# awsProject2

A repo for academic experimentation with Bitnami LAMP on Amazon Bitnami ec2 instance.

# GWMSIST

## awsProject2

### A repo for academic experimentation with Bitnami LAMP on Amazon Bitnami ec2 instance.

1.Requirements listed below documentation.

2. Example link provided:

1. A Bitnami LAMP on Amazon Bitnami ec2 instance to develop a web application that is able to validate user input as well as store, retrieve, update and delete items in a database following URL:

1. `http://ec2-52-33-59-87.us-west-2.compute.amazonaws.com/project2/standard-form.php`

3. User Stories: As a user I want to:

2. Land on a welcome page featuring a prominent greeting navigation menu bar, a side-bar menu and a footer menu; the landing page will also include a section containing a button that directs to the documentation provided herein. In order optimize content appearance and layout while affording maximum responsiveness, the [Bootstrap (v3.3.6)](http://getbootstrap.com/getting-started/) framework will be utilized-more specifically, a rendition of the "Sticky Footer" theme. The landing page will be appropriately meta-tagged and include personal favicon.

3. In order to accommodate the aforementioned, the most recent version of Bootstrap was downloaded and included with source files.

2. The navigation-bar menu should feature an appropriate logo for the production, in addition to three tabs respectively navigating to "Register" and “Members” pages of the site, in addition to a “Dropdown” tab, which may feature three optional pages if need be.

1. Please note, all Bootstrap files are retrieved remotely via the following URL's `https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css` & `https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js`.

3. All of the aforementioned sections are constituents of the general website layout.https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwi-iuX11qjMAhUqtYMKHU\_-AhoQjRwIBw&url=http%3A%2F%2Fwww.tshirtsonscreen.com%2Fsilicon-valley-i-know-html-how-to-meet-ladies%2F&psig=AFQjCNFMqpyrV7xjscafygaWn8HUrcbDqw&ust=1461635562055513

4. Upon populating the form on the landing -page, then clicking on the "Register" tab within the navigation-bar menu, the user will be directed to a page featuring the page title, a subheader including a brief explanation, and then body content instructing the user to access the alternative form featuring standard form-validation with JavaScript.

5. Unlike Project I, Project II form-validation within the "Register section" of landing-page of the site is enabled by manual [php RegEx](http://www.tutorialspoint.com/php/php\_regular\_expression.htm) performing form-validation via php regular expressions (RegEx) in order to ultimately submit the field-form as a record to the MySQL database residiing on the AWS instance, server-side.

6. If the user were to test the validation without populating the necessary fields (in this case, all), the form will not process and flag required fields. However, if the user populates every field, albeit incorrectly, the form will not process as the Modernizr API accounts for all of the required formats listed.

2. First Name

10. >= 2 characters

3. Last Name

11. >= 2 characters

4. E-mail

12. Must contain a "@" character in addition to valid domains.

5. Phone #

13. Phone number must contain the numbers and parenthesized area-code.

6. Address

14. Must contain a specified set of characters, including at least one digit, street direction, street and street type.

7. City

15. Provided city must be accounted for within Modernizr database-API.

8. State

16. Must select a state within drop-down menu selection

9. Zip Code

17. 5 digits only, accounted for by Modernizer database-API

10. Website or Domain

18. No constraint

11. Do you have hosting?

19. "Yes" or “No” radio buttons- only one must be select.

12. Project Description

20. Constraint: input >10 characters <200

7. If the user appropriately populates all of the registration fields, upon hitting the "Submit" action button, the user’s client submits the data via HTTP request, using the [get\_post function](https://developer.wordpress.org/reference/functions/get\_post/) that retrieves post-data given a post ID or post-object for insertion into the previously mentioned MySQL database, server-side, which is verified by the [php ISSET function](http://php.net/manual/en/function.isset.php)- essentially checking for "nullability" or if the field tied to the corresponding variable is set to a corresponding value specified by user-input upon passing form-validation (previously mentioned RegEx). The data collected is retreived by the associated php variables prefixed by '$' to the assigned values corresponding to the values definining the given table(s) of the server-side MySQL database- again all assigned to and inserted according to the assigned tags input name-class tags featured within the div-class tag-group pertaining to each corresponding field (e.g. <input name=”email” will be rendered as email=”[field input]”, connected by [php declarations connecting to the server-side MySQL database](http://www.w3schools.com/php/php\_mysql\_connect.asp).

