**Description**

The emphasis of the web application built is for it to be Modular and Extensible. Taking these two properties into consideration it was decided that an MVC framework was the most logical choice. Upon deciding to use the MVC strategy, the best framework for our use was Codeigniter due to its minimal footprint and extensive documentation which made understanding the framework trivial.

Moreover, our model contained 5 files, each for a specific user and one for login and logout. Each model was built with additional features specific to each user type. This prevented the grotesque method of repeating code for each user and simplified adding a new user type.

Additional, the view contained many files with not only views specific to the user type, but header, footer and login views. This increased Modularity since each view did not depend on one another which is one of the goals of the application and enhanced Extensibility since the developer can modify a view without worrying about the underlying features.

Lastly, there are controllers for each user type and one for logging in and out. Each controller related to a specific user thus it enhanced Modularity and Extensibility because changes to one controller would not affect the other controllers and adding on functionality to a user is easy by simply calling the functionality from a model without knowing how the models work.

**Justification**

**Justification of the Models:**

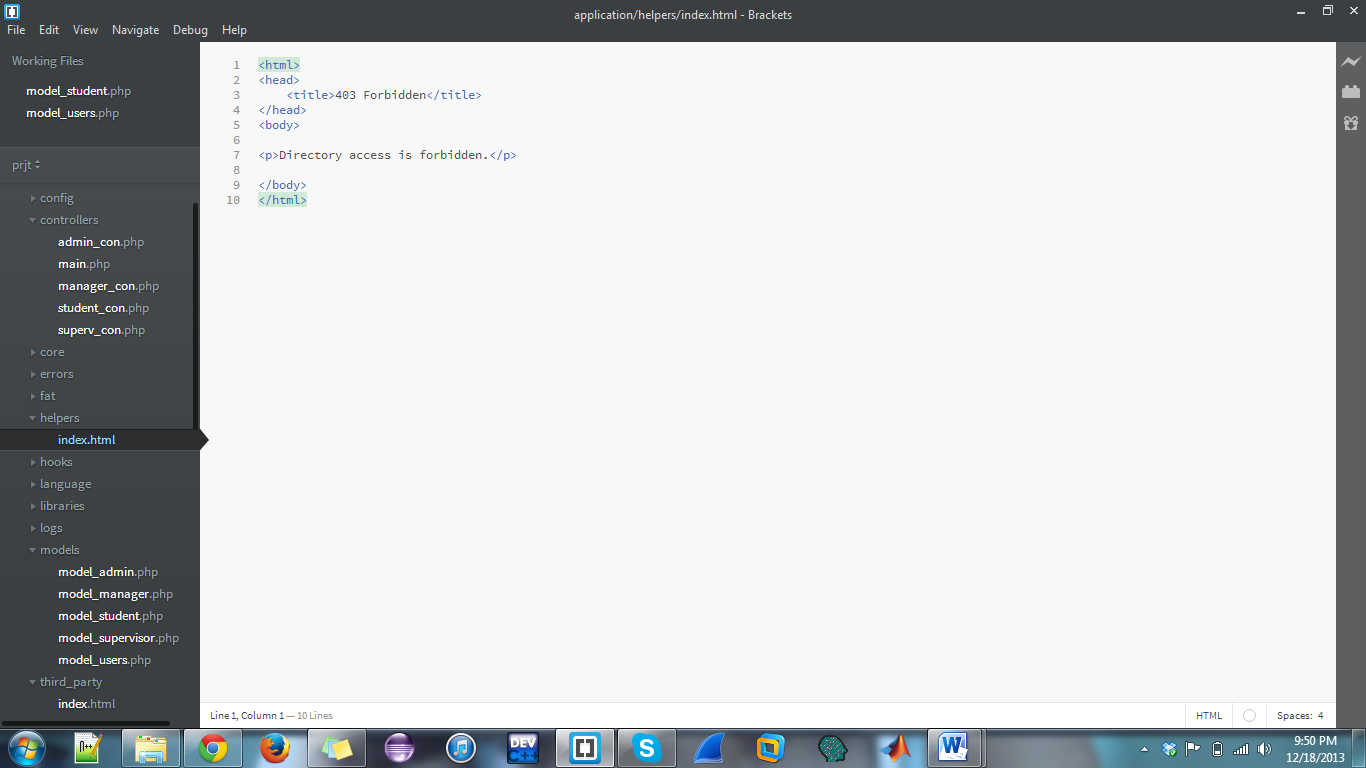


Figure 1. The figure shows the different Models in the application

In our MVC structured application, we had a Model for each type of user and as well as a general model for all users. These models had names respecting their operation such as “model\_admin” and “model\_supervisor”; which would be the models corresponding to their type of user.

