

High level design specification

EMS-PSS



By:

Ben Lorantfy

Grigory Kozyrev

Kevin Li

Michael Da Silva

March 13, 2015

Dev till Death

Contents

[1. Introduction 3](#_Toc414032785)

[1.1. Description 3](#_Toc414032786)

[1.2. Definitions 3](#_Toc414032787)

[2. Functions 4](#_Toc414032788)

[2.1. Login 4](#_Toc414032789)

[2.1.1. Overview 4](#_Toc414032790)

[2.1.2. User Interface 4](#_Toc414032791)

[2.1.3. Business Logic 4](#_Toc414032792)

[2.2. Search 5](#_Toc414032793)

[2.2.1. Overview 5](#_Toc414032794)

[2.2.2. User Interface 5](#_Toc414032795)

[2.2.3. Business Logic 5](#_Toc414032796)

[2.3. View Employees 6](#_Toc414032797)

[2.3.1. Overview 6](#_Toc414032798)

[2.3.2. User Interface 6](#_Toc414032799)

[2.3.3. Business Logic 6](#_Toc414032800)

[2.4. Add Employee 7](#_Toc414032801)

[2.4.1. Overview 7](#_Toc414032802)

[2.4.2. User Interface 7](#_Toc414032803)

[2.4.3. Business Logic 7](#_Toc414032804)

[2.5. Edit Employee 8](#_Toc414032805)

[2.5.1. Overview 8](#_Toc414032806)

[2.5.2. User Interface 8](#_Toc414032807)

[2.5.3. Business Logic 8](#_Toc414032808)

[2.6. Timecard Entry 9](#_Toc414032809)

[2.6.1. Overview 9](#_Toc414032810)

[2.6.2. User Interface 9](#_Toc414032811)

[2.6.3. Business Logic 9](#_Toc414032812)

[2.7. Reports 10](#_Toc414032813)

[2.7.1. Overview 10](#_Toc414032814)

[2.7.2. User Interface 10](#_Toc414032815)

[2.7.3. Business Logic 10](#_Toc414032816)

[2.8. Audit Table 11](#_Toc414032817)

[2.8.1. Overview 11](#_Toc414032818)

[2.8.2. User Interface 11](#_Toc414032819)

[2.8.3. Business Logic 11](#_Toc414032820)

[2.9. Users 12](#_Toc414032821)

[2.9.1. Overview 12](#_Toc414032822)

[2.9.2. User Interface 12](#_Toc414032823)

[2.9.3. Business Logic 12](#_Toc414032824)

[3. Technological Considerations 13](#_Toc414032825)

[4. Appendix A 14](#_Toc414032826)

[4.1. General User 14](#_Toc414032827)

[4.2. Administrative User 15](#_Toc414032828)

# Introduction

## Description

This document is designed to be a High-level design specification to outline the EMS-PSS system in-order to support early phases in its design. The goal of the EMS-PSS system is re-develop the Employee Management System (EMS) into a web application. The EMS-PSS system will be created in PHP with a MySQL database. The EMS-PSS will fill the entire requirement the EMS filled, but with an improved GUI and database management.

This document includes specifications for GUI, database, employee adding/editing business logic, timecard business logic and reports business logic.

## Definitions

* GUI: Short for Graphic User Interface is a interface that allows a user to interact with electronic devices or software via graphical icons, buttons, drop lists, etc. in order to simplify the use of the application.
* HTML: Hypertext Markup Language, most basic language used to construct web pages.
* JavaScript: Scripting Language used to create effects on a web page that the user can interacts with.
* JQuery: Library used to give additional support to JavaScript allowing effects that are more complex.
* CSS: Cascading Style Sheets is a language used to alter the appearance of a web page.
* Ajax: Asynchronous JavaScript and XML, Multiple technologies used in combination to create an interactive web page that is capable of sending requests to a server.
* Hard Coded: Entered not through the system, but programmatically.

