, and locking review results. The photography competition management subsystem consists of 6 parts, which are: *Competition Management*, *Contribution*, *Review*, *Display of Competition, Display of Competition Results*, and *User and Authorization management*.

1. Competition Management

The *Competition Management* has two kinds of functions: competition management and photo management. The super administrator is responsible for managing all competitions. Operations include: creating, modifying, deleting and other operations. The competition administrator can only be responsible for a specific competition. He can delete photos and block an account when the user uploads inappropriate photos (such as a reactionary or pornographic photo).

When super administrator establishes a new competition, he should set the

competition name, start date, end date, reviewer and competition administrator.

Users can find the competitions they have taken part in through the *Competition Management*, and check the photos they have submitted.

2. Review

Competition reviewers can select, evaluate and reward photos.

3. Contribution

User can submit photos to ongoing photography competitions by uploading a photo and fill in description.

4. Display of Competition

The users and visitors can view all the photos of all competitions. While viewing, users also can set different sorting orders for display, and give like's (similar to the like's in facebook.com) to their favorite photos

5. Display of Competition Result

Display of Competition Result is used to display rewarded photos and relevant competition information for users and visitors. They also can click "like" buttons for them.

6. User and Authorization Management

User and Authorization Management module is jointly shared by photo sharing subsystem and photography competition management subsystem. See Section 2.1.1 for more details.

2.2 Non-functional requirement analysis

Non-functional requirement constitute an important component of requirement analysis, they not only enable developers to have a more detailed and accurate understanding of the system, but also improve product quality and make the development more efficient [4].

2.2.1 Feasibility

Due to the diversified background of different users and the overall goal to make the system simple and user-friendly, the system's interface is designed to be concise and clear. We use JavaScript and a tree structure to achieve a visual directory tree in user interface page. In this way users can easily achieve the selecting of path, and this improves system usability and operational efficiency [5].

2.2.2 Scalability and maintainability

The MVC framework is used in order to make the system have good scalability and maintainability, which makes the system's view layer, logical business layer and data manipulation layer to be independent of each other, and also makes it easy to do future development and system maintenance [6].

2.2.3 Security

In order to guarantee the security and integrity of the whole system, it is necessary to verify the user's access level and login status again while logged-in users trying to modify important information (such as password) or logged-in administrators trying to execute high-privilege operations. This kind of examination can avoid malicious operations and provide Anti-Leech.

2.2.4 System performance

It is easy to waste the storage space if it is not effectively managed, because the system mainly stores large-sized high quality photos, and uploading and deleting operations can happen very often. Besides, if the client and the server frequently interact with large amounts of data, a lot of bandwidth will be consumed, resulting in slower respond, worse user experience and maybe other problems. Therefore, we will use AJAX to refresh only part of the page elements, instead of all the elements. It can improve the response speed and user experience, and reduce the burden on the server [7]. We will also incorporate a specific storage space management subsystem to optimize synchronization and transmission process for files, and to make management of the server resources more efficient.

2.3 Use case analysis for the system

2.3.1 Use case analysis of User and Authorization Management module

The *User and Authorization Management module*, as a basic module of the system, provides the registration function to visitors and allows registered users to manage their account. Visitors do not have the relevant account information and operational authority. Only registered users can use all these functions. Use case diagram of this module is shown in Figure 2.1, in which functions of each role are shown. They are:

1. Visitors can register an account.

2. Registered users can log in to the system and log out of the system.

3. Registered users can change or retrieve their own password.

4. Registered users can update their personal information, including the avatar, cover, basic information and contact information.

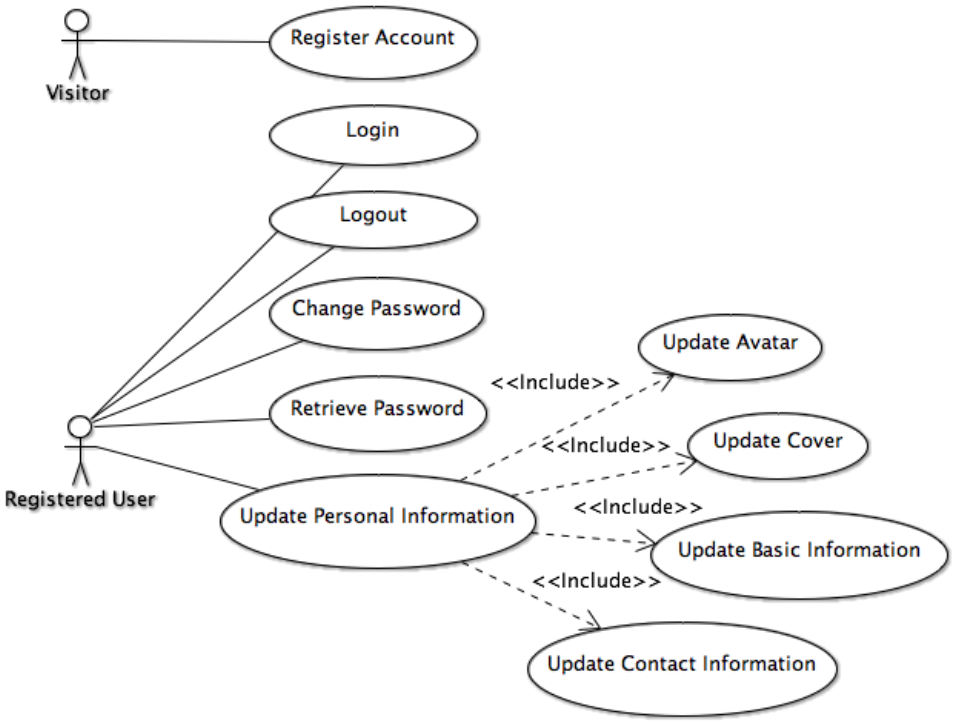


Figure 2.1 Use case diagram of User and Authorization Management module

2.3.2 Use case analysis of Personal Homepage

The *Personal Homepage*, as one of the most important parts of the system, mainly aims to provide photo sharing and photo displaying for users. Only registered users have personal homepages. Visitors can visit any user's personal homepage and browse his photos, albums and collections without registering or logging in. However, in addition to browsing photos, the visitors have no authority to perform any other operations.

The use case diagram of *Personal Homepage* is shown in Figure 2.2. The functions of each role are as follows:

1. Registered users and visitors can visit and browse any users' personal homepage.

2. Registered users and visitors can also browse any users' resources (photos, albums and collections).

3. Registered users and visitors can jump to other users' personal homepage by searching their nickname.

4. Only registered users can perform collect, follow and comment on other users' resources.

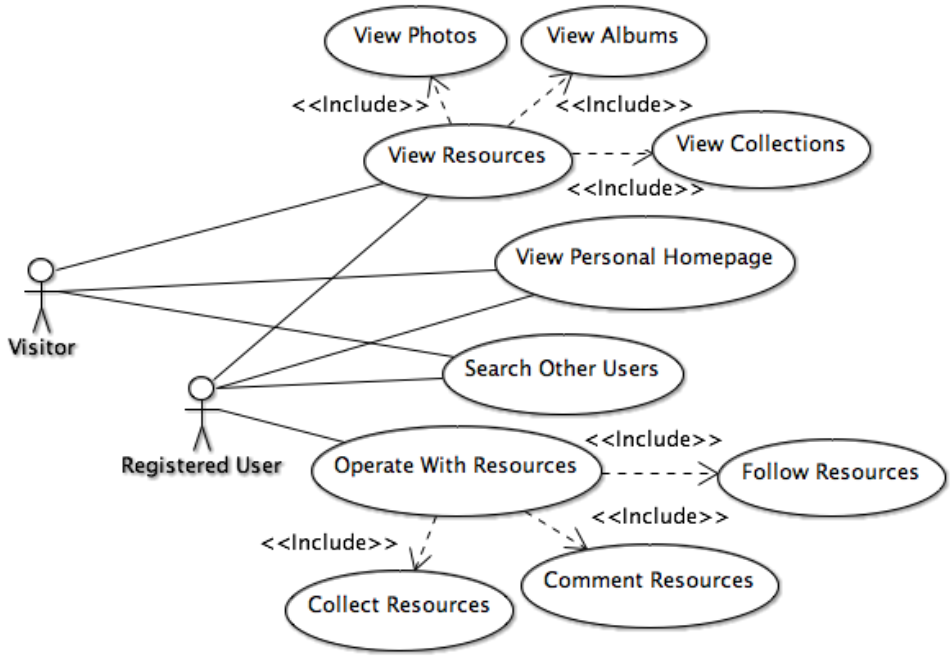


Figure 2.2 Use case diagram of Personal Homepage module

2.3.3 Use case analysis of Photo Management module

The *Photo Management module* is mainly used to manage users' photo resources. The use case analysis diagram for this module is shown in Figure 2.3, the functions of each role are as follows:

1. Registered users can upload photos into the system, by selecting the source files and inputting information, which describes the photos. Each photo is regarded as resource in the system.

2. Registered users can manage the photos. The operations that can be performed include: moving photos, copying photos, deleting photos and modifying photo information.

3. Registered users can also setting the sort mode for photos displaying, including sorting by upload time, sorting by modified time, sorting by the

number of likes and sorting by the number of views

4. Registered users can take part in competitions after select uploaded photo and select specific competition.

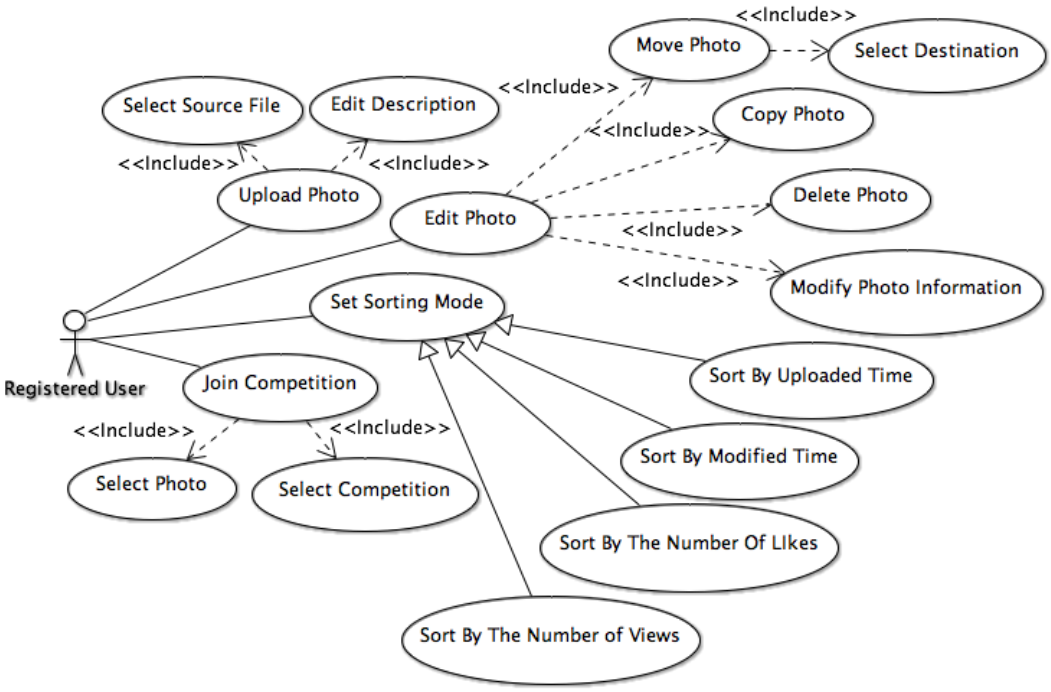


Figure 2.3 Use case diagram of Photo Management module

2.3.4 Use case analysis of Album Management module

The *Album Management module* contains not only all operations and management functions of the album resources, but also the majority operations of the photo resources. The use case analysis diagram of the *Album Management module* is shown in Figure 2.4, the functions of each role are as follows:

1. Registered users can create a new album.

2. Registered users can edit the album, including moving the album, copying the album, deleting the album and updating the album information.

3. Registered users can also set the sorting method for albums, including sorting by created time, sorting by modified time, sorting by the number of likes and sorting by the number of views.

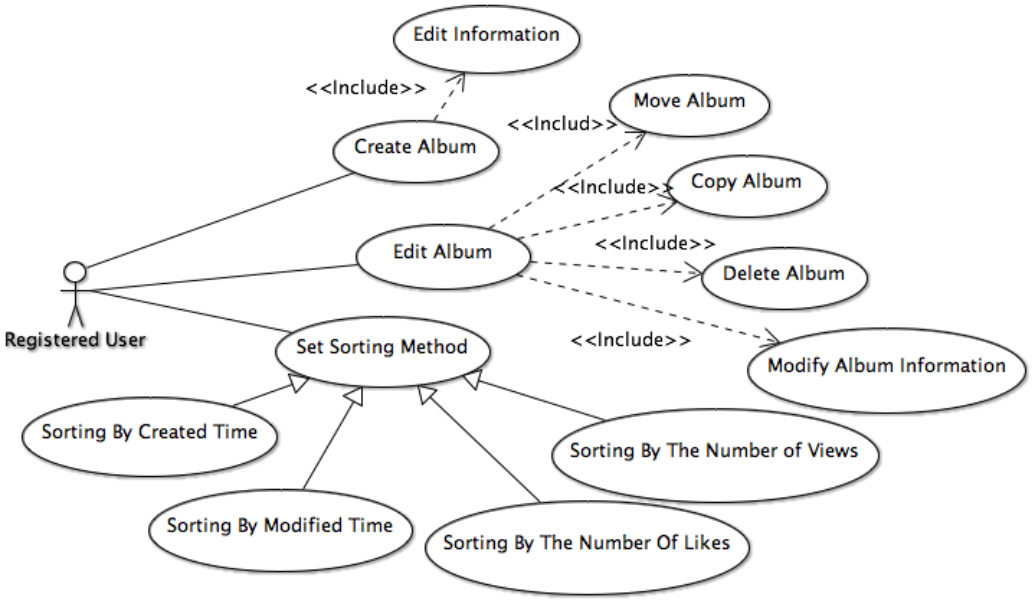


Figure 2.4 Use case diagram of Album Management module

2.3.5 Use case analysis of Collection Management module

Similar to the previous two modules, this module contains operations and management functions of the collections. The use case analysis diagram of the *Collection Management module* is shown in Figure 2.5, in which functions of each role are as follows:

1. Registered users can add their favorite photos or albums into a collection.
2. Registered users can delete collected items from their own collection.
3. Registered users can view their own collection items by clicking the target in the collection management page.

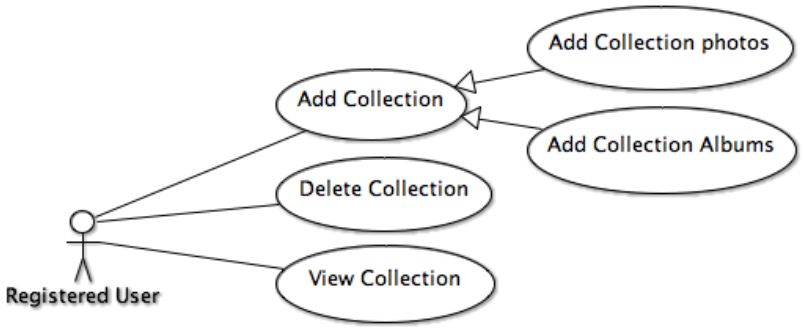


Figure 2.5 Use case diagram of Collection Management module

2.3.6 Use case analysis of Competition Management module

The *Competition Management module*, which is also an important part of the system, mainly manages affairs related to photography competitions. The administrators manage competitions through this module, and a normal user checks his participated competitions through this module too. The use case diagram of the *Competition Management module* is shown in Figure 2.6, in which functions of each role are as follows:

1. Registered users can check their participated competitions through this module, including competition information and contributions (uploaded photos).
2. The super administrator can create and delete competitions.
3. The super administrator can update the competition information, including the name, date, reviewer, and competition administrator.
4. Both the super administrator and the competition administrator can delete photos and lock users. The super administrator can delete photos from all competitions, but the competition administrator can only delete photos included in the competitions under his control.

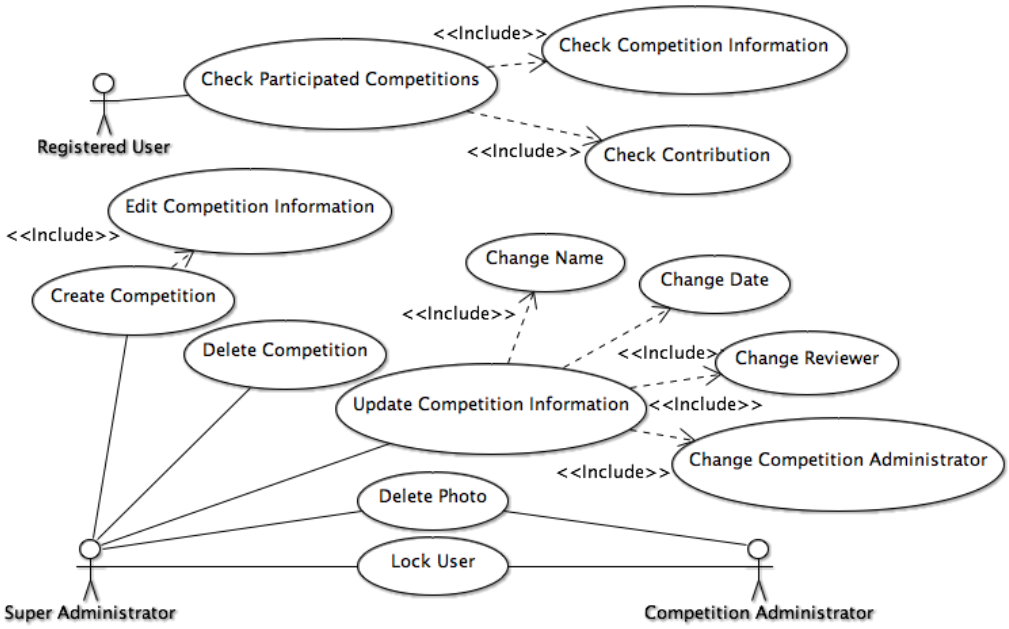


Figure 2.6 Use case diagram of Competition Management module

2.3.7 Use case analysis for Contribution module

Registered users can submit photos for competition through this module. The functions of this module include viewing the competition information and contribution, which is done by uploading photos and providing information about the photos. The use case diagram of the *Contribution module* is shown in the top part of Figure 2.7, in which functions of each role are as follows:

1. Registered users can view the detail information of ongoing competitions.
2. After viewing, the user can choose a specific competition to contribute, including upload the photos and give detail description.

2.3.8 Use case analysis of Review module

With the *Review module*, the reviewers can review photos and set awards, change awards and lock competition results. The use case diagram for the *Review module* is shown in the lower part of Figure 2.7, in which functions of each role are as follows:

1. The reviewer can review the photos, assess the level and set awards.
2. The reviewer can change the results of the assessment.
3. The reviewer can also lock the results of assessment. After locked, the result cannot be changed again.

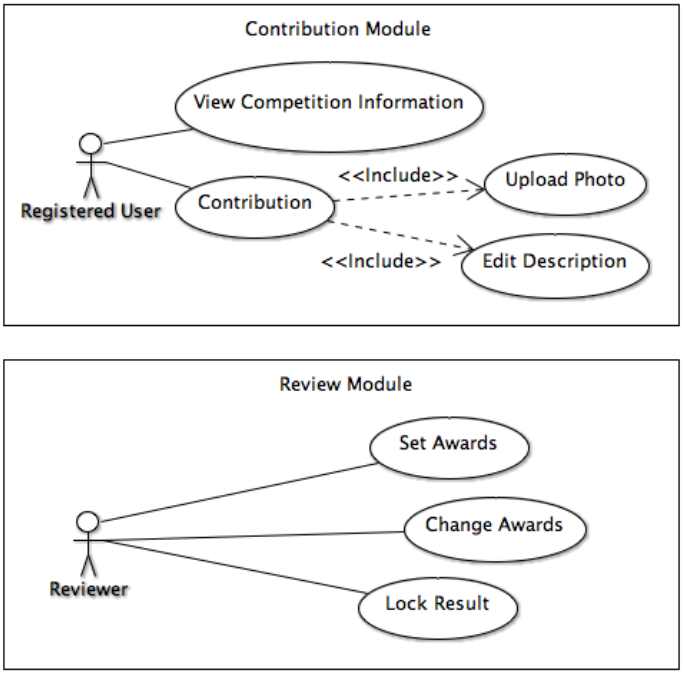


Figure 2.7 Use case diagrams of Contribution module and Review module

2.3.9 Use case analysis of Display of Competition module

The *Display of Competition module* is used to show all photos that have participated in each competition. All the registered users and visitors can browse the photos. The use case diagram of the module is shown in Figure 2.8, in which functions of each role are as follows:

1. Registered users and visitors can view all competition lists and select one competition.
2. Registered users and visitors can browse all the photos in the competition.
3. Registered users and visitors can give their favorite photos like's.
4. Registered users and visitors can set different sorting orders to display all photos, such as sorting by uploaded time, sorting by number of likes, and sorting by number of views.

