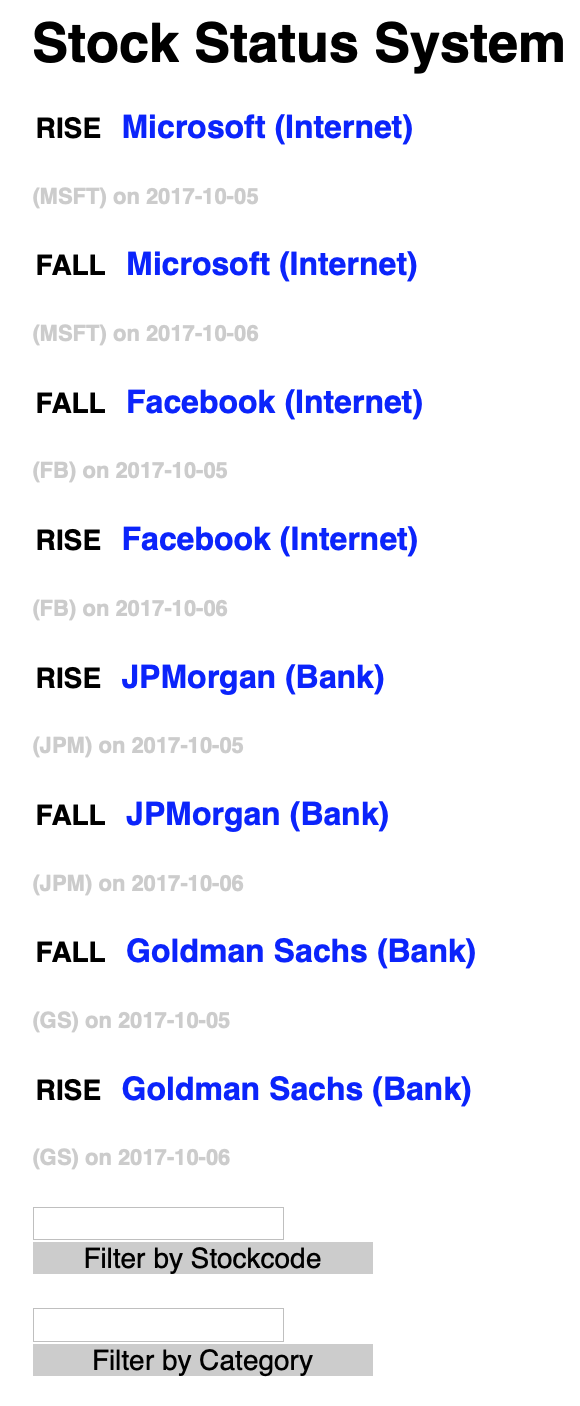
**COMP3322B MODERN TECHNOLOGIES ON WORLD WIDE WEB**

