

Abstract of project presented to the School of Technology Management in fulfillment of the requirements for the degree of Bachelor in Computer Science.

Web Based Payroll System

By

Siew Wing Fei

Jan 2013

As for the abstract, it usually encompasses four (4) elements:

1. Statement of problem of the research or project - issues addressed
2. method used for implementation
3. results and finding
4. conclusion

Web based applications have evolved significantly over recent years and with improvements in security and technology there are plenty of good opportunities to develop a system as web based application.

This dissertation has covered a full report on the Web Based Payroll System developed as a prototype to solve the manual payroll system.

The Web Based Payroll System has the ability to update and maintain employee details, define deductions, tax, pay rates, overtime pay rates, generate payslip, generate charts, etc by using Ruby on Rails web framework, jQuery, jQuery UI, and Highcharts JS.

The Web Based Payroll System uses AJAX technology to exchange data with a server, and update parts of a web page without reloading the whole page, which makes the applications similar to a desktop application.

Acknowledgment

First and foremost I thank to my family for helping me all the way from the very first day of my life.

Right after that I offer my sincerest gratitude to my supervisor, Mr. Amjad Hanesh, who has supported me throughout my studies and thesis with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my *bachelor's* degree to his encouragement and effort and without him this project and thesis, too, would not have been completed or written. Besides my academic studies, I have learned more than a lot from him, which includes RFID technology, Dot Net Framework, Mobile development, CCTV camera programming, and FTIR Multi Touch Screen development etc. One simply could not wish for a better and friendlier supervisor.

My sincere thanks go to all lecturers and members of the staff of the School of Technology Management, Binary University College, who helped me in many ways and made my education journey at Binary University College pleasant and unforgettable. During my part time studies I have been blessed and helped with a friendly and cheerful group of college staff, and fellow students, Mr. Bilal (School of Technology Management Coordinator),

I acknowledge my sincere indebtedness and gratitude to my parents for their love, dream and sacrifice throughout my life. I am really thankful for their sacrifice, patience, and understanding that were inevitable to make this work possible. Their sacrifice had inspired me from the day I learned how to read and write until what I have become now. I cannot find the appropriate words that could properly describe my appreciation for their devotion, support and faith in my ability to achieve my dreams.

Finally, I would like to thanks any person which contributes to my final year project directly or indirectly. I would like to acknowledge their comments and suggestions, which was crucial for the successful completion of this study.

Declaration

I declare this project work in my original work except for the quotations and citation which has been accordingly acknowledged. I also declare this project work has not been previously, and is not currently, submitted for any other degree at Binary University College.

(Signature)

SIEW WING FEI

List of Figures

Figure 1: The synchronous interaction pattern of a traditional web application (top) compared with the asynchronous pattern of an Ajax application (bottom)	4
Figure 2: The traditional model for web applications (left) compared to the Ajax model (right).	17
Figure 3: A typical collaboration of the MVC components.	22
Figure 4: Use Case Diagram – Administrator/HR Administrator	26
Figure 5: Use Case Diagram - Employee	27
Figure 6: Generic Overview of Web Based Payroll System	41
Figure 7: Level 0 Data Flow Diagram	42
Figure 8: Level 1 Data Flow Diagram	43
Figure 9: Class Diagram of Domain Model.....	44
Figure 10: Class Diagram of Controller	45
Figure 11: All-in-one Architecture Model.....	47
Figure 12: Flat Architecture Model.....	47
Figure 13: Hub-and-spoke (or Daisy) pattern	48
Figure 14: Web Based Payroll System Accordion	49
Figure 15: Paging interface of Web Based Payroll System.....	49
Figure 16: rails server command	54
Figure 17: The main project folder	59
Figure 18: The application folder	60
Figure 19: Running unit test in Rails.....	61
Figure 20: Running functional test in Rails	61
Figure 21: Running integration test in Rails.....	62
Figure 22: Black Box testing	63

List of Tables

Table 1: Actors Grid	28
Table 2: Use Cases Grid.....	29
Table 3: Use Case Diagram Summary	31
Table 4: Operating Environment.....	35
Table 5: Chart used	36
Table 6: Five levels of testing	62
Table 7: Test Case Planning Format	63
Table 8: Test cases for Web Based Payroll System	64

TABLE OF CONTENTS

Abstract.....	ii
Acknowledgment	iii
Declaration.....	iv
List of Figures	v
List of Tables	vi
CHAPTER ONE INTRODUCTION.....	1
1.1: Introduction.....	1
1.2: Problem Statement.....	1
1.3: Significance of the project.....	2
1.3.1: Benefits of web based application.....	2
1.4: Payroll System	3
1.5: Ajax web application.....	3
1.5.1: Why Ajax is widely used in Web Applications	4
1.5.2: Technologies used in Ajax	5
1.6: Methodology	6
1.7: Objectives.....	6
1.8: Project write-up Layout	6
1.9: Limitations.....	7
CHAPTER TWO LITERATURE REVIEW.....	8
2.1: Web based application.....	8
2.1.1: History	8
2.1.2: Interface	10
2.1.3: Structure.....	10
2.1.4: Business use.....	10
2.1.5: Web application development	11
2.1.6: Benefits.....	11
2.1.7: Drawbacks	12
2.2: Single-page application.....	13
2.2.1: Architectural characteristics.....	13
2.2.2: Technical approaches.....	14

2.2.3: Running locally.....	15
2.2.4: Challenges with the SPA model	15
2.2.5: Page lifecycle	16
2.3: Ajax	16
2.3.1: History	17
2.3.2: Technologies	18
2.3.3: Drawbacks	18
2.4: Payroll System	20
2.5: Model-view-controller (MVC) design pattern	21
2.5.1: History	21
2.5.2: Component interactions.....	21
2.5.3: Dependency hierarchy	22
2.5.4: Use in web applications.....	22
CHAPTER THREE REQUIREMENT AND ANALYSIS.....	24
3.1: Requirements Analysis in general	24
3.1.1: Software prototyping	25
3.1.2: Outline of the prototyping process	25
3.2: Software Requirements Specification.....	26
3.2.1: USE CASE Diagram	26
3.2.2: Requirement Diagram	34
3.2.3: Operating Environment	35
3.2.4: Constraints and Dependencies.....	35
CHAPTER FOUR DESIGN.....	40
4.1: Design in Software Engineering.....	40
4.2: Flow of the data.....	42
4.3: Class Diagram	44
4.4: Overall Sequence diagram	46
4.5: Website Architecture	46
4.5.1: Architecture Model.....	47
4.6: Navigation Design.....	48
4.6.1: Accordion.....	48
4.6.2: Paging	49
CHAPTER FIVE IMPLEMENTATION	50
5.1: Implementation in Software Engineering	50

5.2: Before writing the Web Based Payroll System.....	50
5.3: Selecting the right tools, programming language, frameworks, technologies, and IDEs	51
5.3.1: APIs and Libraries.....	51
5.3.2: Frameworks	53
5.3.3: Technologies.....	55
5.3.4: Integrated Development Environment	56
CHAPTER SIX TESTING.....	57
6.1: Testing in Software Engineering	57
6.2: Goals and Types of Testing.....	57
6.2.1: System Testing	58
6.2.2: Component Testing	58
6.3: Black box testing	63
6.4: Test Cases.....	63
CHAPTER SEVEN FUTURE WORK/MAINTENANCE	72
7.1: Recommendations and Features.....	72
7.1.1: Leave Management System	72
7.1.2: Time Management System.....	73
7.1.3: Import/Export	73
References	74
APPENDIX A COMPLETE SOURCE CODE.....	76

CHAPTER ONE

INTRODUCTION

1.1: Introduction

Web based applications have evolved significantly over recent years and with improvements in security and technology there are plenty of good opportunities to develop a system as web based application. The Payroll System project is a fully Ajax enabled web application which is developed to solve the manual payroll system. The framework that was used in the project is Ruby on Rails, which is open source full-stack web application framework for the Ruby programming language. Ruby on Rails uses the Model-View-Controller architecture pattern to organize application programming. In this dissertation, we will go through all the steps that were involved in the development process of the system.

1.2: Problem Statement

Manual payroll system calculates the employee's salary entirely on paper. If the system is not systematic, searching for particular employee detail is very difficult. In addition, the pay slips for the employee will have to be manually generated. This problem will increase the amount of time to process the payroll transaction. Besides, it is very easy to make mistakes when processing payroll transaction manually, especially mistakes in taxing, which can be very costly. Furthermore, if the detail of the employee is kept on document, the document maybe seen by other people. All of this problem can be a burden for the payroll administrator due to the amount of manual works to be done by the payroll administrator before the pay slips can be given to employee. In order to solve the manual payroll system, a computerized payroll system is essential. Some suggestion has been made to solve the problem that faced by the payroll administrator. For example, a user has to enter their username and password in order to access the system. This will make the system more secure. The payroll administrator also can control the user authority to make sure the data is secure and only can be accessed by the authorized user. Moreover the chances of mistakes are lower because all the calculation is done by the system itself. The data is kept in a systematic way where it allows data to be searched. For example, payroll administrator can search by employee id, employee name, and others. The pay slips for employee will also automatically generated by the system with a mouse click. Payroll administrator can print out the pay slips anytime they want, and the pay slips will display all the details for the calculation of employee's salary. Data modifications are also easy because it can be done with just a mouse click.

1.3: Significance of the project

The significance of the project is listed below:

- The Payroll System will help to reduce work load and reduce the risk of information loss.
- The Payroll System will help the payroll administrator to manage the employee's salary in an efficient manner and increase work performance.
- The Payroll System will able to calculate the salary based on overtime, hourly rate, deduction and allowances according to the data inserted by the administrator.
- User can retrieve and view the records of employee's detail by using the search function from the system or generate various charts.
- The system will be more user friendly with web based application. The system will eliminate the time-consuming and potentially inaccurate method of handwritten notes and manually counted salary.
- The work performance of the payroll administrator will be improved, since using computerized system can save a lot of time.
- The information can be retrieved in a shorter time compared to the manual system. It will also reduce the data input time.
- Payroll administrator can maintain the employee's detail in an easier way by just using the update and delete function from the system.
- The system uses database to store the employee's detail, which encourages the integration of data and makes data more widely available.

1.3.1: Benefits of web based application

There are certain numbers of benefits of the project to be developed as a web based application. The benefits include:

- **Cross platform compatibility**

Most web based applications are far more compatible across platforms than traditional installed software. Typically the minimum requirement would be a web browser of which there are many. (Internet Explorer, Firefox, Google Chrome, to name but a few). These web browsers are available for a multitude of operating systems which includes Windows, Linux, or Mac OS.

- **More manageable**

Web based systems need only be installed on the server placing minimal requirements on the end user workstation. This makes maintaining and updating the

system much simpler as usually it can all be done on the server. Any client updates can be deployed via the web server with relative ease.

- **Highly deployable**

Due to the manageability and cross platform support deploying web applications to the end user is far easier. They are also ideal where bandwidth is limited and the system and data is remote to the user. At their most deployable the developer simply need to provide the user a website address to log in to and provide them with internet access.

This has huge implications allowing the developer to widen access to the systems, streamline processes and improve relationships by providing more users with access to the systems.

- **Secure live data**

Typically in larger more complex systems data is stored and moved around separate systems and data sources. In web based systems these systems and processes can often be consolidated reducing the need to move data around.

Web based applications also provide an added layer of security by removing the need for the user to have access to the data and back end servers.

- **Reduced costs**

Web based applications can dramatically lower costs due to reduced support and maintenance, lower requirements on the end user system and simplified architecture.
[1]

1.4: Payroll System

A Payroll System helps an employer to process its payroll. Consequently, payroll cannot be processed without a payroll system. A payroll system allows the employer to pay employees on time and accurately, plus comply with other statutory regulations. [2]

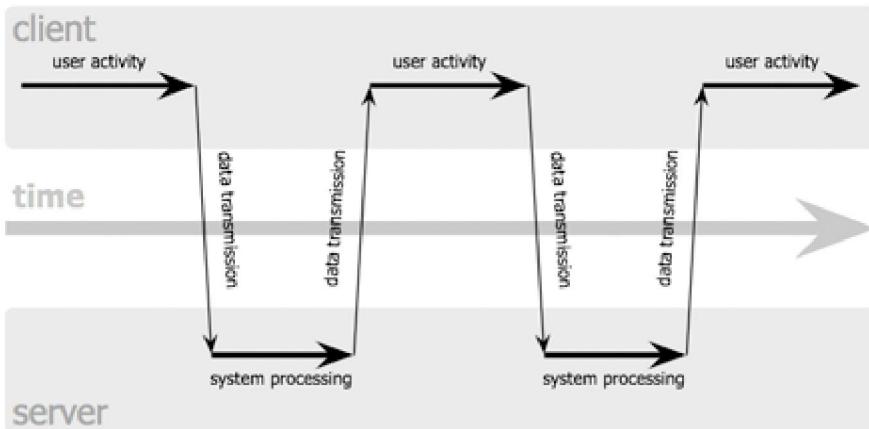
1.5: Ajax web application

Ajax (an acronym for Asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create asynchronous web applications.

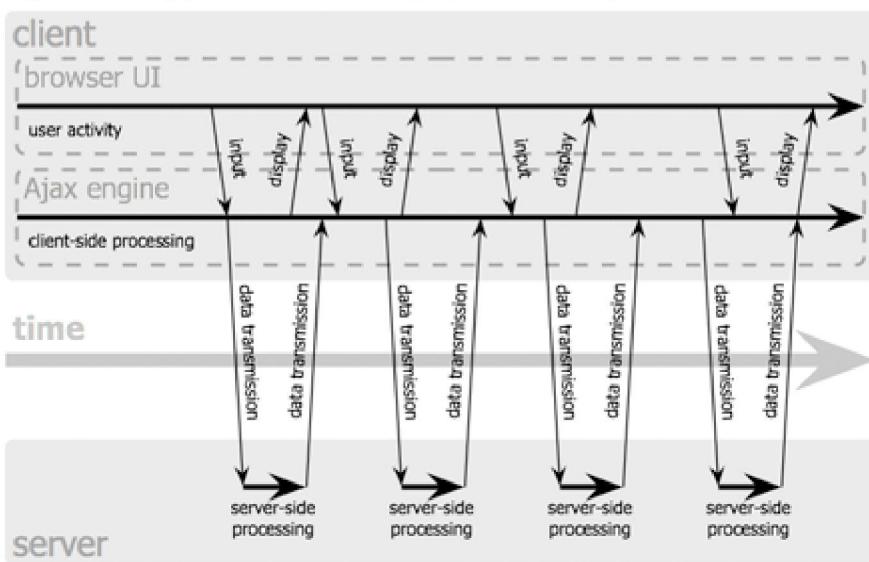
[3] In classic web application model, most user actions in the interface trigger an HTTP request back to a server. The server does some processing and then returns a HTML page

to the client. This approach makes a lot of technical sense, but it doesn't make for a great user experience. [4] With Ajax, web applications can send data to, and retrieve data from, a server asynchronously without interfering with the display and behavior of the existing page. Through Ajax, a web page feels like a desktop application. [3]

classic web application model (synchronous)



Ajax web application model (asynchronous)



Jesse James Garrett / adaptivepath.com

Figure 1: The synchronous interaction pattern of a traditional web application (top) compared with the asynchronous pattern of an Ajax application (bottom)

http://www.adaptivepath.com/uploads/archive/images/publications/essays/ajax-fig2_small.png

1.5.1: Why Ajax is widely used in Web Applications

- They can use a standard web browser, such as Firefox, Internet Explorer or Safari, as their only user interface.
- They don't force the user to wait for the web server every time the user clicks a button. This is what "asynchronous" means. For instance, Gmail fetches new email

messages in the background ("asynchronously") without forcing the user to wait. This makes an AJAX application respond much more like a "real" application on the user's computer, such as Microsoft Outlook.

- The Ajax engine works within the Web browser (through JavaScript and the DOM) to render the Web application and handle any requests that the customer might have of the Web server. The beauty of it is that because the Ajax engine is handling the requests, it can hold most information in the engine itself, while allowing the interaction with the application and the customer to happen as asynchronously and independently of any interaction with the server.
- They use standard JavaScript features (including the unofficial XMLHttpRequest standard, pioneered by Microsoft and adopted by Firefox and other browsers) to fetch data in the background and display different email messages or other data "on the fly" when the user clicks on appropriate parts of the interface.
- Usually they manipulate data in XML format. This allows AJAX applications to interact easily with server-side code written in a variety of languages, such as PHP, Perl/CGI, Python and ASP.NET. Using XML isn't absolutely necessary, and in fact many "AJAX" applications don't -- they use the XMLHttpRequest object to send and receive data "on the fly," but they don't actually bother packaging that data as XML.
- with Ajax, the JavaScript that is loaded when the page loads handles most of the basic tasks such as data validation and manipulation, as well as display rendering the Ajax engine handles without a trip to the server. At the same time that it is making display changes for the customer, it is sending data back and forth to the server. But the data transfer is not dependent upon actions of the customer. [5]

1.5.2: Technologies used in Ajax

JavaScript

- Loosely typed scripting language
- JavaScript function is called when an event in a page occurs
- Glue for the whole Ajax operation

DOM

- API for accessing manipulating structured documents
- Represents the structure of XML and HTML documents

CSS

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript

XMLHttpRequest

- JavaScript object that performs asynchronous interaction with the server [6]

1.6: Methodology

The methodology that was used in the Payroll System is Software Prototyping technique. The Software Prototyping technique refers to the activity of creating prototypes of software applications. A prototype typically simulates only a few aspects of, and may be completely different from the final product. The process of prototyping involves the following steps, which are identifying basic requirements, develop initial prototype, review, and revise and enhance the prototype.

1.7: Objectives

The objective of the project is to develop a web based payroll system using Ruby on Rails and Ajax technologies to:

- allow payroll administrator to create new or maintain existing employee details and the salary structure
- organize the entire employee's salary structure in one database that can only be accessed by authorized administrators
- allow payroll administrator to define the deductions, tax, generate payslip, and generate various charts
- allow employee to maintain his/her own details
- allow employee to view payslip
- allow employee to generate various charts

1.8: Project write-up Layout

The dissertation is divided into seven chapters, briefly described as follow:

- Chapter Two: will demonstrate the literature review on traditional Payroll System that was not developed as Ajax web application.
- Chapter Three: Analysis and Requirements Specification reveals the purpose, goal, scope of the Web Based Payroll System and preliminary investigation. It also clearly specifies the functional and non-functional requirements, which identifies the complete specification of requirements for the system development.

- Chapter Four: System Design presents that how the system process flow and the design of the system process for the development of the system.
- Chapter Five: Implementation presents that how the system is being written and developed according to the design that has been developed to lead the implementation.
- Chapter Six: System testing presents the testing strategy and test case to test the Web Based Payroll System and test results of the system.

1.9: Limitations

In this final year project illustrating its concept, there are subjected to some limitations which include:

- The time constraint
- The allocated project budget
- Knowledge in developing the software
- Knowledge on developing good payroll system software
- Selection of material or database information based on its simplicity, availability, affordability, and reliability
- The project is only a prototype, where it is not intended for production use due to time constraint

CHAPTER TWO

LITERATURE REVIEW

In the process of completing this project, the developers have taken upon to properly research the literature that already exists for the Web Based Application, Single-Page Application, Ajax technology, Payroll System, and Model-View-Controller (MVC) design pattern. The purpose is to get a better understanding on the subject before going further into the research and design phase of the development.

As such this paper will detail the sources in which the developer refer to and the knowledge gathered by the process. The main point of this research consists of example, and type of preexisting system that currently in used for the purpose of student information.

2.1: Web based application

A web based application is an application that is accessed by users over a network such as the Internet or an intranet. The term may also mean a computer software application that is coded in a browser-supported programming language (such as JavaScript, combined with a browser-rendered markup language like HTML) and reliant on a common web browser to render the application executable.

Web based applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web based applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. [7]

2.1.1: History

In earlier computing models, e.g. in client-server, the load for the application was shared between code on the server and code installed on each client locally. In other words, an application had its own client program which served as its user interface and had to be separately installed on each user's personal computer. An upgrade to the server-side code of the application would typically also require an upgrade to the client-side code installed on each user workstation, adding to the support cost and decreasing productivity.

In contrast, web applications use web documents written in a standard format such as HTML and JavaScript, which are supported by a variety of web browsers. Web applications can be

considered as a specific variant of client-server software where the client software is downloaded to the client machine when visiting the relevant web page, using standard procedures such as HTTP. Client web software update may happen each time the web page is visited. During the session, the web browser interprets and displays the pages, and acts as the universal client for any web application.

In the early days of the Web each individual web page was delivered to the client as a static document, but the sequence of pages could provide an interactive experience, as user input is returned through web form elements embedded in the page markup.

In 1995 Netscape introduced a client-side scripting language called JavaScript allowing programmers to add some dynamic elements to the user interface that ran on the client side. So instead of sending data to the server in order to generate an entire web page, the embedded scripts of the downloaded page can perform various tasks such as input validation or showing/hiding parts of the page.

In 1996, Macromedia introduced Flash, a vector animation player that could be added to browsers as a plug-in to embed animations on the web pages. It allowed the use of a scripting language to program interactions on the client side with no need to communicate with the server.

In 1999, the "web application" concept was introduced in the Java language in the Servlet Specification version 2.2. At that time both JavaScript and XML had already been developed, but Ajax had still not yet been coined and the XMLHttpRequest object had only been recently introduced on Internet Explorer 5 as an ActiveX object.

In 2005, the term Ajax was coined, and applications like Gmail started to make their client sides more and more interactive. A web page script is able to contact the server for storing/retrieving data without downloading an entire web page.

In 2011, HTML5 was finalized, which provides graphic and multimedia capabilities without the need of client side plugins. HTML5 also enriched the semantic content of documents. The APIs and document object model (DOM) are no longer afterthoughts, but are fundamental parts of the HTML5 specification. WebGL API paved the way for advanced 3D graphics based on HTML5 canvas and JavaScript language. These have significant importance in creating truly platform and browser independent rich web applications. [7]

2.1.2: Interface

Web developers often use client-side scripting to add functionality, especially to create an interactive experience that does not require page reloading. Recently, technologies have been developed to coordinate client-side scripting with server-side technologies such as PHP. Ajax, a web development technique using a combination of various technologies, is an example of technology which creates a more interactive experience. [7]

2.1.3: Structure

Applications are usually broken into logical chunks called "tiers", where every tier is assigned a role. Traditional applications consist only of 1 tier, which resides on the client machine, but web applications lend themselves to a n-tiered approach by nature. Though many variations are possible, the most common structure is the three-tiered application. In its most common form, the three tiers are called presentation, application and storage, in this order. A web browser is the first tier (presentation), an engine using some dynamic Web content technology (such as ASP, ASP.NET, CGI, ColdFusion, JSP/Java, PHP, Perl, Python, Ruby on Rails or Struts2) is the middle tier (application logic), and a database is the third tier (storage). The web browser sends requests to the middle tier, which services them by making queries and updates against the database and generates a user interface. [7]

2.1.4: Business use

An emerging strategy for application software companies is to provide web access to software previously distributed as local applications. Depending on the type of application, it may require the development of an entirely different browser-based interface, or merely adapting an existing application to use different presentation technology. These programs allow the user to pay a monthly or yearly fee for use of a software application without having to install it on a local hard drive. A company which follows this strategy is known as an application service provider (ASP), and ASPs are currently receiving much attention in the software industry.

Security breaches on these kinds of applications are a major concern because it can involve both enterprise information and private customer data. Protecting these assets is an important part of any web application and there are some key operational areas that must be included in the development process. This includes processes for authentication, authorization, asset handling, input, and logging and auditing. Building security into the applications from the beginning can be more effective and less disruptive in the long run.

In cloud computing model web applications are software as a service (SaaS). There are business applications provided as SaaS for enterprises for fixed or usage dependent fee.

Other web applications are offered free of charge, often generating income from advertisements shown in web application interface. [7]

2.1.5: Web application development

Writing of web applications is often simplified by open source software such as Django, Ruby on Rails or Symfony called web application frameworks. These frameworks facilitate rapid application development by allowing a development team to focus on the parts of their application which are unique to their goals without having to resolve common development issues such as user management. While many of these frameworks are open source, this is by no means a requirement.

The use of web application frameworks can often reduce the number of errors in a program, both by making the code simpler, and by allowing one team to concentrate on the framework while another focuses on a specified use case. In applications which are exposed to constant hacking attempts on the Internet, security-related problems can be caused by errors in the program. Frameworks can also promote the use of best practices such as GET after POST.

In addition, there is potential for the development of applications on Internet operating systems, although currently there are not many viable platforms that fit this model. [7]

2.1.6: Benefits

- Web applications do not require any complex "roll out" procedure to deploy in large organizations. A compatible web browser is all that is needed.
- Browser applications typically require little or no disk space on the client.
- They require no upgrade procedure since all new features are implemented on the server and automatically delivered to the users.
- Web applications integrate easily into other server-side web procedures, such as email and searching.
- They also provide cross-platform compatibility in most cases (i.e., Windows, Mac, Linux, etc.) because they operate within a web browser window.
- With the advent of HTML5, programmers can create richly interactive environments natively within browsers. Included in the list of new features are native audio, video and animations, as well as improved error handling. [7]

2.1.7: Drawbacks

- In practice, web interfaces, compared to thick clients, typically force significant sacrifice to user experience and basic usability.
- Web applications absolutely require compatible web browsers. If a browser vendor decides not to implement a certain feature, or abandons a particular platform or operating system version, this may affect a huge number of users.
- Standards compliance is an issue with any non-typical office document creator, which causes problems when file sharing and collaboration becomes critical.
- Browser applications rely on application files accessed on remote servers through the Internet. Therefore, when connection is interrupted, the application is no longer usable. However, if it uses HTML5 API's such as Offline Web application caching,[10] it can be downloaded and installed locally, for offline use. Google Gears, although no longer in active development, is a good example of a third party plugin for web browsers that provides additional functionality for creating web applications.
- Since many web applications are not open source, there is also a loss of flexibility, making users dependent on third-party servers, not allowing customizations on the software and preventing users from running applications offline (in most cases). However, if licensed, proprietary software can be customized and run on the preferred server of the rights owner.
- They depend entirely on the availability of the server delivering the application. If a company goes bankrupt and the server is shut down, the users have little recourse. Traditional installed software keeps functioning even after the demise of the company that produced it (though there will be no updates or customer service).
- Likewise, the company has much greater control over the software and functionality. They can roll out new features whenever they wish, even if the users would like to wait until the bugs have been worked out before upgrading. The option of simply skipping a weak software version is often not available. The company can foist unwanted features on the users or cut costs by reducing bandwidth. Of course, companies will try to keep the good will of their customers, but the users of web applications have fewer options in such cases unless a competitor steps in and offer a better product and easy migration.
- The company can theoretically track anything the users do. This can cause privacy problems. [7]

2.2: Single-page application

A single-page application (SPA), also known as single-page interface (SPI), is a web application or web site that fits on a single web page with the goal of providing a more fluid user experience akin to a desktop application.

In an SPA, either all necessary code – HTML, JavaScript, and CSS – is retrieved with a single page load, or partial changes are performed loading new code on demand from the web server, usually driven by user actions. The page does not automatically reload during user interaction with the application, nor does control transfer to another page. Updates to the displayed page may or may not involve interaction with a server. [8]

2.2.1: Architectural characteristics

2.2.1.1: The problem

The way traditional web applications work causes disruption in the user experience and workflow.

Traditional web applications work by reloading the entire web page. In order to advance through a workflow, the user interacts with page elements (such as hyperlinks and form submit buttons) that cause the browser to issue a request to the server for a completely new page.

User experience: Continual page redraws disrupt the user experience because the network latencies cannot be hidden from the user. There is typically a perceivable transitional jolt from one page to the next. The next page's data is retrieved from the server, the old page is unloaded, and the new page is rendered to screen.

Stable UI affordances, such as toolbars, navigation elements, database query results and so forth, continually disappear and reappear.

Performance: The complete page reload that occurs on each user interaction results in unnecessary re-transmission of data over the wire. This can make the overall performance of the web-site, when the entire session is taken into account, slower.

2.2.1.2: How SPAs address the problem

SPAs address these issues by requiring no page reload by the browser during an application session. All user interaction and changes of the application state are handled in the context of a single Web document.

The user experience becomes more continuous and fluid, and network latencies can be hidden more easily. [8]

2.2.2: Technical approaches

There are various techniques available that enable the browser to retain a single page even when the application requires server communication.

AJAX

The most prominent technique currently being used is Ajax. Predominantly using the XMLHttpRequest object from JavaScript, other AJAX approaches include using IFRAME or script HTML elements. Popular libraries like jQuery, that normalize AJAX behavior across browsers from different manufacturers, have further popularized the AJAX technique.

Node.js/SignalR

Asynchronous calls to the server may also be achieved using Node.js or SignalR in conjunction with Socket.io.

Browser plugins

Asynchronous calls to the server may also be achieved using browser plug-in technologies such as Silverlight, Flash or Java applets.

Data transport (XML, JSON and AJAX)

Requests to the server typically result in either raw data (e.g. XML or JSON), or new HTML being returned. In the case where HTML is returned by the server, JavaScript on the client updates a partial area of the DOM (Document Object Model). When raw data is returned, oftentimes a client-side JavaScript XML / (XSL) process (and in case of JSON a template) is used to translate the raw data into HTML, which is then used to update a partial area of the DOM.

Thin server architecture

An SPA moves logic from the server to the client. This results in the role of the web server evolving into a pure data API or web service. This architectural shift has, in some circles, been coined "Thin Server Architecture" to highlight that complexity has been moved from the server to the client, with the argument that this ultimately reduces overall complexity of the system.

Thick server architecture

The server keeps the necessary state in memory of the client state of the page. In this way, when any request hits the server (usually user actions), the server sends the appropriate HTML and/or JavaScript with the concrete changes to bring the client to the new desired state (usually adding/deleting/updating a part of the client DOM). At the same time the state in server is updated. Most of the logic is executed in server and HTML is usually also rendered in server. In some ways the server simulates a web browser, receiving events and performing delta changes in server state which are automatically propagated to client. This approach needs more server memory and server processing, but the advantage is a simplified development model because a) the application is usually fully coded in server, b) data and UI state in server are shared in the same memory space with no need of custom client/server communication bridges. [8]

2.2.3: Running locally

Some SPAs may be executed from a local file using the file URI scheme. This gives users the ability to download the SPA from a server and run the file from a local storage device, without depending on server connectivity. If such an SPA wants to store and update data, it must be self-modifying. That is, the SPA must be capable of writing itself to disk, including a representation of the state that is to be persisted. These applications benefit from advances available with HTML5, particularly Web Storage. [8]

2.2.4: Challenges with the SPA model

Because the SPA is an evolution away from the stateless page-redraw model that browsers were originally designed for, some new challenges have emerged. Each of these problems has an effective solution with:

- Client-side JavaScript libraries addressing various issues.
- Server side web frameworks that specialize in the SPA model.
- The evolution of browsers and the HTML5 specification aimed at the SPA model.

2.2.4.1: Browser history

With an SPA being, by definition, "a single page", the model breaks the browser's design for page history navigation using the Forward/Back buttons. This presents a usability impediment when a user presses the back button, expecting the previous screen state within the SPA, but instead the application's single page unloads and the previous page in the browser's history is presented.

The traditional solution for SPA's has been to change the browser URL's hash fragment identifier in accord with the current screen state. This can be achieved with JavaScript, and causes URL history events to be built up within the browser. As long as the SPA is capable of resurrecting the same screen state from information contained within the URL hash, the expected back button behavior is retained.

To further address this issue, the HTML5 specification has introduced pushState and replaceState providing programmatic access to the actual URL and browser history. [8]

2.2.5: Page lifecycle

An SPA is fully loaded in the initial page load and then page regions are replaced or updated with new page fragments loaded from the server on demand. To avoid excessive downloading of unused features, an SPA will often progressively download more features as they become required, either small fragments of the page, or complete screen modules.

In this way an analogy exists between "states" in an SPA and "pages" in a traditional web site. Because "state navigation" in the same page is analogous to page navigation, in theory any page based web site could be converted to single-page replacing in the same page only the changed parts result of comparing consecutive pages in a non-SPA.

The SPA approach on the web is similar to the Single Document Interface (SDI) presentation technique popular in native desktop applications. [8]

2.3: Ajax

Ajax (also AJAX; an acronym for Asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Data can be retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not required (JSON is often used instead), and the requests do not need to be asynchronous.

Ajax is not a single technology, but a group of technologies. HTML and CSS can be used in combination to mark up and style information. The DOM is accessed with JavaScript to dynamically display, and to allow the user to interact with the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads. [3]

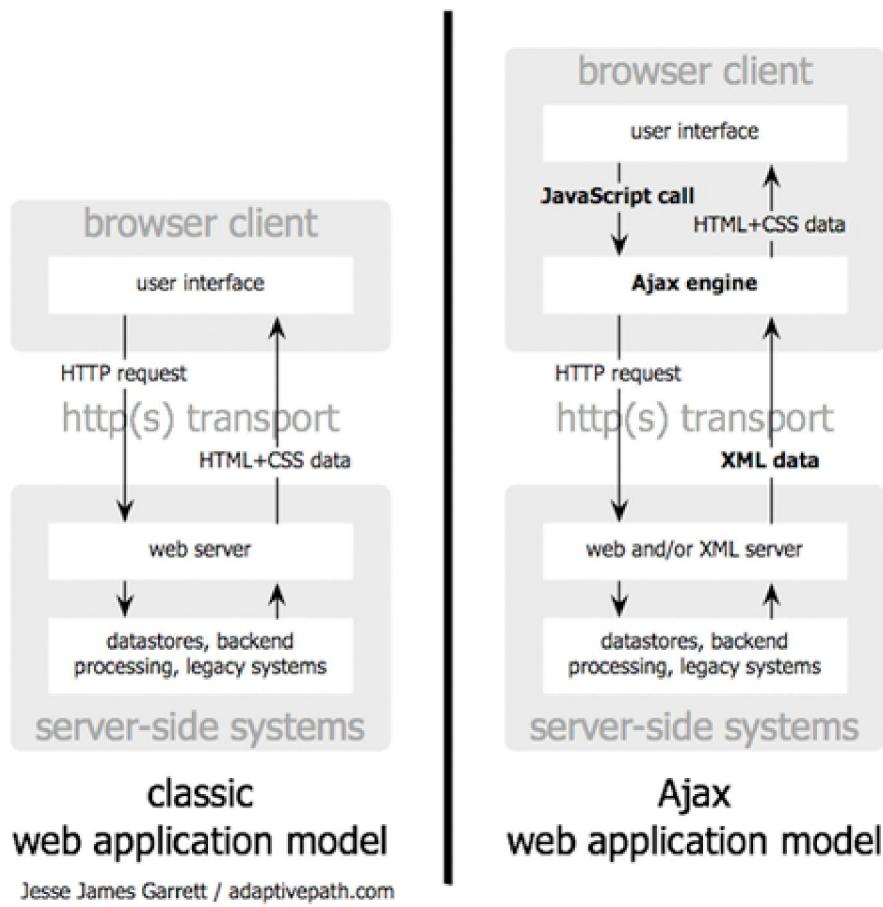


Figure 2: The traditional model for web applications (left) compared to the Ajax model (right).

http://www.adaptivepath.com/uploads/archive/images/publications/essays/ajax-fig1_small.png

2.3.1: History

In the 1990s, most web sites were based on complete HTML pages. Each user action required that the page be reloaded from the server (or a new page loaded). This process was inefficient, as reflected by the user experience: all page content disappeared then reappeared. Each time a page was reloaded due to a partial change, all of the content had to be re-sent, even though only some of the information had changed. This placed additional load on the server and used excessive bandwidth.

In 1996, the `iframe` tag was introduced by Internet Explorer to load content asynchronously.

In 1998, Microsoft Outlook Web Access team implemented the first component XMLHTTP by client script.

In 1999, Microsoft utilized its `iframe` technology to dynamically update the news stories and stock quotes on the default page for Internet Explorer, and created the XMLHTTP ActiveX control in Internet Explorer 5, which was later adopted by Mozilla, Safari, Opera and other browsers as the `XMLHttpRequest` JavaScript object. Microsoft has adopted the native

XMLHttpRequest model as of Internet Explorer 7, though the ActiveX version is still supported. The utility of background HTTP requests to the server and asynchronous web technologies remained fairly obscure until it started appearing in full scale online applications such as Outlook Web Access (2000) and Oddpost (2002).

Google made a wide deployment of standards-compliant, cross browser Ajax with Gmail (2004) and Google Maps (2005).

The term Ajax was coined on 18 February 2005 by Jesse James Garrett in an article entitled "Ajax: A New Approach to Web Applications", based on techniques used on Google pages.

On 5 April 2006 the World Wide Web Consortium (W3C) released the first draft specification for the XMLHttpRequest object in an attempt to create an official web standard. [3]

2.3.2: Technologies

The term Ajax has come to represent a broad group of web technologies that can be used to implement a web application that communicates with a server in the background, without interfering with the current state of the page. The following technologies are incorporated in Ajax:

- HTML (or XHTML) and CSS for presentation
- The Document Object Model (DOM) for dynamic display of and interaction with data
- XML for the interchange of data, and XSLT for its manipulation
- The XMLHttpRequest object for asynchronous communication
- JavaScript to bring these technologies together

There have been a number of developments in the technologies used in an Ajax application, and the definition of the term Ajax. XML is not required for data interchange and therefore XSLT is not required for the manipulation of data. JavaScript Object Notation (JSON) is often used as an alternative format for data interchange, although other formats such as preformatted HTML or plain text can also be used. [3]

2.3.3: Drawbacks

- In pre-HTML5 browsers, pages dynamically created using successive Ajax requests did not automatically register themselves with the browser's history engine, so clicking the browser's "back" button may not have returned the browser to an earlier state of the Ajax-enabled page, but may have instead returned to the last full page visited before it. Such behavior — navigating between pages instead of navigating between page states — may be desirable, but if fine-grained tracking of page state is

required then a pre-Ajax workaround was to use invisible iframes to trigger changes in the browser's history. A workaround implemented by Ajax techniques is to change the URL fragment identifier (the part of a URL after the '#') when an Ajax-enabled page is accessed and monitor it for changes. HTML5 provides an extensive API standard for working with the browser's history engine.

- Dynamic web page updates also make it difficult to bookmark and return to a particular state of the application. Solutions to this problem exist, many of which again use the URL fragment identifier. The solution provided by HTML5 for the above problem also applies for this.
- Depending on the nature of the Ajax application, dynamic page updates may interfere disruptively with user interactions, especially if working on an unstable Internet connection. For instance, editing a search field may trigger a query to the server for search completions, but the user may not know that a search completion popup is forthcoming, and if the internet connection is slow, the popup list may show up at an inconvenient time, when the user has already proceeded to do something else.
- Because most web crawlers do not execute JavaScript code, publicly indexable web applications should provide an alternative means of accessing the content that would normally be retrieved with Ajax, thereby allowing search engines to index it.
- Any user whose browser does not support JavaScript or XMLHttpRequest, or simply has this functionality disabled, will not be able to properly use pages which depend on Ajax. Devices such as smartphones and PDAs may not have support for the required technologies, though this is becoming less of an issue. The only way to let the user carry out functionality is to fall back to non-JavaScript methods. This can be achieved by making sure links and forms can be resolved properly and not relying solely on Ajax.
- Similarly, some web applications which use Ajax are built in a way that cannot be read by screen-reading technologies, such as JAWS. The WAI-ARIA standards provide a way to provide hints in such a case.
- Screen readers that are able to use Ajax may still not be able to properly read the dynamically generated content.
- The same origin policy prevents some Ajax techniques from being used across domains, although the W3C has a draft of the XMLHttpRequest object that would enable this functionality. Methods exist to sidestep this security feature by using a special Cross Domain Communications channel embedded as an iframe within a page, or by the use of JSONP.

- The asynchronous callback-style of programming required can lead to complex code that is hard to maintain, to debug and to test. [3]

2.4: Payroll System

A payroll system involves everything that has to do with the payment of employees and the filing of employment taxes. This includes keeping track of hours, calculating wages, withholding taxes and other deductions, printing and delivering checks and paying employment taxes to the government.

Withholding and paying taxes is one of the most important responsibilities of the payroll system. In the United States, the following are the major withholdings required by the government:

- Federal income tax
- State and local income taxes (where applicable)
- Social Security tax
- Medicare tax

When an employer withholds taxes from a paycheck, he acts as the trustee for those funds until they are paid to the IRS, the Social Security Administration (SSA) or other government agency. To avoid confusing this money with profits or other business income, all withheld taxes must be held in a separate bank account or trust fund.

In the case of Social Security and Medicare withholdings, when it's time to hand that money over to the government, the employer is required to match the employee's contributions. For example, if an employee is paying 6.2 percent of every check for Social Security, then the employer has to pay an equal 6.2 percent.

In addition to matching Social Security and Medicare contributions, the employer has to pay federal and state unemployment taxes (FUTA and SUTA) for each employee. The employer pays these taxes himself, meaning nothing is withheld from the paycheck.

There are numerous other possible deductions, withholdings and contributions that can be subtracted from an employee's gross wages and that need to be tracked by the payroll system:

- Health insurance or life insurance premiums
- 401(k) or other retirement fund contributions
- Workman's compensation
- Union dues

- Vacation days
- Sick days
- Employee loans
- Court-ordered wage garnishments (for outstanding debts)
- Child support payments

At the end of the year, an employer uses the payroll system to take all of the wage and withholding information from the previous year and summarize it on a W-2 form for full-time employees or a 1099 form for contract workers. Copies of that form must be sent to the employee, the IRS and the SSA. [9]

2.5: Model-view-controller (MVC) design pattern

Model–view–controller (MVC) is a software architecture pattern that separates the representation of information from the user's interaction with it. The model consists of application data, business rules, logic, and functions. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a bar chart for management and a tabular view for accountants. The controller mediates input, converting it to commands for the model or view. The central ideas behind MVC are code reusability and separation of concerns. [10]

2.5.1: History

The model-view-controller pattern was originally formulated in the late 1970s by Trygve Reenskaug at Xerox PARC, as part of the Smalltalk system. [10]

2.5.2: Component interactions

In addition to dividing the application into three kinds of components, the MVC design defines the interactions between them.

- A controller can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).
- A model notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.

- A view requests from the model the information that it needs to generate an output representation. [10]

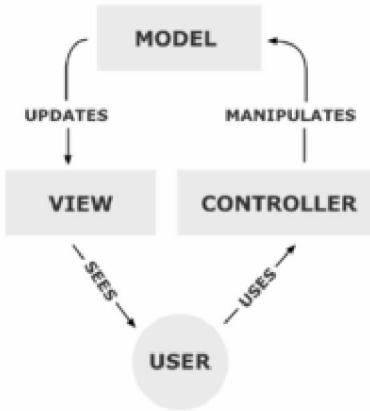


Figure 3: A typical collaboration of the MVC components.

<http://upload.wikimedia.org/wikipedia/commons/thumb/f/fd/MVC-Process.png/200px-MVC-Process.png>

2.5.3: Dependency hierarchy

There is usually a kind of hierarchy in the MVC pattern. The Model knows only about itself. That is, the source code of the Model has no references to either the View or Controller.

The View however, knows about the Model. It will poll the Model about the state, to know what to draw. That way, the View can draw something that is based on what the Model has done. But the View knows nothing about the Controller.

The Controller knows about both the Model and the View. To take an example from a game: If you click on the "fire" button on the mouse, the Controller knows what fire function in the Model to call. If you press the button for switching between first and third person, the Controller knows what function in the View to call to request the display change.

The reason to keep it this way is to minimize dependencies. No matter how the View class is modified, the Model will still work. Even if the system is moved from Windows to an app in a smart phone, the Model can be moved with no changes. But the View probably needs to be updated, as will the Controller. [10]

2.5.4: Use in web applications

Although originally developed for personal computing, Model View Controller has been widely adapted as architecture for World Wide Web applications in all major programming languages. Several commercial and noncommercial application frameworks have been

created that enforce the pattern. These frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server.

Early web MVC frameworks took a thin client approach that placed almost the entire model, view and controller logic on the server. In this approach, the client sends either hyperlink requests or form input to the controller and then receives a complete and updated web page (or other document) from the view; the model exists entirely on the server. As client technologies have matured, frameworks such as JavaScriptMVC and Backbone have been created that allow the MVC components to execute partly on the client. [10]

CHAPTER THREE

REQUIREMENT AND ANALYSIS

3.1: Requirements Analysis in general

Requirements Analysis is the process of determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements. [1]

Requirements are description of how a system should behave or a description of system properties or attributes. It can alternatively be a statement of ‘what’ an application is expected to do. [1]

Requirements analysis includes three types of activities:

- Eliciting requirements: the task of identifying the various types of requirements from customers and users. This is sometimes also called requirements gathering.
- Analyzing requirements: determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent conflicts.
- Recording requirements: Requirements may be documented in various forms, usually including a summary list and may include natural-language documents, use cases, user stories, or process specifications. [1]

Requirements analysis can be a long and arduous process during which many delicate psychological skills are involved. New systems change the environment and relationships between people, so it is important to identify all the stakeholders, take into account all their needs and ensure they understand the implications of the new systems. Analysts can employ several techniques to elicit the requirements from the customer. These may include the development of scenarios (represented as user stories in agile methods), the identification of use cases, the use of workplace observation or ethnography, holding interviews, or focus groups (more aptly named in this context as requirements workshops, or requirements review sessions) and creating requirements lists. Prototyping may be used to develop an example system that can be demonstrated to stakeholders. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced. [1]

In general, requirements are partitioned into functional requirements and non-functional requirements. Functional requirements are associated with specific functions, tasks or behaviors the system must support (e.g. the system must output the data in json format, or the system must provide the login function). While non-functional requirements are constraints on various attributes of these functions or tasks. The functional requirements address the quality characteristic of functionality while the other quality characteristics are concerned with various kinds of non-functional requirements (e.g. the system must be easily accessible, the quality must follow a particular standard). Because non-functional requirements tend to be stated in terms of constraints on the results of tasks which are given as functional requirements (e.g. constraints on the speed or efficiency of a given task)?

Software prototyping are used to gather the functional and non-functional requirements.

3.1.1: Software prototyping

Software prototyping refers to the activity of creating prototypes of software applications, i.e., incomplete versions of the software program being developed. It is an activity that can occur in software development and is comparable to prototyping as known from other fields. A prototype typically simulates only a few aspects of, and may be completely different from the final product. The main purpose of a prototype is to allow users of the software to evaluate developers' proposals for the design of the eventual product by actually trying them out, rather than having to interpret and evaluate the design based on descriptions. Prototyping can also be used by end users to describe and prove requirements that developers have not considered. [2]

3.1.2: Outline of the prototyping process

The process of prototyping involves the following steps

1. Identify basic requirements

Determine basic requirements including the input and output information desired.

Details, such as security, can typically be ignored.

2. Develop Initial Prototype

The initial prototype is developed that includes only user interfaces.

3. Review

The customers, including end-users, examine the prototype and provide feedback on additions or changes.

4. Revise and Enhance the Prototype

Using the feedback both the specifications and the prototype can be improved. Negotiation about what is within the scope of the contract/product may be necessary. If changes are introduced then a repeat of steps #3 and #4 may be needed. [2]

3.2: Software Requirements Specification

A Software requirements specification (SRS), a requirements specification for a software system, is a complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non-functional requirements. [3]

We will discuss the Use case diagram on the next page.

3.2.1: USE CASE Diagram

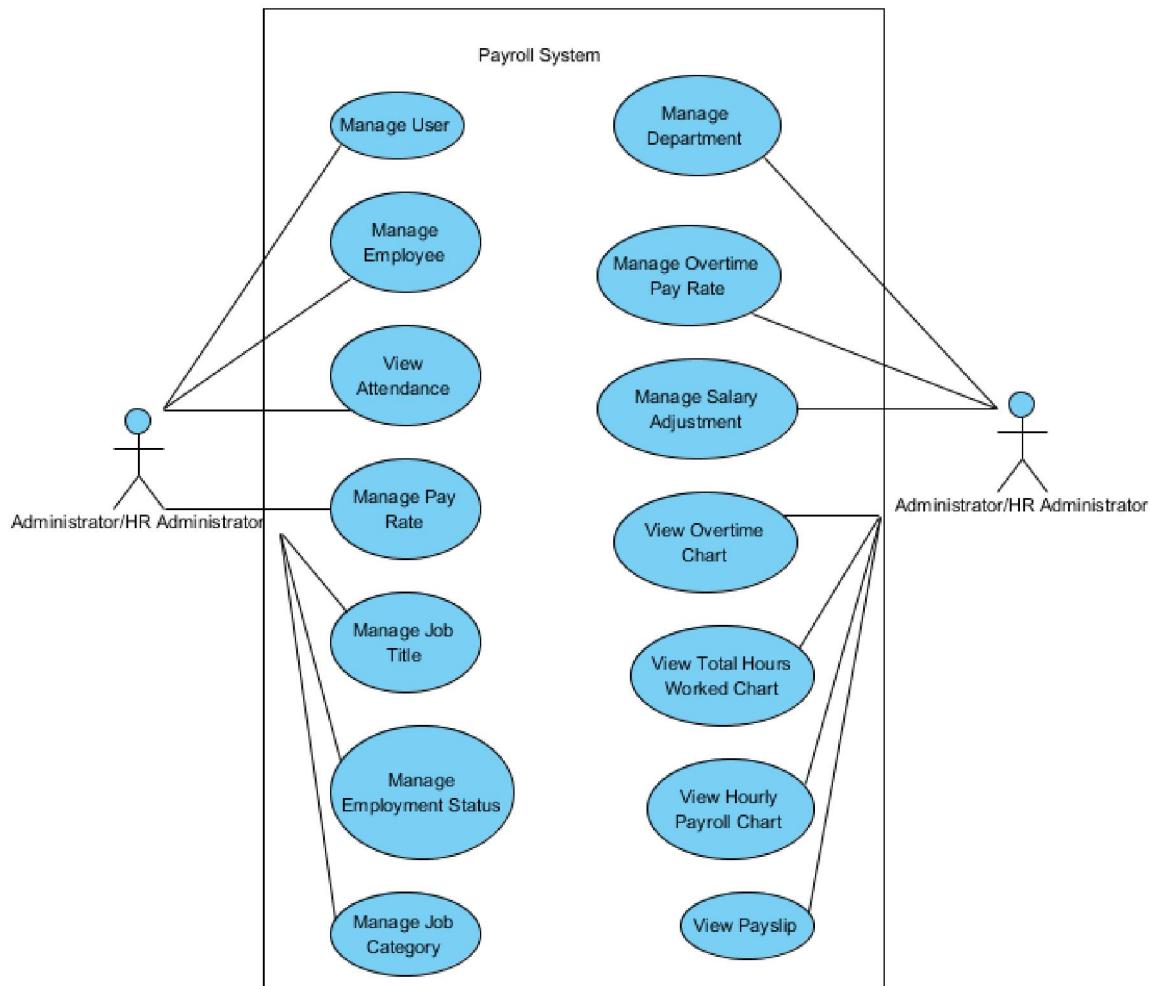


Figure 4: Use Case Diagram – Administrator/HR Administrator

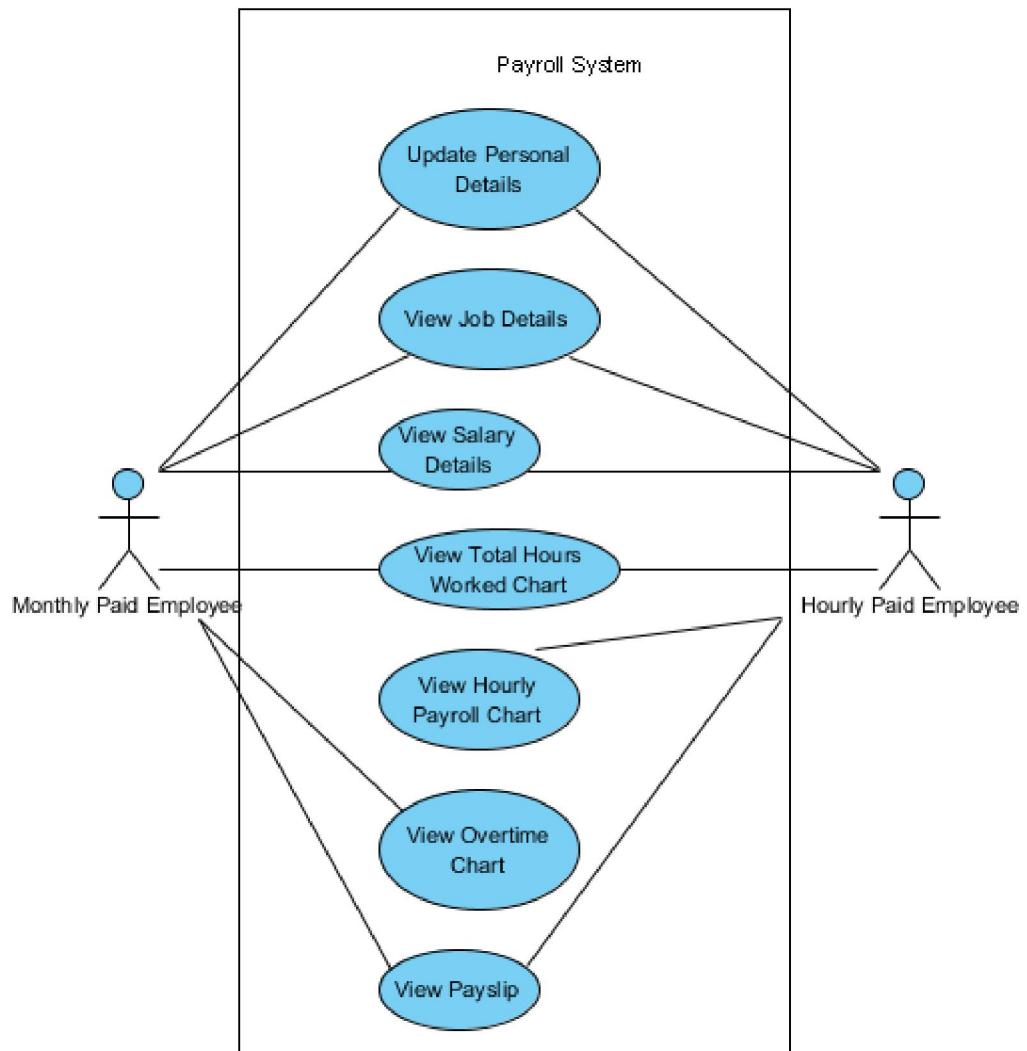


Figure 5: Use Case Diagram - Employee

Table 1: Actors Grid

ID	Name	Related use cases
1	Administrator/HR Administrator	Manage User, Manage Employee, View Attendance, Manage Pay Rate, Manage Job Title, Manage Employment Status, Manage Job Category, Manage Department, Manage Overtime Pay Rate, Manage Salary Adjustment, View Overtime Chart, View Total Work Hours Chart, View Hourly Payroll Chart, View Payslip
2	Monthly Paid Employee	Update Personal Details, View Job Details, View Salary Details, View Total Work Hours Chart, View Overtime Chart, View Payslip
3	Hourly Paid Employee	Update Personal Details, View Job Details, View Salary Details, View Total Work Hours Chart, View Hourly Payroll Chart, View Payslip

Table 2: Use Cases Grid

ID	Name	Primary actors	Supporting actors
1	Manage User	Administrator/HR Administrator	
2	Manage Employee	Administrator/HR Administrator	
3	View Attendance	Administrator/HR Administrator	
4	Manage Pay Rate	Administrator/HR Administrator	
5	Manage Job Title	Administrator/HR Administrator	
6	Manage Employment Status	Administrator/HR Administrator	
7	Manage Job Category	Administrator/HR Administrator	
8	Manage Department	Administrator/HR Administrator	
9	Manage Overtime Pay Rate	Administrator/HR Administrator	
10	View Overtime Chart	Administrator/HR Administrator, Monthly Paid Employee	
11	View Total Work Hours Chart	Administrator/HR Administrator, Monthly Paid Employee, Hourly Paid Employee	

12	Manage Salary Adjustment	Administrator/HR Administrator	
13	View Hourly Payroll Chart	Administrator/HR Administrator, Hourly Paid Employee	
14	View Payslip	Administrator/HR Administrator, Monthly Paid Employee, Hourly Paid Employee	
15	Update Personal Details	Monthly Paid Employee, Hourly Paid Employee	
16	View Job Details	Monthly Paid Employee, Hourly Paid Employee	
17	View Salary Details	Monthly Paid Employee, Hourly Paid Employee	

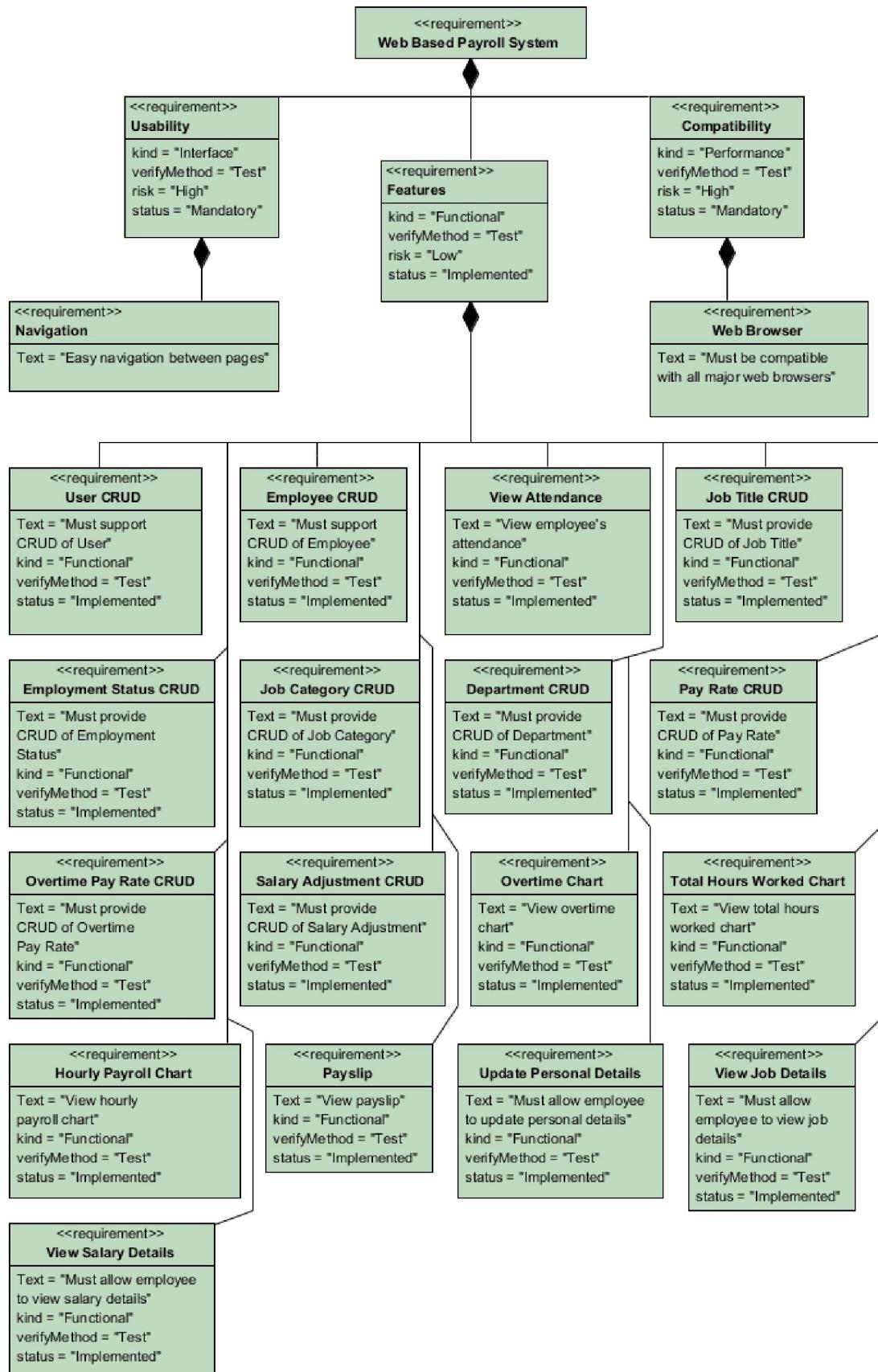
Table 3: Use Case Diagram Summary

Name	Documentation
 Administrator/ HR Administrator	Manage employee details, manage salary structure for each employee, and generate employee payslip at end of month.
 Manage User	Create, Update, and Delete user. A user is required in order to access the system.
 Manage Employee	Create, Update, and Delete employee. Employee details include personal details, contact details, job details, salary details, and qualification details.
 View Attendance	View employee's attendance record.
 Manage Pay Rate	Create, Update, and Delete pay rate.
 Manage Job Title	Create, Update, and Delete job title.
 Manage Employment Status	Create, Update, and Delete employment status.
 Manage Job Category	Create, Update, and Delete job category.
 Manage Department	Create, Update, and Delete department.
 Manage Overtime Pay Rate	Create, Update, and Delete overtime pay rate.
 Manage Salary	Create, Update, and Delete salary adjustment.

Adjustment	
 View Overtime Chart	View monthly employee's overtime chart.
 View Total Hours Worked Chart	View hourly paid employee's total hours worked chart.
 View Hourly Payroll Chart	View employee's payroll chart.
 View Payslip	The home page of the user where all the friend's, group's, page's, activities appears.
 Monthly Paid Employee/Hourly Paid Employee	The employee who can access system to update personal details, view salary details, view payroll charts, and generate payslip.
 Update Personal Details	Allows authenticated employee to update his/her personal details.
 View Job Details	Allows authenticated employee to view his/her job details.
 View Salary Details	Allows authenticated employee to view his/her salary details.
 View Total Hours Worked Chart	Allows authenticated hourly paid employee to view his/her total number of hours worked chart.
 View Hourly Payroll Chart	Allows authenticated hourly paid employee to view his/her hourly payroll chart.
 View Overtime Chart	Allows authenticated monthly paid employee to view his/her overtime chart.

 View Payslip	Allows authenticated employee to view/print his/her payslip.
 Payroll System	A system which allows Administrator to manage employee details, manage salary structure for each employee, and generate employee payslip at end of month, and allows employee to update personal details, view salary details, view payroll charts, and generate payslip.

3.2.2: Requirement Diagram



3.2.3: Operating Environment

In computing, an operating environment is the environment in which users run application software, whether by a command-line interface (such as in MS-DOS or the UNIX shell) or a graphical user interface (such as in the Macintosh operating system or a web browser). [4] Table X has briefly shows the operating environment of the Web based Payroll System.