6. Upon clicking the "Members" tab within the navigation-menu bar, the page features a corresponding table rendering the fields processed by the corresponding form located on the landing-page that is responsively formatted with a “table-responsive” div-class featured by Bootstrap. Essentially, the action method’s will be contained within each HTML element, will correspond to the MySQL query (e.g. the email id assigned by the GET action method should render in the corresponding table-row cell) for each iteration of the JSON string via [jQuery](https://jquery.com/)- a fast small, feature-rich JavaScript library, making things like HTML document traversal and manipulation, event handling, animation and Ajax mich simpler with an API [application program interface] that works across a multitude of broswers. The requirements specified static-seed data in the table, which includes contact-information for my web development company, [Bucephalus Web Development](https://bucephalus.io).

7. Lastly, the footer menus on every page will link to the "Home" page, an “About” page featuring the project requirements and an experimental “Contact” form, also facilitated by Bootstrap styling, in addition to the action methods correspondingly assigned but doesn’t feature form-validation in Release 0, but more than likely in the next iteration.

4. Additional Resources

9. Production schema and structure were verified by the following tools:

8. [Google’s Structured Data Testing Tool](https://developers.google.com/structured-data/testing-tool/)

9. [and JSFIddle](https://jsfiddle.net/).

\* http://www.w3schools.com/php/php\_mysql\_update.asp

\* http://www.w3schools.com/jquery/default.asp

\* http://www.w3schools.com/jquery/jquery\_ajax\_intro.asp

\* http://www.tutorialspoint.com/php/mysql\_update\_php.htm

\* https://github.com/myprogrammingblog/myprogrammingblog.com/tree/master/UI%20Components/contact%20form/js

\* http://www.w3schools.com/bootstrap/bootstrap\_dropdowns.asp

\* http://www.w3schools.com/bootstrap/bootstrap\_ref\_css\_tables.asp

\* http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs\_ref\_table-responsive&stacked=h

\* http://www.w3schools.com/bootstrap/bootstrap\_buttons.asp

\* http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs\_button\_elements&stacked=h

\* http://www.techrepublic.com/blog/microsoft-office/accommodate-different-headers-and-footers-in-a-word-document/

\* http://www.w3schools.com/tags/tag\_br.asp

\* http://www.w3schools.com/tags/tag\_br.asp

\* http://v4-alpha.getbootstrap.com/layout/grid/

\* http://uxmovement.com/forms/where-to-align-buttons-on-different-form-layouts/

\* http://getbootstrap.com/css/

\* http://www.w3schools.com/bootstrap/bootstrap\_buttons.asp

\* http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs\_button\_styles&stacked=h

\* http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs\_button\_elements&stacked=h

\* http://www.w3schools.com/bootstrap/exercise.asp?filename=exercise\_tables4

\* http://www.w3schools.com/tags/tryit.asp?filename=tryhtml\_span

\* http://www.w3schools.com/tags/tag\_span.asp

\* https://developers.google.com/structured-data/testing-tool/

\* https://validator.w3.org/check

\* http://www.w3schools.com/bootstrap/bootstrap\_forms.asp

\* http://www.w3schools.com/css/css\_inline-block.asp

\* http://www.w3schools.com/bootstrap/bootstrap\_tables.asp

\* http://www.generatedata.com/

\* http://getbootstrap.com/css/

\* http://www.bootply.com/88026

\* http://www.bootply.com/101100#

\* http://daringfireball.net/projects/markdown/dingus

\* http://www.w3schools.com/bootstrap/bootstrap\_buttons.asp

# GWMSIST

## Project II Requirements

### Project Description

The goal of the second student project is to develop a web application that is able to validate user

input as well as store, retrieve, update and delete items in a database.

This assignment consists of a web application that has, at a minimum, the following features::

1. \*Validate user input using PHP:\*The web application should allow the user to enter data in a form and then validate the

user input using PHP. This form should have at least 5 input fields. Once the user submits

the form, a PHP script should validate the data entered by the user in all input fields. In

other words, all mandatory and required fields should be validated. If the user input data

is not valid (i.e., missing and/or incorrect data), this data should not be stored in the

database and the web application should point out to the user which field(s) have invalid

data and why. If the user input data is valid, this data should be stored in the database.