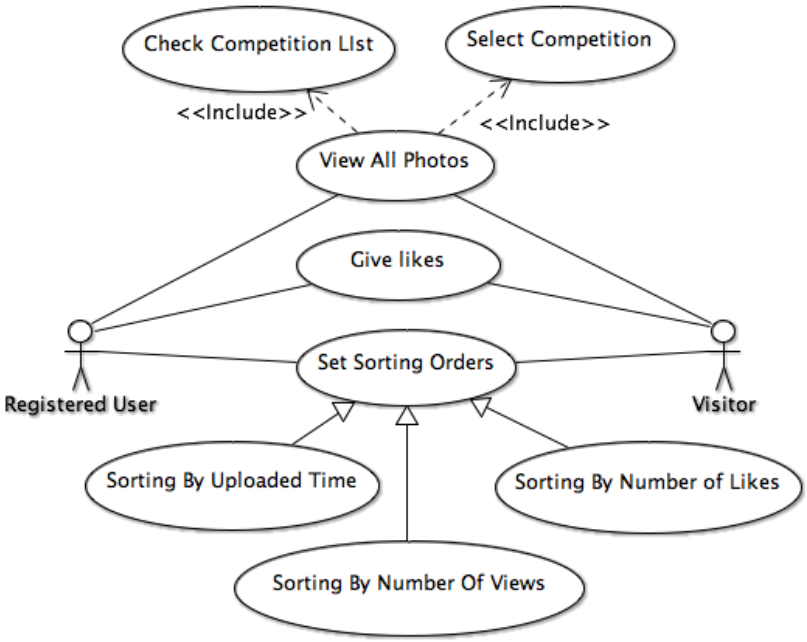


Figure 2.8 Use case diagram of Display of Competition module

2.3.10 Use case analysis of the Display of Competition Result module

As one of main function modules of the photography competition management subsystem, this module aims to display results and offer related operations to users. The use case diagram of the *Display of Competition module* is shown in Figure 2.9, in

which functions of each role are as follows:

1. Registered users and visitors can view the winning photos, including checking the list of finished competitions and selecting one competition to view.
2. Visitors and registered users can browse their favorite photos and give them like's.

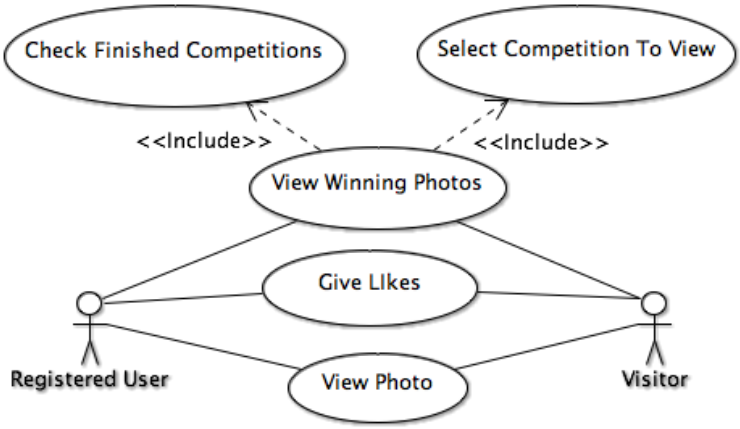


Figure 2.9 Use case diagram of Display of Competition Result module

2.4 Summary of this chapter

In this chapter, detailed functional and non-functional requirements of the system are analyzed. In the functional requirement analysis, the system is divided into two subsystems and ten major functional modules. Then a list for main objectives and implementing functions of each module is given. In the non-functional requirement analysis, the feasibility, scalability, maintainability, security and system performance are analyzed. In the last section of this chapter, the use case diagram of each module in the system is given，which is drawn by using UML (Unified Modeling Language). The UML not only visualize system requirements, but also illustrate each role and their operations in use case analysis. It allows users and developers to have a more intuitive and comprehensive understanding of the system requirements analysis and functions [8].

Chapter 3 System Design

3.1 The overall structure of the system

We use MVC framework as the overall structure in the design of this system. And the system is divided into 3 layers: view, control and model layer, which not only makes functions of different levels to be independent of each other, but also achieve the loose coupling of the system [9]. This structure greatly facilitates the system development and testing, and lays a solid foundation for the future improvement and expansion of the system [10]. The overall structure of the system is as shown in Figure 3.1

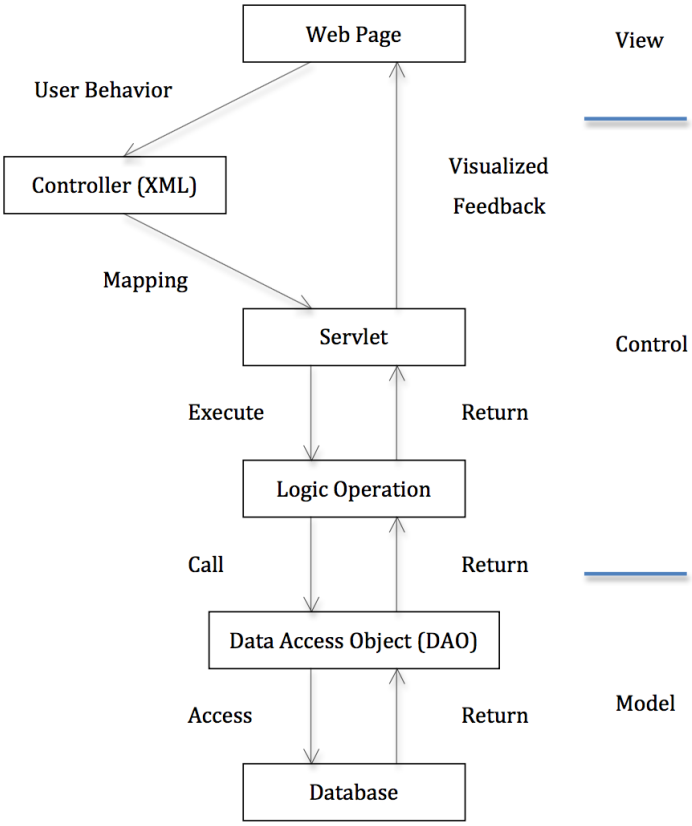


Figure 3.1 Overall MVC structure of the system

View layer: view layer is mainly used to provide users with visualized operations, to facilitate interactions between users and the system. A good view layer should contain

only the user visualization interface and the data of user operation, it should avoid containing business logic operation of the system and operation on the database [11].

Control layer: as the core of the MVC structure, control layer is mainly used to receive data from the view layer, to perform business logic operation and to call the functions of model layer. When the view layer sends a request to the control layer, the web application service normally filters and match the requested URL with the controller (xml configuration file), then sends it to the specified Servlet for relevant logic processing, and finally the control layer and the model layer interact with each other [12].

Model layer: model layer is mainly used for the data operation, such as CRUD (create, retrieve, update and delete) operation. In this system, we also import DAO (data access object) Entity, encapsulation of the database, and data objects oriented operation, to make the model layer to be loose coupling and manageable [13].

3.2 Module division and function design

According to the results of user survey and system requirement analysis, this J2EE-based photo sharing and photography competition management system is divided into two subsystems, the photo-sharing subsystem and the photography competition management subsystem, as shown in Figure 3.2. The photo sharing subsystem is divided into 5 modules, which are the *Personal Homepage*, *Photo Management module*, *Album Management module*, *Collection Management module*, and *User and Authorization Management module*. The photography competition management subsystem is divided into 6 modules, which are *Competition Management module*, *Contribution module*, *Review module*, *Display of Competition module*, *Display of Competition Result module* and *User and Authorization management module*. Among the modules, there is only one *User and Authorization Management module*, which is shared by the two subsystems.

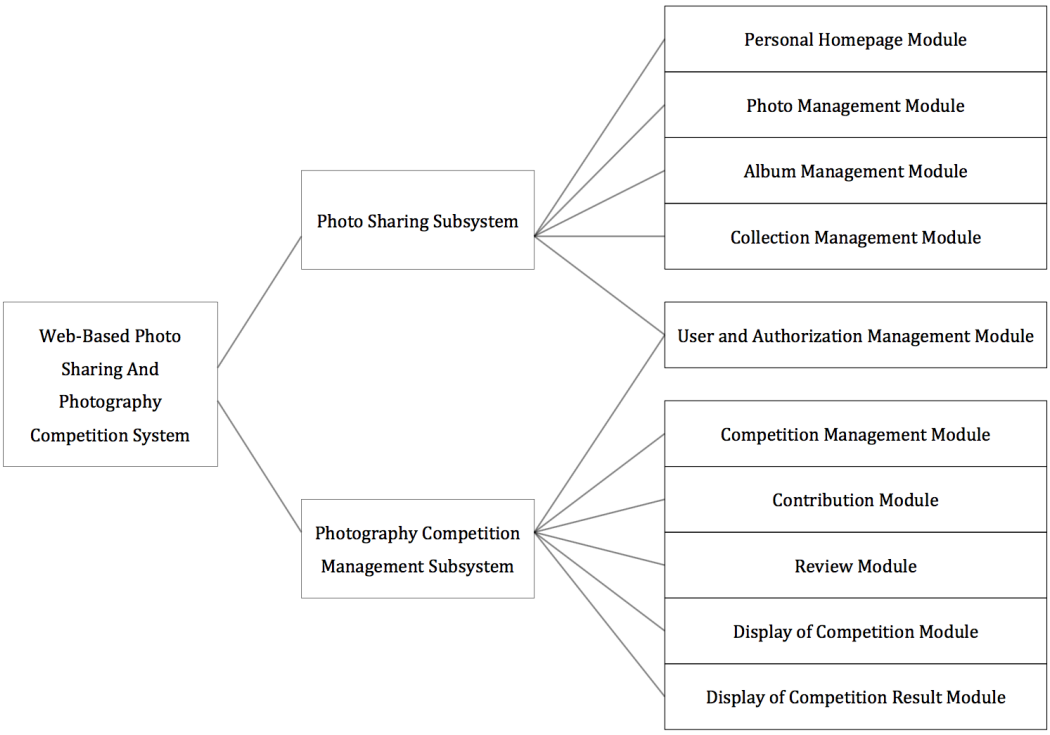


Figure 3.2 Division of system modules

The functions of each module are designed, which are described below.

1. User and Authorization Management module

This module aims to manage users and authorizations, whose functions include: registering account, logging into the system, logging out of the system, modifying user information, changing password, retrieving the password and authorization verifying.

(1) Registering account

Visitors can input relevant information in the registration page, and become a user after submit and verify. After successful registration, it will automatically login and jump to the user's personal homepage.

(2) Logging into the system

Registered users can log into the system by providing the user name and password. After the system verify that both of them are correct, it will automatically jump to the user's homepage.

(3) Logging out of the system

Logged-in users can log out at any time, and return to the login page to protect the user account security.

(4) Modifying user information

Registered users can modify and update their personal information at any time, including name, gender, birthday, avatar, self-introduction, contact information and so on.

(5) Changing password

Registered users can change their password through this module. They will be asked to enter the current password and the new password to set. If the current password is verified to be correct, the new password will be set into the system, replacing the old one.

(6) Retrieving the password

In case a registered user lost his password, he can retrieve his password through Email. What he needs to do is to input his username on the login page, and click on the "Forgot Password" link, after that the system will automatically send an email to the user and help him reset his password.

(7) Authorization verifying

When a user (including administrator) accesses some common modules or attempts to perform a high-privilege operation, the system will automatically verify current user's authority to determine whether the operation is allowed or restricted. If allowed, the system will jump to corresponding webpage, otherwise, warning information is displayed.

2. Personal Homepage

As a core module in the photo sharing subsystem, *Personal Homepage*'s main function is to show the user's own avatars, personal information, representative photos and selected albums. This homepage contains 5 parts: personal information display, personal representative photo display, personal selected album display, user customize and user operation.

(1) Personal information display

This function will automatically count the number of photos, the number of albums and the number of collections of the user, showing them in the personal homepage. It also displays the basic information of the user, including: name and labels.

(2) Personal representative photo display

This part displays the user's representative photos, which are the photos with the highest click rate and the photos selected by the user from all his photos. All users can

click on the photos for detailed viewing.

(3) Personal selected album display

This part displays the user's selected albums, which are the albums with the highest click rate and the albums selected by the user from all his albums. All users can click on any album for detailed viewing.

(4) User customize

This part displays the user customize, including: avatars and background picture.

(5) User operation

Users can comment, collect and like other users' photos and albums, and also can follow other users.

3. Photo Management module

This module is mainly used to manage the user's photos, with operations including: uploading photos, participating in competitions, moving photos, copying photos, deleting photos, modifying photo information and sort setting.

(1) Uploading photos

Users can select photos in their local storage and upload and save them onto the server side. After uploading, they can view and manipulate these photos on a web page with a visualized directory tree.

(2) Moving photos

Users can move photos from one subdirectory to another subdirectory. This operation is similar to Copying, except that it deletes photos copied.

(3) Copying photos

Users can copy photos from one subdirectory (called the source) to another subdirectory (called the target).

(4) Deleting photos

Users can delete the specified photo to repair the user's operation errors, such as: error uploading, redundant uploading and so on.

(5) Participating in competitions

Users can select photos uploaded to the system to participate in competitions. While doing so, he needs to choose specific competitions on the competition list.

(6) Modifying photo information

Users can modify relevant information of their own photos, including: name, category and so on.

(7) Sort setting

Sorting orders can be set by users, including sorting by upload time, sorting by number of likes, sorting by modifying time and sorting by number of views.

4. Album Management module

This module is mainly used to operate on albums and photos, including creating albums, moving albums, copying albums, deleting albums, modifying album information and sort settings in addition to the same functions as the *Photo Management module*.

(1) Creating albums

Users can create albums to store photos and albums, which has greatly enriched the photo and album folder structure, and made the photo and album resource management more reasonable. When creating an album, you need to input information about the album's name, category and so on.

(2) Moving albums

Users can move the album from one subdirectory to another subdirectory.

(3) Copying albums

Similarly, the specified album can be selected, and then the copy object of the album is created in the current subdirectory.

(4) Deleting albums

Users can delete the specified album or delete photos in a batch way.

(5) Modifying album information

Users can modify relevant information of their own albums, including: name, category and so on.

(6) Sort settings

Different sorting orders can be used for album display, including sorting by uploaded time, sorting by number of likes, sorting by modified time and sorting by number of views.

5. Collection Management module

The *Collection Management module* provides functions related to user collections, including: adding collection, viewing collection, and deleting collection.

(1) Adding collection

When users are viewing photos or albums of other users, they can use this function to add resources（photo or album） into his collection. The system will draw the associated information from the database and insert into the collection list.

(2) Viewing collection

After collection, users can directly click on the collection items, which will directly jump to the target photo or album display page.

(3) Deleting collection

When a user is no longer interested in some items of resource, he can simply remove them by deleting the items from the collection.

6. Competition Management module

The super administrators are responsible for managing competitions. Operations that can be performed include creating a new competition, modifying competition information, deleting the competition. As to normal users, they can only use this module to view the competitions they have participated in.

(1) Creating a new competition

The super administrators can create a new photography competition with this function, during which he must input the relevant information about the competition, such as the name, the dates of the competition, and assign administrators and reviewers for the competition.

(2) Modifying the competition information

The super administrators can modify the information of each competition, including the name, start date, end date, reviewers and administrators of the competition.

(3) Deleting competitions

The super administrators can also delete any competitions.

(4) Viewing the competitions participated in

When a user accesses this module, he is allowed to view his contributed photos and the associated competitions he has participated in.

(5) Deleting the photo

The super administrators and his designated competition administrator can delete inappropriate works (such as reactionary works, pornographic works, etc.) for a specific competition.

(6) Locking the user

The super administrators and the competition administrators can also lock the user who uploaded inappropriate works and prohibit him from continued uploading.

7. Contribution module

Users can contribute through this model. Operations include: viewing competitions and contributing.

(1) Viewing competitions

Users can view all the information and photos of all the ongoing competitions.

(2) Contribution

Users can choose competitions and upload their own photos to participate in.

8. Review module

Through this module, a reviewer can review all the works in the competition and give his evaluation results, including: setting levels of awards, changing awards level later on and locking review result when competition administrator believes no changes will be made in the future.

(1) Setting levels of awards

The reviewers can enter the review results into the system through this function.

(2) Changing awards level

The reviewers can change the levels of awards that they have already evaluated to be.

(3) Locking review result

When the review function of a competition is locked, the reviewer can no longer change review results and related information.

9. Display of Competition module

The module is mainly used for showing all photos in the competition. The operations all users can perform are: viewing photos, giving photos like's, browsing photos and setting sorting order.

(1) Viewing all photos

Both the visitors and the registered users can view all photos. They first select one competition from the competition lists, and enter into the webpage of the corresponding competition to view all the photos in the competition.

(2) Giving photos like's

Registered users and visitors can give their favorite photos like's.

(3) Browsing photos

All the photos are shown in thumbnails in the showing webpages. Users can browse the photos by clicking on the thumbnail.

(4) Setting sort order

This function is prepared for the uses to set the sorting order for displaying photos. The orders allowed include sorting by uploaded time, sorting by the number of likes and sorting by number of views.

10. Display of Competition result module

The function of this module is to implement the showing of competition results. Operations provided include: viewing winning photos, giving photos like's and browsing photos.

(1) Viewing winning photos

Users can view competition results by viewing the list of completed competitions and then selecting the competition they want to view.

(2) Giving photos like's

Registered user and visitors can click on the link below the winning photos to give their favorite photos like's.

(3) Browsing photos

Registered users and visitors can browse photos by clicking the target thumbnail photos.

3.3 The detailed design of main function modules

Some modules' functions of this system are similar, such as deleting function of *Photo Management module*, *Album Management module* and *Collection management module*, as well as moving, copying, modifying information and setting sort order functions. Although these functions call different classes of entity and different methods, but they are similar in sequence diagrams. To avoid repetition, sequence diagrams are only drawn for a part of functions with a detailed process analysis in the detailed design of each module in this section.

3.3.1 User and Authorization Management module

The *User and Authorization Management module* is a basic module of both the photo sharing and the photography competition management subsystem.

In this module, the registration function is used for visitors to apply for user accounts. After a visitor registers successfully or logs in with the correct user name and password, the system will change the user's status and the operation authority directly. At the same time, page's automatically redirect function can be achieved according to the current user's authorization. After logging in, the user can modify the personal information, change password, change customize settings and logged out. Therefore, the login function is the core function of this module. The sequence diagram when a user logs in is shown in Figure 3.3.

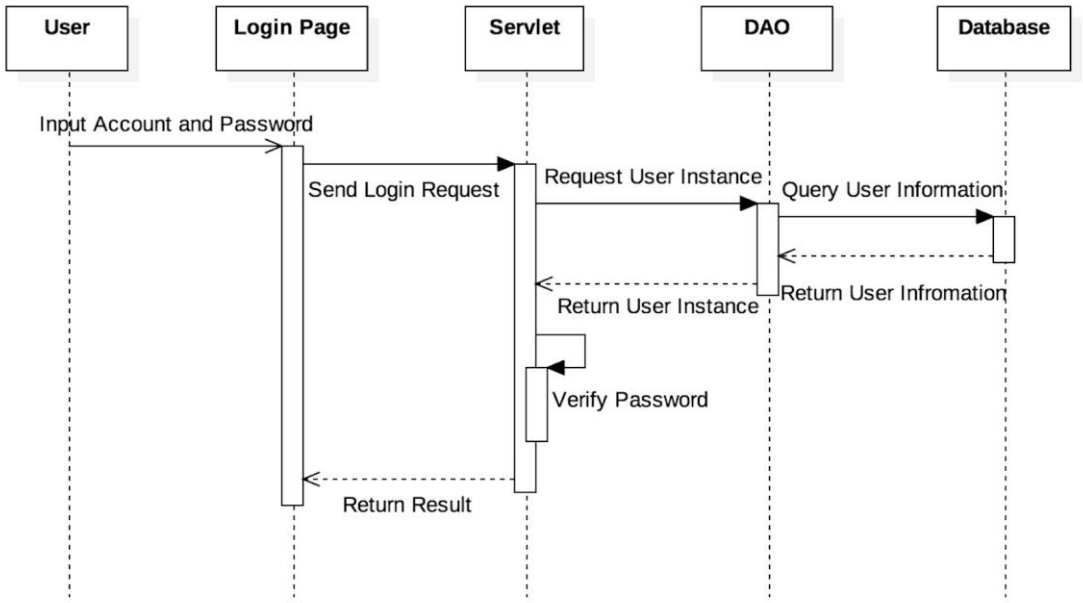


Figure 3.3 The sequence diagram when a user logging in

The basic flow of user login is as follows:

1. The user accesses the login page and input his account and password.
2. The login page sends the login request together with the user's input information to the LoginServlet
3. LoginServlet calls the user DAO (data access object) to perform query operation on the database, and returns the instance object of user.
4. loginByUserEmail method Compare and verify the input password with the correct password (saved in the instance object).
5. If the password is wrong, login fails and then returns to the login page， reporting an error. Otherwise, login succeeds and then jumps to the user's personal homepage.