Table 4: Operating Environment

No	Requirement	Reason
1	Google Chrome	Google Chrome web browser best work with the application
2	Any Operating System with decent web browser	To access the system via local intranet or internet
3	Ruby on Rails with jQuery UI as the client side user interface, jQuery as the javascript library, and Highcharts to display charts	To develop the system

3.2.4: Constraints and Dependencies

The Payroll System is dependent on jQuery [5], JQuery UI [6], and Highcharts [7] JavaScript library. jQuery is used to send Ajax request to Ruby code on the server side, resulting in a fast and asynchronous data transmission in a web browser. Meanwhile jQuery UI is used to create highly interactive web interface.

Highcharts is used to display intuitive and interactive charts in the payroll web application.

3.2.4.1: Dependency on jQuery

The system is completely dependent on the jQuery as the system is a fully ajax enabled web application. The goal of using jQuery is because it is fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

3.2.4.2: Dependency on jQuery UI

The system uses this library to create the front end web interface. jQuery UI is built on top of jQuery JavaScript library that provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets.

This library is completely dependent on the jQuery library.

3.2.4.3: Dependency on Highcharts

Highcharts is a charting library written in pure JavaScript, which offers intuitive, interactive charts to a web application. The system uses this library to show various chart types, such as line, bar, and pie.

Table 5: Chart used

Category	Chart Type	Description
Overtime Chart	Bar Chart	Shows monthly paid employee's overtime record for a particular year.
Total Work Hours Chart	Line Chart	Shows hourly paid employee's total working hours worked record for a particular year.
Hourly Payroll Chart	Pie Chart & Bar Chart	Shows hourly paid employee's payroll amount for a particular year.

3.2.5: System Features requirement

Section below will illustrate the required features of the Payroll System. The feature requirements fall under the function requirements because they interact directly with the user. [8]

3.2.5.1: Feature 1: User CRUD functions

Create, Read, Update, and Delete User. This feature allows administrator to create or maintain users which allow to access the system. Each user can be enabled or disabled. If a user is disabled, the user will not able to access the system. The administrator can search by username, employee name, user role, and status.

3.2.5.2: Feature 2: Employee CRUD functions

Create, Read, Update, and Delete Employee. This feature allows administrator to create or maintain employees in the system. Each employee can be associated with a user to allow them to access the system. The administrator can search by employee, employment status, department, staff id, and job title.

3.2.5.3: Feature 3: View Attendance

This feature allows the administrator to view employee's attendance record.

The administrator can search by employee and working date.

3.2.5.4: Feature 4: Job Title CRUD

Create, Read, Update, and Delete Job Title. This feature allows administrator to create or maintain job titles in the system. Each employee should have a job title. The administrator can search by keyword.

3.2.5.5: Feature 5: Employment Status CRUD

Create, Read, Update, and Delete Employment Status. This feature allows administrator to create or maintain employment statuses in the system. Each employee should have an employment status. Example of employment status would be Confirmed, Probation. The administrator can search by keyword.

3.2.5.6: Feature 6: Job Category CRUD

Create, Read, Update, and Delete Job Category. This feature allows administrator to create or maintain job categories in the system. Each employee should have a job category. The administrator can search by keyword.

3.2.5.7: Feature 7: Department CRUD

Create, Read, Update, and Delete Department. This feature allows administrator to create or maintain departments in the system. Each employee should belong to a department. The administrator can search by keyword.

3.2.5.8: Feature 8: Pay Rate CRUD

Create, Read, Update, and Delete Pay Rate. This feature allows administrator to create or maintain pay rates in the system. The administrator can define pay rate for each hourly paid employee, for each year. The administrator can search by staff id, year, and month.

3.2.5.9: Feature 9: Overtime Pay Rate CRUD

Create, Read, Update, and Delete Overtime Pay Rate. This feature allows administrator to create or maintain overtime pay rates in the system. The administrator can define rate for each year. The administrator can search by year.

3.2.5.10: Feature 10: Salary Adjustment CRUD

Create, Read, Update, and Delete Salary Adjustment. This feature allows administrator to create or maintain salary adjustments in the system. The administrator can define adjustment for a particular employee, for a particular year. The administrator can search by staff id, year, and month.

3.2.5.11: Feature 11: Overtime Chart

Shows monthly paid employee's overtime record for a particular year. The administrator can see the chart for a particular employee and for a particular year. It also has option to select months for the chart to be generated. The employee also can see his/her own chart.

3.2.5.12: Feature 12: Total Hours Worked Chart

Shows hourly paid employee's total working hours worked record for a particular year. The administrator can see the chart for a particular employee and for a particular year. It also has option to select months for the chart to be generated. The employee also can see his/her own chart.

3.2.5.13: Feature 13: Hourly Payroll Chart

Shows hourly paid employee's payroll amount for a particular year. The administrator can see the chart for a particular employee and for a particular year. It also has option to select months for the chart to be generated. The employee also can see his/her own chart.

3.2.5.14: Feature 14: View Payslip

Shows employee payslip. The administrator and employee can view and print the payslip.

3.2.5.15: Feature 15: Update Personal Details

Allows the employee to update the personal details, including contact details and qualification details.

3.2.5.16: Feature 16: View Job Details

Allows the employee to view the job details.

3.2.5.17: Feature 17: View Salary Details

Allows the employee to see his/her own salary details.

CHAPTER FOUR

DESIGN

4.1: Design in Software Engineering

A software design is a meaningful engineering representation of some software product that is to be built. It can be traced to the customer's requirements and can be accessed for quality against predefined criteria. In the software engineering context, design focuses on four major areas of concern: data, architecture, interfaces, and components. [1]

The design process is very important, because the emphasis in design is on quality; therefore it provides the representation of software that can be accessed for quality. Furthermore, this is the only phase in which the customer's requirements can be accurately translated into a finished software product or system. Thus, software design serves as the foundation for all software engineering steps that follow regardless of which process model is being employed. [1]

During the design process the software specifications are transformed into design models that describe the details of the data structures, system architecture, interface and components. Each design product is reviewed for quality before moving to the next phase of software development. At the end of the design process a design specification document is produced, which is composed of the design models that describe the data, architecture, interfaces and components. [1]

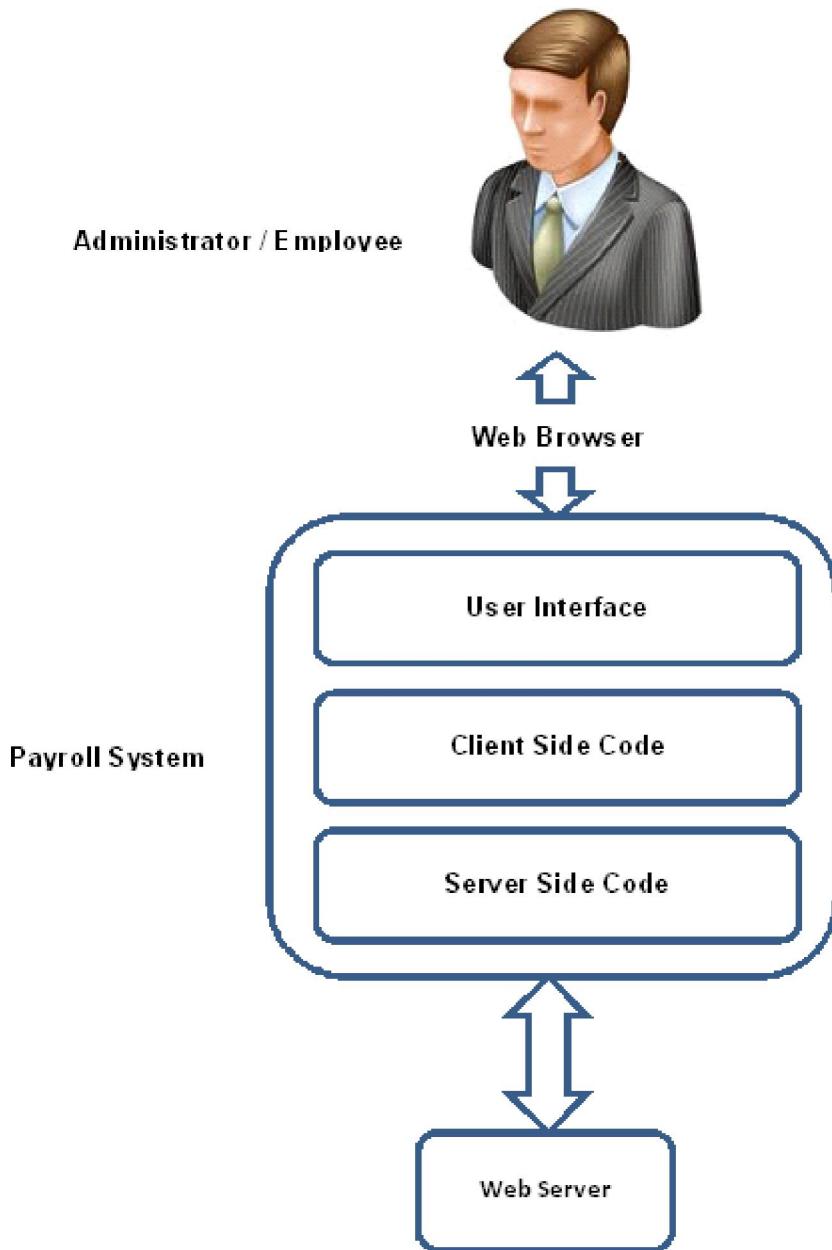


Figure 6: Generic Overview of Web Based Payroll System

4.2: Flow of the data

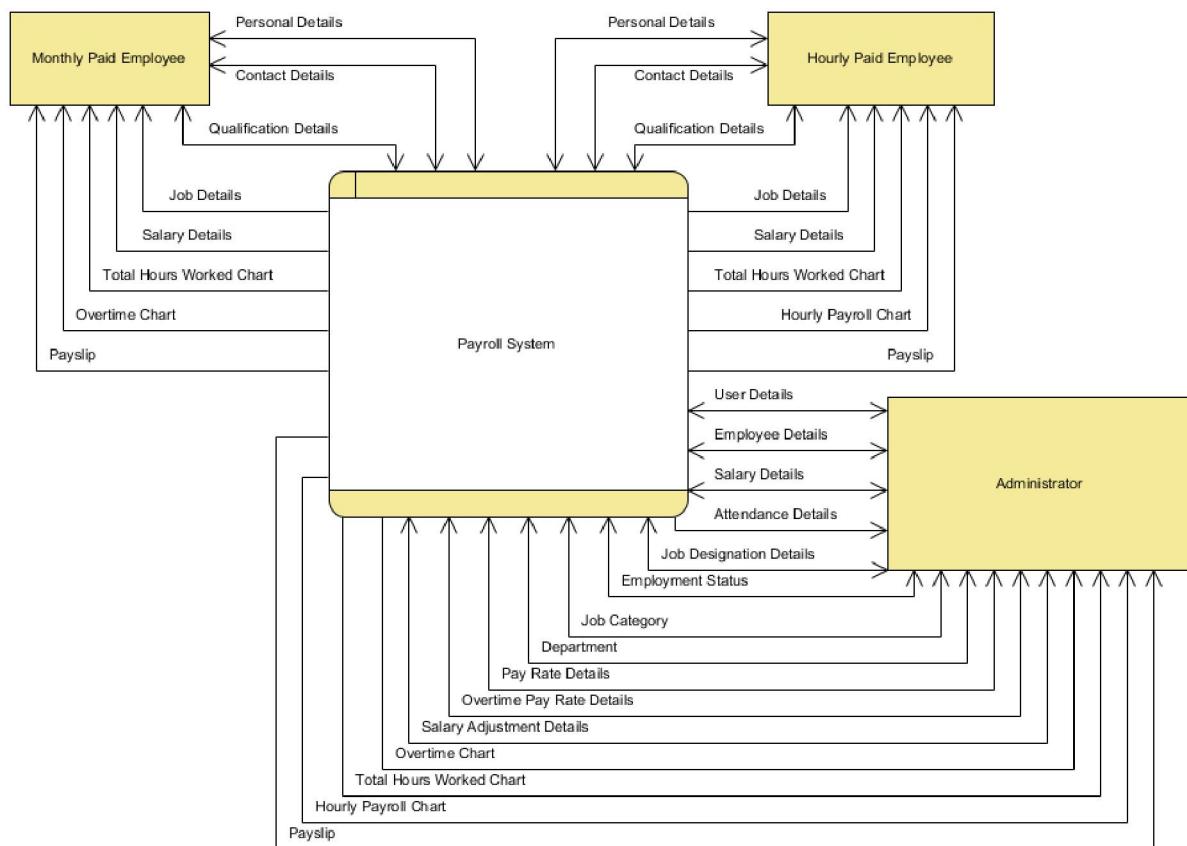


Figure 7: Level 0 Data Flow Diagram

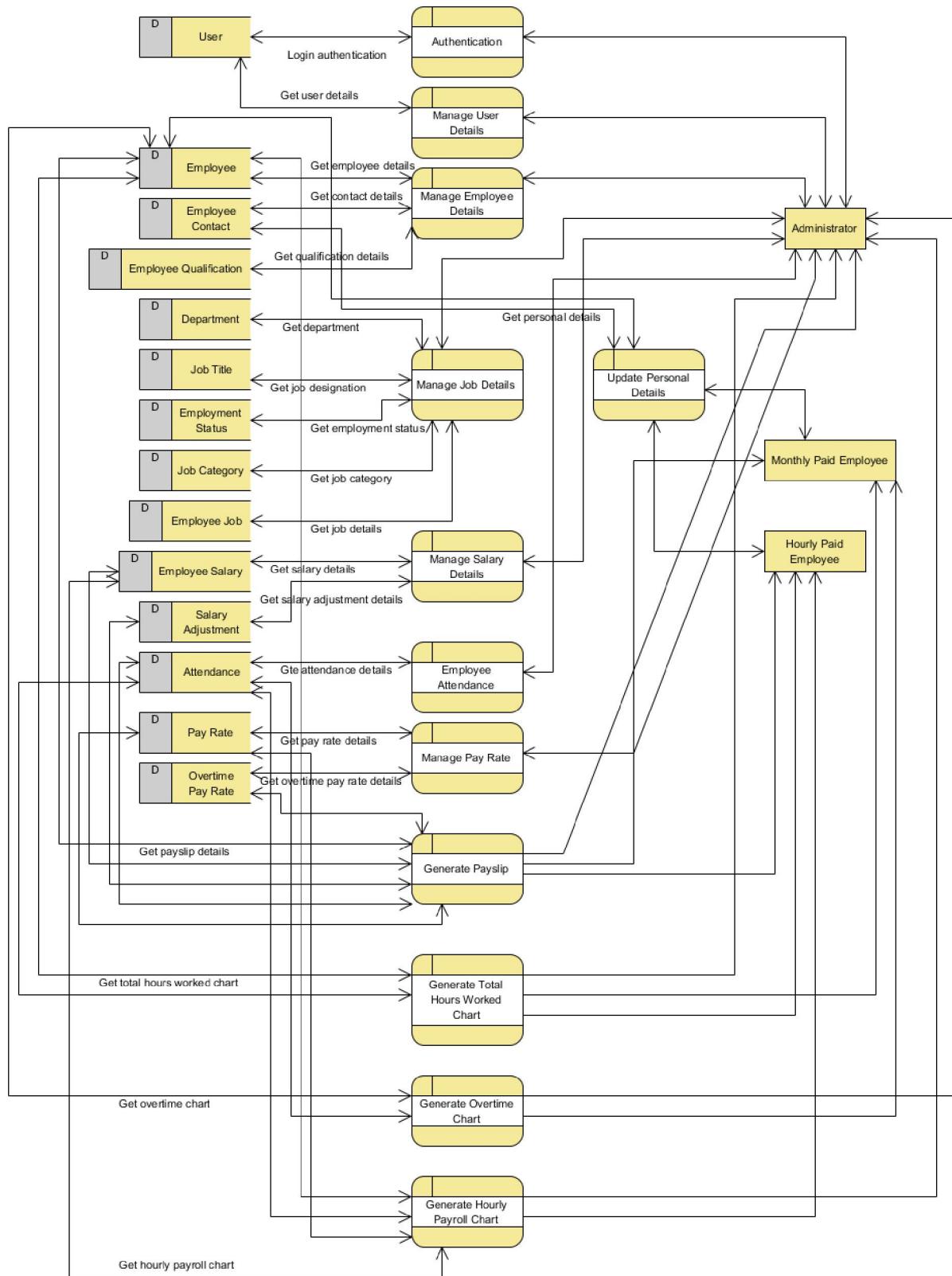


Figure 8: Level 1 Data Flow Diagram

4.3: Class Diagram

A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. A class is the description of a set of objects having similar attributes, operations, relationships and behavior.

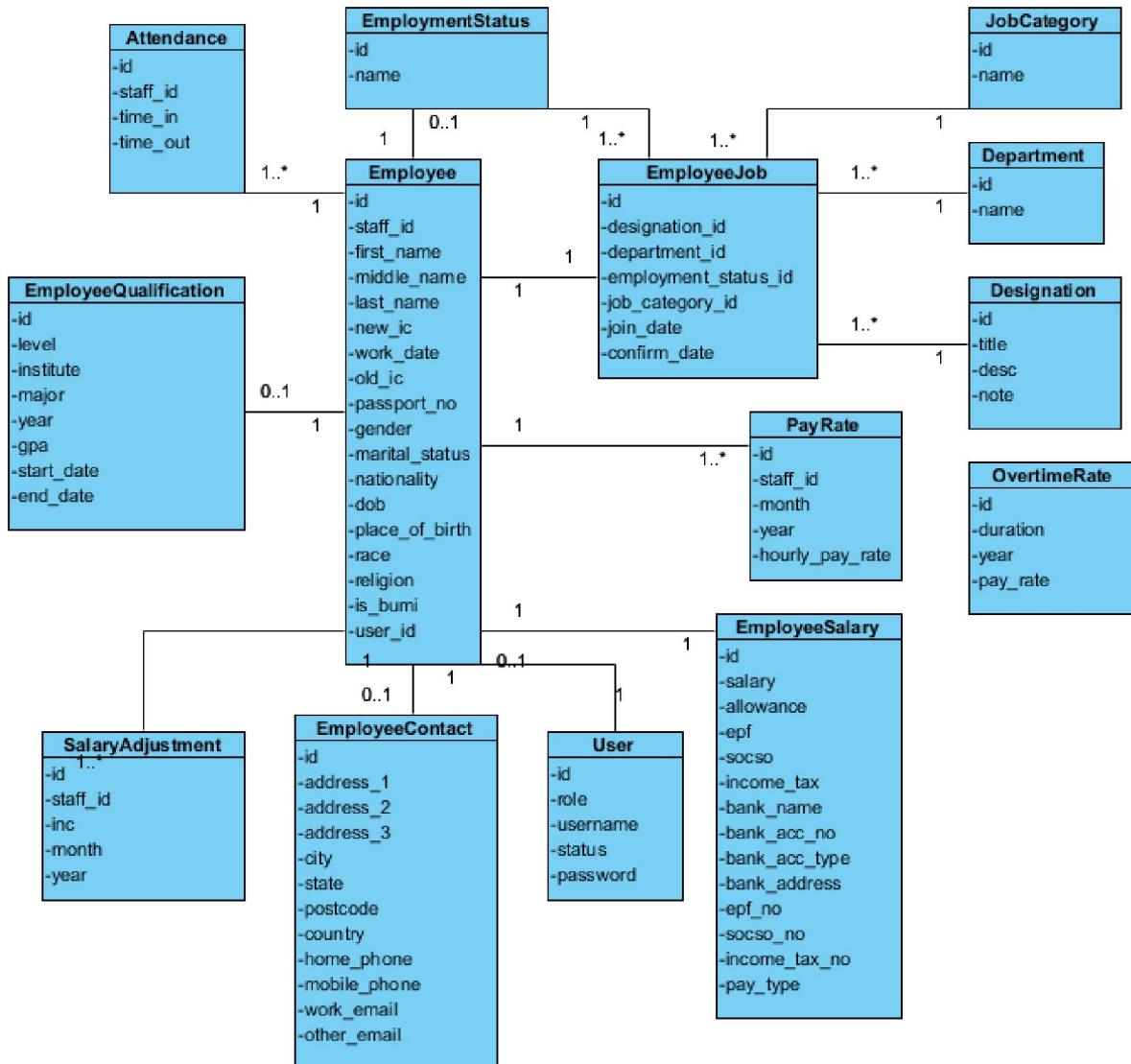


Figure 9: Class Diagram of Domain Model

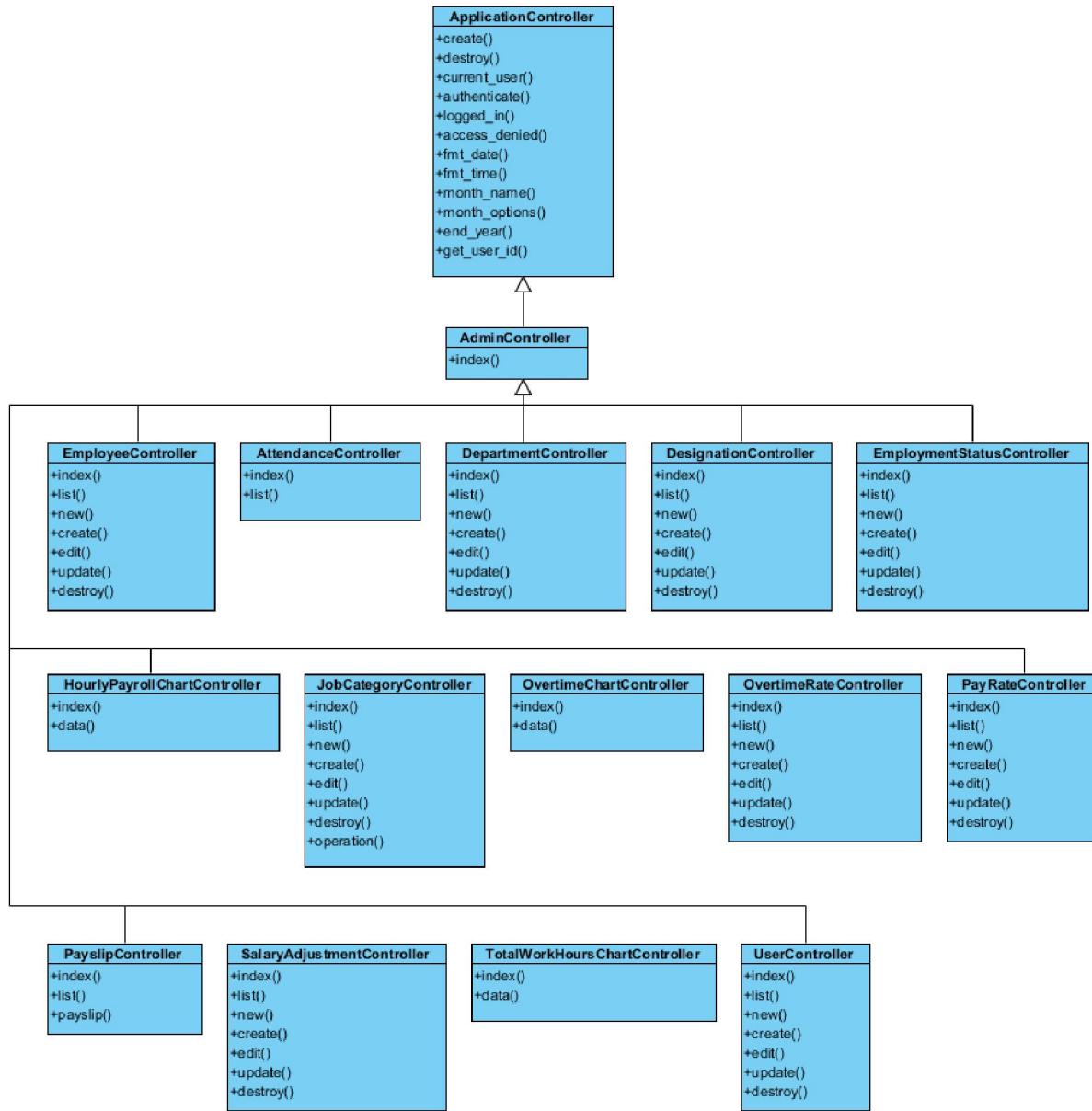
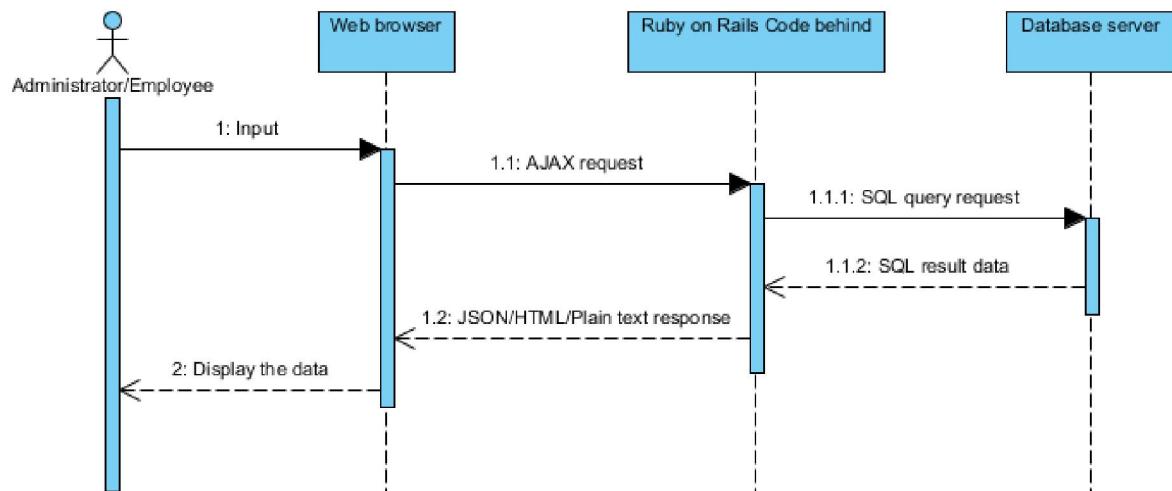


Figure 10: Class Diagram of Controller

4.4: Overall Sequence diagram

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur.



4.5: Website Architecture

Website architecture is an approach to the design and planning of websites which, like architecture itself, involves technical, aesthetic and functional criteria. As in traditional architecture, the focus is properly on the user and on user requirements. This requires particular attention to web content, a business plan, usability, interaction design, information architecture and web design.

"Website architecture" has the potential to be a term used for the intellectual discipline of organizing website content. "Web design", by way of contrast, describes the practical tasks, part-graphic and part-technical, of designing and publishing a website.

Before I got started with anything, I had to keep in mind few design guidelines to lead the project design. The design of the website should utilize the functionality of AJAX in order to make the web as user friendly as possible.

I ensure the following so AJAX can be utilized.

- Ensure no page reloads by dynamically update the content which needs to be updated.
- Ensure only JSON, html, and plain text format to be used in the data transmission between the client side and the server side.

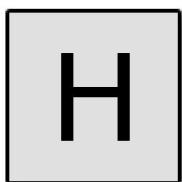
- Ensure there is a progress status on every AJAX request is made so that the user is aware that the request is being sent to the server.

4.5.1: Architecture Model

There are quite a few website architecture models that can be adopted, depends on the type of website you are developing. Following are some of the architecture models; I will briefly explain three models and then the model that has been used to develop the Web Based Payroll System.

Website Architecture Models:

4.5.1.1: All-in-one model



All-in-one Architecture Model

http://www.webdesignfromscratch.com/snippets/ia_diagram_allinone.gif

↳ goes on a single Home page.

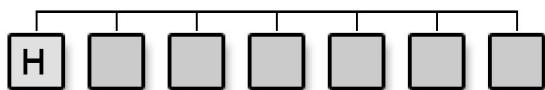


Figure 12: Flat Architecture Model

http://www.webdesignfromscratch.com/snippets/ia_diagram_monicline2.gif

A flat pattern is where all pages are arranged as peers, and everyone is accessible from every other one. This is very common for simple sites, where there are a few standard topics, such as: Home, About Us, Contact Us, Products.

I will be using flat model for the Web Based Payroll System so the accessing and navigation can be simpler and easier.

4.5.1.3: Hub-and-spoke / Daisy model

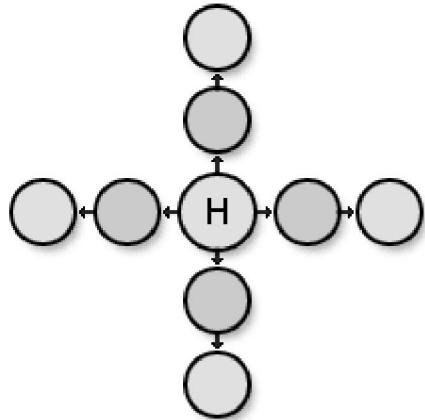


Figure 13: Hub-and-spoke (or Daisy) pattern

http://www.webdesignfromscratch.com/snippets/ia_diagram_hub_and_spoke.gif

This model is useful for multiple, distinct linear workflows. A good example may be an email application, where you will return to your inbox at several points, e.g. after reading a message, after sending a message, or after adding a new contact.

4.6: Navigation Design

Navigation design is the design of moving from one page, content, or area of the website to another. It organizes in such a way that the user will be navigating from one page to another more easily. As of Web Based Payroll System, the user will navigate using the mouse to interact with the web user interface.

4.6.1: Accordion

Accordion is a widget which displays collapsible content panels for presenting information in a limited amount of space.

Accordion is used in the Web Based Payroll System to provide navigation in the web application.

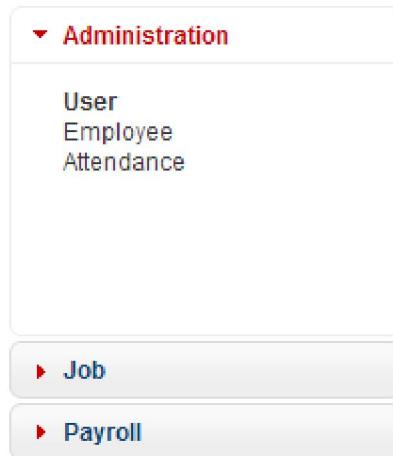


Figure 14: Web Based Payroll System Accordion

4.6.2: Paging

Paging is a term that will be familiar to all web users. This is where you get a piece of content that spans several pages. You are given standard tools that let you navigate previous, next, or jump directly to specific page. In Web Based Payroll System user can choose whether 20, 50, or 100 results is shown upon the request. The previous/next button will be disabled if there is no previous/next page.



Figure 15: Paging interface of Web Based Payroll System

CHAPTER FIVE

IMPLEMENTATION

5.1: Implementation in Software Engineering

In software implementation phase, we build the components of the system, or we can say that implementation is the phase of start writing the system or start doing programming, based on the given architecture documentation from the design phase and the requirement document from the analysis phase. Hence, the team should build exactly what has been requested, though there is still room for innovation and flexibility. For instance, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline. Therefore, the architecture document should give guidance. Sometimes, this guidance is found in the requirement document. This phase deals with issues of quality, performance, baselines, libraries, and debugging. The end deliverable is the product itself and the source code of the system. [1]

5.2: Before writing the Web Based Payroll System

An implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment. Many implementations may exist for a given specification or standard. For example, web browsers contain implementations of World Wide Web Consortium-recommended specifications, and software development tools contain implementations of programming languages. [2]

The implementation phase is a phase where you have to be very clear about each and every requirement of the customer because software implementation is a collaborative effort between the software vendor and the customer. Secondly, clear and open communication is essential. Of course, customer needs to communicate their objectives to the software vendor. But even more important, they must listen to what the software vendor tells you. The biggest sources of failure are the misunderstandings that develop between what the customer expects and what the software vendor can deliver. Be on your guard and learn the software's capabilities and limitations. The software vendor may not volunteer the product's failings unless customer does enough inquiring and it is possible that vendor's power of influence may convince customer to purchase a product that may not fully address the needs.

In the Web Based Payroll System's case, we listed down quite a few functionalities of the system, and the prototype can be demonstrated online at <https://payroll-wfsviewapp.rhcloud.com>. Since it is a prototype, more functions can be added to the system at a later time.

5.3: Selecting the right tools, programming language, frameworks, technologies, and IDEs

The next and important step is to select the right tools, programming language, frameworks, technologies, and Integrated Development Environment. Below I will explain briefly that **what** and **why** I have used to implement the system and I will also be telling **what** I have **not** used and **why not** used. [3]

5.3.1: APIs and Libraries

There are three libraries used in the system, which are jQuery, jQuery UI, and Highcharts JS. jQuery was used as the core JavaScript library to simplify the client-side scripting of HTML and the development of Ajax web application. The reason I choose jQuery is because it is easy to use, fast, small, and feature-rich JavaScript library. Besides, it is the most popular JavaScript library in use today. The library is free and open source software. jQuery includes the following features:

- DOM element selections using the multi-browser open source selector engine
- DOM traversal and modification (including support for CSS 1-3)
- DOM manipulation based on CSS selectors that uses node elements name and node elements attributes (id and class) as criteria to build selectors
- Events
- Effects and animations
- AJAX
- Extensibility through plug-ins
- Utilities - such as user agent information, feature detection
- Compatibility methods that are natively available in modern browsers but need fall backs for older ones - For example the inArray() and each() functions
- Multi-browser (not to be confused with cross-browser) support [4]

The second library that I used is jQuery UI. I used jQuery UI to create the web user interface. jQuery UI is a JavaScript library that provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets, built on top of the jQuery

JavaScript library, that can be used to build interactive web applications. The library includes the following:

Widgets

- **Accordion** – Accordion containers
- **Autocomplete** – Auto-complete boxes based on what the user types
- **Button** – Enhanced button appearance, turn radio buttons and checkboxes into pushbuttons
- **Datepicker** – Advanced date-picker
- **Dialog** – Show dialog boxes on top of other content, easily and robustly
- **Menu** – Show a Menu
- **Progressbar** – Progress bars, both animated and not
- **Slider** – Fully customizable sliders with various features
- **Spinner** – Show a Number Spinner
- **Tabs** – Tabbed user interface handling, with both inline and demand-loaded content
- **Tooltip** – Show a Tooltip

Effects

- **Color Animation** – Animate the transition from one color to another
- **Toggle Class, Add Class, Remove Class, Switch Class** – Animate the transition from one set of styles to another
- **Effect** - A variety of effects (appear, slide-down, explode, fade-in, etc.)
- **Toggle** - Toggle an effect on and off
- **Hide, Show** – Using the effects above

Utilities

- **Position** - Set an element's position relative to another element's position (alignment) [5]

The third library that I used is Highcharts JS. I used Highcharts JS to show charts in the system. Highcharts is a charting library written in pure JavaScript, offering an easy way of adding interactive charts to your website or web application. Highcharts supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, arearange, areasplinerange, columnrange and polar chart types. It works in all modern browsers including the iPhone/iPad and Internet Explorer from version 6. Standard browsers use SVG for the graphics rendering. In legacy Internet Explorer graphics are drawn using VML. It is also free to use for personal website, a school site or a non-profit organization. Since I use it for the

final year project, I can freely use it without the license. It is solely based on native browser technologies and doesn't require client side plugins like Flash or Java. Furthermore, no installation is required on the server. Besides, it also supports jQuery library. There is also exporting module, where users can export the chart to PNG, JPG, PDF or SVG format at the click of a button, or print the chart directly from the web page. [6]

There are many more libraries available in the open source today, besides jQuery, JQuery UI, and Highcharts JS. Some examples are MooTools, Prototype, Dojo, and YUI.

The main reason I choose jQuery over these libraries is because I found out that jQuery is easier to learn and use, and the code is shorter while achieving the same functionality.

5.3.2: Frameworks

The framework that was used in the server-side is Ruby on Rails, which is an open source web framework created in Ruby programming language that's optimized for programmer happiness and sustainable productivity. It lets the programmer write beautiful code by favoring convention over configuration. The framework is the best MVC framework I had ever used compared to ASP.NET MVC 3 (Microsoft), Django (python), and Spring MVC framework (java). Besides, it is also easy to deploy to the OpenShift, which is a free, auto-scaling Platform as a Service (PaaS) for applications. [7] In addition Ruby is also a dynamic and object-oriented programming language.

There are few useful commands in Rails which helps the programmer in the development, which are listed below:

- rails console
- rails new app_name
- rails server
- rails generate
- rails dbconsole
- rake

1. rails console

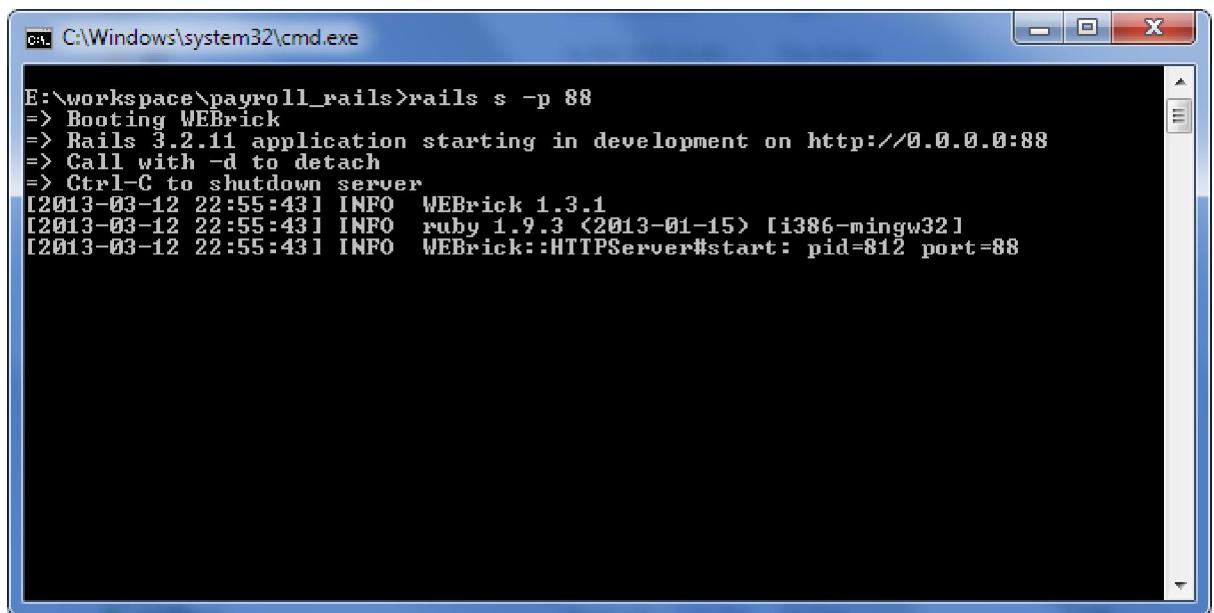
The *console* command lets the programmer interact with the application from the command line. This is useful for testing out quick ideas with code and changing data server-side without touching the website.

2. rails new

It creates a new Rails application.

3. rails server

This command launches a small web server named WEBrick which comes bundled with Ruby. It is needed to access the application through a web browser.

A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The window contains the following text:

```
E:\workspace\payroll_rails>rails s -p 88
=> Booting WEBrick
=> Rails 3.2.11 application starting in development on http://0.0.0.0:88
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2013-03-12 22:55:43] INFO  WEBrick 1.3.1
[2013-03-12 22:55:43] INFO  ruby 1.9.3 <2013-01-15> [i386-mingw32]
[2013-03-12 22:55:43] INFO  WEBrick::HTTPServer#start: pid=812 port=88
```

The window has a standard blue title bar and a black background for the text area.

Figure 16: rails server command

4. rails generate

The *rails generate* command uses templates to create a whole lot of things. Running *rails generate* by itself gives a list of available generators. Using generators will save a large amount of time by writing boilerplate code, code that is necessary for the application to work. The common usage would be *rails generate controller*, which is used to generate controller, and *rails generate model*, which is used to generate data model.

5. rails dbconsole

rails dbconsole figures out which database are currently using and drops the programmer into whichever command line interface where the programmer would use with it.

6. rake

Rake is used for common administration tasks, especially ones that build off of each other. Some common usage would be *rake db:create* (creates the database), *rake*

`db:migrate` (runs database migrations), `rake db:seed` (load seed data into the database), and `rake assets:precompile` (precompile the assets, such as JavaScripts and CSS files).

5.3.3: Technologies

The project uses Ajax technology, with Model-View-Controller design pattern at the server-side, which is Ruby on Rails. The web application uses Single-page application technique to provide a more fluid user experience similar to a desktop application. The client-side uses Ajax to send HTTP request to the server-side, which returns JSON, plain text, and html response. The reason I use JSON instead of XML is because JSON is simpler to understand and it requires less configuration overhead. XML is good for situations in which

- you need message validation
- you're using XSLT
- your messages include a lot of marked-up text
- you need to interoperate with environments that don't support JSON
- you need attributes or namespacing [8]

Example of JSON response:

```
{"menu": {  
  "id": "file",  
  "value": "File",  
  "popup": {  
    "menutitem": [  
      {"value": "New", "onclick": "CreateNewDoc()"},  
      {"value": "Open", "onclick": "OpenDoc()"},  
      {"value": "Close", "onclick": "CloseDoc()"}  
    ]  
  }  
}
```

The same response expressed as XML:

```
<menu id="file" value="File">  
  <popup>  
    <menutitem value="New" onclick="CreateNewDoc()"/>  
    <menutitem value="Open" onclick="OpenDoc()"/>  
    <menutitem value="Close" onclick="CloseDoc()"/>  
  </popup>  
</menu>
```

5.3.4: Integrated Development Environment

The Integrated Development Environment that was used to develop the project is Aptana Studio 3, which is a professional, open source development tool for the open web. The core features include:

HTML, CSS, and JavaScript Code Assist

- Aids in authoring of HTML, CSS, JavaScript, PHP, and Ruby.
- Supports the latest HTML5 specifications.
- Includes information about the level of support for each element in the major web browsers.

Integrated Debugger

- Set breakpoints, inspect variables, and control execution.
- The integrated Ruby & Rails and JavaScript debuggers help debugging easier.

Git Integration

- Easily put projects under git source code control, such as Github.
- Facilitates git-based deployments

Built-in Terminal

- Quickly access a command line terminal for execution of operating system commands and language utilities such as gem, rake, etc. [9]

CHAPTER SIX

TESTING

6.1: Testing in Software Engineering

Testing is an activity that is used to discover errors and correct them, so that we are able to create a defect-free product for our customer or user. Testing is an important phase in the software development life cycle. The objective of testing is to evaluate if we have created the system correctly. During the earlier stages, the focus was to check what is being built but in testing when we have the end product ready, our focus shifts to validate whether the product that has been built has been built correctly or not. Hence, the focus shifts from building the product right to building the right product. [1]

There are two basic types of software testing, which are black box testing and white box testing. General testing process for large system development starts with the testing of individual program units such as functions, classes or objects. These are then integrated into sub-system and systems, and the interactions of these units were tested. Finally after delivery of the system, the customer may carry out a series of acceptance tests to check that the system performs as specified. [2]

Whereas, for smaller system or for system that are developed through scripting or reuse, there are often fewer distinct stages in the process.

The two fundamental testing activities are component testing, testing the parts of the system – and system testing, testing the system as a whole. [2]

6.2: Goals and Types of Testing

Basically, there are two distinct goals of the software testing process:

- To demonstrate to the developer and the customer that the software meets its requirements.
- To discover faults or defects in the software where the behavior of the software is incorrect, undesirable or does not conform to its specification.

The first goal, where you expect the system to perform correctly using a given set of test cases that reflect the systems expected use, leads to validation testing. The second goal leads to defect testing, where the test cases are designed to expose defects. The main types of testing approaches are defined below: [2]

6.2.1: System Testing

System testing involves integrating two or more components that implement system functions or features and then testing this integrated system. For most complex systems, there are two distinct phases to system testing – Integration Testing and Release Testing. As for the Web Based Payroll System, the system had to go through both Integration and Release testing.

6.2.1.1: Integration Testing

Integration testing is mostly concerned with finding defects in the system, where the test team has access to the source code of the system. If the problem is discovered, the team goes through the source code to find the components that have to be debugged. [2]

Integration testing was done after every unit or feature being added to the system. For example, if the current has three features (show payroll chart, overtime chart, total hours worked chart), when the fourth feature, unit, or component is being attached or added to the system, integration testing had to be done throughout although there are still many units or components to be added to the system.

6.2.1.2: Release Testing

In release testing that version of the system is tested that could be released to users or customers. The test team here validates if the system meets its requirements and also ensures system dependability. It is usually black-box testing where the test team is simply concerned with demonstrating the system does or does not work properly. If problems are discovered then they are reported to the development team whose job is to debug the program. Acceptance Testing is key aspect of release testing, where the customers or users are involved in release testing. If the release is good enough, the customer or user may then accept it for use.

After all the units were integrated and combined together to form a complete system, a release test was run to make sure that system's components are not affecting the other components after integrating them. Few feedbacks from the users were collected, according to the feedback the system has to go through the debugging and testing again. [2]

6.2.2: Component Testing

Also known as Unit Testing is the process of testing individual components in the system, to expose faults in these components, and the software developers are responsible for this testing. There are different types of component that may be tested at this stage:

1. Individual functions or methods within an object
2. Object classes that have several attributes and methods
3. Composite components made up of several different objects or function.

These composite components have a defined interface that is used to access their functionality.

The Web Based Payroll System was created in components and units. A new Rails project was created in such a way that each model, view, and controller are organized in separate folder. Below is the screenshot of the project structure.

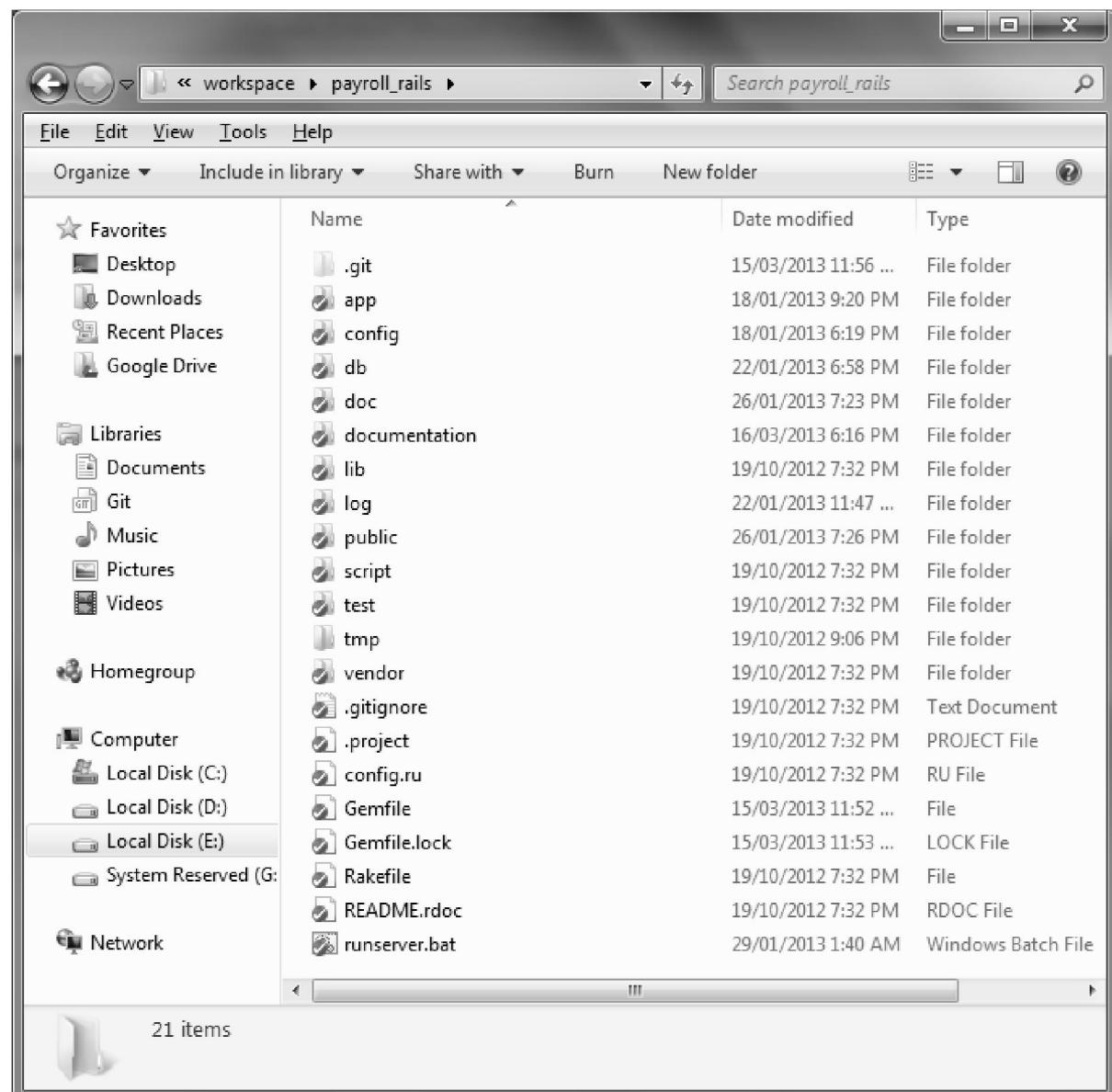


Figure 17: The main project folder

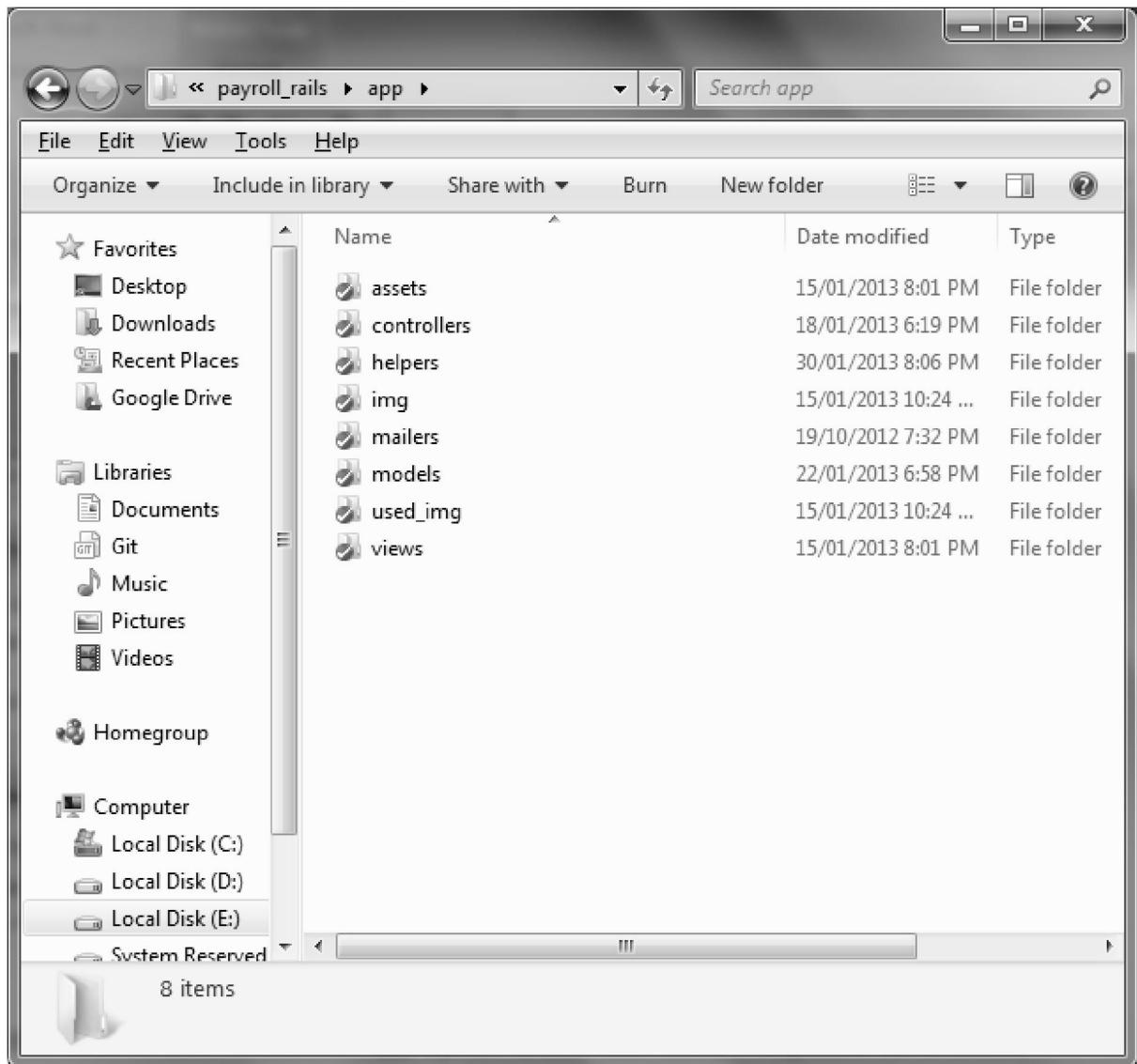
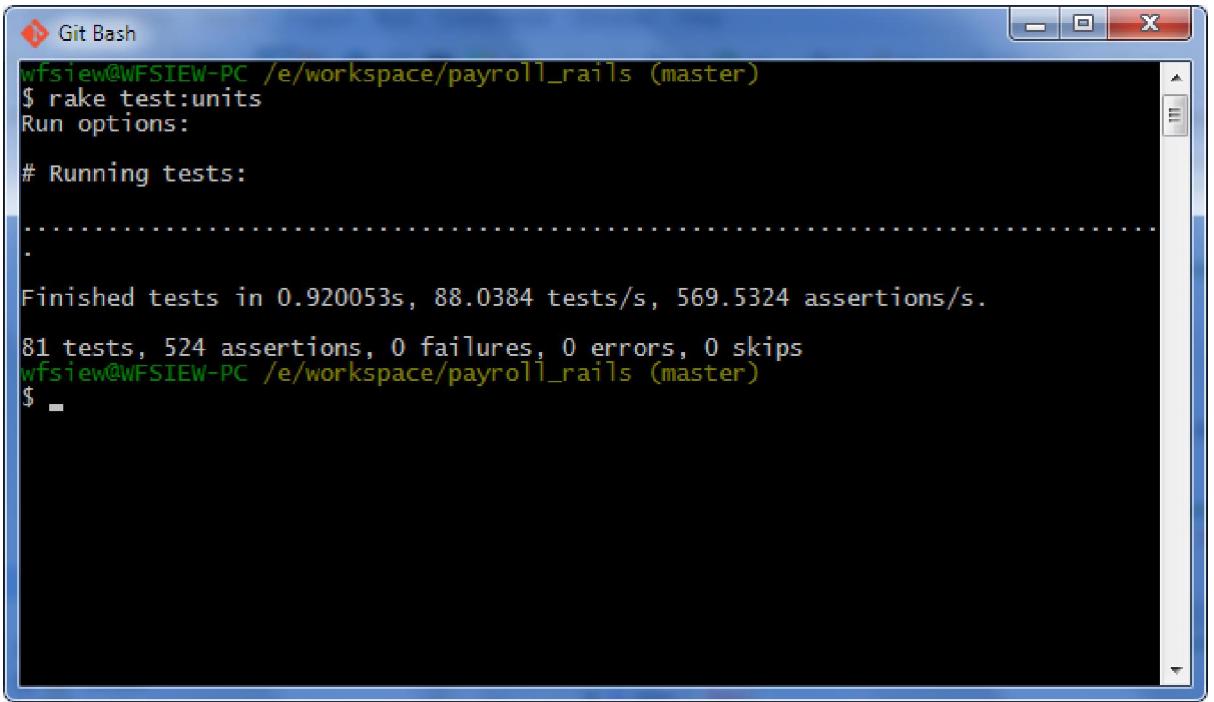


Figure 18: The application folder

Figure above shows the units of the Web Based Payroll System which later were integrated as a whole and was tested using the system testing method.

In Rails, there are three different kinds of tests that can be written, which are unit, functional, and integration test.

- Unit testing tests the models (the command to run the unit test is **rake test:units**).



```
wfsiew@WFSIEW-PC /e/workspace/payroll_rails (master)
$ rake test:units
Run options:

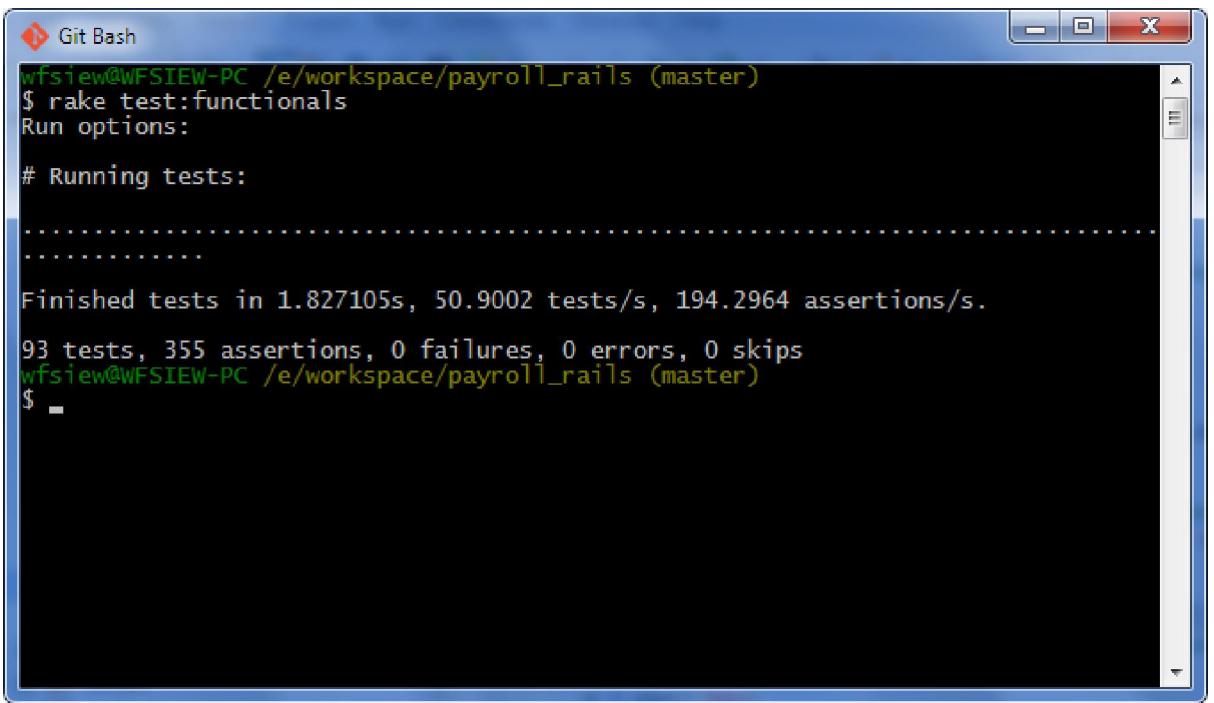
# Running tests:
.....
.

Finished tests in 0.920053s, 88.0384 tests/s, 569.5324 assertions/s.

81 tests, 524 assertions, 0 failures, 0 errors, 0 skips
wfsiew@WFSIEW-PC /e/workspace/payroll_rails (master)
$ -
```

Figure 19: Running unit test in Rails

- Functional testing tests the controllers (the command to run the functional test is **rake test:functionals**).



```
wfsiew@WFSIEW-PC /e/workspace/payroll_rails (master)
$ rake test:functionals
Run options:

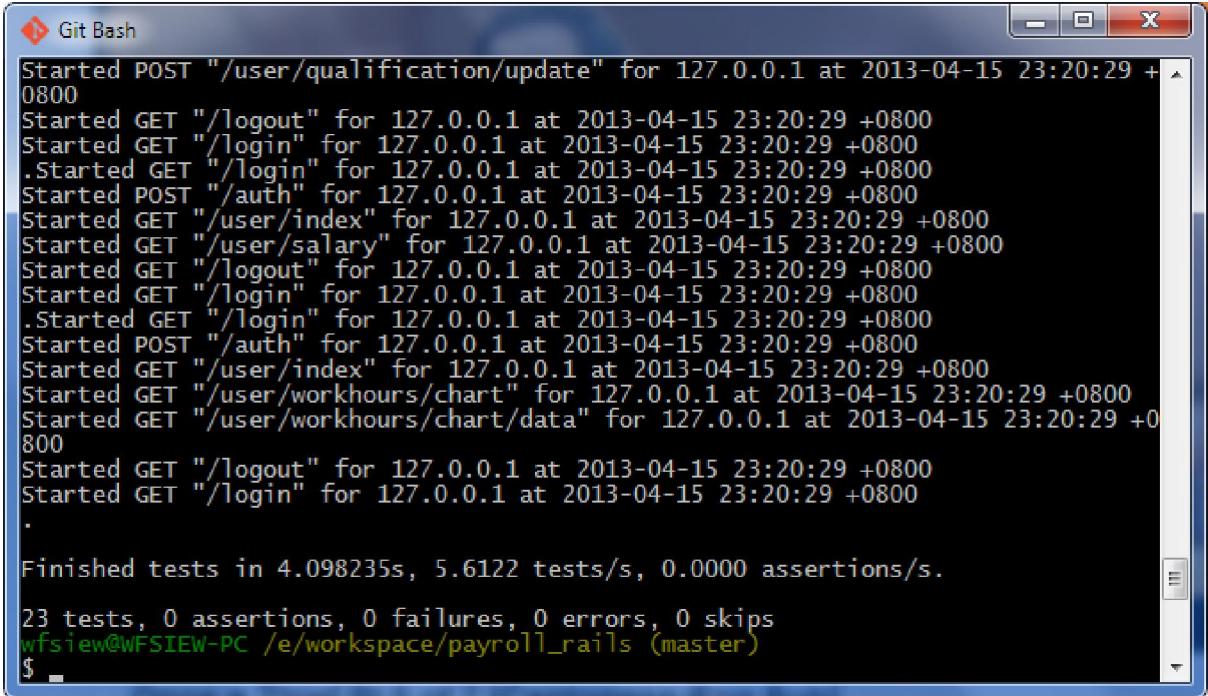
# Running tests:
.....
.

Finished tests in 1.827105s, 50.9002 tests/s, 194.2964 assertions/s.

93 tests, 355 assertions, 0 failures, 0 errors, 0 skips
wfsiew@WFSIEW-PC /e/workspace/payroll_rails (master)
$ -
```

Figure 20: Running functional test in Rails

- Integration testing tests at a high level through multiple controllers (the command to run the integration test is **rake test:integration**).



```

Git Bash
Started POST "/user/qualification/update" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/logout" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/login" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/login" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started POST "/auth" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/user/index" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/user/salary" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/logout" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/login" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/login" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started POST "/auth" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/user/index" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/user/workhours/chart" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/user/workhours/chart/data" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/logout" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
Started GET "/login" for 127.0.0.1 at 2013-04-15 23:20:29 +0800
.
.
.
Finished tests in 4.098235s, 5.6122 tests/s, 0.0000 assertions/s.

23 tests, 0 assertions, 0 failures, 0 errors, 0 skips
wfsiew@WFSIEW-PC /e/workspace/payroll_rails (master)
$ 

```

Figure 21: Running integration test in Rails

Table 6: Five levels of testing

Testing Type	Specification	General Scope	Opacity	Who generally does it?
Unit	Low-Level Design Actual Code Structure	Small unit of code no larger than a class	White Box	Programmer who wrote code
Integration	Low-Level Design High-Level Design	Multiple classes	White Box Black Box	Programmers who wrote code
Functional	High Level Design	Whole product	Black Box	Independent tester
System	Requirements Analysis	Whole product in representative environments	Black Box	Independent tester
Acceptance	Requirements Analysis	Whole product in customer's environment	Black Box	Customer

6.3: Black box testing

Black box testing, also known as functional testing and behavioral testing, focuses on determining whether or not a program does what it is supposed to do based on its functional requirements. Black box testing attempts to find errors in the external behavior of the code in the following categories (1) incorrect or missing functionality; (2) interface errors; (3) errors in data structures used by interfaces; (4) behavior or performance errors; and (5) initialization and termination errors. Through this testing, we can determine if the functions appear to work according to specifications. However, it is important to note that no amount of testing can unequivocally demonstrate the absence of errors and defects the code.

