**Back end**

**1. Info on the Back end**

**[Restful web services]**

After setting up the database RESTful web service from Database was used to import the necessary data from the database. This web service will automatically produce setters and getters for every data in the table row including all the necessary queries ready made for a letter use. Unlike RESTful web service form patterns this web service will also give FacadeREST.jave files for all the imported entities. This Facade has readymade get and post annotations to work with the front end. With restful web services, we made ‘User registration & login’, ‘Search images by tag & adding the tag to the image’ and ‘Like & dislike’.

**>> User Registration & Login**

This Image sharing application is not public, so users have to register first and then log in with the password and username they have registered with. We implemented ‘user registration’ using ‘create’ method, which has ‘Post’ annotation in UserFacadeREST.java. To check whether user name and password in database, we made ‘LoginforUsers’ method which has type ‘Get’ in the same java class. If username and password in the database, it returns ‘Yes’ (which means). If it is not, it returns ‘No’ to ‘signinSignUp.js’. The signinSignUp.js is Ajax and it gets response from the LoginforUser link and if it is a right user and password, it opens thumbnailGallery.html, which is main gallery page. If it is not, it displays the wrong password or ID in login modal to show the status of login to a user by signInSignUp.js.

When users login, we store the username in the localStorage to pass username to the other page like XMLParser.js. The username is used to show ‘welcome message with username’ in the navigation bar and for rating system later on. Later, when users log out, we remove the username from localStorage.

**>> Search by tag & Adding tag**

After logging in, users can see all the images uploaded by other users in thumbnailGallery.html. Users can search these images, which were tagged by a tag name. In TagFacadeREST.java, we implemented ‘SearchByTag’ for searching images by a tag. Tag list added to Image ArrayList, so that images which have a tag can start to be searched by a tag.

Users can also add a tag to a certain image. Images table and Tag table has ‘many to many relation’, so we need to use ‘getImageCollection’ from tag object and ‘getTagCollection’ from image object in attachTagToimg method. In this method, it checks whether image has a duplicated tag or not, so that the image can have multiple tags and tag can have multiple images too without error. This will help users to search a certain image tagged by that tag name.

**>> Rating system (Like / Dislike)**

In ImageFacadeREST.java, there is ‘rateImage’ method,which gets imgId, rate, username to add the new rate to the image. And it also checks whether the rater and image ID are in the database and if the combination of this two attributes are already in database, it start to check whether click the like button (which as a value ‘1’) or dislike button(which shas value ‘-1’) and return ‘You liked/disliked this picture already.’ With countLikes and countDislikes, it checked the image ID to match with the image and add the new likes/dislike to the original points of them and it returns final integer value to XMLParser.js. It has Ajax which gets the response from rateImage and it replies to user ‘You just liked/dislike this picture.’ with an alert. For passing the image ID without user’s notice in the front end, in the XMLParser.js we need to use ‘input type=”hidden” and we can pass the image ID to rateImage method. And for the username, we used the user name, which was stored in localStorage before to match with the logged in user.

**[Servlet]**

**>>Image Upload in upload.java**

We used one servlet for uploading a file with image metadata such as image title, image description, upload time, path and image category. Its main purpose is uploading image that will be given by a registered user in index.html, after login, to the images table in the database.

**[AJAX and Parsing XML]**

We tried to get the JSON from glassfish by using Jersey-bundle-1.9.1.jar, Jersey-server.1.9.1.jar, Jersey-container-server.jar. But returning JSON from glassfish was not stable. Sometimes it returns JSON but sometimes it returns only XML. In the last moment of the project, we decide to go with XML to display images when users browser or search images. We parsed XML with Ajax in XMLParser.js and tagImgModel.js to display images in the thumbnailGallery.html. For displaying categories in the upload modal, we used Ajax to parse the Allcategory get link and we could send the category ID and category name from database to frontend, so that it will be updated automatically when the category table gets changed.

We had one difficult challenge of Ajax. When we need to get the value of countLikes and countDislike methods, Ajax is asynchronous, so it already moves to next steps before it returning the value. So we got the undefined value of likes and dislikes at the first time. But later on, we use ‘.done’ after the 2 methods, so we can get the certain value of like/dislike before it moves to next steps.

**[Some features, which were implemented in the backend only]**

**>> Deleting user & finding user by user primary key**

This web app also have the functionality to search users with their primary key which is given to every user by the time they register to this application automatically by the database. Finally after using this app if users by any reason decided to stop using this web application they can just type in their primary key and delete all their information from the database.

**Conclusion:**

In making this image sharing web application, it has been lots of new things learned. All the way from the backend to the front end, it has been totally a huge practice for all of us. It has helped us to understand how the server and databases are operating and how to interact between the frontend and backend. And also in the beginning of semester, we have learned about making a web application that uses a database and that has a host server in virtual machine. This has helped us all to understand how server and backend works. Additionally, we could improve our java EE knowledge and also we get to practice java language, while we were using servlet and restful web service. Learning and utilizing Javascript, Jquery, Ajax helped us to implement making better appearance of the web application. And it also has been used to communicate the back end and the front end. Overall, we could draw a big picture of how to make the web application from frontend to backend, including Database. This semester has made all of us grown day by day.