**Lab 3 PHP, JavaScript and AJAX**

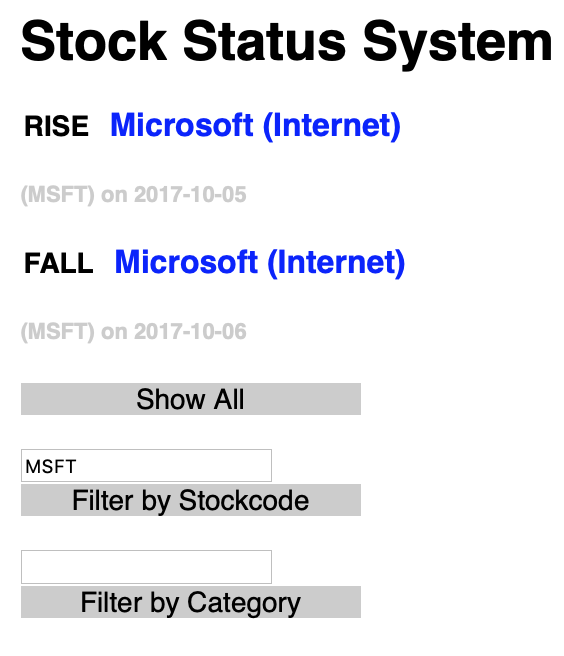
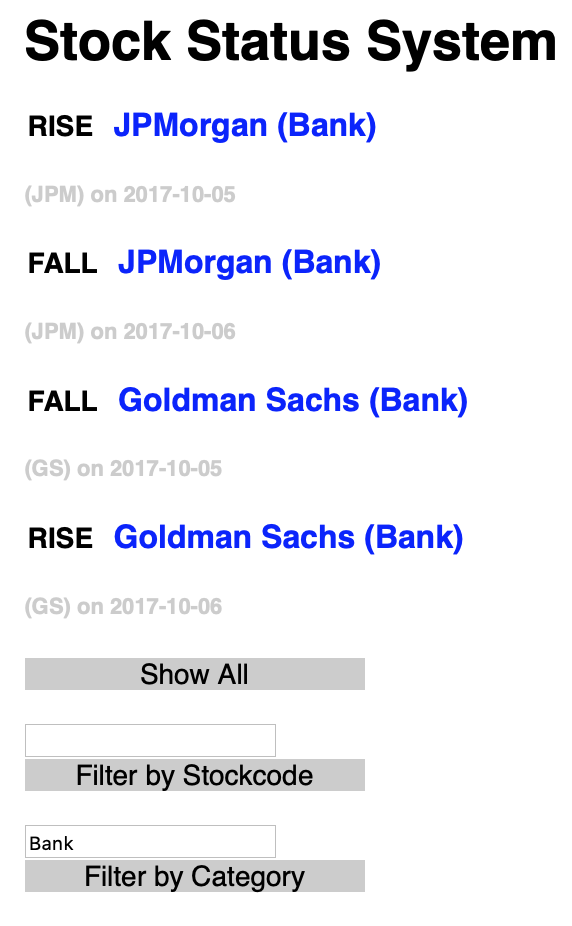
**Overview**

In this lab, we will develop a simple web-based stock status system, with which we can maintain a stock list. As shown in Fig. 1, each stock entry includes the status (RISE or FALL), stock name, stock category, stock code and trading date. We will practice loading the list from the database and toggling between the RISE status and the FALL status. We will also implement a few buttons to decide the (selected) entries to be displayed. For example, when “MSFT” is typed into the textbox above “Filter by Stockcode” button, the page shows up as in Fig. 2; when “Bank” is typed into the textbox above “Filter by Category” button, the page shows up as in Fig. 3. Note that a “Show All” button is displayed in Fig. 2 and Fig. 3, and when it is clicked, the page view goes back to Fig. 1.

These functionalities will be implemented through server-side implementation (PHP) and the client-side implementation (HTML, JavaScript, AJAX). We adopt MySQL as the back-end database server to store the status information.



