Stefan Grulović (20150280)

Konstantinos Vezeridis

Computer Science 412: Object Oriented Programming

Fall 2017

CS412 – Final Project Word Report

# Application Requirements

In order for the application to run it need a virtual host in order to simulate a domain. For example, AMPPS, APACHE, XAMPP etc.



Example of virtual host applications.

The application uses PHP as main language, HTML, CSS and JavaScript/jQuery for styling.

****

The application folder consists of 4 folders the *base library*, *midterm exam, final exam* and *test application.*

The *base library* is the base with no specific use it is where the basic *model, view* and *controller* are located. They are used in other applications alongside with their specific controllers, models and views.

The *midterm exam* is an application where there is a company which has Crews which work on Projects that consist of Assignments. The application organizes the entities and provides a display for the same.

The *final exam* is an application where there is two types of user of the application the regular User and Register. The application starts with a login form which checks the user name and password and if correct the user is checked for their role and the corresponding application is displayed.

The *test application* is an application with a menu which holds the options for all applications and test for basic elements from the library.

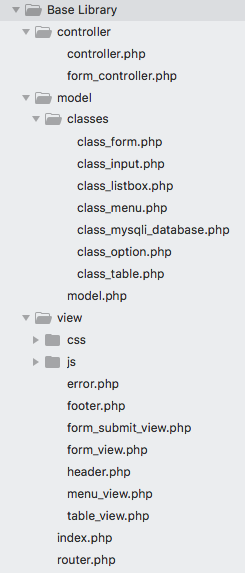
# Application Design

The application is designed using the Model-View-Controller(MVC) Pattern. The MVC pattern splits the application and functionality into three groups Model, View and Controller.

* Model is where the connection with the database is located. It is the location from where the data is exchanged.
* View is the output representation of the data.
* Controller is where user actions and inputs are interpreted and converted to commands for model and view.

By splitting and grouping the application functionality we have better organization of the code which is better than having everything mixed into one big file. This way the extension of the application and library is better as for whatever task we want we can reuse and or create a new building block which will make our work easier and more efficient. Overall, it provides a better organization, structure and reusability of code.

The application I have created is organized into three folders, the *model* inside with *classes, view* inside with *CSS* and *JS,* and finally *controller.* Also outside those folders we have two important files *router.php* and *index.php*.



*Example of folder/file structure of base library*

The *router.php* is where the logic of reading the URL is located. The URL basic logic used for the application is structured with three parameters:

* Controller which specifies the name for the controller. Which is consistent with the name of the class and files.
* Action which specifies the method of that controller which will be called.
* Params which are the arguments which will be sent to that action of that controller that we are calling.

Inside the *index.php* is where we instantiate the router object and start it by using function invoke where we pass three arguments default controller, action and parameters which will be called when the application is called even though the URL has not yet been modified.

In my application inside the *model folder* there is a *classes* folder where generic classes with get and set methods are located. Also, the *MySQLi* class is located there which is the main way of connecting to a database. It provides basic structure of establishing connection, connecting, closing connection and running a basic SQL command.

Inside *model folder* there is a file *model.php* which is where the *MySQLi* objects is instantiated. Using the basic functionality of the *MySQLi* object and new organization and more complex functions provide basic functionality used by other models in order to select, delete, insert or update the data.

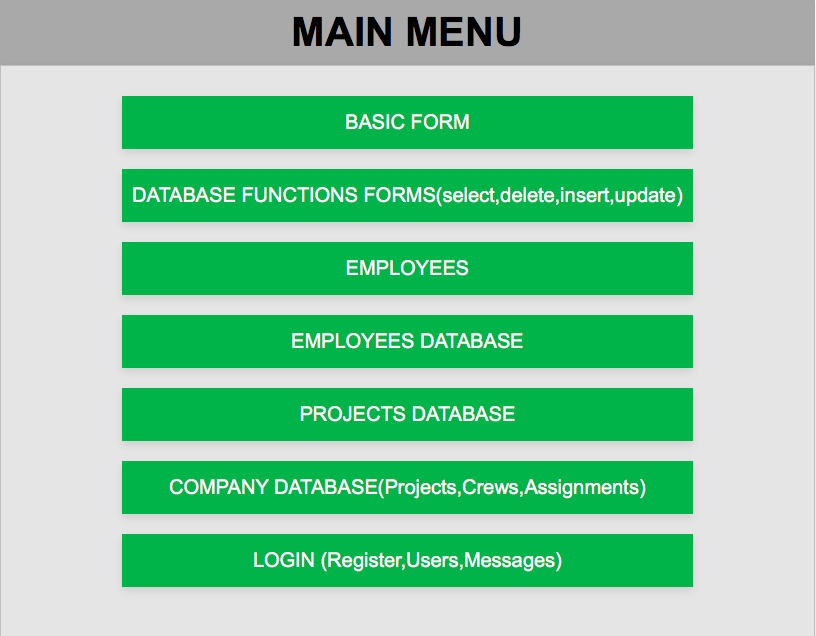
Inside *view folder* generic HTML files with PHP infused are located. The basic logic and structure of view files is the HTML structure with PHP references to the variables sent through controller. The variables can be either just strings which are echoed with specific HTML tags or objects which can be echoed in same manner but now using the get and set methods. Also, in case it is an array of objects those objects can be looped using foreach and get and set methods from the classes of those sent objects in order to be displayed.