2. \*Store user input in the MySQL database:\* The web application should store the data entered by the user in the form and validated

using PHP in the MySQL database. All data entered by the user and validated should be

stored in the database (i.e., mandatory and/or optional input fields).

3. \*Retrieve items in the MySQL database:\* The web application should allow the user to retrieve all items currently stored in the

database. Specifically, the user should be able to request a list of all items stored in the

database in a table format. The table columns and the user input form fields (associated

with functionality #1) must necessarily match. The data on the table is dynamic. In other

words, the content of the table changes according to user actions. For example, after a

user stores an item in the database, updates an item in the database, or delete item(s) in

the database, the table should reflect the respective changes in the database.

4. \*Update item in the MySQL database:\* The web application should allow the user to update an item currently stored in the

database. Specifically, the user should be able to choose an item to update from a list of

all items currently in the database. Once the user selects which item they want to update,

the web application should provide the user a form with the same input fields as the one

used to add items to the database. However, the fields of the form to update an item

should be filled in with the data currently stored in the database for that item. After the

user updates the field(s) they want to change and submits the form, the user input should

be validated using PHP in the same manner described above (refer to functionality #1).

The item should only be updated if the user input data is valid.

5. \*Delete item(s) the MySQL database:\* The web application should allow the user to delete item(s) currently stored in the

database. Specifically, the user should be able to choose item(s) to delete from a list of all

items currently in the database. Once the user selects which item(s) they want to delete,

the web application should remove such item(s) from the database.

###Characteristics

1. \*Semantic Markup:\* The HTML markup should avoid excessive, unnecessary inclusion of CSS and PHP.

With respect to CSS, external style sheets should be used to include CSS as opposed to

inline or embedded CSS. In the case of PHP files, the web programmer should strive to

keep the PHP script separate from the HTML markup to the extent possible. One

effective strategy is to use include statements in the PHP files to separate markup from

script.

2. \*Responsive Design:\* The entire web application (i.e., all web pages) should have a general design (e.g., top

navigation bar, bottom navigation bar, side items) that is responsive to different screen

sizes. All the functionalities described above should be available in a convenient layout

on devices of different screen sizes.

3. \*Navigation:\* There should be enough links in the web application to allow the user to easily navigate

through the web pages associated with the various functionalities described above. Note

that not all links included on your web pages need to actually work (i.e., they may point

to "#").

You may want to use your student project I as the basis for your student project II. For example,

the user input form and the table that you developed in the first project could be used in the

second project to gather data to be stored in the database and to display the items retrieved in the

database to the user, respectively. Furthermore, you may want to use the CSS stylesheets and

media queries you used in your first project to create a responsive design for your second project.

### NOTES

1. You are not required to use client-side validation in your second student project.

Regarding validation, you will only be graded based on server-side validation (i.e.,

validation using PHP).

2. You are not required to use project I as the basis for project II.

###Grading Criteria

\* PHP Validation: 18%

\* Store Item: 18%

\* Retrieve Items: 18%

\* Update Item: 18%

\* Delete Item(s): 18%

\* Semantic markup: 2.5%

\* Responsive design: 2.5%

\* Navigation: 5%

### Submission Instructions:

You must submit all the files you created/used in your web application and a Word document

that describes the functionality and characteristics of your application as well as the URL of your

project. Submission files may include:

###Submission Instrutions

You must submit\*\* all the files\*\* you created/used in your web application and a \*\*Word document\*\* that describes the functionality of your application. The submitted files include:

\* HTML file(s)

\* CSS file(s)

\* Javascript file(s)

\* php File(s)

\* Any other file you may have used (e.g., image files)

The Word document should briefly explain how and where you met the requirements of the

project both in terms of functionalities and characteristics as outlined above. It should be a

concise document that explains what you did (e.g., user input form validation using PHP), how

you did it (e.g., which fields are mandatory and which ones are optional, what are the validation

rules for each field), and where you did it (e.g., validation.php lines 10-100). In this description,

do not focus on the code you wrote (since you will also submit the files you created), but on the

functionality and characteristics of your application. Functionalities and characteristics that are

not documented will not be considering when evaluating your work.