

**SIM 3150 Web Application Development**

**Report of**

**Administrator Dashboard Module**

**Lecturer: Dr Ng Keng Yap**

**Group Members**

|  |  |
| --- | --- |
| **Name** | **Matric Number** |
| **Hanis Shahfiqah Muhamad Hanafi** | **179358** |
| **Anis Najib** | **177430** |
| **Muhammad Fakhrullah Bin Ahmad** | **180670** |
| **Chan Kai Fung** | **179960** |
| **Muhammad Alif Bin Azali** | **180159** |

**Table of Contents**

1. **Module overview……………………………………………………….………………3-4**

**1.0 Introduction…………………………………………………………………………...3**

**1.1 Project description……………………………………………………………………3**

**1.2 Objective………………………………………………………………………………3**

**1.3 Scope………………………………………………………………………………......4**

1. **Requirements…………………………..………………………………………………...4**

**2.0 Functional requirements…………………………………………………………......4**

**2.1 System use cases………………………………………………………………………5**

**2.1.1 View admin profile…………………………………………………………......5**

**2.1.2 Update admin profile…………………………………………………………..6**

**2.1.3 Create user………………………………………………………………….......6**

**2.1.4 View user profile………………………………………………………………..7**

**2.1.5 Delete user………………………………………………………….……...........7**

**2.1.6 Update user privilege…………………………………………………………..8**

**2.1.7 Update status…………………………………………………………………...8**

**2.1.8 View total record and total user……………………………………………....8**

**2.0 Use Case diagram…………………………………………………………..........9**

**2.2 Non functional requirements……………………………………………………….11**

**2.3 Activity diagram…………………………………………………………………….12**

1. **Architectural design……………………………………………………………………13**

**3.0 Model-View-Controller (MVC) pattern…………………………………………...13**

**3.1 Layered architectural style……………………………………………………...13-15**

**3.2 Component based architectural style………………………………………………15**

1. **Entity relationship diagram……………...…………………………………………….16**
2. **User interface designs……………………………………………………………….17-27**
3. **Abbreviations and definitions…..……………………………………………………...27**
4. **References………………………………………………………………………………27**

**1. MODULE OVERVIEW**

**1.0 Introduction**

This SRS describes the functional and nonfunctional requirements for the admin dashboard module. This document is intended to be used by the members of the project team that will implement and verify correct functioning of the system. This documents also included use case diagram and scenarios, objectives, architectural design and Entity-Relationship Diagram (ERD) of the admin dashboard.

**1.1 Project Description**

This project basically to build as system which will help other company to generate directors report of the audit profit and loss account, and balance sheet. This system will require secretary to upload raw data from like ledger and profit and loss payment. This data will process by accountant and then will re-check by auditor. From all data will be process and the output will be the final report and can be view and print out by the company’s director.

**1.2 Objective**

This module is built for the use of administrator of the accounting system. Below are the few objectives:

1. Admin should be able to view his/her information.
2. Admin should be able to edit and update his/her information.
3. Admin should be able to create users which are the accountants, auditors, directors and secretaries.
4. Admin should be able to view users’ information.
5. Admin should be able to delete users from the accounting system.
6. Admin should be able to update users’ privileges.

**1.3 Scope**

This admin dashboard applicable to the administrator of the accounting system. This module focuses on helping an administrator to monitor and keep track on the users. The users here are the accountants, auditor, directors and company secretaries. The admin is able to view, delete and manage users’ privileges in the system.

**2. REQUIREMENTS**

**2.0 Functional Requirements**

* Admin shall able view his/her profile
* Admin shall able edit/update his/her profile
* Admin shall able create users which are accountants, auditors, directors, secretary, accountant`s delegate and auditor`s delegate.
* Admin shall able to view user's profile.
* Admin shall able delete user from the system.
* Admin shall able update user’s privileges.
* Admin shall able to display dashboard status.
* Admin shall able to view the total record entry and total number of user.

**2.1 System Use Cases**

2.1.1 View Admin Profile (U1)

1. U1
2. Objective – To allow admin to view the admin profile
3. Priority – Medium
4. Source – None
5. Actors - Admin
6. Flow of Events

6.1 Basic Flow

6.1.1 Admin click on the admin profile button

6.1.2 The admin profile page displayed

1. Includes - None
2. Preconditions – None
3. Post conditions – Admin is able to view admin profile
4. Notes/Issues - None

## 2.1.2 Update Admin Profile (U2)

1. U2
2. Objective – To allow admin in updating admin profile
3. Priority – Medium
4. Source – None
5. Actors - Admin
6. Flow of Events

6.1 Basic Flow

6.1.1 Admin apply any changes regarding admin profile

6.1.2 Admin click update button to save the changes

1. Includes - None
2. Preconditions – None
3. Post conditions – Any changes applied
4. Notes/Issues - None

## 2.1.3 Create User (U3)

1. U3
2. Objective – To allow admin to create user in the admin dashboard
3. Priority – Medium
4. Source – None
5. Actors - Admin
6. Flow of Events

6.1 Basic Flow

6.1.1 Admin field in the user’s details

6.1.2 Admin created the user by clicking create user button

1. Includes - None
2. Preconditions – Admin field in user’s details
3. Post conditions – User created
4. .Notes/Issues - None

## 2.1.4 View User Profile (U4)

1. U4
2. Objective – To allow admin to view created user’s profile
3. Priority – Medium
4. Source – None
5. Actors - Admin
6. Flow of Events

6.1 Basic Flow

6.1.1 Admin click on user’s profile

1. Includes - None
2. Preconditions – Admin is logged in
3. Post conditions – User’s profile is displayed
4. .Notes/Issues - None

## 

## 2.1.5 Delete User (U5)

1. U5
2. Objective – To allow admin to delete user from the admin dashboard
3. Priority – High
4. Source – None
5. Actors - Admin
6. Flow of Events

6.1 Basic Flow

6.1.1 Admin delete user from the system by clicking “delete” user button

6.1.2 User deleted

1. Includes - None
2. Preconditions – User existed
3. Post conditions – User deleted
4. Notes/Issues – None

## 2.1.6 Update User Privilege (U6)

1. U6
2. Objective – To allow admin to update user privilege
3. Priority – High
4. Source – None
5. Actors - Admin
6. Flow of Events

6.1 Basic Flow

6.1.1 Admin tick on the permission to allow privilege for user based on user’s role

1. Includes - None
2. Preconditions – User’s existed
3. Post conditions – User’s privilege updated
4. Notes/Issues - None