**Fig. 1**

** **

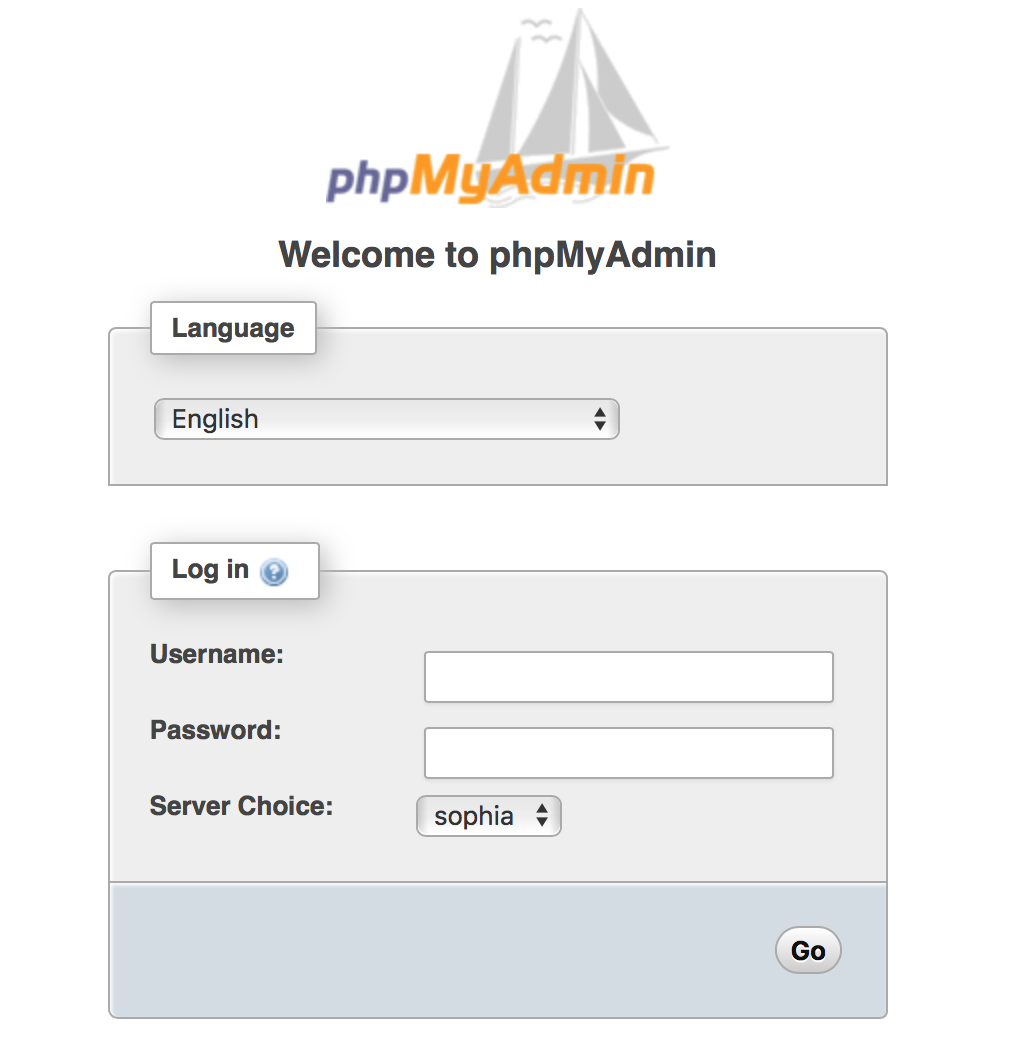
**Fig. 2 Fig. 3**

**Lab Exercise 1 – Prepare the database**

**Step1:** Get your MySQL account (if you do not have one) at <https://intranet.cs.hku.hk/common/mysqlacct/register.php>. It takes up to **one** working day for CS technical office to activate your account.

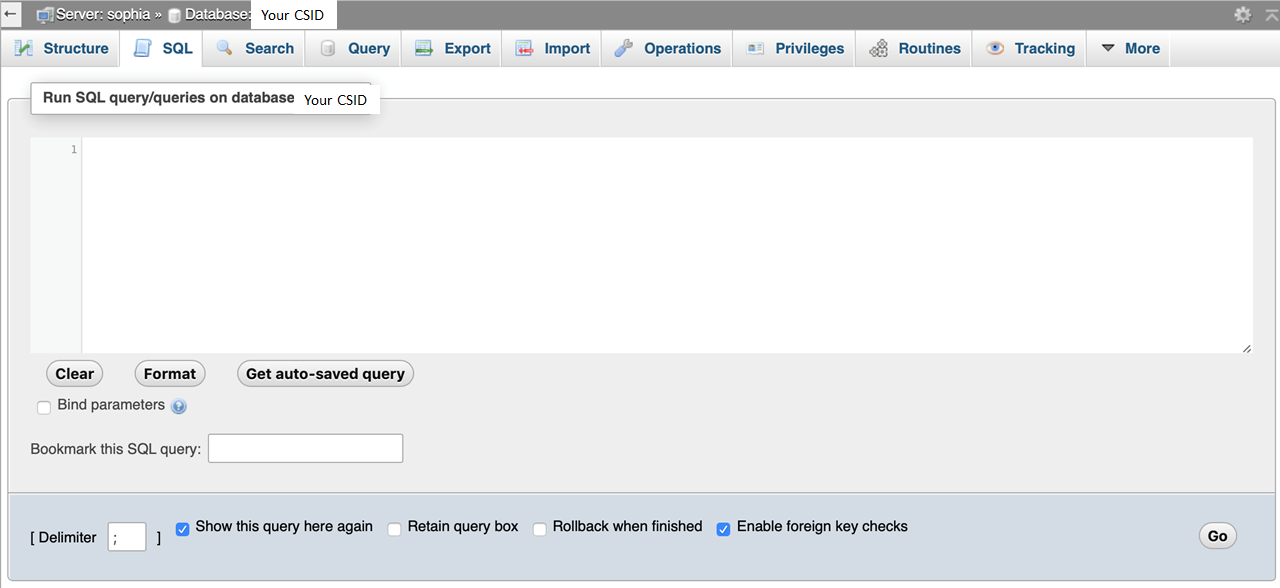
**Step2:** Log in to MySQL using phpMyAdmin.

