CIS 550
Course Project: Pennterest
Final Report

Yunkai Cui, Di Wu, Tianyi Zhang, Bofei Wang
2013/12/15
1. Introduction and project goals

1.1 Project Introduction:

The goal in this project is to develop a “pinning” site, along the lines of Pinterest, Digg, Reddit, Slashdot, etc. Such sites present each user with a custom login screen containing
(1) Items they like and have “pinned”
(2) Items that are popular among everyone or among friends
(3) Platform should support sharing of hyperlinks to any kind of external content.

The project will exercise the following aspects: schema design (relational, XML) and view-based access control; cloud hosting; XML interchange; REST and JSON data interchange; SQL-based recommendation; caching in a key-value store; performance evaluation.

1.2 Pinterest website Introduction

Pinterest is a pinboard-style photo-sharing website that allows users to create and manage theme-based image collections such as events, interests, and hobbies. Users can browse other pinboards for images, "re-pin" images to their own pinboards, or "like" photos.

In Pinterest, users can upload, save, sort and manage images, known as pins, and other media content (e.g. videos) through collections known as pinboards. Pinterest acts as a personalized media platform, whereby users’ content and the content of others can be browsed on the main page. Users can then save individual pins to one of their own boards using the "Pin It" button, with Pinboards typically organized by a central topic or theme. Content can also be found outside of Pinterest and similarly uploaded to a board via the "Pin It" button which can be downloaded to the bookmark bar on a web browser, or be implemented by a webmaster directly on the website.

On the main Pinterest page, a "pin feed" appears, displaying the chronological activity from the Pinterest boards that a user follows. When browsing for new boards and relevant pins, users can visit a "Tastemakers" page that recommend pinboards with content similar to previous pins saved by a user. For both guests and Pinterest users, there are currently four main sections to browse:
everything, videos, popular, and gifts.

Quick links to Pinterest include the "pin it" button that can be added to the bookmark bar of a web browser, "Follow me" and "Pin it" buttons added to personal website or blog page, and the Pinterest iPhone application available through the App Store.

A "board" is where the user's pins are located; users can have several boards for various items. A "pin" is an image that has either been uploaded or linked from a website. Once users create boards and add pins, other users can now "repin", meaning that they can "pin" one user's image to their board as well. Once the user has set up their account and boards, they can browse, comment, and like other "pins". If a user wants to turn an image online into a "pin," there are a few simple steps to do so. First, the user must select the image to pin. Second, the user then clicks on the "Add +" button on Pinterest. Third, the user enters an image URL into the link box. Next, the user will select the exact image they wish to pin and place it on the designated board. Users can then describe the pin and share it via Twitter or Facebook. Other users can now click on the pin to see which board the image is pinned in, who pinned the image previously, where the original pin is from, and who has liked, commented, or repinned the image.

1.3 Project Goals
1. Use a version control system like subversion to save and share source code across your team.
2. Provide import and export capabilities to a standard XML interchange format shared by all projects. You will import data from multiple groups.
3. Implement user profiles and sharing groups.
4. Implement recommendation and search features.
5. Implement caching using a key/value store like Amazon SimpleDB, S3, or DynamoDB, or MongoDB.
6. Forward search results to Yahoo via a REST Web services API.

2. Basic architecture

2.1 Schema Design

- pinsite (operator string, web_url string)
- Users (login string, email string, givenname string, surname string, affiliation string, birthday data, password string);
- Board (userID int, boardName string, private int, description string);
- Object (objectID int, sourceID string, url string, counts int);
- Pin (objectID int, sourceID string, userID string, boardName string);
- Rating(objectID int, sourceID string, userID string, rating int);
- Tag(objectID int, sourceID string, tag string)
- Friend(userID string, following string)
- ObjectClass(objectID int, sourceID string, type string)
- Interests(userID string, interest string)

2.2 Web Basic Structure

Our website design uses Model View Control paradigm. The model part includes:

<table>
<thead>
<tr>
<th>Module</th>
<th>Class</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board</td>
<td>CreateNewBoard.js</td>
<td>Create New Board in database</td>
</tr>
<tr>
<td>Object</td>
<td>CreateNewObjects.js</td>
<td>Create New Object in database</td>
</tr>
<tr>
<td>Friend</td>
<td>FriendAdd.js</td>
<td>Add a friend in database</td>
</tr>
<tr>
<td></td>
<td>FriendRead.js</td>
<td>Read friend profile from database</td>
</tr>
<tr>
<td>Personal</td>
<td>IndividualCheck.js</td>
<td>Change the pictures from MongoDB URLs, then load the personal page</td>
</tr>
<tr>
<td>LoginCheck</td>
<td>LoginCheckForNoPass.js</td>
<td>Update the password for user who does not have one</td>
</tr>
<tr>
<td></td>
<td>LoginCheck.js</td>
<td>Check the login information</td>
</tr>
<tr>
<td>Pin</td>
<td>Pin.js</td>
<td>Insert a pin relationship in database</td>
</tr>
<tr>
<td></td>
<td>UnPin.js</td>
<td>Delete a pin relationship in database</td>
</tr>
<tr>
<td>Rate</td>
<td>RateAdd.js</td>
<td>Add a rating in database</td>
</tr>
<tr>
<td></td>
<td>RateRead.js</td>
<td>Load the rating values from database</td>
</tr>
<tr>
<td></td>
<td>RateUpdate.js</td>
<td>Update the rating values from database</td>
</tr>
</tbody>
</table>
### 3. Key technical challenges and solution

