**CMPUT 391 Project Report**

Tiancheng Shen

Hong Wang

Di Zhang

**Introduction:**

This project is a three-tiered web-based Ocean Observation System. The three tiers are the database server, the web server and the client. These components run on different machines and are connected via the Internet. JSP files are used to build the interface between the website and the database server. Oracle SQL is utilized for the database server, and Tomcat is utilized for the web server. Our clients currently have the authority to access this system through Google-Chrome on Linux systems.

**Modules:**

**Login Module:**

All users can use Login Module. It enables users to log in the Ocean Observation System with proper privileges. They can modify their personal information or password. It also set session so that other pages can use session to detect if user has privileges to access the page.

login.html:

This is the start page of our Ocean Observation System. User can input username and password at this page. It will also detect error if user input nothing and click “Log In”.

login.jsp:

This is the page after user login. Although it shows “Loading”, it actually doesn't load anything, because we just need a page to detect the username and password error. If user input wrong username or password, it will show the error and go to login.html. If user input correct username and password, it will show welcome and go to the homepage of the user's role automatically. It also set session so that other pages can use session to detect if user has privileges to access the page.

account.jsp:

This page shows user's personal information. It uses “String UserID = (String) session.getAttribute("userid")” to get user's Person Id, and uses SQL statement “select \* from persons” to get other detailed information of user.

It also provide link for modifying personal information (“Modify Personal Information” button), changing password (“Change Password” button), entering user's homepage (“Go to HomePage” button), and logging out (“Log Out” button).

editPerson.jsp:

This page shows up when user click “Modify Personal Information” button at account.jsp page. It enables users to edit his information expect Person ID. Person ID is the primary key, so only administrators can change it. It uses "select \* from persons” statement to get detailed information and shows them at text blank, so user can edit it easier. After user modifying it, clicking “Save” button to submit or user can just cancel the modifying by clicking “Cancel” button.

editPersonProcess.jsp:

After user click “Save” button at editPerson.jsp page, the editPersonProcess.jsp page will get new information and use “update persons set ...” statement to update the information. Then it will go to account.jsp page automatically.

password.jsp:

This page shows up when user click “Change Password” button at account.jsp page. It lets user input new password twice. It will detect error if user inputs nothing or these two inputs are not matched. After user inputs correctly, clicking “Save” button to submit or user can just cancel the modifying by clicking “Cancel” button.

passwordProcess.jsp:

After user click “Save” button at password.jsp page, the passwordProcess.jsp page will get the new password and use "update users set password" statement to update the password. Then it will go to account.jsp page automatically.

logout.jsp:

This page shows up when user click “Log Out” button at account.jsp page. It will update user login status to session, so user cannot access any pages except login.html page.

**Sensor and User Management Module:**

<all files involving sensor and user management module>

**Subscribe Module:**

subscribe.jsp:

This page displays a table of all existing sensors and the sensor\_ids that the logged in scientist subscribed to. The SQL statement to select all sensors is “SELECT \* FROM sensors”, then “while(rset.next())” will output all sensor information in the database. The scientist's person\_id is retrieved with “(String)session.getAttribute(“userid”);”, and “userid” is set in login.jsp. Now to extract the sensor\_ids this scientist subscribed to, person\_id can be used to get this information from the subscriptions table, the SQL statement “SELECT sensor\_id FROM subscriptions WHERE person\_id=”+person\_id; and “while(rset.next())” will output all sensor\_ids this scientist subscribed to.

To add or remove a sensor, input the sensor\_id the scientist wishes to add or remove and select the appropriate radio button. Once the “submit” button is clicked, the scientist will be directed to alter\_subscribe.jsp to find out whether the sensor is added or removed.

alter\_subscribe.jsp:

This page displays whether the scientist has successfully added or removed a subscription to a sensor. Unsuccessful attempts to add/remove a sensor subscription include: sensor does not exist, sensor already subscribed when adding subscription, sensor not subscribed when trying to remove. The SQL queries for alter\_subscribe.jsp file is the same as the queries for subscribe.jsp. The scientist can choose to return to subscribe page or account page.