In the “model\_student” model, there are functions to accomplish checking their personal information as well as seeing their usage graphs and tables. Admins, supervisors and managers can also view this information for them and will access this via the same code as from the “model\_student” model instead of repeating the same code in its own respective model. This removes code repetition and increases modularity.

**Justification of the Views:**

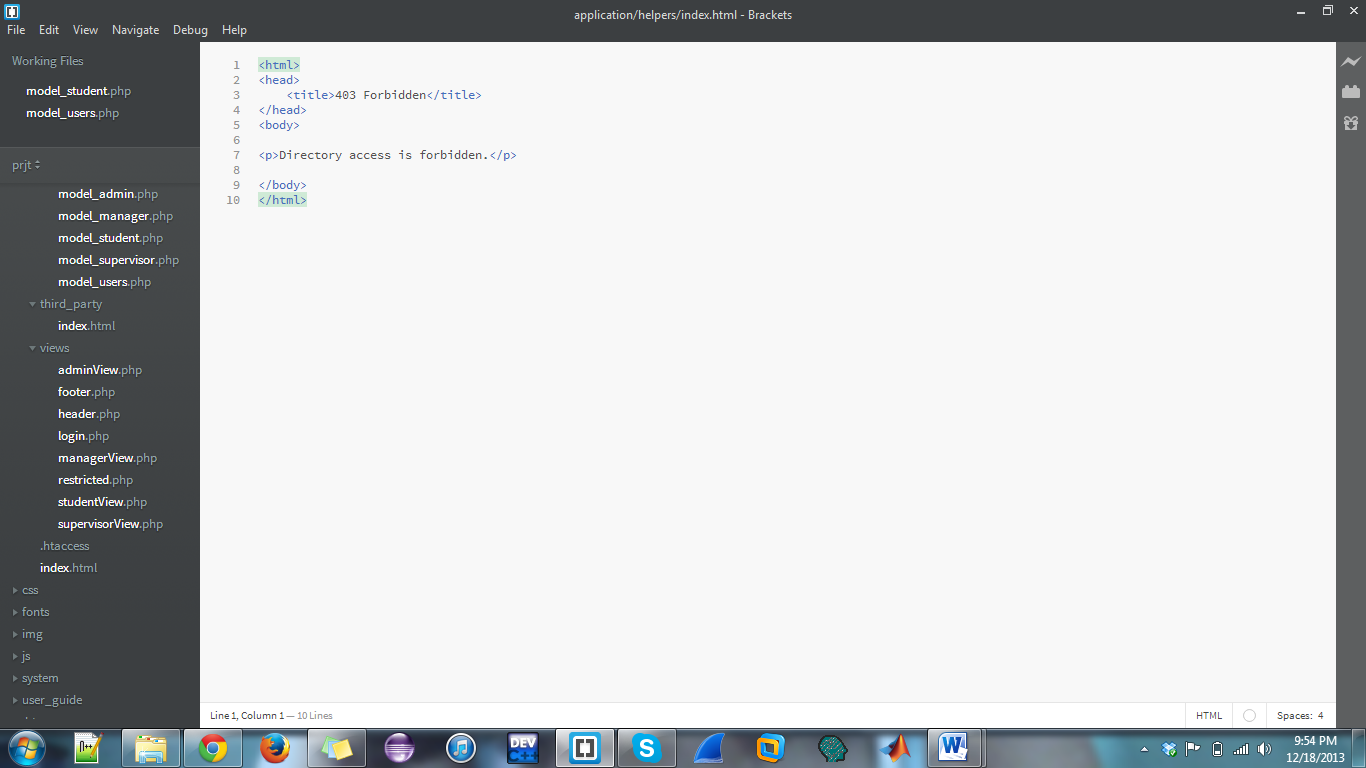


Figure 2. The figure shows the different Views in the application

**Justification of the Controllers:**

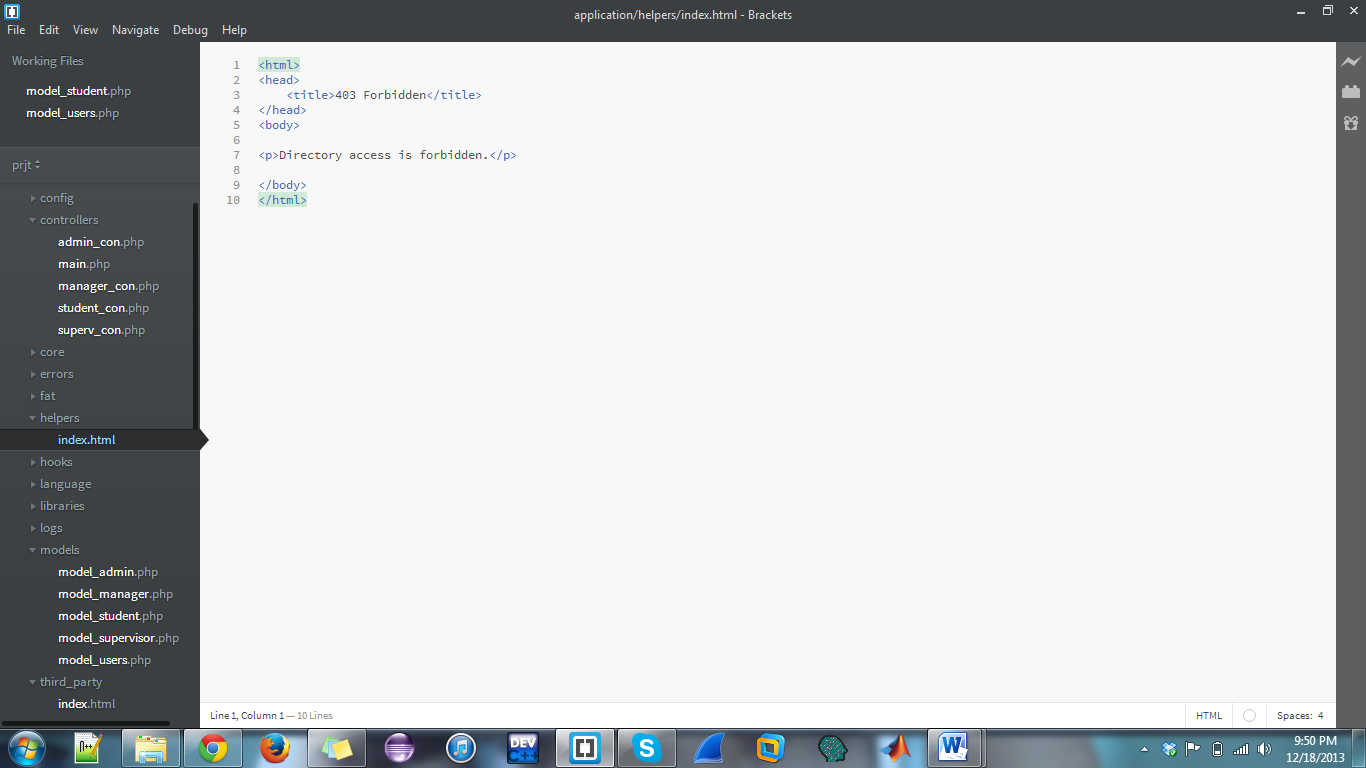


Figure 3. The figure shows the different Controllers in the application

There are 5 controller files, each representing a user type and one (main.php) for the logging in and out of users. Since a controller can send commands to the model to update the model's state and also send commands to its associated view to change the view's presentation of the model, we ensured that the controllers adhered to this format.

When a user logs into the application, it get the user type and displays a view corresponding to the user. An example is that if a Manager singed in then he would be presented with the Manger View and is not able to modify the URL to access another view as he/she would be presented with the restricted view. All this is done in the controller to ensure Modularity, Extensibility and prevent repeated code.

In addition, the controller is responsible for accessing the model to get functionality for that user. If the Manager decides to view all the information in the log table, then he/she would click the button for this and the information would be sent to the controller which would then call the model. The model would perform the task of getting all the log information and send it to the controller which would replay this information and it would be presented in the view.

**Justification of other components:**

Other components that we used in this project were Highcharts, Bootstrap and Codeigniter. Highcharts were used because it is a very easy-to-use and intuitive way of incorporating interactive and dynamic charts in a web application.

We used Bootstrap for designing our pages. It has a great set of readymade libraries which we incorporated into our design. Also bootstrap is responsive; this means that if our web application is viewed on some other device, such as an Ipad or Iphone, bootstrap will adapt to the change.

CodeIgniter is an MVC framework that we used for this project. It has the smallest footprint for any MVC structured template. It also had extremely user-friendly features and easy-to-follow tutorials in order to accomplish our needs. It made creating our own MVC based application a breeze.