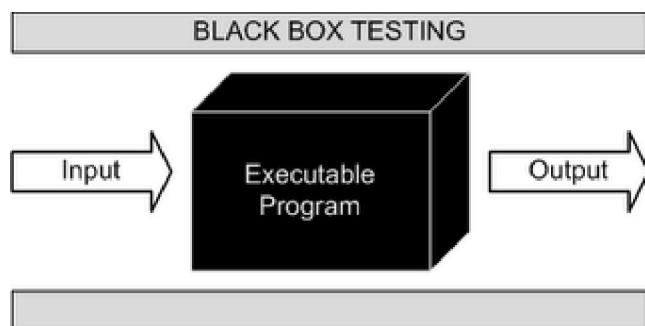


Figure 22: Black Box testing

http://2.bp.blogspot.com/_fOOSCGT3XIw/Sa4cVsX7T7I/AAAAAAAAM/i_YOWD6xHIQ/s320/BlackBoxTesting.gif

6.4: Test Cases

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not.

The format of the test case design is very important. I will use a particular format for the test cases, as shown in Table.

Table 7: Test Case Planning Format

Test ID	Description	Expected Results	Actual Results

Table 8: Test cases for Web Based Payroll System

Test ID	Description	Expected Results	Actual Results
1	Admin user entered the username and password.	Admin user should be redirected to the admin home page.	Admin user redirected to the admin home page. Test case passed
2	Admin user creates a user.	The newly created user should appear in the list.	The newly created user appears in the list. Test case passed
3	Admin user edits a user.	The user should be successfully updated.	The user successfully updated. Test case passed
4	Admin user selects user and delete.	The selected users should be successfully deleted.	The selected users successfully deleted. Test case passed
5	Admin user search user.	The list of users should be displayed based on the search criteria.	The list of users displayed based on the search criteria. Test case passed
6	Admin user creates a job title.	The newly created job title should appear in the list.	The newly created job title appears in the list. Test case passed
7	Admin user edits a job title.	The job title should be successfully updated.	The job title successfully updated. Test case passed
8	Admin user selects job title and delete.	The selected job titles should be successfully deleted.	The selected job titles successfully deleted. Test case passed
9	Admin user search job title.	The list of job titles should be displayed based on the search criteria.	The list of job titles displayed based on the search criteria. Test case passed

10	Admin user creates an employment status.	The newly created employment status should appear in the list.	The newly created employment status appears in the list. Test case passed
11	Admin user edits an employment status.	The employment status should be successfully updated.	The employment status successfully updated. Test case passed
12	Admin user selects employment status and delete.	The selected employment statuses should be successfully deleted.	The selected employment statuses successfully deleted. Test case passed
13	Admin user search employment status.	The list of employment statuses should be displayed based on the search criteria.	The list of employment statuses displayed based on the search criteria. Test case passed
14	Admin user creates a job category.	The newly created job category should appear in the list.	The newly created job category appears in the list. Test case passed
15	Admin user edits a job category.	The job category should be successfully updated.	The job category successfully updated. Test case passed
16	Admin user selects job category and delete.	The selected job categories should be successfully deleted.	The selected job categories successfully deleted. Test case passed
17	Admin user search job category.	The list of job categories should be displayed based on the search criteria.	The list of job categories displayed based on the search criteria. Test case passed
18	Admin user creates a department.	The newly created department should appear in the list.	The newly created department appears in the list.

			Test case passed
19	Admin user edits a department.	The department should be successfully updated.	The department successfully updated. Test case passed
20	Admin user selects department and delete.	The selected departments should be successfully deleted.	The selected departments successfully deleted. Test case passed
21	Admin user search department.	The list of departments should be displayed based on the search criteria.	The list of departments displayed based on the search criteria. Test case passed
22	Admin user creates a employee.	The newly created employee should appear in the list.	The newly created employee appears in the list. Test case passed
23	Admin user edits an employee.	The employee should be successfully updated.	The employee successfully updated. Test case passed
24	Admin user selects employee and delete.	The selected employees should be successfully deleted.	The selected employees successfully deleted. Test case passed
25	Admin user search employee.	The list of employees should be displayed based on the search criteria.	The list of employees displayed based on the search criteria. Test case passed
26	Admin user search employee's attendance.	The list of attendance should be displayed based on the search criteria.	The list of attendance displayed based on the search criteria. Test case passed
27	Admin user creates a pay rate.	The newly created pay rate should appear in the list.	The newly created pay rate appears in the list. Test case passed

28	Admin user edits a pay rate.	The pay rate should be successfully updated.	The pay rate successfully updated. Test case passed
29	Admin user selects pay rate and delete.	The selected pay rates should be successfully deleted.	The selected pay rates successfully deleted. Test case passed
30	Admin user search pay rate.	The list of pay rates should be displayed based on the search criteria.	The list of pay rates displayed based on the search criteria. Test case passed
31	Admin user creates an overtime pay rate.	The newly created overtime pay rate should appear in the list.	The newly created overtime pay rate appears in the list. Test case passed
32	Admin user edits an overtime pay rate.	The overtime pay rate should be successfully updated.	The overtime pay rate successfully updated. Test case passed
33	Admin user selects overtime pay rate and delete.	The selected overtime pay rates should be successfully deleted.	The selected overtime pay rates successfully deleted. Test case passed
34	Admin user search overtime pay rate.	The list of overtime pay rates should be displayed based on the search criteria.	The list of overtime pay rates displayed based on the search criteria. Test case passed
35	Admin user creates a salary adjustment.	The newly created salary adjustment should appear in the list.	The newly created salary adjustment appears in the list. Test case passed
36	Admin user edits a salary adjustment.	The salary adjustment should be successfully updated.	The salary adjustment successfully updated. Test case passed

37	Admin user selects salary adjustment and delete.	The selected salary adjustments should be successfully deleted.	The selected salary adjustments successfully deleted. Test case passed
38	Admin user search salary adjustment.	The list of salary adjustments should be displayed based on the search criteria.	The list of salary adjustments displayed based on the search criteria. Test case passed
39	Admin user view all employees overtime chart.	The chart should be displayed.	The chart is displayed. Test case passed
40	Admin user generates filtered overtime chart.	The chart should be displayed based on the filtered criteria.	The chart is displayed based on the filtered criteria. Test case passed
41	Admin user view all hourly paid employees total work hours chart.	The chart should be displayed.	The chart is displayed. Test case passed
42	Admin user generates filtered total work hours chart.	The chart should be displayed based on the filtered criteria.	The chart is displayed based on the filtered criteria. Test case passed
43	Admin user view all employees hourly payroll chart.	The chart should be displayed.	The chart is displayed. Test case passed
44	Admin user generates filtered hourly payroll chart.	The chart should be displayed based on the filtered criteria.	The chart is displayed based on the filtered criteria. Test case passed
45	Admin user view all employees payslip.	The payslip should be opened in a window.	The payslip is opened in a window. Test case passed
46	Admin user search employee and view selected employee's payslip.	The list of employees should be displayed based on the search	The list of employees displayed based on the search criteria

		criteria and the payslip should be opened in a window.	and the payslip is opened in a window. Test case passed
47	Admin user clicks Logout button.	Admin user should be redirected to the login page.	Admin user redirected to the login page. Test case passed
48	Hourly paid employee entered the username and password.	Hourly paid employee should be redirected to the hourly paid employee home page.	Hourly paid employee redirected to the hourly paid employee home page. Test case passed
49	Hourly paid employee edits personal details.	The details should be successfully updated.	The details successfully updated. Test case passed
50	Hourly paid employee edits contact details.	The details should be successfully updated.	The details successfully updated. Test case passed
51	Hourly paid employee view job details.	The job details should be displayed.	The job details displayed. Test case passed
52	Hourly paid employee view salary details.	The salary details should be displayed.	The salary details displayed. Test case passed
53	Hourly paid employee edits qualification details.	The details should be successfully updated.	The details successfully updated. Test case passed
54	Hourly paid employee view total work hours chart.	The chart should be displayed.	The chart is displayed. Test case passed
55	Hourly paid employee generates filtered total work hours chart.	The chart should be displayed based on the filtered criteria.	The chart is displayed based on the filtered criteria. Test case passed
56	Hourly paid employee view hourly	The chart should be	The chart is

	payroll chart.	displayed.	displayed. Test case passed
57	Hourly paid employee generates filtered hourly payroll chart.	The chart should be displayed based on the filtered criteria.	The chart is displayed based on the filtered criteria. Test case passed
58	Hourly paid employee view payslip for selected month and year.	The payslip should be opened in a window.	The payslip is opened in a window. Test case passed
59	Hourly paid employee clicks Logout button.	Hourly paid employee should be redirected to the login page.	Hourly paid employee redirected to the login page. Test case passed
60	Monthly paid employee entered the username and password.	Monthly paid employee should be redirected to the monthly paid employee home page.	Monthly paid employee redirected to the monthly paid employee home page. Test case passed
61	Monthly paid employee edits personal details.	The details should be successfully updated.	The details successfully updated. Test case passed
62	Monthly paid employee edits contact details.	The details should be successfully updated.	The details successfully updated. Test case passed
63	Monthly paid employee view job details.	The job details should be displayed.	The job details displayed. Test case passed
64	Monthly paid employee view salary details.	The salary details should be displayed.	The salary details displayed. Test case passed
65	Monthly paid employee edits qualification details.	The details should be successfully updated.	The details successfully updated. Test case passed
66	Monthly paid employee view total work hours chart.	The chart should be displayed.	The chart is displayed.

			Test case passed
67	Monthly paid employee generates filtered total work hours chart.	The chart should be displayed based on the filtered criteria.	The chart is displayed based on the filtered criteria. Test case passed
68	Monthly paid employee view overtime chart.	The chart should be displayed.	The chart is displayed. Test case passed
69	Monthly paid employee generates filtered overtime chart.	The chart should be displayed based on the filtered criteria.	The chart is displayed based on the filtered criteria. Test case passed
70	Monthly paid employee view payslip for selected month and year.	The payslip should be opened in a window.	The payslip is opened in a window. Test case passed
71	Monthly paid employee clicks Logout button.	Monthly paid employee should be redirected to the login page.	Monthly paid employee redirected to the login page. Test case passed

71 test cases were developed to accomplish the Black Box testing for the Web Based Payroll System.

CHAPTER SEVEN

FUTURE WORK/MAINTENANCE

As for the future work on the Web Based Payroll System, few tasks, ideas, features, and functionalities are queued up to be added in the system. Not all the Payroll features have been added to the Web Based Payroll System but can be added in the future to the system. The source code of the project can be accessed at <https://github.com/wfsview/Payroll>, and the web application can be accessed at <https://payroll-wfsviewapp.rhcloud.com>

7.1: Recommendations and Features

Not all the Payroll features has been added to the Web Based Payroll System as the system is the proof of concept and does not encompass the full features. More time and developing team are required to develop the Web Based Payroll System with the full features of the Payroll.

The features that are going to be added to the Web Based Payroll System in the future are the following:

7.1.1: Leave Management System

When an employee applies for leave he/she can know how many leaves are in balance (Annual Leave, Emergency Leave, Sick Leave, etc) through the Leave Management System. During Leave Application, the system captures the following data:

- Leave Application Date
- Leave type, such as Annual Leave, Sick Leave (shows the Balance Leaves to the employee while applying)
- Leave Period i.e. the total number of days for which the leave is applied
- Reason for applying the Leave
- Address and Contact Number during Leave, in case of Emergency purposes

The Leave which the employee has applied for, can be Approved fully or partly.

The details given in the leave application are linked to the payroll system and the employee's salary is generated in accordance with his attendance.

7.1.2: Time Management System

Employees can be assigned to different shifts. The management can define Shifts and the following data is captured:

- Shift name
- Shift number
- Shift period
- Shift Start timing and End timing
- Late coming limit and Late sitting limit
- Overtime in minutes
- Break time (Lunch, Breakfast, etc)
- Description related to Shift
- Weekly off (can be any day in the week)

The time management takes care of physical absence due to travel undertaken for official purposes (in case of marketing personnel, etc).

7.1.3: Import/Export

Allow administrator to import data to the system from csv file, or excel file, and export the data to csv file, or excel file.

References

- [1] "The Benefits of Web Based Applications and Systems," 2007. [Online]. Available: <http://www.dbnetsolutions.co.uk/Articles/BenefitsOfWebBasedApplications.aspx>. [Accessed March 2013].
- [2] N. Grace, "An Introduction to Payroll Systems," [Online]. Available: http://www.ehow.com/info_7771494_introduction-payroll-systems.html. [Accessed March 2013].
- [3] Wikipedia, "Ajax (programming) - Wikipedia, the free encyclopedia," March 2013. [Online]. Available: [http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming)). [Accessed March 2013].
- [4] J. J. Garrett, "Ajax: A New Approach to Web Applications - Adaptive Path," [Online]. Available: <http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications>. [Accessed March 2013].
- [5] S. R. Pawar, "Importance of Ajax in web applications - Tutorial," [Online]. Available: http://www.javacertificate.net/ajax_article.php. [Accessed March 2013].
- [6] "Technologies Used in AJAX," [Online]. Available: http://www.tutorialspoint.com/ajax/ajax_technology.htm. [Accessed March 2013].
- [7] Wikipedia, "Web application - Wikipedia, the free encyclopedia," [Online]. Available: http://en.wikipedia.org/wiki/Web_application. [Accessed March 2013].
- [8] Wikipedia, "Single-page application - Wikipedia, the free encyclopedia," [Online]. Available: http://en.wikipedia.org/wiki/Single-page_application. [Accessed March 2013].
- [9] D. Roos, "HowStuffWorks "What is a Payroll System?"," [Online]. Available: <http://money.howstuffworks.com/payroll-system1.htm>. [Accessed March 2013].
- [10] Wikipedia, "Model–view–controller - Wikipedia, the free encyclopedia," [Online]. Available: <http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>. [Accessed March 2013].
- [11] Wikipedia, "Requirements analysis - Wikipedia, the free encyclopedia," [Online]. Available: http://en.wikipedia.org/wiki/Requirements_analysis. [Accessed March 2013].
- [12] Wikipedia, "Software prototyping - Wikipedia, the free encyclopedia," [Online]. Available: http://en.wikipedia.org/wiki/Software_prototyping. [Accessed March 2013].
- [13] Wikipedia, "Software requirements specification - Wikipedia, the free encyclopedia," [Online]. Available: http://en.wikipedia.org/wiki/Software_requirements_specification. [Accessed March 2013].

2013].

- [14] Wikipedia, "Operating environment - Wikipedia, the free encyclopedia," [Online]. Available: http://en.wikipedia.org/wiki/Operating_environment. [Accessed March 2013].
- [15] jQuery, "jQuery," [Online]. Available: <http://jquery.com/>. [Accessed March 2013].
- [16] jQuery UI, "jQuery UI," [Online]. Available: <http://jqueryui.com/>. [Accessed March 2013].
- [17] Highcharts JS, "Highcharts - Interactive JavaScript charts for your webpage," [Online]. Available: <http://www.highcharts.com/>. [Accessed March 2013].
- [18] Wikipedia, "Functional requirement - Wikipedia, the free encyclopedia," [Online]. Available: http://en.wikipedia.org/wiki/Functional_requirements. [Accessed March 2013].
- [19] A. A. & C. Greenidge, March 2013. [Online]. Available: http://www.cavehill.uwi.edu/staff/eportfolios/paulwalcott/courses/comp2145/2009/design_-_concepts_and_principles.htm.
- [20] R. L. Burback, "The Implementation Phase," 14 December 1998. [Online]. Available: <http://infolab.stanford.edu/~burback/watersluice/node17.html>. [Accessed March 2013].
- [21] Wikipedia, "Implementation - Wikipedia, the free encyclopedia," [Online]. Available: <http://en.wikipedia.org/wiki/Implementation>. [Accessed March 2013].
- [22] A. L. Tharp, "Selecting the “right” programming language," 1982.
- [23] OpenShift, "OpenShift by Red Hat," [Online]. Available: <https://openshift.redhat.com/app/>. [Accessed March 2013].
- [24] P. Bergantino, "When would you use XML over JSON for Ajax? - Stack Overflow," January 2009. [Online]. Available: <http://stackoverflow.com/questions/478855/when-would-you-use-xml-over-json-for-ajax>. [Accessed March 2013].
- [25] aptana, "Aptana | Studio," [Online]. Available: <http://www.aptana.com/products/studio3>. [Accessed March 2013].
- [26] angad, "Testing In Software Engineering," 2 November 2012. [Online]. Available: <http://techforum4u.com/content.php/417-Testing-In-Software-Engineering>. [Accessed March 2013].
- [27] A. Bertolino, "Software testing research and practice," 2003.

APPENDIX A

COMPLETE SOURCE CODE

```

E:\workspace\payroll_rails\config\application.rb

require File.expand_path('../boot', __FILE__)
require 'rails/all'

if defined?(Bundler)
  # If you precompile assets before deploying to production, use this line
  Bundler.require(*Rails.groups(:assets => %w(development test)))
  # If you want your assets lazily compiled in production, use this line
  # Bundler.require(:default, :assets, Rails.env)
end

module Payroll
  class Application < Rails::Application
    # Settings in config/environments/* take precedence over those specified here.
    # Application configuration should go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded.

    # Custom directories with classes and modules you want to be autoloadable.
    # config.autoload_paths += %W({config.root}/extras)

    # Only load the plugins named here, in the order given (default is alphabetical).
    # :all can be used as a placeholder for all plugins not explicitly named.
    # config.plugins = [ :exception_notification, :ssl_requirement, :all ]

    # Activate observers that should always be running.
    # config.active_record.observers = :cacher, :garbage_collector, :forum_observer

    # Set Time.zone default to the specified zone and make Active Record auto-convert to
    # this zone.
    # Run "rake -D time" for a list of tasks for finding time zone names. Default is UTC.
    # config.time_zone = 'Central Time (US & Canada)'
    config.time_zone = 'Kuala Lumpur'

    # The default locale is :en and all translations from config/locales/*.rb,yml are auto
    # loaded.
    # config.i18n.load_path += Dir[Rails.root.join('my', 'locales', '*.{rb,yml}').to_s]
    # config.i18n.default_locale = :de

    # Configure the default encoding used in templates for Ruby 1.9.
    config.encoding = "utf-8"

    # Configure sensitive parameters which will be filtered from the log file.
    config.filter_parameters += [:password]

    # Enable escaping HTML in JSON.
    config.active_support.escape_html_entities_in_json = true

    # Use SQL instead of Active Record's schema dumper when creating the database.
    # This is necessary if your schema can't be completely dumped by the schema dumper,
    # like if you have constraints or database-specific column types
    # config.active_record.schema_format = :sql

    # Enforce whitelist mode for mass assignment.
    # This will create an empty whitelist of attributes available for mass-assignment for
    # all models in your app. As such, your models will need to explicitly whitelist or
    # blacklist accessible parameters by using an attr_accessible or attr_protected
    # declaration.
    config.active_record.whitelist_attributes = true

    # Enable the asset pipeline
    config.assets.enabled = true

    # Version of your assets, change this if you want to expire all your assets
    config.assets.version = '1.0'

    config.assets.precompile += %w(loginui.css _payslip.css _payslip.js
      dark-hive/jquery-ui-1.10.2.custom.min.css admin.css admin.js user.css user.js)

    jquitheme = %w(blitzer dark-hive trontastic humanity)
    jquicss = 'jquery-ui-1.10.2.custom.min.css'

    jquitheme.each do |t|
      config.assets.precompile << "#{t}/#{jquicss}"
    end
  end

```

```

end

E:\workspace\payroll_rails\config\database.yml

# MySQL. Versions 4.1 and 5.0 are recommended.
#
# Install the MySQL driver
#   gem install mysql2
#
# Ensure the MySQL gem is defined in your Gemfile
#   gem 'mysql2'
#
# And be sure to use new-style password hashing:
#   http://dev.mysql.com/doc/refman/5.0/en/old-client.html
development:
  adapter: mysql2
  encoding: utf8
  reconnect: false
  database: payroll_development
  pool: 5
  username: root
  password: root
  host: 127.0.0.1
  port: 3307

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
  adapter: mysql2
  encoding: utf8
  reconnect: false
  database: payroll_test
  pool: 5
  username: root
  password: root
  host: 127.0.0.1
  port: 3307

production:
  adapter: mysql2
  encoding: utf8
  reconnect: false
  database: payroll_development
  pool: 5
  username: root
  password: root
  host: 127.0.0.1
  port: 3307

E:\workspace\payroll_rails\config\environment.rb

# Load the rails application
require File.expand_path('../application', __FILE__)

# Initialize the rails application
Payroll::Application.initialize!

Rails.logger = Logger.new(STDOUT)

E:\workspace\payroll_rails\config\routes.rb

Payroll::Application.routes.draw do
  root :to => 'admin/admin#index'

  match 'login' => 'application#new', :as => :login
  match 'auth' => 'application#create', :as => :auth
  match 'logout' => 'application#destroy', :as => :logout

  namespace :admin do
    match 'index' => 'admin#index', :as => :index, :via => :get

    scope 'user', :as => 'user' do
      match '' => 'user#index', :via => :get
      match 'list' => 'user#list', :as => :list, :via => [:get, :post]
      match 'new' => 'user#new', :as => :new, :via => :get
      match 'create' => 'user#create', :as => :create, :via => :post
    end
  end
end

```

```

match 'edit(/:id)' => 'user#edit', :as => :edit, :via => :get
match 'update(/:id)' => 'user#update', :as => :update, :via => :post
match 'delete' => 'user#destroy', :as => :delete, :via => :post
end

scope 'employee', :as => 'employee' do
  match '' => 'employee#index', :via => :get
  match 'list' => 'employee#list', :via => [:get, :post]
  match 'new' => 'employee#new', :as => :new, :via => :get
  match 'create' => 'employee#create', :as => :create, :via => :post
  match 'edit(/:id)' => 'employee#edit', :as => :edit, :via => :get
  match 'update(/:id)' => 'employee#update', :as => :update, :via => :post
  match 'delete' => 'employee#destroy', :as => :delete, :via => :post
end

scope 'designation', :as => 'designation' do
  match '' => 'designation#index', :via => :get
  match 'list' => 'designation#list', :as => :list, :via => [:get, :post]
  match 'new' => 'designation#new', :as => :new, :via => :get
  match 'create' => 'designation#create', :as => :create, :via => :post
  match 'edit(/:id)' => 'designation#edit', :as => :edit, :via => :get
  match 'update(/:id)' => 'designation#update', :as => :update, :via => :post
  match 'delete' => 'designation#destroy', :as => :delete, :via => :post
end

scope 'empstatus', :as => 'empstatus' do
  match '' => 'employment_status#index', :via => :get
  match 'list' => 'employment_status#list', :as => :list, :via => [:get, :post]
  match 'new' => 'employment_status#new', :as => :new, :via => :get
  match 'create' => 'employment_status#create', :as => :create, :via => :post
  match 'edit(/:id)' => 'employment_status#edit', :as => :edit, :via => :get
  match 'update(/:id)' => 'employment_status#update', :as => :update, :via => :post
  match 'delete' => 'employment_status#destroy', :as => :delete, :via => :post
end

scope 'jobcat', :as => 'jobcat' do
  match '' => 'job_category#index', :via => :get
  match 'list' => 'job_category#list', :as => :list, :via => [:get, :post]
  match 'new' => 'job_category#new', :as => :new, :via => :get
  match 'create' => 'job_category#create', :as => :create, :via => :post
  match 'edit(/:id)' => 'job_category#edit', :as => :edit, :via => :get
  match 'update(/:id)' => 'job_category#update', :as => :update, :via => :post
  match 'delete' => 'job_category#destroy', :as => :delete, :via => :post
end

scope 'dept', :as => 'dept' do
  match '' => 'department#index', :via => :get
  match 'list' => 'department#list', :as => :list, :via => [:get, :post]
  match 'new' => 'department#new', :as => :new, :via => :get
  match 'create' => 'department#create', :as => :create, :via => :post
  match 'edit(/:id)' => 'department#edit', :as => :edit, :via => :get
  match 'update(/:id)' => 'department#update', :as => :update, :via => :post
  match 'delete' => 'department#destroy', :as => :delete, :via => :post
end

scope 'payrate', :as => 'payrate' do
  match '' => 'pay_rate#index', :via => :get
  match 'list' => 'pay_rate#list', :as => :list, :via => [:get, :post]
  match 'new' => 'pay_rate#new', :as => :new, :via => :get
  match 'create' => 'pay_rate#create', :as => :create, :via => :post
  match 'edit(/:id)' => 'pay_rate#edit', :as => :edit, :via => :get
  match 'update(/:id)' => 'pay_rate#update', :as => :update, :via => :post
  match 'delete' => 'pay_rate#destroy', :as => :delete, :via => :post
end

scope 'hourly', :as => 'hourly' do

  match 'chart' => 'hourly_payroll_chart#index', :via => :get
  match 'chart/data' => 'hourly_payroll_chart#data', :via => [:get, :post]
end

scope 'payslip', :as => 'payslip' do
  match '' => 'payslip#index', :via => :get
  match 'list' => 'payslip#list', :via => [:get, :post]
  match 'slip(/:id(/:month(/:year)))' => 'payslip#payslip', :as => :slip, :via => :get
end

```

```

scope 'att', :as => 'att' do
  match '' => 'attendance#index', :via => :get
  match 'list' => 'attendance#list', :via => [:get, :post]
end

scope 'overtime', :as => 'overtime' do
  scope 'rate', :as => 'rate' do
    match '' => 'overtime_rate#index', :via => :get
    match 'list' => 'overtime_rate#list', :as => :list, :via => [:get, :post]
    match 'new' => 'overtime_rate#new', :as => :new, :via => :get
    match 'create' => 'overtime_rate#create', :as => :create, :via => :post
    match 'edit(/:id)' => 'overtime_rate#edit', :as => :edit, :via => :get
    match 'update(/:id)' => 'overtime_rate#update', :as => :update, :via => :post
    match 'delete' => 'overtime_rate#destroy', :as => :delete, :via => :post
  end

  match 'chart' => 'overtime_chart#index', :via => :get
  match 'chart/data' => 'overtime_chart#data', :via => [:get, :post]
end

scope 'workhours', :as => 'workhours' do
  match 'chart' => 'total_work_hours_chart#index', :via => :get
  match 'chart/data' => 'total_work_hours_chart#data', :via => [:get, :post]
end

scope 'salaryadj', :as => 'salaryadj' do
  match '' => 'salary_adjustment#index', :via => :get
  match 'list' => 'salary_adjustment#list', :as => :list, :via => [:get, :post]
  match 'new' => 'salary_adjustment#new', :as => :new, :via => :get
  match 'create' => 'salary_adjustment#create', :as => :create, :via => :post
  match 'edit(/:id)' => 'salary_adjustment#edit', :as => :edit, :via => :get
  match 'update(/:id)' => 'salary_adjustment#update', :as => :update, :via => :post
  match 'delete' => 'salary_adjustment#destroy', :as => :delete, :via => :post
end
end

namespace :user do
  match 'index' => 'user#index', :as => :index, :via => :get

  scope 'info', :as => 'info' do
    match '' => 'info#index', :via => :get
    match 'update' => 'info#update', :via => :post
  end

  scope 'contact', :as => 'contact' do
    match '' => 'contact#index', :via => :get
    match 'update' => 'contact#update', :via => :post
  end

  scope 'job', :as => 'job' do
    match '' => 'job#index', :via => :get
  end

  scope 'salary', :as => 'salary' do
    match '' => 'salary#index', :via => :get
  end

  scope 'qualification', :as => 'qualification' do
    match '' => 'qualification#index', :via => :get
    match 'update' => 'qualification#update', :via => :post
  end

  scope 'overtime', :as => 'overtime' do
    match 'chart' => 'overtime_chart#index', :via => :get
    match 'chart/data' => 'overtime_chart#data', :via => [:get, :post]
  end

  scope 'workhours', :as => 'workhours' do
    match 'chart' => 'total_work_hours_chart#index', :via => :get
    match 'chart/data' => 'total_work_hours_chart#data', :via => [:get, :post]
  end

  scope 'hourly', :as => 'hourly' do
    match 'chart' => 'hourly_payroll_chart#index', :via => :get
    match 'chart/data' => 'hourly_payroll_chart#data', :via => [:get, :post]
  end
end

```

```

scope 'payslip', :as => 'payslip' do
  match '' => 'payslip#index', :via => :get
  match 'slip(/:month(/:year))' => 'payslip#payslip', :as => :slip, :via => :get
end
end

# The priority is based upon order of creation:
# first created -> highest priority.

# Sample of regular route:
#   match 'products/:id' => 'catalog#view'
# Keep in mind you can assign values other than :controller and :action

# Sample of named route:
#   match 'products/:id/purchase' => 'catalog#purchase', :as => :purchase
# This route can be invoked with purchase_url(:id => product.id)

# Sample resource route (maps HTTP verbs to controller actions automatically):
#   resources :products

# Sample resource route with options:
#   resources :products do
#     member do
#       get 'short'
#       post 'toggle'
#     end
#   #
#   collection do
#     get 'sold'
#   end
# end

# Sample resource route with sub-resources:
#   resources :products do
#     resources :comments, :sales
#     resource :seller
#   end

# Sample resource route with more complex sub-resources
#   resources :products do
#     resources :comments
#     resources :sales do
#       get 'recent', :on => :collection
#     end
#   end
# end

# Sample resource route within a namespace:
#   namespace :admin do
#     # Directs /admin/products/* to Admin::ProductsController
#     # (app/controllers/admin/products_controller.rb)
#     resources :products
#   end

# You can have the root of your site routed with "root"
# just remember to delete public/index.html.
# root :to => 'welcome#index'

# See how all your routes lay out with "rake routes"

# This is a legacy wild controller route that's not recommended for RESTful applications.
# Note: This route will make all actions in every controller accessible via GET requests.
# match ':controller(/:action(/:id))(.:format)'
end

```

```

E:\workspace\payroll_rails\app\models\attendance.rb

# Model for attendance table.
class Attendance < ActiveRecord::Base
attr_accessible :id, :staff_id, :time_in, :time_out, :work_date

self.table_name = 'attendance'

belongs_to :employee, :foreign_key => 'staff_id', :primary_key => 'staff_id'

validates_presence_of :work_date, :message => 'Work date is required'
validates_presence_of :time_in, :message => 'Time in is required'
validates_presence_of :time_out, :message => 'Time out is required'
end

E:\workspace\payroll_rails\app\models\department.rb

# Model for department table.
class Department < ActiveRecord::Base
attr_accessible :id, :name

self.table_name = 'department'

validates_presence_of :name, :message => 'Name is required'
validates_uniqueness_of :name, :message => "Department %{value} already exist"
end

E:\workspace\payroll_rails\app\models\designation.rb

# Model for designation table.
class Designation < ActiveRecord::Base
attr_accessible :id, :title, :desc, :note

self.table_name = 'designation'

has_many :employee_job

validates_presence_of :title, :message => 'Job Title is required'
validates_uniqueness_of :title, :message => "Job Title %{value} already exist"
end

E:\workspace\payroll_rails\app\models\employee.rb

# Model for employee table.
class Employee < ActiveRecord::Base
attr_accessible :dob, :staff_id, :user_id, :first_name, :gender, :id, :is_bumi,
                :last_name, :marital_status, :middle_name, :nationality, :new_ic,
                :old_ic, :passport_no, :place_of_birth, :race, :religion

self.table_name = 'employee'

has_one :employee_contact, :foreign_key => 'id'
has_one :employee_job, :foreign_key => 'id'
has_one :employee_salary, :foreign_key => 'id'
has_one :employee_qualification, :foreign_key => 'id'
has_one :employee, :foreign_key => 'id'
has_many :attendance, :foreign_key => 'staff_id', :primary_key => 'staff_id'
belongs_to :user

validates_presence_of :staff_id, :message => 'Employee ID is required'
validates_presence_of :first_name, :message => 'First Name is required'
validates_presence_of :last_name, :message => 'Last Name is required'
validates_presence_of :new_ic, :message => 'New IC No. is required'
validates_presence_of :gender, :message => 'Gender is required'
validates_presence_of :marital_status, :message => 'Marital Status is required'
validates_presence_of :nationality, :message => 'Nationality is required'
validates_presence_of :dob, :message => 'Date of Birth is required'
validates_presence_of :place_of_birth, :message => 'Place of Birth is required'
validates_presence_of :race, :message => 'Race is required'

validates_uniqueness_of :staff_id, :message => "Employee ID %{value} already exist"
end

```

```

E:\workspace\payroll_rails\app\models\employee_contact.rb

# Model for employee contact table.
class EmployeeContact < ActiveRecord::Base
  attr_accessible :address_1, :address_2, :address_3, :city, :country, :home_phone, :id,
                  :mobile_phone, :other_email, :postcode, :state, :work_email

  self.table_name = 'employee_contact'

  belongs_to :employee, :foreign_key => 'id'

  validates_presence_of :address_1, :message => 'Address 1 is required'
  validates_presence_of :city, :message => 'City is required'
  validates_presence_of :state, :message => 'State is required'
  validates_presence_of :postcode, :message => 'Postal Code is required'
  validates_presence_of :country, :message => 'Country is required'
  validates_presence_of :work_email, :message => 'Work Email is required'
end

E:\workspace\payroll_rails\app\models\employee_job.rb

# Model for employee_job table.
class EmployeeJob < ActiveRecord::Base
  attr_accessible :confirm_date, :department_id, :designation_id, :employment_status_id,
                  :id, :job_category_id, :join_date

  self.table_name = 'employee_job'

  belongs_to :employee, :foreign_key => 'id'
  belongs_to :designation
  belongs_to :department
  belongs_to :employment_status
  belongs_to :job_category

  validates_presence_of :designation_id, :message => 'Designation is required'
  validates_presence_of :department_id, :message => 'Department is required'
  validates_presence_of :employment_status_id, :message => 'Employment Status is required'
  validates_presence_of :job_category_id, :message => 'Job Category is required'
  validates_presence_of :join_date, :message => 'Join Date is required'
end

E:\workspace\payroll_rails\app\models\employee_qualification.rb

# Model for employee qualification table.
class EmployeeQualification < ActiveRecord::Base
  attr_accessible :end_date, :gpa, :id, :institute, :level, :major, :start_date, :year

  self.table_name = 'employee_qualification'

  belongs_to :employee, :foreign_key => 'id'

  validates_presence_of :level, :message => 'Qualification Level is required'
  validates_presence_of :institute, :message => 'Institute name is required'
  validates_presence_of :year, :message => 'Year obtained is required'
  validates_presence_of :start_date, :message => 'Start Date is required'
  validates_presence_of :end_date, :message => 'End Date is required'
end

E:\workspace\payroll_rails\app\models\employee_salary.rb

# Model for employee salary table.
class EmployeeSalary < ActiveRecord::Base
  attr_accessible :allowance, :bank_acc_no, :bank_acc_type, :bank_address, :bank_name,
                  :epf, :epf_no, :id, :income_tax, :income_tax_no, :salary, :socso_no,
                  :pay_type, :socso

  self.table_name = 'employee_salary'

  belongs_to :employee, :foreign_key => 'id'

  validates_presence_of :salary, :message => 'Salary is required'
  validates_presence_of :bank_name, :message => 'Bank Name is required'
  validates_presence_of :bank_acc_no, :message => 'Bank Account No. is required'
  validates_presence_of :bank_acc_type, :message => 'Bank Account Type is required'
  validates_presence_of :bank_address, :message => 'Bank Address is required'
  validates_presence_of :epf_no, :message => 'EPF No. is required'
  validates_presence_of :pay_type, :message => 'Pay Type is required'

```

```

validates_presence_of :epf, :message => 'EPF Deduction is required'

validates_numericality_of :salary, :greater_than_or_equal_to => 0,
                           :message => 'Salary is invalid'
validates_numericality_of :allowance, :greater_than_or_equal_to => 0,
                           :message => 'Allowance is invalid'
validates_numericality_of :epf, :greater_than_or_equal_to => 0,
                           :message => 'EPF Deduction is invalid'
validates_numericality_of :socso, :greater_than_or_equal_to => 0,
                           :message => 'SOCZO Deduction is invalid'
validates_numericality_of :income_tax, :greater_than_or_equal_to => 0,
                           :message => 'Income Tax Deduction is invalid'

def allowance
  a = read_attribute(:allowance)
  a.blank? ? 0 : a
end

def epf
  a = read_attribute(:epf)
  a.blank? ? 0 : a
end

def income_tax
  a = read_attribute(:income_tax)
  a.blank? ? 0 : a
end

def salary
  a = read_attribute(:salary)
  a.blank? ? 0 : a
end

def socso
  a = read_attribute(:socso)
  a.blank? ? 0 : a
end

def display_pay_type
  case self.pay_type
  when 1
    'Monthly'

  when 2
    'Hourly'

  else
    ''
  end
end
end

```

E:\workspace\payroll_rails\app\models\employment_status.rb

```

# Model for employment_status table.
class EmploymentStatus < ActiveRecord::Base
  attr_accessible :id, :name

  self.table_name = 'employment_status'

  has_many :employee_job

  validates_presence_of :name, :message => 'Name is required'
  validates_uniqueness_of :name, :message => "Employment Status %{value} already exist"
end

```

E:\workspace\payroll_rails\app\models\job_category.rb

```

# Model for job category table.
class JobCategory < ActiveRecord::Base
  attr_accessible :id, :name

  self.table_name = 'job_category'

  has_many :employee_job

  validates_presence_of :name, :message => 'Name is required'

```

```

    validates_uniqueness_of :name, :message => "Category %{value} already exist"
end

E:\workspace\payroll_rails\app\models\overtime_rate.rb

# Model for overtime_rate table.
class OvertimeRate < ActiveRecord::Base
attr_accessible :duration, :pay_rate, :year

self.table_name = 'overtime_rate'

validates_presence_of :duration, :message => 'Duration is required'
validates_presence_of :year, :message => 'Year is required'
validates_presence_of :pay_rate, :message => 'Pay Rate is required'

validates_numericality_of :duration, :greater_than_or_equal_to => 0,
                           :message => 'Duration is invalid'
validates_numericality_of :year, :greater_than => 0, :message => 'Year is invalid'
validates_numericality_of :pay_rate, :greater_than => 0,
                           :message => 'Pay Rate is invalid'

validates_uniqueness_of :year,
                       :message => "Overtime rate for year %{value} already exist"

def duration
  a = read_attribute(:duration)
  a.blank? ? 0 : a
end

def pay_rate
  a = read_attribute(:pay_rate)
  a.blank? ? 0 : a
end
end

E:\workspace\payroll_rails\app\models\pay_rate.rb

# Model for pay_rate table.
class PayRate < ActiveRecord::Base
attr_accessible :hourly_pay_rate, :id, :month, :staff_id, :year

self.table_name = 'pay_rate'

validates_presence_of :staff_id, :message => 'Staff ID is required'
validates_presence_of :month, :message => 'Month is required'
validates_presence_of :year, :message => 'Year is required'
validates_presence_of :hourly_pay_rate, :message => 'Hourly pay rate is required'

validates_numericality_of :month, :greater_than => 0, :message => 'Month is invalid'
validates_numericality_of :month, :less_than_or_equal_to => 12,
                           :message => 'Month is invalid'
validates_numericality_of :year, :greater_than => 0, :message => 'Year is invalid'
validates_numericality_of :hourly_pay_rate, :greater_than => 0,
                           :message => 'Hourly pay rate is invalid'

def hourly_pay_rate
  a = read_attribute(:hourly_pay_rate)
  a.blank? ? 0 : a
end
end

E:\workspace\payroll_rails\app\models\salary_adjustment.rb

# Model for salary_adjustment table.
class SalaryAdjustment < ActiveRecord::Base
attr_accessible :id, :inc, :month, :staff_id, :year

self.table_name = 'salary_adjustment'

validates_presence_of :staff_id, :message => 'Staff ID is required'
validates_presence_of :inc, :message => 'Increment is required'
validates_presence_of :month, :message => 'Month is required'
validates_presence_of :year, :message => 'Year is required'

validates_numericality_of :inc, :greater_than => 0,
                           :message => 'Increment is invalid'
validates_numericality_of :month, :greater_than => 0, :message => 'Month is invalid'

```

```

validates_numericality_of :month, :less_than_or_equal_to => 12,
                           :message => 'Month is invalid'
validates_numericality_of :year, :greater_than => 0, :message => 'Year is invalid'

def inc
  a = read_attribute(:inc)
  a.blank? ? 0 : a
end
end

E:\workspace\payroll_rails\app\models\user.rb

require 'digest'

# Model for user table.
class User < ActiveRecord::Base
  attr_accessible :id, :pwd, :role, :status, :username, :pwd_confirmation
  attr_accessor :pwd

  self.table_name = 'user'

  has_one :employee

  validates :username, :uniqueness => { :message => "Username %{value} already exist" },
           :length => { :within => 3..50,
                         :message => "Minimum is %{count} characters" }
  validates :pwd, :confirmation => { :message => "Password doesn't match confirmation" },
           :length => { :within => 4..20,
                         :message => "Minimum is %{count} characters" },
           :presence => { :message => "Password is required" },
           :if => :password_required?

  before_save :encrypt_new_password

UNCHANGED_PASSWORD = '*****'
ADMIN = 1
NORMAL_USER = 2

@@roles = { 'Admin' => 1, 'Normal User' => 2 }
@@statuses = { 'Enabled' => 1, 'Disabled' => 0 }

def self.authenticate(username, password)
  user = find_by_username(username)
  if user && user.authenticated?(password) && user.enabled?
    return user
  end
end

def authenticated?(password)
  self.password == encrypt(password)
end

def enabled?
  self.status == true
end

def self.roles
  @@roles
end

def self.statuses
  @@statuses
end

def role_display
  role == 1 ? 'Admin' : 'Normal User'
end

def status_display
  status == true ? 'Enabled' : 'Disabled'
end

protected

def encrypt_new_password
  return if pwd.blank?
  self.password = encrypt(pwd)

```

```

end

def password_required?
  if pwd == UNCHANGED_PASSWORD
    return false
  end
  self.password.blank? || pwd.present?
end

def encrypt(string)
  Digest::SHA1.hexdigest(string)
end
end

E:\workspace\payroll_rails\app\controllers\application_controller.rb

# The main application's controller.
class ApplicationController < ActionController::Base
  protect_from_forgery

  LAYOUT = {
    :admin => 'admin',
    :chart => 'chart',
    :list => 'list',
    :user => 'user'
  }

  # Authenticates a user.
  def create
    user = User.authenticate(params[:username], params[:password])
    if user.present?
      session[:user_id] = user.id
      if user.role == User::ADMIN
        redirect_to admin_index_path and return
      else
        employee = user.employee
        if employee.present?
          session[:employee_id] = employee.id
          session[:staff_id] = employee.staff_id
          session[:supervisor_id] = employee.id
          redirect_to user_index_path and return
        else
          flash.now[:alert] = %Q{No employee record found. Please contact the administrator to create your employee record.}
        end
      end
    end
    else
      flash.now[:alert] = 'Incorrect username or password'
    end
    render :action => 'new'
  end

  # Logs out a user.
  def destroy
    reset_session
    redirect_to login_path
  end

  protected

  # Returns the current user's user_id.
  def current_user
    return unless session[:user_id]
    @current_user ||= session[:user_id]
  end

  # Checks whether a user is authenticated.
  def authenticate
    logged_in? ? true : access_denied
  end

  # Checks whether a user is logged in.
  def logged_in?

```

```

    current_user.present?
end

# Prevents a user from accessing the system without logging in.
def access_denied
  redirect_to login_path, :notice => 'Please log in to continue' and return false
end

# Formats a date into dd-mm-yyyy .
# Helper method.
def fmt_date(dt)
  if dt.present?
    dt.strftime('%d-%m-%Y')
  end
end

# Formats a time into hh:mm:ss AM/PM .
# Helper method.
def fmt_time(t)
  if t.present?
    t.in_time_zone('Kuala Lumpur').strftime('%l:%M %p')
  end
end

# Returns the month name from a given number.
# Helper method.
def month_name(i)
  ApplicationHelper.month_name(i)
end

# Returns a hash of month names.
def month_options
  months = I18n.t('date.month_names')
  o = {}

  (1..12).each do |m|
    o[months[m]] = m
  end
  o
end

# Returns the end year for year selection in view.
# Helper method.
def end_year
  2000
end

# Returns the user_id.
def get_user_id
  session[:user_id]
end

helper_method :current_user
helper_method :logged_in
helper_method :fmt_date
helper_method :fmt_time
helper_method :month_name
helper_method :end_year
end

```

E:\workspace\payroll_rails\app\controllers\admin\admin_controller.rb

```

# Admin base controller.
# All Admin controllers must inherit this controller.
# Admin controller serves incoming requests from an authenticated Admin User.
class Admin::AdminController < ApplicationController
  layout false

  # checks whether a user is authenticated before serving any request
  before_filter :authenticate

  # Display the admin page.
  # GET /admin/index
  def index
    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:admin] }
    end
  end

```

```

    end
end

E:\workspace\payroll_rails\app\controllers\admin\attendance_controller.rb

# This controller serves incoming requests to display out the Attendance records.
class Admin::AttendanceController < Admin::AdminController

  # List all records.
  # GET /att
  # GET /att.json
  def index
    @data = AttendanceHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /att/list
  # GET /att/list.json
  def list
    _work_date = params[:work_date]
    employee = params[:employee].blank? ? '' : params[:employee]

    work_date = Date.strptime(_work_date, '%d-%m-%Y') if _work_date.present?

    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgsiz = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
    sortcolumn = params[:sortcolumn].blank? ? AttendanceHelper::DEFAULT_SORT_COLUMN
      : params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? AttendanceHelper::DEFAULT_SORT_DIR :
      params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    filters = { :work_date => work_date,
                :employee => employee }

    if work_date.blank? && employee.blank?
      @data = AttendanceHelper.get_all(pgnum, pgsiz, sort)

    else
      @data = AttendanceHelper.get_filter_by(filters, pgnum, pgsiz, sort)
    end

    respond_to do |fmt|
      fmt.html { render :partial => 'list' }
      fmt.json { render :json => @data }
    end
  end
end

E:\workspace\payroll_rails\app\controllers\admin\department_controller.rb

# This controller serves incoming requests to display out the Department records.
class Admin::DepartmentController < Admin::AdminController

  # List all records.
  # GET /dept
  # GET /dept.json
  def index
    @data = DepartmentHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /dept/list
  # GET /dept/list.json
  def list
    keyword = params[:keyword].blank? ? '' : params[:keyword]

```

```

pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
pagesize = params[:pagesize].blank? ? 0 : params[:pagesize].to_i
sortcolumn = params[:sortcolumn].blank? ? DepartmentHelper::DEFAULT_SORT_COLUMN
                                         : params[:sortcolumn]
sortdir = params[:sortdir].blank? ? DepartmentHelper::DEFAULT_SORT_DIR :
params[:sortdir]

sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

if keyword.blank?
  @data = DepartmentHelper.get_all(pgnum, pagesize, sort)
else
  @data = DepartmentHelper.get_filter_by(keyword, pgnum, pagesize, sort)
end

respond_to do |fmt|
  fmt.html { render :partial => 'list' }
  fmt.json { render :json => @data }
end
end

# Display the Department form.
# GET /dept/new
# GET /dept/new.json
def new
  @dept = Department.new
  @form_id = 'add-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @dept }
  end
end

# Create new Department record.
# POST /dept/create
def create
  o = Department.new(:name => params[:name])

  if o.save
    render :json => { :success => 1,
                      :message => 'Department was successfully added.' }

  else
    render :json => DepartmentHelper.get_errors(o.errors)
  end
end

# Display the Department form, with existing record to edit.
# GET /dept/edit/1
# GET /dept/edit/1.json
def edit
  @dept = Department.find(params[:id])
  @form_id = 'edit-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @dept }
  end
end

# Update Department record.
# POST /dept/update/1
def update
  o = Department.find(params[:id])

  if o.update_attributes(:name => params[:name])
    render :json => { :success => 1,
                      :message => 'Department was successfully updated.' }

  else
    render :json => DepartmentHelper.get_errors(o.errors)
  end
end

# Delete a list of Department records.

```

```

# POST /dept/delete
def destroy
  keyword = params[:keyword].blank? ? '' : params[:keyword]
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  ids = params[:id]

  Department.delete_all(:id => ids)

  itemscount = DepartmentHelper.item_message(keyword, pgnum, pgszie)

  render :json => { :success => 1,
                     :itemscount => itemscount,
                     :message => "#{ids.size} Department(s) was successfully deleted." }
end
end

E:\workspace\payroll_rails\app\controllers\admin\designation_controller.rb

# This controller serves incoming requests to display out the Designation records.
class Admin::DesignationController < Admin::AdminController

  # List all records.
  # GET /designation
  # GET /designation.json
  def index
    @data = DesignationHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /designation/list
  # GET /designation/list.json
  def list
    find = params[:find].blank? ? 0 : params[:find].to_i
    keyword = params[:keyword].blank? ? '' : params[:keyword]
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
    sortcolumn = params[:sortcolumn].blank? ? DesignationHelper::DEFAULT_SORT_COLUMN
                                              : params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? DesignationHelper::DEFAULT_SORT_DIR :
                                              params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    if find == 0 && keyword.blank?
      @data = DesignationHelper.get_all(pgnum, pgszie, sort)
    else
      @data = DesignationHelper.get_filter_by(find, keyword, pgnum, pgszie, sort)
    end

    respond_to do |fmt|
      fmt.html { render :partial => 'list' }
      fmt.json { render :json => @data }
    end
  end

  # Display the Designation form.
  # GET /designation/new
  # GET /designation/new.json
  def new
    @designation = Designation.new
    @form_id = 'add-form'

    respond_to do |fmt|
      fmt.html { render :partial => 'form' }
      fmt.json { render :json => @designation }
    end
  end

  # Create new Designation record.
  # POST /designation/create

```

```

def create
o = Designation.new(:title => params[:title], :desc => params[:desc],
                     :note => params[:note])

if o.save
  render :json => { :success => 1,
                     :message => 'Job Title was successfully added.' }

else
  render :json => DesignationHelper.get_errors(o.errors)
end
end

# Display the Designation form, with existing record to edit.
# GET /designation/edit/1
# GET /designation/edit/1.json
def edit
  @designation = Designation.find(params[:id])
  @form_id = 'edit-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @designation }
  end
end

# Update Designation record.
# POST /designation/update/1
def update
  o = Designation.find(params[:id])

  if o.update_attributes(:title => params[:title], :desc => params[:desc],
                        :note => params[:note])
    render :json => { :success => 1,
                     :message => 'Job Title was successfully updated.' }

  else
    render :json => DesignationHelper.get_errors(o.errors)
  end
end

# Delete a list of Designation records.
# POST /designation/delete
def destroy
  keyword = params[:keyword].blank? ? '' : params[:keyword]
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  ids = params[:id]

  Designation.delete_all(:id => ids)

  itemscount = DesignationHelper.item_message(keyword, pgnum, pgszie)

  render :json => { :success => 1,
                    :itemscount => itemscount,
                    :message => "#{ids.size} Job Title(s) was successfully deleted." }
end
end

```

```

E:\workspace\payroll_rails\app\controllers\admin\employee_controller.rb

require 'securerandom'

# This controller serves incoming requests to display out the Employee records.
class Admin::EmployeeController < Admin::AdminController

  # List all records.
  # GET /employee
  # GET /employee.json
  def index
    @data = EmployeeHelper.get_all
    @employmentstatus = EmploymentStatus.order(:name).all
    @designation = Designation.order(:title).all
    @dept = Department.order(:name).all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /employee/list
  # GET /employee/list.json
  def list
    employee = params[:employee].blank? ? '' : params[:employee]
    staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
    employment_status = params[:employment_status].blank? ? 0 :
      params[:employment_status].to_i
    designation = params[:designation].blank? ? 0 : params[:designation].to_i
    dept = params[:dept].blank? ? 0 : params[:dept].to_i
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
    sortcolumn = params[:sortcolumn].blank? ? EmployeeHelper::DEFAULT_SORT_COLUMN :
      params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? EmployeeHelper::DEFAULT_SORT_DIR :
      params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    filters = { :employee => employee,
                :staff_id => staff_id,
                :employment_status => employment_status,
                :designation => designation,
                :dept => dept }

    if employee.blank? && staff_id.blank? && employment_status == 0 && designation == 0 &&
      dept == 0
      @data = EmployeeHelper.get_all(pgnum, pgszie, sort)
    else
      @data = EmployeeHelper.get_filter_by(filters, pgnum, pgszie, sort)
    end

    respond_to do |fmt|
      fmt.html { render :partial => 'list' }
      fmt.json { render :json => @data }
    end
  end

  # Display the Employee form.
  # GET /employee/new
  # GET /employee/new.json
  def new
    @employee = Employee.new
    @employee_contact = EmployeeContact.new
    @employee_job = EmployeeJob.new
    @employee_salary = EmployeeSalary.new
    @employee_qualification = EmployeeQualification.new
    @form_id = 'add-form'
    @users = User.order(:username).all
    @designations = Designation.order(:title).all
    @employment_statuses = EmploymentStatus.order(:name).all
    @job_categories = JobCategory.order(:name).all
    @departments = Department.order(:name).all
  end

```

```

respond_to do |fmt|
  fmt.html { render :partial => 'form' }
  fmt.json { render :json => @employee }
end
end

# Create new Employee record.
# POST /employee/create
def create
  o = EmployeeHelper.employee_obj(params)

  b1 = EmployeeContactHelper.is_empty_params?(params)
  oc = EmployeeContactHelper.employee_contact_obj(o, params)

  b2 = EmployeeJobHelper.is_empty_params?(params)
  oej = EmployeeJobHelper.employee_job_obj(o, params)

  b3 = EmployeeSalaryHelper.is_empty_params?(params)
  osa = EmployeeSalaryHelper.employee_salary_obj(o, params)

  b4 = EmployeeQualificationHelper.is_empty_params?(params)
  oq = EmployeeQualificationHelper.employee_qualification_obj(o, params)

  v1 = o.valid?
  v2 = b1 ? true : oc.valid?
  v3 = b2 ? true : oej.valid?
  v4 = b3 ? true : osa.valid?
  v5 = b4 ? true : oq.valid?

  if !v1 || !v2 || !v3 || !v4 || !v5
    employee_errors = EmployeeHelper.get_errors(o.errors)
    employee_contact_errors = EmployeeContactHelper.get_errors(oc.errors)
    employee_job_errors = EmployeeJobHelper.get_errors(oej.errors)
    employee_salary_errors = EmployeeSalaryHelper.get_errors(osa.errors)
    employee_qualification_errors = EmployeeQualificationHelper.get_errors(oq.errors)

    errors = { :error => 1, :employee => employee_errors,
               :employee_contact => employee_contact_errors,
               :employee_job => employee_job_errors,
               :employee_salary => employee_salary_errors,
               :employee_qualification => employee_qualification_errors }
    render :json => errors and return
  end

  ActiveRecord::Base.transaction do
    o.save
    oc.save if b1
    oej.save if b2
    osa.save if b3
    oq.save if b4
  end

  render :json => { :success => 1,
                    :message => 'Employee was successfully added.' }
end

# Display the Employee form, with existing record to edit.
# GET /employee/edit/1
# GET /employee/edit/1.json
def edit
  @employee = Employee.find(params[:id])
  @employee_contact = @employee.employee_contact.blank? ? EmployeeContact.new
    : @employee.employee_contact
  @employee_job = @employee.employee_job.blank? ? EmployeeJob.new
    : @employee.employee_job
  @employee_salary = @employee.employee_salary.blank? ? EmployeeSalary.new
    : @employee.employee_salary
  @employee_qualification = @employee.employee_qualification.blank? ?
    EmployeeQualification.new : @employee.employee_qualification
  @form_id = 'edit-form'
  @users = User.order(:username).all
  @designations = Designation.order(:title).all
  @employment_statuses = EmploymentStatus.order(:name).all
  @job_categories = JobCategory.order(:name).all
  @departments = Department.order(:name).all

  respond_to do |fmt|

```

```

    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @employee }
  end
end

# Update Employee record.
# POST /employee/update
def update
  o = Employee.find(params[:id])
  oc = o.employee_contact
  oej = o.employee_job
  osa = o.employee_salary
  oq = o.employee_qualification

  oc_new = false
  if oc.blank?
    oc = EmployeeContactHelper.employee_contact_obj(o, params)
    oc_new = true
  end

  oej_new = false
  if oej.blank?
    oej = EmployeeJobHelper.employee_job_obj(o, params)
    oej_new = true
  end

  osa_new = false
  if osa.blank?
    osa = EmployeeSalaryHelper.employee_salary_obj(o, params)
    osa_new = true
  end

  oq_new = false
  if oq.blank?
    oq = EmployeeQualificationHelper.employee_qualification_obj(o, params)
    oq_new = true
  end

  b1 = EmployeeContactHelper.is_empty_params?(params)
  b2 = EmployeeJobHelper.is_empty_params?(params)
  b3 = EmployeeSalaryHelper.is_empty_params?(params)
  b4 = EmployeeQualificationHelper.is_empty_params?(params)

  v1 = o.valid?
  v2 = b1 ? true : oc.valid?
  v3 = b2 ? true : oej.valid?
  v4 = b3 ? true : osa.valid?
  v5 = b4 ? true : oq.valid?

  if !v1 || !v2 || !v3 || !v4 || !v5
    employee_errors = EmployeeHelper.get_errors(o.errors)
    employee_contact_errors = EmployeeContactHelper.get_errors(oc.errors)
    employee_job_errors = EmployeeJobHelper.get_errors(oej.errors)
    employee_salary_errors = EmployeeSalaryHelper.get_errors(osa.errors)
    employee_qualification_errors = EmployeeQualificationHelper.get_errors(oq.errors)

    errors = { :error => 1, :employee => employee_errors,
               :employee_contact => employee_contact_errors,
               :employee_job => employee_job_errors,
               :employee_salary => employee_salary_errors,
               :employee_qualification => employee_qualification_errors }
    render :json => errors and return
  end

  ActiveRecord::Base.transaction do
    EmployeeHelper.update_obj(o, params)

    if oc_new
      oc.save
    else
      EmployeeContactHelper.update_obj(oc, params)
    end

    if oej_new
      oej.save
    end
  end
end

```

```

    else
      EmployeeJobHelper.update_obj(oej, params)
    end

    if osa_new
      osa.save

    else
      EmployeeSalaryHelper.update_obj(osa, params)
    end

    if oq_new
      oq.save

    else
      EmployeeQualificationHelper.update_obj(oq, params)
    end
  end

  render :json => { :success => 1,
                     :message => 'Employee was successfully updated.' }
end

# Delete a list of Employee records.
# POST /employee/delete
def destroy
  employee = params[:employee].blank? ? '' : params[:employee]
  staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
  employment_status = params[:employment_status].blank? ? 0 :
  params[:employment_status].to_i
  designation = params[:designation].blank? ? 0 : params[:designation].to_i
  dept = params[:dept].blank? ? 0 : params[:dept].to_i
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  ids = params[:id]

  Employee.delete_all(:id => ids)

  filters = { :employee => employee,
              :staff_id => staff_id,
              :employment_status => employment_status,
              :designation => designation,
              :dept => dept }

  if employee.blank? && staff_id.blank? && employment_status == 0 && designation == 0 &&
  dept == 0
    itemscount = EmployeeHelper.item_message(nil, pgnum, pgszie)

  else
    itemscount = EmployeeHelper.item_message(filters, pgnum, pgszie)
  end

  render :json => { :success => 1,
                     :itemscount => itemscount,
                     :message => "#{ids.size} employee(s) was successfully deleted." }
end
end

```

```

E:\workspace\payroll_rails\app\controllers\admin\employment_status_controller.rb

# This controller serves incoming requests to display out the EmploymentStatus records.
class Admin::EmploymentStatusController < Admin::AdminController

  # List all records.
  # GET /empstatus
  # GET /empstatus.json
  def index
    @data = EmploymentStatusHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /empstatus/list
  # GET /empstatus/list.json
  def list
    keyword = params[:keyword].blank? ? '' : params[:keyword]
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgszie = params[:pgszie].blank? ? 0 : params[:pgszie].to_i
    sortcolumn = params[:sortcolumn].blank? ? EmploymentStatusHelper::DEFAULT_SORT_COLUMN
                                              : params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? EmploymentStatusHelper::DEFAULT_SORT_DIR
                                      : params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    if keyword.blank?
      @data = EmploymentStatusHelper.get_all(pgnum, pgszie, sort)
    else
      @data = EmploymentStatusHelper.get_filter_by(keyword, pgnum, pgszie, sort)
    end

    respond_to do |fmt|
      fmt.html { render :partial => 'list' }
      fmt.json { render :json => @data }
    end
  end

  # Display the EmploymentStatus form.
  # GET /empstatus/new
  # GET /empstatus/new.json
  def new
    @empstatus = EmploymentStatus.new
    @form_id = 'add-form'

    respond_to do |fmt|
      fmt.html { render :partial => 'form' }
      fmt.json { render :json => @empstatus }
    end
  end

  # Create new EmploymentStatus record.
  # POST /empstatus/create
  def create
    o = EmploymentStatus.new(:name => params[:name])

    if o.save
      render :json => { :success => 1,
                        :message => 'Employment Status was successfully added.' }

    else
      render :json => EmploymentStatusHelper.get_errors(o.errors)
    end
  end

  # Display the EmploymentStatus form, with existing record to edit.
  # GET /empstatus/edit/1
  # GET /empstatus/edit/1.json
  def edit
    @empstatus = EmploymentStatus.find(params[:id])
    @form_id = 'edit-form'
  end