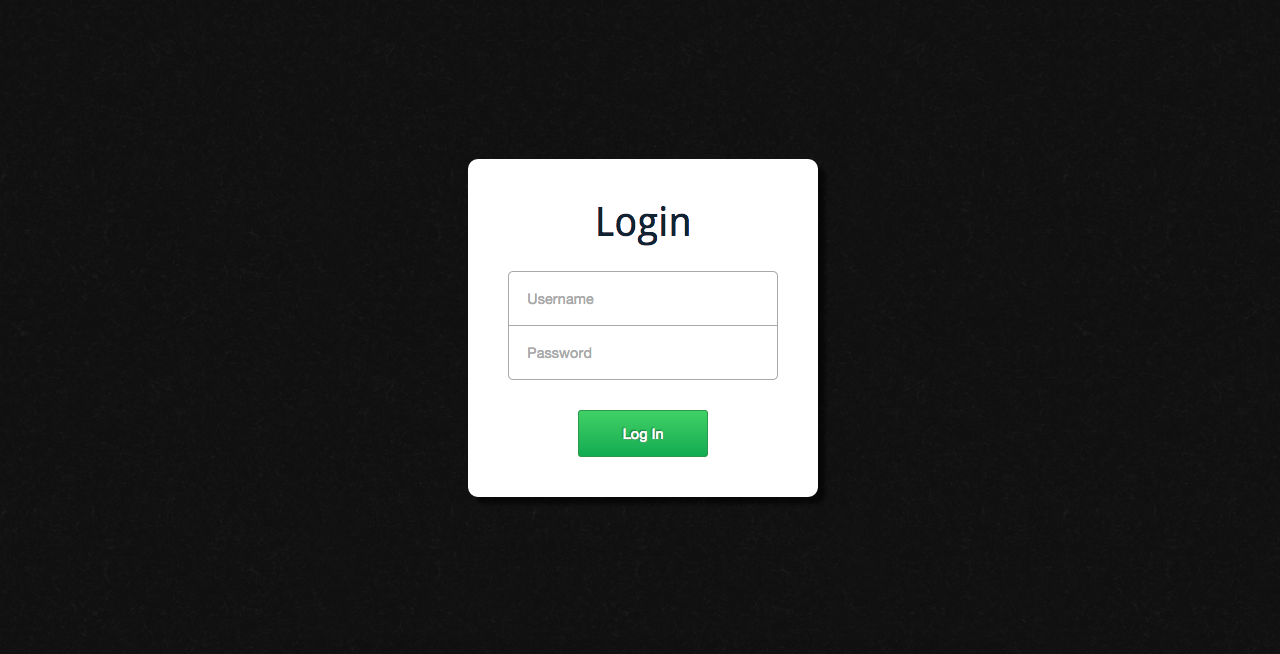
# Functions

## Login

### Overview

The login section will contain a login form with a username and password field. There will not be an option to sign up because the system is not a publically exposed service. Instead, an administrator account will be able to create other user accounts. The first administrator account may have to be hard coded into the database.

### User Interface



### Business Logic

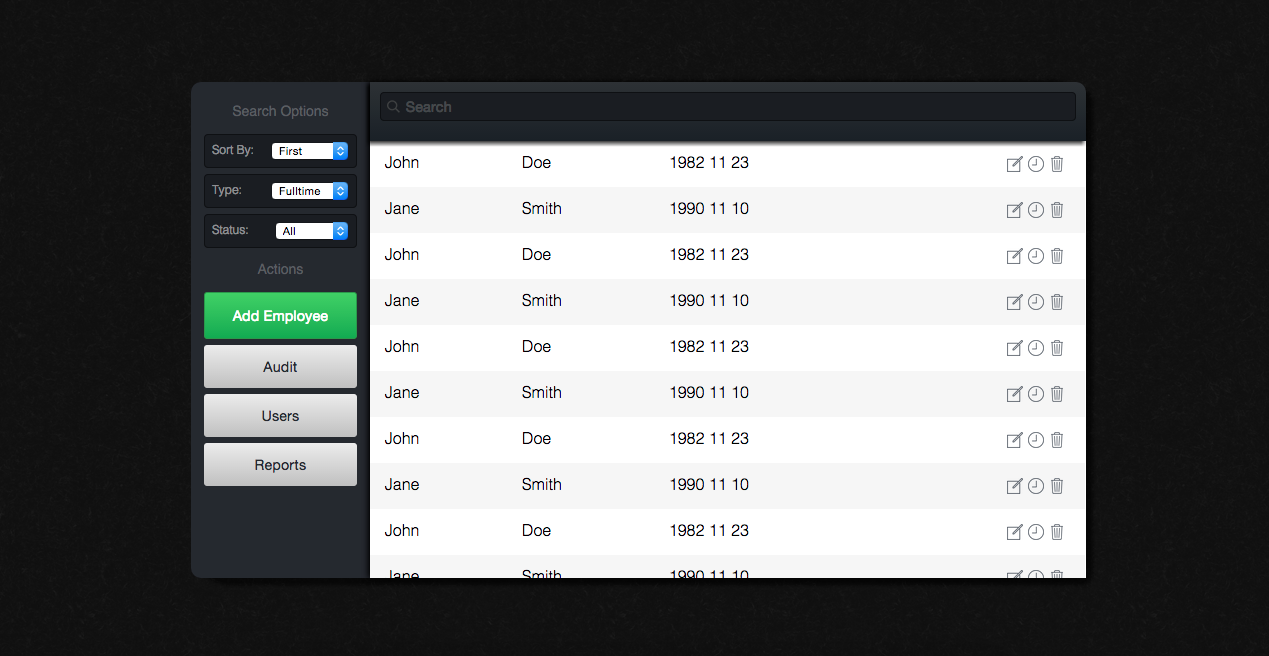
* Username should be unique in database
* Password will be converted to harsh value and save to database

