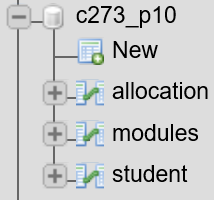
Worksheet 10

**Manage Students**

# Setup

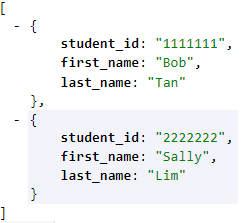
1. Download *C273\_L10CRUD.zip* from LEO. Save and extract it to C:\xampp\htdocs\C273\_L10CRUD\.
2. Start NetBeans. Create a PHP Project with Existing Source that points to above folder.
3. Start the XAMPP Control Panel, and **start both the Apache and MySQL component**.
4. Type <http://localhost/phpmyadmin> on your browser to open PHPMyAdmin.
5. Create database **c273\_p10**. Import the file **c273\_p10.sql** into that database. You should have the following tables:



# Read function (View students)

#### In this section, we are going to use AJAX to make an asynchronous call to a PHP webservice to retrieve database records and display them in an HTML table.

1. Go to <http://localhost/C273_L10CRUD/getStudents.php> from the browser to see the JSON message below:

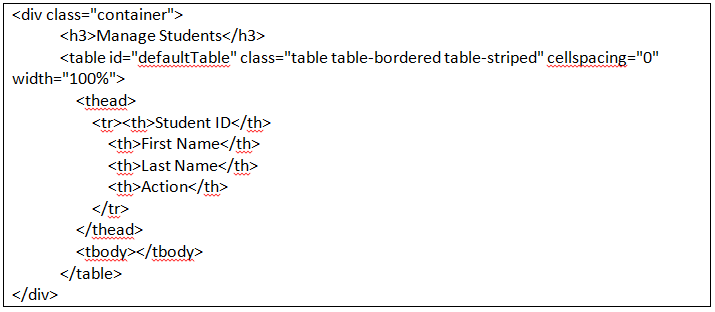


**Note** that this webservice returns a JSON response that contains multiple objects, therefore you need to use a for loop in the jQuery code to process the parsed JavaScript response that is an array of objects.

1. Create a new HTML page, **manageStudents.html** and link up the following files:

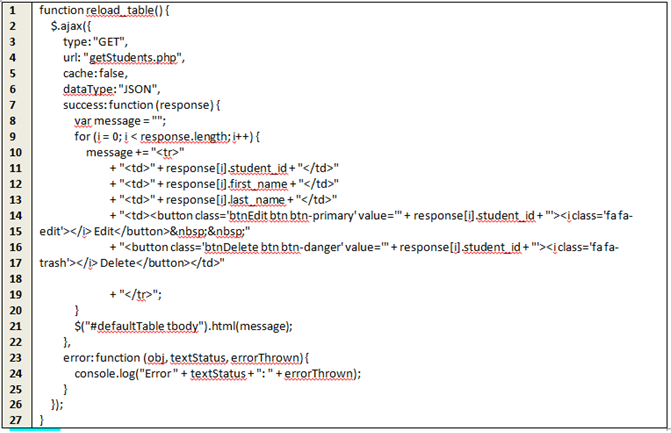


1. Type the HTML code within the <body> element to create an empty table to store the list of students:



1. Create a new JavaScript file **manageStudents.js** inside the js/ folder. Remember to link manageStudents.js file to manageStudents.html.
2. Type the following jQuery code in **manageStudents.js** to create a function called **reload\_table()** which makes an AJAX call to getStudents.php and display the student records in the id "defaultTable".

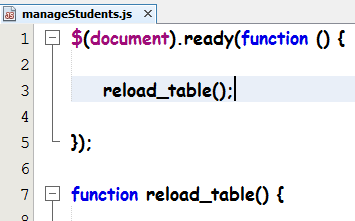
Functions should be placed outside $(document).ready(function() {…})



Line 9-20 – The JSON response contains multiple students, thus a for loop is used to iterate the array of JavaScript objects. Each object contains 3 attributes: student\_id, first\_name and last\_name. Here the student id, first name and last name is displayed on each row of the table.

Line 14 and Line 16 - 2 buttons: Edit and Delete are also created in each row of the table with the student\_id attached to their values. We will write jQuery events to trigger these buttons in the Sections Edit and Delete Student.

1. Call reload\_table() inside $(document).ready(function () {…});



1. Run manageStudents.html and you should see the list of students displayed in the Bootstrap table.

# Create function (Add Student)

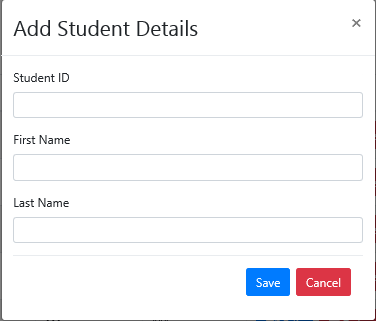
### In this section, we will be adding a Add button and a Bootstrap modal to hold the form

**Resources**: <http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-modals.php>

1. We need to create a [**Add**] button in **manageStudents.html**. Add the following HTML code between the title and table.



