**List of KW Assignments**

1. **Create Reports Tab**
   1. In **resources/views/layouts/admin/sidemenu.blade.php**, created a **Reports** tab with a **User Status** tab underneath
   2. Underneath the **User Status** tab, created a route that goes to **pie-chart.blade.php**
   3. In **routes/web\_admin.php**, created a new group called **admin/reports** that only admin users can have access to
      1. Created a pie-chart route and make it return **pie-chart.blade.php**
2. **ECharts Pie Chart**
   1. The **pie-chart.blade.php** is located in **kw\_middleware/resources/views/admin/setting/user**
      1. Contains the html code of the pie-chart api.
      2. Created a button called Refresh that will call the javascript **refresh()** function
      3. Imported **echarts.min.js** which is located in the **public/vendor/echart** folder
      4. Imported **setting-pie-list.js** which is located in **public/js/views/admin** folder
      5. Set hidden id of ***“hd\_define”*** and data-refresh-link of ***“/api/v1/reports/refresh”***
   2. The **setting-pie-list.js** is the javascript that makes the pie chart on the **pie-chart.blade.php**
      1. Created function called **refresh()** that makes an ajax call to the url of “hd\_define data-refresh-link” (located in the **pie-chart.blade.php**)
      2. In the success function of the ajax call, get the json data from the previous url and input the data into the options of the echart
      3. Any time the **Refresh** button is pressed, the **refresh()** function is called which makes the pie-chart refresh.
3. **API Restful Implementation.**
   1. In **routes/api\_v1.php** we create a new group of v1/reports and route it to **Restful\UserController@refresh**
   2. In **app/Http/Controllers/Restful/UserController.php**, we create a function called **refresh()** that uses a new service specifically for reports called **app/Http/Service/ReportService.php**
   3. We create **ReportService.php** and **ReportServiceImpl.php** to send the json code of the number of activated,non-activated, and deleted users to the ajax call.
4. **Fix the side menu’s syntax (resources/views/layouts/admin/sidemenu.blade.php)**
   1. The sidemenu incorrectly displays the names of the tabs and have the hierarchy of the tabs jumbled up.
   2. To fix this we removed the additional span tags as well as nav-item tags that have made the display work incorrectly
5. **Fix the side menu’s saved state of being closed/open**
   1. Go to **resources/views/layouts/admin/app.blade.php**
   2. To save the toggle state, we create a ***“toggle-Nav-Bar”*** id in the page wrapper
   3. When ***“toggle-Nav-Bar”*** is clicked, we need to make an ajax function that can save the state of the toggle in session.
   4. Set hidden id of ***“side”*** and data-show-link of ***“/admin/savedNavBarState”***
   5. We create the ***savedNavBarState()*** function with an ajax call to the url of “side data-show-link” (located in the itself)
   6. ***savedNavBarState()*** function sends the state of whether the toggle button has a class called mini-navbar to the url
   7. Back in **routes/web\_admin** make a post route of ***/admin/savedNavBarState*** to **savedNavBarState** in **app/Http/Controllers/AdminController.php**
   8. The Session boolean is saved into the **‘show’** key of the **savedNavBarState** Function
   9. Back in **resources/views/layouts/admin/app.blade.php**, inside the body class we put an if statement of whether the Session key of **‘show’** is true.
      1. If it's true, add **mini-navbar** into the class
      2. If it's false do nothing
6. **Create a custom column visibility drop down bar (*still in progress*)**
   1. Create a select drop down box html code, with the options of the datatables' column names, in **resources/views/admin/setting/user/list.blade.php**
   2. Create a multiselect function in **public/js/views/admin/setting-user-list.js** with filtering and select all options.