```

```

respond_to do |fmt|
  fmt.html { render :partial => 'form' }
  fmt.json { render :json => @empstatus }
end
end

# Update EmploymentStatus record.
# POST /empstatus/update/1
def update
  o = EmploymentStatus.find(params[:id])

  if o.update_attributes(:name => params[:name])
    render :json => { :success => 1,
                      :message => 'Employment Status was successfully updated.' }

  else
    render :json => EmploymentStatusHelper.get_errors(o.errors)
  end
end

# Delete a list of EmploymentStatus records.
# POST /empstatus/delete
def destroy
  keyword = params[:keyword].blank? ? '' : params[:keyword]
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgsiz = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  ids = params[:id]

  EmploymentStatus.delete_all(:id => ids)

  itemscount = EmploymentStatusHelper.item_message(keyword, pgnum, pgsiz)

  render :json => { :success => 1,
                    :itemscount => itemscount,
                    :message => "%{#{ids.size}} Employment Status(es) was successfully deleted." }
end
end

```

E:\workspace\payroll_rails\app\controllers\admin\hourly_payroll_chart_controller.rb

```

# This controller serves incoming requests to display Hourly Payroll Chart.
class Admin::HourlyPayrollChartController < Admin::AdminController

```

```

# Display the main page.
# GET /hourly/chart
def index
  @month_hash = month_options

  respond_to do |fmt|
    fmt.html { render :layout => LAYOUT[:chart] }
  end
end

# Get the hourly payroll json data to be populated to the chart.
# GET /hourly/chart/data
def data
  staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
  month = params[:month].blank? ? '0' : params[:month]
  year = params[:year].blank? ? 0 : params[:year].to_i

  title = 'Hourly Payroll'
  yaxis = 'Total Amount (RM)'

  months = I18n.t('date.month_names')

  o = Array.new(12) { |x| [months[x + 1], 0] }
  b = Array.new(12) { |x| 0 }
  categories = Array.new(12) { |x| months[x + 1][0..2] }

  if staff_id.present?
    liststaff = [staff_id]

  else
    list = PayRate.select('distinct(staff_id)').all
    liststaff = list.collect { |x| x.staff_id }
  end

```

```

    end

    if year != 0
      listyear = [year]
      title = "Hourly Payroll for #{year}"

    else
      list = PayRate.select('distinct(year)').all
      listyear = list.collect { |x| x.year }
    end

    if month != '0'
      listmonth = month.collect { |x| x.to_i }

    else
      list = PayRate.select('distinct(month)').all
      listmonth = list.collect { |x| x.month }
    end

    listyear.each do |y|
      listmonth.each do |m|
        liststaff.each do |s|
          filters = { :year => y, :month => m, :staff_id => s }
          total_hours = AttendanceHelper.get_total_hours(filters)
          rate = PayRateHelper.get_pay_rate(filters)
          v = total_hours * rate
          o[m - 1][1] = v
          b[m - 1] += v
        end
      end
    end

    c = b.collect { |x| x.round(2) }

    @data = { :pie => o, :column => { :data => c,
                                         :categories => categories,
                                         :yaxis => yaxis },
              :title => title }

    render :json => @data
  end
end

```

E:\workspace\payroll_rails\app\controllers\admin\job_category_controller.rb

```

# This controller serves incoming requests to display out the JobCategory records.
class Admin::JobCategoryController < Admin::AdminController

  # List all records.
  # GET /jobcat
  # GET /jobcat.json
  def index
    @data = JobCategoryHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /jobcat/list
  # GET /jobcat/list.json
  def list
    keyword = params[:keyword].blank? ? '' : params[:keyword]
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgsze = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
    sortcolumn = params[:sortcolumn].blank? ? JobCategoryHelper::DEFAULT_SORT_COLUMN : params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? JobCategoryHelper::DEFAULT_SORT_DIR : params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    if keyword.blank?
      @data = JobCategoryHelper.get_all(pgnum, pgsze, sort)
    else
      @data = JobCategoryHelper.get_by_keyword(keyword, pgnum, pgsze, sort)
    end
  end
end

```

```

    else
      @data = JobCategoryHelper.get_filter_by(keyword, pgnum, pgszie, sort)
    end

    respond_to do |fmt|
      fmt.html { render :partial => 'list' }
      fmt.json { render :json => @data }
    end
  end

# Display the JobCategory form.
# GET /jobcat/new
# GET /jobcat/new.json
def new
  @jobcat = JobCategory.new
  @form_id = 'add-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @jobcat }
  end
end

# Create new JobCategory record.
# POST /jobcat/create
def create
  o = JobCategory.new(:name => params[:name])

  if o.save
    render :json => { :success => 1,
                      :message => 'Job Category was successfully added.' }

  else
    render :json => JobCategoryHelper.get_errors(o.errors)
  end
end

# Display the JobCategory form, with existing record to edit.
# GET /jobcat/edit/1
# GET /jobcat/edit/1.json
def edit
  @jobcat = JobCategory.find(params[:id])
  @form_id = 'edit-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @jobcat }
  end
end

# Update JobCategory record.
# POST /jobcat/update/1
def update
  o = JobCategory.find(params[:id])

  if o.update_attributes(:name => params[:name])
    render :json => { :success => 1,
                      :message => 'Job Category was successfully updated.' }

  else
    render :json => JobCategoryHelper.get_errors(o.errors)
  end
end

# Delete a list of JobCategory records.
# POST /jobcat/delete
def destroy
  keyword = params[:keyword].blank? ? '' : params[:keyword]
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  ids = params[:id]

  JobCategory.delete_all(:id => ids)

  itemscount = JobCategoryHelper.item_message(keyword, pgnum, pgszie)

  render :json => { :success => 1,

```

```

        :itemscount => itemscount,
        :message => %Q{#{ids.size} Job Categori(es) was successfully
                           deleted.} }
    end
end

E:\workspace\payroll_rails\app\controllers\admin\overtime_chart_controller.rb

# This controller serves incoming requests to display Employee Overtime Chart.
class Admin::OvertimeChartController < Admin::AdminController

  # Display the main page.
  # GET /overtime/chart
  def index
    @month_hash = month_options

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:chart] }
    end
  end

  # Get the overtime json data to be populated to the chart.
  # GET /overtime/chart/data
  def data
    staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
    month = params[:month].blank? ? '0' : params[:month]
    year = params[:year].blank? ? 0 : params[:year].to_i

    criteria = Attendance
    title = 'Overtime'
    yaxis = 'Duration (hours)'

    if staff_id.present?
      criteria = criteria.where(:staff_id => staff_id)
    end

    if year != 0
      criteria = criteria.where('year(work_date) = ?', year)
      title = "Overtime for Year #{year}"
    end

    if month != '0'
      criteria = criteria.where('month(work_date) in (?)', month)
    end

    list = criteria.order(:work_date).all
    months = I18n.t('date.month_names')

    b = Array.new(12) { |x| 0 }
    categories = Array.new(12) { |x| months[x + 1][0..2] }
    list.each do |o|
      to = ApplicationHelper.localtime(o.time_out)
      v = Time.new(to.year, to.month, to.day, 18, 0, 0, '+08:00')
      x = (to - v) / 3600.0
      m = o.work_date.month
      if x > 0
        b[m - 1] += x
      end
    end

    c = b.collect { |x| x.round(2) }

    @data = { :data => c, :categories => categories, :title => title, :yaxis => yaxis }

    render :json => @data
  end
end

```

```

E:\workspace\payroll_rails\app\controllers\admin\overtime_rate_controller.rb

# This controller serves incoming requests to display out the OvertimeRate records.
class Admin::OvertimeRateController < Admin::AdminController

  # GET /overtimerate
  # GET /overtimerate.json
  def index
    @data = OvertimeRateHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /overtimerate/list
  # GET /overtimerate/list.json
  def list
    year = params[:year].blank? ? 0 : params[:year].to_i
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgsiz = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
    sortcolumn = params[:sortcolumn].blank? ? OvertimeRateHelper::DEFAULT_SORT_COLUMN : params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? OvertimeRateHelper::DEFAULT_SORT_DIR : params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    filters = { :year => year }

    if year == 0
      @data = OvertimeRateHelper.get_all(pgnum, pgsiz, sort)
    else
      @data = OvertimeRateHelper.get_filter_by(filters, pgnum, pgsiz, sort)
    end

    respond_to do |fmt|
      fmt.html { render :partial => 'list' }
      fmt.json { render :json => @data }
    end
  end

  # Display the OvertimeRate form.
  # GET /overtimerate/new
  # GET /overtimerate/new.json
  def new
    @rate = OvertimeRate.new
    @form_id = 'add-form'

    respond_to do |fmt|
      fmt.html { render :partial => 'form' }
      fmt.json { render :json => @rate }
    end
  end

  # Create new OvertimeRate record.
  # POST /overtimerate/create
  def create
    o = OvertimeRate.new(:duration => params[:duration], :year => params[:year],
                         :pay_rate => params[:pay_rate])

    if o.save
      render :json => { :success => 1,
                        :message => 'Overtime Rate successfully added.' }

    else
      render :json => OvertimeRateHelper.get_errors(o.errors)
    end
  end

  # Display the OvertimeRate form, with existing record to edit.
  # GET /overtimerate/edit/1
  # GET /overtimerate/edit/1.json
  def edit

```

```

@rate = OvertimeRate.find(params[:id])
@form_id = 'edit-form'

respond_to do |fmt|
  fmt.html { render :partial => 'form' }
  fmt.json { render :json => @rate }
end
end

# Update OvertimeRate record.
# POST /overtimerate/update
def update
  o = OvertimeRate.find(params[:id])

  if o.update_attributes(:duration => params[:duration], :year => params[:year],
                        :pay_rate => params[:pay_rate])
    render :json => { :success => 1,
                      :message => 'Pay Rate was successfully updated.' }

  else
    render :json => OvertimeRateHelper.get_errors(o.errors)
  end
end

# Delete a list of OvertimeRate records.
# POST /overtimerate/delete
def destroy
  year = params[:year].blank? ? 0 : params[:year].to_i
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  ids = params[:id]

  OvertimeRate.delete_all(:id => ids)

  filters = { :year => year }

  if year == 0
    itemscount = OvertimeRateHelper.item_message(nil, pgnum, pgszie)

  else
    itemscount = OvertimeRateHelper.item_message(filters, pgnum, pgszie)
  end

  render :json => { :success => 1,
                    :itemscount => itemscount,
                    :message => "%{ids.size} Overtime rate(s) was successfully deleted." }
end
end

E:\workspace\payroll_rails\app\controllers\admin\pay_rate_controller.rb

require 'securerandom'

# This controller serves incoming requests to display out the PayRate records.
class Admin::PayRateController < Admin::AdminController

  # GET /payrate
  # GET /payrate.json
  def index
    @data = PayRateHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /payrate/list
  # GET /payrate/list.json
  def list
    staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
    month = params[:month].blank? ? 0 : params[:month].to_i
    year = params[:year].blank? ? 0 : params[:year].to_i
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  end

```

```

sortcolumn = params[:sortcolumn].blank? ? PayRateHelper::DEFAULT_SORT_COLUMN : params[:sortcolumn]
sortdir = params[:sortdir].blank? ? PayRateHelper::DEFAULT_SORT_DIR : params[:sortdir]

sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

filters = { :staff_id => staff_id,
            :month => month,
            :year => year }

if staff_id.blank? && month == 0 && year == 0
  @data = PayRateHelper.get_all(pgnum, pgszie, sort)
else
  @data = PayRateHelper.get_filter_by(filters, pgnum, pgszie, sort)
end

respond_to do |fmt|
  fmt.html { render :partial => 'list' }
  fmt.json { render :json => @data }
end
end

# Display the PayRate form.
# GET /payrate/new
# GET /payrate/new.json
def new
  @payrate = PayRate.new
  @form_id = 'add-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @payrate }
  end
end

# Create new PayRate record.
# POST /payrate/create
def create
  o = PayRate.new(:id => SecureRandom.uuid, :staff_id => params[:staff_id],
                  :month => params[:month], :year => params[:year],
                  :hourly_pay_rate => params[:pay_rate])

  if o.save
    render :json => { :success => 1,
                      :message => 'Pay Rate successfully added.' }

  else
    render :json => PayRateHelper.get_errors(o.errors)
  end
end

# Display the PayRate form, with existing record to edit.
# GET /payrate/edit/1
# GET /payrate/edit/1.json
def edit
  @payrate = PayRate.find(params[:id])
  @form_id = 'edit-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @payrate }
  end
end

# Update PayRate record.
# POST /payrate/update
def update
  o = PayRate.find(params[:id])

  if o.update_attributes(:staff_id => params[:staff_id], :month => params[:month],
                        :year => params[:year], :hourly_pay_rate => params[:pay_rate])
    render :json => { :success => 1,
                      :message => 'Pay Rate was successfully updated.' }

  else
    render :json => PayRateHelper.get_errors(o.errors)
  end
end

```

```

    end
end

# Delete a list of PayRate records.
# POST /payrate/delete
def destroy
  staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
  month = params[:month].blank? ? 0 : params[:month].to_i
  year = params[:year].blank? ? 0 : params[:year].to_i
  status = params[:status].blank? ? 0 : params[:status].to_i
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  ids = params[:id]

  PayRate.delete_all(:id => ids)

  filters = { :staff_id => staff_id,
              :month => month,
              :year => year }

  if staff_id.blank? && month == 0 && year == 0
    itemscount = PayRateHelper.item_message(nil, pgnum, pgszie)
  else
    itemscount = PayRateHelper.item_message(filters, pgnum, pgszie)
  end

  render :json => { :success => 1,
                    :itemscount => itemscount,
                    :message => "#{ids.size} pay rate(s) was successfully deleted." }
end
end

```

E:\workspace\payroll_rails\app\controllers\admin\payslip_controller.rb

```

# This controller serves incoming requests to display out the Employee records for pay
# slip generation.
class Admin::PayslipController < Admin::AdminController

  # List all records.
  # GET /payslip
  # GET /payslip.json
  def index
    @data = EmployeeHelper.get_all
    @employmentstatus = EmploymentStatus.order(:name).all
    @designation = Designation.order(:title).all
    @dept = Department.order(:name).all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /payslip/list
  # GET /payslip/list.json
  def list
    employee = params[:employee].blank? ? '' : params[:employee]
    staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
    employment_status = params[:employment_status].blank? ? 0 :
      params[:employment_status].to_i
    designation = params[:designation].blank? ? 0 : params[:designation].to_i
    dept = params[:dept].blank? ? 0 : params[:dept].to_i
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
    sortcolumn = params[:sortcolumn].blank? ? EmployeeHelper::DEFAULT_SORT_COLUMN :
      params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? EmployeeHelper::DEFAULT_SORT_DIR : params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    filters = { :employee => employee,
                :staff_id => staff_id,
                :employment_status => employment_status,
                :designation => designation,
                :dept => dept }
  end
end

```

```

if employee.blank? && staff_id.blank? && employment_status == 0 && designation == 0 &&
dept == 0
@data = EmployeeHelper.get_all(pgnum, pgszie, sort)

else
@data = EmployeeHelper.get_filter_by(filters, pgnum, pgszie, sort)
end

respond_to do |fmt|
  fmt.html { render :partial => 'list' }
  fmt.json { render :json => @data }
end
end

# Display the pay slip information.
# GET /payslip/slip/1/1/2012
# GET /payslip/slip/1/1/2012.json
def payslip
@employee = Employee.find(params[:id])
@employee_salary = @employee.employee_salary

_month = params[:month].to_i
month = month_name(params[:month].to_i)
year = params[:year].blank? ? Time.now.year : params[:year].to_i
@period = "#{month}-#{year}"

if @employee_salary.blank?
@total_earnings = 0
@total_deduct = 0
@nett_salary = 0

@employee_salary = EmployeeSalary.new

respond_to do |fmt|
  fmt.html { render 'payslip_monthly' }
  fmt.json { render :json => [@employee, @total_earnings, @total_deduct,
@nett_salary] }
end

else
# checks whether the salary type is monthly
if @employee_salary.pay_type == 1
filters = { :year => year, :month => _month, :staff_id => @employee.staff_id }
@total_overtime = PayslipHelper.total_overtime(filters)
@total_overtime_earnings = PayslipHelper.total_overtime_earnings(filters,
@total_overtime)
@adjustment = SalaryAdjustmentHelper.get_salary_adjustment(filters)

@total_earnings = PayslipHelper.total_earnings(@employee_salary, @adjustment,
@total_overtime_earnings)
@total_deduct = PayslipHelper.total_deductions(@employee_salary)
@nett_salary = PayslipHelper.nett_salary(@total_earnings, @total_deduct)

@basic_pay = @employee_salary.salary + @adjustment

respond_to do |fmt|
  fmt.html { render 'payslip_monthly' }
  fmt.json { render :json => [@employee, @total_earnings, @total_deduct,
@nett_salary, @total_overtime,
@total_overtime_earnings, @adjustment, @basic_pay] }
end

# hourly type
else
filters = { :year => year, :month => _month, :staff_id => @employee.staff_id }
@total_earnings, @total_hours, @hourly_pay_rate =
PayslipHelper.total_earnings_hourly(@employee_salary, filters)
@total_deduct = PayslipHelper.total_deductions(@employee_salary)
@nett_salary = PayslipHelper.nett_salary_hourly(@total_earnings, @total_deduct)

respond_to do |fmt|
  fmt.html { render 'payslip_hourly' }
  fmt.json { render :json => [@employee, @total_earnings, @total_deduct,
@nett_salary, @total_hours, @hourly_pay_rate] }
end
end
end

```

```

    end
  end
end

E:\workspace\payroll_rails\app\controllers\admin\salary_adjustment_controller.rb

require 'securerandom'

# This controller serves incoming requests to display out the SalaryAdjustment records.
class Admin::SalaryAdjustmentController < Admin::AdminController

  # GET /salaryadj
  # GET /salaryadj.json
  def index
    @data = SalaryAdjustmentHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /salaryadj/list
  # GET /salaryadj/list.json
  def list
    staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
    month = params[:month].blank? ? 0 : params[:month].to_i
    year = params[:year].blank? ? 0 : params[:year].to_i
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
    sortcolumn = params[:sortcolumn].blank? ? SalaryAdjustmentHelper::DEFAULT_SORT_COLUMN :
      params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? SalaryAdjustmentHelper::DEFAULT_SORT_DIR :
      params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    filters = { :staff_id => staff_id,
                :month => month,
                :year => year }

    if staff_id.blank? && month == 0 && year == 0
      @data = SalaryAdjustmentHelper.get_all(pgnum, pgszie, sort)
    else
      @data = SalaryAdjustmentHelper.get_filter_by(filters, pgnum, pgszie, sort)
    end

    respond_to do |fmt|
      fmt.html { render :partial => 'list' }
      fmt.json { render :json => @data }
    end
  end

  # Display the SalaryAdjustment form.
  # GET /salaryadj/new
  # GET /salaryadj/new.json
  def new
    @adj = SalaryAdjustment.new
    @form_id = 'add-form'

    respond_to do |fmt|
      fmt.html { render :partial => 'form' }
      fmt.json { render :json => @adj }
    end
  end

  # Create new SalaryAdjustment record.
  # POST /salaryadj/create
  def create
    staff_id = params[:staff_id]

    o = SalaryAdjustment.new(:id => SecureRandom.uuid, :staff_id => params[:staff_id],
                           :inc => params[:inc], :month => params[:month],
                           :year => params[:year])
  end

```

```

    if o.save
      render :json => { :success => 1,
                         :message => 'Salary Adjustment successfully added.' }

    else
      render :json => SalaryAdjustmentHelper.get_errors(o.errors)
    end
  end

# Display the SalaryAdjustment form, with existing record to edit.
# GET /salaryadj/edit/1
# GET /salaryadj/edit/1.json
def edit
  @adj = SalaryAdjustment.find(params[:id])
  @form_id = 'edit-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @adj }
  end
end

# Update SalaryAdjustment record.
# POST /salaryadj/update
def update
  o = SalaryAdjustment.find(params[:id])

  if o.update_attributes(:staff_id => params[:staff_id], :inc => params[:inc],
                        :month => params[:month], :year => params[:year])
    render :json => { :success => 1,
                      :message => 'Pay Rate was successfully updated.' }

  else
    render :json => PayRateHelper.get_errors(o.errors)
  end
end

# Delete a list of SalaryAdjustment records.
# POST /salaryadj/delete
def destroy
  staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
  month = params[:month].blank? ? 0 : params[:month].to_i
  year = params[:year].blank? ? 0 : params[:year].to_i
  status = params[:status].blank? ? 0 : params[:status].to_i
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgsz = params[:pgsz].blank? ? 0 : params[:pgsz].to_i
  ids = params[:id]

  SalaryAdjustment.delete_all(:id => ids)

  filters = { :staff_id => staff_id,
              :month => month,
              :year => year }

  if staff_id.blank? && month == 0 && year == 0
    itemscount = SalaryAdjustmentHelper.item_message(nil, pgnum, pgsz)

  else
    itemscount = SalaryAdjustmentHelper.item_message(filters, pgnum, pgsz)
  end

  render :json => { :success => 1,
                    :itemscount => itemscount,
                    :message => "%Q{#{ids.size}} salary adjustment(s) was successfully deleted." }
end
end

```

```

E:\workspace\payroll_rails\app\controllers\admin\total_work_hours_chart_controller.rb

# This controller serves incoming requests to display Employee Total Hours Worked Chart.
class Admin::TotalWorkHoursChartController < Admin::AdminController

  # Display the main page.
  # GET /workhours/chart
  def index
    @month_hash = month_options

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:chart] }
    end
  end

  # Get the total hours worked json data to be populated to the chart.
  # GET /workhours/chart/data
  def data
    staff_id = params[:staff_id].blank? ? '' : params[:staff_id]
    month = params[:month].blank? ? '0' : params[:month]
    year = params[:year].blank? ? 0 : params[:year].to_i

    title = 'Total Hours Worked'
    yaxis = 'Duration (hours)'

    months = I18n.t('date.month_names')

    b = Array.new(12) { |x| 0 }
    categories = Array.new(12) { |x| months[x + 1][0..2] }

    if staff_id.present?
      liststaff = [staff_id]

    else
      list = PayRate.select('distinct(staff_id)').all
      liststaff = list.collect { |x| x.staff_id }
    end

    if year != 0
      listyear = [year]
      title = "Total Hours Worked for #{year}"

    else
      list = PayRate.select('distinct(year)').all
      listyear = list.collect { |x| x.year }
    end

    if month != '0'
      listmonth = month.collect { |x| x.to_i }

    else
      list = PayRate.select('distinct(month)').all
      listmonth = list.collect { |x| x.month }
    end

    listyear.each do |y|
      listmonth.each do |m|
        liststaff.each do |s|
          filters = { :year => y, :month => m, :staff_id => s }
          total_hours = AttendanceHelper.get_total_hours(filters)
          v = total_hours
          b[m - 1] += v
        end
      end
    end

    c = b.collect { |x| x.round(2) }

    @data = { :data => c, :categories => categories, :title => title, :yaxis => yaxis }

    render :json => @data
  end
end

```

```

E:\workspace\payroll_rails\app\controllers\admin\user_controller.rb

require 'securerandom'

# This controller serves incoming requests to display out the User records.
class Admin::UserController < Admin::AdminController

  # List all records.
  # GET /user
  # GET /user.json
  def index
    @data = UserHelper.get_all

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:list] }
      fmt.json { render :json => @data }
    end
  end

  # List records by filtering.
  # GET /user/list
  # GET /user/list.json
  def list
    username = params[:username].blank? ? '' : params[:username]
    role = params[:role].blank? ? 0 : params[:role].to_i
    employee = params[:employee].blank? ? '' : params[:employee]
    status = params[:status].blank? ? 0 : params[:status].to_i
    pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
    pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
    sortcolumn = params[:sortcolumn].blank? ? UserHelper::DEFAULT_SORT_COLUMN :
      params[:sortcolumn]
    sortdir = params[:sortdir].blank? ? UserHelper::DEFAULT_SORT_DIR : params[:sortdir]

    sort = ApplicationHelper::Sort.new(sortcolumn, sortdir)

    filters = { :username => username,
                :role => role,
                :employee => employee,
                :status => status }

    if username.blank? && role == 0 && employee.blank? && status == 0
      @data = UserHelper.get_all(pgnum, pgszie, sort)
    else
      @data = UserHelper.get_filter_by(filters, pgnum, pgszie, sort)
    end

    respond_to do |fmt|
      fmt.html { render :partial => 'list' }
      fmt.json { render :json => @data }
    end
  end

  # Display the User form.
  # GET /user/new
  # GET /user/new.json
  def new
    @user = User.new
    @form_id = 'add-form'

    respond_to do |fmt|
      fmt.html { render :partial => 'form' }
      fmt.json { render :json => @user }
    end
  end

  # Create new User record.
  # POST /user/create
  def create
    status = params[:status]
    status = status == '1' ? true : false

    o = User.new(:id => SecureRandom.uuid, :role => params[:role],
                :username => params[:username], :status => status,
                :pwd => params[:pwd], :pwd_confirmation => params[:pwdconfirm])

    if o.save

```

```

    render :json => { :success => 1,
                      :message => 'User was successfully added.' }

  else
    render :json => UserHelper.get_errors(o.errors)
  end
end

# Display the User form, with existing record to edit.
# GET /user/edit/1
# GET /user/edit/1.json
def edit
  @user = User.find(params[:id])
  @form_id = 'edit-form'

  respond_to do |fmt|
    fmt.html { render :partial => 'form' }
    fmt.json { render :json => @user }
  end
end

# Update User record.
# POST /user/update
def update
  o = User.find(params[:id])
  status = params[:status]
  status = status == '1' ? true : false

  if o.update_attributes(:role => params[:role], :username => params[:username],
                        :status => status, :pwd => params[:pwd],
                        :pwd_confirmation => params[:pwdconfirm])
    render :json => { :success => 1,
                      :message => 'User was successfully updated.' }

  else
    render :json => UserHelper.get_errors(o.errors)
  end
end

# Delete a list of User records.
# POST /user/delete
def destroy
  username = params[:username].blank? ? '' : params[:username]
  role = params[:role].blank? ? 0 : params[:role].to_i
  employee = params[:employee].blank? ? '' : params[:employee]
  status = params[:status].blank? ? 0 : params[:status].to_i
  pgnum = params[:pgnum].blank? ? 1 : params[:pgnum].to_i
  pgszie = params[:pgsize].blank? ? 0 : params[:pgsize].to_i
  ids = params[:id]

  User.delete_all(:id => ids)

  filters = { :username => username,
              :role => role,
              :employee => employee,
              :status => status }

  if username.blank? && role == 0 && employee.blank? && status == 0
    itemscount = UserHelper.item_message(nil, pgnum, pgszie)

  else
    itemscount = UserHelper.item_message(filters, pgnum, pgszie)
  end

  render :json => { :success => 1,
                    :itemscount => itemscount,
                    :message => "#{ids.size} user(s) was successfully deleted." }
end
end

```

```

E:\workspace\payroll_rails\app\controllers\user\contact_controller.rb

# This controller serves incoming requests to display out the EmployeeContact record.
class User::ContactController < User::UserController

  # Display the contact.
  # GET /contact
  # GET /contact.json
  def index
    id = get_employee_id
    @employee_contact = EmployeeContactHelper.find(id)

    respond_to do |fmt|
      fmt.html
      fmt.json { render :json => @employee_contact }
    end
  end

  # Update the contact.
  # POST /contact/update
  def update
    id = get_employee_id
    oc = EmployeeContactHelper.find(id)

    oc_new = false
    if oc.id.blank?
      o = Employee.find(id)
      oc = EmployeeContactHelper.employee_contact_obj(o, params)
      oc_new = true
    end

    if oc_new
      if oc.save
        render :json => { :success => 1,
                           :message => 'Contact Details was successfully updated.' }

      else
        render :json => EmployeeContactHelper.get_errors(oc.errors)
      end
    else
      if EmployeeContactHelper.update_obj(oc, params)
        render :json => { :success => 1,
                           :message => 'Contact Details was successfully updated.' }

      else
        render :json => EmployeeContactHelper.get_errors(oc.errors)
      end
    end
  end
end

```

```

E:\workspace\payroll_rails\app\controllers\user\hourly_payroll_chart_controller.rb

# This controller serves incoming requests to display Hourly Payroll Chart.
class User::HourlyPayrollChartController < User::UserController

  # Display the main page.
  # GET /hourly/chart
  def index
    @month_hash = month_options

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:chart] }
    end
  end

  # Get the hourly payroll json data to be populated to the chart.
  # POST /hourly/chart/data
  def data
    month = params[:month].blank? ? '0' : params[:month]
    year = params[:year].blank? ? 0 : params[:year].to_i
    staff_id = get_staff_id

    title = 'Hourly Payroll'
    xaxis = 'Total Amount (RM)'
  end

```

```

months = I18n.t('date.month_names')

o = Array.new(12) { |x| [months[x + 1], 0] }
b = Array.new(12) { |x| 0 }
categories = Array.new(12) { |x| months[x + 1][0..2] }

if year != 0
  listyear = [year]
  title = "Hourly Payroll for #{year}"
else
  list = PayRate.select('distinct(year)').all
  listyear = list.collect { |x| x.year }
end

if month != '0'
  listmonth = month.collect { |x| x.to_i }
else
  list = PayRate.select('distinct(month)').all
  listmonth = list.collect { |x| x.month }
end

listyear.each do |y|
  listmonth.each do |m|
    filters = { :year => y, :month => m, :staff_id => staff_id }
    total_hours = AttendanceHelper.get_total_hours(filters)
    rate = PayRateHelper.get_pay_rate(filters)
    v = total_hours * rate
    o[m - 1][1] = v
    b[m - 1] += v
  end
end

c = b.collect { |x| x.round(2) }

@data = { :pie => o, :column => { :data => c,
                                      :categories => categories,
                                      :yaxis => yaxis },
          :title => title }

render :json => @data
end
end

E:\workspace\payroll_rails\app\controllers\user\info_controller.rb

# This controller serves incoming requests to display out the Employee record.
class User::InfoController < User::UserController

  # Display the information.
  # GET /info
  # GET /info.json
  def index
    id = get_employee_id
    @employee = Employee.find(id)
    @user = @employee.user

    respond_to do |fmt|
      fmt.html
      fmt.json { render :json => @employee }
    end
  end

  # Update the information.
  # POST /info/update
  def update
    id = get_employee_id
    o = Employee.find(id)

    if EmployeeHelper.update_info(o, params)
      render :json => { :success => 1,
                      :message => 'Personal Details was successfully updated.' }
    else
      render :json => EmployeeHelper.get_errors(o.errors)
    end
  end
end

```

```

    end
end

E:\workspace\payroll_rails\app\controllers\user\job_controller.rb

# This controller serves incoming requests to display out the EmployeeJob record.
class User::JobController < User::UserController

  # Display the information.
  # GET /job
  # GET /job.json
  def index
    id = get_employee_id
    @employee_job = EmployeeJobHelper.find(id)

    respond_to do |fmt|
      fmt.html
      fmt.json { render :json => @employee_job }
    end
  end
end

E:\workspace\payroll_rails\app\controllers\user\overtime_chart_controller.rb

# This controller serves incoming requests to display Employee Overtime Chart.
class User::OvertimeChartController < User::UserController

  # Display the main page.
  # GET /overtime/chart
  def index
    @month_hash = month_options

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:chart] }
    end
  end

  # Get the overtime json data to be populated to the chart.
  # GET /overtime/chart/data
  def data
    month = params[:month].blank? ? '0' : params[:month]
    year = params[:year].blank? ? 0 : params[:year].to_i
    staff_id = get_staff_id

    criteria = Attendance
    title = 'Overtime'
    yaxis = 'Duration (hours)'

    criteria = criteria.where(:staff_id => staff_id)

    if year != 0
      criteria = criteria.where('year(work_date) = ?', year)
      title = "Overtime for Year #{year}"
    end

    if month != '0'
      criteria = criteria.where('month(work_date) in (?)', month)
    end

    list = criteria.order(:work_date).all
    months = I18n.t('date.month_names')

    b = Array.new(12) { |x| 0 }
    categories = Array.new(12) { |x| months[x + 1][0..2] }
    list.each do |o|
      to = ApplicationHelper.localtime(o.time_out)
      v = Time.new(to.year, to.month, to.day, 18, 0, 0, '+08:00')
      x = (to - v) / 3600.0
      m = o.work_date.month
      b[m - 1] += x
    end

    c = b.collect { |x| x.round(2) }

    @data = { :data => c, :categories => categories, :title => title, :yaxis => yaxis }

    render :json => @data
  end

```

```

    end
end

E:\workspace\payroll_rails\app\controllers\user\payslip_controller.rb

# This controller serves incoming requests to display out the Employee pay slip.
class User::PayslipController < User::UserController

  # Display the main page.
  # GET /payslip
  def index
    respond_to do |fmt|
      fmt.html
    end
  end

  # Display the pay slip information.
  # GET /payslip/slip
  # GET /payslip/slip.json
  def payslip
    id = params[:employee_id]
    @employee = Employee.find(id)
    @employee_salary = @employee.employee_salary

    _month = params[:month].to_i
    month = month_name(params[:month].to_i)
    year = params[:year].blank? ? Time.now.year : params[:year].to_i
    @period = "#{month}-#{year}"

    if @employee_salary.blank?
      @total_earnings = 0
      @total_deduct = 0
      @nett_salary = 0

      @employee_salary = EmployeeSalary.new

      respond_to do |fmt|
        fmt.html { render 'admin/payslip/payslip_monthly' }
        fmt.json { render :json => [@employee, @total_earnings, @total_deduct,
                                      @nett_salary] }
      end
    end

    else
      # checks whether the salary type is monthly
      if @employee_salary.pay_type == 1
        filters = { :year => year, :month => _month, :staff_id => @employee.staff_id }
        @total_overtime = PayslipHelper.total_overtime(filters)
        @total_overtime_earnings = PayslipHelper.total_overtime_earnings(filters,
          @total_overtime)
        @adjustment = SalaryAdjustmentHelper.get_salary_adjustment(filters)

        @total_earnings = PayslipHelper.total_earnings(@employee_salary, @adjustment,
          @total_overtime_earnings)
        @total_deduct = PayslipHelper.total_deductions(@employee_salary)
        @nett_salary = PayslipHelper.nett_salary(@total_earnings, @total_deduct)

        @basic_pay = @employee_salary.salary + @adjustment

        respond_to do |fmt|
          fmt.html { render 'admin/payslip/payslip_monthly' }
          fmt.json { render :json => [@employee, @total_earnings, @total_deduct,
                                        @nett_salary, @total_overtime,
                                        @total_overtime_earnings, @adjustment, @basic_pay] }
        end
      end

      # hourly type
      else
        filters = { :year => year, :month => _month, :staff_id => @employee.staff_id }
        @total_earnings, @total_hours, @hourly_pay_rate =
          PayslipHelper.total_earnings_hourly(@employee_salary, filters)
        @total_deduct = PayslipHelper.total_deductions(@employee_salary)
        @nett_salary = PayslipHelper.nett_salary_hourly(@total_earnings, @total_deduct)

        respond_to do |fmt|
          fmt.html { render 'admin/payslip/payslip_hourly' }
          fmt.json { render :json => [@employee, @total_earnings, @total_deduct,
                                        @nett_salary, @total_hours, @hourly_pay_rate] }
        end
      end
    end
  end
end

```

```

        end
    end
end
end
end

E:\workspace\payroll_rails\app\controllers\user\qualification_controller.rb

# This controller serves incoming requests to display out the EmployeeQualification record.
class User::QualificationController < User::UserController

# Display the qualification.
# GET /qualification
# GET /qualification.json
def index
id = get_employee_id
@employee_qualification = EmployeeQualificationHelper.find(id)

respond_to do |fmt|
  fmt.html
  fmt.json { render :json => @employee_qualification }
end
end

# Update the qualification.
# POST /qualification/update
def update
id = get_employee_id
oq = EmployeeQualificationHelper.find(id)

oq_new = false
if oq.id.blank?
  o = Employee.find(id)
  oq = EmployeeQualificationHelper.employee_qualification_obj(o, params)
  oq_new = true
end

if oq_new
  if oq.save
    render :json => { :success => 1,
                      :message => 'Qualifications was successfully updated.' }

  else
    render :json => EmployeeQualificationHelper.get_errors(oq.errors)
  end
else
  if EmployeeQualificationHelper.update_obj(oq, params)
    render :json => { :success => 1,
                      :message => 'Qualifications was successfully updated.' }

  else
    render :json => EmployeeQualificationHelper.get_errors(oq.errors)
  end
end
end
end

```

```

E:\workspace\payroll_rails\app\controllers\user\salary_controller.rb

# This controller serves incoming requests to display out the EmployeeSalary record.
class User::SalaryController < User::UserController

  # Display the information.
  # GET /salary
  # GET /salary.json
  def index
    id = get_employee_id
    @employee = Employee.find(id)
    @employee_salary = EmployeeSalaryHelper.find(id)
    @adjustment = SalaryAdjustmentHelper.get_salary_adjustment(
      :staff_id => @employee.staff_id, :year => Time.now.year)
    @basic_pay = @employee_salary.salary + @adjustment

    respond_to do |fmt|
      fmt.html
      fmt.json { render :json => [@employee_salary, @adjustment] }
    end
  end
end

E:\workspace\payroll_rails\app\controllers\user\total_work_hours_chart_controller.rb

# This controller serves incoming requests to display Employee Total Hours Worked Chart.
class User::TotalWorkHoursChartController < User::UserController

  # Display the main page.
  # GET /workhours/chart
  def index
    @month_hash = month_options

    respond_to do |fmt|
      fmt.html { render :layout => LAYOUT[:chart] }
    end
  end

  # Get the total hours worked json data to be populated to the chart.
  # GET /workhours/chart/data
  def data
    month = params[:month].blank? ? '0' : params[:month]
    year = params[:year].blank? ? 0 : params[:year].to_i
    staff_id = get_staff_id

    title = 'Total Hours Worked'
    yaxis = 'Duration (hours)'

    months = I18n.t('date.month_names')

    b = Array.new(12) { |x| 0 }
    categories = Array.new(12) { |x| months[x + 1][0..2] }

    liststaff = [staff_id]

    if year != 0
      listyear = [year]
      title = "Total Hours Worked for #{year}"
    else
      list = PayRate.select('distinct(year)').all
      listyear = list.collect { |x| x.year }
    end

    if month != '0'
      listmonth = month.collect { |x| x.to_i }
    else
      list = PayRate.select('distinct(month)').all
      listmonth = list.collect { |x| x.month }
    end

    listyear.each do |y|
      listmonth.each do |m|
        liststaff.each do |s|
          filters = { :year => y, :month => m, :staff_id => s }
          total_hours = AttendanceHelper.get_total_hours(filters)
        end
      end
    end
  end
end

```

```

        v = total_hours
        b[m - 1] += v
    end
end
end

c = b.collect { |x| x.round(2) }

@data = { :data => c, :categories => categories, :title => title, :yaxis => yaxis }

render :json => @data
end
end

E:\workspace\payroll_rails\app\controllers\user\user_controller.rb

# User base controller.
# All User controllers must inherit this controller.
# User controller serves incoming requests from an authenticated Normal User.
class User::UserController < ApplicationController
layout false

# checks whether a user is authenticated before serving any request
before_filter :authenticate_normal_user

# Display the user page.
# GET /user/index
def index
id = get_employee_id
@employee_salary = EmployeeSalaryHelper.find(id)
@pay_type = @employee_salary.pay_type

respond_to do |fmt|
  fmt.html { render :layout => LAYOUT[:user] }
end
end

protected

# Returns the employee id.
def normal_user
return unless session[:employee_id]
@normal_user ||= session[:employee_id]
end

# Checks whether a normal user is authenticated.
def authenticate_normal_user
unless authenticate
  return false
end
logged_in_normal_user? ? true : access_denied
end

# Checks whether a normal user is logged in.
def logged_in_normal_user?
  normal_user.present?
end

# Returns the employee_id.
def get_employee_id
  session[:employee_id]
end

# Returns the staff_id.
def get_staff_id
  session[:staff_id]
end

helper_method :normal_user
helper_method :logged_in_normal_user
end

```

```

E:\workspace\payroll_rails\app\helpers\application_helper.rb

module ApplicationHelper
  # Pager object for paging purpose
  class Pager
    attr_accessor :total, :pagenum
    attr_reader :pagesize

    @@default_page_size = 50

    def initialize(total, pagenum, pagesize)
      @total = total
      @pagenum = pagenum
      set_pagesize(pagesize)
    end

    def pagesize=(pagesize)
      set_pagesize(pagesize)
    end

    # Get the paging message.
    def item_message
      Utils.item_message(total, pagenum, pagesize)
    end

    # Get the lower bound of the page.
    def lower_bound
      (pagenum - 1) * pagesize
    end

    # Get the upper bound of the page.
    def upper_bound
      upperbound = pagenum * pagesize
      if total < upperbound
        upperbound = total
      end
      upperbound
    end

    # Checks whether the pager has next page.
    def has_next?
      total > upper_bound ? true : false
    end

    # Checks whether the pager has previous page.
    def has_prev?
      lower_bound > 0 ? true : false
    end

    # Get the total pages.
    def total_pages
      (Float(total) / pagesize).ceil
    end

    # Get the default page size.
    def self.default_page_size
      @@default_page_size
    end

    private

    def set_pagesize(pagesize)
      if (total < pagesize || pagesize < 1) && total > 0
        @pagesize = total
      else
        @pagesize = pagesize
      end

      if total_pages < @pagenum
        @pagenum = total_pages
      end

      if @pagenum < 1
        @pagenum = 1
      end
    end
  end
end

```

```

end

module Utils
  require 'time'

  # Get the paging message.
  def self.item_message(total, pagenum, pagesize)
    x = (pagenum - 1) * pagesize + 1
    y = pagenum * pagesize

    if total < y
      y = total
    end

    if total < 1
      return ""
    end

    "#{x} to #{y} of #{total}"
  end

  # Checks for numeric.
  def self.numeric?(val)
    true if Float(val) rescue false
  end
end

class Sort
  attr_accessor :column, :direction

  def initialize(column, dir = 'ASC')
    @column = column
    @direction = dir
  end

  def to_s
    "@{@column} #{@direction}"
  end
end

# Get month name for the specified month number,
# e.g return January for 1
def self.month_name(i)
  Date::MONTHNAMES[i]
end

def self.localtime(t)
  t.in_time_zone('Kuala Lumpur')
end

def self.date_fmt
  '%d-%m-%Y'
end
end

E:\workspace\payroll_rails\app\helpers\attendance_helper.rb

module AttendanceHelper
  DEFAULT_SORT_COLUMN = 'work_date'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all attendance records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = Attendance.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0

    if sort.column == 'e.first_name'
      criteria = get_join({}, sort)
    else
      criteria = Attendance
    end
  end

```

```

list = criteria.order(order).all(:offset => pager.lower_bound,
                               :limit => pager.pagesize)
{ :item_msg => pager.item_message,
  :hasnext => has_next,
  :hasprev => has_prev,
  :nextpage => pager.pagenum + 1,
  :prevpage => pager.pagenum - 1,
  :list => list,
  :sortcolumn => sort.column,
  :sortdir => sort.direction,
  :page => pager.pagenum,
  :totalpage => pager.total_pages }
end

# Get filtered attendance records.
def self.get_filter_by(filters, pagenum = 1,
  pagesize = ApplicationHelper::Pager.default_page_size,
  sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
  criteria, order = get_filter_criteria(filters, sort)
  total = criteria.count
  pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)

  has_next = pager.has_next? ? 1 : 0
  has_prev = pager.has_prev? ? 1 : 0
  list = criteria.order(order).all(:offset => pager.lower_bound,
                                   :limit => pager.pagesize)
  { :item_msg => pager.item_message,
    :hasnext => has_next,
    :hasprev => has_prev,
    :nextpage => pager.pagenum + 1,
    :prevpage => pager.pagenum - 1,
    :list => list,
    :sortcolumn => sort.column,
    :sortdir => sort.direction,
    :page => pager.pagenum,
    :totalpage => pager.total_pages }
end

# Get the total hours worked
def self.get_total_hours(filters)
  criteria = Attendance.where('year(work_date) = ? and month(work_date) = ?',
    filters[:year], filters[:month])
  if filters[:staff_id].present?
    criteria = criteria.where('staff_id = ?', filters[:staff_id])
  end

  list = criteria.all
  total_hours = 0
  list.each do |o|
    to = ApplicationHelper.localtime(o.time_out.localtime)
    ti = ApplicationHelper.localtime(o.time_in.localtime)
    total_hours += (to - ti) / 3600.0
  end

  total_hours
end

private

def self.get_filter_criteria(filters, sort = nil)
  employee_keyword = "%#{filters[:employee]}%"
  order = sort.present? ? sort.to_s : nil
  if filters[:employee].present? || sort.present?
    criteria = get_join(filters, sort)
  end

  criteria = Attendance if criteria.blank?

  if filters[:work_date].present?
    criteria = criteria.where('work_date = ?', filters[:work_date])
  end

  if filters[:employee].present?
    criteria = criteria.where(
      'e.first_name like ? or e.middle_name like ? or e.last_name like ?',
      employee_keyword, employee_keyword, employee_keyword)
  end

```

```

    end

    return criteria, order
end

def self.get_join(filters, sort = nil)
joinhash = {}

if filters.any?
  if filters[:employee].present?
    q = Attendance.joins('inner join employee e on attendance.staff_id = e.staff_id')
    joinhash[:employee] = true
  end
end

if sort.present?
  if sort.column == 'e.first_name'
    if !joinhash.has_key?(:employee)
      q = Attendance.joins(
        'left outer join employee e on attendance.staff_id = e.staff_id')
    end
  end
end

q
end
end

```

E:\workspace\payroll_rails\app\helpers\department_helper.rb

```

module DepartmentHelper
  DEFAULT_SORT_COLUMN = 'name'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all department records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = Department.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = Department.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered department records.
  def self.get_filter_by(keyword, pagenum = 1,
    pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    criteria = Department.where('name like ?', "%#{keyword}%")
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = criteria.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction }
  end
end

```

```

    :sortdir => sort.direction,
    :page => pager.pagenum,
    :totalpage => pager.total_pages }
end

# Get the validation errors.
def self.get_errors(errors)
  { :error => 1, :errors => errors }
end

# Get the item message text.
def self.item_message(keyword, pagenum, pagesize)
  total = 0
  if keyword.blank?
    total = Department.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    return pager.item_message
  else
    criteria = Department.where('name like ?', "%#{keyword}%")
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    return pager.item_message
  end
end
end

```

E:\workspace\payroll_rails\app\helpers\designation_helper.rb

```

module DesignationHelper
  DEFAULT_SORT_COLUMN = 'title'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all designation records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = Designation.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = Designation.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered designation records.
  def self.get_filter_by(find, keyword, pagenum = 1,
    pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    criteria, order = get_filter_criteria(find, keyword, sort)
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = criteria.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end
end

```

```

    :page => pager.pagenum,
    :totalpage => pager.total_pages }
end

# Get the validation errors.
def self.get_errors(errors)
  { :error => 1, :errors => errors }
end

# Get the item message text.
def self.item_message(keyword, pagenum, pagesize)
  total = 0
  if keyword.blank?
    total = Designation.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    return pager.item_message
  end
  else
    criteria = Designation.where('title like ?', "%#{keyword}%")
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    return pager.item_message
  end
end

private

def self.get_filter_criteria(find, keyword, sort = nil)
  text = "%#{keyword}%""
  order = sort.present? ? sort.to_s : nil
  criteria = Designation

  case find
  when 1
    criteria = criteria.where('title like ?', text)
    return criteria, order

  when 2
    criteria = criteria.where(`desc` like ?, text)
    return criteria, order

  when 3
    criteria = criteria.where('note like ?', text)
    return criteria, order

  else
    criteria = criteria.where('title like ? or `desc` like ? or note like ?', text,
      text, text)
    return criteria, order
  end
end
end

```

E:\workspace\payroll_rails\app\helpers\employee_contact_helper.rb

```

module EmployeeContactHelper
  # Get the validation errors.
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Checks whether any contact parameters are present in the POST request.
  def self.is_empty_params?(params)
    q = params[:employee_contact]
    if q[:address_1].blank? && q[:address_2].blank? && q[:address_3].blank? &&
      q[:city].blank? && q[:state].blank? && q[:postcode].blank? && q[:country].blank? &&
      q[:home_phone].blank? && q[:mobile_phone].blank? && q[:work_email].blank? &&
      q[:other_email].blank?
      return true
    end
    false
  end

  # Get the contact object from the POST request.
  def self.employee_contact_obj(o, params)
    q = params[:employee_contact]
  end

```

```

EmployeeContact.new(:id => o.id, :address_1 => q[:address_1],
                    :address_2 => q[:address_2], :address_3 => q[:address_3],
                    :city => q[:city], :state => q[:state], :postcode => q[:postcode],
                    :country => q[:country], :home_phone => q[:home_phone],
                    :mobile_phone => q[:mobile_phone], :work_email => q[:work_email],
                    :other_email => q[:other_email])
end

# Update the contact object.
def self.update_obj(o, params)
  q = params[:employee_contact]

  o.update_attributes(:address_1 => q[:address_1], :address_2 => q[:address_2],
                      :address_3 => q[:address_3], :city => q[:city],
                      :state => q[:state], :postcode => q[:postcode],
                      :country => q[:country], :home_phone => q[:home_phone],
                      :mobile_phone => q[:mobile_phone], :work_email => q[:work_email],
                      :other_email => q[:other_email])
end

# Get the contact object.
def self.find(id)
  o = nil
  begin
    o = EmployeeContact.find(id)

    rescue Exception => e
      o = EmployeeContact.new
    end
  end
end

```

E:\workspace\payroll_rails\app\helpers\employee_helper.rb

```

require 'securerandom'

module EmployeeHelper
  DEFAULT_SORT_COLUMN = 'staff_id'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all employee records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
                  sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = Employee.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0

    if sort.column == 'd.title' || sort.column == 'es.name' ||
       sort.column == 'dept.name' || sort.column == 'e.first_name'
      criteria = get_join({}, sort)
    else
      criteria = Employee
    end

    list = criteria.order(order).all(:offset => pager.lower_bound,
                                      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered employee records.
  def self.get_filter_by(filters, pagenum = 1,
                        pagesize = ApplicationHelper::Pager.default_page_size,
                        sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))

```

```

criteria, order = get_filter_criteria(filters, sort)
total = criteria.count
pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)

has_next = pager.has_next? ? 1 : 0
has_prev = pager.has_prev? ? 1 : 0
list = criteria.order(order).all(:offset => pager.lower_bound,
                                  :limit => pager.pagesize)
{ :item_msg => pager.item_message,
  :hasnext => has_next,
  :hasprev => has_prev,
  :nextpage => pager.pagenum + 1,
  :prevpage => pager.pagenum - 1,
  :list => list,
  :sortcolumn => sort.column,
  :sortdir => sort.direction,
  :page => pager.pagenum,
  :totalpage => pager.total_pages }
end

# Get the validation errors.
def self.get_errors(errors)
  { :error => 1, :errors => errors }
end

# Get the item message text.
def self.item_message(filters, pagenum, pagesize)
  total = 0
  if filters.blank?
    total = Employee.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    return pager.item_message
  else
    criteria, order = get_filter_criteria(filters)
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    return pager.item_message
  end
end

# Get the employee object from the POST request.
def self.employee_obj(params)
  q = params[:employee]

  _dob = q[:dob]
  dob = Date.strptime(_dob, ApplicationHelper.date_fmt) if _dob.present?

  Employee.new(:id => SecureRandom.uuid, :staff_id => q[:staff_id],
               :first_name => q[:first_name], :middle_name => q[:middle_name],
               :last_name => q[:last_name], :new_ic => q[:new_ic],
               :old_ic => q[:old_ic], :passport_no => q[:passport_no],
               :gender => q[:gender], :marital_status => q[:marital_status],
               :nationality => q[:nationality], :dob => dob,
               :place_of_birth => q[:place_of_birth], :race => q[:race],
               :religion => q[:religion], :is_bumi => q[:is_bumi],
               :user_id => q[:user_id])
end

def self.update_obj(o, params)
  q = params[:employee]

  _dob = q[:dob]
  dob = Date.strptime(_dob, ApplicationHelper.date_fmt) if _dob.present?

  o.update_attributes(:staff_id => q[:staff_id], :first_name => q[:first_name],
                      :middle_name => q[:middle_name], :last_name => q[:last_name],
                      :new_ic => q[:new_ic], :old_ic => q[:old_ic],
                      :passport_no => q[:passport_no], :gender => q[:gender],
                      :marital_status => q[:marital_status],
                      :nationality => q[:nationality], :dob => dob,
                      :place_of_birth => q[:place_of_birth], :race => q[:race],
                      :religion => q[:religion], :is_bumi => q[:is_bumi],
                      :user_id => q[:user_id])
end

# Update the employee object.