1. Go to <http://i.cs.hku.hk/tools/phpMyAdmin/>.
2. Log in using your MySQL account, and select the server, sophia. We will use the MySQL database hosted in sophia.cs.hku.hk.



**Step3:** Select your database. After login, select your database on the left column. Your database should be named as your CSID.

**Step4:** Create tables and records. Click the **SQL** tab and execute the SQL statements for creating necessary tables and inserting data for this lab exercise.



|  |
| --- |
| create table stockList (  id int not null primary key,  stockname varchar(256),  stockcode varchar(10),  category varchar(256),  date DATE,  status varchar(20) not null default 'RISE'  );  insert into stockList values(1, 'Microsoft', 'MSFT', 'Internet', '2017-10-05', 'RISE');  insert into stockList values(2, 'Microsoft', 'MSFT', 'Internet', '2017-10-06', 'FALL');  insert into stockList values(3, 'Facebook', 'FB', 'Internet', '2017-10-05', 'FALL');  insert into stockList values(4, 'Facebook', 'FB', 'Internet', '2017-10-06', 'RISE');  insert into stockList values(5, 'JPMorgan', 'JPM', 'Bank', '2017-10-05', 'RISE');  insert into stockList values(6, 'JPMorgan', 'JPM', 'Bank', '2017-10-06', 'FALL');  insert into stockList values(7, 'Goldman Sachs', 'GS', 'Bank', '2017-10-05', 'FALL');  insert into stockList values(8, 'Goldman Sachs', 'GS', 'Bank', '2017-10-06', 'RISE');  ………… |

**Lab Exercise 2 – Load entries from the database**

Download **lab3\_materials.zip** from Moodle. Unzip it and you will find all four files we need in this lab.

Open **index.html** with a text editor. You will see it contains the following HTML content:

|  |
| --- |
| <html>  <head>  <title>Stock Status System</title>  <link rel="stylesheet" type="text/css" href="style.css">  </head>  <body>  <h1>Stock Status System</h1>  <div id="List">  <div id="entries">  </div>  <div id="button\_all" class="buttons">  <p> Show All</p>  </div>    <input id="stockcode" type="text">  <div class="buttons">  <p>Filter by Stockcode</p>  </div>    <input id="category" type="text">  <div class="buttons">  <p>Filter by Category</p>  </div>  </div>    <script>  //to be implemented  </script>  </body>  </html> |

**Task 1. Implement client-side code for loading all entries**

When the page is loaded or the “Show All” button (on the page views as in Fig. 2 or Fig. 3) is clicked, load and display all the stock entries from the stockList table in the database, using AJAX.

**Step 1:** In **index.html,** identify the events that initiate the AJAX communication:

1. <body> is loaded. **Event: onload.**
2. <div id="button\_all" class="buttons"><p> Show All</p></div> is clicked. **Event: onclick.**

**Step 2:** Define the event handler (JavaScript function): **showAll()** for both events.

In the event handler, create an **XMLHttpRequest** object.

|  |
| --- |
| <script>  **function** **showAll**(){  var **xmlhttp**;  if (window.**XMLHttpRequest**){  **xmlhttp** = new **XMLHttpRequest**();  }else{  **xmlhttp** = new **ActiveXObject**("**Microsoft.XMLHTTP**");  }  //More code here for showing all entries  }  </script> |

**Step 3:** Define the response actions when the server’s response is received.

|  |
| --- |
| **xmlhttp**.**onreadystatechange** = **function**(){  if (**xmlhttp**.**readyState** == **4** && **xmlhttp**.**status** ==**200**){  var mesgs = document.getElementById("entries");  mesgs.innerHTML = xmlhttp.responseText;  }  } |

**Step 4:** Define the request that is sent “behind the scenes”:

|  |
| --- |
| **xmlhttp.open("GET","queryEntries.php?show=all", true);** |

**Step 5:** Send the request.

|  |
| --- |
| **xmlhttp.send();** |

**Step 6:** Hide the “Show All” button from the full list display view (as in Fig. 1), as follows:

|  |
| --- |
| **document.getElementById("button\_all").style.display = "none";** |

**Task 2. Implement client-side code for loading selected entries based on “Stockcode” or “Category”**

When the “Filter by Stockcode” button is clicked, load and display all the stock entries whose stockcode matches the entered text in the textbox above the button, using AJAX.