The password saved in the database and user type instance objects have already been encrypted. Therefore, the loginByUserEmail method also needs to encrypt the input password by using MD5 before verifying them. This mechanism can avoid the stealing of password while the database or server is being attacked.

3.3.2 Personal Homepage

The *Personal Homepage* mainly aims to show the user's personal information and their resources. Figure 3.4 shows the sequence diagram of displaying a user's representative photos

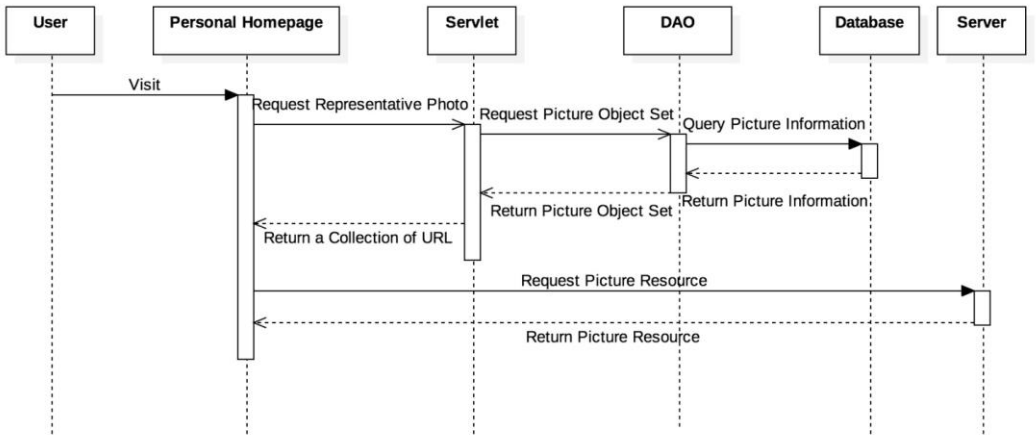


Figure 3.4 Sequence diagram of displaying a user's representative photos

The basic flow of displaying a user's representative photos is as follows:

1. The user visits the personal homepage for browsing.

2.Personal homepage requests the HomepageServlet for relevant information about the current user's representative photos.

3. HomepageServlet calls the user data access object for query operation, and returns a collection of picture objects containing information of the user's representative photos.

4. All photos' URLs are obtained from the collection of representative photo objects, and are returned to the personal homepage.

5. The Personal homepage requests to the server, accesses picture resources and displays representative photos using the URLs.

The personal homepage will display the representative photos by using the URLs, which are returned by the HomepageServlet. But if there is nothing returned, the personal homepage will be empty. So I have adopted a mechanism for automatically generating representative photos in case the user have not set his representative photos yet, which will be discussed in section 4.2.2.

3.3.3 Photo management module

Users can enter the photo management page and upload their own photos for sharing. After successfully uploading, Users can perform a variety of operations on their photos. The sequence diagram of uploading photos is shown in Figure 3.5.

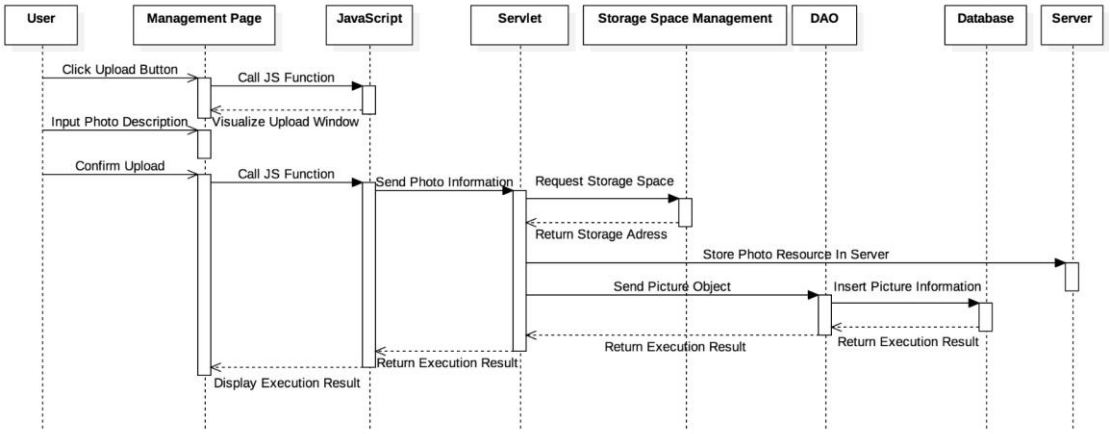


Figure 3.5 The sequence diagram of uploading photos

The basic flow of photo uploading is as follows:

1. The user triggers photo upload action in the photo management page.
2. The photo management page calls the relevant JavaScript function to visualize the uploading window.
3. The user selects local photos and gives the related information in the uploading window.
4. When the user chooses to upload photos and confirms, the relevant data and request will be sent to UploadPhotoServletfor processing.
5. The UploadPhotoServletfor requests the storage space management subsystem to allocate the corresponding storage address, and to store the photo resources in the server.
6. By using the photo data access object method, the value of the photo object (relevant information of input photos) is saved in the database.
7. Finally, results are returned and displayed.

The photo's information and the photo's store address will be saved to the corresponding database tables after the photo has been uploaded successfully (saved in the sever side).

3.3.4 Album management module

Similar to the *Photo Management module*, the *Album Management module* is mainly used for the management of album resources. Through the functions of *Album Management module*, users can create new albums, move albums, delete albums, modify album information, set sorting order and so on, which makes the user's storage

and folder structure more efficient and intuitive. The sequence diagram of moving album is shown in Figure 3.6.

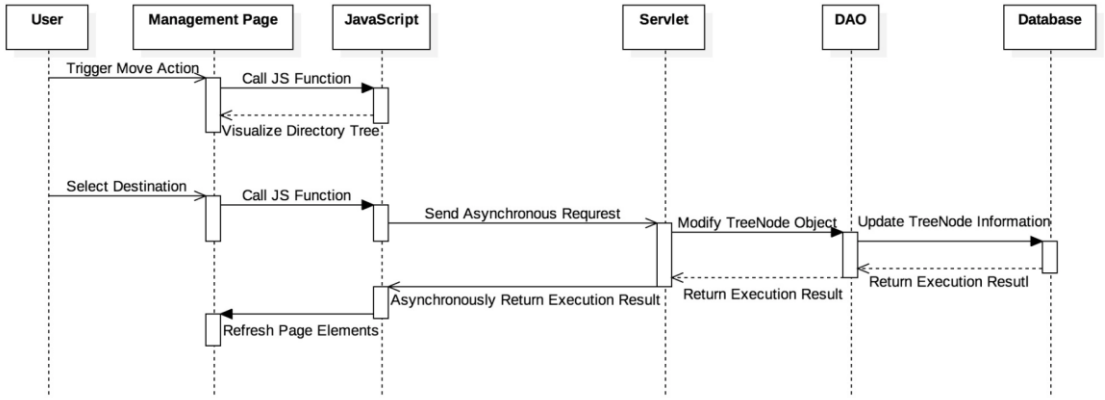


Figure 3.6 The sequence diagram of moving albums

The basic flow of moving albums is as follows:

1. The user triggers moving album action in the album management page.
2. By calling the corresponding JavaScript function, the album management page visualizes the directory tree structure and display it in the pop-up window of user operation.
3. After the user selects the moving destination and confirm it, the album management page will asynchronously send moving requests and related data to the MoveItemServlet for processing.
4. The MoveItemServlet calls the tree node data access object to update the database, and then returns the operation results.
5. The JavaScript modifies page elements of the new album management page by the asynchronous return results, and no longer shows album files moved to other directory.

There is a significant difference between album management and photo management. When a user delete, move or copy an album, the JavaScript needs to modify the directory tree's HTML codes based on the result returned from AJAX. But this won't happen when a user operates on photos. This mechanism can change the directory tree's structure without refreshing the page, while still keeping the directory tree's structure synchronization between the client page and the server side.

3.3.5 Collection management module

The *Collection Management module*, as an auxiliary function module, provides collection operations to users, which greatly improves the access efficiency to resources of other users and enhances interaction among users. Users can create mirror items mapping to other users' photo or album files. By clicking on the items, users can directly jump to the relevant resource page. Besides, users can delete the collection items by using this module's functions. The sequence diagram for deleting items is shown in Figure 3.7.

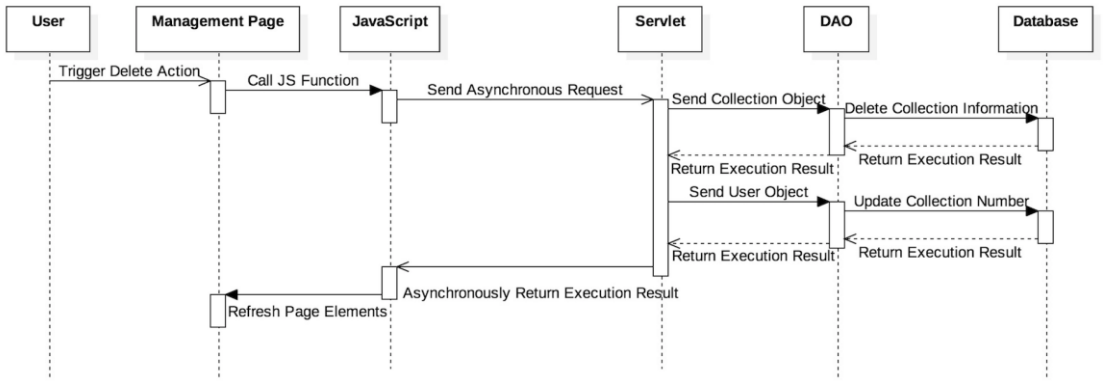


Figure 3.7 The sequence diagram of deleting items

The basic flow for deleting items in a collection is as follows:

1. The user triggers the delete collection action in the collection management page.
2. The collection management page calls the corresponding JavaScript function, and uses AJAX asynchronous transfer technology to send deleting requests and related data to the DeleteItemServlet.
3. The DeleteItemServlet calls collection data access object and user data access object to delete information related to the item, to update database and to modify the total number of collections.
4. Results are returned to the Servlet and JavaScript.
5. Finally, with JavaScript and AJAX asynchronous transfer technology, the current collection page elements are refreshed, and the items that have been deleted is no longer showed.

When the user tries to delete his collection items, the system will delete only the collection item's corresponding records from Collection table. The original photo resource and other relevant information (if any) won't be changed. This can avoid the

unauthorized operations on resource by other users and the loss of original resource.

3.3.6 Competition management module

This module provides the main functions related to competition management. Through these functions, users and administrators can perform different operations relate to competitions. The sequence diagram of creating competition is shown in Figure 3.8.

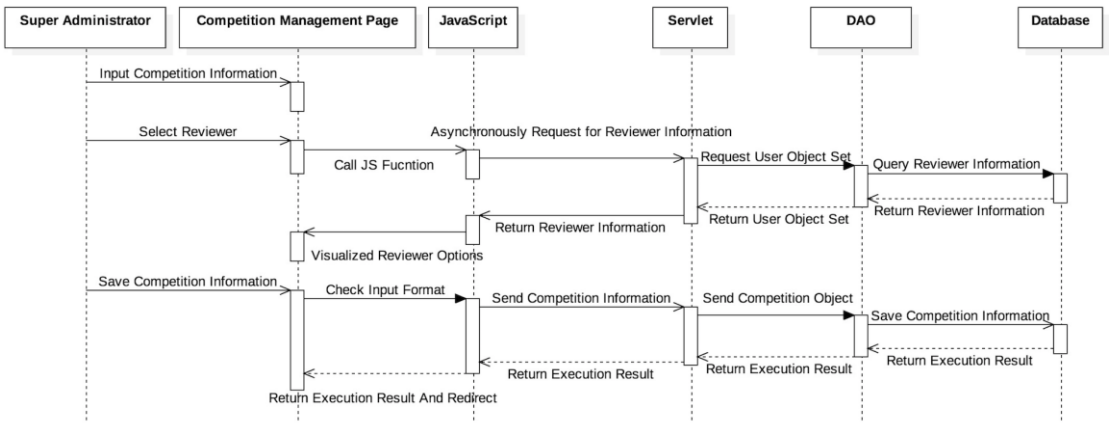


Figure 3.8 The sequence diagram of creating competitions

The basic flow of creating competitions is as follows:

1. The super administrator triggers creating competitions in the competition management page, and enters the related competition information.
2. When the administrator selects the reviewer, he will trigger JavaScript function and use AJAX asynchronous transfer technology to send request to the QueryServlet for obtaining reviewers information.
3. The QueryServlet calls the user data access object for the database query, returns a collection of all reviewers' user object in the database, and extracts the corresponding reviewer information from it.
4. The relevant reviewer options are visualized with asynchronous method and JavaScript functions.
5. The administrator enters all the competition information for saving. The competition management page then calls the JavaScript function to check whether the input information format is regular. If yes, it will send request and competition data to the CreateCompetitionServlet.
6. The CreateCompetitionServlet calls the competition data access object to store

competition information in the database, and returns saving results.

7. Finally, the page will jump to corresponding pages according to the returned results.

This flow use AJAX to transmit data asynchronously and use JavaScript to visualize the options of the reviewers. It is convenient for the administrator to select reviewers since there are only a fewer reviewers in general. And this also improves the system's security (only the reviewers existent in the system can be selected) and prevents the system from Hotlink attack or potential errors. But this may become inconvenience when the reviewers need to be increased. Hence we need to add a retrieving mechanism for quick finding of reviews in the future.

3.3.7 Contribution module

The *Contribution module* provides the relevant functions for participate in competitions and contributions. The sequence diagram of competition contribution is shown in Figure 3.9.

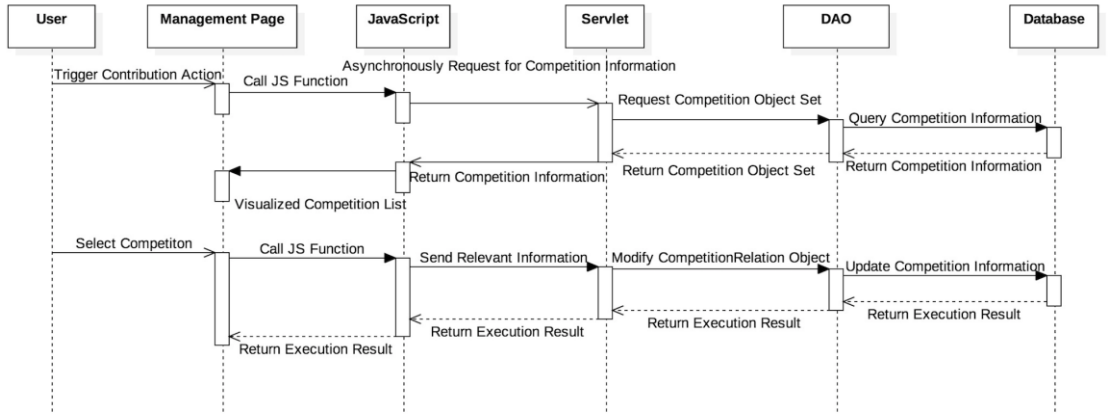


Figure 3.9 Sequence diagram of the competition contribution

The basic flow of the competition contribution is as follows:

1. The user selects photos that have already been uploaded to the system in the management page and click the "Contribute" button to trigger the contribution operation.

2. The management page calls the JavaScript function and use AJAX asynchronous transfer technology to request competition information from a QueryServlet.

3. The QueryServlet calls competition data access object for database query operation, returns the collection of competition instance objects, extracts the

corresponding competition information, and then asynchronously returns to the JavaScript.

4. JavaScript uses the relevant information to show the competition list.

5. The user chooses the competition to participate in.

6. The management page sends the relevant request information to the ContributeServlet for processing.

7. The ContributeServlet calls the competition relation data access object for the insert database operation, and return the results of the operation.

There are two ways for users to contribute. This flow describes the way to contribute by through photo or album management page. Section 4.2.6 describes the way to contribute by through competition page.

3.3.8 Review module

The *Review module* is used to review all users' photos, by which the reviewer can review the winning works in the competition, and modify review results. However, after reviewer lock the results, the review results can no longer be changed. The sequence diagram of review is shown in Figure 3.10.

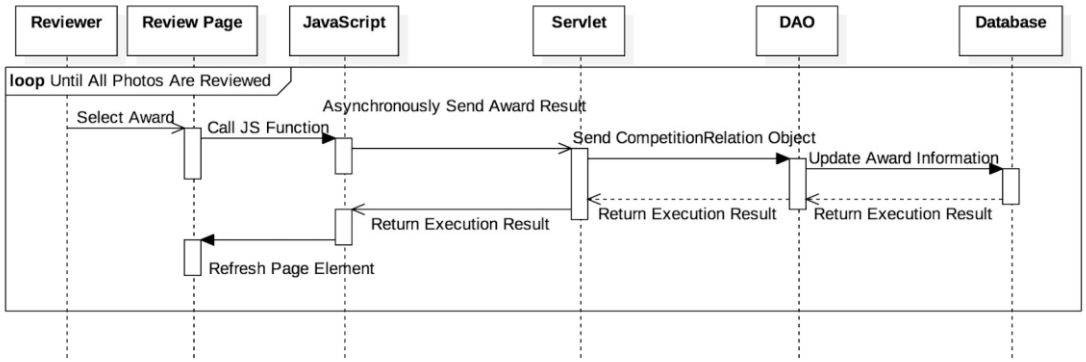


Figure 3.10 The sequence diagram of review

The basic flow of the competition review is as follows:

1. The reviewer visits the review page to review photos.

2. The reviewer selects an award and triggers review operation, calling JavaScript function and use AJAX asynchronous transfer technology to send review request and related data to ReviewServlet.

3.The ReviewServlet calls the competition relation data access object to update data, records the photo award level, and then returns the result of the operation.

4. Repeat 2-3 until all photos are reviewed.

In this flow, step 2 and step 3 can be constantly repeated until all this competition's photos have been reviewed. And the reviewing can be interrupted or continued at any time (unless the completion's review function has been locked by administrator). But currently there is no visible marker or sign showing the specific photos already reviewed which may cause some confusion to the reviewer. Therefore, I will solve this problem in my future work.

3.3.9 Display of competition module

Display of competition module allows the user to view all photos and related information for all photography competitions, and sort or give likes to these photos. Since the display function of this module is similar to the individual representative shown in 3.3.2, the function of sorting according to given conditions is described here. The detailed sequence diagram is shown in Figure 3.11.