## 

## 2.1.7 Update Status (U7)

1. U7
2. Objective – To allow admin to display dashboard status
3. Priority – High
4. Source – None
5. Actors - Admin
6. Flow of Events

6.1 Basic Flow

6.1.1 Admin click on the system status button

6.1.2 Percentage of system status of the dashboard system displayed

1. Includes - None
2. Preconditions – Total number of users
3. Post conditions – System status displayed
4. Notes/Issues - None

## 2.1.8 View Total Record and Total User (U8)

1. U8
2. Objective – To allow admin to view total record and total user created
3. Priority – Medium
4. Source – None
5. Actors - Admin
6. Flow of Events

6.1 Basic Flow

6.1.1 Total number of user displayed in form of pie chart and table list.

1. Includes - None
2. Preconditions – Admin is logged in
3. Post conditions – Total record and total user is being displayed
4. Notes/Issues - None

|  |
| --- |
| WhatsApp Image 2017-01-13 at 12.51.17 AM.jpeg  Figure 1.1 Use Case Diagram for Administrator Dashboard |

Table 1.2: show the detail of the use case diagram for Administrator Dashboard from Figure 1.1.

|  |  |  |
| --- | --- | --- |
| 1. | View admin profile | Admin can view his/her own profile. |
| 2. | Update admin profile | Admin can edit or update his/her profile. |
| 3. | create user | Admin can add and create user which is accountant, auditor, accountant’s delegate, auditor’s delegate, company’s director and company secretary. |
| 4. | view user profile | Admin can view any user profile. |
| 5. | delete user | Admin can delete user. |
| 6. | update user privilege | Admin can give privilege to user based on their roles want. |
| 7. | view dashboard status | Admin can view dashboard status |
| 8. | view total record and total user | Admin can view the total of records entry and how many user that created. |

**2.2 Non-Functional Requirements**

1. **Performance (response time)**

This system shall generated the report within 5 working days after all data completed upload into this system.

1. **Performance (deadline)**

All uploading data must complete by 6pm so that accountant can start processing that data.

Uploading data must be performed before third week of the month.

Final report must be before end of the month

1. **Scalability (request load)**

This system shall able to handle increasing in request load grows in between 100-1000 request load without decreasing in performance.

**2.3 Activity Diagram**

|  |
| --- |
| ActivityDiagram.png  Figure 4.0 Activity Diagram for admin dashboard system |

**3. ARCHITECTURAL DESIGN**

**3.0 Model-View-Controller (MVC) pattern**

Laravel follows the model-view-controller (MVC) architectural pattern, which enforces a separation between “business logic” from the input and presentation logic associated with a graphical user interface (GUI). In the case of Laravel web applications, the business logic typically consists of data models for things like users, blog posts, and the GUI is just a web page in a web browser.

The 3 components of the MVC pattern :

The model - the domain that the software is built around. Models are based on real-world items such as a person, bank account or product. Models are typically permanent and will be stored outside the application, often in a database.

The view - visual representation of a model, given some context. The view layer is responsible for generating a user interface, normally based on data in the model. The view also provides an interface to collect user input.

The controller - controls the flow of information between the model and the view. Using programmed logic to decide what information is pulled from the database via the model and what information is passed to the view and implements business logic. Either by changing the view, or modifying data through the model, or both.

**3.1 Layered Architectural Style**

Layered architecture also used to developing this system. The first layer is user interface of the Administrator Dashboard. Second layer of this system is logic layer which coordinates the system, processes commands, makes logical decisions and evaluations and performs calculation. It also moves data between the two surrounding layers. The third layer is the storage layer. Data is stored and retrieved from the cloud database. The data is then passed back to the logic layer for processing and eventually back to the user. The code is arranged so the data enters the top layer and works its way down each layer until it reaches the bottom, which is database.

|  |
| --- |
| Figure 3.1 Layered Architecture style |

* **Abstraction**. Layers allow changes to be made at the abstract level. You can increase or decrease the level of abstraction you use in each layer of the hierarchical stack.
* **Isolation**. Allows you to isolate technology upgrades to individual layers in order to reduce risk and minimize impact on the overall system.
* **Manageability**. Separation of core concerns helps to identify dependencies, and organizes the code into more manageable sections.
* **Performance**. Distributing the layers over multiple physical tiers can improve scalability, fault tolerance, and performance.
* **Reusability**. Roles promote reusability. For example, in MVC, the Controller can often be reused with other compatible Views in order to provide a role specific or a user-customized view on the same data and functionality.
* **Testability**. Increased testability arises from having well-defined layer interfaces, as well as the ability to switch between different implementations of the layer interfaces. Separated Presentation patterns allow you to build mock objects that mimic the behavior of concrete objects such as the Model, Controller, or View during testing.

**3.2 Component-Based Architectural Style**

For overall system, we implemented component-based architecture style where the system decomposition into six logical components. By using this type of style will help well-defined communication interface that contain methods, events and properties. This provides a higher level of abstraction than object-oriented design principle and does not focus on issues such as communication protocol and shared state.

Benefits of component-based style :

* **Ease of deployment**. As new compatible versions become available, you can replace existing versions with no impact on the other components or the system as a whole.
* **Reduced cost**. The use of third-party components allows you to spread the cost of development and maintenance.
* **Ease of development**. Components implement well-known interfaces to provide defined functionality, allowing development without impacting other parts of the system.
* **Reusable**. The use of reusable components means that they can be used to spread the development and maintenance cost across several applications or systems.
* **Mitigation of technical complexity**. Components mitigate complexity through the use of a component container and its services. Example component services include component activation, lifetime management, method queuing, eventing, and transactions.

|  |
| --- |
| WhatsApp Image 2017-01-13 at 12.10.35 AM.jpeg  Figure 3.0 Component Diagram: Administrator Dashboard |