## Search

### Overview

This functionality allows users of the system to search for specific records in the database. Different search criteria are provided. All records fitting those criteria will be displayed. Then, needed action can be performed by clicking on tool buttons on the right.

### User Interface



The search section will be the first section presented to the user after logging in. It is divided into 3 subsections: search box, search results, search options. If the search box is empty, all records are displayed, with some form of pagination. When the user enters characters into the search box, the system instantly updates the displayed results to be filtered by the text query. It does this in the background without interrupting user input. The user can also further affect the results using the search options in the left pane. The search options allow the user to sort the results based on various criteria, limit the results to a specific employee type, and perform other potential filtering.

When the user finds the employee they are looking for, there are several options they can select. If the user simply clicks the row, they will navigate to a page that simply displays the employee’s information.

Since the search section also serves as a so-called home section, there are also buttons to navigate to other related pages, such as results and users.

### Business Logic

Only main fields’ needs to be displayed: First and last name, Date of Birth.

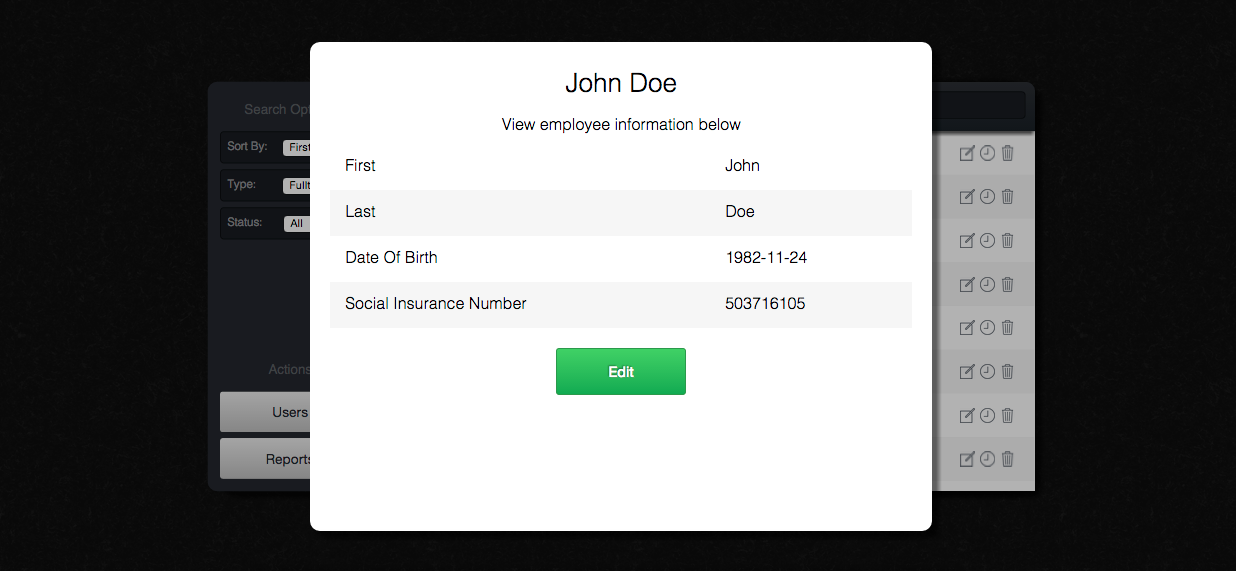
Search fields, types and status are specified in dropdown list.

## View Employees

### Overview

Users of the system are able to view all details about each completed employee. This functionality provides users with basic way of getting information form the system. User can view full details about each employee by clicking on specific button on his/her field in the search table.

### User Interface



The user will be able to access this page through the main page. This page will allow the user to view employee information; the displayed information is dependent on the user’s level of access that is determined by the user permissions. 

The employee-viewing page will show every employee in the database with the option of a search function to limit the number of results displayed.