Inside *controller folder* is where controllers are located which is what sends commands to model and then using view files is how they are then displayed. Besides retrieving the objects from model, controllers may have objects instantiated in the construct function which can be generic things such as menu, items, forms, inputs etc. which are then send to their generic views. It also interprets the user interaction with the view and then again acts accordingly in order to do whatever the buttons inputs etc. specify they do.

# Functionality Examples

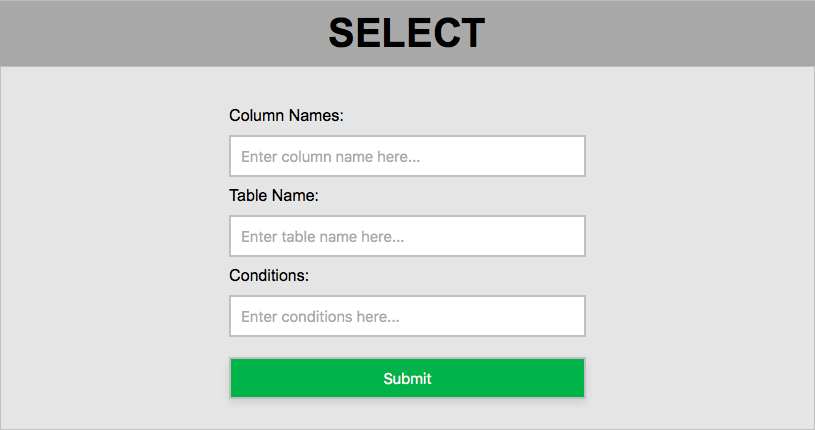
## -Basic functionalities:

* Menus are made from links/options which hold information of the controller, action and parameters which will edit the URL. Which then the *router* will interpret and take to the specific controller’s action with specified parameters.



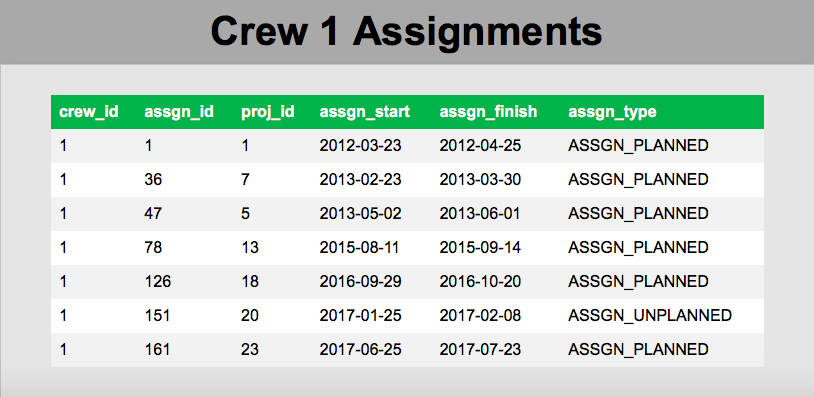
*Menu example (test application)*

* Forms are made from inputs which are read using the *form controller*. Specific forms extend the *form controller* and then invoke the parent function which will return the user values which in the new extended one are taken and acted accordingly to what the task requires.



*Form example(test application -> database function forms -> select)*

* Database Tables are created using the *model* function called select\_table which can represent any SQL statement and display it in the form of a table.

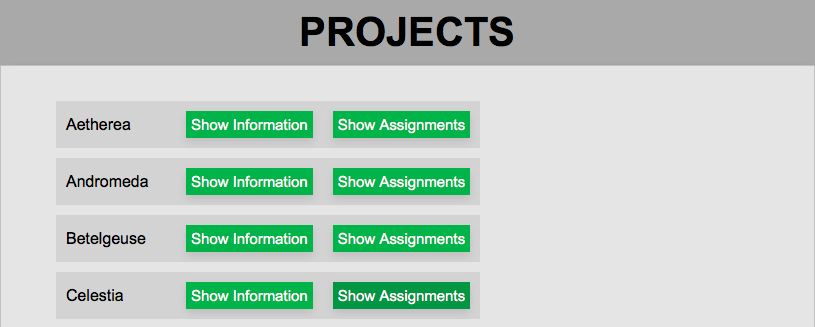


*Example of a SQL statement displayed in table form*

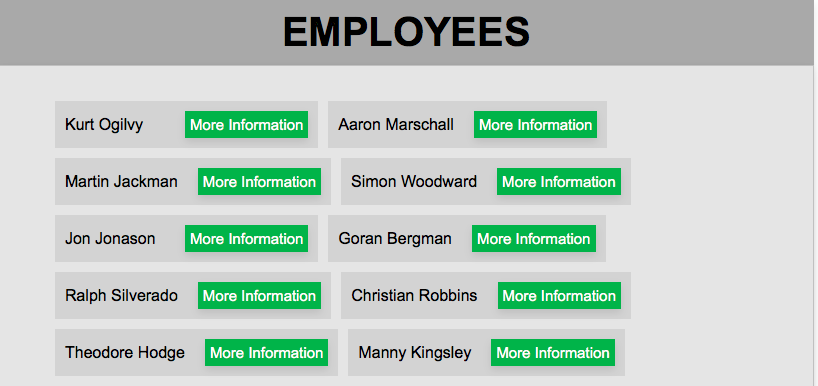
*(test application -> company database ->*