#### 3.1 Recommendation:

We developed a metric to recommend the pictures based on the users’ interests and the average rating of each picture. We will recommend 20 pictures with the highest average ratings that are rated by other users. And we give the user pictures with tags that overlap with the users interests. Also we will give the pictures got from Bing search with the key words that are users’ interests. The challenge is: we have to calculate the average rating of each picture and return the pictures with tags that overlap the user’s interests, which makes the query so slow. So we added indexed on the objectID, sourceID, tags and interests of table Rating, Tag and Interest which makes the query much faster.

#### 3.2 Caching:
In the project, we used a NoSQL database (MongoDB) to implement caching. Any objects that are referenced or pinned by more than 5 users are copied into a database, and retrieved directly from there.

Since we chose to use Node.js and it is an asynchronous. The challenge is: Node.js is event-driven, asynchronous and does not have return values. Originally, I was planning to fetch the image data from MongoDB and storing it locally on our server. We then used the local URL to directly display the local pictures. However, since the characteristic of asynchronous, we cannot control the process of fetching or loading images and thus we do not know if they are sent to the local. The solution that we came up with to deal with the problem is when we get the query result, we classify the data into two groups based on the counts number and change the group whose counts are larger than 5 into new URL with their keys in it. When the router parse the address, it will automatically call the MongoRead function to get the image based on its key (i.e, its id) and get the picture from MongoDB and send the picture to the front end.

3.3 Search:
   1. The basic search: the search on arbitrary number of key words
Here we want to achieve the search function whose input search string contain arbitrary number of search key words which are separated by space or commas. Since the number is arbitrary, it cannot be simply achieved by the UNION. Here we use function nesting to solve it. The main challenge is how to merge the results together while the node.js call is asynchronous. Finally we use the iteration to add one result to the end of the other.

2. The Bing search(EXTRA CREDIT):
The challenges are that at first we do not have any idea of how the bing search works or the idea of AJAX. Besides, the online quick start is about php instead of node.js. Much of the examples online did not help since the new version of Bing search API has changed a lot than before. Fortunately we found brief example online and learn to modify the parameter to improve the performance. [http://stackoverflow.com/questions/20309448/embed-bing-search-into-nodejs](http://stackoverflow.com/questions/20309448/embed-bing-search-into-nodejs).

Simply speaking, a search request can be made via the URL directly. What we need to do is to sign up an account and change some parameters:

**query URL:**

$format: Specifies the format of the response (XML or JSON). Here we choose JSON

$top: Specifies the number of results to return.

$skip: Specifies the offset requested for the starting point of results returned.

Query: the only parameter that is essential.

The service operations (the end of the query URL): Specifies the data source of that is targeted. The options are: Web, Image, Video, News, SpellingSuggestion, RelatedSearch and Composite. Here we choose Image.

**The returned information:**
The results returned contained lots of information. Here we choose MediaUrl to display the images, and Title,SourceUrl to show some description.

3.4 Ajax vs redirect:
When do Ajax through jQuery to submit a form, in our case is to submit the login form, browser will return the result to the Ajax callback and ignore the redirect request from server. At first we want to use Ajax techniques to do some fancy stuff on the login page, but we failed to redirect the user to the main page, so we gave up at last.

But in a lot of places where we don't need to redirect user, for instance when pin a picture, we want to read all the boards of a user to let him/her choose where he/she wants to pin, we succeeded to use lots of Ajax tricks.

3.5 Get vs Post:
We treated all the information related to user as the information which needs to be secured. So for submitting and querying all that type of information, we used post method. But for other usage like redirecting people to login page or showing the cached pictures, we used get method.

3.6 Password Security:
We believe that protecting user's information, especially the password, is extremely important. So we decided to encrypt (technically hash) user password before sending them to the server. So I applied SHA1 algorithm to do the job. However for the consideration of better user experience,
we don't want our user to see the length of their password has been changed. To do that, we create a hidden input tag where we put into the encrypted password, and when user submits the form, we will copy the first n character of the encrypted password to replace the original one, where n is equal to the length of original password.