4. ENTITY RELATIONSHIP DIAGRAM

|  |
| --- |
| last erd.png  Figure 4.0 Entity-Relationship Diagram |

5. USER INTERFACE DESIGNS

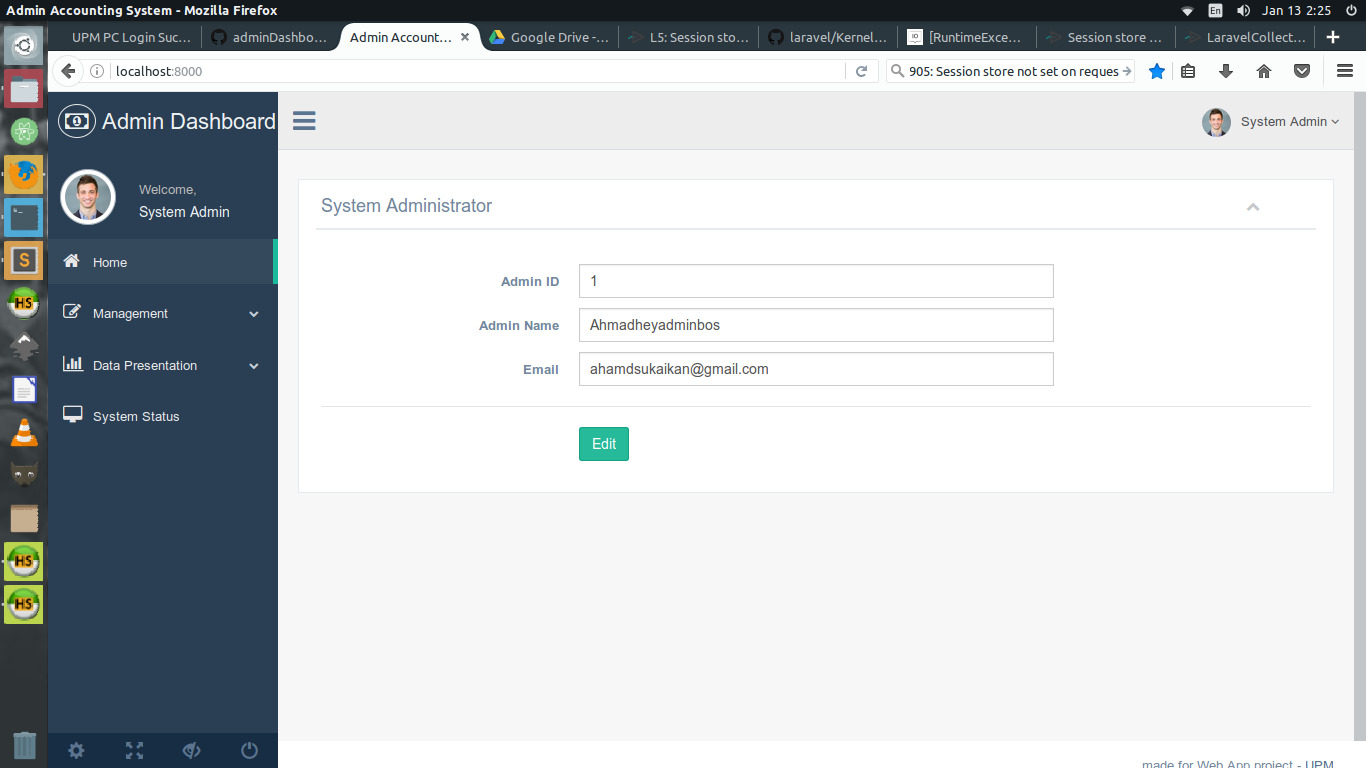


Figure 5.0

This home page is where the admin profile displayed. Admin can update their profile by apply any changes in the field and click the “Edit” button to save the changes applied.

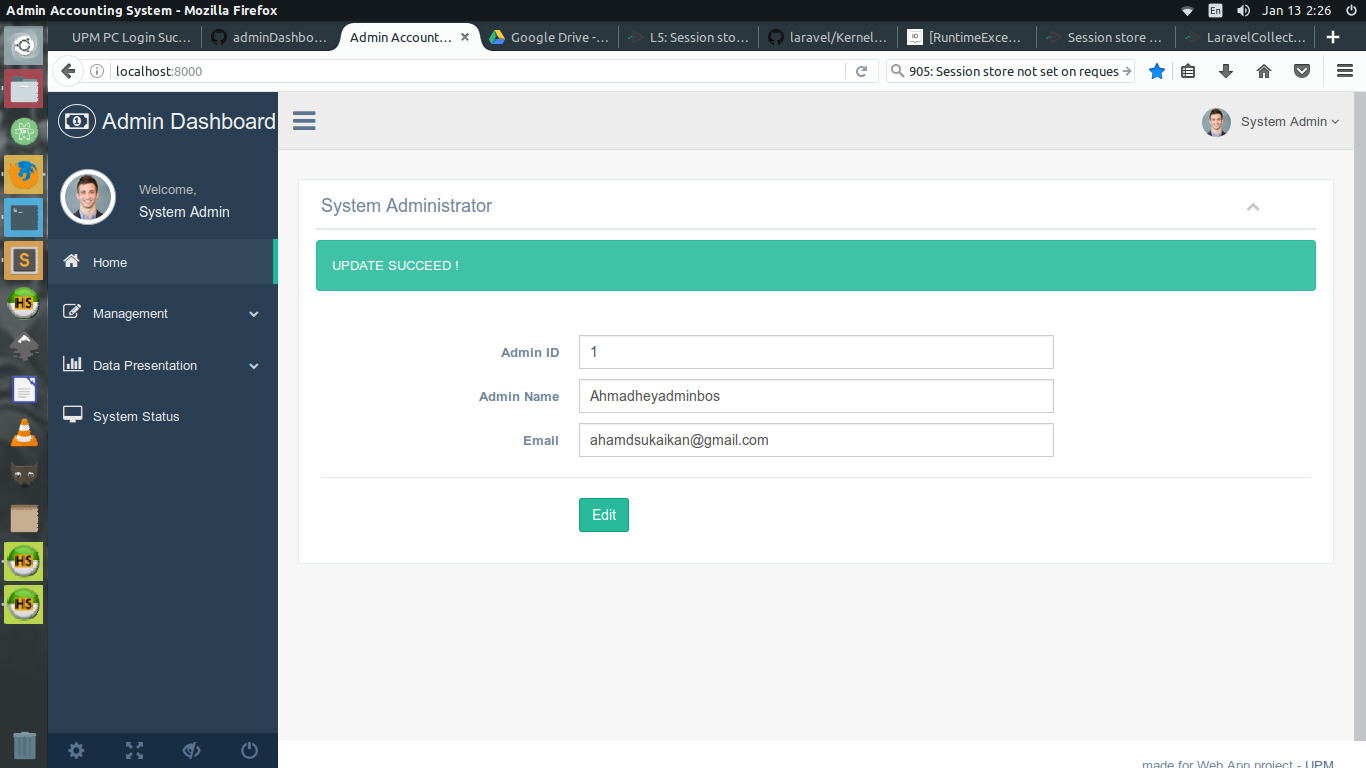


Figure 5.1

Once “Edit” button clicked, the applied changes is updated and “UPDATE SUCCEED!” displayed.