A Bootstrap modal contains 3 sections: modal-header, modal-body and modal-footer.

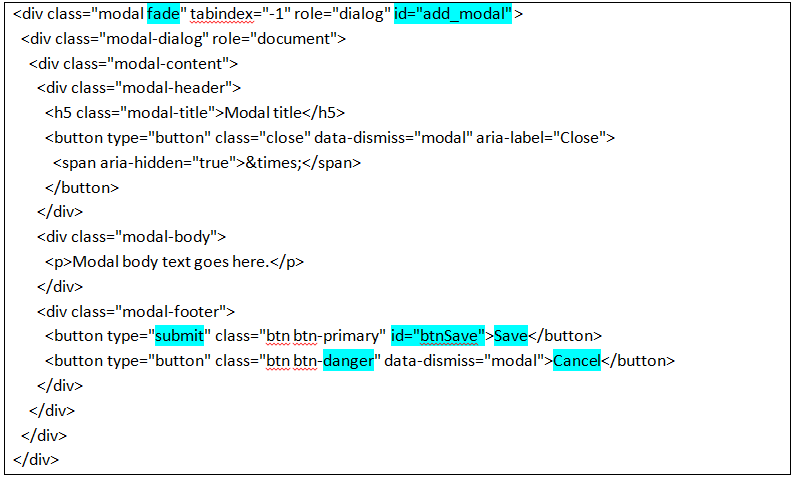


modal-footer

modal-body

modal-header

1. Go to <https://getbootstrap.com/docs/4.0/components/modal/> and copy the modal code containing the modal-header, modal-body and modal-footer. Paste the modal code under the table code in manageStudents.html.
2. Make the following changes to the modal code:



**insert </form> here**

**Replace this code with a Bootstrap form with 3 input fields: studentid, firstname and lastname.**

|  |
| --- |
| <div id="addErrorMsg"></div>  <form action="#" id="add\_form" method="post">  <div class="form-body">  <div class="form-group">  <label>Student ID</label>  <input name="**studentid**" class="form-control" type="text" required>  </div>  <div class="form-group">  <label>First Name</label>  <input name="**firstname**" class="form-control" type="text" required>  </div>  <div class="form-group">  <label>Last Name</label>  <input name="**lastname**" class="form-control" type="text" required>  </div>  </div> |

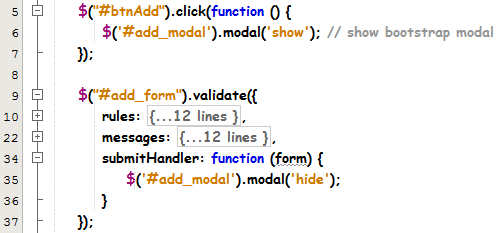
1. In manageStudents.js, type the jQuery code (shown below) inside $(document).ready(function () {…});

**Make sure you specify the rules and corresponding error messages in Line 10 and 22 respectively.**

Student ID: required and numbers only

First Name: required

Last Name: required

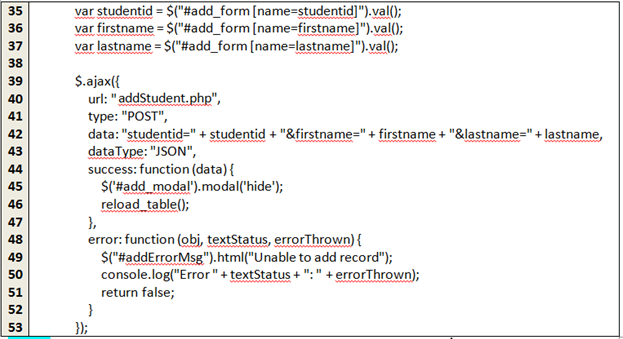


Line 1-3 – Upon clicking the button selector of id “btnAdd”, the modal “add\_modal” will be shown.

Line 35 – Upon clicking the button selector of id “btnSave”, the modal “add\_modal” will be hidden. We only hide the modal, after successful form validation, therefore we need to conditionally handle the submit event.

Run manageStudents.html. Click on the “Add” button to show the modal. Click on the “Save” button to close the modal

1. **Replace** line 35 in the previous step with the code below. The code below will make an ajax request to addStudent.php to insert the student record. Upon successful insert, the add modal will be hidden and the table of student records will be refreshed.