```

```

def self.update_info(o, params)
  q = params[:employee]

  _dob = q[:dob]
  dob = Date.strptime(_dob, ApplicationHelper.date_fmt) if _dob.present?

  o.update_attributes(:first_name => q[:first_name], :middle_name => q[:middle_name],
                      :last_name => q[:last_name], :new_ic => q[:new_ic],
                      :old_ic => q[:old_ic], :passport_no => q[:passport_no],
                      :gender => q[:gender], :marital_status => q[:marital_status],
                      :nationality => q[:nationality], :dob => dob,
                      :place_of_birth => q[:place_of_birth], :race => q[:race],
                      :religion => q[:religion], :is_bumi => q[:is_bumi])
end

private

def self.get_filter_criteria(filters, sort = nil)
  employee_keyword = "%#{filters[:employee]}%"
  staff_id_keyword = "%#{filters[:staff_id]}%"
  supervisor_keyword = "%#{filters[:supervisor]}%"
  order = sort.present? ? sort.to_s : nil
  if filters[:employment_status] != 0 || filters[:designation] != 0 ||
    filters[:dept] != 0 || sort.present?
    criteria = get_join(filters, sort)
  end

  criteria = Employee if criteria.blank?

  if filters[:employee].present?
    criteria = criteria.where(
      'first_name like ? or middle_name like ? or last_name like ?',
      employee_keyword, employee_keyword, employee_keyword)
  end

  if filters[:staff_id].present?
    criteria = criteria.where('staff_id like ?', staff_id_keyword)
  end

  if filters[:employment_status] != 0
    criteria = criteria.where('es.id = ?', filters[:employment_status])
  end

  if filters[:designation] != 0
    criteria = criteria.where('d.id = ?', filters[:designation])
  end

  if filters[:dept] != 0
    criteria = criteria.where('dept.id = ?', filters[:dept])
  end

  return criteria, order
end

def self.get_join(filters, sort = nil)
  joinhash = {}

  if filters.any?
    if filters[:employment_status] != 0
      q = Employee.joins('inner join employee_job ej on employee.id = ej.id')
                  .joins(
                    'inner join employment_status es on ej.employment_status_id = es.id')
      joinhash[:employment_status] = true
    end

    if filters[:designation] != 0
      if q.blank?
        q = Employee.joins('inner join employee_job ej on employee.id = ej.id')
                  .joins('inner join designation d on ej.designation_id = d.id')
      end

      else
        q = q.joins('inner join designation d on ej.designation_id = d.id')
      end

      joinhash[:designation] = true
    end
  end

```

```

if filters[:dept] != 0
  if q.blank?
    q = Employee.joins('inner join employee_job ej on employee.id = ej.id')
      .joins('inner join department dept on ej.department_id = dept.id')

  else
    q = q.joins('inner join department dept on ej.department_id = dept.id')
  end

  joinhash[:dept] = true
end
end

if sort.present?
  if sort.column == 'd.title'
    if joinhash.has_key?(:employment_status) && !joinhash.has_key?(:designation)
      q = q.joins('left outer join designation d on ej.designation_id = d.id')

    elsif !joinhash.has_key?(:employment_status) && !joinhash.has_key?(:designation)
      if q.blank?
        q = Employee.joins('left outer join employee_job ej on employee.id = ej.id')
          .joins('left outer join designation d on ej.designation_id = d.id')

      else
        q = q.joins('left outer join employee_job ej on employee.id = ej.id')
          .joins('left outer join designation d on ej.designation_id = d.id')
      end
    end
  end

  if sort.column == 'es.name'
    if !joinhash.has_key?(:employment_status)
      if q.blank?
        q = Employee.joins('left outer join employee_job ej on employee.id = ej.id')
          .joins(
            'left outer join employment_status es on ej.employment_status_id = es.id')

      else
        q = q.joins('left outer join employee_job ej on employee.id = ej.id')
          .joins(
            'left outer join employment_status es on ej.employment_status_id = es.id')
      end
    end
  end

  if sort.column == 'dept.name'
    if joinhash.has_key?(:employment_status) && !joinhash.has_key?(:dept)
      q = q.joins('left outer join department dept on ej.department_id = dept.id')

    elsif !joinhash.has_key?(:employment_status) && !joinhash.has_key?(:dept)
      if q.blank?
        q = Employee.joins('left outer join employee_job ej on employee.id = ej.id')
          .joins(
            'left outer join department dept on ej.department_id = dept.id')

      else
        q = q.joins('left outer join employee_job ej on employee.id = ej.id')
          .joins('left outer join department dept on ej.department_id = dept.id')
      end
    end
  end

  q
end
end

```

```

E:\workspace\payroll_rails\app\helpers\employee_job_helper.rb

module EmployeeJobHelper
  # Get the validation errors.
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Checks whether any job parameters are present in the POST request.
  def self.is_empty_params?(params)
    q = params[:employee_job]
    if q[:designation_id] == '0' && q[:department_id] == '0' &&
      q[:employment_status_id] == '0' && q[:job_category_id] == '0' &&
      q[:join_date].blank? && q[:confirm_date].blank?
      return true
    end
    false
  end

  # Get the job object from the POST request.
  def self.employee_job_obj(o, params)
    q = params[:employee_job]

    _join_date = q[:join_date]
    _confirm_date = q[:confirm_date]
    join_date = Date.strptime(_join_date,
      ApplicationHelper.date_fmt) if _join_date.present?
    confirm_date = Date.strptime(_confirm_date,
      ApplicationHelper.date_fmt) if _confirm_date.present?

    EmployeeJob.new(:id => o.id, :designation_id => q[:designation_id],
      :department_id => q[:department_id],
      :employment_status_id => q[:employment_status_id],
      :job_category_id => q[:job_category_id], :join_date => join_date,
      :confirm_date => confirm_date)
  end

  # Update the job object.
  def self.update_obj(o, params)
    q = params[:employee_job]

    _join_date = q[:join_date]
    _confirm_date = q[:confirm_date]
    join_date = Date.strptime(_join_date,
      ApplicationHelper.date_fmt) if _join_date.present?
    confirm_date = Date.strptime(_confirm_date,
      ApplicationHelper.date_fmt) if _confirm_date.present?

    o.update_attributes(:designation_id => q[:designation_id],
      :department_id => q[:department_id],
      :employment_status_id => q[:employment_status_id],
      :job_category_id => q[:job_category_id], :join_date => join_date,
      :confirm_date => confirm_date)
  end

  # Get the job object.
  def self.find(id)
    o = nil
    begin
      o = EmployeeJob.find(id)

      rescue Exception => e
        o = EmployeeJob.new
      end
    o
  end
end

```

```

E:\workspace\payroll_rails\app\helpers\employee_qualification_helper.rb

module EmployeeQualificationHelper
  # Get the validation errors.
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Checks whether any qualification parameters are present in the POST request.
  def self.is_empty_params?(params)
    q = params[:employee_qualification]
    if q[:level] == '0' && q[:institute].blank? && q[:major].blank? && q[:year].blank? &&
      q[:gpa].blank? && q[:start_date].blank? && q[:end_date].blank?
      return true
    end
    false
  end

  # Get the qualification object from the POST request.
  def self.employee_qualification_obj(o, params)
    q = params[:employee_qualification]

    _start_date = q[:start_date]
    _end_date = q[:end_date]

    start_date = Date.strptime(_start_date,
      ApplicationHelper.date_fmt) if _start_date.present?
    end_date = Date.strptime(_end_date, ApplicationHelper.date_fmt) if _end_date.present?

    EmployeeQualification.new(:id => o.id, :level => q[:level],
      :institute => q[:institute], :major => q[:major],
      :year => q[:year], :gpa => q[:gpa],
      :start_date => start_date, :end_date => end_date)
  end

  # Update the qualification object.
  def self.update_obj(o, params)
    q = params[:employee_qualification]

    _start_date = q[:start_date]
    _end_date = q[:end_date]

    start_date = Date.strptime(_start_date,
      ApplicationHelper.date_fmt) if _start_date.present?
    end_date = Date.strptime(_end_date, ApplicationHelper.date_fmt) if _end_date.present?

    o.update_attributes(:level => q[:level], :institute => q[:institute],
      :major => q[:major], :year => q[:year], :gpa => q[:gpa],
      :start_date => start_date, :end_date => end_date)
  end

  # Get the contact object.
  def self.find(id)
    o = nil
    begin
      o = EmployeeQualification.find(id)

      rescue Exception => e
        o = EmployeeQualification.new
      end
    o
  end
end

```

```

E:\workspace\payroll_rails\app\helpers\employee_salary_helper.rb

module EmployeeSalaryHelper
  # Get the validation errors
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Checks whether any salary parameters are present in the POST request.
  def self.is_empty_params?(params)
    q = params[:employee_salary]
    if q[:salary].blank? && q[:allowance].blank? && q[:epf].blank? && q[:socso].blank? &&
      q[:bank_name].blank? && q[:bank_acc_no].blank? && q[:bank_acc_type].blank? &&
      q[:bank_address].blank? && q[:epf_no].blank? && q[:socso_no].blank? &&
      q[:income_tax_no].blank?
      return true
    elsif q[:salary] == '0' && q[:allowance] == '0' && q[:epf] == '0' &&
      q[:socso] == '0' && q[:bank_name].blank? && q[:bank_acc_no].blank? &&
      q[:bank_acc_type].blank? && q[:bank_address].blank? && q[:epf_no].blank? &&
      q[:socso_no].blank? && q[:income_tax_no].blank?
      return true
    end
    false
  end

  # Get the salary object from the POST request.
  def self.employee_salary_obj(o, params)
    q = params[:employee_salary]

    EmployeeSalary.new(:id => o.id, :salary => q[:salary], :allowance => q[:allowance],
                       :epf => q[:epf], :socso => q[:socso],
                       :income_tax => q[:income_tax], :bank_name => q[:bank_name],
                       :bank_acc_no => q[:bank_acc_no],
                       :bank_acc_type => q[:bank_acc_type],
                       :bank_address => q[:bank_address], :epf_no => q[:epf_no],
                       :socso_no => q[:socso_no], :income_tax_no => q[:income_tax_no],
                       :pay_type => q[:pay_type])
  end

  # Update the salary object.
  def self.update_obj(o, params)
    q = params[:employee_salary]

    o.update_attributes(:salary => q[:salary], :allowance => q[:allowance],
                        :epf => q[:epf], :socso => q[:socso],
                        :income_tax => q[:income_tax], :bank_name => q[:bank_name],
                        :bank_acc_no => q[:bank_acc_no],
                        :bank_acc_type => q[:bank_acc_type],
                        :bank_address => q[:bank_address], :epf_no => q[:epf_no],
                        :socso_no => q[:socso_no], :income_tax_no => q[:income_tax_no],
                        :pay_type => q[:pay_type])
  end

  # Get the salary object.
  def self.find(id)
    o = nil
    begin
      o = EmployeeSalary.find(id)

      rescue Exception => e
        o = EmployeeSalary.new
      end
    end
  end
end

```

```

E:\workspace\payroll_rails\app\helpers\employment_status_helper.rb

module EmploymentStatusHelper
  DEFAULT_SORT_COLUMN = 'name'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all employment status records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = EmploymentStatus.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = EmploymentStatus.order(order).all(:offset => pager.lower_bound,
                                              :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered employment status records.
  def self.get_filter_by(keyword, pagenum = 1,
    pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    criteria = EmploymentStatus.where('name like ?', "%#{keyword}%")
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = criteria.order(order).all(:offset => pager.lower_bound,
                                      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get the validation errors.
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Get the item message text.
  def self.item_message(keyword, pagenum, pagesize)
    total = 0
    if keyword.blank?
      total = EmploymentStatus.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message
    else
      criteria = EmploymentStatus.where('name like ?', "%#{keyword}%")
      total = criteria.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message
    end
  end
end

```

```

E:\workspace\payroll_rails\app\helpers\job_category_helper.rb

module JobCategoryHelper
  DEFAULT_SORT_COLUMN = 'name'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all job category records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = JobCategory.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = JobCategory.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered job category records.
  def self.get_filter_by(keyword, pagenum = 1,
    pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    criteria = JobCategory.where('name like ?', "%#{keyword}%")
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = criteria.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get the validation errors.
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Get the item message text.
  def self.item_message(keyword, pagenum, pagesize)
    total = 0
    if keyword.blank?
      total = JobCategory.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message
    else
      criteria = JobCategory.where('name like ?', "%#{keyword}%")
      total = criteria.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message
    end
  end
end

```

```

E:\workspace\payroll_rails\app\helpers\overtime_rate_helper.rb

module OvertimeRateHelper
  DEFAULT_SORT_COLUMN = 'year'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all overtime rate records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = OvertimeRate.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = OvertimeRate.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered overtime rate records.
  def self.get_filter_by(filters, pagenum = 1,
    pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    criteria, order = get_filter_criteria(filters, sort)
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = criteria.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get the validation errors.
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Get the item message text.
  def self.item_message(filters, pagenum, pagesize)
    total = 0
    if filters.blank?
      total = OvertimeRate.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message

    else
      criteria, order = get_filter_criteria(filters)
      total = criteria.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message
    end
  end

  private

  def self.get_filter_criteria(filters, sort = nil)

```

```

order = sort.present? ? sort.to_s : nil
criteria = OvertimeRate
if filters[:year] != 0
  criteria = criteria.where(:year => filters[:year])
end

return criteria, order
end
end

E:\workspace\payroll_rails\app\helpers\pay_rate_helper.rb

module PayRateHelper
  DEFAULT_SORT_COLUMN = 'staff_id'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all pay rate records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = PayRate.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = PayRate.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered pay rate records.
  def self.get_filter_by(filters, pagenum = 1,
    pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    criteria, order = get_filter_criteria(filters, sort)
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = criteria.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get the validation errors.
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Get the item message text.
  def self.item_message(filters, pagenum, pagesize)
    total = 0
    if filters.blank?
      total = PayRate.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message
    else

```

```

        criteria, order = get_filter_criteria(filters)
        total = criteria.count
        pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
        return pager.item_message
    end
end

# Get the pay rate for a staff id in the specified year and month.
def self.get_pay_rate(filters)
    o = PayRate.where(:staff_id => filters[:staff_id])
        .where(:year => filters[:year])
        .where(:month => filters[:month]).first
    rate = 0
    rate = o.hourly_pay_rate if o.present?
    rate
end

private

def self.get_filter_criteria(filters, sort = nil)
    staff_id_keyword = "%#{filters[:staff_id]}%"
    order = sort.present? ? sort.to_s : nil
    criteria = PayRate
    if filters[:staff_id].present?
        criteria = criteria.where('staff_id like ?', staff_id_keyword)
    end

    if filters[:month] != 0
        criteria = criteria.where(:month => filters[:month])
    end

    if filters[:year] != 0
        criteria = criteria.where(:year => filters[:year])
    end

    return criteria, order
end
end

```

E:\workspace\payroll_rails\app\helpers\payslip_helper.rb

```

module PayslipHelper
    # Get the total deductions for one employee.
    def self.total_deductions(employee_salary)
        return 0.0 if employee_salary.blank?
        employee_salary.epf + employee_salary.socso + employee_salary.income_tax
    end

    # Get the total earnings for one monthly paid employee.
    def self.total_earnings(employee_salary, adjustment, total_overtime_earnings)
        return 0.0 if employee_salary.blank?
        amt = employee_salary.salary + adjustment
        amt + employee_salary.allowance + total_overtime_earnings
    end

    # Get the nett salary for one monthly paid employee.
    def self.nett_salary(earnings, deductions)
        earnings - deductions
    end

    # Get the total earnings for one hourly paid employee.
    def self.total_earnings_hourly(employee_salary, filters)
        total_hours = AttendanceHelper.get_total_hours(filters)
        rate = PayRateHelper.get_pay_rate(filters)

        earnings = (total_hours * rate) + employee_salary.allowance
        return earnings, total_hours, rate
    end

    # Get the nett salary for one hourly paid employee.
    def self.nett_salary_hourly(earnings, deductions)
        earnings - deductions
    end

    # Get the total overtime for a staff id in the specified year and month.
    def self.total_overtime(filters)
        year = filters[:year]

```

```

month = filters[:month]
id = filters[:staff_id]

list = Attendance.where(:staff_id => id)
  .where('month(work_date) = ?', month)
  .where('year(work_date) = ?', year).all

duration = 0

list.each do |o|
  to = o.time_out.localtime
  v = Time.new(to.year, to.month, to.day, 18, 0, 0)
  x = (to - v) / 3600.0
  duration += x
end

duration
end

# Get total overtime earnings for a specified year.
def self.total_overtime_earnings(filters, duration)
  year = filters[:year]
  o = OvertimeRate.where(:year => year).first
  total = 0
  if o.present?
    total = (duration / o.duration) * o.pay_rate
  end

  total
end
end

E:\workspace\payroll_rails\app\helpers\salary_adjustment_helper.rb

module SalaryAdjustmentHelper
  DEFAULT_SORT_COLUMN = 'staff_id'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all salary adjustment records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = SalaryAdjustment.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = SalaryAdjustment.order(order).all(:offset => pager.lower_bound,
                                              :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered salary adjustment records.
  def self.get_filter_by(filters, pagenum = 1,
    pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    criteria, order = get_filter_criteria(filters, sort)
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = criteria.order(order).all(:offset => pager.lower_bound,
                                      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,

```

```

:prevpage => pager.pagenum - 1,
:list => list,
:sortcolumn => sort.column,
:sortdir => sort.direction,
:page => pager.pagenum,
:totalpage => pager.total_pages }
end

# Get the validation errors.
def self.get_errors(errors)
{ :error => 1, :errors => errors }
end

# Get the item message text.
def self.item_message(filters, pagenum, pagesize)
total = 0
if filters.blank?
total = SalaryAdjustment.count
pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
return pager.item_message
else
criteria, order = get_filter_criteria(filters)
total = criteria.count
pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
return pager.item_message
end
end

# Get the salary adjustment for a staff id in the specified year.
def self.get_salary_adjustment(filters)
o = SalaryAdjustment.where(:staff_id => filters[:staff_id])
.where('year <= ?', filters[:year])
.sum('inc')
o.to_f
end

private

def self.get_filter_criteria(filters, sort = nil)
staff_id_keyword = "%#{filters[:staff_id]}%"
order = sort.present? ? sort.to_s : nil
criteria = SalaryAdjustment
if filters[:staff_id].present?
criteria = criteria.where('staff_id like ?', staff_id_keyword)
end

if filters[:month] != 0
criteria = criteria.where(:month => filters[:month])
end

if filters[:year] != 0
criteria = criteria.where(:year => filters[:year])
end

return criteria, order
end
end

```

```

E:\workspace\payroll_rails\app\helpers\user_helper.rb

module UserHelper
  DEFAULT_SORT_COLUMN = 'username'
  DEFAULT_SORT_DIR = 'ASC'

  # Get all user records.
  def self.get_all(pagenum = 1, pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    total = User.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = User.order(order).all(:offset => pager.lower_bound, :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get filtered user records.
  def self.get_filter_by(filters, pagenum = 1,
    pagesize = ApplicationHelper::Pager.default_page_size,
    sort = ApplicationHelper::Sort.new(DEFAULT_SORT_COLUMN, DEFAULT_SORT_DIR))
    criteria, order = get_filter_criteria(filters, sort)
    total = criteria.count
    pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
    order = sort.to_s

    has_next = pager.has_next? ? 1 : 0
    has_prev = pager.has_prev? ? 1 : 0
    list = criteria.order(order).all(:offset => pager.lower_bound,
      :limit => pager.pagesize)
    { :item_msg => pager.item_message,
      :hasnext => has_next,
      :hasprev => has_prev,
      :nextpage => pager.pagenum + 1,
      :prevpage => pager.pagenum - 1,
      :list => list,
      :sortcolumn => sort.column,
      :sortdir => sort.direction,
      :page => pager.pagenum,
      :totalpage => pager.total_pages }
  end

  # Get the validation errors.
  def self.get_errors(errors)
    { :error => 1, :errors => errors }
  end

  # Get the item message text.
  def self.item_message(filters, pagenum, pagesize)
    total = 0
    if filters.blank?
      total = User.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message
    else
      criteria = User.where('username like ?', "%#{keyword}%")
      total = criteria.count
      pager = ApplicationHelper::Pager.new(total, pagenum, pagesize)
      return pager.item_message
    end
  end

  private

  def self.get_filter_criteria(filters, sort = nil)

```

```

employee_keyword = "%#{filters[:employee]}%"
username_keyword = "%#{filters[:username]}%"
order = sort.present? ? sort.to_s : nil
if filters[:employee].present?
  criteria = get_join(filters)

else
  criteria = User
end

if filters[:username].present?
  criteria = criteria.where('username like ?', username_keyword)
end

if filters[:role] != 0
  criteria = criteria.where(:role => filters[:role])
end

if filters[:employee].present?
  criteria = criteria.where(
    'e.first_name like ? or e.middle_name like ? or e.last_name like ?',
    employee_keyword, employee_keyword, employee_keyword)
end

if filters[:status] != 0
  criteria = criteria.where(:status => filters[:status] == 1 ? true : false)
end

return criteria, order
end

def self.get_join(filters)
  if filters[:employee].present?
    User.joins('inner join employee e on user.id = e.user_id')
  end
end
end

```

E:\workspace\payroll_rails\app\views\layouts\admin.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
  <title>Payroll System - Admin</title>
  <%= favicon_link_tag "favicon.ico", :rel => "shortcut icon" %>
  <%= favicon_link_tag "favicon.ico", :rel => "icon", :type => "image/x-icon" %>
  <%= stylesheet_link_tag "admin", :media => "all" %>
  <%= javascript_include_tag "admin" %>
  <meta name="viewport" content="width=device-width; initial-scale=1.0" />
  <%= csrf_meta_tags %>
</head>
<body class="ui-widget-content">
  <%= yield %>
</body>
</html>
```

E:\workspace\payroll_rails\app\views\layouts\application.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <!-- Always force latest IE rendering engine (even in intranet) & Chrome Frame
      Remove this if you use the .htaccess -->
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
  <meta name="keywords" content="wfsview, payroll system, mvc, ruby on rails" />
  <meta name="description" content="This is mvc web application developed using ruby on rails,
jquery, and jquery ui" />
  <title><%= @title %></title>
  <%= favicon_link_tag "favicon.ico", :rel => "shortcut icon" %>
  <%= favicon_link_tag "favicon.ico", :rel => "icon", :type => "image/x-icon" %>
  <%= stylesheet_link_tag "loginui", :media => "all" %>
  <meta name="viewport" content="width=device-width; initial-scale=1.0" />
  <%= csrf_meta_tags %>
</head>

<body class="ui-widget-content">
  <%= yield %>
</body>
</html>
```

E:\workspace\payroll_rails\app\views\layouts\chart.html.erb

```
<div id="table_container">
  <div class="page_title ui-widget-header ui-widget"><%= @title %></div>
  <table class="page_container ui-widget">
    <tbody>
      <tr>
        <td valign="top"><%= yield :search_form %></td>
      </tr>
    </tbody>
  </table>
  <div class="page_container ui-widget">
    <div id="right_box">
      <%= yield :content_list %>
    </div>
  </div>
<%= render 'shared/alert_panel' %>
```

E:\workspace\payroll_rails\app\views\layouts\list.html.erb

```
<div id="table_container">
  <div class="page_title ui-widget-header ui-widget"><%= @title %></div>
  <table class="page_container ui-widget">
    <tbody>
      <tr>
        <td valign="top"><%= yield :search_form %></td>
        <td valign="top"><%= render 'shared/navigate_form' %></td>
      </tr>
    </tbody>
  </table>
  <div class="page_container ui-widget">
    <div id="right_box">
```

```

        <%= yield :content_list %>
    </div>
</div>
</div>
<%= render 'shared/alert_panel' %>
<%= yield :dialog %>

E:\workspace\payroll_rails\app\views\layouts\user.html.erb

<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <title>Payroll System - User</title>
    <%= favicon_link_tag "favicon.ico", :rel => "shortcut icon" %>
    <%= favicon_link_tag "favicon.ico", :rel => "icon", :type => "image/x-icon" %>
    <%= stylesheet_link_tag "user", :media => "screen, projection" %>
    <%= javascript_include_tag "user" %>
    <meta name="viewport" content="width=device-width; initial-scale=1.0" />
    <%= csrf_meta_tags %>
</head>
<body class="ui-widget-content">
    <%= yield %>
</body>
</html>

E:\workspace\payroll_rails\app\views\shared\_add_button.html.erb

<div id="id_add" class="hover ui-state-default ui-corner-all">
    <span class="icons addicon paddicon">Add</span>
</div>

E:\workspace\payroll_rails\app\views\shared\_alert_panel.html.erb

<div id="dialog-message" title="<%= @dialog_title %>">
    <div id="dialog_msg"></div>
</div>

E:\workspace\payroll_rails\app\views\shared\_cancel_button.html.erb

<div class="save_button cancel hover ui-state-default ui-corner-all">
    <span class="icons cancelicon paddicon">Cancel</span>
</div>

E:\workspace\payroll_rails\app\views\shared\_chart_button.html.erb

<span id="id_gen" class="hover ui-state-default ui-corner-all">
    <span class="icons charticon paddicon">Generate</span>
</span>

E:\workspace\payroll_rails\app\views\shared\_delete_button.html.erb

<div id="id_delete" class="hover ui-state-default ui-corner-all">
    <span class="icons deleteicon paddicon">Delete</span>
</div>

E:\workspace\payroll_rails\app\views\shared\_dialog_save.html.erb

<div id="dialog-add" title="<%= @dialog_add_title %>">
    <div id="dialog_add_body"></div>
</div>
<div id="dialog-edit" title="<%= @dialog_edit_title %>">
    <div id="dialog_edit_body"></div>
</div>

E:\workspace\payroll_rails\app\views\shared\_gen_button.html.erb

<span id="id_gen" class="hover ui-state-default ui-corner-all">
    <span class="icons detailicon paddicon">Generate</span>
</span>

E:\workspace\payroll_rails\app\views\shared\_logout.html.erb

<div id="logout" class="top right ui-corner-all ui-state-default">
    <%= link_to 'Logout', logout_path, :class => 'icons logouticon paddicon' %>
</div>

```

```

E:\workspace\payroll_rails\app\views\shared\_navigate_form.html.erb

<table class="nav_table">
  <tr>
    <td>
      Page &nbsp;
      <input id="id_pagenum" type="text" class="pg" /> of <%= @data[:totalpage] %>
      <span id="id_go" class="go_button ui-state-default ui-corner-all">
        <span>Go</span>
      </span>
    </td>
    <td>
      <span id="id_prev" class="page_button ui-state-default ui-corner-all">
        <span class="icons lefticon">&nbsp;</span>
      </span>
    </td>
    <td>
      <span id="id_next" class="page_button ui-state-default ui-corner-all">
        <span class="icons righticon">&nbsp;</span>
      </span>
    </td>
  </tr>
  <tr>
    <td></td>
    <td>
      <label for="id_display">Display:</label>
      <select id="id_display">
        <option value="20">20</option>
        <option value="50" selected="selected">50</option>
        <option value="100">100</option>
      </select>
    </td>
    <td>
      <span class="item display"></span>
    </td>
  </tr>
</table>

E:\workspace\payroll_rails\app\views\shared\_panel_button.html.erb

<table>
  <tr>
    <% if @include_search %>
    <td>
      <%= render 'shared/search_button' %>
    </td>
    <% end %>
    <td>
      <%= render 'shared/add_button' %>
    </td>
    <td>
      <%= render 'shared/delete_button' %>
    </td>
  </tr>
</table>

E:\workspace\payroll_rails\app\views\shared\_progress_panel.html.erb

<div class="ui-widget">
  <div class="ui-state-highlight ui-corner-all" style="margin-top: 20px; padding: 0 .7em;">
    <p>
      <span class="spinner" style="float: left; margin-right: .3em"></span>
      <div style="margin-left: 20px;">Loading ...</div>
    </p>
  </div>
</div>

E:\workspace\payroll_rails\app\views\shared\_save_button.html.erb

<div class="save_button save hover ui-state-default ui-corner-all">
  <span class="icons saveicon paddicon">Save</span>
</div>

```

```

E:\workspace\payroll_rails\app\views\shared\_search_button.html.erb

<span id="id_find" class="hover ui-state-default ui-corner-all">
  <span class="icons findicon paddicon">Find</span>
</span>

E:\workspace\payroll_rails\app\views\shared\_search_input.html.erb

<input id="id_query" type="text" class="search" placeholder="type here to search" title="type here to search" />
<%= render 'shared/search button' %>

E:\workspace\payroll_rails\app\views\shared\_search_panel.html.erb

<%= render 'shared/search input' %>
<%= render 'shared/panel_button' %>

E:\workspace\payroll_rails\app\views\shared\_status_panel.html.erb

<div id="status-panel" class="ui-widget">
  <div id="status-panel_outer" style="margin-top: 20px; padding: 0 .7em;">
    <p>
      <span id="status-panel_inner" style="float: left; margin-right: .3em"></span>
      <div id="status_msg" style="margin-left: 20px;"></div>
    </p>
  </div>
</div>

E:\workspace\payroll_rails\app\views\shared\_theme_select.html.erb

<div id="section_option">
  <span id="main_options" class="hover ui-corner-all ui-state-default">
    <span class="icons optionicon paddicon">Options</span>
  </span>
</div>
<div id="theme_option" class="theme_options ui-state-highlight">
  <div class="ui-widget-header aligncenter" style="height: 17px">
    <span>Select Theme</span>
    <span class="icons cancelicon hover" style="margin-left: 10px"></span>
  </div>
  <div id="theme_body" style="display: inline"></div>
</div>

E:\workspace\payroll_rails\app\views\application\new.html.erb

<% @title = 'Payroll System: Log In' %>

<div class="login_box ui-corner-all ui-state-hover">
  <div class="msg ui-state-highlight ui-corner-all">Please log in to access the application:</div>

  <%= content_tag :div, notice, :class => 'notice ui-state-hover ui-corner-all' if notice.present? %>
  <%= content_tag :div, alert, :class => 'errorblock ui-state-error ui-corner-all' if alert.present? %>

  <%= form_tag(auth_path) do %>
    <div class="row">
      <%= label_tag :username, 'Username', :class => 'editor-label' %>
      <%= text_field_tag :username, 'admin', :class => 'text-box' %>
    </div>
    <div class="row">
      <%= label_tag :password, 'Password', :class => 'editor-label' %>
      <%= password_field_tag :password, 'admin123', :class => 'text-box' %>
    </div>

    <div class="msg ui-corner-all">
      <%= submit_tag 'Log In', :class => 'hover msg login_btn ui-state-hover ui-corner-all' %>
    </div>
  <% end %>
</div>

```

E:\workspace\payroll_rails\app\views\admin\admin_menu.html.erb

```
<div id="menu">
  <h3><%= link_to 'Administration', '#' %></h3>
  <div>
    <div><%= link_to 'User', '#', :onclick => "return user.load()", :id => 'menu_user' %></div>
    <div><%= link_to 'Employee', '#', :onclick => "return emp.load()" %></div>
    <div><%= link_to 'Attendance', '#', :onclick => "return att.load()" %></div>
  </div>
  <h3><%= link_to 'Job', '#' %></h3>
  <div>
    <div><%= link_to 'Job Title', '#', :onclick => "return designation.load()" %></div>
    <div><%= link_to 'Employment Status', '#', :onclick => "return empstatus.load()" %></div>
    <div><%= link_to 'Job Category', '#', :onclick => "return jobcat.load()" %></div>
    <div><%= link_to 'Department', '#', :onclick => "return dept.load()" %></div>
  </div>
  <h3><%= link_to 'Payroll', '#' %></h3>
  <div>
    <div><%= link_to 'Pay Rate', '#', :onclick => "return payrate.load()" %></div>
    <div><%= link_to 'Overtime Pay Rate', '#', :onclick => "return overtimeerate.load()" %></div>
  </div>
  <div><%= link_to 'Salary Adjustment', '#', :onclick => "return saladj.load()" %></div>
  <div><%= link_to 'Overtime Chart', '#', :onclick => "return otcht.load()" %></div>
  <div><%= link_to 'Total Hours Worked Chart', '#', :onclick => "return wkcht.load()" %></div>
  <div><%= link_to 'Hourly Payroll Chart', '#', :onclick => "return hpcht.load()" %></div>
  <div><%= link_to 'Payslip', '#', :onclick => "return payslip.load()" %></div>
</div>
</div>
```

E:\workspace\payroll_rails\app\views\admin\admin_index.html.erb

```
<div id="maincontainer">
  <div id="topsection" class="ui-state-default">
    <div class="innertube">
      <%= render 'shared/status_panel' %>
      <%= render 'shared/logout' %>
      <%= render 'shared/theme_select' %>
    </div>
  </div>
  <div id="contentwrapper">
    <div id="contentcolumn">
      <div class="innertube"></div>
    </div>
  </div>
  <div id="leftcolumn">
    <div class="innertube">
      <%= render 'menu' %>
    </div>
  </div>
</div>
<div id="error-dialog" title="Internal Server Error">
  <div id="error_dialog"></div>
</div>
<div id="progress_status">
  <%= render 'shared/progress panel' %>
</div>
```

E:\workspace\payroll_rails\app\views\admin\attendance_list.html.erb

```
<% if @data.present? && @data[:list].any? %>
  <table class="list_table">
    <thead class="ui-widget-header">
      <tr>
        <th class="data">
          <div id="hd_work_date" class="sorthead hover">
            <span class="colheader">Date</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_e-first_name" class="sorthead hover">
            <span class="colheader">Employee Name</span>
            <span class="sorticon"></span>
          </div>
        </th>
      </tr>
    </thead>
    <tbody>
```

```
 <div id="hd_time_in" class="sorthead hover">         <span class="colheader">Time In</span>         <span class="sorticon"></span>     </div> | <div id="hd_time_out" class="sorthead hover">         <span class="colheader">Time Out</span>         <span class="sorticon"></span>     </div> |


<% @data[:list].each do |o| %>
<% e = o.employee %>
<tr id="tr_<%= o.id %>" class="<% cycle("ui-state-active", "ui-state-default") %>">
    <td class="data">
        <%= h fmt_date(o.work_date) %>
    </td>
    <td class="data">
        <%= h e.first_name %> <%= h e.middle_name %> <%= h e.last_name %>
    </td>
    <td class="data">
        <%= h fmt_time(o.time_in) %>
    </td>
    <td class="data">
        <%= h fmt_time(o.time_out) %>
    </td>

<% end %>

<% end %>
<input id="id_pg" type="hidden" value="<%=@data[:hasprev] %>,<%=@data[:hasnext] %>,<%=@data[:prevpage] %>,<%=@data[:nextpage] %>,<%=@data[:item_msg] %>,<%=@data[:sortcolumn] %>,<%=@data[:sortdir] %>,<%=@data[:page] %>" />

```

E:\workspace\payroll_rails\app\views\admin\attendance_search_form.html.erb

```


|                                                                   |                                                                                                                                                                        |
|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <%= label_tag :id_employee, 'Employee', :class => 'label' %>      | <%= text_field_tag :employee, nil, :id => 'id_employee', :class => 'search',         :placeholder => 'type here to search',         :title => 'type here to search' %> |
| <%= label_tag :id_work_date, 'Working Date', :class => 'label' %> | <%= text_field_tag :work_date, nil, :id => 'id_work_date', :class => 'search',         :date_input', :placeholder => 'dd-mm-YYYY' %>                                   |


<br />
<%= render 'shared/search_button' %>

```

E:\workspace\payroll_rails\app\views\admin\attendance\index.html.erb

```

<% @title = 'Attendance Record' %>
<% @dialog_title = 'Attendance Record' %>
<% content_for :search_form do %>
    <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
    <%= render 'list' %>
<% end %>

```

E:\workspace\payroll_rails\app\views\admin\department_form.html.erb

```
<%= form_tag '.', :id => @form_id, :class => 'save_form ui-widget-content ui-corner-all' do %>
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :name, 'Name', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :name, @dept.name, :id => 'id_name', :class => 'text', :placeholder => 'enter name',
                           :title => 'enter name' %>
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <div class="row_button">
          <%= render 'shared/save_button' %>
          <%= render 'shared/cancel_button' %>
        </div>
      </td>
    </tr>
  </table>
<% end %>
```

E:\workspace\payroll_rails\app\views\admin\department_list.html.erb

```
<% if @data.present? && @data[:list].any? %>
  <table class="list_table">
    <thead class="ui-widget-header">
      <tr>
        <th class="data_chkbox"><input class="hdchk" type="checkbox" /></th>
        <th class="data">
          <div id="hd_name" class="sorthead hover">
            <span class="colheader">Department</span>
            <span class="sorticon"></span>
          </div>
        </th>
      </tr>
    </thead>
    <tbody>
      <% @data[:list].each do |o| %>
        <tr id="tr_<%= o.id %>" class="<%= cycle("ui-state-active", "ui-state-default") %>">
          <td class="data_chkbox"><input class="chk" type="checkbox" /></td>
          <td class="data"><%= h o.name %></td>
        </tr>
      <% end %>
    </tbody>
  </table>
<% end %>
<input id="id_pg" type="hidden" value="<%= @data[:hasprev] %>,<%= @data[:hasnext] %>,<%= @data[:prevpage] %>,<%= @data[:nextpage] %>,<%= @data[:item_msg] %>,<%= @data[:sortcolumn] %>,<%= @data[:sortdir] %>,<%= @data[:page] %>" />
```

E:\workspace\payroll_rails\app\views\admin\department_search_form.html.erb

```
<%= label_tag :id_query, 'Search For', :class => 'label' %>
<%= render 'shared/search_panel' %>
```

E:\workspace\payroll_rails\app\views\admin\department\index.html.erb

```
<% @title = 'Department' %>
<% @dialog_title = 'Department' %>
<% @dialog_add_title = 'Add New Department' %>
<% @dialog_edit_title = 'Edit Department' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'list' %>
<% end %>
<% content_for :dialog do %>
  <%= render 'shared/dialog_save' %>
<% end %>
```

E:\workspace\payroll_rails\app\views\admin\designation_form.html.erb

```
<%= form_tag '.', :id => @form_id, :class => 'save_form ui-widget-content ui-corner-all' do %>
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :title, 'Job Title', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :title, @designation.title, :id => 'id_title', :class => 'text', :placeholder => 'enter job title', :title => 'enter job title' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :desc, 'Job Description', :class => 'label' %>
      </td>
      <td>
        <%= text_area_tag :desc, @designation.desc, :id => 'id_desc', :class => 'textarea', :placeholder => 'enter job description', :title => 'enter job description' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :note, 'Note', :class => 'label' %>
      </td>
      <td>
        <%= text_area_tag :note, @designation.note, :id => 'id_note', :class => 'textarea', :placeholder => 'enter note', :title => 'enter note' %>
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <div class="row_button">
          <%= render 'shared/save_button' %>
          <%= render 'shared/cancel_button' %>
        </div>
      </td>
    </tr>
  </table>
<% end %>
```

E:\workspace\payroll_rails\app\views\admin\designation_list.html.erb

```
<% if @data.present? && @data[:list].any? %>
  <table class="list_table">
    <thead class="ui-widget-header">
      <tr>
        <th class="data chkbox"><input class="hdchk" type="checkbox" /></th>
        <th class="data">
          <div id="hd_title" class="sorthead hover">
            <span class="colheader">Job Title</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_desc" class="sorthead hover">
            <span class="colheader">Job Description</span>
            <span class="sorticon"></span>
          </div>
        </th>
      </tr>
    </thead>
    <tbody>
      <% @data[:list].each do |d| %>
        <tr id="tr <%= d.id %>" class="<%= cycle("ui-state-active", "ui-state-default") %>">
          <td class="data chkbox"><input class="chk" type="checkbox" /></td>
          <td class="data"><%= h d.title %></td>
          <td class="data"><%= h d.desc %></td>
        </tr>
      <% end %>
    </tbody>
```

```

        </table>
<% end %>
<input id="id_pg" type="hidden" value="<%=@data[:hasprev] %>,<%=@data[:hasnext] %>,<%=@data[:prevpage] %>,<%=@data[:nextpage] %>,<%=@data[:item_msg] %>,<%=@data[:sortcolumn] %>,<%=@data[:sortdir] %>,<%=@data[:page] %>" />
```

E:\workspace\payroll_rails\app\views\admin\designation_search_form.html.erb

```

<%= label_tag :id_selection, 'Search By', :class => 'label' %>
<select id="id_selection" class="search_option">
  <%= options_for select({ 'All' => 0, 'Job Title' => 1, 'Job Description' => 2, 'Note' => 3 }) %>
</select>
<%= render 'shared/search_panel' %>
```

E:\workspace\payroll_rails\app\views\admin\designation\index.html.erb

```

<% @title = 'Job Title' %>
<% @dialog_title = 'Job Title' %>
<% @dialog_add_title = 'Add New Job Title' %>
<% @dialog_edit_title = 'Edit Job Title' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'list' %>
<% end %>
<% content_for :dialog do %>
  <%= render 'shared/dialog_save' %>
<% end %>
```

E:\workspace\payroll_rails\app\views\admin\employee_form.html.erb

```

<%= form_tag '.', :id => @form_id, :class => 'save_form ui-widget-content ui-widget ui-corner-all' do %>
  <div id="tabs">
    <ul>
      <li>
        <a href="#tabs-1">Personal Details</a>
      </li>
      <li>
        <a href="#tabs-2">Contact Details</a>
      </li>
      <li>
        <a href="#tabs-3">Job</a>
      </li>
      <li>
        <a href="#tabs-4">Salary</a>
      </li>
      <li>
        <a href="#tabs-5">Qualification</a>
      </li>
    </ul>
    <div id="tabs-1">
      <%= render 'form_employee' %>
    </div>
    <div id="tabs-2">
      <%= render 'form_employee_contact' %>
    </div>
    <div id="tabs-3">
      <%= render 'form_employee_job' %>
    </div>
    <div id="tabs-4">
      <%= render 'form_employee_salary' %>
    </div>
    <div id="tabs-5">
      <%= render 'form_employee_qualification' %>
    </div>
  </div>
  <div class="row_button">
    <%= render 'shared/save_button' %>
    <%= render 'shared/cancel_button' %>
  </div>
<% end %>
```

E:\workspace\payroll_rails\app\views\admin\employee_form_employee.html.erb

```
<div id="form-employee">
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :full_name, 'Full Name', :class => 'label' %>
      </td>
      <td valign="top">
        <%= text_field_tag :first_name, @employee.first_name, :id => 'id_first_name', :class => 'text', :placeholder => 'enter first name',
                           :title => 'enter first name' %>
      </td>
      <td valign="top">
        <%= text_field_tag :middle_name, @employee.middle_name, :id => 'id_middle_name',
                           :class => 'text', :placeholder => 'enter middle name',
                           :title => 'enter middle name' %>
      </td>
      <td valign="top">
        <%= text_field_tag :last_name, @employee.last_name, :id => 'id_last_name', :class => 'text',
                           :placeholder => 'enter last name',
                           :title => 'enter last name' %>
      </td>
    </tr>
    <tr>
      <td></td>
      <td align="center">First Name</td>
      <td align="center">Middle Name</td>
      <td align="center">Last Name</td>
    </tr>
  </table>
  <hr />
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :staff_id, 'Employee Id', :class => 'label' %>
      </td>
      <td class="colspacing">
        <%= text_field_tag :staff_id, @employee.staff_id, :id => 'id_staff_id', :class => 'text',
                           :placeholder => 'enter employee id',
                           :title => 'enter employee id' %>
      </td>
      <td valign="top" class="lbl">
        <%= label_tag :new_ic, 'New I.C No.', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :new_ic, @employee.new_ic, :id => 'id_new_ic', :class => 'text',
                           :placeholder => 'enter new I.C no.',
                           :title => 'enter new I.C no.' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :passport_no, 'Passport No', :class => 'label' %>
      </td>
      <td class="colspacing">
        <%= text_field_tag :passport_no, @employee.passport_no, :id => 'Passport No.', :class => 'text',
                           :placeholder => 'enter passport no.',
                           :title => 'enter passport no.' %>
      </td>
      <td valign="top" class="lbl">
        <%= label_tag :old_ic, 'Old I.C No.', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :old_ic, @employee.old_ic, :id => 'id_old_ic', :class => 'text',
                           :placeholder => 'enter old I.C no.',
                           :title => 'enter old I.C no.' %>
      </td>
    </tr>
    <tr>
      <td colspan="4">
        <hr />
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :gender, 'Gender', :class => 'label' %>
      </td>
    </tr>
```

```

</td>
<td valign="top">
  <%= radio_button_tag :gender, 'M', @employee.gender == 'M' ? true : false %>
  <%= label_tag :gender_m, 'Male' %>
  <%= radio_button_tag :gender, 'F', @employee.gender == 'F' ? true : false %>
  <%= label_tag :gender_f, 'Female', :id => 'lb_gender_f' %> </td>
<td valign="top" class="lbl">
  <%= label_tag :marital_status, 'Marital Status', :class => 'label' %>
</td>
<td>
  <%= select_tag :marital_status,
    options_for_select([['-- Select --', '0'], ['Single', 'S'], ['Married', 'M'],
    ['Other', 'O']], @employee.marital_status.blank? ? '0' : @employee.marital_status),
    :id => 'id_marital_status', :class => 'select' %>
</td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :nationality, 'Nationality', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :nationality, @employee.nationality, :id => 'id_nationality',
    :class => 'text', :placeholder => 'enter nationality',
      :title => 'enter nationality' %>
  </td>
  <td colspan="2"></td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :dob, 'Date of Birth', :class => 'label' %>
  </td>
  <td class="colspacing">
    <%= text_field_tag :dob, fmt_date(@employee.dob), :id => 'id_dob', :class => 'text',
    date_input, :placeholder => 'dd-mm-YYYY' %>
  </td>
  <td valign="top" class="lbl">
    <%= label_tag :place_of_birth, 'Place of Birth', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :place_of_birth, @employee.place_of_birth, :id =>
    'id_place_of_birth', :class => 'text', :placeholder => 'enter place of birth',
      :title => 'enter place of birth' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :race, 'Race', :class => 'label' %>
  </td>
  <td class="colspacing">
    <%= text_field_tag :race, @employee.race, :id => 'id_race', :class => 'text',
    :placeholder => 'enter race',
      :title => 'enter race' %>
  </td>
  <td>
    <%= label_tag :is_bumi, 'Is Bumiputra', :class => 'label' %>
  </td>
  <td>
    <%= check_box_tag :is_bumi, nil, @employee.is_bumi, :id => 'id_is_bumi' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :religion, 'Religion', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :religion, @employee.religion, :id => 'id_religion', :class =>
    'text', :placeholder => 'enter religion',
      :title => 'enter religion' %>
  </td>
  <td colspan="2"></td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :user_id, 'User ID', :class => 'label' %>
  </td>
  <td>
    <select id="id_user_id" class="select">

```

```

        <option value="0">-- Select --</option>
        <%= options_from_collection_for_select(@users, :id, :username,
@employee.user_id.blank? ? '0' : @employee.user_id) %>
    </select>
</td>
<td colspan="2"></td>
</tr>
</table>
</div>

E:\workspace\payroll_rails\app\views\admin\employee\_form_employee_contact.html.erb

<div id="form-employee-contact">
    <table>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :address_1, 'Address 1', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :address_1, @employee_contact.address_1, :id => 'id_address_1',
:class => 'text', :placeholder => 'enter street',
                                :title => 'enter street' %>
            </td>
        </tr>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :address_2, 'Address 2', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :address_2, @employee_contact.address_2, :id => 'id_address_2',
:class => 'text', :placeholder => 'enter street',
                                :title => 'enter street' %>
            </td>
        </tr>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :address_3, 'Address 3', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :address_3, @employee_contact.address_3, :id => 'id_address_3',
:class => 'text', :placeholder => 'enter street',
                                :title => 'enter street' %>
            </td>
        </tr>
        <tr>
            <td valign="top" class="lbl"> <%= label_tag :city, 'City', :class => 'label' %> </td>
            <td> <%= text_field_tag :city, @employee_contact.city, :id => 'id_city', :class =>
'text', :placeholder => 'enter city',
                                :title => 'enter city' %> </td>
        </tr>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :state, 'State', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :state, @employee_contact.state, :id => 'id_state', :class =>
'text', :placeholder => 'enter state',
                                :title => 'enter state' %>
            </td>
        </tr>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :postcode, 'Postal Code', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :postcode, @employee_contact.postcode, :id => 'id_postcode', :class =>
'text', :placeholder => 'enter postal code',
                                :title => 'enter postal code' %>
            </td>
        </tr>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :country, 'Country', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :country, @employee_contact.country, :id => 'id_country', :class =>
'text', :placeholder => 'enter country',

```

```

        :title => 'enter country' %>
    </td>
</tr>
<tr>
    <td colspan="2">
        <hr />
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :home_phone, 'Home Telephone', :class => 'label' %>
    </td>
    <td> <%= text_field_tag :home_phone, @employee_contact.home_phone, :id =>
'id_home_phone', :class => 'text', :placeholder => 'enter home phone',
:title => 'enter home phone' %>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :mobile_phone, 'Mobile', :class => 'label' %>
    </td>
    <td>
        <%= text_field_tag :mobile_phone, @employee_contact.mobile_phone, :id =>
'id_mobile_phone', :class => 'text', :placeholder => 'enter mobile phone',
:title => 'enter mobile phone' %>
    </td>
</tr>
<tr>
    <td colspan="2">
        <hr />
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :work_email, 'Work Email', :class => 'label' %>
    </td>
    <td>
        <%= text_field_tag :work_email, @employee_contact.work_email, :id => 'id_work_email',
:class => 'text', :placeholder => 'enter work email',
:title => 'enter work email' %>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :other_email, 'Other Email', :class => 'label' %>
    </td>
    <td>
        <%= text_field_tag :other_email, @employee_contact.other_email, :id =>
'id_other_email', :class => 'text', :placeholder => 'enter other email',
:title => 'enter other email' %>
    </td>
</tr>
</table>
</div>

```

E:\workspace\payroll_rails\app\views\admin\employee_form_employee_job.html.erb

```

<div id="form-employee-job">
    <table>
        <tr>
            <td valign="top" class="lbl">
                <%= label :designation_id, 'Job Title', :class => 'label' %>
            </td>
            <td>
                <select id="id_designation_id" class="select">
                    <option value="0">-- Select --</option>
                    <%= options_from_collection_for_select(@designations, :id, :title,
@employee_job.designation_id.blank? ? '0' : @employee_job.designation_id) %>
                </select>
            </td>
        </tr>
        <tr>
            <td valign="top" class="lbl">
                <%= label :employment_status_id, 'Employment Status', :class => 'label' %>
            </td>
            <td>
                <select id="id_employment_status_id" class="select">

```

```

        <option value="0">-- Select --</option>
        <%= options_from_collection_for_select(@employment_statuses, :id, :name,
@employee.job.employment_status_id.blank? ? '0' : @employee.job.employment_status_id) %>
        </select>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label :job_category_id, 'Job Category', :class => 'label' %>
    </td>
    <td>
        <select id="id_job_category_id" class="select">
            <option value="0">-- Select --</option>
            <%= options_from_collection_for_select(@job_categories, :id, :name,
@employee.job.job_category_id.blank? ? '0' : @employee.job.job_category_id) %>
        </select>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label :join_date, 'Join Date', :class => 'label' %>
    </td>
    <td>
        <%= text_field_tag :join_date, fmt_date(@employee_job.join_date), :id =>
'id_join_date', :class => 'text_date_input', :placeholder => 'dd-mm-YYYY' %>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label :confirm_date, 'Confirm Date', :class => 'label' %>
    </td>
    <td>
        <%= text_field_tag :confirm_date, fmt_date(@employee_job.confirm_date), :id =>
'id_confirm_date', :class => 'text_date_input', :placeholder => 'dd-mm-YYYY' %>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label :department_id, 'Department', :class => 'label' %>
    </td>
    <td>
        <select id="id_department_id" class="select">
            <option value="0">-- Select --</option>
            <%= options_from_collection_for_select(@departments, :id, :name,
@employee.job.department_id.blank? ? '0' : @employee.job.department_id) %>
        </select>
    </td>
</tr>
</table>
</div>

```

```

E:\workspace\payroll_rails\app\views\admin\employee\_form_employee_qualification.html.erb

<div id="form-employee-qualification">
    <table>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :level, 'Level', :class => 'Label' %>
            </td>
            <td>
                <select id="id_level" class="select">
                    <%= options_for_select([['-- Select --', 0],
                        ['PhD', 1], ['Masters Degree', 2], ['Bachelor Degree', 3], ['Diploma', 4],
                        ['STPM', 5], ['SPM', 6],
                        ['Others', 7]],
                        @employee_qualification.level.blank? ? 0 : @employee_qualification.level) %>
                </select>
            </td>
        </tr>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :institute, 'Institute', :class => 'Label' %>
            </td>
            <td>
                <%= text_field_tag :institute, @employee_qualification.institute, :id =>
'id_institute', :class => 'text', :placeholder => 'enter institute',
:title => 'enter institute' %>
            </td>
        </tr>
    </table>
</div>

```

```

        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :major, 'Major', :class => 'Label' %>
        </td>
        <td>
            <%= text_field_tag :major, @employee_qualification.major, :id => 'id_major', :class =>
'text', :placeholder => 'enter major',
                :title => 'enter major' %>
        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :year, 'Year', :class => 'Label' %>
        </td>
        <td>
            <%= text_field_tag :year, @employee_qualification.year, :id => 'id_year', :class =>
'text', :placeholder => 'enter year',
                :title => 'enter year' %>
        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :gpa, 'GPA/Grade', :class => 'Label' %>
        </td>
        <td>
            <%= text_field_tag :gpa, @employee_qualification.gpa, :id => 'id_gpa', :class =>
'text', :placeholder => 'enter gpa/grade',
                :title => 'enter gpa/grade' %>
        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :start_date, 'Start Date', :class => 'Label' %>
        </td>
        <td>
            <%= text_field_tag :start_date, fmt_date(@employee_qualification.start_date), :id =>
'id_start_date', :class => 'text_date_input', :placeholder => 'dd-mm-YYYY' %>
        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :end_date, 'End Date', :class => 'Label' %>
        </td>
        <td>
            <%= text_field_tag :end_date, fmt_date(@employee_qualification.end_date), :id =>
'id_end_date', :class => 'text_date_input', :placeholder => 'dd-mm-YYYY' %>
        </td>
    </tr>
</table>
</div>

```

E:\workspace\payroll_rails\app\views\admin\employee_form_employee_salary.html.erb

```

<div id="form-employee-salary">
    <table>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :salary, 'Salary', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :salary, @employee_salary.salary, :id => 'id_salary', :class =>
'text', :placeholder => 'enter salary',
                    :title => 'enter salary' %>
            </td>
        </tr>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :allowance, 'Allowance', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :allowance, @employee_salary.allowance, :id => 'id_allowance',
:class => 'text', :placeholder => 'enter allowance',
                    :title => 'enter allowance' %>
            </td>
        </tr>
    </table>
</div>

```

```

<tr>
  <td valign="top" class="lbl">
    <%= label_tag :epf, 'EPF Deduction', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :epf, @employee_salary.epf, :id => 'id_epf', :class => 'text',
:placeholder => 'enter EPF deduction',
                      :title => 'enter EPF deduction' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :socso, 'SOCSCO Deduction', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :socso, @employee_salary.socso, :id => 'id_socso', :class =>
'text', :placeholder => 'enter SOCSCO deduction',
                      :title => 'enter SOCSCO deduction' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :income_tax, 'Income Tax Deduction', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :income_tax, @employee_salary.income_tax, :id => 'id_income_tax',
:class => 'text', :placeholder => 'enter Income Tax deduction',
                      :title => 'enter Income Tax deduction' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :bank_name, 'Bank Name', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :bank_name, @employee_salary.bank_name, :id => 'id_bank_name',
:class => 'text', :placeholder => 'enter bank name',
                      :title => 'enter bank name' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :bank_acc_no, 'Bank Account No.', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :bank_acc_no, @employee_salary.bank_acc_no, :id =>
'id_bank_acc_no', :class => 'text', :placeholder => 'enter bank account no.',
                      :title => 'enter bank account no.' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :bank_acc_type, 'Bank Account Type', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :bank_acc_type, @employee_salary.bank_acc_type, :id =>
'id bank acc type', :class => 'text', :placeholder => 'enter bank account type',
                      :title => 'enter bank account type' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :bank_address, 'Bank Address', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :bank_address, @employee_salary.bank_address, :id =>
'id_bank_address', :class => 'text', :placeholder => 'enter bank address',
                      :title => 'enter bank address' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :epf_no, 'EPF no.', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :epf_no, @employee_salary.epf_no, :id => 'id_epf_no', :class =>
'text', :placeholder => 'enter EPF no.',
```

```

        :title => 'enter EPF no.' %>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :socso_no, 'SOC SO no.', :class => 'label' %>
    </td>
    <td>
        <%= text_field_tag :socso_no, @employee_salary.socso_no, :id => 'id_socso_no', :class => 'text', :placeholder => 'enter SOC SO no.', :title => 'enter SOC SO no.' %>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :income_tax_no, 'Income Tax no.', :class => 'label' %>
    </td>
    <td>
        <%= text_field_tag :income_tax_no, @employee_salary.income_tax_no, :id => 'id_income_tax_no', :class => 'text', :placeholder => 'enter income tax no.', :title => 'enter income tax no.' %>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :pay_type, 'Pay Type', :class => 'label' %>
    </td>
    <td>
        <select id="id_pay_type" class="select">
            <option value="0">-- Select --</option>
            <%= options_for_select([['-- Select --', 0], ['Monthly', 1], ['Hourly', 2]], @employee_salary.pay_type.blank? ? 0 : @employee_salary.pay_type) %>
        </select>
    </td>
</tr>
</table>
</div>

```

E:\workspace\payroll_rails\app\views\admin\employee_list.html.erb

```

<% if @data.present? && @data[:list].any? %>
<table class="list_table">
    <thead class="ui-widget-header">
        <tr>
            <th class="data chkbox"><input class="hdchk" type="checkbox" /></th>
            <th class="data">
                <div id="hd_staff_id" class="sorthead hover">
                    <span class="colheader">Id</span>
                    <span class="sorticon"></span>
                </div>
            </th>
            <th class="data">
                <div id="hd_first_name" class="sorthead hover">
                    <span class="colheader">First (& Middle) Name</span>
                    <span class="sorticon"></span>
                </div>
            </th>
            <th class="data">
                <div id="hd_last_name" class="sorthead hover">
                    <span class="colheader">Last Name</span>
                    <span class="sorticon"></span>
                </div>
            </th>
            <th class="data">
                <div id="hd_d-title" class="sorthead hover">
                    <span class="colheader">Job Title</span>
                    <span class="sorticon"></span>
                </div>
            </th>
            <th class="data">
                <div id="hd_es-name" class="sorthead hover">
                    <span class="colheader">Employment Status</span>
                    <span class="sorticon"></span>
                </div>
            </th>
            <th class="data">
                <div id="hd_dept-name" class="sorthead hover">

```

```

        <span class="colheader">Department</span>
        <span class="sorticon"></span>
    </div>
</th>
</tr>
</thead>
<tbody>
<% @data[:list].each do |o| %>
<tr id="tr_<%= o.id %>" class="<%= cycle("ui-state-active", "ui-state-default") %>">
    <% ej = o.employee_job %>
    <td class="data chkbox"><input class="chk" type="checkbox" /></td>
    <td class="data"><%= h o.staff_id %></td>
    <td class="data"><%= h o.first_name %></td>
    <td class="data"><%= h o.last_name %></td>
    <td class="data">
        <% if ej.present? %>
            <%= h ej.designation.title if ej.designation.present? %>
        <% end %>
    </td>
    <td class="data">
        <% if ej.present? %>
            <%= h ej.employment_status.name if ej.employment_status.present? %>
        <% end %>
    </td>
    <td class="data">
        <% if ej.present? %>
            <%= h ej.department.name if ej.department.present? %>
        <% end %>
    </td>
</tr>
<% end %>
</tbody>
</table>
<% end %>
<input id="id_pg" type="hidden" value="<%=@data[:hasprev] %>,<%=@data[:hasnext] %>,<%=@data[:prevpage] %>,<%=@data[:nextpage] %>,<%=@data[:item_msg] %>,<%=@data[:sortcolumn] %>,<%=@data[:sortdir] %>,<%=@data[:page] %>" />

```

E:\workspace\payroll_rails\app\views\admin\employee_search_form.html.erb

```


|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <%= label_tag :id_employee, 'Employee', :class => 'label' %>     </td>     <td class="colspacing">         <%= text_field_tag :employee, nil, :id => 'id_employee', :class => 'search',         :placeholder => 'type here to search',                     :title => 'type here to search' %>     </td>     <td>         <%= label_tag :id_staff_id, 'Id', :class => 'label' %>     </td>     <td>         <%= text_field_tag :staff_id, nil, :id => 'id_staff_id', :class => 'search',         :placeholder => 'type here to search',                     :title => 'type here to search' %>     </td> </tr> <tr> <td>         &lt;%= label_tag :id_employment_status, 'Employment Status', :class =&gt; 'label' %&gt;     &lt;/td&gt;     &lt;td class="colspacing"&gt;         &lt;select id="id_employment_status" class="select"&gt;             &lt;option value="0"&gt;All&lt;/option&gt;             &lt;%= options_from_collection_for_select(@employmentstatus, :id, :name) %&gt;         &lt;/select&gt;     &lt;/td&gt;     &lt;td&gt;         &lt;%= label_tag :id_designation, 'Job Title', :class =&gt; 'label' %&gt;     &lt;/td&gt;     &lt;td class="colspacing"&gt;         &lt;select id="id_designation" class="select"&gt;             &lt;option value="0"&gt;All&lt;/option&gt;             &lt;%= options_from_collection_for_select(@designation, :id, :title) %&gt;         &lt;/select&gt;     &lt;/td&gt; &lt;/tr&gt; </td> | <%= label_tag :id_employment_status, 'Employment Status', :class => 'label' %>     </td>     <td class="colspacing">         <select id="id_employment_status" class="select">             <option value="0">All</option>             <%= options_from_collection_for_select(@employmentstatus, :id, :name) %>         </select>     </td>     <td>         <%= label_tag :id_designation, 'Job Title', :class => 'label' %>     </td>     <td class="colspacing">         <select id="id_designation" class="select">             <option value="0">All</option>             <%= options_from_collection_for_select(@designation, :id, :title) %>         </select>     </td> </tr> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|


```

```

        </td>
    </tr>
    <tr>
        <td>
            <%= label_tag :id_dept, 'Department', :label => 'label' %>
        </td>
        <td>
            <select id="id_dept" class="select">
                <option value="0">All</option>
                <%= options_from_collection_for_select(@dept, :id, :name) %>
            </select>
        </td>
        <td colspan="2"></td>
    </tr>
</table>
<br />
<% @include_search = true %>
<%= render 'shared/panel_button' %>

```

E:\workspace\payroll_rails\app\views\admin\employee\index.html.erb

```

<% @title = 'Employee' %>
<% @dialog_title = 'Employee' %>
<% @dialog_add_title = 'Add New Employee' %>
<% @dialog_edit_title = 'Edit Employee' %>
<% content_for :search_form do %>
    <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
    <%= render 'list' %>
<% end %>
<% content_for :dialog do %>
    <%= render 'shared/dialog_save' %>
<% end %>

```

E:\workspace\payroll_rails\app\views\admin\employment_status_form.html.erb

```

<%= form_tag '.', :id => @form_id, :class => 'save_form ui-widget-content ui-corner-all' do %>
    <table>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :name, 'Name', :class => 'label' %>
            </td>
            <td>
                <%= text_field_tag :name, @empstatus.name, :id => 'id_name', :class => 'text',
:placeholder => 'enter name',
:title => 'enter name' %>
            </td>
        </tr>
        <tr>
            <td></td>
            <td>
                <div class="row_button">
                    <%= render 'shared/save_button' %>
                    <%= render 'shared/cancel_button' %>
                </div>
            </td>
        </tr>
    </table>
<% end %>

```

E:\workspace\payroll_rails\app\views\admin\employment_status_list.html.erb

```

<% if @data.present? && @data[:list].any? %>
    <table class="list_table">
        <thead class="ui-widget-header">
            <tr>
                <th class="data chkbox"><input class="hdchk" type="checkbox" /></th>
                <th class="data">
                    <div id="hd_name" class="sorthead hover">
                        <span class="colheader">Employment Status</span>
                        <span class="sorticon"></span>
                    </div>
                </th>
            </tr>
        </thead>

```

```

<tbody>
<% @data[:list].each do |o| %>
  <tr id="<%= o.id %>" class="<%= cycle("ui-state-active", "ui-state-default") %>">
    <td class="data chkbox"><input class="chk" type="checkbox" /></td>
    <td class="data"><%= h o.name %></td>
  </tr>
<% end %>
</tbody>
</table>
<% end %>
<input id="id_pg" type="hidden" value="<%= @data[:hasprev] %>,<%= @data[:hasnext] %>,<%= @data[:prevpage] %>,<%= @data[:nextpage] %>,<%= @data[:item_msg] %>,<%= @data[:sortcolumn] %>,<%= @data[:sortdir] %>,<%= @data[:page] %>" />

```

E:\workspace\payroll_rails\app\views\admin\employment_status_search_form.html.erb

```

<%= label_tag :id_query, 'Search For', :class => 'label' %>
<%= render 'shared/search_panel' %>

```

E:\workspace\payroll_rails\app\views\admin\employment_status\index.html.erb

```

<% @title = 'Employment Status' %>
<% @dialog title = 'Employment Status' %>
<% @dialog_add_title = 'Add New Employment Status' %>
<% @dialog_edit_title = 'Edit Employment Status' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'list' %>
<% end %>
<% content_for :dialog do %>
  <%= render 'shared/dialog_save' %>
<% end %>

```

E:\workspace\payroll_rails\app\views\admin\hourly_payroll_chart_chart.html.erb

```

<div id="cht1" style="height: 400px"></div>
<div id="cht2" style="height: 400px"></div>

```

E:\workspace\payroll_rails\app\views\admin\hourly_payroll_chart_search_form.html.erb

```

<table>
  <tr>
    <td>
      <%= label_tag :id_staff_id, 'Id', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= text_field_tag :id_staff_id, nil, :id => 'id_staff_id', :class => 'search',
      :placeholder => 'type here to search',
      :title => 'type here to search' %>
    </td>
    <td>
      <%= label_tag :id_year, 'Year', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= select_year(Date.today, { :start_year => Time.now.year, :end_year => end_year }, {
        :id => 'id_year', :class => 'select' }) %>
    </td>
    <td></td>
  </tr>
  <tr>
    <td>
      <%= label_tag :id_month, 'Month', :class => 'label' %>
    </td>
    <td colspan="4">
      <label>All</label>
      <input type="checkbox" name="month" value="0" class="chkall" />
      <% for k, v in @month_hash %>
        <label><%= k[0..2] %></label>
        <input type="checkbox" name="month" value="<%= v %>" class="chkmonth" />
      <% end %>
    </td>
  </tr>
</table>
<br />
<%= render 'shared/chart_button' %>

```

```
E:\workspace\payroll_rails\app\views\admin\hourly_payroll_chart\index.html.erb
```

```
<% @title = 'Employee Hourly Payroll Chart' %>
<% @dialog_title = 'Employee Hourly Payroll Chart' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'chart' %>
<% end %>
```

```
E:\workspace\payroll_rails\app\views\admin\job_category\_form.html.erb
```

```
<%= form_tag '.', :id => @form_id, :class => 'save_form ui-widget-content ui-corner-all' do %>
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :name, 'Name', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :name, @jobcat.name, :id => 'id_name', :class => 'text',
        :placeholder => 'enter name',
        :title => 'enter name' %>
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <div class="row_button">
          <%= render 'shared/save_button' %>
          <%= render 'shared/cancel button' %>
        </div>
      </td>
    </tr>
  </table>
<% end %>
```

```
E:\workspace\payroll_rails\app\views\admin\job_category\_list.html.erb
```

```
<% if @data.present? && @data[:list].any? %>
  <table class="list_table">
    <thead class="ui-widget-header">
      <tr>
        <th class="data chkbox"><input class="hdchk" type="checkbox" /></th>
        <th class="data">
          <div id="hd_name" class="sorthead hover">
            <span class="colheader">Job Category</span>
            <span class="sorticon"></span>
          </div>
        </th>
      </tr>
    </thead>
    <tbody>
      <% @data[:list].each do |o| %>
        <tr id="tr_<%= o.id %>" class="<<%= cycle("ui-state-active", "ui-state-default") %>">">
          <td class="data chkbox"><input class="chk" type="checkbox" /></td>
          <td class="data"><%= h o.name %></td>
        </tr>
      <% end %>
    </tbody>
  </table>
<% end %>
<input id="id_pg" type="hidden" value="<<%= @data[:hasprev] %>,<%= @data[:hasnext] %>,<%= @data[:prevpage] %>,<%= @data[:nextpage] %>,<%= @data[:item_msg] %>,<%= @data[:sortcolumn] %>,<%= @data[:sortdir] %>,<%= @data[:page] %>" />
```

```
E:\workspace\payroll_rails\app\views\admin\job_category\index.html.erb
```

```
<% @title = 'Job Category' %>
<% @dialog_title = 'Job Category' %>
<% @dialog_add_title = 'Add New Job Category' %>
<% @dialog_edit_title = 'Edit Job Category' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
```

```

<%= render 'list' %>
<% end %>
<% content_for :dialog do %>
  <%= render 'shared/dialog_save' %>
<% end %>

E:\workspace\payroll_rails\app\views\admin\overtime_chart\_chart.html.erb

<div id="cht1" style="height: 400px"></div>

E:\workspace\payroll_rails\app\views\admin\overtime chart\ search form.html.erb

|                                                        |                                                                                                                                                                                                                                                           |  |  |  |
|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|
| <%= label_tag :id_staff_id, 'Id', :class => 'label' %> | <%= text_field_tag :id_staff_id, nil, :id => 'id_staff_id', :class => 'search',     :placeholder => 'type here to search',     :title => 'type here to search' %>                                                                                         |  |  |  |
| <%= label_tag :id_year, 'Year', :class => 'label' %>   | <%= select_year(Date.today, { :start_year => Time.now.year, :end_year => end_year }, {       :id => 'id_year', :class => 'select' }) %>                                                                                                                   |  |  |  |
| <%= label_tag :id_month, 'Month', :class => 'label' %> | <label>All</label>     <input type="checkbox" name="month" value="0" class="chkall" />     <% for k, v in @month_hash %>       <label><%= k[0..2] %></label>       <input type="checkbox" name="month" value="<%= v %>" class="chkmonth" />     <% end %> |  |  |  |


<br />
<%= render 'shared/chart_button' %>

E:\workspace\payroll_rails\app\views\admin\overtime_chart\index.html.erb

<% @title = 'Employee Overtime Chart' %>
<% @dialog_title = 'Employee Overtime Chart' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'chart' %>
<% end %>

E:\workspace\payroll_rails\app\views\admin\overtime_rate\_form.html.erb

<%= form_tag '.', :id => @form_id, :class => 'save_form ui-widget-content ui-corner-all' do %>
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :duration, 'Duration', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :duration, @rate.duration, :id => 'id_duration', :class => 'text',
        :placeholder => 'enter duration',
        :title => 'enter duration' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :year, 'Year', :class => 'label' %>
      </td>
      <td>
        <%= select_year(@rate.year, { :start_year => Time.now.year, :end_year => end_year }, {
          :id => 'id_year', :class => 'select' }) %>
      </td>
    </tr>
  </table>
<%= render 'shared/form_buttons' %>