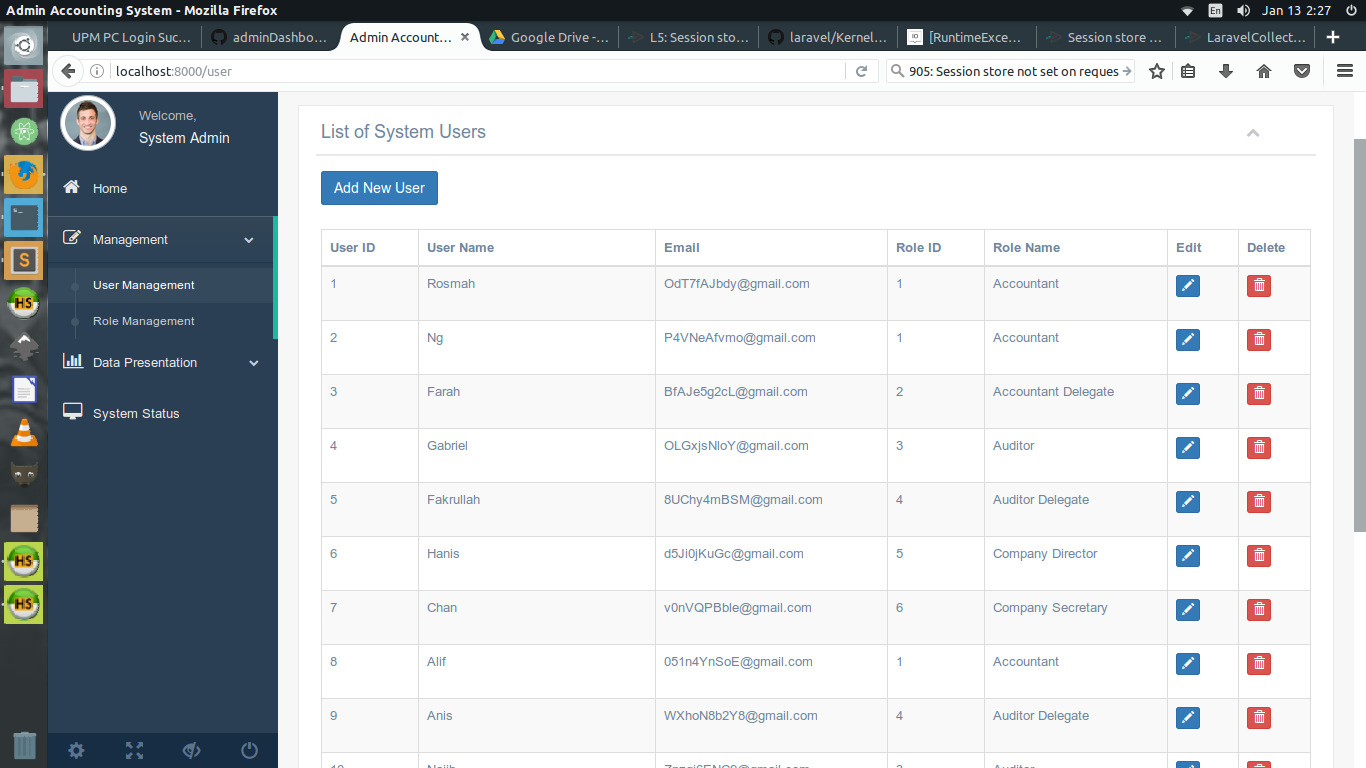


Figure 5.2

This is Management page that display the list of system users details and admin can edit the details and delete the user. There also “Add New User” button to allow admin adding new user to the system.

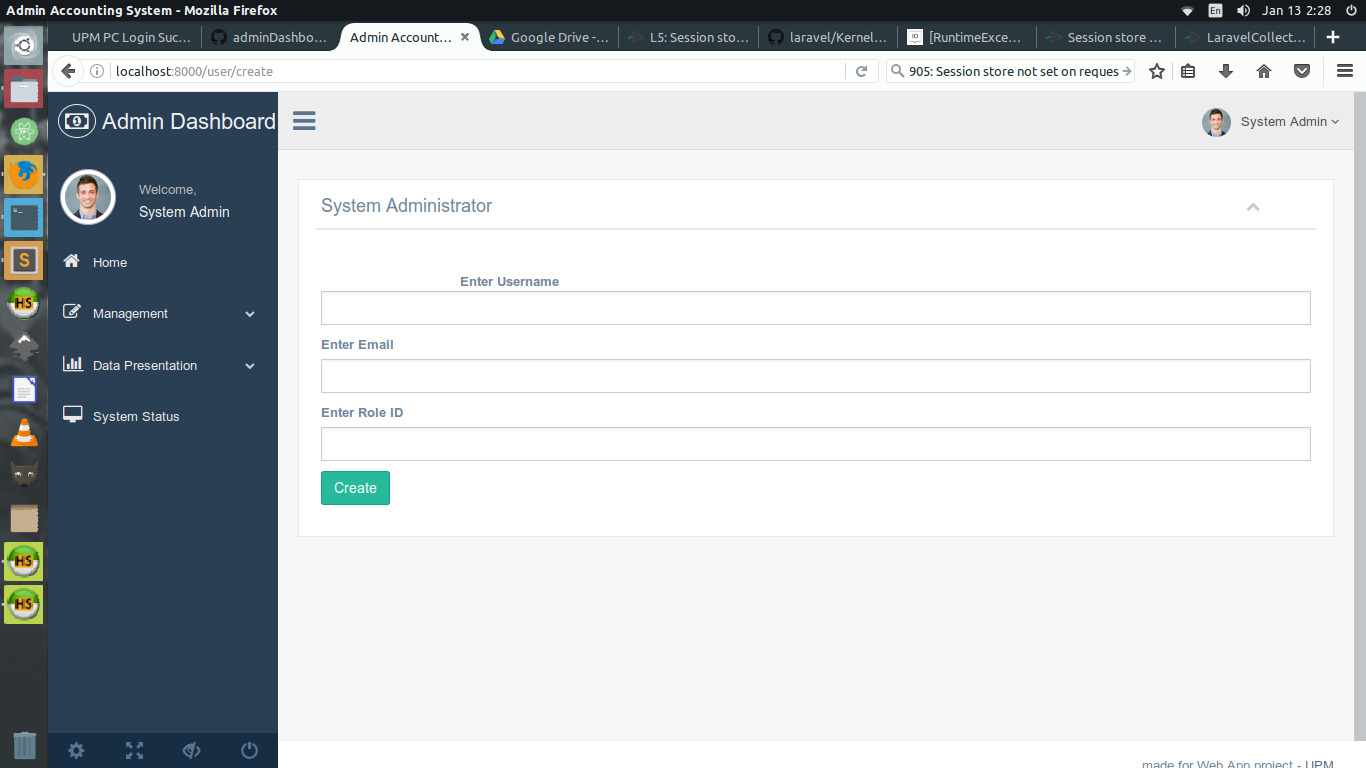


Figure 5.3

This is the create new user interface for admin to field in the user’s details to enable the information being retrieve as for assigning privilege for the user based on the role ID.