3.7 Front-end debug:
We really had a hard time to debug the front-end JavaScript program. But thanks to the alert function that JavaScript provided and Chrome developer tools, we finally found a much easier way to do it. Chrome developer tools are really useful to check broken JavaScript code and CSS style sheet problems, while alert windows will let us know what is actually going on in the code.

3.8 Session and cookie
How to check the login user, the asynchronized character of node.js won't allow us to set any username to the variables. Therefore we used session and cookie concepts in our website to store user related information in the session, and send them back to the client side to store them in the cookie after encrypted. So every time when we need to authenticate user's information or executing user related queries, we will read this information. It's really convenient and safe.

3.9 UI design
We found a lot of good libraries that help us to finish our UI design, such as jQuery UI, Chosen, Harvest, jQuery Rateit etc, which saved us a lot of time.

4. Performance evaluation
In this section, we instrument the application to collect timings on various aspects. We recorded and measured several latency in handling some requests.

4.1 Latency in Handling Events:
The log data is as follows:

**GET / 200 779ms - 3.18kb**
**GET /stylesheets/index.css 200 9ms - 291b**
**GET /sha1-min.js 200 22ms - 2.86kb**
**GET /stylesheets/signin.css 200 2ms - 1.87kb**
**GET /UI/Imagenation.png 200 5ms - 24.57kb**
**GET /UI/InputBox.png 200 3ms - 2.95kb**
**GET /UI/ButtonTransparent.png 200 12ms - 2.87kb**
**GET /UI/SignIn.png 200 5ms - 4.76kb**
**GET /UI/SignUp.png 200 3ms - 4.77kb**
**GET /UI/blurback.jpg 200 554ms - 257.66kb**
**GET / 200 85ms - 3.18kb**
**GET /stylesheets/index.css 200 2ms - 291b**
**GET /sha1-min.js 200 1ms - 2.86kb**
Generally, fetching pictures from MongoDB is slower than directly fetching from its original URL. We also list some picture size to illustrate the relationship between latency and picture size:
<table>
<thead>
<tr>
<th>FileID</th>
<th>108</th>
<th>109</th>
<th>111</th>
<th>113</th>
<th>114</th>
<th>115</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency(ms)</td>
<td>3582</td>
<td>3967</td>
<td>1492</td>
<td>2050</td>
<td>2295</td>
<td>2500</td>
</tr>
<tr>
<td>File size(KB)</td>
<td>690</td>
<td>1126</td>
<td>49</td>
<td>115</td>
<td>176</td>
<td>295</td>
</tr>
</tbody>
</table>

We can see some the relationship between Latency and file size has some function form:

\[
\text{Latency} = \sqrt{\text{size}}
\]

### 4.3 (EXTRA CREDIT) Multiple concurrent requests

We did experiments in which three people using same username to log in and pin at the same time. The result is always someone can pin the picture successfully and someone failed to pin. Also, it will show error message:

```
[Error: ORA-00001: unique constraint (CIS550PROJECT.PK_PHOTO) violated]
[Error: ORA-00001: unique constraint (CIS550PROJECT.PK_PHOTO) violated]
```

The reason is obvious: the pined objects will have the same assigned ID. When they are inserted into database, it will violate the constraint.

The server log is attached:

```
GET /basic_search?name=yell&searchbutton=Search 200 2002ms - 19.2kb
GET /stylesheets/mainstyle.css 304 2ms
GET /stylesheets/pennstyle.css 304 2ms
GET /UI/chosen/chosen.css 304 2ms
GET /UI/chosen/docsupport/prism.css 304 2ms
GET /UI/chosen/docsupport/style.css 304 2ms
GET /UI/css/smoothness/jquery-ui-1.10.3.custom.css 304 0ms
GET /rate/rateit.css 304 2ms
```
4.4 Search

1) Search photos based on tags
   We combine the basic search and Bing search together. The left part of the screen displays the image of basic search and the right part displays the Bing search.
   For the arbitrary number of key word search:
   First, the results of basic search and Bing search seem to interpret in different ways. For example, if I type in 'Apple tree'(suppose our database has such tags, which actually not), the basic search could return the picture of apples and the picture of trees, while the results of the Bing search will return the pictures of the trees with apples on it.
   Second, we failed at the instance where the one tag are composed of multiple words, since we use the ‘exactly equals’ instead of ‘partially equals’.

2) Search loginID based on names.
   Here we use the partially equals while considering the number of input words, in case that we only remember part of the names. If input one word, we find the id of the people that either their first name or the surname includes the string. If input two words, we find the id of the people whose first name contains the first string while the surname contains the second string.
5. Potential future extensions

Search
Currently, the search algorithm has a lot space to improve. For example, how to make the search results invariant to the upper case or the lower case of key words, how to deal with the situation that the tags are made of multiple words, how to deal with the typo of key words, etc.

6. Some screen shot