```

```

</td>
<td>
  <%= select year(@rate.year, { :start year => Time.now.year, :end year => end year,
:field_name => 'year' }, { :id => 'id_year', :class => 'select' }) %>
</td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :pay_rate, 'Pay rate', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :pay_rate, @rate.pay_rate, :id => 'id_pay_rate', :class => 'text',
:placeholder => 'enter overtime pay rate',
              :title => 'enter overtime pay rate' %>
  </td>
</tr>
<tr>
  <td></td>
  <td>
    <div class="row_button">
      <%= render 'shared/save_button' %>
      <%= render 'shared/cancel_button' %>
    </div>
  </td>
</tr>
</table>
<% end %>

```

E:\workspace\payroll_rails\app\views\admin\overtime_rate_list.html.erb

```

<% if @data.present? && @data[:list].any? %>


| <input class="hdchk" type="checkbox" /> | <div class="sorthead hover" id="hd_duration"> <span class="colheader">Duration (hour)</span> <span class="sorticon"></span> </div> | <div class="sorthead hover" id="hd_year"> <span class="colheader">Year</span> <span class="sorticon"></span> </div> | <div class="sorthead hover" id="hd_pay_rate"> <span class="colheader">Pay Rate</span> <span class="sorticon"></span> </div> |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <input class="chk" type="checkbox" />   | <%= number_with_precision o.duration, :precision => 2 %>                                                                           | <%= h o.year %>                                                                                                     | <%= number_with_precision o.pay_rate, :precision => 2 %>                                                                    |


<% end %>
<input id="id_pg" type="hidden" value="<%= @data[:hasprev] %>,<%= @data[:hasnext] %>,<%= @data[:prevpage] %>,<%= @data[:nextpage] %>,<%= @data[:item msg] %>,<%= @data[:sortcolumn] %>,<%= @data[:sortdir] %>,<%= @data[:page] %>" />

```

E:\workspace\payroll_rails\app\views\admin\overtime_rate_search_form.html.erb

```
<table>
  <tr>
    <td>
      <%= label_tag :id_year, 'Year', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= select_year(0, { :start_year => Time.now.year, :end_year => end_year, :prompt =>
'All' }, { :id => 'id_year', :class => 'select' }) %>
    </td>
    <td colspan="2"></td>
  </tr>
</table>
<br />
<% @include_search = true %>
<%= render 'shared/panel_button' %>
```

E:\workspace\payroll_rails\app\views\admin\overtime_rate\index.html.erb

```
<% @title = 'Employee Overtime Pay Rate' %>
<% @dialog_title = 'Employee Overtime Pay Rate' %>
<% @dialog_add_title = 'Add New Overtime Pay Rate' %>
<% @dialog_edit_title = 'Edit Overtime Pay Rate' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'list' %>
<% end %>
<% content_for :dialog do %>
  <%= render 'shared/dialog_save' %>
<% end %>
```

E:\workspace\payroll_rails\app\views\admin\pay_rate_form.html.erb

```
<%= form_tag '.', :id => @form_id, :class => 'save_form ui-widget-content ui-widget ui-corner-all' do %>
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :staff_id, 'Staff ID', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :staff_id, @payrate.staff_id, :id => 'id_staff_id', :class =>
'text', :placeholder => 'enter staff id',
                           :title => 'enter staff id' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :month, 'Month', :class => 'label' %>
      </td>
      <td>
        <%= select_month(@payrate.month, { :field_name => 'month' }, { :id => 'id_month',
:class => 'select' }) %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :year, 'Year', :class => 'label' %>
      </td>
      <td>
        <%= select_year(@payrate.year, { :start_year => Time.now.year, :end_year => end_year,
:field_name => 'year' }, { :id => 'id_year', :class => 'select' }) %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :hourly_pay_rate, 'Pay rate', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :hourly_pay_rate, @payrate.hourly_pay_rate, :id => 'id_pay_rate',
:class => 'text', :placeholder => 'enter hourly pay rate',
                           :title => 'enter hourly pay rate' %>
      </td>
    </tr>
  </table>
```

```

<tr>
  <td></td>
  <td>
    <div class="row_button">
      <%= render 'shared/save_button' %>
      <%= render 'shared/cancel_button' %>
    </div>
  </td>
</tr>
</table>
<% end %>

E:\workspace\payroll_rails\app\views\admin\pay_rate\_list.html.erb

<% if @data.present? && @data[:list].any? %>
<table class="list_table">
  <thead class="ui-widget-header">
    <tr>
      <th class="data chkbox"><input class="hdchk" type="checkbox" /></th>
      <th class="data">
        <div id="hd_staff_id" class="sorthead hover">
          <span class="colheader">Id</span>
          <span class="sorticon"></span>
        </div>
      </th>
      <th class="data">
        <div id="hd_month" class="sorthead hover">
          <span class="colheader">Month</span>
          <span class="sorticon"></span>
        </div>
      </th>
      <th class="data">
        <div id="hd_year" class="sorthead hover">
          <span class="colheader">Year</span>
          <span class="sorticon"></span>
        </div>
      </th>
      <th class="data">
        <div id="hd_hourly_pay_rate" class="sorthead hover">
          <span class="colheader">Hourly Pay Rate</span>
          <span class="sorticon"></span>
        </div>
      </th>
    </tr>
  </thead>
  <tbody>
    <% @data[:list].each do |o| %>
      <tr id="tr_<=% o.id %>" class="<<=%= cycle("ui-state-active", "ui-state-default") %>">">
        <td class="data chkbox"><input class="chk" type="checkbox" /></td>
        <td class="data"><=%= h o.staff_id %></td>
        <td class="data"><=%= h month_name(o.month) %></td>
        <td class="data"><=%= h o.year %></td>
        <td class="data"><=%= number_with_precision o.hourly_pay_rate, :precision => 2 %></td>
      </tr>
    <% end %>
  </tbody>
</table>
<% end %>
<input id="id_pg" type="hidden" value="<<=%= @data[:hasprev] %>,<=%= @data[:hasnext] %>,<=%= @data[:preppage] %>,<=%= @data[:nextpage] %>,<=%= @data[:item_msg] %>,<=%= @data[:sortcolumn] %>,<=%= @data[:sortdir] %>,<=%= @data[:page] %>" />
```

E:\workspace\payroll_rails\app\views\admin\pay_rate_search_form.html.erb

```

<table>
  <tr>
    <td>
      <=%= label_tag :id_staff_id, 'Id', :class => 'label' %>
    </td>
    <td class="colspacing">
      <=%= text_field_tag :id_staff_id, nil, :id => 'id_staff_id', :class => 'search',
      :placeholder => 'type here to search',
      :title => 'type here to search' %>
    </td>
    <td>
      <=%= label_tag :id_month, 'Month', :class => 'label' %>
    </td>
  </tr>
</table>
```

```

<td>
  <%= select_month(0, { :prompt => 'All' }, { :id => 'id_month', :class => 'select' }) %>
</td>
</tr>
<tr>
  <td>
    <%= label_tag :id_year, 'Year', :class => 'label' %>
  </td>
  <td class="colspacing">
    <%= select_year(0, { :start_year => Time.now.year, :end_year => end_year, :prompt =>
'All' }, { :id => 'id_year', :class => 'select' }) %>
  </td>
  <td colspan="2"></td>
  </tr>
</table>
<br />
<% @include_search = true %>
<%= render 'shared/panel_button' %>

```

E:\workspace\payroll_rails\app\views\admin\pay_rate\index.html.erb

```

<% @title = 'Employee Pay Rate' %>
<% @dialog_title = 'Employee Pay Rate' %>
<% @dialog_add_title = 'Add New Pay Rate' %>
<% @dialog_edit_title = 'Edit Pay Rate' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'list' %>
<% end %>
<% content_for :dialog do %>
  <%= render 'shared/dialog_save' %>
<% end %>

```

E:\workspace\payroll_rails\app\views\admin\payslip_list.html.erb

```

<% if @data.present? && @data[:list].any? %>
  <table class="list_table">
    <thead class="ui-widget-header">
      <tr>
        <th class="data">
          <div id="hd_staff_id" class="sorthead hover">
            <span class="colheader">Id</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd first name" class="sorthead hover">
            <span class="colheader">First (&amp; Middle) Name</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_last_name" class="sorthead hover">
            <span class="colheader">Last Name</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_d-title" class="sorthead hover">
            <span class="colheader">Job Title</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_es-name" class="sorthead hover">
            <span class="colheader">Employment Status</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_dept-name" class="sorthead hover">
            <span class="colheader">Department</span>
            <span class="sorticon"></span>
          </div>
        </th>
      </tr>
    </thead>
    <tbody>

```

```

        </tr>
    </thead>
    <tbody>
        <% @data[:list].each do |o| %>
            <tr id="tr_<%= o.id %>" class="<%= cycle("ui-state-active", "ui-state-default") %>">
                <% ej = o.employee_job %>
                <td class="data"><%= h o.staff_id %></td>
                <td class="data"><%= h o.first_name %></td>
                <td class="data"><%= h o.last_name %></td>
                <td class="data">
                    <% if ej.present? %>
                        <%= h ej.designation.title if ej.designation.present? %>
                    <% end %>
                </td>
                <td class="data">
                    <% if ej.present? %>
                        <%= h ej.employment_status.name if ej.employment_status.present? %>
                    <% end %>
                </td>
                <td class="data">
                    <% if ej.present? %>
                        <%= h ej.department.name if ej.department.present? %>
                    <% end %>
                </td>
            </tr>
        <% end %>
    </tbody>
</table>
<% end %>
<input id="id_pg" type="hidden" value="<%=@data[:hasprev] %>,<%=@data[:hasnext] %>,<%=@data[:prevpage] %>,<%=@data[:nextpage] %>,<%=@data[:item_msg] %>,<%=@data[:sortcolumn] %>,<%=@data[:sortdir] %>,<%=@data[:page] %>" />

```

E:\workspace\payroll_rails\app\views\admin\paylip_\search_form.html.erb

```





```

```

</td>
<td>
  <select id="id_dept" class="select">
    <option value="0">All</option>
    <%= options_from_collection_for_select(@dept, :id, :name) %>
  </select>
</td>
<td>
  <%= label_tag :id_month, 'Month', :class => 'label' %>
</td>
<td>
  <%= select_month(Date.today, { }, { :id => 'id_month', :class => 'select' }) %>
</td>
</tr>
<tr>
<td>
  <%= label_tag :id_year, 'Year', :class => 'label' %>
</td>
<td class="colspacing">
  <%= select_year(Date.today, { :start_year => Time.now.year, :end_year => end_year }, { :id => 'id_year', :class => 'select' }) %>
</td>
<td colspan="2"></td>
</tr>
</table>
<br />
<%= render 'shared/search_button' %>

```

E:\workspace\payroll_rails\app\views\admin\payslip\index.html.erb

```

<% @title = 'Employee Payslip' %>
<% @dialog title = 'Employee Payslip' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'list' %>
<% end %>

```

E:\workspace\payroll_rails\app\views\admin\payslip\payslip_hourly.html.erb

```

<!DOCTYPE html>
<html>
<head>
  <title>Monthly Payslip</title>
  <%= favicon_link_tag "favicon.ico", :rel => "shortcut icon" %>
  <%= favicon_link_tag "favicon.ico", :rel => "icon", :type => "image/x-icon" %>
  <%= stylesheet_link_tag "_payslip", :media => "all" %>
  <%= javascript_include_tag "payslip" %>
  <%= csrf_meta_tags %>
</head>
<body class="ui-widget-content">
  <table class="heading">
    <tr>
      <td valign="top"><span class="boldtext">NAME</span> : <%= @employee.first_name %> <%= @employee.middle_name %> <%= @employee.last_name %></td>
      <td align="right">
        <table>
          <tr>
            <td align="left"><span class="boldtext">PERIOD</span></td>
            <td> : <%= @period %></td>
          </tr>
          <tr>
            <td align="left"><span class="boldtext">EMPLOYEE ID</span></td>
            <td> : <%= @employee.staff_id %></td>
          </tr>
          <tr>
            <td align="left"><span class="boldtext">IC NO.</span></td>
            <td> : <%= @employee.new_ic %></td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
  <table class="data_table">
    <thead>
      <tr>

```

```
 >--- EARNINGS ---</th> <th class="amt"><span>--- RM ---</span></th> <th>--- DEDUCTIONS ---</th> <th class="amt"><span>--- RM ---</span></th> </tr> </thead> <tbody> <tr> <td><span class="det boldtext">TOTAL HOURS WORKED</span></td> <td class="amt"><span><%= number_to_currency @total_hours, :unit => '' %></span></td> <td><span class="det boldtext">EMPLOYEE EPF</span></td> <td class="amt"><span><%= number_to_currency @employee_salary.epf, :unit => '' %></span></td> </tr> <tr> <td><span class="det boldtext">HOURLY PAY RATE</span></td> <td class="amt"><span><%= number_to_currency @hourly_pay_rate, :unit => '' %></span></td> </tr> <td><span class="det boldtext">EMPLOYEE SOCSO</span></td> <td class="amt"><span><%= number_to_currency @employee_salary.socso, :unit => '' %></span></td> </tr> <tr> <td><span class="det boldtext">ALLOWANCE</span></td> <td class="amt"><span><%= number_to_currency @employee_salary.allowance, :unit => '' %></span></td> <td><span class="det boldtext">INCOME TAX</span></td> <td class="amt"><span><%= number_to_currency @employee_salary.income_tax, :unit => '' %></span></td> </tr> <tr> <td class="sum amt boldtext"><span>TOTAL : </span></td> <td class="sum amt"><span><%= number_to_currency @total_earnings, :unit => '' %></span></td> <td class="sum amt boldtext"><span>TOTAL : </span></td> <td class="sum amt"><span><%= number_to_currency @total_deduct, :unit => '' %></span></td> </tr> <tr> <td colspan="2"></td> <td class="amt boldtext"><span>NETT PAY : </span></td> <td class="amt"><span><%= number_to_currency @nett_salary, :unit => '' %></span></td> </tr> </tbody> </table> <table class="subdet"> <tr> <td><span class="boldtext">EPF</span></td> <td> : <%= @employee_salary.epf_no %></td> </tr> <tr> <td><span class="boldtext">SOCSO</span></td> <td> : <%= @employee_salary.socso_no %></td> </tr> </table> <div class="print_container"> <div class="print button hover ui-state-default ui-corner-all"> <span class="icons printicon paddicon">Print</span> </div> </div> </body> </html> |
```

E:\workspace\payroll_rails\app\views\admin\payslip\payslip_monthly.html.erb

```

<!DOCTYPE html>
<html>
<head>
<title>Monthly Payslip</title>
<%= favicon_link_tag "favicon.ico", :rel => "shortcut icon" %>
<%= favicon_link_tag "favicon.ico", :rel => "icon", :type => "image/x-icon" %>
<%= stylesheet_link_tag "_payslip", :media => "all" %>
<%= javascript_include_tag "_payslip" %>
<%= csrf_meta_tags %>
</head>
<body class="ui-widget-content">
<table class="heading">

```

```

<tr>
  <td valign="top"><span class="boldtext">NAME</span> : <%= @employee.first_name %> <%= @employee.middle_name %> <%= @employee.last_name %></td>
  <td align="right">
    <table>
      <tr>
        <td align="left"><span class="boldtext">PERIOD</span></td>
        <td> : <%= @period %></td>
      </tr>
      <tr>
        <td align="left"><span class="boldtext">EMPLOYEE ID</span></td>
        <td> : <%= @employee.staff_id %></td>
      </tr>
      <tr>
        <td align="left"><span class="boldtext">IC NO.</span></td>
        <td> : <%= @employee.new_ic %></td>
      </tr>
    </table>
  </td>
</tr>
</table>
<table class="data_table">
  <thead>
    <tr>
      <th>--- EARNINGS ---</th>
      <th class="amt"><span>--- RM ---</span></th>
      <th>--- DEDUCTIONS ---</th>
      <th class="amt"><span>--- RM ---</span></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td><span class="det boldtext">BASIC PAY</span></td>
      <td class="amt"><span><%= number_to_currency @basic_pay, :unit => '' %></span></td>
      <td><span class="det boldtext">EMPLOYEE EPF</span></td>
      <td class="amt"><span><%= number_to_currency @employee_salary.epf, :unit => '' %></span></td>
    </tr>
    <tr>
      <td><span class="det boldtext">ALLOWANCE</span></td>
      <td class="amt"><span><%= number_to_currency @employee_salary.allowance, :unit => '' %></span></td>
      <td><span class="det boldtext">EMPLOYEE SOCSO</span></td>
      <td class="amt"><span><%= number_to_currency @employee_salary.socso, :unit => '' %></span></td>
    </tr>
    <tr>
      <td><span class="det boldtext">TOTAL OVERTIME</span></td>
      <td class="amt"><span><%= number_to_currency @total_overtime, :unit => '' %></span></td>
      <td><span class="det boldtext">INCOME TAX</span></td>
      <td class="amt"><span><%= number_to_currency @employee_salary.income_tax, :unit => '' %></span></td>
    </tr>
    <tr>
      <td><span class="det boldtext">TOTAL OVERTIME EARNINGS</span></td>
      <td class="amt"><span><%= number_to_currency @total_overtime_earnings, :unit => '' %></span></td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td class="sum amt boldtext"><span>TOTAL : </span></td>
      <td class="sum amt"><span><%= number_to_currency @total_earnings, :unit => '' %></span></td>
      <td class="sum amt boldtext"><span>TOTAL : </span></td>
      <td class="sum amt"><span><%= number_to_currency @total_deduct, :unit => '' %></span></td>
    </tr>
    <tr>
      <td colspan="2"></td>
      <td class="amt boldtext"><span>NETT PAY : </span></td>
      <td class="amt"><span><%= number_to_currency @nett_salary, :unit => '' %></span></td>
    </tr>
  </tbody>
</table>
<table class="subdet">

```

```

<tr>
  <td><span class="boldtext">EPF</span></td>
  <td> : <%= @employee.salary.epf_no %></td>
</tr>
<tr>
  <td><span class="boldtext">SOC SO</span></td>
  <td> : <%= @employee.salary.socso_no %></td>
</tr>
</table>
<div class="print_container">
  <div class="print_button hover ui-state-default ui-corner-all">
    <span class="icons printicon paddicon">Print</span>
  </div>
</div>
</body>
</html>

```

E:\workspace\payroll_rails\app\views\admin\salary_adjustment_form.html.erb

```

<%= form_tag '.', :id => @form_id, :class => 'save_form ui-widget-content ui-widget ui-corner-all' do %>
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :staff_id, 'Staff ID', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :staff_id, @adj.staff_id, :id => 'id_staff_id', :class => 'text', :placeholder => 'enter staff id', :title => 'enter staff id' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :inc, 'Increment', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :inc, @adj.inc, :id => 'id_inc', :class => 'text', :placeholder => 'enter increment', :title => 'enter increment' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :month, 'Month', :class => 'label' %>
      </td>
      <td>
        <%= select_month(@adj.month, { :field_name => 'month' }, { :id => 'id_month', :class => 'select' }) %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :year, 'Year', :class => 'label' %>
      </td>
      <td>
        <%= select_year(@adj.year, { :start_year => Time.now.year, :end_year => end_year, :field_name => 'year' }, { :id => 'id_year', :class => 'select' }) %>
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <div class="row_button">
          <%= render 'shared/save_button' %>
          <%= render 'shared/cancel_button' %>
        </div>
      </td>
    </tr>
  </table>
<% end %>

```

E:\workspace\payroll_rails\app\views\admin\salary_adjustment_list.html.erb

```
<% if @data.present? && @data[:list].any? %>
  <table class="list_table">
    <thead class="ui-widget-header">
      <tr>
        <th class="data chkbox"><input class="hdchk" type="checkbox" /></th>
        <th class="data">
          <div id="hd_staff_id" class="sorthead hover">
            <span class="colheader">Id</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_month" class="sorthead hover">
            <span class="colheader">Month</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_year" class="sorthead hover">
            <span class="colheader">Year</span>
            <span class="sorticon"></span>
          </div>
        </th>
        <th class="data">
          <div id="hd_hourly_pay_rate" class="sorthead hover">
            <span class="colheader">Increment</span>
            <span class="sorticon"></span>
          </div>
        </th>
      </tr>
    </thead>
    <tbody>
      <% @data[:list].each do |o| %>
        <tr id="tr_<%= o.id %>" class="<%= cycle("ui-state-active", "ui-state-default") %>">
          <td class="data chkbox"><input class="chk" type="checkbox" /></td>
          <td class="data"><%= h o.staff_id %></td>
          <td class="data"><%= h month_name(o.month) %></td>
          <td class="data"><%= h o.year %></td>
          <td class="data"><%= number_with_precision o.inc, :precision => 2 %></td>
        </tr>
      <% end %>
    </tbody>
  </table>
<% end %>
<input id="id_pg" type="hidden" value="<%=
  @data[:hasprev] %>,<%
  @data[:hasnext] %>,<%
  @data[:prevpage] %>,<%
  @data[:nextpage] %>,<%
  @data[:item_msg] %>,<%
  @data[:sortcolumn] %>,<%
  @data[:sortdir] %>,<%
  @data[:page] %>" />
```

E:\workspace\payroll_rails\app\views\admin\salary_adjustment_search_form.html.erb

```
<table>
  <tr>
    <td>
      <%= label_tag :id_staff_id, 'Id', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= text_field_tag :id_staff_id, nil, :id => 'id_staff_id', :class => 'search',
      :placeholder => 'type here to search',
      :title => 'type here to search' %>
    </td>
  <td>
    <%= label_tag :id_month, 'Month', :class => 'label' %>
  </td>
  <td>
    <%= select_month(0, { :prompt => 'All' }, { :id => 'id_month', :class => 'select' }) %>
  </td>
</tr>
<tr>
  <td>
    <%= label_tag :id_year, 'Year', :class => 'label' %>
  </td>
  <td class="colspacing">
    <%= select_year(0, { :start_year => Time.now.year, :end_year => end_year, :prompt =>
    'All' }, { :id => 'id_year', :class => 'select' }) %>
  </td>
```

```

        <td colspan="2"></td>
    </tr>
</table>
<br />
<% @include_search = true %>
<%= render 'shared/panel_button' %>

E:\workspace\payroll_rails\app\views\admin\salary_adjustment\index.html.erb

<% @title = 'Employee Salary Adjustment' %>
<% @dialog title = 'Employee Salary Adjustment' %>
<% @dialog_add_title = 'Add New Salary Adjustment' %>
<% @dialog_edit_title = 'Edit Salary Adjustment' %>
<% content_for :search_form do %>
    <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
    <%= render 'list' %>
<% end %>
<% content_for :dialog do %>
    <%= render 'shared/dialog_save' %>
<% end %>

E:\workspace\payroll_rails\app\views\admin\total_work_hours_chart\_chart.html.erb

<div id="cht1" style="height: 400px"></div>

E:\workspace\payroll_rails\app\views\admin\total_work_hours_chart\_search_form.html.erb

<table>
    <tr>
        <td>
            <%= label_tag :id_staff_id, 'Id', :class => 'label' %>
        </td>
        <td class="colspacing">
            <%= text_field_tag :id_staff_id, nil, :id => 'id_staff_id', :class => 'search',
            :placeholder => 'type here to search',
            :title => 'type here to search' %>
        </td>
        <td>
            <%= label_tag :id_year, 'Year', :class => 'label' %>
        </td>
        <td class="colspacing">
            <%= select year(Date.today, { :start year => Time.now.year, :end year => end year }, {
            :id => 'id_year', :class => 'select' }) %>
        </td>
        <td></td>
    </tr>
    <tr>
        <td>
            <%= label_tag :id_month, 'Month', :class => 'label' %>
        </td>
        <td colspan="4">
            <label>All</label>
            <input type="checkbox" name="month" value="0" class="chkall" />
            <% for k, v in @month hash %>
            <label><%= k[0..2] %></label>
            <input type="checkbox" name="month" value="<%= v %>" class="chkmonth" />
            <% end %>
        </td>
    </tr>
</table>
<br />
<%= render 'shared/chart_button' %>

E:\workspace\payroll_rails\app\views\admin\total_work_hours_chart\index.html.erb

<% @title = 'Employee Total Hours Worked Chart' %>
<% @dialog title = 'Employee Total Hours Worked Chart' %>
<% content_for :search_form do %>
    <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
    <%= render 'chart' %>
<% end %>
```

E:\workspace\payroll_rails\app\views\admin\user_form.html.erb

```
<%= form_tag '.', :id => @form_id, :class => 'save form ui-widget-content ui-widget ui-corner-all' do %>
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :role, 'User Role', :class => 'label' %>
      </td>
      <td>
        <%= select_tag :role, options_for_select(User.roles, @user.role), :id => 'id_role', :class => 'select' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :username, 'Username', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :username, @user.username, :id => 'id_username', :class => 'text', :placeholder => 'enter username', :title => 'enter username' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :status, 'Status', :class => 'label' %>
      </td>
      <td>
        <%= select_tag :status, options_for_select(User.statuses, @user.status == true || @user.status.blank? ? 1 : 0), :id => 'id_status', :class => 'select' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :pwd, 'Password', :class => 'label' %>
      </td>
      <td>
        <%= password_field_tag :pwd, @user.pwd, :id => 'id_password', :class => 'text', :placeholder => 'enter password', :title => 'Minimum is 4 characters' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :pwd_confirmation, 'Confirm Password', :class => 'label' %>
      </td>
      <td>
        <%= password_field_tag :pwd_confirmation, @user.pwd_confirmation, :id => 'id_passwordconfirm', :class => 'text', :placeholder => 'enter password confirmation', :title => 'enter password confirmation' %>
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <div class="row button">
          <%= render 'shared/save_button' %>
          <%= render 'shared/cancel_button' %>
        </div>
      </td>
    </tr>
  </table>
<% end %>
```

E:\workspace\payroll_rails\app\views\admin\user_list.html.erb

```
<% if @data.present? && @data[:list].any? %>
  <table class="list table">
    <thead class="ui-widget-header">
      <tr>
        <th class="data chkbox"><input class="hdchk" type="checkbox" /></th>
        <th class="data">
          <div id="hd_username" class="sorthead hover">
            <span class="colheader">Username</span>
            <span class="sorticon"></span>
          </div>
        </th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td><input checked="" type="checkbox" /></td>
        <td>
          <div id="hd_username" class="sorthead hover">
            <span class="colheader">Username</span>
            <span class="sorticon"></span>
          </div>
        </td>
      </tr>
    </tbody>
  </table>
```

```

</th>
<th class="data">
  <div id="hd_role" class="sorthead hover">
    <span class="colheader">User Role</span>
    <span class="sorticon"></span>
  </div>
</th>
<th class="data">
  <div id="hd_status" class="sorthead hover">
    <span class="colheader">Status</span>
    <span class="sorticon"></span>
  </div>
</th>
</tr>
</thead>
<tbody>
<% @data[:list].each do |o| %>
  <tr id="tr_<%= o.id %>" class="<% cycle("ui-state-active", "ui-state-default") %>">
    <td class="data checkbox"><input class="chk" type="checkbox" /></td>
    <td class="data"><%= h o.username %></td>
    <td class="data"><%= h o.role_display %></td>
    <td class="data"><%= h o.status_display %></td>
  </tr>
<% end %>
</tbody>
</table>
<% end %>
<input id="id_pg" type="hidden" value="<%=@data[:hasprev] %>,<%=@data[:hasnext] %>,<%=@data[:prevpage] %>,<%=@data[:nextpage] %>,<%=@data[:item_msg] %>,<%=@data[:sortcolumn] %>,<%=@data[:sortdir] %>,<%=@data[:page] %>" />

```

E:\workspace\payroll_rails\app\views\admin\user_search_form.html.erb

```

<table>
  <tr>
    <td>
      <%= label_tag :id_username, 'Username', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= text_field_tag :username, nil, :id => 'id_username', :class => 'text', :placeholder => 'type here to search', :title => 'type here to search' %>
    </td>
    <td>
      <%= label_tag :id_user_role, 'User Role', :class => 'label' %>
    </td>
    <td class="colspacing">
      <select id="id_user_role" class="search_option">
        <%= options_for_select [['All', 0], ['Admin', 1], ['Normal User', 2]] %>
      </select>
    </td>
  </tr>
  <tr>
    <td>
      <%= label_tag :id_employee, 'Employee Name', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= text_field_tag :employee, nil, :id => 'id_employee', :class => 'text', :placeholder => 'type here to search', :title => 'type here to search' %>
    </td>
    <td>
      <%= label_tag :id_status, 'Status', :class => 'label' %>
    </td>
    <td>
      <select id="id_status" class="search_option">
        <%= options_for_select [['All', 0], ['Enabled', 1], ['Disabled', 2]] %>
      </select>
    </td>
  </tr>
</table>
<br />
<% @include_search = true %>
<%= render 'shared/panel_button' %>

```

```
E:\workspace\payroll_rails\app\views\admin\user\index.html.erb
```

```
<% @title = 'User' %>
<% @dialog_title = 'User' %>
<% @dialog_add_title = 'Add New User' %>
<% @dialog_edit_title = 'Edit User' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'list' %>
<% end %>
<% content_for :dialog do %>
  <%= render 'shared/dialog_save' %>
<% end %>
```

```
E:\workspace\payroll_rails\app\views\user\contact\_form.html.erb
```

```
<form id="save-form" class="save_form ui-widget-content ui-widget" method="post">
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :address_1, 'Address 1', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :address_1, @employee_contact.address_1, :id => 'id_address_1',
        :class => 'text', :placeholder => 'enter street',
        :title => 'enter street' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :address_2, 'Address 2', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :address_2, @employee_contact.address_2, :id => 'id_address_2',
        :class => 'text', :placeholder => 'enter street',
        :title => 'enter street' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :address_3, 'Address 2', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :address_3, @employee_contact.address_3, :id => 'id_address_3',
        :class => 'text', :placeholder => 'enter street',
        :title => 'enter street' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl"> <%= label_tag :city, 'City', :class => 'label' %> </td>
      <td> <%= text_field_tag :city, @employee_contact.city, :id => 'id_city', :class =>
      'text', :placeholder => 'enter city',
        :title => 'enter city' %> </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :state, 'State', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :state, @employee_contact.state, :id => 'id_state', :class =>
      'text', :placeholder => 'enter state',
        :title => 'enter state' %>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :postcode, 'Postal Code', :class => 'label' %>
      </td>
      <td>
        <%= text_field_tag :postcode, @employee_contact.postcode, :id => 'id_postcode', :class =>
      'text', :placeholder => 'enter postal code',
        :title => 'enter postal code' %>
      </td>
    </tr>
    <tr>
```

```

<td valign="top" class="lbl">
  <%= label_tag :country, 'Country', :class => 'label' %>
</td>
<td>
  <%= text_field_tag :country, @employee_contact.country, :id => 'id_country', :class =>
'text', :placeholder => 'enter country',
          :title => 'enter country' %>
</td>
</tr>
<tr>
  <td colspan="2">
    <hr />
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :home_phone, 'Home Telephone', :class => 'label' %>
  </td>
  <td> <%= text_field_tag :home_phone, @employee_contact.home_phone, :id =>
'id_home_phone', :class => 'text', :placeholder => 'enter home phone',
          :title => 'enter home phone' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :mobile_phone, 'Mobile', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :mobile_phone, @employee_contact.mobile_phone, :id =>
'id_mobile_phone', :class => 'text', :placeholder => 'enter mobile phone',
          :title => 'enter mobile phone' %>
  </td>
</tr>
<tr>
  <td colspan="2">
    <hr />
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :work_email, 'Work Email', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :work_email, @employee_contact.work_email, :id => 'id_work_email',
:class => 'text', :placeholder => 'enter work email',
          :title => 'enter work email' %>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :other_email, 'Other Email', :class => 'label' %>
  </td>
  <td>
    <%= text_field_tag :other_email, @employee_contact.other_email, :id =>
'id_other_email', :class => 'text', :placeholder => 'enter other email',
          :title => 'enter other email' %>
  </td>
</tr>
</table>
</form>

```

E:\workspace\payroll_rails\app\views\user\contact\index.html.erb

```

<% @dialog_title = 'Contact Details' %>
<div class="page_title ui-widget-header ui-widget">Contact Details</div>
<%= render 'form' %>
<div style="margin-top: 10px">
  <%= render 'shared/save_button' %>
</div>
<%= render 'shared/alert_panel' %>

```

```
E:\workspace\payroll_rails\app\views\user\hourly_payroll_chart\_chart.html.erb
```

```
<div id="cht1" style="height: 400px"></div>
<div id="cht2" style="height: 400px"></div>
```

```
E:\workspace\payroll_rails\app\views\user\hourly_payroll_chart\_search_form.html.erb
```

```
<table>
  <tr>
    <td>
      <%= label_tag :id_year, 'Year', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= select_year(Date.today, { :start_year => Time.now.year, :end_year => end_year }, {
:id => 'id_year', :class => 'select' }) %>
    </td>
    <td></td>
  </tr>
  <tr>
    <td>
      <%= label_tag :id_month, 'Month', :class => 'label' %>
    </td>
    <td colspan="2">
      <label>All</label>
      <input type="checkbox" name="month" value="0" class="chkall" />
      <% for k, v in @month_hash %>
        <label><%= k[0..2] %></label>
        <input type="checkbox" name="month" value="<%= v %>" class="chkmonth" />
      <% end %>
    </td>
  </tr>
</table>
<br />
<%= render 'shared/chart_button' %>
```

```
E:\workspace\payroll_rails\app\views\user\hourly_payroll_chart\index.html.erb
```

```
<% @title = 'Employee Hourly Payroll Chart' %>
<% @dialog_title = 'Employee Hourly Payroll Chart' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'chart' %>
<% end %>
```

```
E:\workspace\payroll_rails\app\views\user\info\_form.html.erb
```

```
<form id="save-form" class="save_form ui-widget-content ui-widget" method="post">
  <table>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :full_name, 'Full Name', :class => 'label' %>
      </td>
      <td valign="top">
        <%= text_field_tag :first_name, @employee.first_name, :id => 'id_first_name', :class => 'text', :placeholder => 'enter first name',
          :title => 'enter first name' %>
      </td>
      <td valign="top">
        <%= text_field_tag :middle_name, @employee.middle_name, :id => 'id_middle_name',
          :class => 'text', :placeholder => 'enter middle name',
          :title => 'enter middle name' %>
      </td>
      <td valign="top">
        <%= text_field_tag :last_name, @employee.last_name, :id => 'id_last_name', :class => 'text',
          :placeholder => 'enter last name',
          :title => 'enter last name' %>
      </td>
    </tr>
    <tr>
      <td></td>
      <td align="center">First Name</td>
      <td align="center">Middle Name</td>
      <td align="center">Last Name</td>
    </tr>
  </table>
```

```

<hr />
<table>
  <tr>
    <td valign="top" class="lbl">
      <%= label_tag :staff_id, 'Employee Id', :class => 'label' %>
    </td>
    <td valign="top" class="colspacing">
      <span><%= h @employee.staff_id %>
    </td>
    <td valign="top" class="lbl">
      <%= label_tag :new_ic, 'New I.C No.', :class => 'label' %>
    </td>
    <td>
      <%= text_field_tag :new_ic, @employee.new_ic, :id => 'id_new_ic', :class => 'text',
      :placeholder => 'enter new I.C no.', :title => 'enter new I.C no.' %>
    </td>
  </tr>
  <tr>
    <td valign="top" class="lbl">
      <%= label_tag :passport_no, 'Passport No', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= text_field_tag :passport_no, @employee.passport_no, :id => 'Passport No.', :class => 'text',
      :placeholder => 'enter passport no.', :title => 'enter passport no.' %>
    </td>
    <td valign="top" class="lbl">
      <%= label_tag :old_ic, 'Old I.C No.', :class => 'label' %>
    </td>
    <td>
      <%= text_field_tag :old_ic, @employee.old_ic, :id => 'id_old_ic', :class => 'text',
      :placeholder => 'enter old I.C no.', :title => 'enter old I.C no.' %>
    </td>
  </tr>
  <tr>
    <td colspan="4">
      <hr />
    </td>
  </tr>
  <tr>
    <td valign="top" class="lbl">
      <%= label_tag :gender, 'Gender', :class => 'label' %>
    </td>
    <td valign="top">
      <%= radio_button_tag :gender, 'M', @employee.gender == 'M' ? true : false %>
      <%= label_tag :gender_m, 'Male' %>
      <%= radio_button_tag :gender, 'F', @employee.gender == 'F' ? true : false %>
      <%= label_tag :gender_f, 'Female', :id => 'lb_gender_f' %> </td>
    <td valign="top" class="lbl">
      <%= label_tag :marital_status, 'Marital Status', :class => 'label' %>
    </td>
    <td>
      <%= select_tag :marital_status,
        options_for_select([['-- Select --', '0'], ['Single', 'S'], ['Married', 'M'],
        ['Other', 'O']], @employee.marital_status.blank? ? '0' : @employee.marital_status),
        :id => 'id_marital_status', :class => 'select' %>
    </td>
  </tr>
  <tr>
    <td valign="top" class="lbl">
      <%= label_tag :nationality, 'Nationality', :class => 'label' %>
    </td>
    <td>
      <%= text_field_tag :nationality, @employee.nationality, :id => 'id_nationality',
      :class => 'text', :placeholder => 'enter nationality',
      :title => 'enter nationality' %>
    </td>
    <td colspan="2"></td>
  </tr>
  <tr>
    <td valign="top" class="lbl">
      <%= label_tag :dob, 'Date of Birth', :class => 'label' %>
    </td>
    <td class="colspacing">

```

```

    <%= text_field_tag :dob, fmt_date(@employee.dob), :id => 'id_dob', :class => 'text'
date_input, :placeholder => 'dd-mm-YYYY' %>
</td>
<td valign="top" class="lbl">
    <%= label_tag :place_of_birth, 'Place of Birth', :class => 'label' %>
</td>
<td>
    <%= text_field_tag :place_of_birth, @employee.place_of_birth, :id =>
'id_place_of_birth', :class => 'text', :placeholder => 'enter place of birth',
:title => 'enter place of birth' %>
</td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :race, 'Race', :class => 'label' %>
    </td>
    <td class="colspacing">
        <%= text_field_tag :race, @employee.race, :id => 'id_race', :class => 'text',
:placeholder => 'enter race',
:title => 'enter race' %>
    </td>
    <td>
        <%= label_tag :is_bumi, 'Is Bumiputra', :class => 'label' %>
    </td>
    <td>
        <%= check_box_tag :is_bumi, nil, @employee.is_bumi, :id => 'id_is_bumi' %>
    </td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :religion, 'Religion', :class => 'label' %>
    </td>
    <td>
        <%= text_field_tag :religion, @employee.religion, :id => 'id_religion', :class =>
'text', :placeholder => 'enter religion',
:title => 'enter religion' %>
    </td>
    <td colspan="2"></td>
</tr>
<tr>
    <td valign="top" class="lbl">
        <%= label_tag :user_id, 'Username', :class => 'label' %>
    </td>
    <td>
        <span><%= h @user.username %></span>
    </td>
    <td colspan="2"></td>
</tr>
</table>
</form>

```

E:\workspace\payroll_rails\app\views\user\info\index.html.erb

```

<% @dialog_title = 'Personal Details' %>
<div class="page_title ui-widget-header ui-widget">Personal Details</div>
<%= render 'form' %>
<div style="margin-top: 10px">
    <%= render 'shared/save_button' %>
</div>
<%= render 'shared/alert_panel' %>

```

E:\workspace\payroll_rails\app\views\user\job\index.html.erb

```

<% @dialog_title = 'Job Details' %>
<div class="page_title ui-widget-header ui-widget">Job Details</div>
<div class="ui-widget-content ui-widget">
    <table>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :designation_id, 'Job Title', :class => 'label' %>
            </td>
            <td>
                <span><%= h @employee_job.designation.title if @employee_job.designation.present?
%></span>
            </td>
        </tr>
        <tr>

```

```

<td valign="top" class="lbl">
  <%= label :employment_status_id, 'Employment Status', :class => 'label' %>
</td>
<td>
  <span><%= h @employee_job.employment_status.name if
@employee_job.employment_status.present? %></span>
</td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label :job_category_id, 'Job Category', :class => 'label' %>
  </td>
  <td>
    <span><%= h @employee_job.job_category.name if @employee_job.job_category.present?
%></span>
  </td>
  </tr>
  <tr>
    <td valign="top" class="lbl">
      <%= label :join_date, 'Join Date', :class => 'label' %>
    </td>
    <td>
      <span><%= h fmt_date(@employee_job.join_date) %></span>
    </td>
  </tr>
  <tr>
    <td valign="top" class="lbl">
      <%= label :confirm_date, 'Confirm Date', :class => 'label' %>
    </td>
    <td>
      <span><%= h fmt_date(@employee_job.confirm_date) %></span>
    </td>
  </tr>
  <tr>
    <td valign="top" class="lbl">
      <%= label :department_id, 'Department', :class => 'label' %>
    </td>
    <td>
      <span><%= h @employee_job.department.name if @employee_job.department.present?
%></span>
    </td>
  </tr>
</table>
</div>
<%= render 'shared/alert_panel' %>
```

E:\workspace\payroll_rails\app\views\user\overtime_chart_chart.html.erb

```
<div id="cht1" style="height: 400px"></div>
```

E:\workspace\payroll_rails\app\views\user\overtime_chart_search_form.html.erb

```

<table>
  <tr>
    <td>
      <%= label_tag :id_year, 'Year', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= select_year(Date.today, { :start_year => Time.now.year, :end_year => end_year }, {
:id => 'id_year', :class => 'select' }) %>
    </td>
    <td></td>
  </tr>
  <tr>
    <td>
      <%= label_tag :id_month, 'Month', :class => 'label' %>
    </td>
    <td colspan="2">
      <label>All</label>
      <input type="checkbox" name="month" value="0" class="chkall" />
      <% for k, v in @month_hash %>
        <label><%= k[0..2] %></label>
        <input type="checkbox" name="month" value="<%= v %>" class="chkmonth" />
      <% end %>
    </td>
  </tr>
</table>
```

```

<br />
<%= render 'shared/chart_button' %>

E:\workspace\payroll_rails\app\views\user\overtime_chart\index.html.erb

<% @title = 'Employee Overtime Chart' %>
<% @dialog_title = 'Employee Overtime Chart' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'chart' %>
<% end %>

E:\workspace\payroll_rails\app\views\user\payslip\_search_form.html.erb

|                                                        |                                                                                                                                   |
|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <%= label_tag :id_year, 'Year', :class => 'label' %>   | <%= select_year(Date.today, { :start_year => Time.now.year, :end_year => end_year }, { :id => 'id_year', :class => 'select' }) %> |
| <%= label_tag :id_month, 'Month', :class => 'label' %> | <%= select_month(Date.today, { }, { :id => 'id_month', :class => 'select' }) %>                                                   |


<br />
<%= render 'shared/gen_button' %>

E:\workspace\payroll_rails\app\views\user\payslip\index.html.erb

Employee Payslip



|                             |
|-----------------------------|
| <%= render 'search_form' %> |
|-----------------------------|


<%= render 'shared/alert_panel' %>

E:\workspace\payroll_rails\app\views\user\qualification\_form.html.erb

<form id="save-form" class="save_form ui-widget-content ui-widget" method="post">


|                                                             |                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <%= label_tag :level, 'Level', :class => 'label' %>         | <select id="id_level" class="select"> <option value="0">-- Select --</option> <%= options_for_select([['-- Select --', 0], ['PhD', 1], ['Masters Degree', 2], ['Bachelor Degree', 3], ['Diploma', 4], ['STPM', 5], ['SPM', 6], ['Others', 7]], @employee_qualification.level.blank? ? 0 : @employee_qualification.level) %> </select> |
| <%= label_tag :institute, 'Institute', :class => 'label' %> | <%= text_field_tag :institute, @employee_qualification.institute, :id => 'id_institute', :class => 'text', :placeholder => 'enter institute', :title => 'enter institute' %>                                                                                                                                                          |


```

```

        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :major, 'Major', :class => 'label' %>
        </td>
        <td>
            <%= text_field_tag :major, @employee_qualification.major, :id => 'id_major', :class =>
'text', :placeholder => 'enter major',
                :title => 'enter major' %>
        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :year, 'Year', :class => 'label' %>
        </td>
        <td>
            <%= text_field_tag :year, @employee_qualification.year, :id => 'id_year', :class =>
'text', :placeholder => 'enter year',
                :title => 'enter year' %>
        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :gpa, 'GPA/Grade', :class => 'label' %>
        </td>
        <td>
            <%= text_field_tag :gpa, @employee_qualification.gpa, :id => 'id_gpa', :class =>
'text', :placeholder => 'enter gpa/grade',
                :title => 'enter gpa/grade' %>
        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :start_date, 'Start Date', :class => 'label' %>
        </td>
        <td>
            <%= text_field_tag :start_date, fmt_date(@employee_qualification.start_date), :id =>
'id_start_date', :class => 'text_date_input', :placeholder => 'dd-mm-YYYY' %>
        </td>
    </tr>
    <tr>
        <td valign="top" class="lbl">
            <%= label_tag :end_date, 'End Date', :class => 'label' %>
        </td>
        <td>
            <%= text_field_tag :end_date, fmt_date(@employee_qualification.end_date), :id =>
'id_end_date', :class => 'text_date_input', :placeholder => 'dd-mm-YYYY' %>
        </td>
    </tr>
</table>
</form>

```

E:\workspace\payroll_rails\app\views\user\qualification\index.html.erb

```

<% @dialog_title = 'Qualifications' %>
<div class="page_title ui-widget-header ui-widget">Qualifications</div>
<%= render 'form' %>
<div style="margin-top: 10px">
    <%= render 'shared/save_button' %>
</div>
<%= render 'shared/alert_panel' %>

```

E:\workspace\payroll_rails\app\views\user\salary\index.html.erb

```

<% @dialog_title = 'Salary Details' %>
<div class="page_title ui-widget-header ui-widget">Salary Details</div>
<div class="ui-widget-content ui-widget">
    <table>
        <tr>
            <td valign="top" class="lbl">
                <%= label_tag :salary, 'Salary', :class => 'label' %>
            </td>
            <td>
                <span><%= number_to_currency @basic_pay, :unit => '' %></span>
            </td>
        </tr>
    </table>

```

```

<tr>
  <td valign="top" class="lbl">
    <%= label_tag :allowance, 'Allowance', :class => 'label' %>
  </td>
  <td>
    <span><%= number_to_currency @employee_salary.allowance, :unit => '' %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :epf, 'EPF Deduction', :class => 'label' %>
  </td>
  <td>
    <span><%= number_to_currency @employee_salary.epf, :unit => '' %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :socso, 'SOCSE Deduction', :class => 'label' %>
  </td>
  <td>
    <span><%= number_to_currency @employee_salary.socso, :unit => '' %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :income_tax, 'Income Tax Deduction', :class => 'label' %>
  </td>
  <td>
    <span><%= number_to_currency @employee_salary.income_tax, :unit => '' %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :bank_name, 'Bank Name', :class => 'label' %>
  </td>
  <td>
    <span><%= h @employee_salary.bank_name %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :bank_acc_no, 'Bank Account No.', :class => 'label' %>
  </td>
  <td>
    <span><%= h @employee_salary.bank_acc_no %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :bank_acc_type, 'Bank Account Type', :class => 'label' %>
  </td>
  <td>
    <span><%= h @employee_salary.bank_acc_type %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :bank_address, 'Bank Address', :class => 'label' %>
  </td>
  <td>
    <span><%= h @employee_salary.bank_address %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :epf_no, 'EPF no.', :class => 'label' %>
  </td>
  <td>
    <span><%= h @employee_salary.epf_no %></span>
  </td>
</tr>
<tr>
  <td valign="top" class="lbl">
    <%= label_tag :socso_no, 'SOCSE no.', :class => 'label' %>
  </td>
  <td>

```

```

        <span><%= h @employee_salary.socso_no %></span>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :income_tax_no, 'Income Tax no.', :class => 'label' %>
      </td>
      <td>
        <span><%= h @employee_salary.income_tax_no %></span>
      </td>
    </tr>
    <tr>
      <td valign="top" class="lbl">
        <%= label_tag :pay_type, 'Pay Type', :class => 'label' %>
      </td>
      <td>
        <span><%= h @employee_salary.display_pay_type %></span>
      </td>
    </tr>
  </table>
</div>
<%= render 'shared/alert_panel' %>

E:\workspace\payroll_rails\app\views\user\total_work_hours_chart\_chart.html.erb

<div id="cht1" style="height: 400px"></div>

E:\workspace\payroll_rails\app\views\user\total_work_hours_chart\_search_form.html.erb

<table>
  <tr>
    <td>
      <%= label_tag :id_year, 'Year', :class => 'label' %>
    </td>
    <td class="colspacing">
      <%= select_year(Date.today, { :start_year => Time.now.year, :end_year => end_year }, {
:id => 'id_year', :class => 'select' }) %>
    </td>
    <td></td>
  </tr>
  <tr>
    <td>
      <%= label_tag :id_month, 'Month', :class => 'label' %>
    </td>
    <td colspan="2">
      <label>All</label>
      <input type="checkbox" name="month" value="0" class="chkall" />
      <% for k, v in @month_hash %>
        <label><%= k[0..2] %></label>
        <input type="checkbox" name="month" value="<%= v %>" class="chkmonth" />
      <% end %>
    </td>
  </tr>
</table>
<br />
<%= render 'shared/chart_button' %>

E:\workspace\payroll_rails\app\views\user\total_work_hours_chart\index.html.erb

<% @title = 'Employee Total Hours Worked Chart' %>
<% @dialog title = 'Employee Total Hours Worked Chart' %>
<% content_for :search_form do %>
  <%= render 'search_form' %>
<% end %>
<% content_for :content_list do %>
  <%= render 'chart' %>
<% end %>
```

E:\workspace\payroll_rails\app\views\user\user_menu.html.erb

```
<div id="menu">
  <h3><%= link_to 'Personal', '#' %></h3>
  <div>
    <div><%= link_to 'Personal Details', '#', :onclick => "return info.load()", :id =>
'menu_info' %></div>
    <div><%= link_to 'Contact Details', '#', :onclick => "return contact.load()" %></div>
    <div><%= link_to 'Job', '#', :onclick => "return job.load()" %></div>
    <div><%= link_to 'Salary', '#', :onclick => "return salary.load()" %></div>
    <div><%= link_to 'Qualification', '#', :onclick => "return qualification.load()" %></div>
  </div>
  <h3><%= link_to 'Payroll', '#' %></h3>
  <div>
    <div><%= link_to 'Total Hours Worked Chart', '#', :onclick => "return wkcht.load()" %></div>
  <% if @pay_type == 1 %>
    <div><%= link_to 'Overtime Chart', '#', :onclick => "return otcht.load()" %></div>
  <% else %>
    <div><%= link_to 'Hourly Payroll Chart', '#', :onclick => "return hpcht.load()" %></div>
  <% end %>
    <div><%= link_to 'Payslip', '#', :onclick => "return payslip.load()" %></div>
  </div>
</div>
```

E:\workspace\payroll_rails\app\views\user\user\index.html.erb

```
<div id="maincontainer">
  <div id="topsection" class="ui-state-default">
    <div class="innertube">
      <%= render 'shared/status_panel' %>
      <%= render 'shared/logout' %>
      <%= render 'shared/theme_select' %>
    </div>
  </div>
  <div id="contentwrapper">
    <div id="contentcolumn">
      <div class="innertube"></div>
    </div>
  </div>
  <div id="leftcolumn">
    <div class="innertube">
      <%= render 'menu' %>
    </div>
  </div>
</div>
<div id="error-dialog" title="Internal Server Error">
  <div id="error_dialog"></div>
</div>
<div id="progress_status">
  <%= render 'shared/progress_panel' %>
</div>
```

```

E:\workspace\payroll_rails\app\assets\javascripts\_payslip.js

//= require self
//= require jquery_lib
//= require jquery.themes.min
//= require utils
//= require admin/wpayslip

E:\workspace\payroll_rails\app\assets\javascripts\admin.js

// This is a manifest file that'll be compiled into application.js, which will include all the
files
// listed below.
//
// Any JavaScript/Coffee file within this directory, lib/assets/javascripts,
vendor/assets/javascripts,
// or vendor/assets/javascripts of plugins, if any, can be referenced here using a relative
path.
//
// It's not advisable to add code directly here, but if you do, it'll appear at the bottom of
the
// the compiled file.
//
// WARNING: THE FIRST BLANK LINE MARKS THE END OF WHAT'S TO BE PROCESSED, ANY BLANK LINE
SHOULD
// GO AFTER THE REQUIRES BELOW.
//
//= require application
//= require admin/base
//= require admin/user
//= require admin/emp
//= require admin/designation
//= require admin/empstatus
//= require admin/jobcat
//= require admin/dept
//= require admin/payrate
//= require admin/overtimerate
//= require admin/hpcht
//= require admin/payslip
//= require admin/att
//= require admin/otchht
//= require admin/wkcht
//= require admin/saladj
// require_tree .

E:\workspace\payroll_rails\app\assets\javascripts\application.js

// This is a manifest file that'll be compiled into application.js, which will include all the
files
// listed below.
//
// Any JavaScript/Coffee file within this directory, lib/assets/javascripts,
vendor/assets/javascripts,
// or vendor/assets/javascripts of plugins, if any, can be referenced here using a relative
path.
//
// It's not advisable to add code directly here, but if you do, it'll appear at the bottom of
the
// the compiled file.
//
// WARNING: THE FIRST BLANK LINE MARKS THE END OF WHAT'S TO BE PROCESSED, ANY BLANK LINE
SHOULD
// GO AFTER THE REQUIRES BELOW.
//
//= require_self
//= require jquery_lib
//= require jquery_ujs
//= require jquery_ui
//= require libs
//= require utils
//= require nav_list
//= require theme
//= require stat
//= require menu
//= require sort
// require_tree .

```

```
E:\workspace\payroll_rails\app\assets\javascripts\menu.js
```

```
var menu = ( function() {

    function menu_click() {
        $('#menu a').removeClass('menu_active');
        $(this).addClass('menu_active');
    }

    function get(url, func) {
        utils.clear_dialogs();
        $('#contentcolumn div.innertube').load(url, func);
        return false;
    }

    function init() {
        $('#menu').accordion({
            autoHeight : false,
            animated : false
        });
        $('#menu a').click(menu_click);
    }

    return {
        get : get,
        init : init
    };
}());
```

```
E:\workspace\payroll_rails\app\assets\javascripts\nav_list.js
```

```
var nav_list = ( function() {
    var config = {
        list_url : '',
        list_func : null,
        del_func : null,
        save_func : null,
        search_param_func : null
    };

    /**
     * @public
     * This function shows the list, and reset back the page number to 1.
     */
    function show_list() {
        $('#id_display').data('pgnum', 1);
        utils.stop_filter_timer();
        update_list();
    }

    /**
     * @public
     * This function reloads the list by using the same navigation parameters such as the
     * display size, current page no., and search parameters.
     */
    function update_list() {
        var pgszie = $('#id_display').val();
        var pgnum = $('#id_display').data('pgnum');
        var sort = get_sort();
        var param = ($isFunction(config.search_param_func) ? config.search_param_func() :
        get_search_param());
        param['pgnum'] = pgnum;
        param['pgsize'] = pgszie;

        if (sort != null) {
            param['sortcolumn'] = sort['column'];
            param['sortdir'] = sort['dir'];
        }

        $.post(config.list_url, param, function(result) {
            $('#right_box').html(result);
            init_navigate();
        });
    }

    /**
     * @private
     */
```

```

* This function navigate the list to the previous page.
*/
function go_prev() {
    var val = $('#id_pg').val();
    var arr = val.split(',');
    var d = $('#id_display').val();
    $('#id_display').data('pgnum', arr[2]);
    update_list();
}

/**
 * @private
 * This function navigate the list to the next page.
*/
function go_next() {
    var val = $('#id_pg').val();
    var arr = val.split(',');
    var d = $('#id_display').val();
    $('#id_display').data('pgnum', arr[3]);
    update_list();
}

/**
 * @private
 * This function navigate the list to the specified page.
*/
function go_page() {
    var pg = $('#id_pagenum').val();
    var page = 1;
    if ($.trim(pg) != '' && $.isNumeric(pg))
        page = parseInt(pg);

    $('#id_display').data('pgnum', page);
    update_list();
}

/**
 * @public
 * This function handles the keypress event on the search textbox.
 * It checks for <enter> key.
*/
function query_keypress(evt) {
    if (evt.keyCode == '13') {
        evt.preventDefault();
        evt.stopPropagation();
        show_list();
    }
}

/**
 * @public
 * This function handles the keyup event on the search textbox.
 * It checks for <enter> key.
*/
function query_keyup(evt) {
    if (evt.keyCode != '13')
        utils.countdown_filter(show_list);
}

/**
 * @public
 * This function sets the item text,
 * i.e 1 to 2 of 2
 * @param arg The text.
*/
function set_item_msg(arg) {
    $('.item_display').text(arg);
}

/**
 * @public
 * This function sets the sort info.
 * @param s The sort info in {}.
*/
function set_sort(s) {
    $('#id_display').data('sort', s);
    update_list();
}

```

```

}

/***
 * @private
 * This function gets the current sort info in {}.
 * @return The sort info in {}.
 */
function get_sort() {
    var s = $('#id_display').data('sort');
    return s;
}

/***
 * @private
 * This function initialize the navigation elements after the list has been loaded.
 * The initialization includes :<br>
 * 1. sets the page no. to 1<br>
 * 2. enable/disable the next/prev buttons<br>
 * 3. initialize the edit/delete/clone buttons
*/
function init_navigate() {
    $('#id_display').data('pgnum', 1);
    if ($.isFunction(config.list_func))
        config.list_func();

    var val = $('#id_pg').val();
    var arr = val.split(',');
    if (arr[0] == '0') {
        utils.set_disabled('#id_prev', 1, null);
    }

    else {
        utils.set_disabled('#id_prev', 0, go_prev);
    }

    if (arr[1] == '0') {
        utils.set_disabled('#id_next', 1, null);
    }

    else {
        utils.set_disabled('#id_next', 0, go_next);
    }

    if (arr[0] == '1' || arr[1] == '1') {
        utils.set_disabled('#id_go', 0, go_page);
    }

    else if (arr[0] == '0' && arr[1] == '0') {
        utils.set_disabled('#id_go', 1, null);
    }

    if ($.isFunction(config.del_func)) {
        if ($.list_table)[0] != null)
            utils.set_disabled('#id_delete', 0, config.del_func);

        else
            utils.set_disabled('#id_delete', 1, null);
    }

    if ($.isFunction(config.save_func)) {
        if ($.list_table)[0] != null)
            utils.set_disabled('#id_save', 0, config.save_func);

        else
            utils.set_disabled('#id_save', 1, null);
    }

    $('#id_pagenum').val(arr[7]);

    sort.init_sort($('#hd_' + utils.safe_replace(arr[5], '.', '-')), arr[6]);
    set_item_msg(arr[4]);
}

/***
 * @private
 * This function returns the search parameters in {}.
 * @return The search parameters in {}.
*/

```

```

        */
function get_search_param() {
    var id_selection = $('#id_selection');
    var keyword = $('#id_query').val();
    var param = {};
    if (id_selection[0] != null)
        param['find'] = id_selection.val();

    if (param['find'] == '0' && keyword == '')
        delete param['find']

    else
        param['keyword'] = keyword;

    return param;
}

function init() {
    init_navigate();
}

return {
    init : init,
    config : config,
    show_list : show_list,
    update_list : update_list,
    query_keypress : query_keypress,
    query_keyup : query_keyup,
    set_item_msg : set_item_msg,
    set_sort : set_sort
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\sort.js

```

var sort = ( function() {
    var icon = 'ui-icon';
    var icon_asc = 'ui-icon-triangle-1-n';
    var icon_desc = 'ui-icon-triangle-1-s';

    /**
     * @public
     * This function sets the sort icon in the selected column header.
     * @param o The column header selected using $.
     */
    function set_sort_css(o) {
        var c = o.children().last();
        var id = o.attr('id');
        id = utils.getitemid(id);
        var column = utils.safe_replace(id, '-', '.');
        var sort = {
            column : column,
            dir : 'ASC'
        };

        var isasc = c.hasClass(icon_asc);
        $('.sorticon').removeClass(icon + ' ' + icon_asc + ' ' + icon_desc);

        if (isasc)
            sort['dir'] = 'DESC';

        init_sort(o, sort.dir);
    }

    return sort;
}

/**
 * @public
 * This function sets the corresponding sort icon in the selected column header.
 * @param o The column header selected using $.
 * @param dir The sort direction ASC/DESC.
 */
function init_sort(o, dir) {
    var c = (dir == 'ASC' ? icon_asc : icon_desc);
    o.children().last().addClass(icon + ' ' + c);
}

```

```

    return {
      set_sort_css : set_sort_css,
      init_sort : init_sort
    };
  }());
}

E:\workspace\payroll_rails\app\assets\javascripts\stat.js

var stat = ( function() {
  var info = 'ui-state-highlight ui-corner-all';
  var info_icon = 'ui-icon ui-icon-check';
  var error = 'ui-state-error ui-corner-all';
  var error_icon = 'ui-icon ui-icon-alert';
  var status_timer = null;

  function show_status(arg, msg) {
    clearTimeout(status_timer);
    var id = '#status-panel';
    var p = '#status_msg';
    var opt = {};
    if (arg == 0) {
      set_info_class(id);
      $(p).html(msg);
      $(id).show('drop', opt, 500, function() {
        status_timer = setTimeout(function() {
          $(id).hide('scale', opt, 500, null);
        }, 5000);
      });
    }
    else {
      set_error_class(id);
      $(p).html(msg);
      $(id).show();
    }
  }

  function set_info_class(id) {
    var div = id + '_outer';
    var span = id + '_inner';
    remove_status_class(div, span);
    $(div).addClass(info);
    $(span).addClass(info_icon);
  }

  function set_error_class(id) {
    var div = id + '_outer';
    var span = id + '_inner';
    remove_status_class(div, span);
    $(div).addClass(error);
    $(span).addClass(error_icon);
  }

  function remove_status_class(div, span) {
    if ($(div).hasClass(info)) {
      $(div).removeClass(info);
      $(span).removeClass(info_icon);
    }
    if ($(div).hasClass(error)) {
      $(div).removeClass(error);
      $(span).removeClass(error_icon);
    }
  }

  return {
    show_status : show_status
  };
}());
}

```

```

E:\workspace\payroll_rails\app\assets\javascripts\theme.js

var theme = ( function() {
  var default_theme = 'darkhive';

  function init() {
    var select_close = $('#theme_option .cancelicon');
    select_close.click(function() {
      $('#theme_option').slideUp();
    });
  }

  $.themes.init({
    themes : ['blitzer', 'darkhive', 'trontastic', 'humanity'],
    defaultTheme : default_theme,
    onSelect : reload_IE
  });
  $('#theme_body').themes();
  $("#main_options").click(show_options);
  utils.bind_hover($("#main_options"));
}

function show_options() {
  $('#theme_option').slideToggle();
}

function current_theme() {
  return $.themes.currentTheme;
}

function reload_IE(id, display, url) {
}

return {
  init : init,
  current_theme : current_theme
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\user.js

```

// This is a manifest file that'll be compiled into application.js, which will include all the
files
// listed below.
//
// Any JavaScript/Coffee file within this directory, lib/assets/javascripts,
vendor/assets/javascripts,
// or vendor/assets/javascripts of plugins, if any, can be referenced here using a relative
path.
//
// It's not advisable to add code directly here, but if you do, it'll appear at the bottom of
the
// the compiled file.
//
// WARNING: THE FIRST BLANK LINE MARKS THE END OF WHAT'S TO BE PROCESSED, ANY BLANK LINE
SHOULD
// GO AFTER THE REQUIRES BELOW.
//
//= require application
//= require user/base
//= require user/info
//= require user/contact
//= require user/job
//= require user/salary
//= require user/qualification
//= require user/otccht
//= require user/hpccht
//= require user/wkcht
//= require user/payslip
// require_tree .