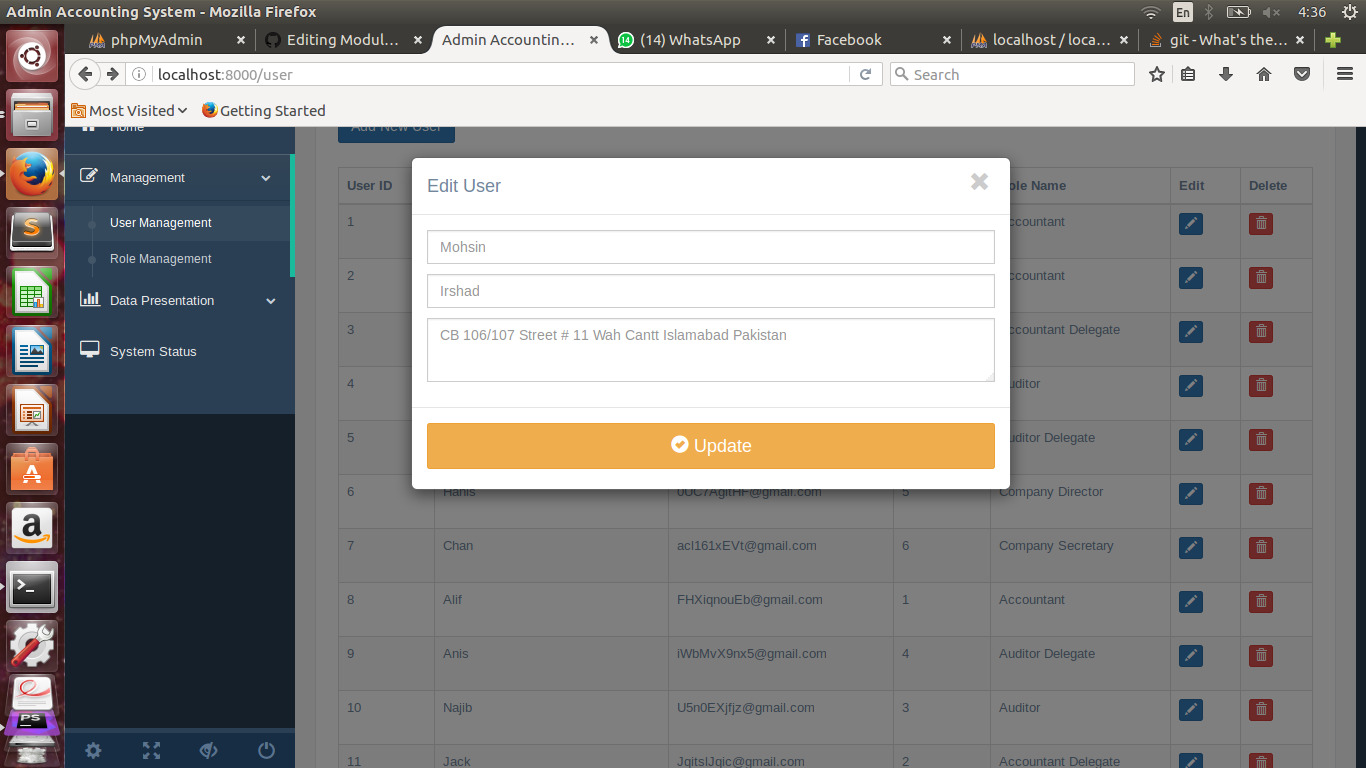


Figure 5.4

This is interface appeared when edit button clicked. Admin can edit user’s details and update to save the changes made.

