Reflection and Critique

Group Members:
Shiyi Wang (581782); Channan Gu (692475)

Index

Design Reflection

This is the first project we spent more time on design than implement. We follow the steps to finish the design and drew URL diagrams one by one. In this case, the benefit of design more showed perfectly in verification and validation :

1. What functions need or can be implement to meet the requirement largely are clear in whole project.
2. The structure of system is clear for the programmer to implement, which means the conflicts between different functions related to system can be reduced to minimum.

To be specific, in the actual operation of the system, some parts of clients may use these kind systems as a network drive to save their images. To these group, what they really need is albums to classify their pictures rather than the image filter to do the edit. So we imported the concept of albums which also can consider as a folder. However, there are still some people need image vision controller to manage their pictures edited in webpage. In this case, version track is another way to manage pictures.

When we think of these functions, the difficulty and possibility to implement is also a significant point to consider with. For example, to reduce the conflicts with the series actions of images in folder, like delete, restore, move to, copy. We create four kinds of folder:

1. Main folder -- All images uploaded by this users will put in this folder.
2. Sharing folder -- All images shared to this users can be seen in this folder.
3. Trash folder -- Images deleted in the folder will put in this folder.
4. Individual folder -- Users (only) can create and delete this kind of folder, and add image which is a mirror this folder.

User can add images from main folder(Kind 1) and sharing folder(Kind 2) to the folder created by himself which is kind 4. In addition, the images seen in individual folders(Kind 4) are just mirrors in main and share, which means the deleting of these kind of images will have no influence on others which may be the same one in other folders. However, when a user destroys one picture in Trash(Kind 3), the copies in individual folder, which are just mirrors will disappear. What we should strongly mentioned is, using this architecture, the system will just stone one image in database no matter how many copies he add to his own folders. The benefits of this design can be seen in three aspects:

1. The concept can easily be understood by users, which means an easy accessing and using of this system.

2. The reduce of conflicts make programmer easy to build the tables of database and implement of system.
3. Since the database just save one data of all the same images. The space of database can reduce to minimum.

When cames to version control parts, we consider it as a totally different part of folder to manage the picture. It should be easy to use, to implement and also to maintain and extend. As we don't have much time on implement and technetic learning, we should minimise our work on that part. We regarded it as the bonus of our project, which also can have several stages to implement. But so far, we have no time to dealing with it.

Implement Reflection

Obviously, ruby on rails is an easy and powerful technology to implement web system. However, it is totally new for us. We spent much time on searching on google and learning on codeschool.

1. The order to implement

In this part for development, we started from database and model. The reason is, we want to finish the functional requirements first, which is the soul of ruby on rails and our system. Besides, dealing with html in views is a huge task taking time which, however, is the most precious for us in this stage of time. So in the very beginning, we planned to provide a simple view with our functions, and modify it later.
We settled down the structure of the database and relations between the models first, and regarded it as a base for us to develop our functions. Once we finished it, tasks can be divided to different people to implement different controllers and views.
After finishing the main functional part on controller, we began to work on the views, which route to the interfaces implemented in controller part.

2. The technique and concept we use
   - 2.1. Ruby on Rails

     Ruby is such a friendly language which is almost like the human languages, and rails is extremely powerful. It is much easier than other language and frameworks like php on Laravel.
   - 2.2. MVC
   - 2.3. FORM Architecture

     This is actually new for us. However, when we were using it, we found the structure of files and codes became clear due to it, and routing methods become easier too.
   - 2.4. GEMs
     - 2.4.1. devise: User detail and login management
     - 2.4.2. RMagick: Do the filter function in server part
     - 2.4.3. carrierwave: manage the submission of image files

3. The difficulties in implement

   The biggest difficulty for us is UI. Both of us are not good at html and also css, etc, which need experience in web design. We didn't use templates, but trying to write by ourselves. Fortunately, the results are not bad. Still, it is a tough thing for us.

Critique & future work

Until now, we finished all functions signed as “Priority High” in our schedule. In fact, these is just a quarter in our blueprint. We designed a complete series of functions for users and also the methods to implement. What we feel mostly regretful is the version control part. We give it two stages to implement the final version on version functions.

1. Give every picture a version to track in the version control tree. The origin picture is the root of version tree, and the pictures edited basing on the pre-version can be the nodes of that picture. In this stage, users have a thick view part. Once he finished editing the picture, he can upload it to web server, and makes it as a subversion of origin picture. In the maintime, database should save the new image data and details for every version, but it needn’t to save the actions of users on the picture.
2. In Stage II, we also use a thick client architecture. Comparing with thin client architecture, users don’t need to spend time on waiting for reaction from the server to dealing with the image editing and refresh the new images. However, server need to track every actions with filters on image. The records of actions can replace the data of total image saved in database in Stage I. Whenever the user call or refresh the image, the server can create the new image using the root image which has the origin image details and the actions of all previous versions. The benefit is obvious:
    2.1. Largely minimise the data to save in the web server, which we considering as the most vital part for a image saving and control system. Since if it facing to the public, millions of images can be submitted up to the server database which means innumerous data. Thus, consider the ability of the database should be necessary in our system.
    2.2. In an actual server machine, doing the algorithmic can be much faster than index and read data from hard disk. Using this method can not only reduce the space of saving, also can improve the performance of the system.

There are also limitations in the functionality in our web application because if we compare this to other social sites that exist on the internet such as Facebook or twitter. Our web application looks too pale and dull. The reasons which we couldn’t finish it totally or satisfy us are mainly in two aspects below:

1. The limit of time, people.
   Time can be the biggest enemy in the end of semester. If we have plenty of time and human resource, we can translate our idea fully to reality which can be a wonderful website.
2. The limit of technique and experience.
   Still, we are just the beginners on ruby and rails. To develop a system not only needs knowledge, the experience should occupied a huge percentage in whole stages. We need time to search the GEMs and descriptions, to understand the concept of new tools. Beside, using working on different parts on MVC is also an annoying task.

What we want to do more:

1. Beautify the views and create well looking and friendly UI for users. In this version, we spent most of time on implement the functions. As UI can be a totally different subject for our system, we need time to learn UI design and the way to use html, css, json etc.

2. Optimize the server part of system and complete rest of functions in low priority, such as the version control and data compression. We want to develop a system which meets functional requirements with high performance and also reliability.
3. Testing. Testing is a significant stage with large workloads, which is pretty similar with UI modification. If it possible, we can use V model which are unit testing, components testing, system testing, maintenance testing and acceptance test corresponding to previous stages.