```

```

E:\workspace\payroll_rails\app\assets\javascripts\utils.js

var utils = ( function() {
    var typing_timer = null;
    var done_typing_interval = 2000;
    var date_format = 'dd-mm-yy';

    /**
     * @public
     * This function makes the selected id to appear as popup dialog to show alert message.
     * @param id The element id.
     */
    function init_alert_dialog(id) {
        $(id).dialog({
            autoOpen : false,
            modal : true,
            buttons : {
                OK : function() {
                    $(this).dialog('close');
                }
            }
        });
    }

    /**
     * @public
     * This function makes the element error-dialog to appear as popup dialog.
     * It also attach the ajaxError event to monitor ajax error.
     */
    function init_server_error_dialog() {
        $(document).ajaxError(function(evt, jqXHR, ajaxOptions, errorThrown) {
            show_error_dialog(jqXHR.responseText);
        });
        $('#error-dialog').dialog({
            autoOpen : false,
            modal : true,
            width : 700,
            height : 500,
            buttons : {
                OK : function() {
                    $(this).dialog('close');
                }
            }
        });
    }

    /**
     * @public
     * This function monitor the progress of an ajax request by showing the progress Loading
     ...
     */
    function init_progress() {
        $(document).ajaxSend(function(evt, jqXHR, ajaxOptions) {
            $('#progress_status').show();
        });
        $(document).ajaxComplete(function() {
            $('#progress_status').hide();
        });
    }

    /**
     * @public
     * This function removes a dialog specified by the dialog id.
     * @param id The dialog id.
     */
    function remove_dialog(id) {
        var o = $(id);
        try {
            o.dialog('destroy');
        }

        catch (e) {}

        o.remove();
    }

    /**

```

```

    * @public
    * This function removes all the dialogs from the DOM.
    */
function clear_dialogs() {
    remove_dialog("div[id^='dialog-message']");
    remove_dialog("div[id^='dialog']");
}

/**
 * @public
 * This function shows a message in a popup dialog.
 * @param arg The parameter to determine whether to show html (1) or plain text (2)
message.
 * @param msg The message.
 */
function show_dialog(arg, msg) {
    if (arg == 1)
        $('#dialog_msg').html(msg);

    else
        $('#dialog_msg').text(msg);

    $('#dialog-message').dialog('open');
}

/**
 * @public
 * This function shows the error message in a popup dialog.
 * @param msg The error message.
 */
function show_error_dialog(msg) {
    $('#error_dialog').html(msg);
    $('#error_dialog_style').remove();
    $('#error-dialog').dialog('open');
}

/**
 * @public
 * This function binds the hover event to the selected object.
 * @param selector The object selected using $
*/
function bind_hover(selector) {
    selector.hover(function() {
        $(this).toggleClass('ui-state-hover');
    }, function() {
        $(this).toggleClass('ui-state-hover');
    });
}

/**
 * @public
 * This function binds the hover event to the selected list.
 * @param selector The object selected using $
*/
function bind_hoverlist(selector) {
    selector.hover(function() {
        $(this).toggleClass('ui-state-highlight hover');
    }, function() {
        $(this).toggleClass('ui-state-highlight hover');
    });
}

/**
 * @public
 * This function executes the specified function when the typing interval has elapsed.
 * @param func The function to be executed.
 */
function countdown_filter(func) {
    stop_filter_timer();
    typing_timer = setTimeout(func, done_typing_interval);
}

/**
 * @public
 * This function stops the typing timer.
 */
function stop_filter_timer() {

```

```

        clearTimeout(typing_timer);
    }

    /**
     * @public
     * This function returns the id from a given element id.
     * e.g if the element id is item_1,
     * the id will be 1
     * @param arg The element id.
     * @return The id.
    */
    function get_itemid(arg) {
        var i = arg.indexOf('_');
        if (i >= 0) {
            var s = arg.substr(i + 1);
            return s;
        }

        return null;
    }

    /**
     * @public
     * This function enable/disable an element.
     * @param id The element id.
     * @param arg The parameter to enable (0) or disable (1) the element.
     * @param handler The function to be attached to the click event.
    */
    function set_disabled(id, arg, handler) {
        var o = $(id);
        o.unbind('click');
        o.unbind('mouseenter');
        o.unbind('mouseleave');
        if (arg == 1) {
            o.attr('disabled', 'disabled');
            o.removeClass('hover ui-state-hover');
            o.addClass('ui-state-disabled');
        }
        else {
            oremoveAttr('disabled');
            o.removeClass('ui-state-disabled');
            o.addClass('hover');
            o.click(handler);
            bind_hover(o);
        }
    }

    /**
     * @public
     * This function replace a string.
     * @param s The string to be replaced.
     * @param a The string to be replaced.
     * @param b The replacement string.
     * @return The replaced string.
    */
    function safe_replace(s, a, b) {
        if (s == null)
            return s;

        return s.replace(a, b);
    }

    /**
     * @public
     * This function returns the options for datepicker.
    */
    function date_opt() {
        return {
            dateFormat : date_format,
            changeMonth : true,
            changeYear : true
        };
    }

    return {
        init_alert_dialog : init_alert_dialog,

```

```

        init_server_error_dialog : init_server_error_dialog,
        init_progress : init_progress,
        remove_dialog : remove_dialog,
        clear_dialogs : clear_dialogs,
        show_dialog : show_dialog,
        show_error_dialog : show_error_dialog,
        bind_hover : bind_hover,
        bind_hoverlist : bind_hoverlist,
        countdown_filter : countdown_filter,
        stop_filter_timer : stop_filter_timer,
        get_itemid : get_itemid,
        set_disabled : set_disabled,
        safe_replace : safe_replace,
        date_format : date_format,
        date_opt : date_opt
    );
};

}());

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\att.js

```

var att = ( function() {
    var url = {
        list : '/admin/att/list/'
    };

    function get_search_param() {
        var param = {
            work_date : $('#id_work_date').val(),
            employee : $('#id_employee').val()
        };
        return param;
    }

    function sort_list() {
        var s = sort.set_sort_css($(this));
        nav_list.set_sort(s);
    }

    function init_list() {
        $('.sortheader').click(sort_list);
    }

    function init() {
        $('.date_input').datepicker(utils.date_opt());
        $('#id_find').click(nav_list.show_list);
        $('#id_display').change(nav_list.show_list);
        $('#id_employee').tooltip({track: true});
        utils.init_alert_dialog('#dialog-message');
        utils.bind_hover($('#id_find'));
        nav_list.config.list_url = url.list;
        nav_list.config.list_func = init_list;
        nav_list.config.search_param_func = get_search_param;
        nav_list.init();
    }

    function load() {
        return menu.get('/admin/att/', init);
    }

    return {
        load : load
    };
}());

```

```
E:\workspace\payroll_rails\app\assets\javascripts\admin\base.js

var app = ( function() {

    function init() {
        menu.init();
        utils.init_progress();
        utils.init_server_error_dialog();
        theme.init();
        utils.bind_hover($('#logout'));
        $('#menu_user').addClass('menu_active');
        user.load();
    }

    return {
        init : init
    };
}());

$(app.init);
```

E:\workspace\payroll_rails\app\assets\javascripts\admin\dept.js

```
var dept = ( function() {
    var url = {
        add : '/admin/dept/new/',
        create : '/admin/dept/create/',
        edit : '/admin/dept/edit/',
        update : '/admin/dept/update/',
        del : '/admin/dept/delete/',
        list : '/admin/dept/list/'
    };

    var popup_dialog_opt = null;

    function init_ui_opt() {
        popup_dialog_opt = {
            autoOpen : false,
            width : 350,
            resizable : false,
            draggable : true,
            modal : false,
            stack : true,
            zIndex : 1000
        };
    }

    function show_form() {
        $('#dialog_add_body').load(url.add, function() {
            $('.save_button.save').click(func_save);
            $('.save_button.cancel').click(func_cancel_add);
            $('#add-form').tooltip({track: true});
            utils.bind_hover($('.save_button'));
            $('#dialog-add').dialog('open');
        });
    }

    function save_success() {
        func_cancel_add();
        nav_list.show_list();
    }

    function update_success() {
        func_cancel_edit();
        nav_list.show_list();
    }

    function func_cancel_add() {
        $('#dialog-add').dialog('close');
        return false;
    }

    function func_cancel_edit() {
        $('#dialog-edit').dialog('close');
        return false;
    }
}
```

```

function func_save() {
    var data = get_data('add');
    $('#add-form input').next().remove();
    $.post(url.create, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            save_success();
        }
        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    $('#add-form input[name=' + e + "']").after(h);
                }
            }
        }
        else
            utils.show_dialog(2, result);
    });
    return false;
}

function func_edit(id) {
    if (!id)
        return false;

    id = utils.get_itemid(id);
    $('#dialog-edit-body').load(url.edit + id, function() {
        $('.save_button.save').click(function() {
            return func_update(id);
        });
        $('.save_button.cancel').click(func_cancel_edit);
        $('#edit-form').tooltip({track: true});
        utils.bind_hover($('.save_button'));
        $('#dialog-edit').dialog('open');
    });
    return false;
}

function func_update(id) {
    var data = get_data('edit');
    $('#edit-form input').next().remove();
    $.post(url.update + id, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            update_success();
        }
        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    $('#edit-form input[name=' + e + "']").after(h);
                }
            }
        }
        else
    });
}

```

```

        utils.show_dialog(2, result);
    });

    return false;
}

function func_delete() {
    var a = $('.chk:checked');
    if (a.length < 1) {
        utils.show_dialog(2, 'Please select record(s).');
        return;
    }

    var l = [];
    var trlist = [];
    a.each(function(idx, elm) {
        var id = $(this).parent().parent().attr('id');
        trlist.push('#' + id);
        id = utils.get_itemid(id);
        l.push(id);
    });

    var val = $('#id_pg').val();
    var arr = val.split(',');
    var currpg = parseInt(arr[3], 10);
    --currpg;
    var pgsize = $('#id_display').val();
    var search_by = $('#id_selection').val();
    var keyword = $('#id_query').val();
    var data = {
        'id[]' : l,
        pgnum : currpg,
        pgsize : pgsize,
        find : search_by,
        keyword : keyword
    };

    $.post(url.del, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            nav_list.set_item_msg(result.itemscount);
            var tr = $(trlist.join(','));
            tr.remove();
            delete tr;
            if ($('.list_table tbody tr').length < 1) {
                $('.list_table').remove();
                utils.set_disabled('#id_delete', 1, null);
            }
        }
    });
}

function select_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chk').attr('checked', 'checked');

    else
        $('.chk').removeAttr('checked');
}

function get_data(t) {
    var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

    var data = {
        name : form.find('#id_name').val()
    };

    return data;
}

function sort_list() {
    var s = sort.set_sort_css($(this));
    nav_list.set_sort(s);
}

function init_list() {

```

```

$('.hdchk').click(select_all);
utils.bind_hoverlist($('.list_table tbody tr'));
$('.list_table tbody').selectable({
  selected : function(evt, ui) {
    var id = ui.selected.id;
    func_edit(id);
  }
});
$('.sortheadr').click(sort_list);
}

function init() {
  init_ui_opt();
  $('#id_add').click(show_form);
  $('#id_find').click(nav_list.show_list);
  $('#id_display').change(nav_list.show_list);
  $('#id_query').keypress(nav_list.query_KeyPress);
  $('#id_query').keyup(nav_list.query_Keyup);
  $('#id_query').tooltip({track: true});
  $('#dialog-add').dialog({popup_dialog_opt});
  $('#dialog-edit').dialog({popup_dialog_opt});
  utils.init_alert_dialog('#dialog-message');
  utils.bind_hover($('#id_add,#id_delete,#id_find'));
  nav_list.config.list_url = url.list;
  nav_list.config.list_func = init_list;
  nav_list.config.del_func = func_delete;
  nav_list.init();
}

function load() {
  return menu.get('/admin/dept/', init);
}

return {
  load : load
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\designation.js

```

var designation = ( function() {
  var url = {
    add : '/admin/designation/new/',
    create : '/admin/designation/create/',
    edit : '/admin/designation/edit/',
    update : '/admin/designation/update/',
    del : '/admin/designation/delete/',
    list : '/admin/designation/list/'
  };

  var popup_dialog_opt = null;

  function init_ui_opt() {
    popup_dialog_opt = {
      autoOpen : false,
      width : 350,
      resizable : false,
      draggable : true,
      modal : false,
      stack : true,
      zIndex : 1000
    };
  }

  function show_form() {
    $('#dialog_add_body').load(url.add, function() {
      $('.save_button.save').click(func_save);
      $('.save_button.cancel').click(func_cancel_add);
      $('#add-form').tooltip({track: true});
      utils.bind_hover($('.save_button'));
      $('#dialog-add').dialog('open');
    });
  }

  function save_success() {
    func_cancel_add();
    nav_list.show_list();
  }
}

```

```

}

function update_success() {
    func_cancel_edit();
    nav_list.show_list();
}

function func_cancel_add() {
    $('#dialog-add').dialog('close');
    return false;
}

function func_cancel_edit() {
    $('#dialog-edit').dialog('close');
    return false;
}

function func_save() {
    var data = get_data('add');
    $('#add-form input').next().remove();
    $.post(url.create, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            save_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    $('#add-form input[name=' + e + "']").after(h);
                }
            }
        }

        else
            utils.show_dialog(2, result);
    });
}

return false;
}

function func_edit(id) {
    if (!id)
        return false;

    id = utils.get_itemid(id);
    $('#dialog_edit_body').load(url.edit + id, function() {
        $('.save_button.save').click(function() {
            return func_update(id);
        });

        $('.save_button.cancel').click(func_cancel_edit);
        $('#edit-form').tooltip({track: true});
        utils.bind_hover($('.save_button'));
        $('#dialog-edit').dialog('open');
    });
    return false;
}

function func_update(id) {
    var data = get_data('edit');
    $('#edit-form input').next().remove();
    $.post(url.update + id, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            update_success();
        }

        else if (result.error == 1) {

```

```

        for (var e in result.errors) {
            var d = $('#error_' + e).get(0);
            if (!d) {
                var o = {
                    field : e,
                    msg : result.errors[e][0]
                };
                var h = new EJS({
                    url : '/assets/tpl/label_error.html',
                    ext : '.html'
                }).render(o);
                $("#edit-form input[name='" + e + "']").after(h);
            }
        }
    }

    else
        utils.show_dialog(2, result);
});

return false;
}

function func_delete() {
    var a = $('.chk:checked');
    if (a.length < 1) {
        utils.show_dialog(2, 'Please select record(s).');
        return;
    }

    var l = [];
    var trlist = [];
    a.each(function(idx, elm) {
        var id = $(this).parent().parent().attr('id');
        trlist.push('#' + id);
        id = utils.get_itemid(id);
        l.push(id);
    });

    var val = $('#id_pg').val();
    var arr = val.split(',');
    var currpg = parseInt(arr[3], 10);
    --currpg;
    var pgsize = $('#id_display').val();
    var search_by = $('#id_selection').val();
    var keyword = $('#id_query').val();
    var data = {
        'id[]' : l,
        pgnum : currpg,
        pgsize : pgsize,
        find : search_by,
        keyword : keyword
    };

    $.post(url.del, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            nav_list.set_item_msg(result.itemscount);
            var tr = $(trlist.join(','));
            tr.remove();
            delete tr;
            if ($('#list_table tbody tr').length < 1) {
                $('#list_table').remove();
                utils.set_disabled('#id_delete', 1, null);
            }
        }
    });
}

function select_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chk').attr('checked', 'checked');

    else
        $('.chk').removeAttr('checked');
}

```

```

function get_data(t) {
  var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

  var data = {
    title : form.find('#id_title').val(),
    desc : form.find('#id_desc').val(),
    note : form.find('#id_note').val()
  };

  return data;
}

function sort_list() {
  var s = sort.set_sort_css($(this));
  nav_list.set_sort(s);
}

function init_list() {
  $('.hdchk').click(select_all);
  utils.bind_hoverlist($('.list_table tbody tr'));
  $('.list_table tbody').selectable({
    selected : function(evt, ui) {
      var id = ui.selected.id;
      func_edit(id);
    }
  });
  $('.sorthead').click(sort_list);
}

function init() {
  init_ui_opt();
  $('#id_add').click(show_form);
  $('#id_find').click(nav_list.show_list);
  $('#id_display,#id_selection').change(nav_list.show_list);
  $('#id_query').keypress(nav_list.query_KeyPress);
  $('#id_query').keyup(nav_list.query_Keyup);
  $('#id_query').tooltip({track: true});
  $('#dialog-add').dialog(popup_dialog_opt);
  $('#dialog-edit').dialog(popup_dialog_opt);
  utils.init_alert_dialog('#dialog-message');
  utils.bind_hover($('#id_add,#id_delete,#id_find'));
  nav_list.config.list_url = url.list;
  nav_list.config.list_func = init_list;
  nav_list.config.del_func = func_delete;
  nav_list.init();
}

function load() {
  return menu.get('/admin/designation/', init);
}

return {
  load : load
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\emp.js

```

var emp = ( function() {
  var url = {
    add : '/admin/employee/new/',
    create : '/admin/employee/create/',
    edit : '/admin/employee/edit/',
    update : '/admin/employee/update/',
    del : '/admin/employee/delete/',
    list : '/admin/employee/list/'
  };

  var popup_dialog_opt = null;

  function init_ui_opt() {
    popup_dialog_opt = {
      autoOpen : false,
      width : 750,
      resizable : false,
      draggable : true,

```

```

        modal : false,
        stack : true,
        zIndex : 1000
    );
}

function show_form() {
    $('#dialog_add_body').load(url.add, function() {
        $(this).find('#tabs').tabs();
        $('input.date_input').datepicker(utils.date_opt());
        $('.save_button.save').click(func_save);
        $('.save_button.cancel').click(func_cancel_add);
        $('#add-form').tooltip({track: truefunction save_success() {
    func_cancel_add();
    nav_list.show_list();
}

function update_success() {
    func_cancel_edit();
    nav_list.show_list();
}

function func_cancel_add() {
    $('#dialog-add').dialog('close');
    return false;
}

function func_cancel_edit() {
    $('#dialog-edit').dialog('close');
    return false;
}

function func_save() {
    var data = get_data('add');
    data = get_data_contact('add', data);
    data = get_data_job('add', data);
    data = get_data_salary('add', data);
    data = get_data_qualification('add', data);
    $('#add-form input[type="text"]').next().remove();
    $('#add-form select').next().remove();
    $('#add-form').find('#lb_gender_f').next().remove();
    $.post(url.create, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            save_success();
        }

        else if (result.error == 1) {
            if (result.employee.error == 1) {
                for (var e in result.employee.errors) {
                    var d = $('#add-form #form-employee #error_' + e).get(0);
                    if (!d) {
                        var o = {
                            field : e,
                            msg : result.employee.errors[e][0]
                        };
                        var h = new EJS({
                            url : '/assets/tpl/label_error.html',
                            ext : '.html'
                        }).render(o);
                        if (e == 'gender')
                            $('#add-form #form-employee #lb_gender_f').after(h);

                        else if (e == 'marital_status')
                            $('#add-form #form-employee #id_marital_status').after(h);

                        else
                            $('#add-form #form-employee input[name=' + e + "']").after(h);
                }
            }
        }
    });
}

```

```

    if (result.employee_contact.error == 1) {
        for (var e in result.employee_contact.errors) {
            var d = $('#add-form #form-employee-contact #error_' + e).get(0);
            if (!d) {
                var o = {
                    field : e,
                    msg : result.employee_contact.errors[e][0]
                };
                var h = new EJS({
                    url : '/assets/tpl/label_error_inline.html',
                    ext : '.html'
                }).render(o);
                $("#" + "#add-form #form-employee-contact input[name='"
                    + e + "']").after(h);
            }
        }
    }

    if (result.employee_job.error == 1) {
        for (var e in result.employee_job.errors) {
            var d = $('#add-form #form-employee-job #error_' + e).get(0);
            if (!d) {
                var o = {
                    field : e,
                    msg : result.employee_job.errors[e][0]
                };
                var h = new EJS({
                    url : '/assets/tpl/label_error_inline.html',
                    ext : '.html'
                }).render(o);
                if (e == 'designation_id' || e == 'department_id' || e ==
                    'employment_status_id' || e == 'job_category_id')
                    $('#add-form #form-employee-job #id_' + e).after(h);

                else
                    $("#" + "#add-form #form-employee-job input[name='"
                        + e + "']").after(h);
            }
        }
    }

    if (result.employee_salary.error == 1) {
        for (var e in result.employee_salary.errors) {
            if (!d) {
                var o = {
                    field : e,
                    msg : result.employee_salary.errors[e][0]
                };
                var h = new EJS({
                    url : '/assets/tpl/label_error_inline.html',
                    ext : '.html'
                }).render(o);
                $("#" + "#add-form #form-employee-salary input[name='"
                    + e + "']").after(h);
            }
        }
    }

    if (result.employee_qualification.error == 1) {
        for (var e in result.employee_qualification.errors) {
            if (!d) {
                var o = {
                    field : e,
                    msg : result.employee_qualification.errors[e][0]
                };
                var h = new EJS({
                    url : '/assets/tpl/label_error_inline.html',
                    ext : '.html'
                }).render(o);
                if (e == 'level')
                    $('#add-form #form-employee-qualification #id_' + e).after(h);

                else
                    $("#" + "#add-form #form-employee-qualification input[name='"
                        + e + "']").after(h);
            }
        }
    }
}

```

```

        utils.show_dialog(1, 'There are validation errors in the form. Please complete all
the required fields.');
    }

    else
        utils.show_dialog(2, result);
    });

    return false;
}

function func_edit(id) {
    if (!id)
        return false;

    id = utils.get_itemid(id);
    $('#dialog-edit-body').load(url.edit + id, function() {
        $(this).find('#tabs').tabs();
        $('.date_input').datepicker(utils.date_opt());
        $('.save_button.save').click(function() {
            return func_update(id);
        });
        $('.save_button.cancel').click(func_cancel_edit);
        $('#edit-form').tooltip({track: true});
        utils.bind_hover($('.save_button'));
        $('#dialog-edit').dialog('open');
    });
    return false;
}

function func_update(id) {
    var data = get_data('edit');
    data = get_data_contact('edit', data);
    data = get_data_job('edit', data);
    data = get_data_salary('edit', data);
    data = get_data_qualification('edit', data);
    $('#edit-form input[type="text"]').next().remove();
    $('#edit-form select').next().remove();
    $('#edit-form').find('#lb_gender_f').next().remove();
    $.post(url.update + id, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            update_success();
        }

        else if (result.error == 1) {
            if (result.employee.error == 1) {
                for (var e in result.employee.errors) {
                    var d = $('#edit-form #form-employee #error_' + e).get(0);
                    if (!d) {
                        var o = {
                            field : e,
                            msg : result.employee.errors[e][0]
                        };
                        var h = new EJS({
                            url : '/assets/tpl/label_error.html',
                            ext : '.html'
                        }).render(o);
                        if (e == 'gender')
                            $('#edit-form #form-employee #lb_gender_f').after(h);

                        else if (e == 'marital_status')
                            $('#edit-form #form-employee #id_marital_status').after(h);

                        else
                            $('#edit-form #form-employee input[name=' + e + ']').after(h);
                    }
                }
            }

            if (result.employee_contact.error == 1) {
                for (var e in result.employee_contact.errors) {
                    var d = $('#edit-form #form-employee-contact #error_' + e).get(0);
                    if (!d) {
                        var o = {
                            field : e,
                            msg : result.employee_contact.errors[e][0]
                        };
                    }
                }
            }
        }
    });
}

```

```

    };
    var h = new EJS({
        url : '/assets/tpl/label_error_inline.html',
        ext : '.html'
    }).render(o);
    $("#" + edit_form_id + " #form-employee-contact input[name='"
        + e + "']").after(h);
}
}

if (result.employee_job.error == 1) {
    for (var e in result.employee_job.errors) {
        var d = $("#" + add_form_id + " #form-employee-job #error_"
            + e).get(0);
        if (!d) {
            var o = {
                field : e,
                msg : result.employee_job.errors[e][0]
            };
            var h = new EJS({
                url : '/assets/tpl/label_error_inline.html',
                ext : '.html'
            }).render(o);
            if (e == 'designation_id' || e == 'department_id' || e ==
                'employment_status_id' || e == 'job_category_id')
                $("#" + edit_form_id + " #form-employee-job #id_"
                    + e).after(h);

            else
                $("#" + edit_form_id + " #form-employee-job input[name='"
                    + e + "']").after(h);
        }
    }
}

if (result.employee_salary.error == 1) {
    for (var e in result.employee_salary.errors) {
        if (!d) {
            var o = {
                field : e,
                msg : result.employee_salary.errors[e][0]
            };
            var h = new EJS({
                url : '/assets/tpl/label_error_inline.html',
                ext : '.html'
            }).render(o);
            $("#" + edit_form_id + " #form-employee-salary input[name='"
                + e + "']").after(h);
        }
    }
}

if (result.employee_qualification.error == 1) {
    for (var e in result.employee_qualification.errors) {
        if (!d) {
            var o = {
                field : e,
                msg : result.employee_qualification.errors[e][0]
            };
            var h = new EJS({
                url : '/assets/tpl/label_error_inline.html',
                ext : '.html'
            }).render(o);
            if (e == 'level')
                $("#" + edit_form_id + " #form-employee-qualification #id_"
                    + e).after(h);

            else
                $("#" + edit_form_id + " #form-employee-qualification input[name='"
                    + e + "']").after(h);
        }
    }
}

else
    utils.show_dialog(2, result);
});

return false;
}

```

```

function func_delete() {
    var a = $('.chk:checked');
    if (a.length < 1) {
        utils.show_dialog(2, 'Please select record(s).');
        return;
    }

    var l = [];
    var trlist = [];
    a.each(function(idx, elm) {
        var id = $(this).parent().parent().attr('id');
        trlist.push('#' + id);
        id = utils.get_itemid(id);
        l.push(id);
    });

    var val = $('#id_pg').val();
    var arr = val.split(',');
    var currpg = parseInt(arr[3], 10);
    --currpg;
    var pgsize = $('#id_display').val();
    var data = get_search_param();
    data['id[]'] = l;
    data['pgnum'] = currpg;
    data['pgsize'] = pgsize;

    $.post(url.del, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            nav_list.set_item_msg(result.itemscount);
            var tr = $(trlist.join(','));
            tr.remove();
            delete tr;
            if ($('#list_table tbody tr')[0] == null) {
                $('#list_table').remove();
                utils.set_disabled('#id_delete', 1, null);
            }
        }
    });
}

function select_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chk').attr('checked', 'checked');

    else
        $('.chk').removeAttr('checked');
}

function get_data(t) {
    var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

    var data = {
        staff_id : form.find('#id_staff_id').val(),
        first_name : form.find('#id_first_name').val(),
        middle_name : form.find('#id_middle_name').val(),
        last_name : form.find('#id_last_name').val(),
        new_ic : form.find('#id_new_ic').val(),
        old_ic : form.find('#id_old_ic').val(),
        passport_no : form.find('#id_passport_no').val(),
        gender : form.find('input[name="gender"]:checked').val(),
        marital_status : form.find('#id_marital_status').val(),
        nationality : form.find('#id_nationality').val(),
        dob : form.find('#id_dob').val(),
        place_of_birth : form.find('#id_place_of_birth').val(),
        race : form.find('#id_race').val(),
        religion : form.find('#id_religion').val(),
        is_bumi : form.find('#id_is_bumi').prop('checked'),
        user_id : form.find('#id_user_id').val()
    };

    return { employee : data };
}

function get_data_contact(t, o) {
    var f = (t == 'add' ? $('#add-form') : $('#edit-form'));

```

```

var form = f.find('#form-employee-contact');

var data = {
    address_1 : form.find('#id_address_1').val(),
    address_2 : form.find('#id_address_2').val(),
    address_3 : form.find('#id_address_3').val(),
    city : form.find('#id_city').val(),
    state : form.find('#id_state').val(),
    postcode : form.find('#id_postcode').val(),
    country : form.find('#id_country').val(),
    home_phone : form.find('#id_home_phone').val(),
    mobile_phone : form.find('#id_mobile_phone').val(),
    work_email : form.find('#id_work_email').val(),
    other_email : form.find('#id_other_email').val()
};

o['employee_contact'] = data;

return o;
}

function get_data_job(t, o) {
    var f = (t == 'add' ? $('#add-form') : $('#edit-form'));
    var form = f.find('#form-employee-job');

    var data = {
        designation_id : form.find('#id_designation_id').val(),
        department_id : form.find('#id_department_id').val(),
        employment_status_id : form.find('#id_employment_status_id').val(),
        job_category_id : form.find('#id_job_category_id').val(),
        join_date : form.find('#id_join_date').val(),
        confirm_date : form.find('#id_confirm_date').val()
    };

    o['employee_job'] = data;

    return o;
}

function get_data_salary(t, o) {
    var f = (t == 'add' ? $('#add-form') : $('#edit-form'));
    var form = f.find('#form-employee-salary');

    var data = {
        salary : form.find('#id_salary').val(),
        allowance : form.find('#id_allowance').val(),
        epf : form.find('#id_epf').val(),
        socso : form.find('#id_socso').val(),
        income_tax : form.find('#id_income_tax').val(),
        bank_name : form.find('#id_bank_name').val(),
        bank_acc_no : form.find('#id_bank_acc_no').val(),
        bank_acc_type : form.find('#id_bank_acc_type').val(),
        bank_address : form.find('#id_bank_address').val(),
        epf_no : form.find('#id_epf_no').val(),
        socso_no : form.find('#id_socso_no').val(),
        income_tax_no : form.find('#id_income_tax_no').val(),
        pay_type : form.find('#id_pay_type').val()
    };

    o['employee_salary'] = data;

    return o;
}

function get_data_qualification(t, o) {
    var f = (t == 'add' ? $('#add-form') : $('#edit-form'));
    var form = f.find('#form-employee-qualification');

    var data = {
        level : form.find('#id_level').val(),
        institute : form.find('#id_institute').val(),
        major : form.find('#id_major').val(),
        year : form.find('#id_year').val(),
        gpa : form.find('#id_gpa').val(),
        start_date : form.find('#id_start_date').val(),
        end_date : form.find('#id_end_date').val()
    };
}

```

```

        o['employee_qualification'] = data;
    }

    return o;
}

function get_search_param() {
    var param = {
        employee : $('#id_employee').val(),
        staff_id : $('#id_staff_id').val(),
        employment_status : $('#id_employment_status').val(),
        designation : $('#id_designation').val(),
        dept : $('#id_dept').val()
    };

    return param;
}

function sort_list() {
    var s = sort.set_sort_css($('#this'));
    nav_list.set_sort(s);
}

function init_list() {
    $('.hdchk').click(select_all);
    utils.bind_hoverlist($('.list_table tbody tr'));
    $('.list_table tbody').selectable({
        selected : function(evt, ui) {
            var id = ui.selected.id;
            func_edit(id);
        }
    });
    $('.sorthead').click(sort_list);
}

function init() {
    init_ui_opt();
    $('#id_add').click(show_form);
    $('#id_find').click(nav_list.show_list);
    $('#id_display').change(nav_list.show_list);
    $('#id_employee,#id_staff_id').tooltip({track : truetrue, width : 400});
    $('#dialog-edit').dialog({modal : true, width : 400});
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_add,#id_delete,#id_find'));
    nav_list.config.list_url = url.list;
    nav_list.config.list_func = init_list;
    nav_list.config.del_func = func_delete;
    nav_list.config.search_param_func = get_search_param;
    nav_list.init();
}

function load() {
    return menu.get('/admin/employee/', init);
}

return {
    load : load
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\empstatus.js

```

var empstatus = (function() {
    var url = {
        add : '/admin/empstatus/new/',
        create : '/admin/empstatus/create/',
        edit : '/admin/empstatus/edit/',
        update : '/admin/empstatus/update/',
        del : '/admin/empstatus/delete/',
        list : '/admin/empstatus/list/'
    };

    var popup_dialog_opt = null;

    function init_ui_opt() {
        popup_dialog_opt = {

```

```

        autoOpen : false,
        width : 350,
        resizable : false,
        draggable : true,
        modal : false,
        stack : true,
        zIndex : 1000
    );
}

function show_form() {
    $('#dialog-add-body').load(url.add, function() {
        $('.save_button.save').click(func_save);
        $('.save_button.cancel').click(func_cancel_add);
        $('#add-form').tooltip({track: truefunction save_success() {
    func_cancel_add();
    nav_list.show_list();
}

function update_success() {
    func_cancel_edit();
    nav_list.show_list();
}

function func_cancel_add() {
    $('#dialog-add').dialog('close');
    return false;
}

function func_cancel_edit() {
    $('#dialog-edit').dialog('close');
    return false;
}

function func_save() {
    var data = get_data('add');
    $('#add-form input').next().remove();
    $.post(url.create, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            save_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    $('#add-form input[name=' + e + ']').after(h);
                }
            }
        }

        else
            utils.show_dialog(2, result);
    });
}

return false;
}

function func_edit(id) {
    if (!id)
        return false;
}

```

```

id = utils.get_itemid(id);
$('#dialog_edit_body').load(url.edit + id, function() {
    $('.save_button.save').click(function() {
        return func_update(id);
    });
    $('.save_button.cancel').click(func_cancel_edit);
    $('#edit-form').tooltip({track: truereturn false;
}

function func_update(id) {
    var data = get_data('edit');
    $('#edit-form input').next().remove();
    $.post(url.update + id, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            update_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    $('#edit-form input[name=' + e + "']").after(h);
                }
            }
        }
    });

    else
        utils.show_dialog(2, result);
});
return false;
}

function func_delete() {
    var a = $('.chk:checked');
    if (a.length < 1) {
        utils.show_dialog(2, 'Please select record(s).');
        return;
    }

    var l = [];
    var trlist = [];
    a.each(function(idx, elm) {
        var id = $(this).parent().parent().attr('id');
        trlist.push('#' + id);
        id = utils.get_itemid(id);
        l.push(id);
    });

    var val = $('#id_pg').val();
    var arr = val.split(',');
    var currpg = parseInt(arr[3], 10);
    --currpg;
    var pgsize = $('#id_display').val();
    var search_by = $('#id_selection').val();
    var keyword = $('#id_query').val();
    var data = {
        'id[]' : l,
        pgnum : currpg,
        pgsize : pgsize,
        find : search_by,
        keyword : keyword
    };
}

```

```

$.post(url.del, data, function(result) {
    if (result.success == 1) {
        stat.show_status(0, result.message);
        nav_list.set_item_msg(result.itemscount);
        var tr = $(trlist.join(','));
        tr.remove();
        delete tr;
        if ($('.list_table tbody tr').length < 1) {
            $('.list_table').remove();
            utils.set_disabled('#id_delete', 1, null);
        }
    }
});

function select_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chk').attr('checked', 'checked');

    else
        $('.chk').removeAttr('checked');
}

function get_data(t) {
    var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

    var data = {
        name : form.find('#id_name').val()
    };

    return data;
}

function sort_list() {
    var s = sort.set_sort_css($(this));
    nav_list.set_sort(s);
}

function init_list() {
    $('#hdchk').click(select_all);
    utils.bind_hoverlist($('.list_table tbody tr'));
    $('.list_table tbody').selectable({
        selected : function(evt, ui) {
            var id = ui.selected.id;
            func_edit(id);
        }
    });
    $('.sortheader').click(sort_list);
}

function init() {
    init_ui_opt();
    $('#id_add').click(show_form);
    $('#id_find').click(nav_list.show_list);
    $('#id_display').change(nav_list.show_list);
    $('#id_query').keypress(nav_list.query_keypress);
    $('#id_query').keyup(nav_list.query_keyup);
    $('#id_query').tooltip({track: truefunction load() {
    return menu.get('/admin/empstatus/', init);
}

return {
    load : load
};
}();
}

```

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\hpcht.js

var hpcht = ( function() {
    var url = {
        data : '/admin/hourly/chart/data/'
    };

    // Radialize the colors
    Highcharts.getOptions().colors = $.map(Highcharts.getOptions().colors, function(color) {
        return {
            radialGradient : {
                cx : 0.5,
                cy : 0.3,
                r : 0.7
            },
            stops : [[0, color], [1, Highcharts.Color(color).brighten(-0.3).get('rgb')]] // darken
        }
    });
}

function chart1(result) {
    // Build the chart
    var chart = new Highcharts.Chart({
        chart : {
            renderTo : 'cht1',
            plotBackgroundColor : null,
            plotBorderWidth : null,
            plotShadow : false
        },
        title : {
            text : result.title
        },
        tooltip : {
            pointFormat : '{series.name}: <b>{point.percentage}%</b>',
            percentageDecimals : 1
        },
        plotOptions : {
            pie : {
                allowPointSelect : true,
                cursor : 'pointer',
                dataLabels : {
                    enabled : true,
                    color : '#000000',
                    connectorColor : '#000000',
                    formatter : function() {
                        return '<b>' + this.point.name + '</b>: ' + this.percentage.toFixed(2) + ' %';
                    }
                }
            }
        },
        series : [
            {
                type : 'pie',
                name : result.title,
                data : result.pie
            }
        ]
    });
}

function chart2(result) {
    var chart = new Highcharts.Chart({
        chart : {
            renderTo : 'cht2',
            type : 'column'
        },
        title : {
            text : result.title
        },
        subtitle : {
            text : ''
        },
        xAxis : {
            categories : result.column.categories
        },
        yAxis : {
            min : 0,
            title : {
                text : result.column.yaxis
            }
        }
    });
}

```

```

        }
    },
    legend : {
        layout : 'vertical',
        backgroundColor : '#FFFFFF',
        align : 'left',
        verticalAlign : 'top',
        x : 100,
        y : 70,
        floating : true,
        shadow : true
    },
    tooltip : {
        formatter : function() {
            return '' + this.x + ': ' + this.y;
        }
    },
    plotOptions : {
        column : {
            pointPadding : 0.2,
            borderWidth : 0
        }
    },
    series : [{
        name : 'Hourly Payroll',
        data : result.column.data,
        showInLegend : false
    }]
});
}

function draw_chart() {
    var data = get_search_param();
    $.post(url.data, data, function(result) {
        chart1(result);
        chart2(result);
    });
}

function get_checked_month() {
    if ($('#chkall').attr('checked') == 'checked')
        return 0;

    else {
        var a = [];
        $('.chkmonth:checked').each(function(idx, elm) {
            a.push($(this).val());
        });
        return (a == [] ? 0 : a);
    }
}

function get_search_param() {
    var month = get_checked_month();

    var param = {
        staff_id : $('#id_staff_id').val(),
        year : $('#id_year').val()
    };
    if (month == 0)
        param['month'] = 0;

    else
        param['month[]'] = month;

    return param;
}

function uncheck_all_month() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chkmonth').removeAttr('checked');
}

function uncheck_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')

```

```

        $('.chkall').removeAttr('checked');
    }

    function init() {
        $('#id_gen').click(draw_chart);
        $('.chkall').click(uncheck_all_month);
        $('.chkmonth').click(uncheck_all);
        $('#id_staff_id').tooltip({track: truefunction load() {
        menu.get('/admin/hourly/chart/', init);
    }

    return {
        load : load
    };
}();

E:\workspace\payroll_rails\app\assets\javascripts\admin\jobcat.js

var jobcat = (function() {
    var url = {
        add : '/admin/jobcat/new/',
        create : '/admin/jobcat/create/',
        edit : '/admin/jobcat/edit/',
        update : '/admin/jobcat/update/',
        del : '/admin/jobcat/delete/',
        list : '/admin/jobcat/list/'
    };

    var popup_dialog_opt = null;

    function init_ui_opt() {
        popup_dialog_opt = {
            autoOpen : false,
            width : 350,
            resizable : false,
            draggable : true,
            modal : false,
            stack : true,
            zIndex : 1000
        };
    }

    function show_form() {
        $('#dialog_add_body').load(url.add, function() {
            $('.save_button.save').click(func_save);
            $('.save_button.cancel').click(func_cancel_add);
            $('#add-form').tooltip({track: truefunction save_success() {
        func_cancel_add();
        nav_list.show_list();
    }

    function update_success() {
        func_cancel_edit();
        nav_list.show_list();
    }

    function func_cancel_add() {
        $('#dialog-add').dialog('close');
        return false;
    }

    function func_cancel_edit() {
        $('#dialog-edit').dialog('close');
        return false;
    }
})

```

```

function func_save() {
    var data = get_data('add');
    $('#add-form input').next().remove();
    $.post(url.create, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            save_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    $("#add-form input[name='" + e + "']").after(h);
                }
            }
        }

        else
            utils.show_dialog(2, result);
    });
}

return false;
}

function func_edit(id) {
    if (!id)
        return false;

    id = utils.get_itemid(id);
    $('#dialog_edit_body').load(url.edit + id, function() {
        $('.save_button.save').click(function() {
            return func_update(id);
        });
        $('.save_button.cancel').click(func_cancel_edit);
        $('#edit-form').tooltip({track: true});
        utils.bind_hover($('.save_button'));
        $('#dialog-edit').dialog('open');
    });
    return false;
}

function func_update(id) {
    var data = get_data('edit');
    $('#edit-form input').next().remove();
    $.post(url.update + id, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            update_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    $("#edit-form input[name='" + e + "']").after(h);
                }
            }
        }
    });
}

```

```

        else
            utils.show_dialog(2, result);
    });

    return false;
}

function func_delete() {
    var a = $('.chk:checked');
    if (a.length < 1) {
        utils.show_dialog(2, 'Please select record(s).');
        return;
    }

    var l = [];
    var trlist = [];
    a.each(function(idx, elm) {
        var id = $(this).parent().parent().attr('id');
        trlist.push('#' + id);
        id = utils.get_itemid(id);
        l.push(id);
    });

    var val = $('#id_pg').val();
    var arr = val.split(',');
    var currpg = parseInt(arr[3], 10);
    --currpg;
    var pgsize = $('#id_display').val();
    var search_by = $('#id_selection').val();
    var keyword = $('#id_query').val();
    var data = {
        'id[]' : l,
        pgnum : currpg,
        pgsize : pgsize,
        find : search_by,
        keyword : keyword
    };

    $.post(url.del, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            nav_list.set_item_msg(result.itemscount);
            var tr = $(trlist.join(','));
            tr.remove();
            delete tr;
            if ($('#list_table tbody tr').length < 1) {
                $('#list_table').remove();
                utils.set_disabled('#id_delete', 1, null);
            }
        }
    });
}

function select_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chk').attr('checked', 'checked');

    else
        $('.chk').removeAttr('checked');
}

function get_data(t) {
    var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

    var data = {
        name : form.find('#id_name').val()
    };

    return data;
}

function sort_list() {
    var s = sort.set_sort_css($(this));
    nav_list.set_sort(s);
}

```

```

function init_list() {
    $('.hdchk').click(select_all);
    utils.bind_hoverlist($('.list_table tbody tr'));
    $('.list_table tbody').selectable({
        selected : function(evt, ui) {
            var id = ui.selected.id;
            func_edit(id);
        }
    });
    $('.sortheader').click(sort_list);
}

function init() {
    init_ui_opt();
    $('#id_add').click(show_form);
    $('#id_find').click(nav_list.show_list);
    $('#id_display').change(nav_list.show_list);
    $('#id_query').keypress(nav_list.query_keypress);
    $('#id_query').keyup(nav_list.query_keyup);
    $('#id_query').tooltip({track: true});
    $('#dialog-add').dialog({popup_dialog_opt});
    $('#dialog-edit').dialog({popup_dialog_opt});
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_add,#id_delete,#id_find'));
    nav_list.config.list_url = url.list;
    nav_list.config.list_func = init_list;
    nav_list.config.del_func = func_delete;
    nav_list.init();
}

function load() {
    return menu.get('/admin/jobcat/', init);
}

return {
    load : load
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\otcht.js

```

var otcht = ( function() {
    var url = {
        data : '/admin/overtime/chart/data/'
    };

    function chart1(result) {
        var chart = new Highcharts.Chart({
            chart : {
                renderTo : 'crtl',
                type : 'bar'
            },
            title : {
                text : result.title
            },
            subtitle : {
                text : ''
            },
            xAxis : {
                categories : result.categories,
                title : {
                    text : null
                }
            },
            yAxis : {
                min : 0,
                title : {
                    text : result.yaxis,
                    align : 'high'
                }
            },
            labels : {
                overflow : 'justify'
            },
            tooltip : {
                formatter : function() {
                    return '' + this.x + ':' + this.y + ' hours';
                }
            }
        });
    }
});
```

```

        }
    },
    plotOptions : {
        bar : {
            dataLabels : {
                enabled : true
            }
        }
    },
    legend : {
        layout : 'vertical',
        align : 'right',
        verticalAlign : 'top',
        x : -100,
        y : 100,
        floating : true,
        borderWidth : 1,
        backgroundColor : '#FFFFFF',
        shadow : true
    },
    credits : {
        enabled : false
    },
    series : [{
        name : 'Overtime',
        data : result.data,
        showInLegend : false
    }]
});
}

function draw_chart() {
    var data = get_search_param();
    $.post(url.data, data, function(result) {
        chart1(result);
    });
}

function get_checked_month() {
    if ($('.chkall').attr('checked') == 'checked')
        return 0;

    else {
        var a = [];
        $('.chkmonth:checked').each(function(idx, elm) {
            a.push($('this').val());
        });
        return (a == [] ? 0 : a);
    }
}

function get_search_param() {
    var month = get_checked_month();

    var param = {
        staff_id : $('#id_staff_id').val(),
        year : $('#id_year').val()
    };
    if (month == 0)
        param['month'] = 0;

    else
        param['month[]'] = month;

    return param;
}

function uncheck_all_month() {
    var a = $('this').attr('checked');
    if (a == 'checked')
        $('chkmonth').removeAttr('checked');
}

function uncheck_all() {
    var a = $('this').attr('checked');
    if (a == 'checked')
        $('.chkall').removeAttr('checked');
}

```

```

}

function init() {
    $('#id_gen').click(draw_chart);
    $('.chkall').click(uncheck_all_month);
    $('.chkmonth').click(uncheck_all);
    $('#id_staff_id').tooltip({track: true});
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_gen'));
    draw_chart();
}

function load() {
    return menu.get('/admin/overtime/chart/', init);
}

return {
    load : load
};

}());

E:\workspace\payroll_rails\app\assets\javascripts\admin\overtimerate.js

var overtimerate = ( function() {
    var url = {
        add : '/admin/overtime/rate/new/',
        create : '/admin/overtime/rate/create/',
        edit : '/admin/overtime/rate/edit/',
        update : '/admin/overtime/rate/update/',
        del : '/admin/overtime/rate/delete/',
        list : '/admin/overtime/rate/list/'
    };

    var popup_dialog_opt = null;

    function init_ui_opt() {
        popup_dialog_opt = {
            autoOpen : false,
            width : 350,
            resizable : false,
            draggable : true,
            modal : false,
            stack : true,
            zIndex : 1000
        };
    }

    function show_form() {
        $('#dialog-add_body').load(url.add, function() {
            $('.save_button.save').click(func_save);
            $('.save_button.cancel').click(func_cancel_add);
            $('#add-form').tooltip({track: true});
            utils.bind_hover($('.save_button'));
            $('#dialog-add').dialog('open');
        });
    }

    function save_success() {
        func_cancel_add();
        nav_list.show_list();
    }

    function update_success() {
        func_cancel_edit();
        nav_list.show_list();
    }

    function func_cancel_add() {
        $('#dialog-add').dialog('close');
        return false;
    }

    function func_cancel_edit() {
        $('#dialog-edit').dialog('close');
        return false;
    }
}

```

```

function func_save() {
    var data = get_data('add');
    $('#add-form input').next().remove();
    $.post(url.create, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            save_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    if (e == 'year')
                        $('#add-form #id_' + e).after(h);

                    else
                        $("#add-form input[name='" + e + "']").after(h);
                }
            }
        }
        else
            utils.show_dialog(2, result);
    });
}

return false;
}

function func_edit(id) {
    if (!id)
        return false;

    id = utils.get_itemid(id);
    $('#dialog_edit_body').load(url.edit + id, function() {
        $('.save_button.save').click(function() {
            return func_update(id);
        });
        $('.save_button.cancel').click(func_cancel_edit);
        $('#edit-form').tooltip({track: true});
        utils.bind_hover($('.save_button'));
        $('#dialog-edit').dialog('open');
    });
    return false;
}

function func_update(id) {
    var data = get_data('edit');
    $('#edit-form input').next().remove();
    $.post(url.update + id, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            update_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    if (e == 'year')
                        $('#edit-form #id_' + e).after(h);
                }
            }
        }
    });
}

```

```

        else
            $("#edit-form input[name='" + e + "']").after(h);
        }
    }

    else
        utils.show_dialog(2, result);
});

return false;
}

function func_delete() {
    var a = $('.chk:checked');
    if (a.length < 1) {
        utils.show_dialog(2, 'Please select record(s).');
        return;
    }

    var l = [];
    var trlist = [];
    a.each(function(idx, elm) {
        var id = $(this).parent().parent().attr('id');
        trlist.push('#' + id);
        id = utils.get_itemid(id);
        l.push(id);
    });

    var val = $('#id_pg').val();
    var arr = val.split(',');
    var currpg = parseInt(arr[3], 10);
    --currpg;
    var pgsize = $('#id_display').val();
    var search_by = $('#id_selection').val();
    var keyword = $('#id_query').val();
    var data = get_search_param();
    data['id[]'] = l;
    data['pgnum'] = currpg;
    data['pgsize'] = pgsize;

    $.post(url.del, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            nav_list.set_item_msg(result.itemscount);
            var tr = $(trlist.join(','));
            tr.remove();
            delete tr;
            if ($('.list_table tbody tr')[0] == null) {
                $('.list_table').remove();
                utils.set_disabled('#id_delete', 1, null);
            }
        }
    });
}

function select_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chk').attr('checked', 'checked');

    else
        $('.chk').removeAttr('checked');
}

function get_data(t) {
    var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

    var data = {
        duration : form.find('#id_duration').val(),
        year : form.find('#id_year').val(),
        pay_rate : form.find('#id_pay_rate').val()
    };

    return data;
}

```

```

function get_search_param() {
  var param = {
    year : $('#id_year').val()
  };

  return param;
}

function sort_list() {
  var s = sort.set_sort_css($('#this'));
  nav_list.set_sort(s);
}

function init_list() {
  $('.hdchk').click(select_all);
  utils.bind_hoverlist($('.list_table tbody tr'));
  $('.list_table tbody').selectable({
    selected : function(evt, ui) {
      var id = ui.selected.id;
      func_edit(id);
    }
  });
  $('.sorthead').click(sort_list);
}

function init() {
  init_ui_opt();
  $('#id_add').click(show_form);
  $('#id_find').click(nav_list.show_list);
  $('#id_display').change(nav_list.show_list);
  $('#id_year').tooltip({track: true});
  $('#dialog-add').dialog(popup_dialog_opt);
  $('#dialog-edit').dialog(popup_dialog_opt);
  utils.init_alert_dialog('#dialog-message');
  utils.bind_hover($('#id_add,#id_delete,#id_find'));
  nav_list.config.list_url = url.list;
  nav_list.config.list_func = init_list;
  nav_list.config.del_func = func_delete;
  nav_list.config.search_param_func = get_search_param;
  nav_list.init();
}

function load() {
  return menu.get('/admin/overtime/rate/', init);
}

return {
  load : load
};
}();
}

E:\workspace\payroll_rails\app\assets\javascripts\admin\payrate.js

var payrate = ( function() {
  var url = {
    add : '/admin/payrate/new/',
    create : '/admin/payrate/create/',
    edit : '/admin/payrate/edit/',
    update : '/admin/payrate/update/',
    del : '/admin/payrate/delete/',
    list : '/admin/payrate/list/'
  };

  var popup_dialog_opt = null;

  function init_ui_opt() {
    popup_dialog_opt = {
      autoOpen : false,
      width : 350,
      resizable : false,
      draggable : true,
      modal : false,
      stack : true,
      zIndex : 1000
    };
  }
}

```

```

function show_form() {
    $('#dialog-add-body').load(url.add, function() {
        $('.save_button.save').click(func_save);
        $('.save_button.cancel').click(func_cancel_add);
        $('#add-form').tooltip({track: true});
        utils.bind_hover($('.save_button'));
        $('#dialog-add').dialog('open');
    });
}

function save_success() {
    func_cancel_add();
    nav_list.show_list();
}

function update_success() {
    func_cancel_edit();
    nav_list.show_list();
}

function func_cancel_add() {
    $('#dialog-add').dialog('close');
    return false;
}

function func_cancel_edit() {
    $('#dialog-edit').dialog('close');
    return false;
}

function func_save() {
    var data = get_data('add');
    $('#add-form input').next().remove();
    $.post(url.create, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            save_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    if (e == 'month' || e == 'year')
                        $('#add-form #id_' + e).after(h);

                    else
                        $('#add-form input[name=' + e + "']").after(h);
                }
            }
        }
        else
            utils.show_dialog(2, result);
    });
}

return false;
}

function func_edit(id) {
    if (!id)
        return false;

    id = utils.get_itemid(id);
    $('#dialog-edit-body').load(url.edit + id, function() {
        $('.save_button.save').click(function() {
            return func_update(id);
        });
        $('.save_button.cancel').click(func_cancel_edit);
        $('#edit-form').tooltip({track: true});
    });
}

```

```

        utils.bind_hover($('.save_button'));
        $('#dialog-edit').dialog('open');
    });
    return false;
}

function func_update(id) {
    var data = get_data('edit');
    $('#edit-form input').next().remove();
    $.post(url.update + id, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            update_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    if (e == 'month' || e == 'year')
                        $('#edit-form #id_' + e).after(h);

                    else
                        $('#edit-form input[name=' + e + "']").after(h);
                }
            }
        }

        else
            utils.show_dialog(2, result);
    });
    return false;
}

function func_delete() {
    var a = $('.chk:checked');
    if (a.length < 1) {
        utils.show_dialog(2, 'Please select record(s).');
        return;
    }

    var l = [];
    var trlist = [];
    a.each(function(idx, elm) {
        var id = $(this).parent().parent().attr('id');
        trlist.push('#' + id);
        id = utils.get_itemid(id);
        l.push(id);
    });

    var val = $('#id_pg').val();
    var arr = val.split(',');
    var currpg = parseInt(arr[3], 10);
    --currpg;
    var pgszie = $('#id_display').val();
    var search_by = $('#id_selection').val();
    var keyword = $('#id_query').val();
    var data = get_search_param();
    data['id[]'] = l;
    data['pgnum'] = currpg;
    data['pgsize'] = pgszie;

    $.post(url.del, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            nav_list.set_item_msg(result.itemscount);
            var tr = $(trlist.join(','));
            tr.remove();
        }
    });
}

```

```

        delete tr;
        if ($.list_table tbody tr')[0] == null) {
            $('.list_table').remove();
            utils.set_disabled('#id_delete', 1, null);
        }
    });
}

function select_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chk').attr('checked', 'checked');

    else
        $('.chk').removeAttr('checked');
}

function get_data(t) {
    var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

    var data = {
        staff_id : form.find('#id_staff_id').val(),
        month : form.find('#id_month').val(),
        year : form.find('#id_year').val(),
        pay_rate : form.find('#id_pay_rate').val()
    };

    return data;
}

function get_search_param() {
    var param = {
        staff_id : $('#id_staff_id').val(),
        month : $('#id_month').val(),
        year : $('#id_year').val()
    };

    return param;
}

function sort_list() {
    var s = sort.set_sort_css($(this));
    nav_list.set_sort(s);
}

function init_list() {
    $('#hdchk').click(select_all);
    utils.bind_hoverlist($('.list_table tbody tr'));
    $('.list_table tbody').selectable({
        selected : function(evt, ui) {
            var id = ui.selected.id;
            func_edit(id);
        }
    });
    $('.sortheader').click(sort_list);
}

function init() {
    init_ui_opt();
    $('#id_add').click(show_form);
    $('#id_find').click(nav_list.show_list);
    $('#id_display').change(nav_list.show_list);
    $('#id_staff_id').tooltip({track: true});
    $('#dialog-add').dialog(popup_dialog_opt);
    $('#dialog-edit').dialog(popup_dialog_opt);
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_add,#id_delete,#id_find'));
    nav_list.config.list_url = url.list;
    nav_list.config.list_func = init_list;
    nav_list.config.del_func = func_delete;
    nav_list.config.search_param_func = get_search_param;
    nav_list.init();
}

function load() {
    return menu.get('/admin/payrate/', init);
}

```

```

}

return {
  load : load
};

}());

E:\workspace\payroll_rails\app\assets\javascripts\admin\payslip.js

var payslip = ( function() {
  var url = {
    list : '/admin/payslip/list/',
    salaryslip : '/admin/payslip/slip/'
  };

  function func_generate(id) {
    if (!id)
      return false;

    id = utils.get_itemid(id);
    var month = $('#id_month').val();
    var year = $('#id_year').val();
    var f = 'width=800,height=400,menubar=1';
    open(url.salaryslip + id + '/' + month + '/' + year, '_blank', f);
    return false;
  }

  function get_search_param() {
    var param = {
      employee : $('#id_employee').val(),
      staff_id : $('#id_staff_id').val(),
      employment_status : $('#id_employment_status').val(),
      designation : $('#id_designation').val(),
      dept : $('#id_dept').val()
    };

    return param;
  }

  function sort_list() {
    var s = sort.set_sort_css($(this));
    nav_list.set_sort(s);
  }

  function init_list() {
    utils.bind_hoverlist($('.list_table tbody tr'));
    $('.list_table tbody').selectable({
      selected : function(evt, ui) {
        var id = ui.selected.id;
        func_generate(id);
      }
    });
    $('.sortheader').click(sort_list);
  }

  function init() {
    $('#id_find').click(nav_list.show_list);
    $('#id_display').change(nav_list.show_list);
    $('#id_employee,#id_staff_id').tooltip({track : true});
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_find'));
    nav_list.config.list_url = url.list;
    nav_list.config.list_func = init_list;
    nav_list.config.search_param_func = get_search_param;
    nav_list.init();
  }

  function load() {
    return menu.get('/admin/payslip/', init);
  }

  return {
    load : load
  };
}());

```

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\saladj.js

var saladj = ( function() {
  var url = {
    add : '/admin/salaryadj/new/',
    create : '/admin/salaryadj/create/',
    edit : '/admin/salaryadj/edit/',
    update : '/admin/salaryadj/update/',
    del : '/admin/salaryadj/delete/',
    list : '/admin/salaryadj/list/'
  };

  var popup_dialog_opt = null;

  function init_ui_opt() {
    popup_dialog_opt = {
      autoOpen : false,
      width : 350,
      resizable : false,
      draggable : true,
      modal : false,
      stack : true,
      zIndex : 1000
    };
  }

  function show_form() {
    $('#dialog-add_body').load(url.add, function() {
      $('.save_button.save').click(func_save);
      $('.save_button.cancel').click(func_cancel_add);
      $('#add-form').tooltip({track: true});
      utils.bind_hover($('.save_button'));
      $('#dialog-add').dialog('open');
    });
  }

  function save_success() {
    func_cancel_add();
    nav_list.show_list();
  }

  function update_success() {
    func_cancel_edit();
    nav_list.show_list();
  }

  function func_cancel_add() {
    $('#dialog-add').dialog('close');
    return false;
  }

  function func_cancel_edit() {
    $('#dialog-edit').dialog('close');
    return false;
  }

  function func_save() {
    var data = get_data('add');
    $('#add-form input').next().remove();
    $.post(url.create, data, function(result) {
      if (result.success == 1) {
        stat.show_status(0, result.message);
        save_success();
      }

      else if (result.error == 1) {
        for (var e in result.errors) {
          var d = $('#error_' + e).get(0);
          if (!d) {
            var o = {
              field : e,
              msg : result.errors[e][0]
            };
            var h = new EJS({
              url : '/assets/tpl/label_error.html',
              ext : '.html'
            }).render(o);

```

```

        if (e == 'month' || e == 'year')
            $('#add-form #id_' + e).after(h);

        else
            $("#" + "#add-form input[name=' + e + "']").after(h);
    }
}
else
    utils.show_dialog(2, result);
});

return false;
}

function func_edit(id) {
if (!id)
    return false;

id = utils.get_itemid(id);
$('#dialog_edit_body').load(url.edit + id, function() {
    $('.save_button.save').click(function() {
        return func_update(id);
    });
    $('.save_button.cancel').click(func_cancel_edit);
    $('#edit-form').tooltip({track: true});
    utils.bind_hover($('.save_button'));
    $('#dialog-edit').dialog('open');
});
return false;
}

function func_update(id) {
var data = get_data('edit');
$('#edit-form input').next().remove();
$.post(url.update + id, data, function(result) {
    if (result.success == 1) {
        stat.show_status(0, result.message);
        update_success();
    }

    else if (result.error == 1) {
        for (var e in result.errors) {
            var d = $('#error_' + e).get(0);
            if (!d) {
                var o = {
                    field : e,
                    msg : result.errors[e][0]
                };
                var h = new EJS({
                    url : '/assets/tpl/label_error.html',
                    ext : '.html'
                }).render(o);
                if (e == 'month' || e == 'year')
                    $('#edit-form #id_' + e).after(h);

                else
                    $("#" + "#edit-form input[name=' + e + "']).after(h);
            }
        }
    }

    else
        utils.show_dialog(2, result);
});

return false;
}

function func_delete() {
var a = $('.chk:checked');
if (a.length < 1) {
    utils.show_dialog(2, 'Please select record(s).');
    return;
}

var l = [];

```

```

var trlist = [];
a.each(function(idx, elm) {
    var id = $(this).parent().parent().attr('id');
    trlist.push('#' + id);
    id = utils.get_itemid(id);
    l.push(id);
});

var val = $('#id_pg').val();
var arr = val.split(',');
var currpg = parseInt(arr[3], 10);
--currpg;
var pgszie = $('#id_display').val();
var search_by = $('#id_selection').val();
var keyword = $('#id_query').val();
var data = get_search_param();
data['id[]'] = l;
data['pgnum'] = currpg;
data['pgsize'] = pgszie;