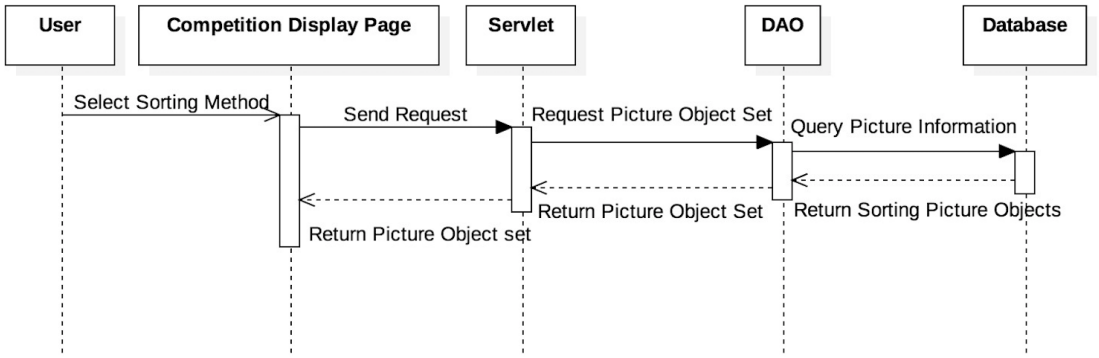


Figure 3.11 The sequence diagram of sort according to given conditions

The basic flow of sorting according to given conditions is as follows:

1. The users can sort according to specific conditions in the competition display page, such as: uploaded time, modified time and so on.
2. The photo display page sends request and related data to the GetPicturesServlet.
3. The Servlet calls competition relation data access object for query the database under specific conditions (for example, flashback according to uploaded time). And then returns a collection of picture instance objects, which meet relevant conditions and sorts.
4. Finally jump to the new page, the relevant photo information is also returned to the new page.

When the sever intend to return the sorted page to the client, the specific JSP will extract URLs from the obtained collection of picture instance objects, and use their URLs to build the HTML code.

3.3.10 Display of competition result module

The *Display of Competition Result module* is designed to allow the user to view the winning works of the photo competition and to give their favorite photos like's. The sequence diagram of display of competition result is shown in Figure 3.12.

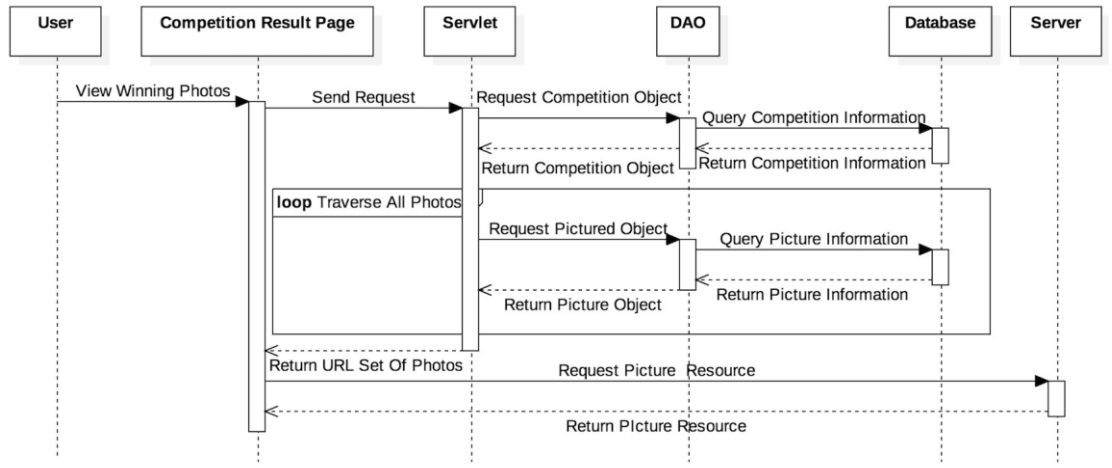


Figure 3.12 The sequence diagram of display of competition result

The basic flow of this module is as follows:

1. The user visits the display page of competition result.
2. The competition result page sends a request to the GetPicturesServlet.
3. The GetPicturesServlet calls the competition data access object for the query operation on the database, and returns competition information.
4. The GetPicturesServlet calls the competition relation DAO and the picture DAO for the database query operation, and extracts the picture instance object from awarded photos of this competition.
5. Repeat 4 until all picture instance objects from awarded photos of this competition are gathered.
6. Return URL and relevant information of all photos to the competition result page.
7. The competition result page requests picture resources from server and displays them.

During repeating step 4, the Servlet will user different parameters (for different awards) to get all wining photo's instances, and then the specific JSP will use these instance's URLs to build the HTML code for the client page.

3.4 The design of Background system

3.4.1 The design of system role and authorization mechanism

There are 4 roles in the system, as shown in Table 3.1

Table 3.1 System role and privilege

Number	Role	Description
01	Ordinary users	Can only manage their own resources or competitions they participated in
02	Review users	Can review the prizes for specified photography competitions
03	Competition administrator	Cannot create a competition, but can delete photos of the specified competition and lock the user
04	Super administrator	Can create the competition, modify the competition and set competition results, and can delete photos of the specified competition and lock the user

The access control mechanism for this system adopts the resource privilege mode based on system roles and system functions in table 3.1. Each page is regarded as a separate resource. When different roles access certain resources, the system needs to check weather the users have the privileges to access relevant pages and use relevant functions. The setting of resource privilege prevents non-administrator users from violating the privilege, and avoids the behavior that may damage the system [14].

The objectives of this access control mechanism:

1. After the user logs in, the system queries the user's role information, and saves related information in the session for later usage. When the user attempts to execute a high-privilege operation, the system will check the user's role information saved in the session, avoiding frequent access to database.
2. When the user accesses specific pages and perform operations on the

resources, the system will query the user's role information to determine whether the resources belong to this user, and whether the user has the relevant privilege to operate. If he does not meet the conditions, he will be returned to the previous page with a warning given. This processing can ensure the safety of system operations.

3. Some pages are shared by different roles, and will display different modules and content based on the current user's role and privilege. The competition management module is an example. This scheme will achieve highly role-oriented and flexible page functions.

3.4.2 Database tree structure

In this system， many operations involve management of photos and album folders. If the original data rows are directly stored in the tables of the database, the correlation among them will be poor, which is not conducive to achieve efficient operation and management of data, and users cannot perform intuitive and convenient operation on their own resources, which also easily leads to waste of system resources and inefficient use of the database.

In order to improve the efficiency of the system and realize the directory tree resource management, we introduce tree structure in the database. Every object (including photos, albums, collections) is regarded as a tree node. At the same time, the ID information of the parent node is added to this tree node's information. Then the tree structure among objects is established through corresponding node ID information, and achieved the directory tree resource management. It highly improves the efficiency of searching data, and the flexibility of data storage [15]. After that, we can use tree node information in record to find their father node and child node conveniently.

The tree node records need to include the following metadata:

1. Tree node ID: used to uniquely identify a tree node records
2. Association ID: the unique ID of the object associated with the tree node.
3. Association type: the type of object associated with the tree node (photos, albums, collections, etc.).
4. Father node ID: the unique ID of this tree node's father node.
5. User ID: the unique ID of the user who own this node.
6. Layers: the number of layers in which the tree node is located.

Through the user ID, the tree node ID and the father node ID, the system can retrieve the tree structure of the specified user from the database. And the combination of association ID and association type can point to a specific object corresponding to the node. The use of the number of layers can greatly improve the efficiency of query to avoid unnecessary operations.

For example, to find the father object of the current album folder, only 3 steps are needed:

1. To query the database and take out tree node records binding to the album folder through the current album folder ID and object type (album).
2. To get the corresponding father node record through the father node ID saved in the tree node record.
3. To extract the father object of the current album file through association ID and association type in the father node record.

3.4.3 Storage space management subsystem

We have developed and incorporated an independent storage space management subsystem into this system, so that photos and other resources can be stored in different devices and different physical address (but the same logical storage address). The subsystem not only provides possibility for storage space expansion and distributed storage, but also realizes flexible management of the storage space on the server. At the same time, we also reference rsync algorithm to design a unique file synchronization function, it can avoid repetitive transmission of the same data and make the server synchronization and backup more efficient. The storage space management subsystem includes two modules: file synchronization module and resource management module.

1.File synchronization module

The file synchronization module mainly serves for data synchronization and backup among different databases. This module uses a unique algorithm to improve the efficiency of synchronization, reduce unnecessary data transmission, and avoid network resources and system resources waste. By using this algorithm, the sender will not sync all the files to the other side, but identify a different part from the receiver by its included algorithm, and then transfer it to the other side.

In the synchronization algorithm design of this module, we reference the rsync

algorithm in the Unix-like system, which is usually used for data backup, file synchronization between remote hosts, and network disk data update process. Rsync algorithm can distinguish different parts of two files by operation, and then only transmit different parts, which is the most important feature of the algorithm [16].

The main flow of the algorithm is as follows:

(1) The receiver calculates the checksum.

The receiver first divides its file into equal size blocks, and then calculates the rolling checksum and MD5 checksum for each block, then submit the results to the sender.

The rolling checksum used in rsync is based on the Adler-32 checksum invented by Mark Adler, which is a 32-bit weak checksum [17]. While the MD5 checksum is a widely used 128-bit strong checksum [18].

The reason for using the two checksums in this algorithm is that although MD5 is used to check whether the two file blocks are the same with the low probability of collision, but its operation speed is lower because of the limitations of computer hardware and processing speed. Therefore, it is necessary to have a faster rolling checksum with a slightly lower accuracy rate, but sufficient to determine whether the two blocks are different. Beyond that, the MD5 checksum used between the same rolling checksum blocks to ensure that the two blocks are exactly the same. After checking and comparing two checksums, it is found that the difference between the two blocks is almost negligible, with the probability of 1/2 ^ (128 + 32) [19].

(2) The sender calculates and compares the checksum

The sender also divides the files into different blocks with the same size as the receiver, and then quickly computes the rolling checksum of each block. If the checksum of the block is not the same as that of any block of the receiver, it means that the block has different content between two sides. If the rolling checksum is the same, the sender will calculate the MD5 checksum and compare it with the receiver's MD5 checksum, determine whether they are identical [20].

It is important to note that the file block movement on the sender side is scrolling rather than jumping at a fixed value. If the content of the current block differs from the content of any block on the receiver, the block will move back by 1 byte instead of the size of the entire block before calculating the checksum. Only when the block is exactly the same as a block on the receiver will it move back by the size of the entire block. Specific rolling process can refer to Figure 3.13.

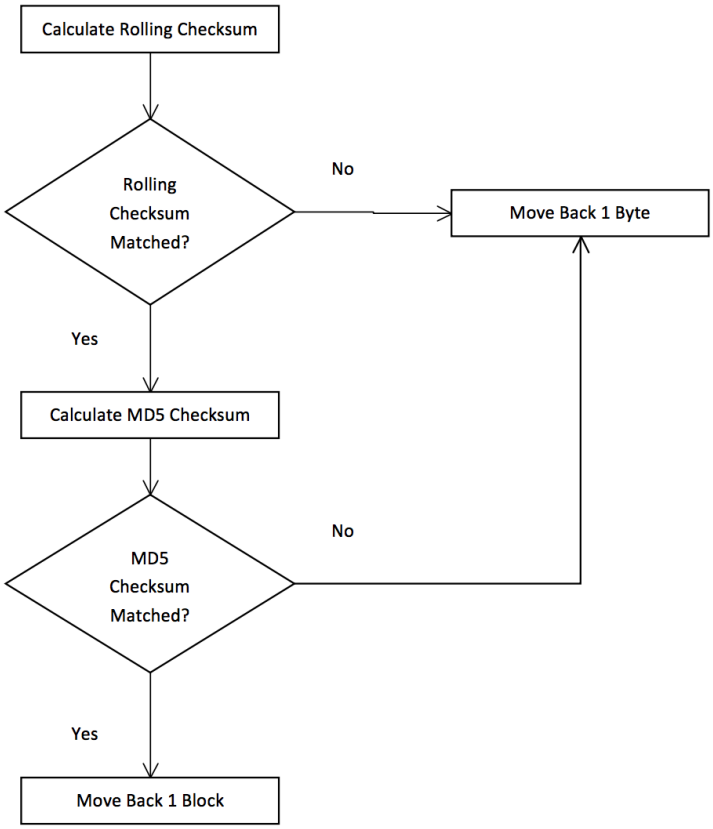


Figure 3.13 Scrolling diagram of the rsync algorithm

(3) The sender sends data

The sender sends contents of the blocks of checksum mismatch and their location information to the receiving side. The receiving end updates file contents according to the obtained data to realize the synchronization of the file.

2.Resource management module

The resource management module is applicable for a variety of resources, including photos, text, etc. The resource management module, as the core module of the storage space management subsystem, mainly provides functions for the CRUD operation of the server resource, which realizes a unique and reasonable storage mechanism to control system resources (such as disk size, block size, file size, etc.).

Resource management module's storage structure has five layers, including （from outside to inside）the server layer, virtual disk layer, resource type layer, block layer and resource files. Locating and access each resource files need to know the

corresponding information of each layer. For example, the full path of a resource is [Service]/[VirtualDisk]/[ResourceType]/[Block]/[FileName]. The detailed storage structure is shown in Figure 3.14.

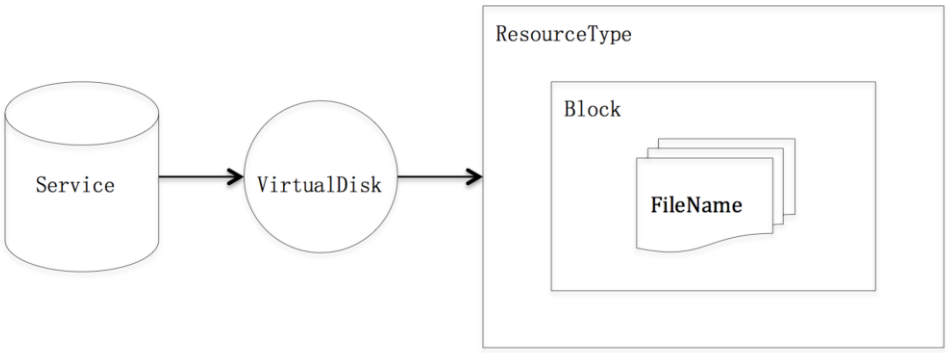


Figure 3.14 Storage structure

(1) Service layer: service layer refers to server or storage device, and it is usually hardware with operating systems. The service layer can make the system resources stored in different servers or storage devices, to achieve resource scalability and distributed storage [21].

(2) Virtual disk layer: virtual disk usually is the root directory file or disk partition of the system. And a single service layer object can have multiple virtual disk layer objects at the same time.

(3) Resource type layer: the resource type layer is mainly used to divide the resource type, in order to facilitate the administrator's browsing and management.

(4) Block layer: the block layer is established based on the resource type layer. Each resource type has multiple block objects corresponding to its own type. At the same time, each block limited the maximum storage resources.

(5) Resource files: stored resource files.

3.5 Database Design

The database, as the foundation of the whole system, is the key in the process of system design and development. Efficient system development processing is often based on the sound and detailed database structure [22]. The data in the database not only describe the relevant information of each element in the system, but also quantize the user operations.

After analyzing the logical relationship between the system demands and the system, we draw the demands on data, and set up data tables to meet the demand. We have designed the E-R relationship among data tables, which are shown in Figure 3.15.

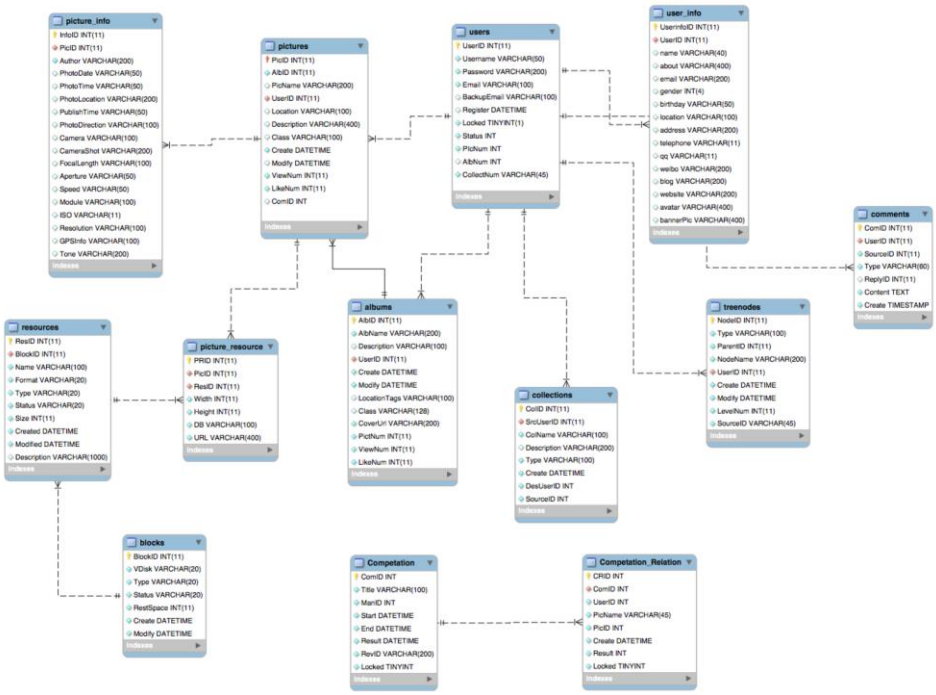


Figure 3.15 ER diagram of main database tables of the system

1. User table

User table mainly stores the user's basic information, including: user name, password, mailbox, user privilege level, user lock status and other fields, as shown in Table A.1 in Appendix A. It is the major table of the database and the whole system.

2.Picture table

Picture table is used to store relevant picture information and the relationship between albums and users, containing 13 fields: photo ID, album ID, user ID, competition ID, photo name, upload time and total number of browse, and so on, which are shown in Table A.2 in Appendix A. Each picture has a corresponding record in this table.

3.Picture resource table

The picture resource table mainly records information of the picture resource, which contains the following fields: picture resource ID, associated picture ID, associated space resource ID and URL, as show in Table A.3 in Appendix A. This table records the relationship between pictures and resources. It also allows different

picture table records link to a same space resource record, which can avoid large amount of repetition storage and save the storage space of the system. But it also makes the resource deleting more complicated.

4.Album table

The album table is used to record basic information about the album and the relation among users, containing 12 fields, such as album ID, album name, user ID, album class, as shown in Table A.4 in Appendix A.

5.Collection table

Similar to the picture table and the album table, the collection table records the user's collection information and related logical relationship, including: collection ID, custom collection name, collected user ID, collecting user ID, and so on, which are shown in Table A.5 in Appendix A. Combine SourceID field and Type field can define a specific resource (e.g. a picture or an album)

6. Comment table

The table mainly stores the user's comment and content, containing following fields: the comment ID, user ID, resource type of the commented object, and ID of the resource corresponding to the commented object. Regardless of whether the object being commented is a photo or an album, you can find the relevant data by looking up the photo table or the album table, as long as you know the relevant SourceID and the type. The specific structure of the table is shown in Table A.6 in Appendix A.

7. Competition table

Information related to the photography competition is stored in this table, containing competition ID, competition name, sponsor ID, start date, end date, published result date, reviewer ID, and review lock status, where reviewer ID refers to ID information of multiple reviewers with the Varchar data type, as shown in Table A.7 in Appendix A.

8. Competition relationship table

This table records the logical relationship among the user, the user's photo, and the photography competition, and the winning results of the corresponding photo, containing competition relationship ID, competition ID, user ID, picture ID, and result, as shown in Table A.8 in Appendix A. The reason for not setting a foreign key in this table is to prevent from disturbing when the administrator deletes the competition or when the user deletes the photo, which can avoid the photos on the other module from being displayed normally.

9.Tree node table

This table is mainly used to store information about the directory tree structure, which is one of the most important tables in the database, containing: tree node ID, node resource type, father node ID, and user ID. The detailed structure of the table is shown in Table A.9 in Appendix A.

10.Space resource table

This table is mainly used for storing data for the space management subsystem. Using data in the space resource table and the space block table, you can determine the unique resource and information about its storage path. The space resource table contains: resource ID, block ID, original name of resource, resource size, and created time, as shown in Table A.10 in Appendix A for details.

11.Block table

This table records partial information about the location where the resource is stored, including the block ID, VDisk, storage status, and remaining storage space. The detailed structure is shown in Table A.11 in Appendix A.

3.6 Summary of this chapter

This chapter introduces the design of the photo sharing and photography competition management system, including system structure, module function, background system and database design. First of all, a detailed analysis of the process of some functions is conducted through the sequence diagram, and then a detailed description is performed for the background system, privilege setting and storage space management module. Finally, the relationship among database tables and detailed table fields are shown. All of the design processes are explained concisely and clearly.

Chapter 4 Implementation of the System

Based on demand analysis and system design of the system in the first two chapters, we fulfilled the implementation of this photo sharing and photography competition management system, which is described in this chapter. The implementation of each module is described by giving corresponding user interface screenshots, procedures and part of code line (See Appendix B: Part of System Codes).