## -More specific functionalities:

* Entities display such as projects and employees with buttons to display more specific information.

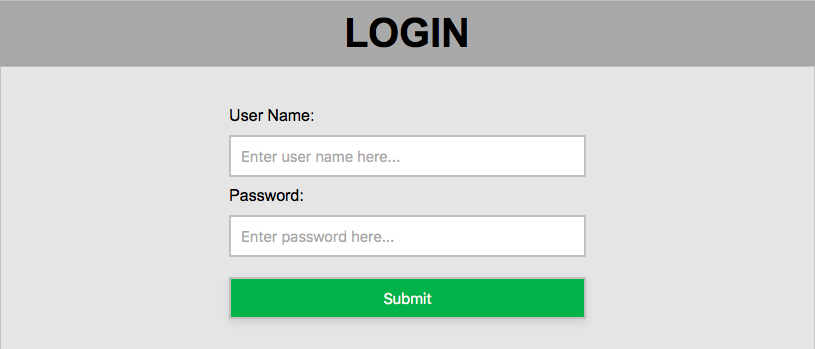


*Example of projects displayed with buttons to display more specific information and their assignments*

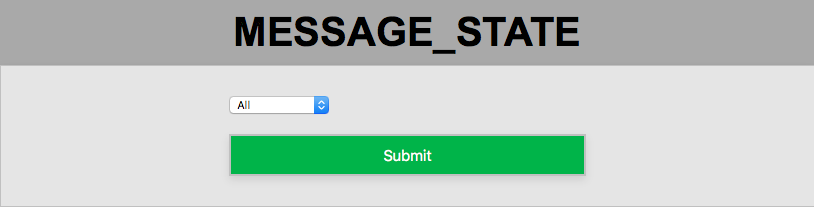


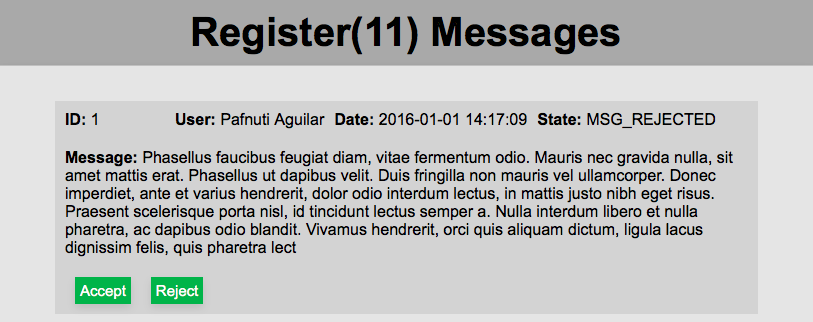
*Example of employees and button to display information about those employees*

* Login form which takes user inputted user name and password then goes to the database and check whether the combination exists. If it the combinations exits it checks the user’s role and accordingly to that displays the page which they need to see.

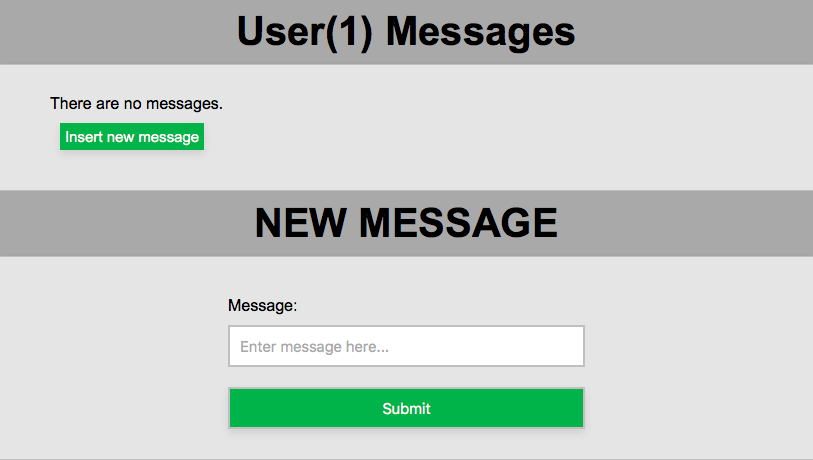


*Example of the login form*





*Example of register role (which can choose what messages to see)*



*Example of user role, where they see messages rejected by register and ability to insert a new message which will be sent to the register in order to get accepted or rejected*