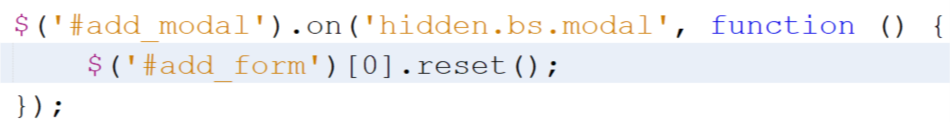
Line 46 – Call reload\_table() to refresh the list of students with the newly created student details

Line 49 – Display an error message if the webservice call returns any errors

Line 51 – return false for control to remain on the modal

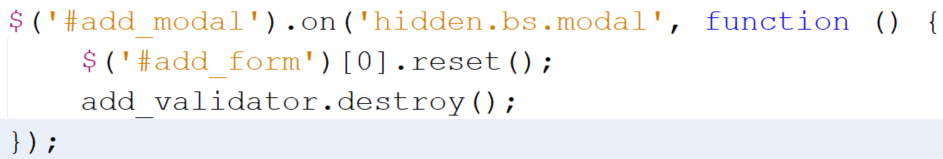
Run it again, click on the “Add” button and enter new student details in the modal. Click on the “Save” button and check that record is saved into the database via PHPMyAdmin.

1. Test the app by clicking on the Add button, key in some values and save the form. Check that record is saved into the database via PHPMyAdmin.
2. Perform the same test again and you will see that the modal form is populated with values from the previous entry. Add in the code below in manageStudents.js to reset the form values once the add modal is hidden.



1. You may also want to remove any error messages triggered by the Validator plugin. See <https://jqueryvalidation.org/Validator.destroy/> . Assign the validation instance to a variable and invoke the destroy() method on that instance.





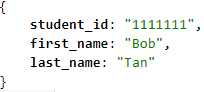
# Update function (Edit Student)

Watch the video “3\_Edit\_Student\_Part1” in LEO on how to code Step 1-3 of this function.

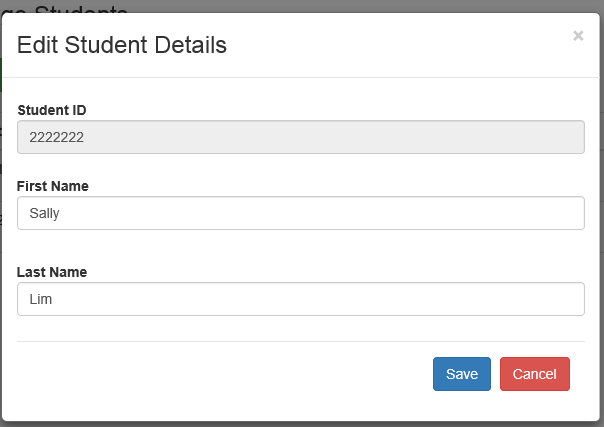
Watch the video “3\_Edit\_Student\_Part2” in LEO on how to code Step 5.

After the video, you need to continue coding step 7 in this section.

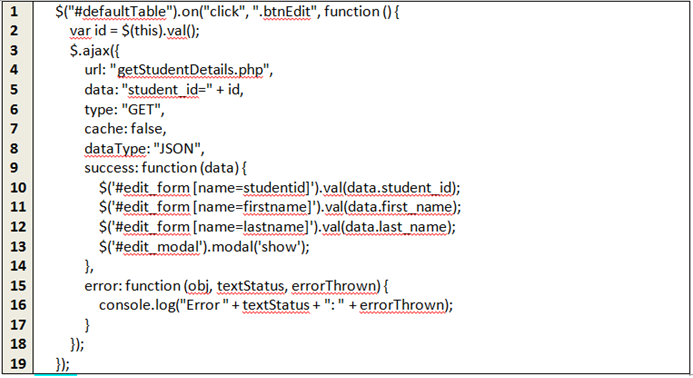
1. Go to <http://localhost/C273_L10CRUD/getStudentDetails.php?student_id=1111111> from the browser to see the JSON message below:



1. In **manageStudents.html**, add HTML code for another modal for edit (refer to Section Add Student Step 2). The modal should have id "edit\_modal" and its form should have id "edit\_form". This form displays 2 fields for editing "firstname" and "lastname". "studentid" field is a readonly input field.



1. In manageStudents.js, type the jQuery code inside $(document).ready(function () {…});



Line 1 – Since the Edit buttons are created dynamically in reload\_table() function, the click event cannot be attached to the class “btnEdit” i.e $(“.btnEdit").click(function() {..}); The on() method is used to bind the click event to the table id “defaultTable”. This is called delegated binding.