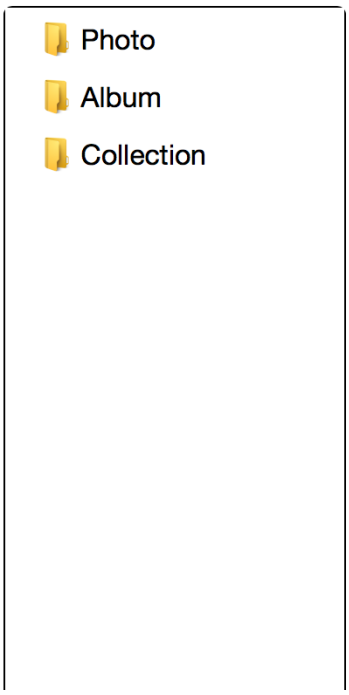
4.1 Common module implementation

The common part of the system is extracted and encapsulated into the common module to avoid redundant development. This can also improve the structure of the system, and reduce the coupling between different modules or functions to achieve a more object-oriented programming purpose [23]. This system contains two common modules, namely the directory tree resource management interface and the storage space management subsystem.

4.1.1 Directory Tree Resource Management Interface

The directory tree resource interface enables users to perform simple and intuitive operations on managed resources of the system [24], which is used in the *Photo Management module*, the *Album Management module*, the *Collection Management module* and Contribution functions.

When the user does nothing, the directory tree interface displays only three defaulted system folders: the photo folder, the album and the collection folder, as shown in Figure 4.1 (a). When the user clicks on any one of the folders, the directory tree interface will be expanded to show its subfolders. If the clicked folder does not contain subfolders, the folder will be highlighted, as shown in Figure 4.1 (b) below.



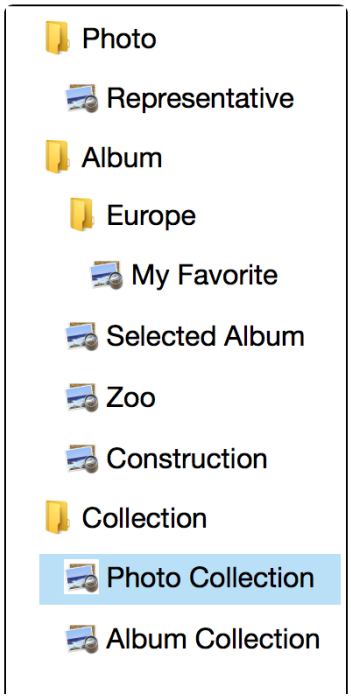


Figure 4.1(a) The unexpanded directory tree Figure 4.1(b) The expanded directory tree

The main processing flow when a user accesses the directory tree resource management interface is as follows:

1. When a logged-in user accesses a page that contains a directory tree, the client sends a request to the server together with the information about the user

2. After the server receives the request and the user data, it submits them to the GetTreeServlet for processing.

3.The GetTreeServlet calls getTreeByUserName method to access the database, gets the user's tree node collection, which is simplified after got, then only the relevant information is extracted, such as node ID, name, type, etc., and then passes it to the corresponding JSP.

4. The JSP extracts the relationship among each node from the received data (the system default photo, album and collection folder as the base), and then constructs the corresponding HTML code, and finally returns it to the client page.

5. The client page re-uses JavaScript and CSS to display directory tree.

This processing flow mainly describes the initialization of directory tree resource management interface. But there is a foreseeable problem: some parts of the expanded directory tree may be lost while refreshing, due to the non-real time data transmissions. Also, how to save the current status of expanded directory tree before

sending request is an important part of this module. Therefore, I have introduced a variable to save the expanded status of all folder nodes in the directory tree while transmitting with service. Some background codes for accessing the directory tree resource management interface are shown in Appendix B.1.

4.1.2 Storage space management subsystem

Storage space management subsystem is mainly used when normal users upload photos and administrators perform storage management or database backup. Because this part involves system security and key module confidentiality, only the operations when users upload photos are described below.

The main processing flow of the storage space management subsystem when users upload photos is as follows:

1.The upload function module calls the createResourceEntityAndURL method of the storage space management subsystem to apply for allocation of the storage space.

2. Using createResourceEntityAndURL method and getBlockAuto method, the distributed block instance is obtained.

3.With getBlockAuto method, all block information stored in the database is obtained, and then a non–full photo type resources block is picked and returned.

4. With the createResourceEntityAndURL method, the system configuration, the block instance and URL are allocated to the photo resource, and the relevant information is stored into the space resource table. The ResID (unique identification ID) of the record in the space resource table is obtained. And then the URL address and ResID are returned.

5. Finally, the uploading function module stores the photo to the specified URL and saves the photo's UserID, AlbID, ResID and other related data to the corresponding database table.

This procedure describes the most important part of the resource management module of storage space management subsystem. Before uploading, the system needs to find a non-full block for saving the uploaded resource, and the selected block are only allowed to save the same type resources as the uploaded one. For example, while uploading photos, the system must find a non-full photo block. After that, the system will extract the block's information from database (including Service field and VirtualDisk field), and combine them to get a complete URL. Then save the resource

to the specific URL and save relevant information in database. Otherwise, if there is not non-full block, the system will automatically generate a new empty block, and save block's information in space resource table. Part of the codes for the storage management subsystem is shown in Appendix B.2.

4.2 Implementation of the major modules

4.2.1 Implementation of the User and Authorization Management module

The *User and Authorization Management module* not only realizes login, logout and management account, but also achieves the user authorization management to improve the system's stability and security.

When a user accesses to a specific interface or attempts to use high-privilege operations (such as creation of photography competition and photography competition reviewing), the server backstage will check the user's privilege level and determine whether he can perform the operation. Without the corresponding operation privilege, error prompt is popped and the control will return to the previous page. Moreover, when the user attempts to operate on he own password, information, photos and other data, the current user's login status will also be verified to check whether these resources belong to the user. If the verification is failed, the operation will not be done. This two-tier review of privileges can avoid malicious attacks, hotlink and other system security risks.

The user can log in by clicking the Login button at the top right of the page or by visiting the login page directly. After successful login, users can also use drop-down menu options at the top the page for account management and other related operations, including changing password, modifying settings and logout. The effect of the drop-down menu is shown in Figure 4.2.

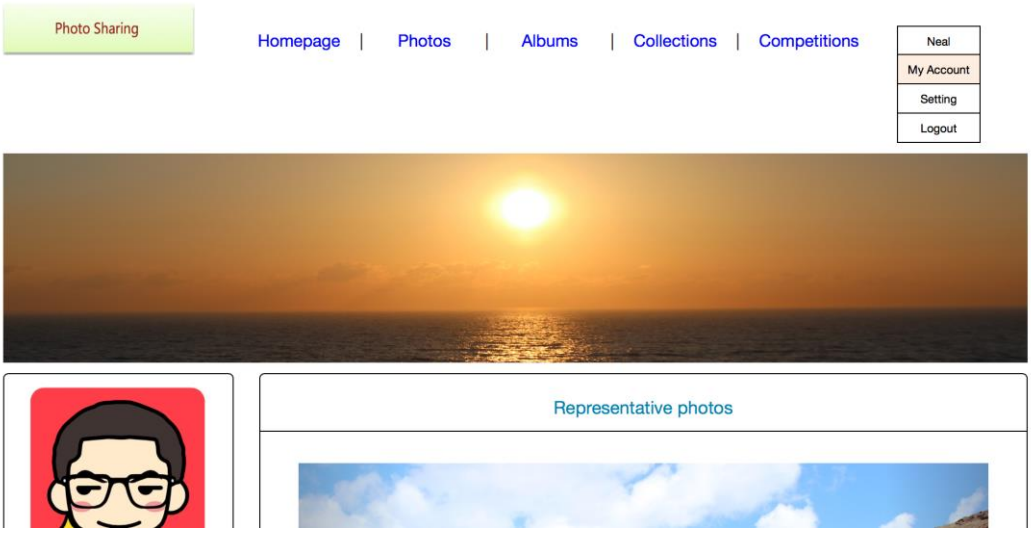


Figure 4.2 Account Management drop-down menu

The login interface is shown in Figure 4.3, which provides not only the registration function, but also the password retrieving function. Visitors can click on the “Create a new account” link below to apply for becoming registered users.

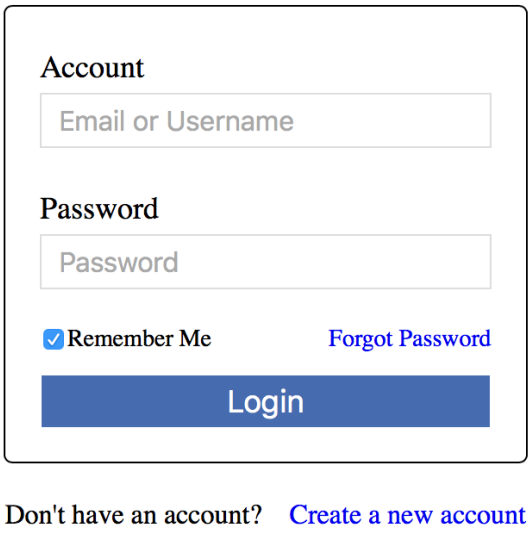


Figure 4.3 Login interface

The main process of the module’s processing when a user logs in is as follows:

1. When the server receives the login request and related information submitted by the client page, it uses URL mapping of the controller to filter URL, and transfers them to the LoginServlet for processing.
2. The Servlet calls the doLogin method and submits the received data to it for

processing.

3. doLogin method checks the user's request and information. If there is an error, it will return wrong message and jump to login page. If there is no error, it will call loginByUserEmail method to verify the user name and password.

4. loginByUserEmail method first calls userDAO to retrieve the database to determine whether the user name (email) exists. If it does not exist, it returns false, and if it exists, it will continue to do the password verification. If the authentication succeeds, it returns true and saves the login status of the current user in the cookie. If the authentication fails, it returns false.

5. The doLogin method then evaluates the return value of loginByUserEmail. If it is true, it will return the user name and ID information to the client and jump to the user's personal homepage, and if it is false, it will return an error prompt and jump back to the login page.

During logging in, the *User and Authorization Management module* firstly needs to check the request and input information, this can reduce the redundant access of database and avoid malicious attack. Then it will verify password by compare input data with database records. After successful login, the system will automatically save current user's login status for future authorization. And the login status also can protect the system for Hotlinking and exceeding authorized access. The main code when the user logs in is shown in Appendix B.3.

When the user attempts to change his password, he needs to enter the correct current password and the new password twice. The operation page for modifying the password is shown in Figure 4.4.

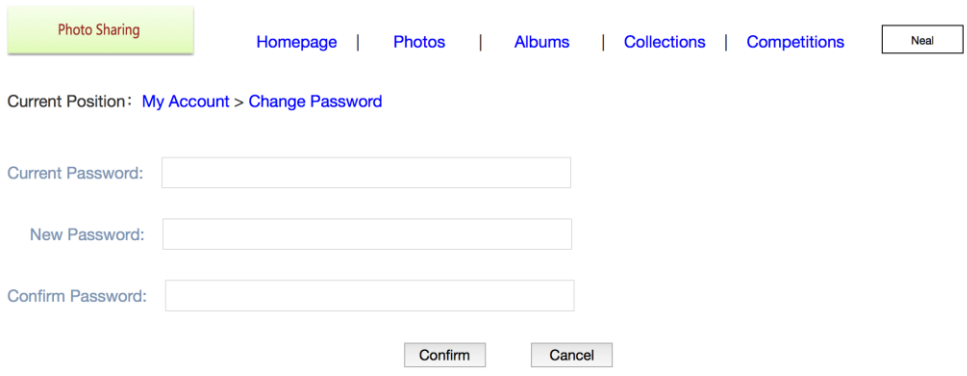


Figure 4.4 The user interface for changing the password

The main process of the module's processing when a user changes his password is as follows:

1. When the server receives the change password request and information submitted by the client page, it uses URL mapping of the controller to filter URL, and transfer them to the ChangePasswordServlet for processing.
2. The Servlet calls the updateUserPassword method and submits the received data to it for processing.
3. updateUserPassword uses getCurrentUser method to obtain the current login user instance, and then passes the relevant data to the changePassword method for validation.
4. changePassword method firstly determines whether two new passwords are the same. If not, it will interrupt and return false; if yes, it will continue to verify the user's old password. If the password does not match that in the database, it will interrupt and return false. When both the two previous judgments are true, the method will encrypt the new password, and call the user DAO update method to update the data in the database.
5. Finally, the updateUserPassword method returns corresponding results to the client based on the returned boolean value.

Although the client page have already verify whether the two new passwords are the same by using JavaScript before sending request, but it is still necessary for the server side to check it again to prevent malicious attack. And the password saved in database has already been encrypted, so if the server wants to verify or change password, the input data must be encrypted by the same way before verify or update. Part of codes to change the password is shown in Appendix B.4.

4.2.2 Implementation of the Personal homepage

One of the core demands of the photo sharing and photography competition management system is to share photos with others. The functions of this module is to show the user's personal information and photos, and to allow others to access, view the photos and communicate with him.

This module's interface is the user's personal homepage. The left part of the page shows his personal information, including the name, the number of likes obtained, the number of followers, the total number of photos, the total number of albums and so on. Under the user's personal information, you can also search for other users and jump to

their personal homepages. At the right of this page, there are the owner's representative works and selected albums. The representative works is a set of photos that is selected by the owner by moving them into a special folder or are selected by the system which are of the top three photos in the rank of descending order of click numbers. The selected albums are chosen in a similar way. The representative works also have a rotation function. The layout of the personal homepage is shown in Figure 4.5.

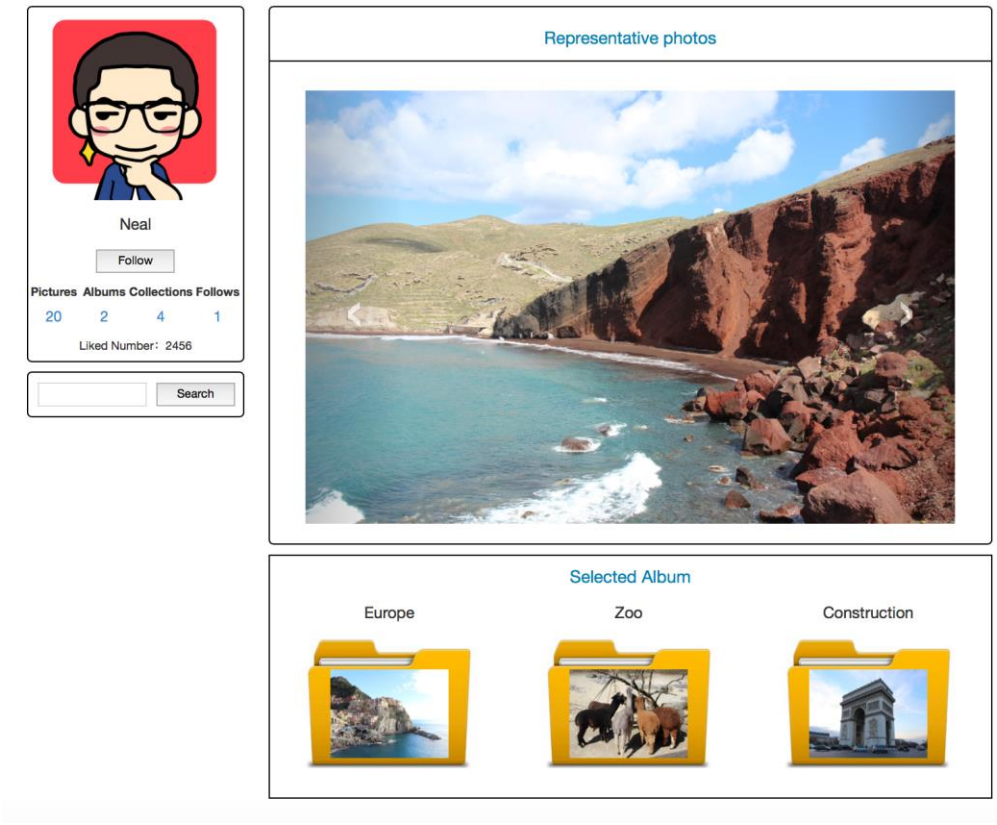


Figure 4.5 The personal homepage

The main steps for obtaining the user's representative works are as follows:

1. When the client attempts to access the user's personal homepage, it sends relevant requests and data to the server. The server-side backstage then submits the requests and data to the HomepageServlet for processing according to the URL mapping rules.

2. The Servlet calls showUserhome function, and uses incoming parameters and getUserByUserName method for query the database to get the user's User class instance.

3. showUserhome calls getSysTreeNodeByNodeNameAndUserID, getAbumByTreeNodeID and getPhotosByAbumIDAndSortByViews to access

the database. Through the TreeNode table, the Album table, and the relationship among data in the Photos table, The method obtains the user's representative work folder, and take outs the list of objects containing up to 3 photos in the rank of descending order of click numbers.

4. Then the method determines the size of the list. If the photo number is less than three, it uses getPhotoByUserIDAndSortByViews to take out the lack part, and mergers the new list and the previous list.

5. Finally returns the list of representative photos.

The automatically generation of representative works may cause some comprehension difficulties. Therefore I would like to explain the major parts of this procedure (step 3 and 4). While obtaining the user's representative works, the system firstly will extra three photos with the highest click numbers from current user's representative work folder. If the photos in current user's representative work folder are less then three (e.g. only have two photos), then the system will extract the highest click number photos for all the rest folders for supplement (e.g. find one from all the rest folders for supplement). Part of the codes to obtain the main representative works of the user is shown in Appendix B.5.

Users can also comment on the photos while browsing them. The comment interface is shown in Figure 4.6.

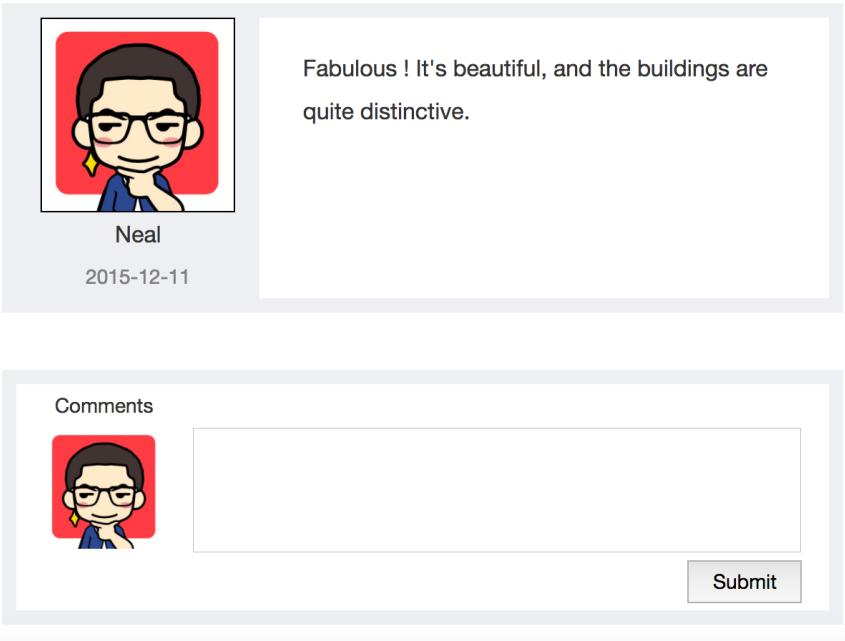


Figure 4.6 The user interface for comment

The main process of the module's processing when a user comment photos is as follows:

1. After the user completes comments in the photo enlarge page and clicks submit, the client sends a request and related information to the server-side designated Servlet for processing.
2. CommentServlet calls postComment function to obtain the current logged-on user's instance object, create a new Comment instance object and store relevant information submitted by the client page in it. At the same time it calls setCreateTime method to save the time of the comment operation, and then calls comment DAO to insert the relevant data into database table.
3. Finally, the method returns the Comment instance to servlet and JSP, so that the newly published comments are displayed on the new page.

The process of user comment photos is quite simple, and the key point of this process is to create an instance for the comment operation and it's contents, then persistent it into database. The core codes when commenting are as shown in Appendix B.6.

4.2.3 Implementation of the Photo Management module

This module realizes user operation and management of photo resources. It is closely related to the *Album Management module* and the *Collection Management module*, as the basis for their implementation.

In the photo management page, the user can do operations on the photos through directory tree on the left side and operation buttons on the upper right side, including: uploading photos, editing information, moving, copying, deleting and participating in competitions. The user can also click on the photos on the right to enlarge them. The photo management page is shown in Figure 4.7.