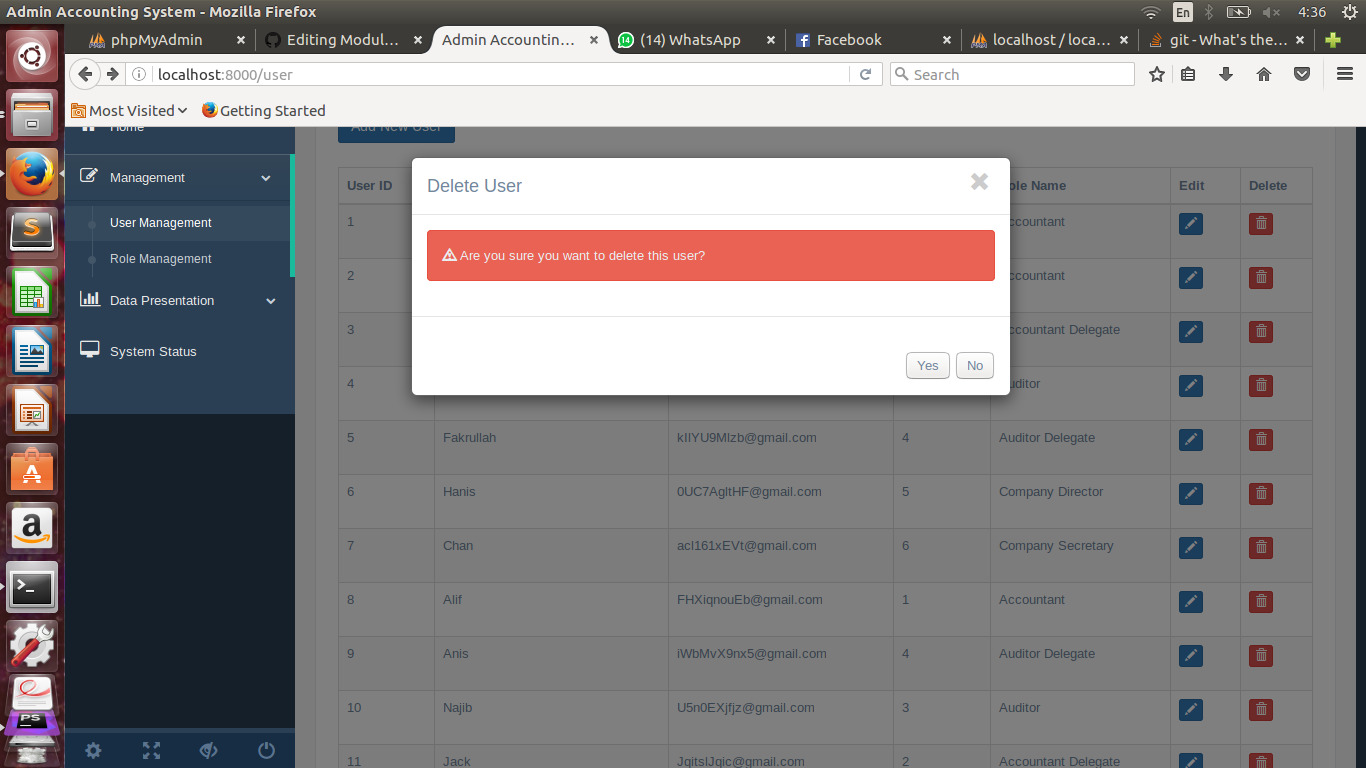


Figure 5.5

This is the delete user interface conformation that appeared when the delete button clicked.

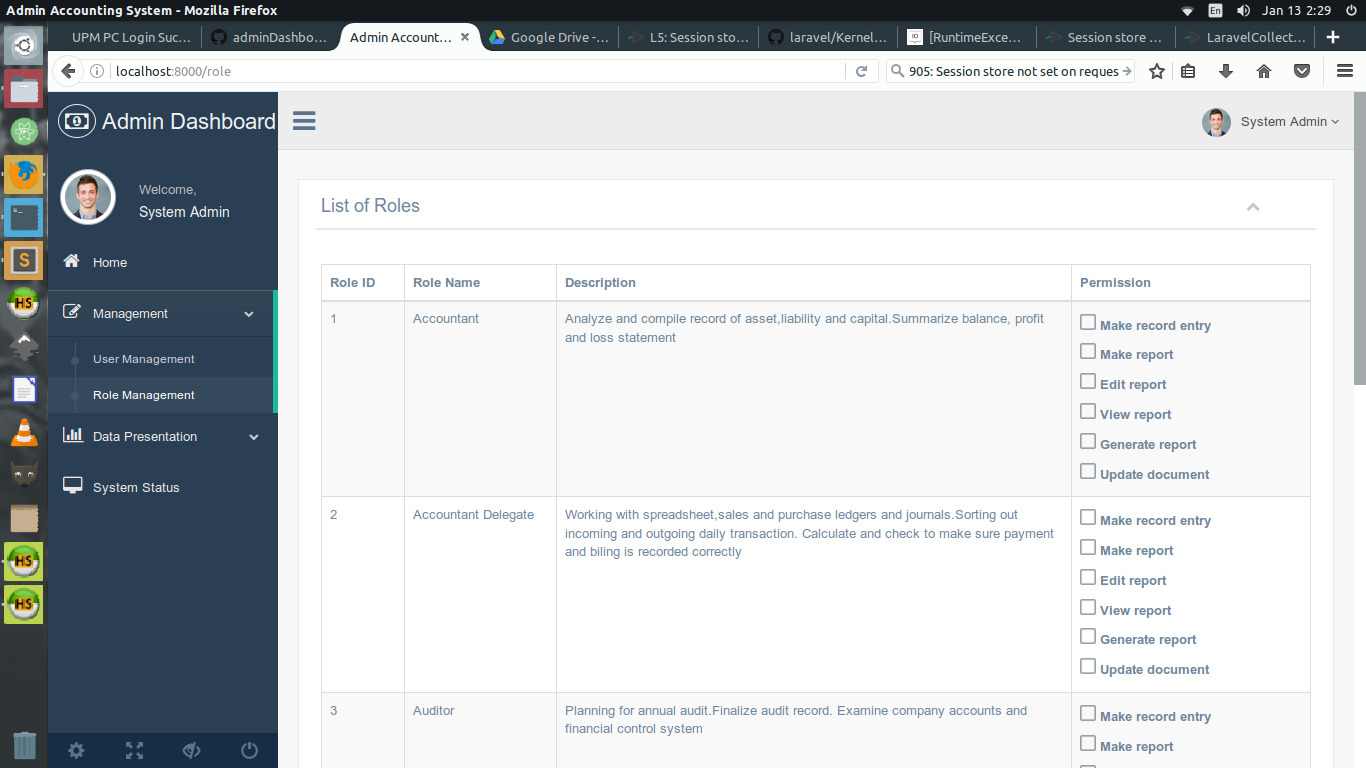


Figure 5.6

This is Management page which display the list of role of the users and the privileges they have according to their roles id.