**Step 1:** In **index.html,** identify the event that initiates the AJAX communication.

**Step 2:** Define the event handler (JavaScript function): **filterS()** for the event.

In the event handler, create an **XMLHttpRequest** object and define the response actions when the server’s response is received, following steps 2 and 3 in Task 1.

Define the request that is sent “behind the scenes” as follows:

|  |
| --- |
| **xmlhttp.open("GET","queryEntries.php?show=stockcode&value="+stockCode, true);** |

Here stockCode should be the value user enters in the textbox above the “Filter by Stockcode” button.

Send the request following step 5 in Task 1.

**Step 3:** Display the “Show All” button in the page view (as in Fig. 2), learning from what we did in step 6 of Task 1.

Similar to the above steps, implement the client-side code to achieve the functionality that when “Filter by Category” button is clicked, load and display all the stock entries whose category matches the entered text in the textbox above the button, using AJAX.

**Task 3. Implement server-side code for loading entries from the database**

Create **queryEntries.php** to load required entries from the database upon client’s request.

**Step1:** Connect to database.

|  |
| --- |
| <?php  //Connect to database  **$conn=mysqli\_connect**(**'database-server'**, **'user-name'**, **'password'**) or **die**('Error! '. **mysqli\_error**($conn));  ?> |

**Step2:** Select database.

|  |
| --- |
| //Select database  **mysqli\_select\_db**($conn, **'your-db'**) or **die**('Error! '. **mysqli\_error**($conn)); |

**Step 3:** Create a SQL query according to client request (i.e., select all entries, or entries whose stockcode or category matches the received value).

For example,the following query selects all entries from the stockList table.

|  |
| --- |
| //Construct your SQL query here  $**query** = ' **select \* from stockList**; |

**Step 4:** Execute the query.

|  |
| --- |
| //Execute SQL query  $**result** = **mysqli\_query**($conn, $**query**) or **die**('Error! '. **mysql\_error**($conn)); |

**Step 5:** Fetch the results and generate HTML code that displays the entries.

|  |
| --- |
| while($row = **mysqli\_fetch\_array**($result)) {  print "<div id=".$row['id'].">";  print " …" //Add code to display the entries  print "</div>";  } |

In the second “print” line above, you should print the HTML code to display each retrieved entry. Organize each entry as <span></span><h3></h3><h5></h5>. The **<span>** element is used to show RISE or FALL. **<h3>** is used to show the stock name and category. **<h4>** is used to show the stock code and date.

**Lab Exercise 3 – Toggle RISE/FALL**

Tapping the word “RISE” or “FALL” in each entry on the page will change the status from one to the other, and initiate AJAX communication to the server for changing the attribute “status” of this entry in the database table, without reloading the entire web page.

**Task 1**

Add an event handler changeState(this) to the onclick event on the **<span>** element of each loaded entry. Implement the event handler function changeState(elem) by following the steps in Task 1 of Lab Exercise 2, to implement the AJAX code for asynchronous communication with the server, and to display the received new status in the entry.

|  |
| --- |
| function changeState(elem) {  var oldValue = elem.innerHTML;  var newvalue;  var itemID = elem.parentNode.getAttribute('id');  if (oldValue == 'RISE') {  newvalue = 'FALL';  } else {  newvalue = 'RISE';  }  var xmlhttp;  // Add AJAX code here  ………………………….. |

**Task 2**

Create **updateState.php** to implement the server-side logic of changing the “status” attribute of the corresponding entry in the database. The following code is given for your reference.

|  |
| --- |
| $value = $\_GET['newValue'];  $query = "update stockList set status= '$value' where id=".$\_GET['id'];  ……………………. |

**Submission**

Please finish this lab assignment before 23:59 Sunday Mar 3, 2019. Please upload all files (**index.html, queryEntries.php, updateState.php, and style.css**) to **i.cs.hku.hk** web server under “**lab3**”, similar to what you did in previous labs. The URL to access your page should be **http://i.cs.hku.hk/~[your\_CSID]/lab3/index.html**. We will check the page for your lab 3 marking. Be sure to check the page yourself after uploading it to the web server.