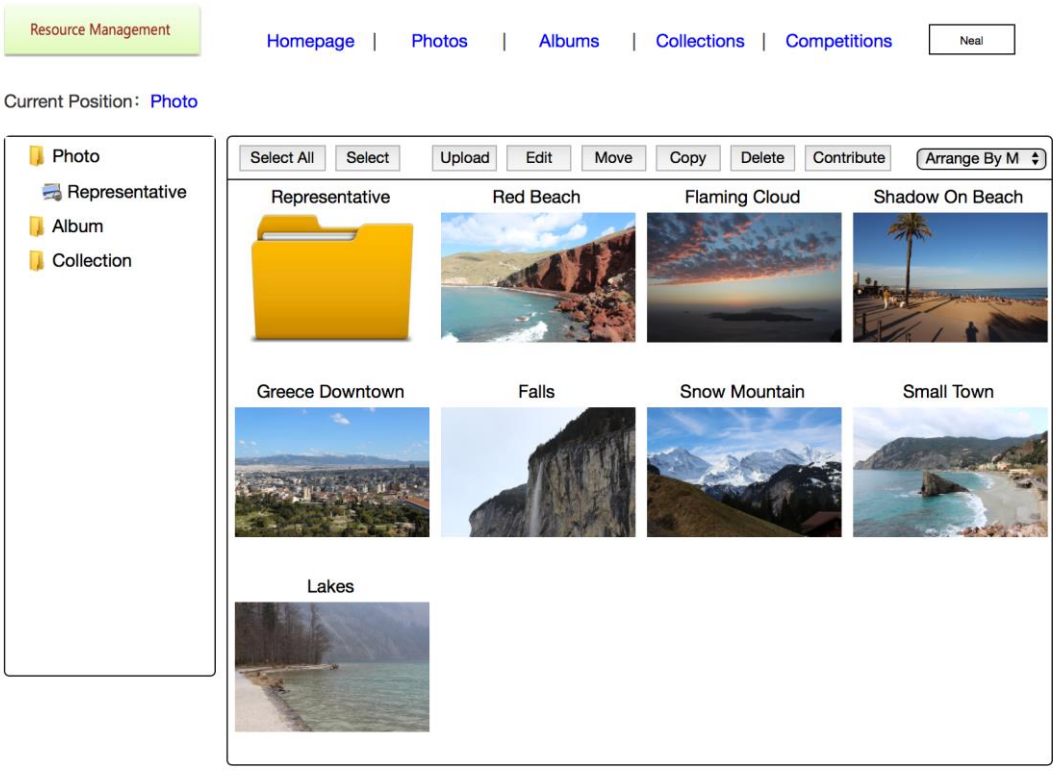


Figure 4.7 The photo management page

When the user tries to operate on the photos, he needs to click the "Select " button on the left side of the button bar, and the page will display the hidden check box, so that the user can select the desired photos. Then the user can perform the 6 operations described above. In the following, only the moving function is described, and the function interface for moving photos is shown in Figure 4.8.

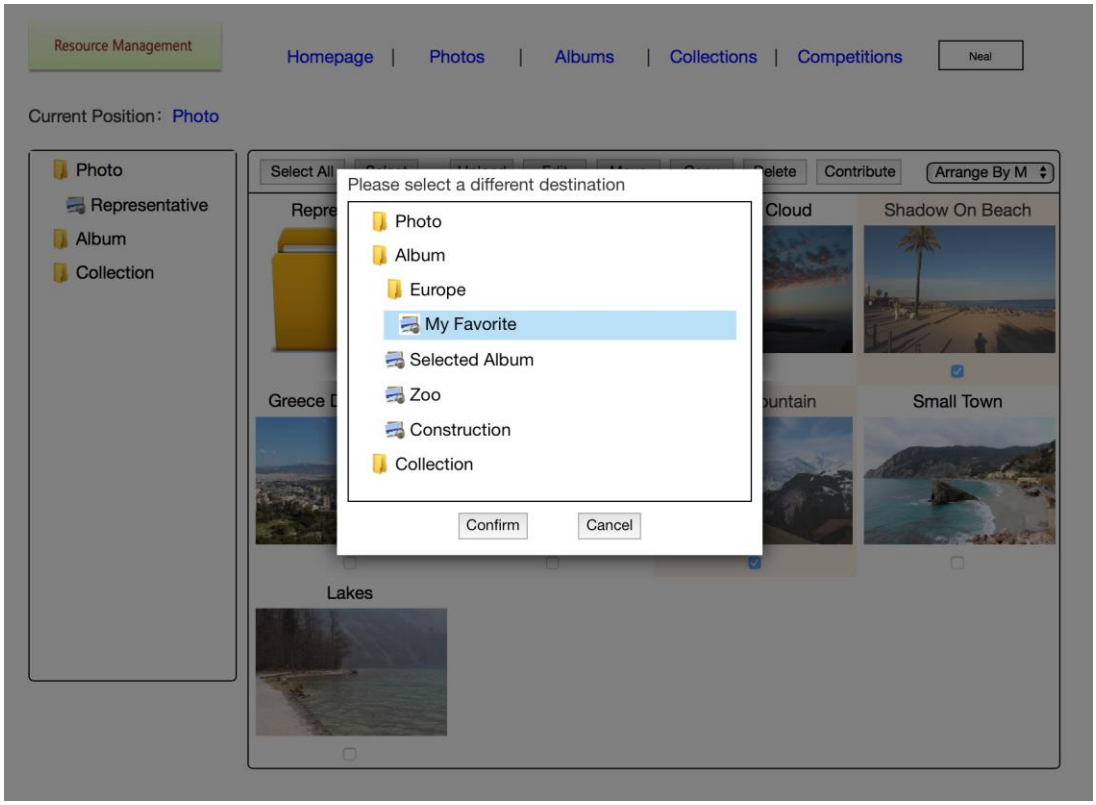


Figure 4.8 The user interface for moving photos

The processing steps of moving photos is as follows:

1. After the user selects the photos to be moved and clicks the Move button. This page will call JS function to visualize the directory tree for the user to select the desired destination folder.

2. After the user selects the target folder and clicks OK, this module will call the JS function again and use AJAX technology to send data and a request asynchronously to the server.

3. They are transferred to the MoveItemServlet for processing. The Servlet uses the incoming object file's TreeNodeID to obtain the album's Album type instance, and then passes PhotoID and AlbumID to the movePhoto method for processing.

4. The movePhoto method checks whether the moved photos and the destination album are both belong to the user. If not, a false will be return directly. Then it checks whether the destination album is the same as the one, which contains the photos to be moved. If yes, a false will be return directly.

5. If the previous two judgments are true, the total number of photos of the album to which the moved photo belongs is decremented by one, and the total number of photos of the target album is incremented by one, the album ID of the

moved photo is changed to that of the target album, and the database information is updated.

6. Finally, the MoveItemServlet will send new data back to the JS function, which will use the data to modify page elements, with the deleted photos no longer shown.

In this procedure, the system updates database three times in step 5, including: updates PicNum (the total numbers of photos) of the target album's data in Album table, updates the PicNum of the current album's data in Album table and updates the AlbID (the ID of the album containing this photo) of the moved photo's data in Picture table. If one of three updates fails, it may cause some big mistake or bugs, because the other two updates may still successfully finished. Therefore, I have combined these three operations in a method, and if any update fails, the system will make roll back the other two executed updates. Part of the codes for moving photos is shown in Appendix B.7.

4.2.4 Implementation of the Album Management module

Album Management module's function is similar to the *Photo Management module*. In the album management page, the user can perform operations such as modifying, moving, copying, deleting photos and albums, creating albums, but cannot uploading photos. Because the implementation detail and methods are similar to the *Photo Management module*, to avoid redundancy, only the modifying function is described below, and the user interface for modifying albums is shown in Figure 4.9.

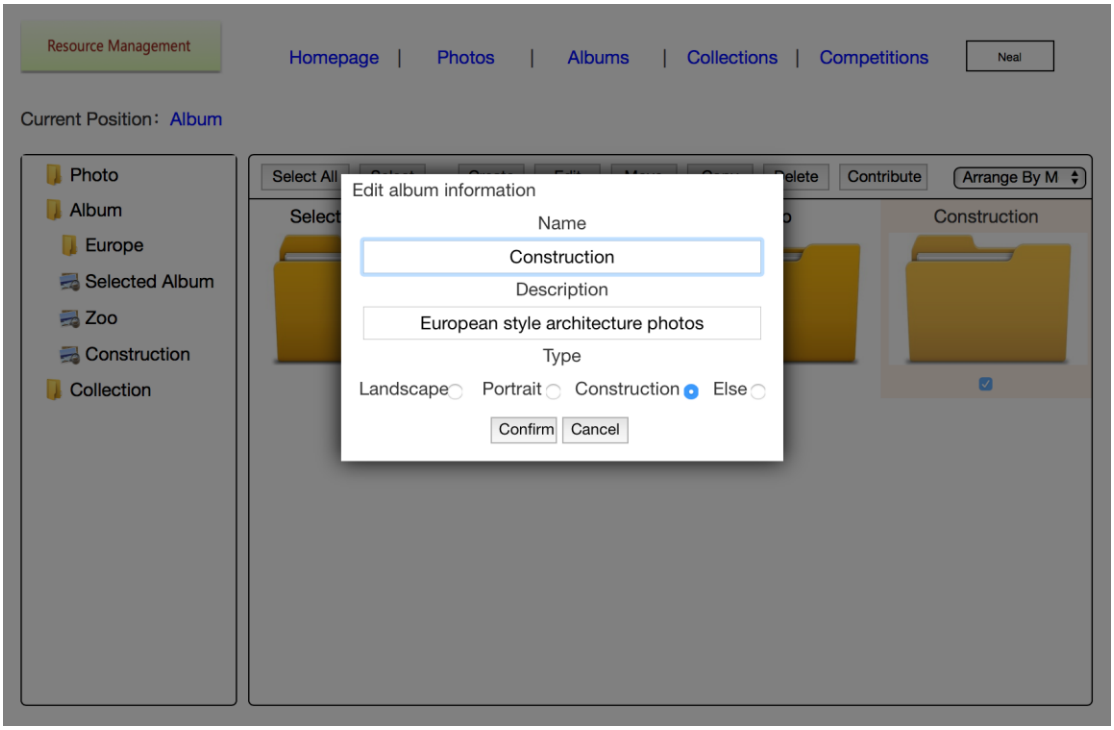


Figure 4.9 The user interface for modifying albums

The processing steps of modifying albums is as follows:

1. After the user click the modify button, JS will use AJAX asynchronous technology to communicate with the corresponding server, and send a request, then new album's information are sent to the ModifyItemServlet for processing.
2. The ModifyItemServlet calls updateAlbum method to modify the album information.
3. Through the passed AlbumID data, the method obtains the target album class instance, and then modifies the instance's name, description and class attributes.
4. At the same time the method checks whether the album belongs to the current login user, and if not, it will return error "insufficient privilege."
5. If the album belongs to the current login user, it will call DAO to update the database, and replace the old data with the new data.
6. JS gets the results and updates the page elements.

In this process, the use of album class instance may looks like redundant, but it can protect the original data in database before update and avoid other unexpected errors. Furthermore, It is better to check weather the album belongs to the current login user before createing album class instance, which can avoid unnecessary system overhead. Part of the codes for modifying albums is shown in Appendix B.8.

4.2.5 Implementation of Collection Management module

Collection management module is defined to provide users with the collection-related functions. When users collect their favorite photos or albums, they can manage them in this page. Users can click on items in collection page to view or delete them. However, the resource is not actually owned by the user who collects it, so when the user deletes a resource, the actual photo or album is not deleted, only the collection information data associated with the resource is deleted. The function interface for deleting collection item is shown in Figure 4.10.

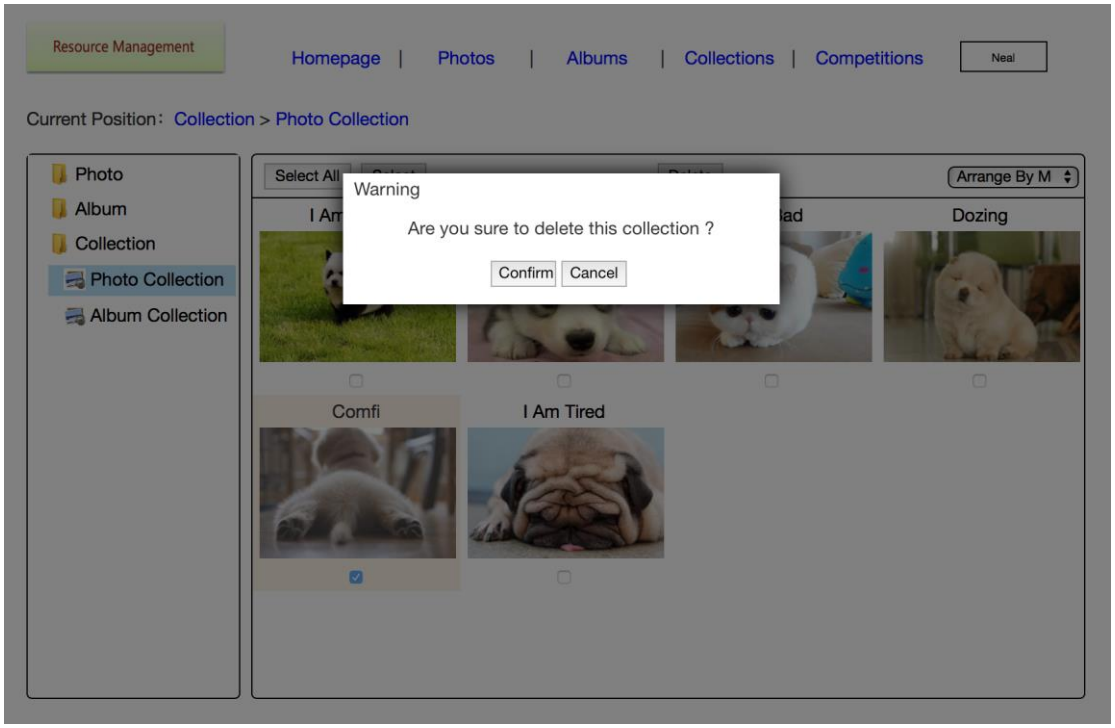


Figure 4.10 The user interface for deleting collection items

The main steps of the module's processing when a user deletes collection items are as follows:

1. When a user deletes a collection item, the client page will send a request to the server. The server submits the request and data to the DeleteItemServlet for processing.
2. The DeleteItemServlet calls removeCollection method to delete the specified collection item and related data.
3. The method uses sourceID, sourceType and UserID to extract the corresponding Collection instance object.
4. The method deletes the data in the database, and modifies the user's total

number of collection items.

5. The method returns the results of the operation, and uses AJAX asynchronous transfer technology to modify the elements of the collection page, and no longer shows the collection item deleted.

The same as discussed in 4.2.3, the delete of collection data and the update of current user's total number of collection items also need to be combined for potential roll back when some error happens. Part of the codes for deleting collections is shown in Appendix B.9.

4.2.6 Implementation of the Competition Management module

The *Competition Management module* mainly provides users and administrators with photography competition-related functions. The super administrator can create a new competition or modify/delete an existing competition through this module. And ordinary users can view their participated competitions and contributed photos by using this module. The competition management page is shown in Figure 4.11.

NO	Competition Name	Start Date	End Date	Announcing Results	Sponsor	Reviewer		
001	2015 Tongji Staff Photography Competition	2015-03-01	2015-06-10	2015-07-10	Li Ming	Zhang Jianguo, Li Fu, Zhang Weili	Modify	Delete
002	2015 "The Most Beautiful Europe" Photography Competition	2015-03-10	2015-08-31	2015-10-01	Wang Zhiying	Liu Bin, Wang Wei, Shao Xiangqian	Modify	Delete
003	The 3rd Software College Photography Competition	2015-03-15	2015-05-31	2015-06-15	Xu Ming	Hu Shiping, Zhao Junfeng	Modify	Delete
004	2015 "UnionPay" Photography Competition	2015-06-01	2015-11-10	2015-12-01	Li Li	Cheng Qingxia, Cheng Meijuan, Zhang Hanqing	Modify	Delete
005	The 5th "Li Shui" Photography Competition	2015-08-06	2015-12-12	2016-1-1	Zhang Chaoyang	Zhou Jing, Wang Jingjing, Wang Xi	Modify	Delete
006	The 1st Tongji University Mobile Phone Photography Comeptition	2015-09-01	2015-12-15	2016-01-10	Zhang Jiang	Gao Bo, Liu Wei	Modify	Delete

Figure 4.11 The competition management page

When a super administrator clicks the "Create Competition" button in the upper left

of the competition management page shown in Figure 4.11, the creating competition page will be jumped to, which is shown in Figure 4.12.

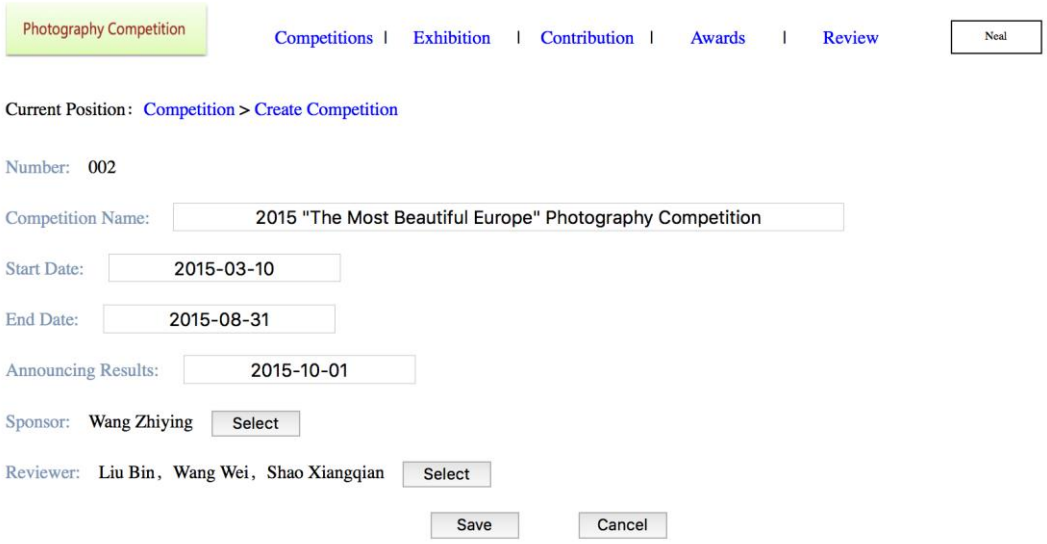


Figure 4.12 The user interface of creating competition

The main steps of the module's processing when a super administrator creates a new competition is as follows:

1. The super administrator sends a request of creating competition to the server after filling in the related information.
2. The server submits the request and the relevant data to the CreateCompetitionServlet for processing according to URL mapping in the controller.
3. The CreateCompetitionServlet calls newCompetition method to create a new competition.
4. The method checks whether the current login user has the privilege to create a new competition. If yes, the relevant information is stored into a new Competition instance object, and the insert function of competition DAO is called to save the instance object into the database.
5. Results are returned, and the interface jumps to the competition management page.

Before the client page sends creating competition request, it will use JavaScript and AJAX to visualize the sponsor and reviewer's list respectively (after click "select"), and also will use JavaScript to check weather the input information is corresponding to the required format. Part of the codes for creating competitions is

shown is Appendix B.10.

4.2.7 Implementation of the Contribution module

By using this module, users can contribute photos in two ways. (1) select the photo they want to submit in the photo or album management page, and then click on the "Contribute" button, followed by selecting the competition to contribute to. (2) select one specified competition in the competition page, then upload photos and provide relevant information for the photos to the contribution. Figure 4.13 shows the contribution page of the photography competition management subsystem.

Figure 4.13 The contribution page

The main steps of the module's processing when a user contributes to a competition is as follows:

1. When a user confirms his contribution photo, the page will calls Node.js to process the uploaded photo, and sends a request to the server side.
2. The server sends the request and related data to the ContributeServlet for processing according to the URL mapping saved in the controller.
3. The ContributeServlet calls uploadPhoto method for incoming data processing.
4. The uploadPhoto method requests the storage space management subsystem to assign storage space, URL and the corresponding space resource ID.
5. The method uploads the photo resource to the target address based on the returned URL.

6. At the same time, the method saves the relevant photo information and competition relationship information to the database table.

7. Finally, the contribution page will show the contribution results to the user.

In this procedure, the user must input the relevant information of uploading photo. Otherwise, the server will deny the uploading request. Part of the codes for contributing is shown in Appendix B.11.