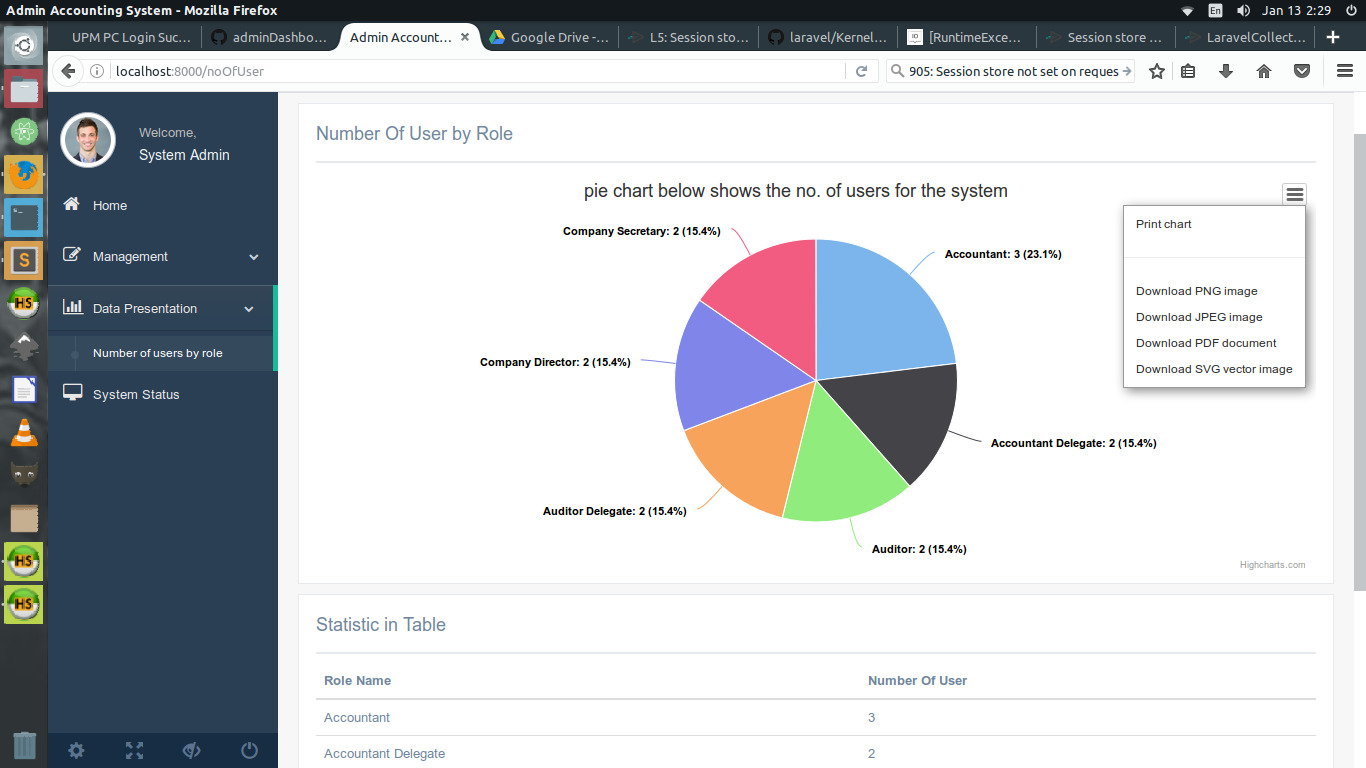


Figure 5.7

This is Data Presentation page for the admin to see total number of user by role for the system by pie chart presentation. Admin is able to print or download the pie chart.

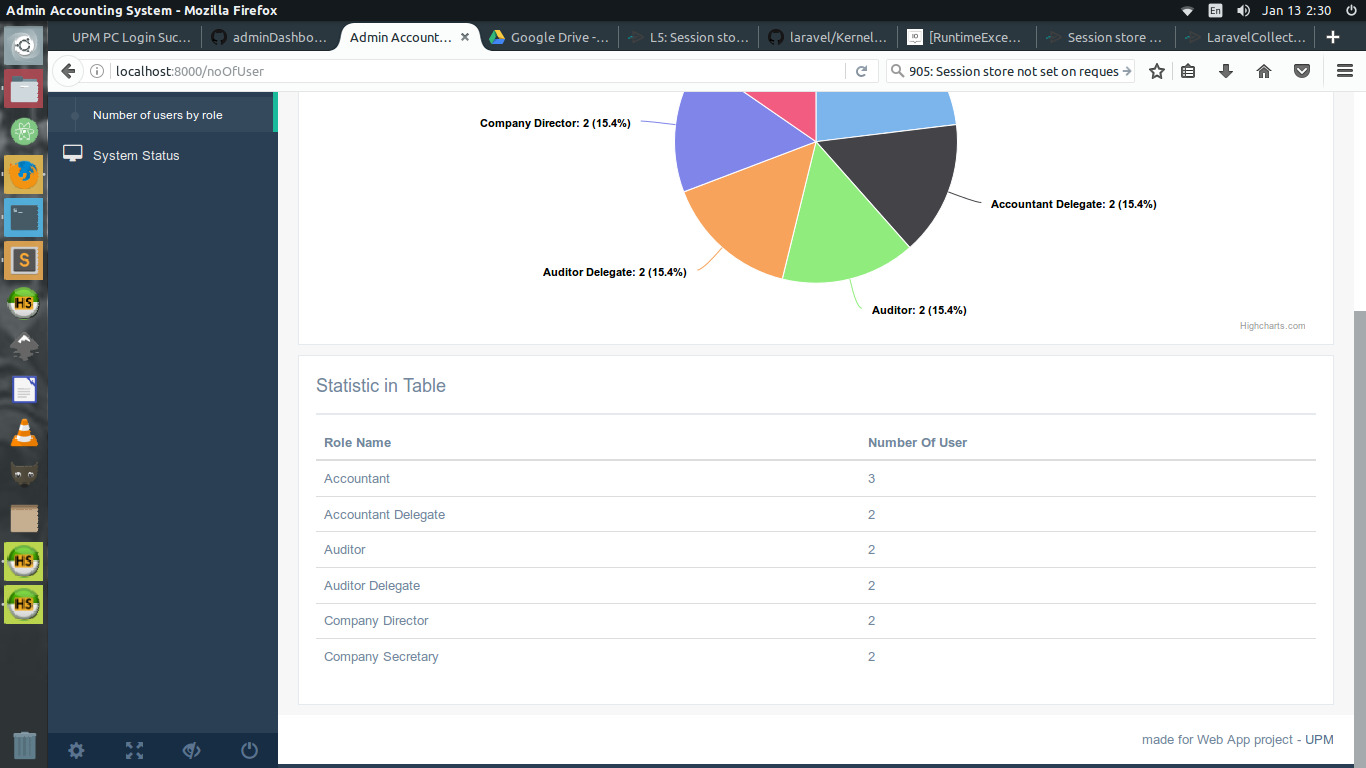


Figure 5.8

This is Data Presentation page that display the total number of user in table presentation.