Line 2 – The value of the id is stored in the Edit (class=btnEdit) button

Line 5 – The id is used as a parameter when making the AJAX call to retrieve a specific record

Line 10-12 – The JSON response will be converted into a JavaScript object with attributes student\_id, first\_name and last\_name. They will be set into the form field values using the jQuery’s val() method. Note that the selector $("#edit\_form [name=studentid]") is used instead of $("[name=studentid]") as there are 2 forms using the same input name "studentid".

Line 13 – The edit modal will be displayed

1. Run manageStudents.html. Click on the “Edit” button for one of the student records to show the modal.
2. Upon clicking the Save button in the edit modal form, the submit event will be triggered. Write the jQuery code (Refer to Section Add Student step 4) for the submit event in manageStudents.js to cater for the saving of form input values for the Edit function. The webservice to call is <http://localhost/C273_L10CRUD/editStudent.php>
3. Test your code by clicking on the “Edit button” for one of the student records and modify the fields in the modal form. Click on the “Save” button and check that changes to the record is saved into the database via PHPMyAdmin.
4. Write the jQuery code similar to Steps 6 and 7 in Section Add Student to reset the edit form once the edit modal is hidden and initialize the Bootstrap validator plugin when the edit modal is shown. This is similar to Add Student step 8.

# Delete function (Delete Student)

Complete the jQuery code for the Delete function in manageStudents.js

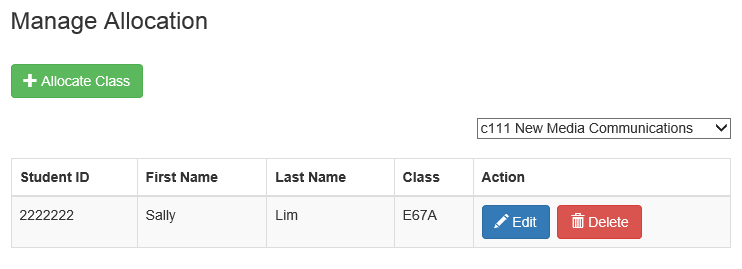
**Hint:** The code is similar to step 3 of the Update function. Use the confirm() method to check with the user on the deletion of the student record. Make the ajax call to deleteStudent.php webservice to remove the student record.

# Complete the Problem Statement

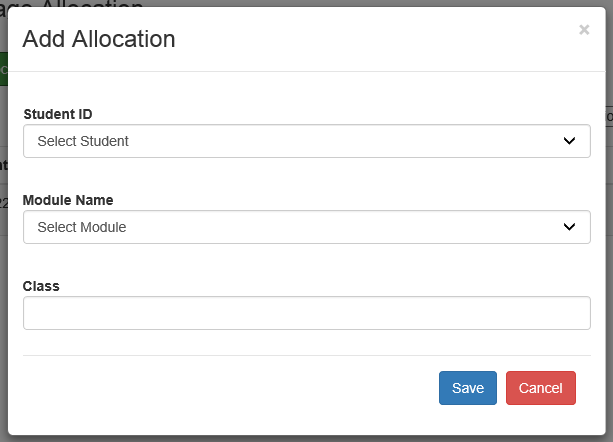
* 1. Create CRUD functions for module table.

# Challenge Yourself

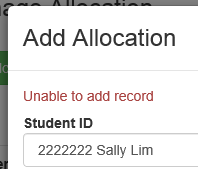
* 1. Develop manageAllocations.html like the diagram below. When the user selects the module, all students allocated to that module will be displayed in the table.



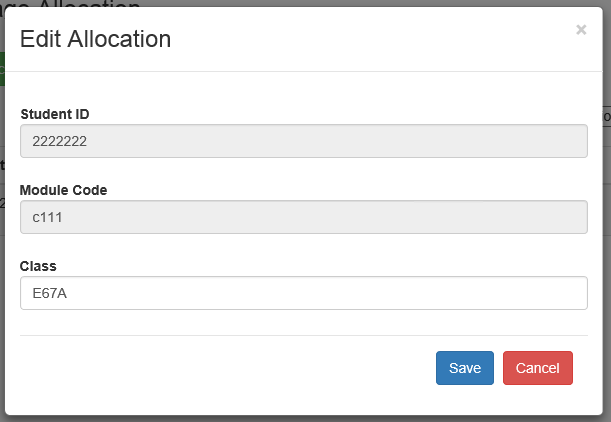
* 1. When the user clicks on [Allocate Class] button, he/she sees the screen below. All student records will be displayed in the [Select Student] dropdown list and all modules will be displayed in the [Select Module] dropdown list.



In the event that student is already allocated to that module, an error message will be displayed like the diagram below:



* 1. When the user clicks on [Edit] button, he/she sees the modal below with the student id , module code and class prepopulated. The student id and module code fields are readonly.



* 1. When the user clicks on [Delete] button, the allocation record will be removed.