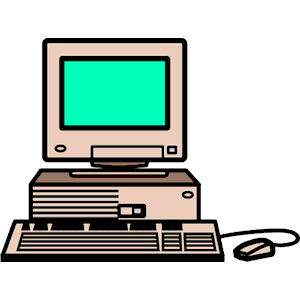
**ARCHITECTURAL DESIGN**

2. Route to appropriate Laravel controller

1. Submit user request

Route



Controller

-HomeController

3. Interact with data model

Model

-User.php

-Role.php

View

-accountant.blade.php

- accountantclerk.blade.php

-admin.blade.php

-admin2.blade.php

-auditor.blade.php

-auditorclerk.blade.php

-companydirector.blade.php

-companysecretary.blade.php

-editprofile.blade.php

-home.blade.php

-update.blade.php

-welcome.blade.php

4. Controller invoke result view

Database

5. Render view for user

Client server architecture is a producer-consumer computing architecture where the server acts as the producer and the client as a consumer. Client server architecture works when the client computer sends a resource or process request to the server over the network connection. A server computer can manage several clients simultaneously, whereas one client can be connected to several clients at a time. Simultaneous users can enter the website at the same time without any difficulty.

In this system, we propose the use of client server architecture because this system main operation is done online, main operation is for example the functionality to check trial balance and etc. Server will be the server of this system and client will be all the actor. Each of this actor in this system will interact with the server through a layer of communication. Aside from architecture style, we also propose to use MVC architecture pattern because it is compatible with client server and it provides a separation of concerns. MVC pattern allows the separation of each module so that they are easier to use and would not affect other module greatly.

When interacting with a Laravel application, a browser sends a request from the user, which is received by a web server and passed on to the Laravel routing engine. The Laravel router receives the request and redirects to the appropriate controller class method based on the routing URL pattern.

Accountant, auditor, accountant delegate, auditor delegate can have more than one role. They can request the new role by fill out a form and submit to the admin for verification. User with certain level of authority will only be able to access information or data according to their respective level. The controller will handle user request using suitable controller whether using HomeController.

Controller will interact with data model to retrieve data. After invoking the model, the controller then renders the final view and returns the complete web page to the user’s browser.

**Physical view**

|  |
| --- |
|  |

Diagram above shows the physical view of accounting system for module 1. Browser will send all the data such as forms authentication and roles through Https which is secured protocol. Browser pass the data to web server. for this module, we use MVC architecture pattern and this web server is protected network. Next, the web server will store the data in database through TCP/IP. Database will store the data in user table and roles table.

**Component Diagram**

|  |
| --- |
|  |

Diagram above shows the component diagram that User component provides information about user through IRegister and role component provides role information through IRole to the home controller. Meanwhile the information in the home controller is required by User\_Role component and Database component through IUser\_Role and IDatabase respectively.