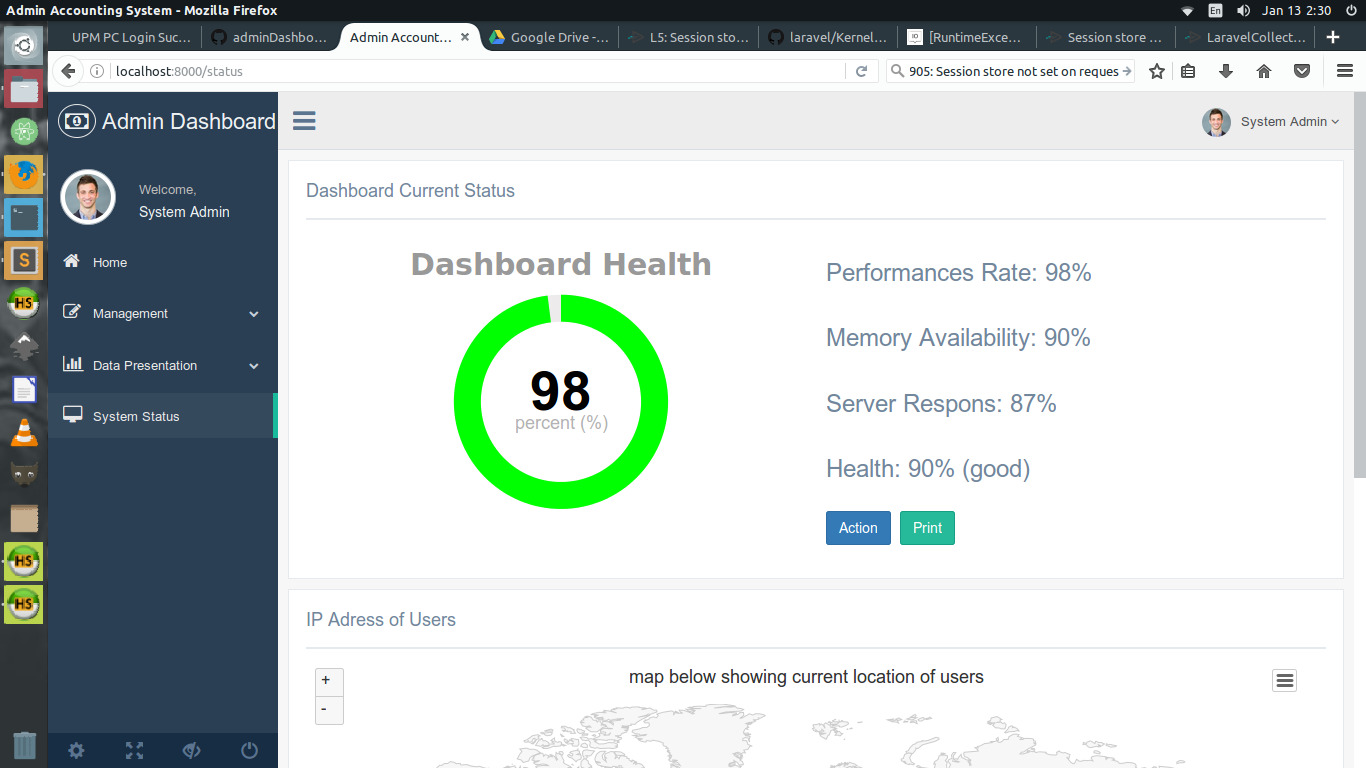


Figure 5.9

This is System Status page that display the dashboard system status. The “Action” button for the admin to conduct maintenance task for the system and admin can print the dashboard status as report.

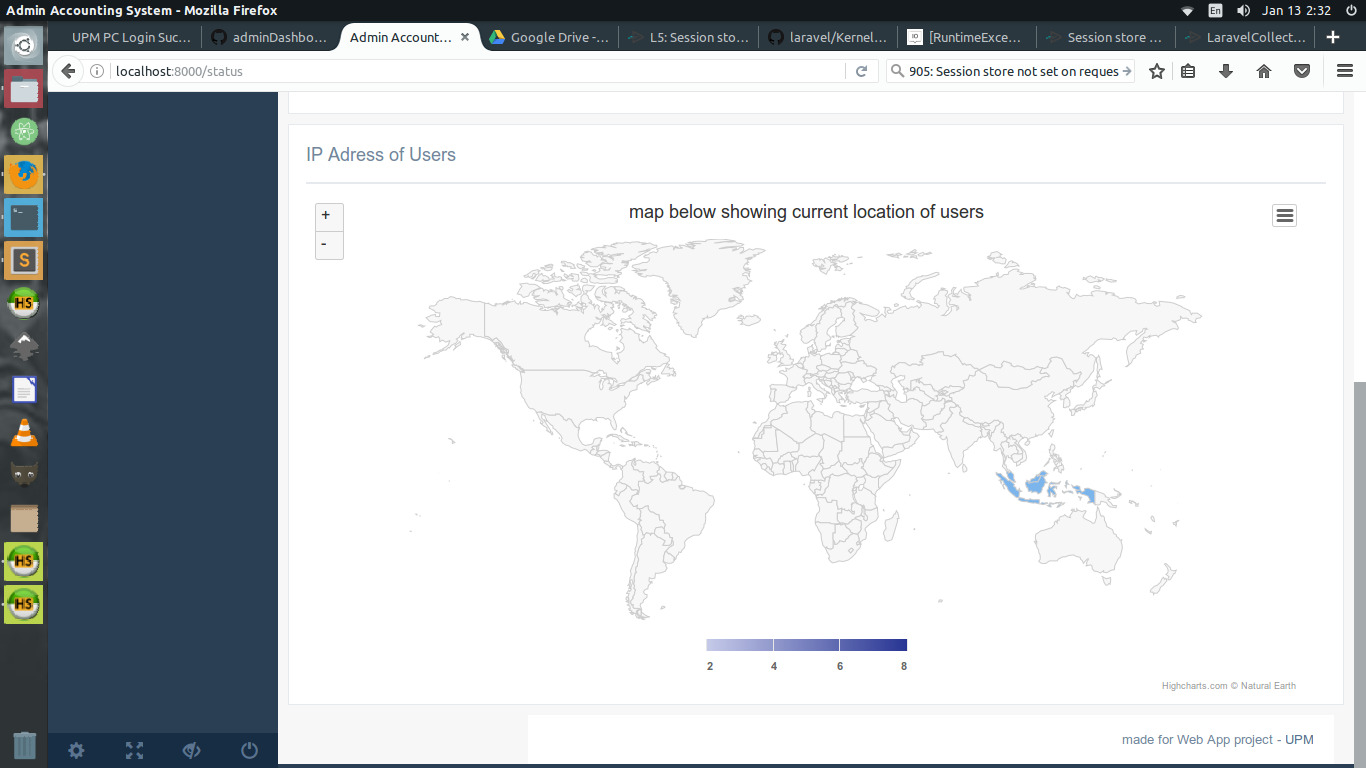


Figure 5.10

This is Data Presentation showing the current location of the user by IP address.

6. ABBREVIATIONS AND DEFINITIONS

**Abbreviations : IP** - Internet Protocol

**Definitions: Functional Requirement -** Describes required behavior in terms of required activities

**Non-Functional Requirement -** Describes some quality characteristic and constraints that the software must posses

7. REFERENCES

Laracasts website:

<https://laracasts.com/series/laravel-5-fundamentals>

Laravel :

https://laravel.com/docs/5.2/quickstart