Each employee displayed will only show a set amount of attributes, the user must click the view more option to see all the attributes related to that employee. This will be shown in a popup window in front of the previous window.

### Business Logic

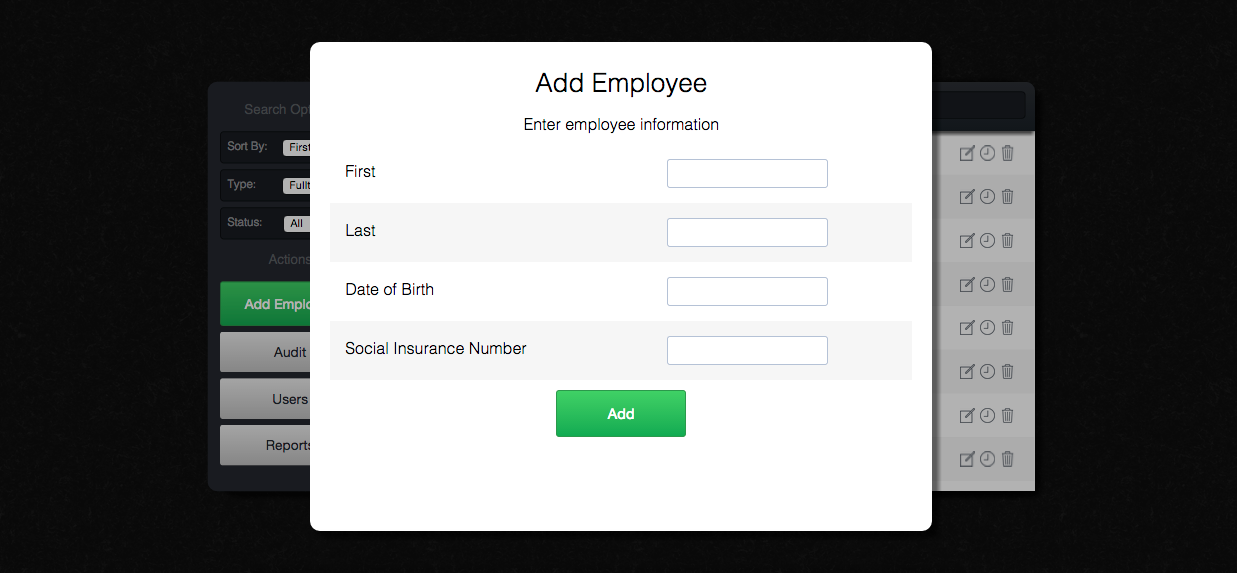
If field is empty, “unspecified” text should be displayed.

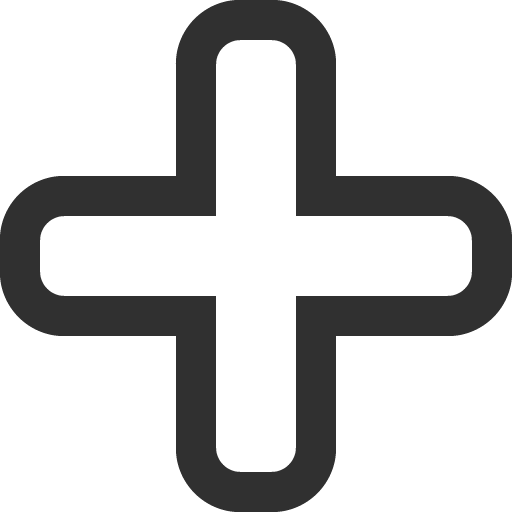
## Add Employee

### Overview

Users of the system are able to add new employee record to the database. Adding logic and interface is different, depending on a user permission level.

### User Interface



The user will be able to access the adding page via a button that lives on the viewing page. This page will act similar to the single employee view. The window will pop up in front of the view page with all the fields blank.

The user will be able to fill out the forms and submit when finished. Once submitted the new employee will be added to the database provided no errors occurred. If any field is invalid, upon submission, a flagged will be trigger and an error message will be displayed next to the invalid field