4.2.8 Implementation of the Review module

The *Review module* is only available for reviewers. After a reviewer accesses this module and selects a specific competition, the user interface will automatically jump to the review page. Then reviewer can click on the drop-down box below the photo to review and select award. The function interface for reviewing is shown in Figure 4.14.

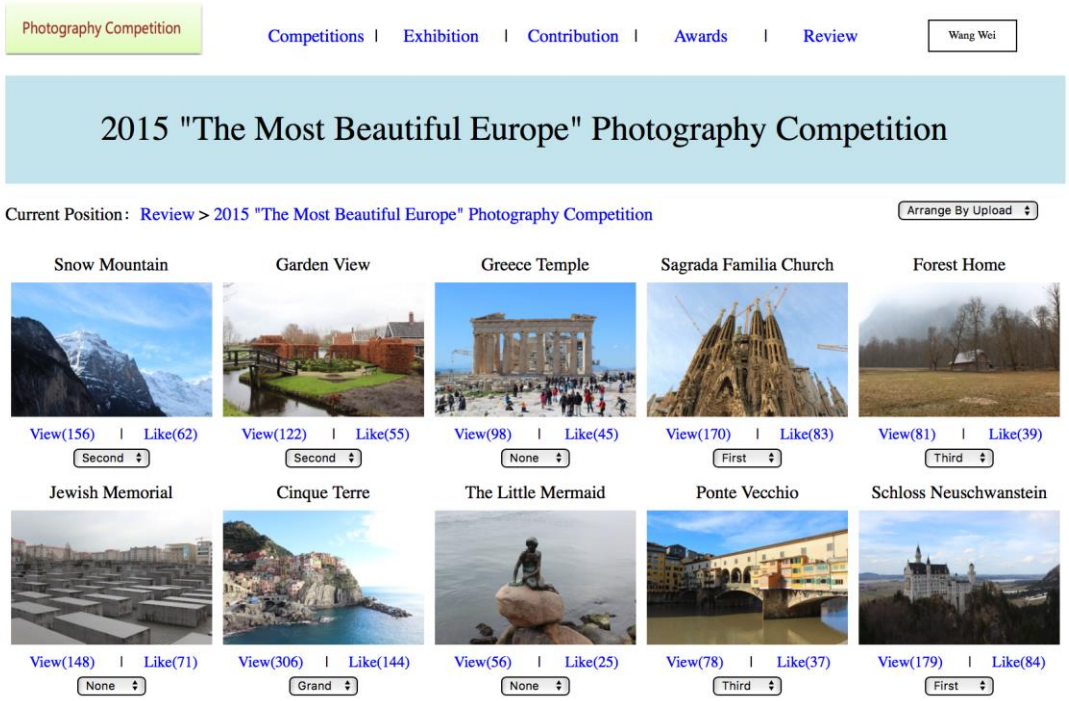


Figure 4.14 The user interface of reviewing

The main steps of the module's processing of reviewing is as follows:

1. The reviewer sends a request and related information to the server, which in turn submits it to the ReviewServlet for processing.

2.The ReviewServlet calls the changePrize method to modify the award data.

3. The method checks the access level of the current user, and then uses the getCompetationRelationByPhotoIDandGameID method to get the CompetitonRelation instances of the target photos, and then modifies the Prize

attributes in the instances and update the database with the results.

4. Finally, the results of these operations are fed back to the reviewer.

Once the reviewer select an award from the drop-down box, the reviewing request will be send to the server, and the reviewer dose not need to click any “submit” or “confirm” buttons. After that, the server will return the operation result to the client page, and showing the result by popup window. This way of processing is convenient for plenty of review operations, but is also may cause some out of sync problems between the client page and the server database when the network is poor or interrupted. Part of the codes for reviewing is shown in Appendix B.12.

4.2.9 Implementation of the Display of Competition module

The functions of the *Display of Competition module* can be fulfilled by the support of the exhibition page. When a user accesses the exhibition page and selects a specific competition for displaying, all the thumbnails of the photos for that competition will be displayed. The user can browse the photos or give like’s by clinking on the “View” or the “Like” hotlinks below the thumbnails. The user can also select the sorting methods through the drop-down selection button. The exhibition page for a specific competition is shown in Figure 4.15.

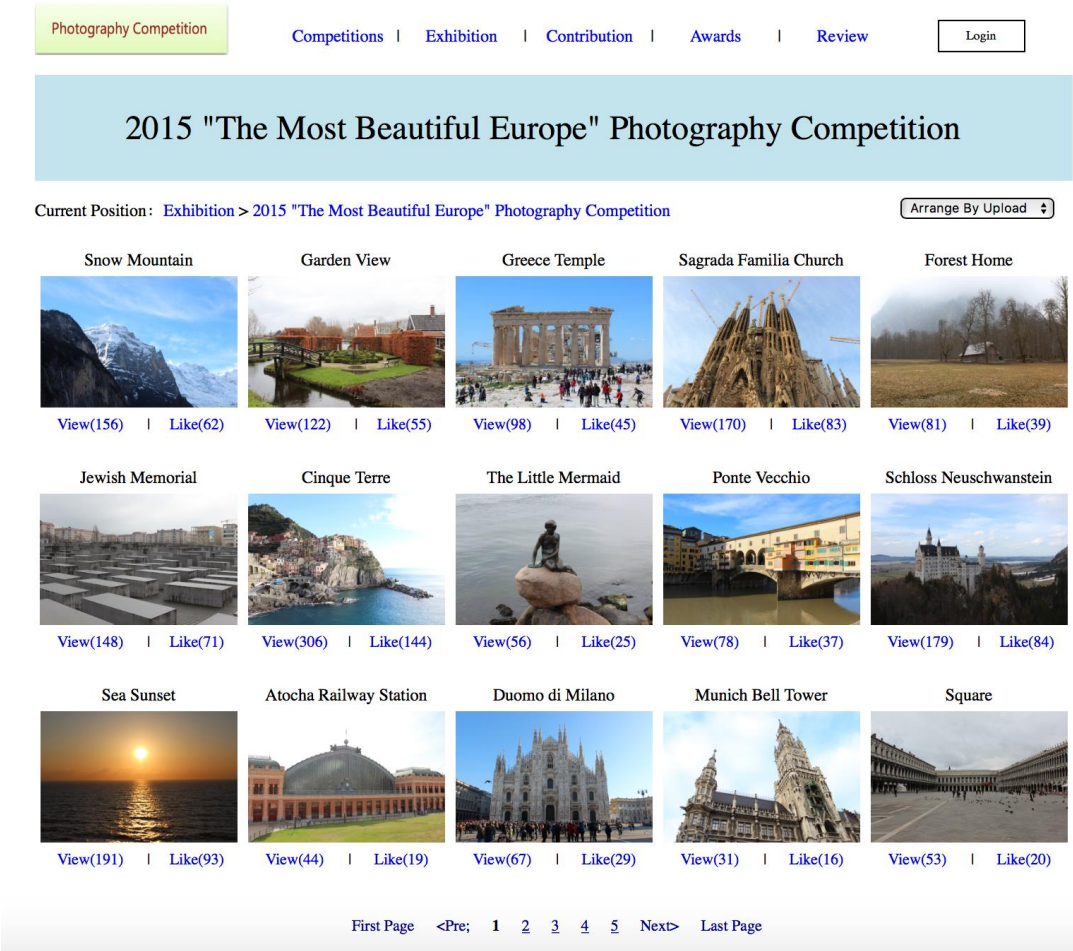


Figure 4.15 The exhibition page

The main steps of the module's processing when a user sets sorting by the upload time on the exhibition page is as follows:

1. When a user selects arrange by upload time, the client page will send a request and relevant data to the server, and the server will send it to the GetPicturesServlet for processing.
2. The servlet calls the sortPhoto method for incoming data processing.
3. The sortPhoto method passes the passed in parameters in to the key-value pair.
4. It calls the getPhotosByCompetitionID method to retrieve the data sorted in descending order of the upload time in the database and save the data in a collection of Photo object.
5. The results will be transmitted to the JSP, and JSP modifies HTML page elements based on the relevant information, and returns the new page back to the client.

In these processing steps, only the specific photos of current page will be obtained and displayed. For example, the first page will only request the first 15 photos ordered by upload time, and the second page will request photos ordered by upload time from 16th-30th. This can significantly improve the efficiency of data transmission between server and client pages and reduce the redundant query for database. Part of the codes for arranging by upload time is shown in Appendix B.13.

4.2.10 Implementation of the Display of Competition Result module

This module mainly has three functions: to show the winning works, browsing photos or giving photo likes through the “View” and “Like” link below. The awards page of this module is shown in Figure 4.16.

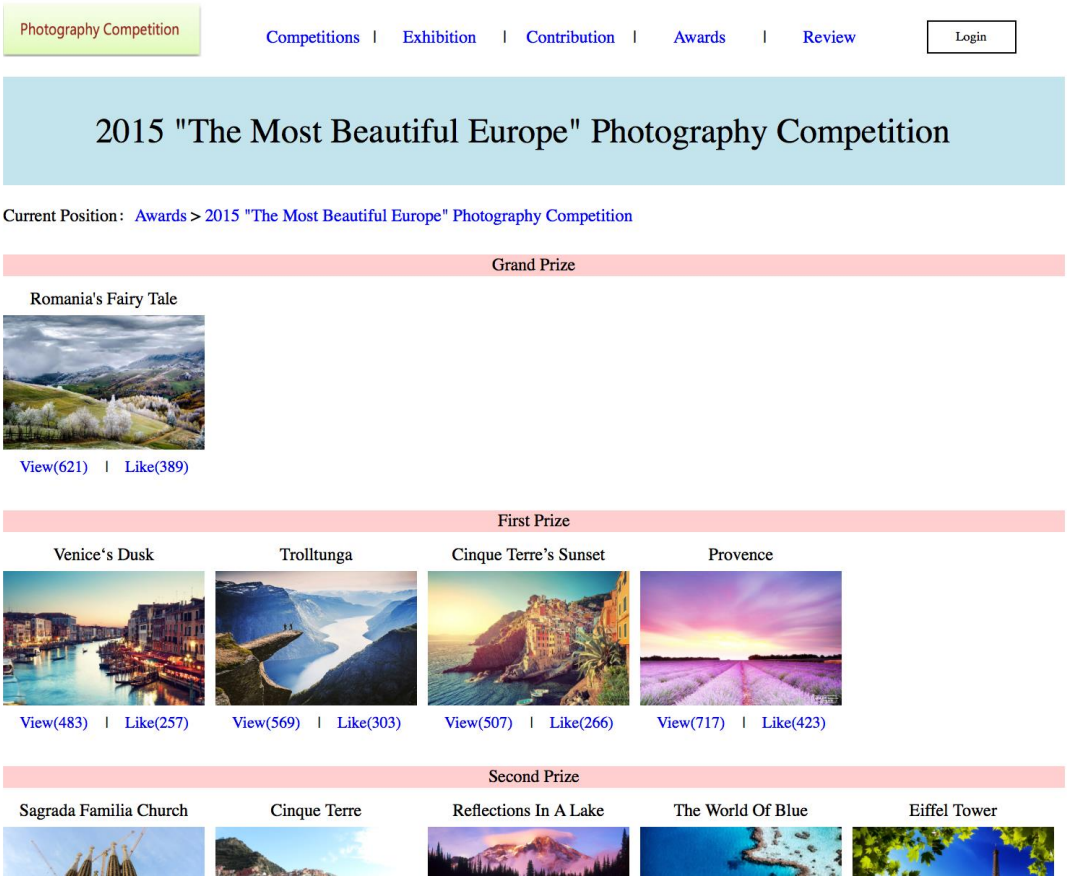


Figure 4.16 The awards page

The main steps of the module’s processing when a user gives a photo like’s is as follows:

1. When a user gives his favorite award-winning photo like’s, the client page will use JavaScript functions and AJAX technology to send a request and related data to the server.

2. The server submits the request to the LikeServlet for processing.

3.The LikeServlet calls the updateLikeNum method to change relevant information of the target photo, which is saved in the database, and then the number of the photo's likes is increased by one.

4. The LikeServlet returns results to the JavaScript function through asynchronous AJAX, and then JavaScript function refreshes the number of likes based on the returned results.

There is one thing that can be improved in the future work. For now, all the visitors can give their favorite photo Likes, and they can like the same photo multiple times. It will be more reasonable if one user can give at most only one Like to each photo. Part of the codes of giving a photo like's is shown in Appendix B.14.

Chapter 5 System test

5.1 Running environment

During the development of the system, I choose Java as my programming language, because it is object-oriented and widely used for web-based programing. Another reason is that I am more familiar with it in comparison with other programming languages.

And I choose MySQL as database because it is a lightweight database (compare with Oracle or DB2) and it is free. The most important reason is that it more than enough in meeting the demands of our system.

I choose MyEclpse as Development tool is because it is very powerful and include a lot of tools.

I user Apache Tomcat because it is necessary for the Java web development.

And the StarUML and log4j is just widely used in their working area.

5.2 System function test

System function test, as an important part of the system development, allows developers to detect whether the system has achieved the desired functions, and to find out shortcomings and vulnerabilities of related functions [25]. After the test, the developers can remove the bugs and make improvements of the system, ensuring that the system reach the quality standard [26]. Therefore, we use a variety of test cases for the relevant functional testing on the system.

1. Test of the User and Authorization Management module

The *User and Authorization Management module* is tested by use the administrator and ordinary user login, access to the review page, and deletion of other user photos as test case. The details of the test are shown in Table 5.1.

Table 5.1 Test cases and results of the User and Authorization Management module

Test projects	Test cases	Expected results	Test results
Administrator login	User name：Neal Correct password：12345678 Entered password 12345678	Successfully login system, and jump to personal homepage	User login function is working properly
Users access to the review page	Login with the general user account, click the link of the review page	Jump to false prompt page, and then jump to the personal homepage	Ordinary users can not access the review interface
Delete other users' photos	Add relevant parameters in the URL, and try to delete the photo of other users	Return an error message	User can not operate on other user's resources

2. Test of the Personal Homepage module

Table 5.2 describes the test of the *Personal Homepage module*, with searching for other users' personal homepages and viewing representative works as test cases.

Table 5.2 Test cases and results of the Personal Homepage module

Test projects	Test cases	Expected results	Test results
Search other users' personal homepage	Enter user name: TinyBear	Successfully jumps to TinyBear's personal homepage	Search and jump to TinyBear's homepage normally
View representative works	Go to the personal homepage and click the left and right scroll icons	Left or right can scroll to show the next representative	Representative photos' scroll function is working well

3. Test of the Photo Management module

The *Photo Management module* is tested by use uploading photos, moving photos, copying photos and participating in the competition as test cases. Description and test results are given in Table 5.3.

Table 5.3 Test cases and results of the Photo Management module

Test projects	Test cases	Expected results	Test results
Uploading photos	Select local photos to upload	After successful uploading, the uploaded photos are displayed in the page	Photo uploading function is working well
Moving photos	Select photos and move to another album folder	The current page no longer displays the photo, and views the target album, including the photo	Photo moving function is working well
Copying photos	Select the target photo to copy	Create a new photo that is the same as the destination in the current directory	Photo copying function is working well
Participating in the competition	Select photos for participating in the competition	Inquire about participated competitions in competition management page	Function of participating in the competition is working well

4. Test of the Album Management module

Table 5.4 describes the test of *Album Management module,* with creating a new album and editing the album information as the test cases.

Table 5.4 Test cases and results of the Album Management module

Test projects	Test cases	Expected results	Test results
Create a new album	Create a new album, but do not enter the album name	Show unnamed album	Function of creating a new album is normal
Modify the album information	Modify album name	Show new album name	Function of modifying album information is normal

5. Test of the Collection Management module

Table 5.5 describes the test cases and related conclusions of the *Collection Management module.*

Table 5.5 Test cases and results of the Collection Management module

Test projects	Test cases	Expected results	Test results
Access collection	Click the collection item	Jump to the corresponding target interface	Function of accessing collection is working well
Delete collection	Select the collection item to delete	Favorite page does not display this collection target anymore	Function of deleting collection item is working well

6. Test of the Competition Management module

The *Competition Management module* is tested by use creating a new competition and deleting a competition by super administrator as the test cases. The description and test results of the test are shown in Table 5.6.

Table 5.6 Test cases and results of the Competition Management module

Test projects	Test cases	Expected results	Test results
Administrator creates a new competition	The administrator creates a new competition, and enters the relevant information	The competition management page shows the newly created competition	Function of creating a new competition is working well
The administrator deletes a competition	Deletes a competition with the administrator account	The competition management page does not show the deleted competition	Function of deleting a competition is working well

7. Test of the Contribution module

Table 5.7 describes the test cases and results of the test of the *Contribution module*.

Table 5.7 Test cases and results of the Contribution module

Test projects	Test cases	Expected results	Test results
Participate in a competition	Upload photos and choose the competition to participate in	Find the participated competition in the competition management page	Function of uploading photos and participating in the competition is working well

8. Test of the Review module

Table 5.8 describes the test cases and results for the *Review module*.

Table 5.8 Test cases and results of the Review module

Test projects	Test cases	Expected results	Test results
Review awards	Access the review page with the account of the reviewer, and review the photos	Refresh the current page, and reviewing results are reserved	Function of reviewing awards is working well

9. Test of the Display of Competition module

Table 5.9 describes the test of the *Display of Competition module*, with selecting sorting methods and giving like to photos as the test cases.

Table 5.9 Test cases and results of the Display of Competition module

Test projects	Test cases	Expected results	Test results
Select sorting methods	Select the photo sorting method in the works display page, with sorting according to the upload time, the number of browse and the number of giving likes	All sorting according to the relevant demands	Function of photo sorting is working well
Give like to photos	Give like to photos	The count of giving likes is increased by one	Function of giving likes is working well

10. Test of the Display of Competition Result module

Table 5.10 describes the test cases and the results for the *Display of Competition Result module.*

Table 5.10 Test cases and the results of the Display of Competition Result module

Test projects	Test cases	Expected results	Test results
Show the results of the competition	Visit the competition result page of the specified competition, view the winning works	The result page of the competition is displayed normally	Function of showing the results of the competition is normal

Chapter 6 Conclusion and Future Work

6.1 Conclusion

This paper describes the design and implementation of the J2EE-based photo sharing and photography competition management system. This system provides a user-friendly platform that highly meets the needs of photography enthusiasts and the current market. During the design and development of this system, we have completed the following work:

1. An in-depth investigation is conducted through communication with shutterbugs. And according to the investigation results, system requirements are analyzed by using UML and object-oriented methods, and the relevant use case diagrams are drawn.

2. The overall structure of the system is designed. The system is divided into two subsystems: the photo sharing subsystem and the photography competition management subsystem. The former consists of 5 modules: the *Personal Homepage*, the *Photo Management*, the *Album Management*, the *Collection Management*, and the *User and Authorization Management*. The latter consists of 6 modules: the *User and Authorization Management*, the *Competition Management*, the *Review*, the *Contribution*, the *Display of Competition* and the *Display of Competition Result*.

3. The detailed design is performed for each module, and the processing sequence diagram of each module is drawn, the processing steps of each module is also given.

4. The database design is fulfilled. The E-R chart and database tables are given. We have adopted a tree structure in the classification of resources, so that the resources and records in the database have a good relationship and structure.

5. A unique storage space management mechanism is proposed and incorporated into the system, which provides efficient and flexible storage management for resources on the server, and makes the file synchronization and transmission more efficient.

6.The UI of the system and each module are designed and implemented. Finally, the whole system is programed and tested, and bugs are removed based on the test.

6.2 The direction of further work

Although the design and implementation of the system have been finished, it is far from perfect, and there exists improvements that can make the system works better. For example, the webpages are not pleasing enough to the eye, and the correlation among multiple subsystems is not strong. Tables and structures of the database can also be further optimized. Besides, the system has not been optimized for performance in large workload, such as the response time, the maximum number of concurrent users, and etc. I believe there exists a large development space in the future.

References

[1] Lu, Qian. “In 2015, China's domestic tourism exceeded 4 billion, and the tourism industry contributed over 10% of GDP.” The State Council The People’s Republic of China, 4 Jan. 2016. 10 Feb. 2016. <http://www.gov.cn/xinwen/2016-01/04/content_5030593.htm>

[2] Tan, Jie Ni. “The Study on the Photography behavior of tourist.” Diss. Chongqing Normal University, 2011.