**Upload Module:**

upload.jsp:

This page displays the types of data the data curator can upload. Upon choosing a data type and hit “Add”, the data curator will be directed to uploadData.jsp to upload data.

uploadData.jsp:

This page displays an html “multiart/form-data” form for file uploading. The data curator uploads a file that matches the file type chosen in upload.jsp. Scalar data accepts .csv files, image accepts .jpg files and audio accepts .wav files. Each form can only upload one file at a time. If a .csv file is uploaded, the data curator will be directed to uploadScalar.jsp. If a .jpg file is uploaded, the user will be directed to UploadImage class. If a .wav file is uploaded, the user will be directed to UploadAudio class.

uploadScalar.jsp:

This page displays whether the .csv file is read and the data is inserted into the database. If file input is empty, the user will be directed to uploadFinish.jsp and receive file empty message. If a correct file is uploaded, each line is split into three parts and inserted into the database using the query “INSERT INTO scalar\_data VALUES(“+id+”,”+sensor\_id+”,to\_date(“+date+”,'dd/mm/yyyy hh24:mi:ss'),”+value+”)”;

UploadImage:

This class inserts an uploaded image and its thumbnail to the database. This class processes the uploaded image and creates a thumbnail by shrinking the original image. With the query “SELECT SEQ\_IMAGE\_ID.NEXTVAL FROM dual”, a unique image\_id is generated and this id corresponds to the uploaded image. After the id is generated, it is used to create a row in images table with default information with the exception of image\_id. “INSERT INTO images VALUES(“+image\_id+”3333,SYSDATE','testdesc',empty\_blob(),empty\_blob())” is the SQL query. Then, the class updates the empty\_blob() to its corresponding image with “SELECT \* FROM images WHERE image\_id=”+image\_id+” FOR UPDATE”. The image and thumbnail are converted into type blob and updated in the row with corresponding image\_id. Upon completion, the data curator is redirected to uploadImage.jsp.

uploadImage.jsp:

This page displays all image sensors and let the data curator upload image information. The SQL query to display all image sensors is “SELECT \* FROM sensors WHERE sensor\_type='i'”. The data curator then enters the sensor\_id of the image, date, time and description. When the upload button is clicked, the information is updated in the database using the SQL statement "update images set date\_created=TO\_DATE('"+datetime+"','dd/mm/yyyy hh24:mi:ss'),sensor\_id="+sensor\_id+",description='"+description+"' where image\_id="+image\_id). Upon completion, the user knows that the file is uploaded.

UploadAudio

This class uploads an audio file to the database. A unique recording\_id is generated with the SQL query “SELECT recording\_id.nextval FROM dual”. The uploaded file is processed and the binary blob file is inserted into a row with its corresponding recording\_id and default information in the audio\_recordings table. Upon completion, the data curator is redirected to uploadAudio.jsp.

uploadAudio.jsp:

This page displays all audio sensors and let the data curator upload audio information. The SQL query to display all audio sensors is “SELECT \* FROM sensors WHERE sensor\_type='a'”. The data curator then enters the sensor\_id of the audio, date, time, length, and description. When the upload button clicked, the information is updated in the database using the SQL statement "update audio\_recordings set date\_created=TO\_DATE('"+datetime+"','dd/mm/yyyy hh24:mi:ss'),sensor\_id="+sensor\_id+",description='"+description+"',length="+length+" where recording\_id="+recording\_id). Upon completion, the user knows that the file is uploaded.

UploadFinish.jsp:

This page displays a message that file input for upload module is empty.

**Search Module:**

<all files involving search module>

**Data Analysis Module:**

Only scientists can access this module. It enables scientists to view the average, minimum, and maximum values of the selected sensor on five levels of time hierarchies.

dataAnalysis.jsp:

This page shows up when user click “Data Analysis” button at scientist.jsp page. Firstly, it uses SQL statement "select \* from sensors" to show all sensors information and "select sensor\_id from subscriptions” to show the sensors user has subscribed. Since user can only choose the subscribed sensors, it also provide a link (“Go To Subscribe” button) to subscribe.jsp page so that user can edit his subscribed sensors. After that, user can select which sensor he want to view and the time hierarchies, and then click “Submit” button.

analysisResult.jsp:

After user click “Submit” button at dataAnalysis.jsp page, the analysisResult.jsp page will get the sensor id in “Sensor” and time hierarchies. Firstly, it uses SQL statement “select \* from sensors” to show the detailed information of the selected sensor. Then it uses SQL statement "SELECT to\_char(date\_created, '(time format)'), avg(sd.value), max(sd.value), min(sd.value) FROM scalar\_data sd WHERE sd.sensor\_id = '"+Sensor+"' GROUP BY to\_char(date\_created, '(time format)') ORDER BY to\_char(date\_created, '(time format)')" to show the average, minimum, and maximum values according to the time hierarchies. The “(time format)” can be “yyyy”, “yyyy-q”, “yyyy-mm”, “yyyy-ww”, or “yyyy-mm-dd”, depending on which time hierarchy user chooses. After viewing the scalar values of the sensor, user can click “Back” button to go back to dataAnalysis.jsp page to select the sensor and/or time hierarchies again.