### Business Logic

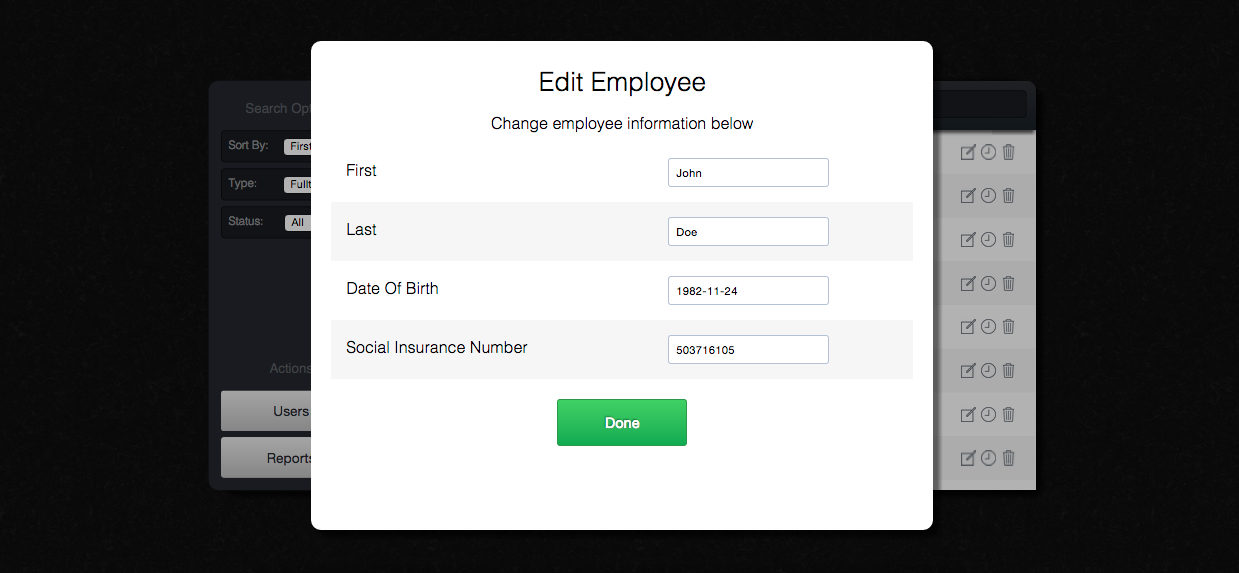
All necessary fields must be entered.

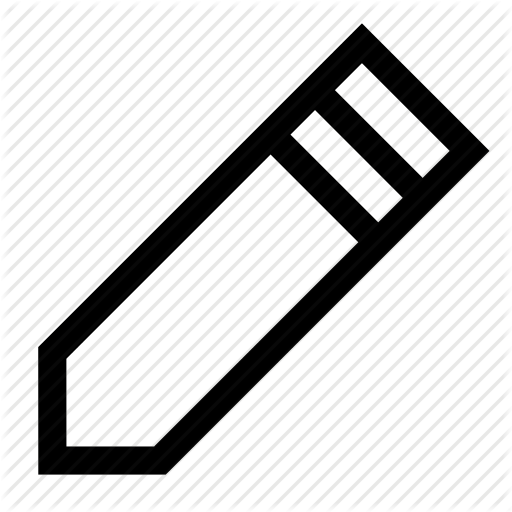
## Edit Employee

### Overview

Users of the system are able to edit already existing record in the system. Edit field are different depending on the user permission level.

### User Interface



The user will be able to edit employee information from the employee view. This will have to be unlocked by the user by pressing the pencil icon next to the employee’s name. This will bring up the editing popup. This page will contain all the fields filled with the employee's information and allow the user to edit it as desired. 

Once complete the user can submit and the information will replace the employees old information provided no errors occurred. If an error were to occur, a flag would be triggered and an error message will be displayed beside the invalided field.

### Business Logic

Initial value of the field is its current value.