[3] Hong-tao, D. E. N. G. "Research on Processing Method of Tree Structure Information in Relational Database [J]." Journal of Jianghan University (Natural Sciences) 2 (2010): 016.

[4] Chung, Lawrence, et al. Non-functional requirements in software engineering. Vol. 5. Springer Science & Business Media, 2012.

[5] Tilkov, Stefan, and Steve Vinoski. "Node. js: Using JavaScript to build high-performance network programs." IEEE Internet Computing 14.6 (2010): 80-83.

[6] Juan, T. I. A. N., and X. U. Zhao. "Analysis & Consideration of MVC Design Model Based on J2EE [J]." Computer and Modernization 10 (2010): 015.

[7] Yang, Jin Cheng, L. Qiang, and Y. E. Han-Minb. "Realization of Web Treeview Component Based on Ajax." Journal of Shanghai University of Electric Power 26.2 (2010): 205-209.

[8] Sun, Chang Ai, et al. "Object-Oriented Requirements Analysis Based on UML." Acta Aeronautica ET Astronautica Sinica 24.1 (2003): 75-78.

[9] Alur, Deepak, et al. Core J2EE Patterns (Core Design Series): Best Practices and Design Strategies. Sun Microsystems, Inc., 2003.

[10] Lu, Rongxing, et al. "Study and implementation of MVC design pattern on J2EE platform." Application Research of Computers 20.3 (2003): 144-146.

[11] Fei, W. A. N. G. "Rationalize the View Tier of MVC." Science Technology and Engineering 22 (2009): 067.

[12] WU, Chenqing, and Zhenhua RONG. "Implementing Web Application Using JSP/Servlet Technology [J]." Computer Engineering 1 (2001): 066.

[13] Shilin, Xiao Aihua Wang. "A COMMON WAY IMPLEMENTS DATA ACCESS OBJECT PATTERN FOR J2EE APPLICATION [J]." Computer Applications and Software 9 (2005): 051.

[14] Gong, Fu Qiang. “Development and Application of User Authority Management System Based on RBAC.” Diss. Northwestern Poly-Technical University, 2007.

[15] Ramakrishnan, Raghu, and Johannes Gehrke. Database management systems. McGraw Hill, 2000.

[16] Tridgell, Andrew, and Paul Mackerras. "The rsync algorithm." (1996).

[17] Wikipedia contributors. "Rolling hash." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 4 Mar. 2016. Web. 17 Aug. 2016. <https://en.wikipedia.org/wiki/Rolling_hash>

[18] Wikipedia contributors. "MD5." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 19 Mar. 2016. Web. 21 Aug. 2016. <https://en.wikipedia.org/wiki/MD5>

[19] Tang, Xiao-Di, et al. "Design and implementation of remote file synchronization system based on difference." Computer Engineering and Design 31.20 (2010): 4389-4392.

[20] Hao,Cheng. “The Core Algorithm of Rsync”. CoolShell. 17 May. 2012. 13 Feb. 2016.
< http://coolshell.cn/articles/7425.html >

[21] Sun, Jie. “Design and Implementation of Distributed Storage and Management of Spatial Data” Diss. Jilin University, 2015

[22] XU, Chang-sheng, Chao DAI, and Li XIE. "Research of rapid development of web application [J]." Computer Engineering and Design 12 (2004): 037.

[23] Freeman, Eric, et al. Head First Design Patterns: A Brain-Friendly Guide. " O'Reilly Media, Inc.", 2004.

[24] ZHANG, Yu-fang, Xiang-qian Hu, and Zhong-yang Xiong. "Building treeview with recursive algorithm in JSP [J]." Computer Engineering and Design 1 (2005): 014.

[25] Myers, Glenford J., Corey Sandler, and Tom Badgett. The art of software testing. John Wiley & Sons, 2011.

[26] SHANG, Dong-juan, et al. "A Study of Test Cases and Reuse in Software Testing [J]." Computer Technology and Development 1 (2006): 021.

Appendix A: Database Tables

Table A.1 User table

Field	Description	Data type	Remarks
UserID	User ID	INT	Primary Key
Username	User name	VARCHAR	Not Null
Password	User password	VARCHAR	Not Null
Email	User email	VARCHAR	Not Null
BackupEmail	Backup email	VARCHAR	
Register	Register time	DATETIME	Not Null
Status	User level	INT	Not Null
Locked	Locked state	TINYINT	Not Null
PicNum	Total number of photos	INT	Not Null
AlbNum	Total number of albums	INT	Not Null
CollectNum	Total number of collections	INT	Not Null

Table A.2 Picture table

Field	Description	Data type	Remarks
PicID	Photo ID	INT	Primary Key
PicName	Picture Name	VARCHAR	Not Null
AlbID	Album ID	INT	Foreign Key
UserID	User ID	INT	Foreign Key
ComID	Competition ID	INT	
Class	Picture class	VARCHAR	
Location	Shooting location label	VARCHAR	
Description	Relevant description	VARCHAR	
Time	Shooting time	DATETIME	
Create	Uploaded time	DATETIME	Not Null
Modify	Modified time	DATETIME	Not Null

ViewNum	Total number of browse	INT	Not Null
LikeNum	Total number of likes	INT	Not Null

Table A.3 Photo resource table

Field	Description	Data type	Remarks
PRID	Picture resource ID	INT	Primary Key
PicID	Associated photo ID	INT	Foreign Key
ResID	Associated resource ID	INT	Foreign Key
Width	Width	INT	Not Null
Height	Height	INT	Not Null
DB	Storage space name	VARCHAR	Not Null
URL	Storage address	VARCHAR	Not Null

Table A.4 Album table

Field	Description	Data type	Remarks
AlbID	Album ID	INT	Primary Key
AlbName	Album name	VARCHAR	Not Null
Description	Description	VARCHAR	
UserID	User ID	INT	Foreign Key
Class	Album class	VARCHAR	
Location	Location label	VARCHAR	
Create	Created time	DATETIME	Not Null
Modify	Modified time	DATETIME	Not Null
CoverUrl	Cover storage location	VARCHAR	Not Null
PicNum	Total number of photos	INT	Not Null
ViewNum	Total number of browse	INT	Not Null
LikeNum	Total number of giving likes	INT	Not Null

Table A.5 Collection table

Field	Description	Data type	Remarks
ColID	Collection ID	INT	Primary Key
ColName	Custom collection name	VARCHAR	Not Null
SrcUserID	Collected user ID	INT	Not Null
DesUserID	Collecting user ID	INT	Foreign Key
SourceID	Collected object ID	INT	Not Null
Type	Collected object type	VARCHAR	Not Null
Create	Collected time	DATETIME	Not Null
Description	Description	VARCHAR	

Table A.6 Comment table

Field	Description	Data type	Remarks
ComID	Comment ID	INT	Primary Key
UserID	User ID	INT	Foreign Key
Type	Resource type of the commented object	VARCHAR	Not Null
SourceID	ID corresponding to the resource	INT	Not Null
Content	Comment content	VARCHAR	Not Null
Create	Created time	TimeStamp	Not Null
ReplyID	Reply to comment ID	INT	

Table A.7 Competition table

Field	Description	Data type	Remarks
ComID	Competition ID	INT	Primary Key
Title	Competition name	VARCHAR	Not Null
ManID	Competition administrator ID	INT	Not Null
Start	Start date	DATETIME	Not Null
End	End date	DATETIME	Not Null
Result	Published result date	DATETIME	Not Null
RevID	Reviewer ID	VARCHAR	Not Null

Locked	Review lock status	TINYINT	Not Null

Table A.8 Competition relationship table

Field	Description	Data type	Remarks
CRID	Competition relationship ID	INT	Primary Key
ComID	Competition ID	INT	Foreign Key
UserID	User ID	INT	Not Null
PicID	Picture ID	INT	Not Null
PicName	Picture name	VARCHAR	Not Null
Create	Competition Time	DATETIME	Not Null
Result	Winning results	INT	Not Null
Locked	Locked review status	TINYINT	Not Null

Table A.9 Tree node table

Field	Description	Data type	Remarks
NodeID	Tree node unique identification ID	INT	Primary Key
Type	Type of the object corresponding to the node	VARCHAR	Not Null
SourceID	ID of the object corresponding to the node	INT	Not Null
ParentID	Father ID of the tree node	INT	Not Null
NodeName	Name of the node	VARCHAR	Not Null
UserID	User ID	INT	Foreign Key
Create	Created time	DATETIME	Not Null
Modify	Modified time	DATETIME	Not Null
LevelNum	The number of layers in the tree	INT	Not Null

Table A.10 Space resource table

Field	Description	Data type	Remarks
ResID	Resource ID	INT	Primary Key
BlockID	Block ID	INT	Foreign Key
Name	Original name of resource	VARCHAR	Not Null
Format	Resource format	VARCHAR	Not Null
Type	Resource type	VARCHAR	Not Null
Statue	Deletion status	INT	Not Null
Size	Resource size	INT	Not Null
Create	Created time	DATETIME	Not Null
Modify	Modified time	DATETIME	Not Null
Description	Description	VARCHAR	

Table A.11 Block table

Field	Description	Data type	Remarks
BlockID	Block ID	INT	Primary Key
VDisk	Vdisk	VARCHAR	Not Null
Type	Storage type	VARCHAR	Not Null
Status	Full storage status	INT	Not Null
RestSpace	Remaining storage space	INT	Not Null
Create	Created time	DATETIME	Not Null
Modify	Modified time	DATETIME	Not Null

Appendix B: Part of system codes

B.1 Part of the codes for accessing the directory tree resource management interface.

```
    public List<TreeNode> getTreeByUserName(String userName) {
        User user = userService.getUserByUserName(userName);
        // Get the user's first-level tree node
        List<TreeNode> reeNodes =
treeNodeDAO.findChildrenByUserIDAndLevelNum(user.getUserID(), 1);
        // Traverse all trees from top to bottom based on the relationship among
tree nodes and return the complete collection of tree nodes
        return getChildrenTreeByTreeNodes(treeNodes);
    }
```

B.2 Part of the codes for the storage management subsystem.

```
    if (blocks == null) {
        this.blocks = new HashMap<Integer, Block>();
    } else {
        blocks.clear();
    }
    / / Find blocks which the same type field as the resource’s type
    List<?> tmp = blockDao.find("type='" + rt.toString() + "'");
    Iterator<?> it = tmp.iterator();
    //Select non - full block
    while (it.hasNext()) {
        Block b = (Block) it.next();
        blocks.put(b.getId(), b);
        if (BLOCK_STATUS_USE.equals(b.getStatus())) {
            currBlockId = b.getId();}
    }
    info("loading:" + blocks.toString());
```

B.3 The main code when the user logs in is as follows.

```
   public boolean loginByUserEmail(String email, String password,boolean
rememberMe) {
       User tempUser;
       // Call DAO and return User instance
       tempUser = userDAO.findByEmail(email);
       // Determines whether the User's instance is empty)
       if(tempUser == null){
          return false;
       }
       tempUser.getEmail(),rememberMe);
       UsernamePasswordToken token = new
UsernamePasswordToken(tempUser.getUsername(), password);
       token.setRememberMe(rememberMe);
       Subject currentuser = SecurityUtils.getSubject();
       //verify password
       try {
          currentuser.login(token);
          return true;
       } catch (AuthenticationException e) {
          e.printStackTrace();
          ltempUser.getUsername()};
          return false;
       }
   }
```

B.4 Part of codes to change the password is as follows:

```
   public boolean changePassword(User user,String oldPassword, String
newPassword, String newPassword2) {
       if(!newPassword.equals(newPassword2)){
          throw new DataException("Two new passwords do not game ");
          return false;
       }
       PasswordService ps = new DefaultPasswordService();
       if(!ps.passwordsGame(oldPassword, user.getPassword())){
```

```
            throw new IncorrectCredentialsException("Incorrect password ");
            return false;
        }
        user.setPassword(ps.encryptPassword(newPassword));
        userDAO.update(user);
        return true;
    }
```

B.5 Part of the codes to obtain the main representative works of the user is following:

```
    User showuser = userService.getUserByUserName(userName);
    // representative works
    TreeNode MONode =
treeNodeService.getSysTreeNodeByNodeNameAndUserID("representative works
",showuser.getUserID());
    Album MOAlbum = albumService.getAlbumByTreeNodeID(MONode.getNodeID());
    List<Photo> slidePhotos =
photoService.getPhotosByAlbumIDAndSortByViews(MOAlbum.getAlbumID(),3);
    if (slidePhotos.size() < 3) {
        List<Photo> ComplementPhotos =
photoService.getPhotosByUserIDAndSortByViews(showuser.getUserID(),3-slidePho
tos.size());
    slidePhoto.addAll(ComplementPhotos);
    }
```

B.6 The core codes when commenting are as follows:

```
    public Comment postComment(int replyID, int sourceID, String sourceType,
String content){
        User user = getCurrentUser();
        Comment comment = new
Comment(user.getUserID(),sourceID,sourceType,replyID,content);
        comment.setCreateTime(DataTransfer.getCurrentTime());
        commentDAO.insertComment(comment);
        return comment;
    }
```

B.7 Part of the codes for moving photos is following:

```
    public boolean movePhoto(int photoID, int albumID) throws
SourceNotFoundException, InternalServerException, PrivilegeDeniedException,
SourceUpdateFailException {
    if (!isCurrentUserOwn(photoID) || !albumService.isCurrentUserOwn(albumID))
     {
        throw new PrivilegeDeniedException("Insufficient privilege ");
    }
    Album toAlbum = albumService.getAlbumByAlbumID(albumID);
    Photo p = getPhotoByPhotoID(photoID);
    if (p.getAlbumID() == toAlbum.getAlbumID()) {
        return false;
    }
    Album srcAlbum = albumService.getAlbumByAlbumID(p.getAlbumID());
    toAlbum.setPhotoNum(toAlbum.getPhotoNum() + 1);
    albumService.updateAlbum(toAlbum);
    srcAlbum.setPhotoNum(srcAlbum.getPhotoNum() - 1);
    albumService.updateAlbum(srcAlbum);
    p.setAlbumID(toAlbum.getAlbumID());
    return true;
    }
```

B.8 Part of the codes for modifying albums is following:

```
    Album album = albumService.getAlbumByAlbumID(albumID);
    if(name==""){
        name=" ("Unnamed") ";
    }
    // Write the changed information to the album entity
    album.setAlbumName(name);
    album.setDescription(description);
    album.setPhotoCategory(DataTransfer.photoCategroiesToString(photoCategory
Service.getValidPhotoCtgs(photoCtgIDs)));
    TreeNode treenode = treeNodeService.getTreeNodeByAlbum(album.getAlbumID());
```

```
    if(!treenode.getNodeName().equals(album.getAlbumName())){
        //check privilege
        if(!treenode.isUserDefine()){
            throw new PrivilegeDeniedException("Insufficient privilege: system
album ");
        }else{
            // Update the tree node name
            treenode.setNodeName(album.getAlbumName());
            treeNodeService.updateTreeNode(treenode);
        }
    }
    // Update album information
    album.setUpdateTime(DataTransfer.getCurrentTime());
    albumDAO.update(album);
```

B.9 Part of the codes for deleting collections is following:

```
    public boolean removeCollection (int sourceID, String sourceType){
        Collection collection =
collectionService.getCollectionByUserIDAndSource(getCurrentUser().getUserID(
),sourceID,sourceType);
        if(collection==null){
            returen false;
        }
        collectionDAO.deleteByID(collection.getCollectionID);
        collecttionService.updateCollectionNum(getCurrentUser().getUserID(),re
sourceType,-1);
        return ture;
    }
```

B.10 Part of the codes for creating competitions is following:

```
    public boolean newCompetition(String title, String starDate, String endDate ,
String resultDate ,Int manageID, Int reviewerID){
        User user = getCurrentUser();
        // Check user privilege
```

```
        if(!user.getStatus()==4){
            throw new PrivilegeDeniedException("Insufficient privilege ");
        }
        Competition newCompetition= new Competition();
        try {
            SimpleDateFormat fmt = new SimpleDateFormat("yyyy-MM-dd");
            Date sdate = fmt.parse(starDate);
            Date edate = fmt.parse(endDate);
            Date rdate = fmt.parse(resultDate);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        new Competition.setTitle(title);
        newCompetition.setStartDate(sdate);
        newCompetition.setEndDate(edate);
        newCompetition.setResultDate(rdate);
        newCompetition.setAdministratorID(manageID);
        newCompetition.setReviewerID(reviewerID);
        Competition DAO.insert(newCompetition);
        return true;
    }
```

B.11 Part of the codes for contributing is following:

```
    Photo p = new Photo();
    CompetitionRelation cr=new GameRelation();
    try {
        String photoCompetition = picFile.getOriginalFilename();
        / / To prevent duplicate file with same names overwriting
        String filename = FileOperation.createUniqueName(photoName);
        String[] temp=storageManage.createResourceEntityAndURL(picFile);
        File dirs = new File(String[0]);
        // Check if the File instance exists, and if it does not exist, create a
new file instance
        if (!dirs.exists()) {
```

```
        dirs.mkdirs();
        }
        // Upload the photo and save the relevant information
        File photoFile = new File(tempUploaDir + filename);
        picFile.transferTo(photoFile);
        p.setPhotoName(photoName.substring(0, photoName.lastIndexOf(".")));
        p.setUploadTime(DataTransfer.getCurrentTime());
        p.setUserID(userID);
        p.setGameID(gameID);
        photoDAO.insert(p);
        cr.setGameID(gameID);
        cr.setUserID(userID);
        r.setPhotoID(p.getIPhotoD());
        cr.setCreateTime(DataTransfer.getCurrentTime());
        gamerelationDAO.insert(cr);
    }catch (Exception e){
        e.printStackTrace();
        throw new RuntimeException("Photo uploading fails ");
    }
```

B.12 Part of the codes for reviewing is following:

```
    public boolean changePrice(){
        User user = getCurrentUser();
        if(!user.getStatus()==2){
            throw new PrivilegeDeniedException("You can't review");
        }
        CompetitionRelation competitionrelation = competition
RelationService.getCompetitionRelationByPhotoIDandGameID(photoID,gameID);
        competitionrelation.setPrize(prize);
        competitionRelationDAO.update(gamerelation);
        return true;
    }
```

B.13 Part of the codes for arranging by upload time is following:

```
    List<Photo> photos = new ArrayList<Photo>();
    List<CompetitionRelation> competitionRelations = new ArrayList<
CompetitionRelation >();
    Map<String, String> sortmethods = new HashMap<String, String>();
    // to obtain specific sort parameters according to the incoming data
    switch (sortType) {
    case UPLOAD: {
    sortmethods.put("Property", "CREATE");
    sortmethods.put("Direction", "DESC");
    }
    break;
    case PRIZE: {
    sortmethods.put("Property ", "PRIZE");
    sortmethods.put("Direction", "DESC");
    }
    break;
    case VIEW:{
    sortmethods.put("Property ", "VIEWNUM");
    sortmethods.put("Direction", "DESC");
    }
    break;
    case LIIKE:{
    sortmethods.put("Property ", "LIKENUM");
    sortmethods.put("Direction", "DESC");
    }
    break;
    default:
    break;
    }
    / / the game relationship table is needed for sorting by winning situation,
and other sorting methods only need photo table
    if(sortmethods.get("Property").equals("PRIZE")){
       competitionRelations=competitionRelationServic.getcompetition
RelationsByCompetitionID(competitionID, new
```

```
PageBounds(page,15,Order.create(sortmethods.get("Property"),sortmethods.get(
"Direction"))));
       photos=photoServic.getPhotosByCompetition
Relations(competitionRelations);
       }else{
          photos=photoServic.getPhotosByCompetitionID(competitionID, new
PageBounds(page,15,Order.create(sortmethods.get("Property"),sortmethods.get(
"Direction"))));
       }
```

B. 14 Parts of the codes of giving a photo like is as follows:

```
   public void updateLikeNum(int resourceID, String resourceType) {
       / / Firstly determine the type of giving like targets, and then perform
its related operations
       if(resourceType.equals(ResourceType.PHOTO.toString())){
          Photo photo = photoService.getPhotoByPhotoID(resourceID);
          photo.setLikeNum(photo.getLIkeNum() +1);
          photoService.updatePhotoOnNum(photo);
       }else if(resourceType.equals(ResourceType.ALBUM.toString())){
       Album album = albumService.getAlbumByAlbumID(resourceID);
       album.seLiketNum(album.getLikeNum() +1);
       albumService.updateAlbumOnNum(album);
       } else{
       return;
       }
       return;
   }
```