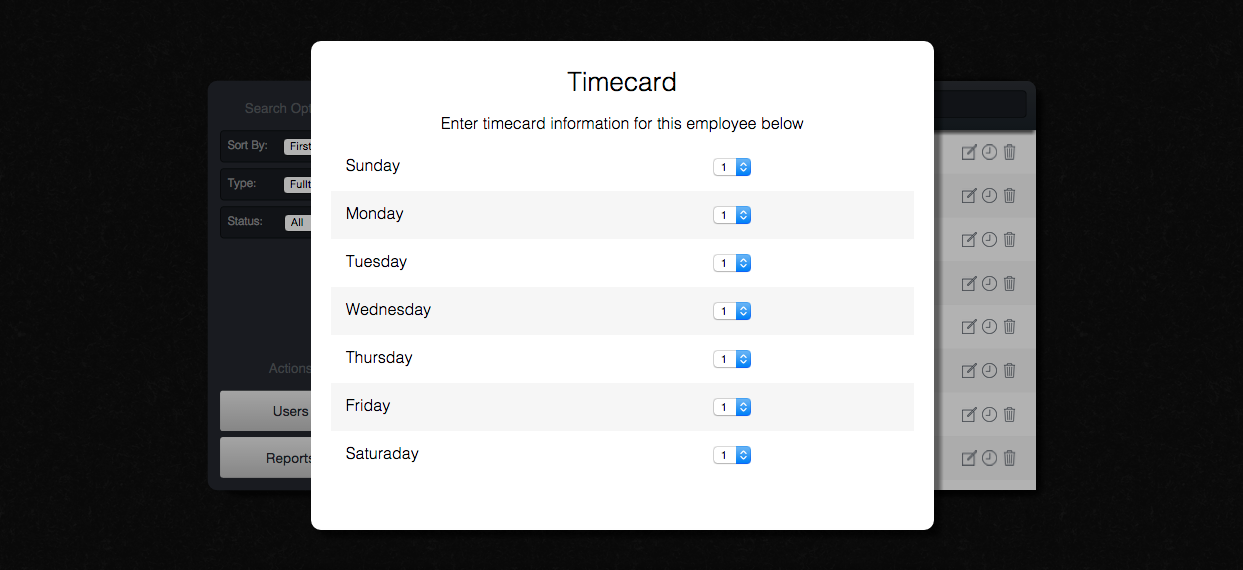
If field is empty, initial field is empty.

## Timecard Entry

### Overview

Users of the system are able to edit timecards for each employee, which allows setting number of hours worked and number of pieces per week.

### User Interface



General User can click the Clock Icon to enter this section after they login the system. When the user gets into this Timecard section, they will be able to do time-card entry for fullTime, partTime and seasonal employees on a daily basis. Timecard section does not apply to the contract employees.

This time-card entry has the information (total hours of the day, pieces accomplished) for each day of the month. Also there is a status column that show detail status (total work hours, work status, etc.) of the week.

User can simply change the information by selecting the option in the dropdown menu. To change the status, they need to click the text area and input the information. All these information will be store in the database as user completed, and will be used to generate the Payroll Report for current pay period.

### Business Logic

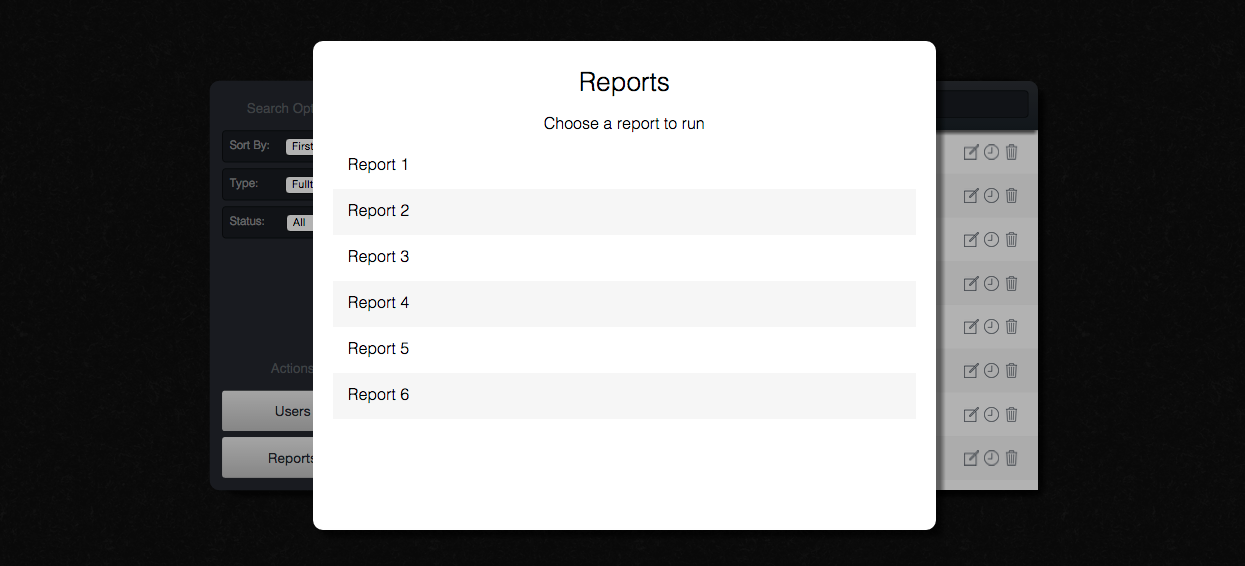
Only allow user to select the day in the dropdown menu, so no need to do validation

## Reports

### Overview

Users of the system are able to generate different reports, to get specific information in comfortable way. Types of report to run are depends on user’s permission level.

### User Interface



This section will allow Administrative users and General users to generate different reports according to the employee information (payroll, seniority, employment). General user will be able to run the Seniority Report and Weekly Hours Worked Report, and Administrative users can run all of the same reports as General users as well as Payroll Report, Active Employment Report, and Inactive Employment Report. All the data shown on the report will be obtain from database in real time.

User will be able to select the report they want to run on the list of possible reports. When the cursor is moved to the report, it will show the general description about the report, telling user what will be shown on the report. After selecting the report, it will be display on the web, user can print it out, or save it to local as PDF file.

### Business Logic

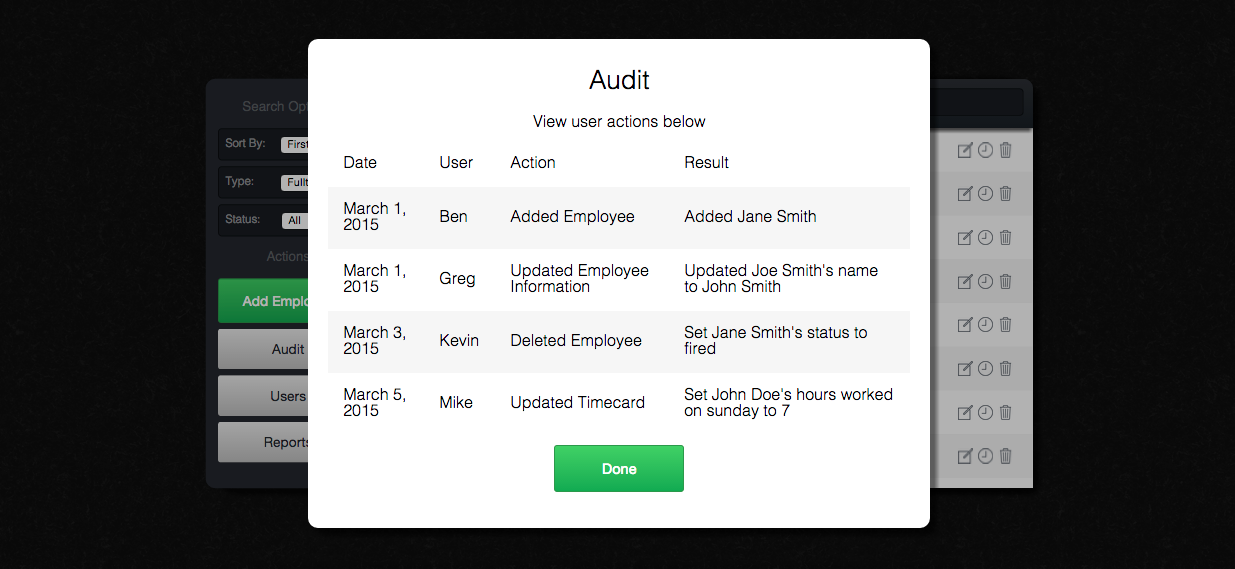
Reports will be created with print-friendly formatting.

## Audit Table

### Overview

Users of the system are able to view all changes being made in the database. Function will be provided depending on user’s permission level.

### User Interface



This section will allow the user to browse the audit table, which displays the histories of user’s actions. The audit table will contain the Date, User, Action and Result of every changes has been made.

### Business Logic

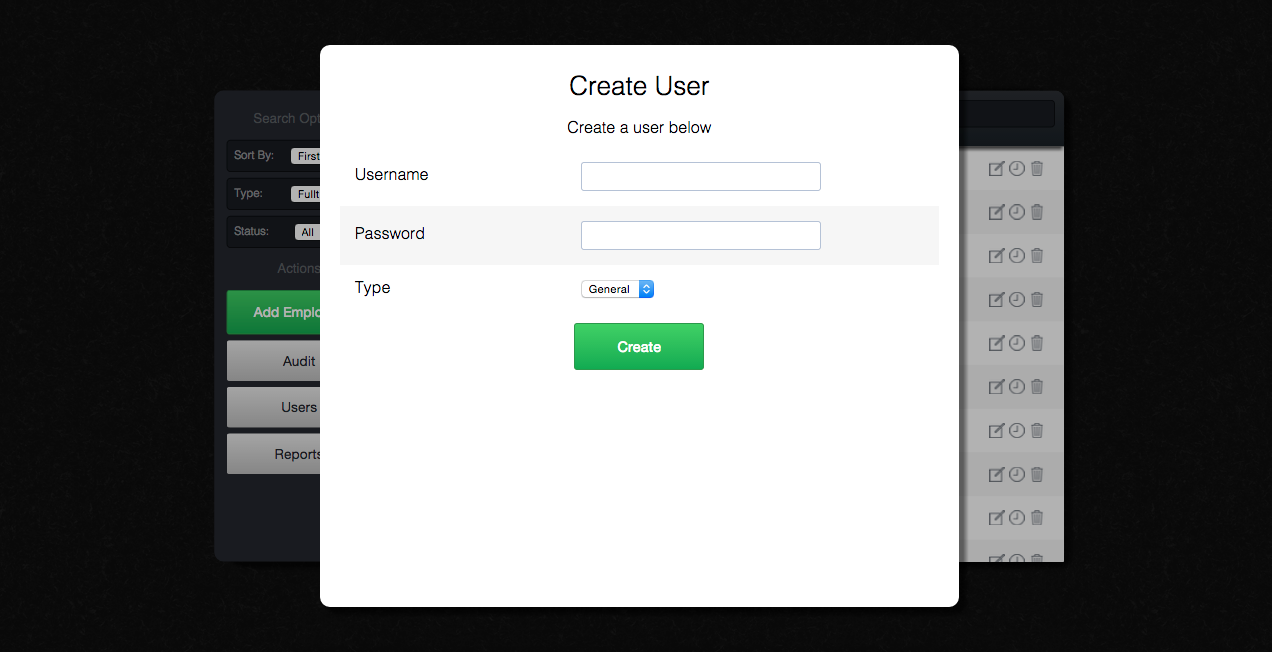
Reflects When and What change was made, by Whom, old and new value.

## Users

### Overview

Users of the system are able to create new users, in order to allow other people to use the system. General information is used to create a new user, and permission level can be assigned. Such a function will be provided depending on user’s permission level.

### User Interface



This section is allow Administrative users to create a new user and set permissions to either general or administrative. After logging, user can click Users to get in this section. It will ask user for Username, Password and security level. When the action is complete, the account information will be saved into system and then be available to use.

### Business Logic

General User adds employee to the system. Administrative user completes records and manages system.

# Technological Considerations

The system will use several technologies that fit well together. For the front-end user interface, it will use HTML, CSS, and JavaScript. For the back-end, server-side processing, it will use PHP. For the back-end database, it will use MySQL.

For the front-end, the JavaScript framework jQuery will also be used. jQuery will help simplify the clunky DOM interface and result in less lines of code. Each section’s markup will be contained in a separate file and then included into the main age when the system launches.

The revealing module JavaScript pattern will be used to code the user interface. The revealing module is a simple way to effectively write static classes in JavaScript. A variant of this pattern can also create non-static classes. More information on the revealing module pattern can be found here: <http://addyosmani.com/resources/essentialjsdesignpatterns/book/#revealingmodulepatternjavascript>

A module will be created for each user interface section of the web application. Inheritance can also be done using jQuery’s extend method. For example, a base section class will be created and each specific section will inherit from it. Communicating between modules will be done using custom events. This ensures there is low coupling between modules. This jQuery plugin can be used to create custom events : <https://github.com/BenLorantfy/CustomEvent>

Server-side, functionality will be separated into collective classes that can be called with AJAX when requested. For example, one class might be called login.class.php while another would be called audit.class.php. There will also be classes for each employee type, inherited from a general employee class, which can be used to store the current employee as it is being edited or added.

Another jQuery plugin / design pattern will help communicate between the JavaScript and PHP more easily. It can be found here: <https://github.com/BenLorantfy/PostCall>

This design pattern allows PHP methods within a class to be coded one way for both regular synchronous PHP calls and AJAX asynchronous calls. Formerly, you would have to use the $\_POST super global within the method in order to access AJAX parameters instead of using the function parameters as usual. This design pattern includes a PHP script that receives the AJAX post parameters and calls the specified function, passing in the parameters to the function as function parameters. This simplifies the syntax of calling a function. The plugin also allows you to return data types other than string by using JSON encoding, and starts the session before calling the method if requested.

The database backend will be used strictly for storage of data. All processing and validation will occur within the PHP. This keeps storage and business logic separated from each other, allowing for low coupling. PHP is also better at processing / validating because it is a procedural language, rather than a relational language as is SQL.

Furthermore, the user table will store passwords hashed. This prevents a hacker from trying passwords on other user services if they gain access to the database.

# Appendix A

## C:\Users\Grigory\AppData\Local\Microsoft\Windows\INetCache\Content.Word\General User - New Page.pngGeneral User

## Administrative User