$.post(url.del, data, function(result) {
    if (result.success == 1) {
        stat.show_status(0, result.message);
        nav_list.set_item_msg(result.itemscount);
        var tr = $(trlist.join(','));
        tr.remove();
        delete tr;
        if ($.list_table tbody tr)[0] == null) {
            $('.list_table').remove();
            utils.set_disabled('#id_delete', 1, null);
        }
    }
});

function select_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chk').attr('checked', 'checked');

    else
        $('.chk').removeAttr('checked');
}

function get_data(t) {
    var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

    var data = {
        staff_id : form.find('#id_staff_id').val(),
        inc : form.find('#id_inc').val(),
        month : form.find('#id_month').val(),
        year : form.find('#id_year').val()
    };

    return data;
}

function get_search_param() {
    var param = {
        staff_id : $('#id_staff_id').val(),
        month : $('#id_month').val(),
        year : $('#id_year').val()
    };

    return param;
}

function sort_list() {
    var s = sort.set_sort_css($(this));
    nav_list.set_sort(s);
}

function init_list() {
    $('.hdchk').click(select_all);
    utils.bind_hoverlist($('.list_table tbody tr'));
    $('.list_table tbody').selectable({
        selected : function(evt, ui) {
}

```

```

        var id = ui.selected.id;
        func_edit(id);
    }
});
$('.sortheader').click(sort_list);
}

function init() {
    init_ui_opt();
    $('#id_add').click(show_form);
    $('#id_find').click(nav_list.show_list);
    $('#id_display').change(nav_list.show_list);
    $('#id_staff_id').tooltip({track: true});
    $('#dialog-add').dialog(popup_dialog_opt);
    $('#dialog-edit').dialog(popup_dialog_opt);
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_add,#id_delete,#id_find'));
    nav_list.config.list_url = url.list;
    nav_list.config.list_func = init_list;
    nav_list.config.del_func = func_delete;
    nav_list.config.search_param_func = get_search_param;
    nav_list.init();
}

function load() {
    return menu.get('/admin/salaryadj/', init);
}

return {
    load : load
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\user.js

```

var user = ( function() {
    var url = {
        add : '/admin/user/new/',
        create : '/admin/user/create/',
        edit : '/admin/user/edit/',
        update : '/admin/user/update/',
        del : '/admin/user/delete/',
        list : '/admin/user/list/'
    };

    var popup_dialog_opt = null;

    function init_ui_opt() {
        popup_dialog_opt = {
            autoOpen : false,
            width : 350,
            resizable : false,
            draggable : true,
            modal : false,
            stack : true,
            zIndex : 1000
        };
    }

    function show_form() {
        $('#dialog_add_body').load(url.add, function() {
            $('.save_button.save').click(func_save);
            $('.save_button.cancel').click(func_cancel_add);
            $('#add-form').tooltip({track: true});
            utils.bind_hover($('.save_button'));
            $('#dialog-add').dialog('open');
        });
    }

    function save_success() {
        func_cancel_add();
        nav_list.show_list();
    }

    function update_success() {
        func_cancel_edit();
        nav_list.show_list();
    }
}

```

```

}

function func_cancel_add() {
    $('#dialog-add').dialog('close');
    return false;
}

function func_cancel_edit() {
    $('#dialog-edit').dialog('close');
    return false;
}

function func_save() {
    var data = get_data('add');
    $('#add-form input').next().remove();
    $.post(url.create, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            save_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,
                        msg : result.errors[e][0]
                    };
                    var h = new EJS({
                        url : '/assets/tpl/label_error.html',
                        ext : '.html'
                    }).render(o);
                    $('#add-form input[name=' + e + "']").after(h);
                }
            }
        }
        else
            utils.show_dialog(2, result);
    });
}

return false;
}

function func_edit(id) {
    if (!id)
        return false;

    id = utils.get_itemid(id);
    $('#dialog-edit_body').load(url.edit + id, function() {
        $('#edit-form').find('#id_password').val('*****');
        $('#save_button.save').click(function() {
            return func_update(id);
        });
        $('.save_button.cancel').click(func_cancel_edit);
        $('#edit-form').tooltip({track: true});
        utils.bind_hover($('.save_button'));
        $('#dialog-edit').dialog('open');
    });
    return false;
}

function func_update(id) {
    var data = get_data('edit');
    $('#edit-form input').next().remove();
    $.post(url.update + id, data, function(result) {
        if (result.success == 1) {
            stat.show_status(0, result.message);
            update_success();
        }

        else if (result.error == 1) {
            for (var e in result.errors) {
                var d = $('#error_' + e).get(0);
                if (!d) {
                    var o = {
                        field : e,

```

```

        msg : result.errors[e][0]
    };
    var h = new EJS({
        url : '/assets/tpl/label_error.html',
        ext : '.html'
    }).render(o);
    $("#edit-form input[name='" + e + "']").after(h);
}
}

else
    utils.show_dialog(2, result);
});

return false;
}

function func_delete() {
var a = $(".chk:checked");
if (a.length < 1) {
    utils.show_dialog(2, 'Please select record(s).');
    return;
}

var l = [];
var trlist = [];
a.each(function(idx, elm) {
    var id = $(this).parent().parent().attr('id');
    trlist.push('#' + id);
    id = utils.get_itemid(id);
    l.push(id);
});

var val = $('#id_pg').val();
var arr = val.split(',');
var currpg = parseInt(arr[3], 10);
--currpg;
var pgszie = $('#id_display').val();
var search_by = $('#id_selection').val();
var keyword = $('#id_query').val();
var data = get_search_param();
data['id[]'] = l;
data['pgnum'] = currpg;
data['pgsize'] = pgszie;

$.post(url.del, data, function(result) {
    if (result.success == 1) {
        stat.show_status(0, result.message);
        nav_list.set_item_msg(result.itemscount);
        var tr = $(trlist.join(','));
        tr.remove();
        delete tr;
        if ($.list_table tbody tr)[0] == null) {
            $('.list_table').remove();
            utils.set_disabled('#id_delete', 1, null);
        }
    }
});
}

function select_all() {
var a = $(this).attr('checked');
if (a == 'checked')
    $('.chk').attr('checked', 'checked');

else
    $('.chk').removeAttr('checked');
}

function get_data(t) {
var form = (t == 'add' ? $('#add-form') : $('#edit-form'));

var data = {
    role : form.find('#id_role').val(),
    username : form.find('#id_username').val(),
    status : form.find('#id_status').val(),
}
}

```

```

        pwd : form.find('#id_password').val(),
        pwdconfirm : form.find('#id_passwordconfirm').val()
    );
}

return data;
}

function get_search_param() {
    var param = {
        username : $('#id_username').val(),
        role : $('#id_user_role').val(),
        employee : $('#id_employee').val(),
        status : $('#id_status').val()
    };
}

return param;
}

function sort_list() {
    var s = sort.set_sort_css($(this));
    nav_list.set_sort(s);
}

function init_list() {
    $('.hdchk').click(select_all);
    utils.bind_hoverlist($('.list_table tbody tr'));
    $('.list_table tbody').selectable({
        selected : function(evt, ui) {
            var id = ui.selected.id;
            func_edit(id);
        }
    });
    $('.sorthead').click(sort_list);
}

function init() {
    init_ui_opt();
    $('#id_add').click(show_form);
    $('#id_find').click(nav_list.show_list);
    $('#id_display').change(nav_list.show_list);
    $('#id_username,#id_employee').tooltip({track: true});
    $('#dialog-add').dialog({modal: true, title: 'Add User'});
    $('#dialog-edit').dialog({modal: true, title: 'Edit User'});
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_add,#id_delete,#id_find'));
    nav_list.config.list_url = url.list;
    nav_list.config.list_func = init_list;
    nav_list.config.del_func = func_delete;
    nav_list.config.search_param_func = get_search_param;
    nav_list.init();
}

function load() {
    return menu.get('/admin/user/', init);
}

return {
    load : load
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\wkcht.js

```

var wkcht = ( function() {
    var url = {
        data : '/admin/workhours/chart/data/'
    };

    function chart1(result) {
        var chart = new Highcharts.Chart({
            chart : {
                renderTo : 'crtl',
                type : 'line'
            },
            title : {
                text : result.title
            },

```

```

        subtitle : {
            text : ''
        },
        xAxis : {
            categories : result.categories
        },
        yAxis : {
            title : {
                text : result.yaxis
            },
            plotLines : [{
                value : 0,
                width : 1,
                color : '#808080'
            }]
        },
        tooltip : {
            formatter : function() {
                return '' + this.x + ':' + this.y + ' hours';
            }
        },
        plotOptions : {
            line : {
                dataLabels : {
                    enabled : true
                }
            }
        },
        legend : {
            layout : 'vertical',
            align : 'right',
            verticalAlign : 'top',
            x : -10,
            y : 100,
            borderWidth : 0
        },
        series : [{
            name : 'Hours Worked',
            data : result.data,
            showInLegend : false
        }]
    });
}

function draw_chart() {
    var data = get_search_param();
    $.post(url.data, data, function(result) {
        chart1(result);
    });
}

function get_checked_month() {
    if ($('.chkall').attr('checked') == 'checked')
        return 0;

    else {
        var a = [];
        $('.chkmonth:checked').each(function(idx, elm) {
            a.push($(this).val());
        });
        return (a == [] ? 0 : a);
    }
}

function get_search_param() {
    var month = get_checked_month();

    var param = {
        staff_id : $('#id_staff_id').val(),
        year : $('#id_year').val()
    };
    if (month == 0)
        param['month'] = 0;

    else
        param['month[]'] = month;
}

```

```

        return param;
    }

    function uncheck_all_month() {
        var a = $(this).attr('checked');
        if (a == 'checked')
            $('.chkmonth').removeAttr('checked');
    }

    function uncheck_all() {
        var a = $(this).attr('checked');
        if (a == 'checked')
            $('.chkall').removeAttr('checked');
    }

    function init() {
        $('#id_gen').click(draw_chart);
        $('.chkall').click(uncheck_all_month);
        $('.chkmonth').click(uncheck_all);
        $('#id_staff_id').tooltip({track: true});
        utils.init_alert_dialog('#dialog-message');
        utils.bind_hover($('#id_gen'));
        draw_chart();
    }

    function load() {
        return menu.get('/admin/workhours/chart/', init);
    }

    return {
        load : load
    };
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\admin\wpayslip.js

```

var wpayslip = ( function() {
    function init() {
        utils.bind_hover($('.print_button'));
        $('.print_button').click(function() {
            print();
        });
        var w = opener;
        var _theme = w.theme.current_theme();
        $.themes.init({defaultTheme : _theme});
    }

    return {
        init : init
    };
}());

$(document).ready(wipayslip.init);

```

E:\workspace\payroll_rails\app\assets\javascripts\user\base.js

```

var app = ( function() {

    function init() {
        menu.init();
        utils.init_progress();
        utils.init_server_error_dialog();
        theme.init();
        utils.bind_hover($('#logout'));
        $('#menu_info').addClass('menu_active');
        info.load();
    }

    return {
        init : init
    };
}());

$(app.init);

```

```

E:\workspace\payroll_rails\app\assets\javascripts\user\contact.js

var contact = ( function() {
    var url = {
        update : '/user/contact/update/'
    };

    function func_save() {
        var data = get_data();
        $('#save-form input[type="text"]').next().remove();
        $.post(url.update, data, function(result) {
            if (result.success == 1)
                stat.show_status(0, result.message);

            else if (result.error == 1) {
                for (var e in result.errors) {
                    var d = $('#save-form #error_' + e).get(0);
                    if (!d) {
                        var o = {
                            field : e,
                            msg : result.errors[e][0]
                        };
                        var h = new EJS({
                            url : '/assets/tpl/label_error_inline.html',
                            ext : '.html'
                        }).render(o);
                        $('#save-form input[name=' + e + "']").after(h);
                    }
                }
            }
            else
                utils.show_dialog(2, result);
        });
    }

    return false;
}

function get_data() {
    var form = $('#save-form');

    var data = {
        address_1 : form.find('#id_address_1').val(),
        address_2 : form.find('#id_address_2').val(),
        address_3 : form.find('#id_address_3').val(),
        city : form.find('#id_city').val(),
        state : form.find('#id_state').val(),
        postcode : form.find('#id_postcode').val(),
        country : form.find('#id_country').val(),
        home_phone : form.find('#id_home_phone').val(),
        mobile_phone : form.find('#id_mobile_phone').val(),
        work_email : form.find('#id_work_email').val(),
        other_email : form.find('#id_other_email').val()
    };

    return {
        employee_contact : data
    };
}

function init() {
    $('.save_button.save').click(func_save);
    $('#save-form').tooltip({track : true});
    utils.bind_hover($('.save_button'));
    utils.init_alert_dialog('#dialog-message');
}

function load() {
    return menu.get('/user/contact/', init);
}

return {
    load : load
};
}();
}

```

```

E:\workspace\payroll_rails\app\assets\javascripts\user\hpcht.js

var hpcht = ( function() {
  var url = {
    data : '/user/hourly/chart/data/'
  };

  // Radialize the colors
  Highcharts.getOptions().colors = $.map(Highcharts.getOptions().colors, function(color) {
    return {
      radialGradient : {
        cx : 0.5,
        cy : 0.3,
        r : 0.7
      },
      stops : [[0, color], [1, Highcharts.Color(color).brighten(-0.3).get('rgb')]] // darken
    }
  });
}

function chart1(result) {
  // Build the chart
  var chart = new Highcharts.Chart({
    chart : {
      renderTo : 'cht1',
      plotBackgroundColor : null,
      plotBorderWidth : null,
      plotShadow : false
    },
    title : {
      text : result.title
    },
    tooltip : {
      pointFormat : '{series.name}: <b>{point.percentage}%</b>',
      percentageDecimals : 1
    },
    plotOptions : {
      pie : {
        allowPointSelect : true,
        cursor : 'pointer',
        dataLabels : {
          enabled : true,
          color : '#000000',
          connectorColor : '#000000',
          formatter : function() {
            return '<b>' + this.point.name + '</b>: ' + this.percentage.toFixed(2) + ' %';
          }
        }
      }
    },
    series : [
      {
        type : 'pie',
        name : result.title,
        data : result.pie
      }
    ]
  });
}

function chart2(result) {
  var chart = new Highcharts.Chart({
    chart : {
      renderTo : 'cht2',
      type : 'column'
    },
    title : {
      text : result.title
    },
    subtitle : {
      text : ''
    },
    xAxis : {
      categories : result.column.categories
    },
    yAxis : {
      min : 0,
      title : {
        text : result.column.yaxis
      }
    }
  });
}

```

```

        }
    },
    legend : {
        layout : 'vertical',
        backgroundColor : '#FFFFFF',
        align : 'left',
        verticalAlign : 'top',
        x : 100,
        y : 70,
        floating : true,
        shadow : true
    },
    tooltip : {
        formatter : function() {
            return '' + this.x + ': ' + this.y;
        }
    },
    plotOptions : {
        column : {
            pointPadding : 0.2,
            borderWidth : 0
        }
    },
    series : [{
        name : 'Hourly Payroll',
        data : result.column.data,
        showInLegend : false
    }]
});
}

function draw_chart() {
    var data = get_search_param();
    $.post(url.data, data, function(result) {
        chart1(result);
        chart2(result);
    });
}

function get_checked_month() {
    if $('.chkall').attr('checked') == 'checked')
        return 0;

    else {
        var a = [];
        $('.chkmonth:checked').each(function(idx, elm) {
            a.push($(this).val());
        });
        return (a == [] ? 0 : a);
    }
}

function get_search_param() {
    var month = get_checked_month();

    var param = {
        year : $('#id_year').val()
    };
    if (month == 0)
        param['month'] = 0;

    else
        param['month[]'] = month;

    return param;
}

function uncheck_all_month() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chkmonth').removeAttr('checked');
}

function uncheck_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chkall').removeAttr('checked');
}

```

```

}

function init() {
    $('#id_gen').click(draw_chart);
    $('.chkall').click(uncheck_all_month);
    $('.chkmonth').click(uncheck_all);
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_gen'));
    draw_chart();
}

function load() {
    menu.get('/user/hourly/chart/', init);
}

return {
    load : load
};

}());

E:\workspace\payroll_rails\app\assets\javascripts\user\info.js

var info = ( function() {
    var url = {
        update : '/user/info/update/'
    };

    function func_save() {
        var data = get_data();
        $('#save-form input[type="text"]').next().remove();
        $('#save-form select').next().remove();
        $('#save-form').find('#lb_gender_f').next().remove();
        $.post(url.update, data, function(result) {
            if (result.success == 1)
                stat.show_status(0, result.message);

            else if (result.error == 1) {
                for (var e in result.errors) {
                    var d = $('#save-form #error_' + e).get(0);
                    if (!d) {
                        var o = {
                            field : e,
                            msg : result.errors[e][0]
                        };
                        var h = new EJS({
                            url : '/assets/tpl/label_error.html',
                            ext : '.html'
                        }).render(o);
                        if (e == 'gender')
                            $('#save-form #lb_gender_f').after(h);

                        else if (e == 'marital_status')
                            $('#save-form #id_marital_status').after(h);

                        else
                            $('#save-form input[name=' + e + "']").after(h);
                    }
                }
            }
            else
                utils.show_dialog(2, result);
        });
    }

    return false;
}

function get_data() {
    var form = $('#save-form');

    var data = {
        first_name : form.find('#id_first_name').val(),
        middle_name : form.find('#id_middle_name').val(),
        last_name : form.find('#id_last_name').val(),
        new_ic : form.find('#id_new_ic').val(),
        old_ic : form.find('#id_old_ic').val(),
        passport_no : form.find('#id_passport_no').val(),
    }
}

```

```

        gender : form.find("input[name='gender']:checked").val(),
        marital_status : form.find('#id_marital_status').val(),
        nationality : form.find('#id_nationality').val(),
        dob : form.find('#id_dob').val(),
        place_of_birth : form.find('#id_place_of_birth').val(),
        race : form.find('#id_race').val(),
        religion : form.find('#id_religion').val(),
        is_bumi : form.find('#id_is_bumi').prop('checked')
    };

    return {
        employee : data
    };
}

function init() {
    $('.date_input').datepicker(utils.date_opt());
    $('.save_button.save').click(func_save);
    $('#save-form').tooltip({track : true});
    utils.bind_hover($('.save_button'));
    utils.init_alert_dialog('#dialog-message');
}

function load() {
    return menu.get('/user/info/', init);
}

return {
    load : load
};
}();

```

E:\workspace\payroll_rails\app\assets\javascripts\user\job.js

```

var job = ( function() {

    function init() {
        utils.init_alert_dialog('#dialog-message');
    }

    function load() {
        return menu.get('/user/job/', init);
    }

    return {
        load : load
    };
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\user\otcht.js

```

var otcht = ( function() {
    var url = {
        data : '/user/overtime/chart/data/'
    };

    function chart1(result) {
        var chart = new Highcharts.Chart({
            chart : {
                renderTo : 'cht1',
                type : 'bar'
            },
            title : {
                text : result.title
            },
            subtitle : {
                text : ''
            },
            xAxis : {
                categories : result.categories,
                title : {
                    text : null
                }
            },
            yAxis : {
                min : 0,
                title : {

```

```

        text : result.yaxis,
        align : 'high'
    },
    labels : {
        overflow : 'justify'
    }
},
tooltip : {
    formatter : function() {
        return '' + this.x + ':' + this.y + ' hours';
    }
},
plotOptions : {
    bar : {
        dataLabels : {
            enabled : true
        }
    }
},
legend : {
    layout : 'vertical',
    align : 'right',
    verticalAlign : 'top',
    x : -100,
    y : 100,
    floating : true,
    borderWidth : 1,
    backgroundColor : '#FFFFFF',
    shadow : true
},
credits : {
    enabled : false
},
series : [<{
        name : 'Overtime',
        data : result.data,
        showInLegend : false
}]]);
}

function draw_chart() {
    var data = get_search_param();
    $.post(url.data, data, function(result) {
        chart1(result);
    });
}

function get_checked_month() {
    if ($('.chkall').attr('checked') == 'checked')
        return 0;

    else {
        var a = [];
        $('.chkmonth:checked').each(function(idx, elm) {
            a.push($('.this').val());
        });
        return (a == [] ? 0 : a);
    }
}

function get_search_param() {
    var month = get_checked_month();

    var param = {
        year : $('#id_year').val()
    };
    if (month == 0)
        param['month'] = 0;

    else
        param['month[]'] = month;

    return param;
}

function uncheck_all_month() {
}

```

```

var a = $(this).attr('checked');
if (a == 'checked')
    $('.chkmonth').removeAttr('checked');
}

function uncheck_all() {
    var a = $(this).attr('checked');
    if (a == 'checked')
        $('.chkall').removeAttr('checked');
}

function init() {
    $('#id_gen').click(draw_chart);
    $('.chkall').click(uncheck_all_month);
    $('.chkmonth').click(uncheck_all);
    utils.init_alert_dialog('#dialog-message');
    utils.bind_hover($('#id_gen'));
    draw_chart();
}

function load() {
    return menu.get('/user/overtime/chart/', init);
}

return {
    load : load
};
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\user\payslip.js

```

var payslip = ( function() {
    var url = {
        salaryslip : '/user/payslip/slip/'
    };

    function func_generate() {
        var month = $('#id_month').val();
        var year = $('#id_year').val();
        var f = 'width=800,height=400,menubar=1';
        open(url.salaryslip + month + '/' + year, '_blank', f);
        return false;
    }

    function init() {
        $('#id_gen').click(func_generate);
        utils.init_alert_dialog('#dialog-message');
        utils.bind_hover($('#id_gen'));
    }

    function load() {
        return menu.get('/user/payslip/', init);
    }

    return {
        load : load
    };
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\user\qualification.js

```

var qualification = ( function() {
    var url = {
        update : '/user/qualification/update/'
    };

    function func_save() {
        var data = get_data();
        $('#save-form input[type="text"]').next().remove();
        $('#save-form select').next().remove();
        $.post(url.update, data, function(result) {
            if (result.success == 1)
                stat.show_status(0, result.message);

            else if (result.error == 1) {
                for (var e in result.errors) {
                    var d = $('#save-form #error_' + e).get(0);

```

```

        if (!d) {
            var o = {
                field : e,
                msg : result.errors[e][0]
            };
            var h = new EJS({
                url : '/assets/tpl/label_error_inline.html',
                ext : '.html'
            }).render(o);
            $("#save-form input[name='" + e + "']").after(h);
        }
    }

    else
        utils.show_dialog(2, result);
});

return false;
}

function get_data() {
    var form = $('#save-form');

    var data = {
        level : form.find('#id_level').val(),
        institute : form.find('#id_institute').val(),
        major : form.find('#id_major').val(),
        year : form.find('#id_year').val(),
        gpa : form.find('#id_gpa').val(),
        start_date : form.find('#id_start_date').val(),
        end_date : form.find('#id_end_date').val()
    };

    return {
        employee_qualification : data
    };
}

function init() {
    $('.date_input').datepicker(utils.date_opt());
    $('.save_button.save').click(func_save);
    $('#save-form').tooltip({track : true});
    utils.bind_hover($('.save_button'));
    utils.init_alert_dialog('#dialog-message');
}

function load() {
    return menu.get('/user/qualification/', init);
}

return {
    load : load
};
}();
}

E:\workspace\payroll_rails\app\assets\javascripts\user\salary.js

var salary = ( function() {

    function init() {
        utils.init_alert_dialog('#dialog-message');
    }

    function load() {
        return menu.get('/user/salary/', init);
    }

    return {
        load : load
    };
}());

```

```

E:\workspace\payroll_rails\app\assets\javascripts\user\wkcht.js

var wkcht = ( function() {
    var url = {
        data : '/user/workhours/chart/data/'
    };

    function chart1(result) {
        var chart = new Highcharts.Chart({
            chart : {
                renderTo : 'cht1',
                type : 'line'
            },
            title : {
                text : result.title
            },
            subtitle : {
                text : ''
            },
            xAxis : {
                categories : result.categories
            },
            yAxis : {
                title : {
                    text : result.yaxis
                },
                plotLines : [{
                    value : 0,
                    width : 1,
                    color : '#808080'
                }]
            },
            tooltip : {
                formatter : function() {
                    return '' + this.x + ': ' + this.y + ' hours';
                }
            },
            plotOptions : {
                line : {
                    dataLabels : {
                        enabled : true
                    }
                }
            },
            legend : {
                layout : 'vertical',
                align : 'right',
                verticalAlign : 'top',
                x : -10,
                y : 100,
                borderWidth : 0
            },
            series : [
                {
                    name : 'Hours Worked',
                    data : result.data,
                    showInLegend : false
                }
            ]);
    }

    function draw_chart() {
        var data = get_search_param();
        $.post(url.data, data, function(result) {
            chart1(result);
        });
    }

    function get_checked_month() {
        if ('.chkall').attr('checked') == 'checked'
            return 0;

        else {
            var a = [];
            $('.chkmonth:checked').each(function(idx, elm) {
                a.push($('.this').val());
            });
        }
        return (a == [] ? 0 : a);
    }
}

```

```

        }

    function get_search_param() {
        var month = get_checked_month();

        var param = {
            year : $('#id_year').val()
        };
        if (month == 0)
            param['month'] = 0;

        else
            param['month[]'] = month;

        return param;
    }

    function uncheck_all_month() {
        var a = $(this).attr('checked');
        if (a == 'checked')
            $('.chkmonth').removeAttr('checked');
    }

    function uncheck_all() {
        var a = $(this).attr('checked');
        if (a == 'checked')
            $('.chkall').removeAttr('checked');
    }

    function init() {
        $('#id_gen').click(draw_chart);
        $('.chkall').click(uncheck_all_month);
        $('.chkmonth').click(uncheck_all);
        utils.init_alert_dialog('#dialog-message');
        utils.bind_hover($('#id_gen'));
        draw_chart();
    }

    function load() {
        return menu.get('/user/workhours/chart/', init);
    }

    return {
        load : load
    };
}());

```

E:\workspace\payroll_rails\app\assets\javascripts\tpl\label_error.html

```

<div id="error_<%= field %>" class="ui-state-error"><%= msg %></div>

```

E:\workspace\payroll_rails\app\assets\javascripts\tpl\label_error_inline.html

```

<span id="error_<%= field %>" class="ui-state-error" style="margin-left: 10px"><%= msg %></span>

```

```

E:\workspace\payroll_rails\app\assets\stylesheets\_payslip.css

/*
*= require_self
*= require_base
* require blitzer/jquery_ui
*= require jquery.themes
*/



E:\workspace\payroll_rails\app\assets\stylesheets\admin.css

/*
 * This is a manifest file that'll be compiled into application.css, which will include all
the files
 * listed below.
 *
 * Any CSS and SCSS file within this directory, lib/assets/stylesheets,
vendor/assets/stylesheets,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative
path.
 *
 * You're free to add application-wide styles to this file and they'll appear at the top of
the
 * compiled file, but it's generally better to create a new file per style scope.
 *
*= require_self
*= require_application
* require_tree .
*/



E:\workspace\payroll_rails\app\assets\stylesheets\application.css

/*
 * This is a manifest file that'll be compiled into application.css, which will include all
the files
 * listed below.
 *
 * Any CSS and SCSS file within this directory, lib/assets/stylesheets,
vendor/assets/stylesheets,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative
path.
 *
 * You're free to add application-wide styles to this file and they'll appear at the top of
the
 * compiled file, but it's generally better to create a new file per style scope.
 *
*= require_self
* require blitzer/jquery_ui
*= require jquery.themes
*= require_base
* require_tree .
*/



E:\workspace\payroll_rails\app\assets\stylesheets\base.css.scss

body {
margin: 0;
padding: 0;
line-height: 1.5em;
font: 8pt Trebuchet MS, Tahoma, Verdana, Arial, sans-serif;
margin: 5px;
border: none !important;
}

#maincontainer {
/*width: 840px; /*Width of main container*/
}

#topsection {
height: 40px; /*Height of top section*/
/*width: 995px;*/
}

#contentwrapper {
float: left;
width: 100%;
}

```

```

#contentcolumn {
    margin-left: 200px; /*Set left margin to LeftColumnWidth*/
}

#leftcolumn {
    float: left;
    width: 200px; /*Width of left column*/
    /*margin-left: -840px; /*Set left margin to -(MainContainerWidth)*/
    margin-left: -100%;
}

.innertube {
    margin: 2px;
    /*Margins for inner DIV inside each column (to provide padding)*/
    margin-top: 0;
}

.top_right {
    float: right;
    clear: all;
    margin-right: 10px;
    margin-top: 10px;
    display: inline-block;
    width: 70px;
    padding-left: 10px;
}

#progress_status {
    width: 200px;
    position: absolute;
    top: -14px;
    left: 20px;
    display: none;
}

#menu a:LINK {
    text-decoration: none;
}

.menu_active {
    font-weight: bold;
}



```

```

width: 100%;
border-collapse: collapse;
}

.list_table .data {
text-align: justify;
}

.list_table .checkbox {
width: 50px;
}

.nav_table {
float: right;
border-collapse: collapse;
}

.nav_table td {
text-align: right;
}

.page_button {
text-align: center;
white-space: nowrap;
padding: 0 50px 2px 50px;
}

.go_button {
text-align: center;
white-space: nowrap;
padding: 0 15px 2px 15px;
display: inline-block;
}

#id_add, #id_delete {
width: 70px;
padding: 2px 0 2px 15px;
}

#id_find {
padding: 2px 17px 2px 3px;
}

#id_gen {
padding: 2px 45px 2px 3px;
}

.hover {
cursor: pointer;
}

.page_container {
width: 100%;
}

.space {
margin-top: 5px;
}

.link {
text-decoration: underline !important;
}

.text {
width: 200px;
}

.textarea {
width: 200px;
height: 100px;
}

.textarea2 {
width: 250px;
height: 50px;
}

```

```

.select {
  width: 202px;
  font-size: 1em;
}

.list_button {
  margin-top: 2px;
}

.save_button, .print_button {
  font-size: 1em;
  padding: 2px 8px 2px 8px;
  display: inline-block;
  width: 60px;
}

.save_button.save {
  margin-right: 5px;
}

div.row_button {
  clear: both;
  margin-right: 5px;
  padding-top: 12px;
  padding-bottom: 12px;
  float: left;
}

.save_form .lbl {
  text-align: right;
  white-space: nowrap;
}

.spinner {
  display: inline-block;
  background: url('/assets/loading.gif');
  width: 15px;
  height: 15px;
}

.label:after {
  content: ":";
}

.colheader,.sorticon {
  display: inline-block !important;
}

.colheader {
  height: 20px;
}

.aligncenter {
  text-align: center;
}

#main_options {
  float: right;
  margin-right: 20px;
  margin-top: 10px;
  display: inline-block;
  width: 70px;
  text-align: justify;
  padding-left: 10px;
}

.theme_options {
  display: none;
  height: 82px;
  position: absolute;
  top: 45px;
  /*left: 890px;*/
  left: 83%;
  z-index: 3000;
}

#status-panel {

```

```

width: 300px;
position: absolute;
top: -14px;
left: 20px;
display: none;
}

.spacing {
padding-left: 30px;
}

.colspacing {
padding-right: 50px;
}

input[type='text'], input[type='password'] {
border: 1px solid #ccc;
}

input[type='text']:HOVER, input[type='password']:HOVER {
border: 1px solid #000;
}

input[type='text']:FOCUS, input[type='password']:FOCUS {
background-color: #fffc;
}

/* start payroll css */

.heading, .subdet, .data_table th {
font-size: 1.2em;
}

.heading {
width: 100%;
}

.data_table {
width: 100%;
border: 1px solid;
border-collapse: collapse;
}

.data_table td {
font-size: 1.1em;
}

.data_table th, .data_table td {
border-right: 1px solid;
}

.data_table td.sum {
border-bottom: 1px solid;
}

.boldtext {
font-weight: bold;
}

.amt {
text-align: right;
}

.amt span {
margin-right: 15px;
}

.data_table span.det {
margin-left: 15px;
}

.print_container {
margin-top: 10px;
}

/* end payroll css */

```

```

.paddicon {
  padding-left: 18px;
}

.icons {
  background: url('/assets/icons.png') no-repeat !important;
  width: 16px;
  height: 16px;
  display: inline-block;
}

.addicon {
  background-position: 0 0 !important;
}
.deleteicon {
  background-position: 0 -21px !important;
}
.findicon {
  background-position: 0 -42px !important;
}
.saveicon {
  background-position: 0 -63px !important;
}
.cancelicon {
  background-position: 0 -84px !important;
}
.lefticon {
  background-position: 0 -105px !important;
}
.righticon {
  background-position: 0 -126px !important;
}
.accepticon {
  background-position: 0 -147px !important;
}
.printicon {
  background-position: 0 -168px !important;
}
.optionicon {
  background-position: 0 -189px !important;
}
.charticon {
  background-position: 0 -210px !important;
}
.detailicon {
  background-position: 0 -231px !important;
}
.logouticon {
  background-position: 0 -252px !important;
}
.loginicon {
  background-position: 0 -273px !important;
}

```

E:\workspace\payroll_rails\app\assets\stylesheets\login.css.scss

```

body {
  font: 11pt Trebuchet MS, Tahoma, Verdana, Arial, sans-serif;
  border: none !important;
}

form {
  margin-bottom: 0px;
}

.top_right {
  float: right;
  clear: all;
  margin-right: 10px;
  margin-top: 10px;
  display: inline-block;
}

.login_box {
  width: 500px;
  margin: 0 auto;
  padding: 10px;
}

```

```

margin-top: 12%;
background-image: none !important;
}

.login_box .msg {
text-align: center;
padding: 8px;
margin: 8px;
}

.editor-label:after {
content: ":";

}

.text-box {
width: 25em;
}

.login_btn {
padding-top: 10px;
padding-bottom: 10px;
width: 150px;
font: 12pt Trebuchet MS, Tahoma, Verdana, Arial, sans-serif bolder !important;
}

.login_btn:HOVER {
border: 1px groove #ccc;
}

div.row {
clear: both;
padding-top: 5px;
margin-left: 5px;
}

div.row label {
float: left;
text-align: right;
margin-top: 5px;
margin-right: 5px;
width: 100px;
}

.hover {
cursor: pointer;
}

.notice {
border: 3px solid #000000;
padding: 8px;
margin: 16px;
text-align: center;
}

.errorblock {
border: 3px solid #ff0000;
padding: 8px;
margin: 16px;
text-align: center;
}

input[type='text'], input[type='password'], textarea {
border: 1px solid #ccc;
}

input[type='text']:HOVER, input[type='password']:HOVER, textarea:HOVER {
border: 1px solid #000;
}

input[type='text']:FOCUS, input[type='password']:FOCUS, textarea:FOCUS {
background-color: #ffc;
}

```

```
E:\workspace\payroll_rails\app\assets\stylesheets\loginui.css

/*
 *= require_self
 *= require dark-hive/jquery_ui
 *= require login
 *= require_tree .
 */

E:\workspace\payroll_rails\app\assets\stylesheets\user.css

/*
 * This is a manifest file that'll be compiled into application.css, which will include all
the files
 * listed below.
 *
 * Any CSS and SCSS file within this directory, lib/assets/stylesheets,
vendor/assets/stylesheets,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative
path.
 *
 * You're free to add application-wide styles to this file and they'll appear at the top of
the
 * compiled file, but it's generally better to create a new file per style scope.
 *
 *= require_self
 *= require application
 *= require_tree .
 */
```

```

E:\workspace\payroll_rails\test\fixtures\attendance.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  staff_id: C0001
  work_date: 2013-01-12
  time_in: 2013-01-12 23:18:51
  time_out: 2013-01-12 23:18:51

two:
  id: 2
  staff_id: C0002
  work_date: 2013-01-12
  time_in: 2013-01-12 23:18:51
  time_out: 2013-01-12 23:18:51

E:\workspace\payroll_rails\test\fixtures\department.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  name: R & D

two:
  id: 2
  name: Marketing

E:\workspace\payroll_rails\test\fixtures\designation.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  title: Programmer
  desc: Software development
  note: Programming job

two:
  id: 2
  title: Admin
  desc: Admin work
  note: Admin job

E:\workspace\payroll_rails\test\fixtures\employee.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  staff_id: C0001
  first_name: ben
  middle_name: ong
  last_name: mun
  new_ic: 554498824
  old_ic:
  passport_no: 12889100025
  gender: M
  marital_status: S
  nationality: Malaysian
  dob: 2012-12-17
  place_of_birth: Serdang
  race: Chinese
  religion: Christian
  is_bumi: false
  user_id: 2

two:
  id: 2
  staff_id: C0002
  first_name: ken
  middle_name: looi
  last_name: kin
  new_ic: 7654399054
  old_ic:

```

```

passport_no: 560083200
gender: F
marital_status: O
nationality: Malaysian
dob: 2012-12-17
place_of_birth: Puchong
race: Chinese
religion: Buddhist
is_bumi: false
user_id: 3

E:\workspace\payroll_rails\test\fixtures\employee_contact.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  address_1: 8
  address_2: Jalan Puteri
  address_3: Taman Puteri
  city: Kuala Lumpur
  state: W.P
  postcode: 54000
  country: Malaysia
  home_phone: 038766056
  mobile_phone: 0123398765
  work_email: ken@gmail.com
  other_email: ken@yahoo.com

two:
  id: 2
  address_1: 9
  address_2: Jalan Maju
  address_3: Taman Bahagia
  city: Kuala Lumpur
  state: W.P
  postcode: 56000
  country: Malaysia
  home_phone: 038876599
  mobile_phone: 0128744522
  work_email: ben@gmail.com
  other_email: ben@yahoo.net

E:\workspace\payroll_rails\test\fixtures\employee_job.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  designation_id: 1
  department_id: 1
  employment_status_id: 1
  job_category_id: 1
  join_date: 2012-12-17
  confirm_date: 2012-12-17

two:
  id: 2
  designation_id: 2
  department_id: 2
  employment_status_id: 1
  job_category_id: 2
  join_date: 2012-12-17
  confirm_date: 2012-12-17

E:\workspace\payroll_rails\test\fixtures\employee_qualification.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  level: 4
  institute: APIT
  major: IT
  year: 2009
  gpa: 3.4
  start_date: 2004-05-06

```

```

end_date: 2006-05-09

two:
  id: 2
  level: 4
  institute: Sedaya
  major: Engineering
  year: 2008
  gpa: 3.7
  start_date: 2003-03-01
  end_date: 2005-05-10

E:\workspace\payroll_rails\test\fixtures\employee_salary.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  salary: 4200
  allowance: 20
  epf: 350
  socso: 10
  bank_name: Maybank
  bank_acc_no: 0956644
  bank_acc_type: Savings
  bank_address: Serdang
  epf_no: 776499
  socso_no: 9954788
  income_tax_no: 0096734567
  pay_type: 1

two:
  id: 2
  salary: 3400
  allowance: 50
  epf: 300
  socso: 30
  bank_name: Public
  bank_acc_no: 88650066
  bank_acc_type: Savings
  bank_address: Puchong
  epf_no: 560099
  socso_no: 5930033
  income_tax_no: 0958467443
  pay_type: 2

E:\workspace\payroll_rails\test\fixtures\employment_status.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  name: Probation

two:
  id: 2
  name: Terminated

E:\workspace\payroll_rails\test\fixtures\job_category.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  name: Marketing

two:
  id: 2
  name: Sales

E:\workspace\payroll_rails\test\fixtures\overtime_rate.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  duration: 1.5
  year: 2010

```

```

pay_rate: 1.5

two:
  duration: 1.5
  year: 2008
  pay_rate: 1.5

E:\workspace\payroll_rails\test\fixtures\pay_rate.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  staff_id: C0001
  month: 1
  year: 2011
  hourly_pay_rate: 1.5

two:
  id: 2
  staff_id: C0002
  month: 2
  year: 2011
  hourly_pay_rate: 1.5

E:\workspace\payroll_rails\test\fixtures\salary_adjustment.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

one:
  id: 1
  staff_id: C0001
  inc: 1.5
  month: 1
  year: 2013

two:
  id: 2
  staff_id: C0002
  inc: 1.5
  month: 1
  year: 2013

E:\workspace\payroll_rails\test\fixtures\user.yml

# Read about fixtures at http://api.rubyonrails.org/classes/ActiveRecord/Fixtures.html

admin:
  id: 1
  role: 1
  username: admin
  status: true
  password: e5e9falba31ecd1ae84f75caaa474f3a663f05f4 # secret

ben:
  id: 2
  role: 2
  username: ben
  status: true
  password: e5e9falba31ecd1ae84f75caaa474f3a663f05f4 # secret

ken:
  id: 3
  role: 2
  username: ken
  status: true
  password: e5e9falba31ecd1ae84f75caaa474f3a663f05f4 # secret

```

```

E:\workspace\payroll_rails\test\test_helper.rb

ENV["RAILS_ENV"] = "test"
require File.expand_path('../config/environment', __FILE__)
require 'rails/test_help'

class ActiveSupport::TestCase
  # Setup all fixtures in test/fixtures/*.yml|csv for all tests in alphabetical order.
  #
  # Note: You'll currently still have to declare fixtures explicitly in integration tests
  # -- they do not yet inherit this setting
  self.use_transactional_fixtures = true
  set_fixture_class :employment_status => 'EmploymentStatus'
  fixtures :all

  # Add more helper methods to be used by all tests here...
  # Add more helper methods to be used by all tests here...
  def login_as(user)
    user = user(user)
    @request.session[:user_id] = user.id
    if user.role != User::ADMIN
      employee = user.employee
      @request.session[:employee_id] = employee.id
      @request.session[:staff_id] = employee.staff_id
      @request.session[:supervisor_id] = employee.id
    end
  end

  def normal_user(test_module)
    open_session do |user|
      def user.logs_in(username, password)
        get login_path
        assert response :success
        assert_template 'new'

        post_via_redirect auth_path, :username => username, :password => password

        assert_response :success
        assert_equal user_index_path, path
        assert_template 'index'
        assert_not_nil session[:user_id]
        assert_not_nil session[:employee_id]
        assert_not_nil session[:staff_id]
        assert_not_nil session[:supervisor_id]
      end

      def user.logs_out
        get_via_redirect logout_path
        assert_response :success
        assert_equal login_path, path
        assert_template 'new'
        assert_nil session[:user_id]
        assert_nil session[:employee_id]
        assert_nil session[:staff_id]
        assert_nil session[:supervisor_id]
      end

      user.extend(test_module)
      yield user if block_given?
    end
  end

  def admin_user(test_module)
    open_session do |user|
      def user.logs_in(username, password)
        get login_path
        assert response :success
        assert_template 'new'

        post_via_redirect auth_path, :username => username, :password => password

        assert_response :success
        assert_equal admin_index_path, path
        assert_template 'index'
        assert_not_nil session[:user_id]
      end
    end
  end

```

```

def user.logs_out
  get via redirect logout path
  assert_response :success
  assert_equal login_path, path
  assert_template 'new'
  assert_nil session[:user_id]
end

user.extend(test_module)
yield user if block_given?
end
end
end

E:\workspace\payroll_rails\test\unit\attendance_test.rb

require 'test_helper'

class AttendanceTest < ActiveSupport::TestCase

  test 'should create attendance' do
    o = Attendance.new
    o.id = 3
    o.staff_id = 'S0007'
    o.time_in = Time.new(2009, 4, 3, 8, 34, 0, '+08:00')
    o.time_out = Time.new(2009, 4, 3, 18, 34, 0, '+08:00')
    o.work_date = Date.new(2009, 4, 3)

    assert o.save
  end

  test 'should find attendance' do
    id = attendance(:one).id
    assert_nothing_raised { Attendance.find(id) }
  end

  test 'should update attendance' do
    o = attendance(:two)
    assert o.update_attributes(:staff_id => 'S0008')

    o.work_date = Date.new(2010, 6, 8)
    assert o.save
    o = Attendance.find(o.id)
    assert_equal '2010-06-08', o.work_date.strftime('%Y-%m-%d')
  end

  test 'should destroy attendance' do
    o = attendance(:one)
    o.destroy
    assert_raise(ActiveRecord::RecordNotFound) { Attendance.find(o.id) }
  end

  test 'should not create a attendance without required fields' do
    o = Attendance.new
    assert !o.valid?
    assert o.errors[:work_date].any?
    assert o.errors[:time_in].any?
    assert o.errors[:time_out].any?

    assert_equal ['Work date is required'], o.errors[:work_date]
    assert_equal ['Time in is required'], o.errors[:time_in]
    assert_equal ['Time out is required'], o.errors[:time_out]
  end
end

E:\workspace\payroll_rails\test\unit\department_test.rb

require 'test_helper'

class DepartmentTest < ActiveSupport::TestCase

  test 'should create department' do
    o = Department.new
    o.name = 'Account'
  end

```

```

        assert o.save
end

test 'should find department' do
  id = department(:one).id
  assert_nothing_raised { Department.find(id) }
end

test 'should update department' do
  o = department(:two)
  assert o.update_attributes(:name => 'QC')

  o.name = 'Support'
  assert o.save
  o = Department.find(o.id)
  assert_equal 'Support', o.name
end

test 'should destroy department' do
  o = department(:one)
  o.destroy
  assert_raise(ActiveRecord::RecordNotFound) { Department.find(o.id) }
end

test 'should not create a department without required fields' do
  o = Department.new
  assert !o.valid?
  assert o.errors[:name].any?

  assert_equal ['Name is required'], o.errors[:name]
end

test 'should not create a department with duplicate name' do
  o = Department.new
  o.name = 'Marketing'
  assert !o.valid?
  assert o.errors[:name].any?

  assert_equal ['Department Marketing already exist'], o.errors[:name]
end
end

E:\workspace\payroll_rails\test\unit\designation_test.rb

require 'test_helper'

class DesignationTest < ActiveSupport::TestCase

  test 'should create designation' do
    o = Designation.new
    o.title = 'Manager'

    assert o.save
end

  test 'should find designation' do
    id = designation(:one).id
    assert_nothing_raised { Designation.find(id) }
end

  test 'should update designation' do
    o = designation(:two)
    assert o.update_attributes(:title => 'Admin')

    o.title = 'Account'
    assert o.save
    o = Designation.find(o.id)
    assert_equal 'Account', o.title
end

  test 'should destroy designation' do
    o = designation(:one)
    o.destroy
    assert_raise(ActiveRecord::RecordNotFound) { Designation.find(o.id) }
end

  test 'should not create a designation without required fields' do

```

```

o = Designation.new
assert !o.valid?
assert o.errors[:title].any?

  assert_equal ['Job Title is required'], o.errors[:title]
end

test 'should not create a designation with duplicate title' do
  o = Designation.new
  o.title = 'Admin'
  assert !o.valid?
  assert o.errors[:title].any?

  assert_equal ['Job Title Admin already exist'], o.errors[:title]
end
end

E:\workspace\payroll_rails\test\unit\employee_contact_test.rb

require 'test_helper'

class EmployeeContactTest < ActiveSupport::TestCase

  test 'should create employee_contact' do
    o = EmployeeContact.new
    o.id = 3
    o.address_1 = 'Jalan Maju'
    o.address_2 = 'Blok 6'
    o.address_3 = 'Taman Gembira'
    o.city = 'KL'
    o.state = 'WP'
    o.postcode = '56000'
    o.country = 'Malaysia'
    o.home_phone = '666'
    o.mobile_phone = '777'
    o.work_email = 'test@gmail.com'
    o.other_email = 'home@gmail.com'

    assert o.save
  end

  test 'should find employee_contact' do
    id = employee_contact(:one).id
    assert_nothing_raised { EmployeeContact.find(id) }
  end

  test 'should update employee_contact' do
    o = employee_contact(:two)
    assert o.update_attributes(:address_1 => 'Jalan Rasa', :address_2 => 'Taman Kerinchi',
                               :address_3 => 'Taman Pinang')

    o.state = 'Selangor'
    assert o.save
    o = EmployeeContact.find(o.id)
    assert_equal 'Selangor', o.state
    assert_equal 'Taman Pinang', o.address_3
  end

  test 'should destroy employee_contact' do
    o = employee_contact(:one)
    o.destroy
    assert_raise(ActiveRecord::RecordNotFound) { EmployeeContact.find(o.id) }
  end

  test 'should not create a employee_contact without required fields' do
    o = EmployeeContact.new
    assert !o.valid?
    assert o.errors[:address_1].any?
    assert o.errors[:city].any?
    assert o.errors[:state].any?
    assert o.errors[:postcode].any?
    assert o.errors[:country].any?
    assert o.errors[:work_email].any?

    assert_equal ['Address 1 is required'], o.errors[:address_1]
    assert_equal ['City is required'], o.errors[:city]
    assert_equal ['State is required'], o.errors[:state]
  end

```

```

    assert_equal ['Postal Code is required'], o.errors[:postcode]
    assert_equal ['Country is required'], o.errors[:country]
    assert_equal ['Work Email is required'], o.errors[:work_email]
end
end

E:\workspace\payroll_rails\test\unit\employee_job_test.rb

require 'test_helper'

class EmployeeJobTest < ActiveSupport::TestCase

test 'should create employee_job' do
  o = EmployeeJob.new
  o.id = 8
  o.confirm_date = Date.new(2010, 7, 1)
  o.department_id = 1
  o.designation_id = 2
  o.employment_status_id = 3
  o.job_category_id = 4
  o.join_date = Date.new(2010, 6, 1)

  assert o.save
end

test 'should find employee_job' do
  id = employee_job(:one).id
  assert_nothing_raised { EmployeeJob.find(id) }
end

test 'should update employee_job' do
  o = employee_job(:two)
  assert o.update_attributes(:confirm_date => Date.new(2009, 5, 4), :department_id => 5,
                            :designation_id => 4, :employment_status_id => 6,
                            :job_category_id => 4, :join_date => Date.new(2009, 2, 6))

  o.join_date = Date.new(2009, 2, 3)
  assert o.save
  o = EmployeeJob.find(o.id)
  assert_equal '2009-02-03', o.join_date.strftime('%Y-%m-%d')
end

test 'should destroy employee_job' do
  o = employee_job(:one)
  o.destroy
  assert_raise(ActiveRecord::RecordNotFound) { EmployeeJob.find(o.id) }
end

test 'should not create a employee_job without required fields' do
  o = EmployeeJob.new
  assert !o.valid?
  assert o.errors[:designation_id].any?
  assert o.errors[:department_id].any?
  assert o.errors[:employment_status_id].any?
  assert o.errors[:job_category_id].any?
  assert o.errors[:join_date].any?

  assert_equal ['Designation is required'], o.errors[:designation_id]
  assert_equal ['Department is required'], o.errors[:department_id]
  assert_equal ['Employment Status is required'], o.errors[:employment_status_id]
  assert_equal ['Job Category is required'], o.errors[:job_category_id]
  assert_equal ['Join Date is required'], o.errors[:join_date]
end
end

E:\workspace\payroll_rails\test\unit\employee_qualification_test.rb

require 'test_helper'

class EmployeeQualificationTest < ActiveSupport::TestCase

test 'should create employee_qualification' do
  o = EmployeeQualification.new
  o.id = 3
  o.level = 4
  o.institute = 'Informatics'
  o.major = 'Computer Science'

```

```

    o.year = 2005
    o.gpa = 3.0
    o.start_date = Date.new(2005, 8, 6)
    o.end_date = Date.new(2007, 3, 7)

    assert o.save
end

test 'should find employee_qualification' do
  id = employee_qualification(:one).id
  assert_nothing_raised { EmployeeQualification.find(id) }
end

test 'should update employee_qualification' do
  o = employee_qualification(:two)
  assert o.update_attributes(:level => 5)

  o.institute = 'UPM'
  assert o.save
  o = EmployeeQualification.find(o.id)
  assert_equal 'UPM', o.institute
end

test 'should destroy employee_qualification' do
  o = employee_qualification(:one)
  o.destroy
  assert_raise(ActiveRecord::RecordNotFound) { EmployeeQualification.find(o.id) }
end

test 'should not create a department without required fields' do
  o = EmployeeQualification.new
  assert !o.valid?
  assert o.errors[:level].any?
  assert o.errors[:institute].any?
  assert o.errors[:year].any?
  assert o.errors[:start_date].any?
  assert o.errors[:end_date].any?

  assert_equal ['Qualification Level is required'], o.errors[:level]
  assert_equal ['Institute name is required'], o.errors[:institute]
  assert_equal ['Year obtained is required'], o.errors[:year]
  assert_equal ['Start Date is required'], o.errors[:start_date]
  assert_equal ['End Date is required'], o.errors[:end_date]
end
end

```

E:\workspace\payroll_rails\test\unit\employee_salary_test.rb

```

require 'test_helper'

class EmployeeSalaryTest < ActiveSupport::TestCase

  test 'should create employee_salary' do
    o = EmployeeSalary.new
    o.id = 3
    o.salary = 5000
    o.allowance = 40
    o.epf = 67
    o.socso = 88
    o.income_tax = 89
    o.bank_name = 'cimb'
    o.bank_acc_no = '67886554'
    o.bank_acc_type = 'savings'
    o.bank_address = 'klcc'
    o.epf_no = '4478966'
    o.socso_no = '5590876'
    o.income_tax_no = '667546'
    o.pay_type = 1

    assert o.save
  end

  test 'should find employee_salary' do
    id = employee_salary(:one).id
    assert_nothing_raised { EmployeeSalary.find(id) }
  end
end

```

```

test 'should update employee_salary' do
  o = employee_salary(:two)
  assert o.update_attributes(:salary => 5500, :allowance => 56)

  o.bank_name = 'rhb'
  assert o.save
  o = EmployeeSalary.find(o.id)
  assert_equal 'rhb', o.bank_name
end

test 'should destroy employee_salary' do
  o = employee_salary(:one)
  o.destroy
  assert_raise(ActiveRecord::RecordNotFound) { EmployeeSalary.find(o.id) }
end

test 'should not create a employee_salary without required fields' do
  o = EmployeeSalary.new
  o.salary = -90
  o.epf = -88
  assert !o.valid?
  assert o.errors[:salary].any?
  assert o.errors[:bank_name].any?
  assert o.errors[:bank_acc_no].any?
  assert o.errors[:bank_acc_type].any?
  assert o.errors[:bank_address].any?
  assert o.errors[:epf_no].any?
  assert o.errors[:pay_type].any?
  assert o.errors[:epf].any?

  assert_equal ['Salary is invalid'], o.errors[:salary]
  assert_equal ['Bank Name is required'], o.errors[:bank_name]
  assert_equal ['Bank Account No. is required'], o.errors[:bank_acc_no]
  assert_equal ['Bank Account Type is required'], o.errors[:bank_acc_type]
  assert_equal ['Bank Address is required'], o.errors[:bank_address]
  assert_equal ['EPF No. is required'], o.errors[:epf_no]
  assert_equal ['Pay Type is required'], o.errors[:pay_type]
  assert_equal ['EPF Deduction is invalid'], o.errors[:epf]
end
end

E:\workspace\payroll_rails\test\unit\employee_test.rb

require 'test_helper'

class EmployeeTest < ActiveSupport::TestCase

  test 'should create employee' do
    o = Employee.new
    o.id = '0007'
    o.staff_id = 'S0007'
    o.first_name = 'Ken'
    o.middle_name = 'Chua'
    o.last_name = 'Leong'
    o.new_ic = '887760489'
    o.gender = 'M'
    o.marital_status = 'S'
    o.nationality = 'Malaysian'
    o.dob = Date.new(1987, 9, 6)
    o.place_of_birth = 'PJ'
    o.race = 'Chinese'
    o.religion = 'Buddhist'
    o.is_bumi = false
    o.user_id = 2

    assert o.save
  end

  test 'should find employee' do
    id = employee(:one).id
    assert_nothing_raised { Employee.find(id) }
  end

  test 'should update employee' do
    o = employee(:two)
    assert o.update_attributes(:new_ic => '776593367', :last_name => 'Paul')
  end

```

```

    o.first_name = 'Jerry'
    assert o.save
    o = Employee.find(o.id)
    assert_equal 'Jerry', o.first_name
end

test 'should destroy employee' do
  o = employee(:one)
  o.destroy
  assert_raise(ActiveRecord::RecordNotFound) { Employee.find(o.id) }
end

test 'should not create a employee without required fields' do
  o = Employee.new
  assert !o.valid?
  assert o.errors[:staff_id].any?
  assert o.errors[:first_name].any?
  assert o.errors[:last_name].any?
  assert o.errors[:new_ic].any?
  assert o.errors[:gender].any?
  assert o.errors[:marital_status].any?
  assert o.errors[:nationality].any?
  assert o.errors[:dob].any?
  assert o.errors[:place_of_birth].any?
  assert o.errors[:race].any?

  assert_equal ['Employee ID is required'], o.errors[:staff_id]
  assert_equal ['First Name is required'], o.errors[:first_name]
  assert_equal ['Last Name is required'], o.errors[:last_name]
  assert_equal ['New IC No. is required'], o.errors[:new_ic]
  assert_equal ['Gender is required'], o.errors[:gender]
  assert_equal ['Marital Status is required'], o.errors[:marital_status]
  assert_equal ['Nationality is required'], o.errors[:nationality]
  assert_equal ['Date of Birth is required'], o.errors[:dob]
  assert_equal ['Place of Birth is required'], o.errors[:place_of_birth]
  assert_equal ['Race is required'], o.errors[:race]
end

test 'should not create a employee with duplicate staff_id' do
  o = Employee.new
  o.staff_id = 'C0002'
  assert !o.valid?
  assert o.errors[:staff_id].any?

  assert_equal ['Employee ID C0002 already exist'], o.errors[:staff_id]
end
end

E:\workspace\payroll_rails\test\unit\employment_status_test.rb

require 'test_helper'

class EmploymentStatusTest < ActiveSupport::TestCase

  test 'should create employment_status' do
    o = EmploymentStatus.new
    o.name = 'Confirmed'

    assert o.save
  end

  test 'should find employment_status' do
    id = employment_status(:one).id
    assert_nothing_raised { EmploymentStatus.find(id) }
  end

  test 'should update employment_status' do
    o = employment_status(:two)
    assert o.update_attributes(:name => 'Temporary')

    o.name = 'Contract'
    assert o.save
    o = EmploymentStatus.find(o.id)
    assert_equal 'Contract', o.name
  end

  test 'should destroy employment_status' do

```

```

    o = employment_status(:one)
    o.destroy
    assert_raise(ActiveRecord::RecordNotFound) { EmploymentStatus.find(o.id) }
end

test 'should not create a employment_status without required fields' do
  o = EmploymentStatus.new
  assert !o.valid?
  assert o.errors[:name].any?

  assert_equal ['Name is required'], o.errors[:name]
end

test 'should not create a employment_status with duplicate name' do
  o = EmploymentStatus.new
  o.name = 'Probation'
  assert !o.valid?
  assert o.errors[:name].any?

  assert_equal ['Employment Status Probation already exist'], o.errors[:name]
end
end

E:\workspace\payroll_rails\test\unit\job_category_test.rb

require 'test_helper'

class JobCategoryTest < ActiveSupport::TestCase

  test 'should create job_category' do
    o = JobCategory.new
    o.name = 'Technical'

    assert o.save
  end

  test 'should find job_category' do
    id = job_category(:one).id
    assert_nothing_raised { JobCategory.find(id) }
  end

  test 'should update job_category' do
    o = job_category(:two)
    assert o.update_attributes(:name => 'QC')

    o.name = 'Support'
    assert o.save
    o = JobCategory.find(o.id)
    assert_equal 'Support', o.name
  end

  test 'should destroy job_category' do
    o = job_category(:one)
    o.destroy
    assert_raise(ActiveRecord::RecordNotFound) { JobCategory.find(o.id) }
  end

  test 'should not create a job_category without required fields' do
    o = JobCategory.new
    assert !o.valid?
    assert o.errors[:name].any?

    assert_equal ['Name is required'], o.errors[:name]
  end

  test 'should not create a job_category with duplicate name' do
    o = JobCategory.new
    o.name = 'Marketing'
    assert !o.valid?
    assert o.errors[:name].any?

    assert_equal ['Category Marketing already exist'], o.errors[:name]
  end
end

```

```

E:\workspace\payroll_rails\test\unit\overtime_rate_test.rb

require 'test_helper'

class OvertimeRateTest < ActiveSupport::TestCase

  test 'should create overtime_rate' do
    o = OvertimeRate.new
    o.duration = 2
    o.year = 2009
    o.pay_rate = 89

    assert o.save
  end

  test 'should find overtime_rate' do
    id = overtime_rate(:one).id
    assert_nothing_raised { OvertimeRate.find(id) }
  end

  test 'should update overtime_rate' do
    o = overtime_rate(:two)
    assert o.update_attributes(:duration => 3, :year => 2007, :pay_rate => 86)

    o.duration = 1
    assert o.save
    o = OvertimeRate.find(o.id)
    assert_equal 1, o.duration
  end

  test 'should destroy overtime_rate' do
    o = overtime_rate(:one)
    o.destroy
    assert_raise(ActiveRecord::RecordNotFound) { OvertimeRate.find(o.id) }
  end

  test 'should not create a overtime_rate without required fields' do
    o = OvertimeRate.new
    o.duration = -8
    o.pay_rate = 0
    assert !o.valid?
    assert o.errors[:duration].any?
    assert o.errors[:year].any?
    assert o.errors[:pay_rate].any?

    assert_equal ['Duration is invalid'], o.errors[:duration]
    assert_equal ['Year is required', 'Year is invalid'], o.errors[:year]
    assert_equal ['Pay Rate is invalid'], o.errors[:pay_rate]
  end

  test 'should not create a overtime_rate with duplicate year' do
    o = OvertimeRate.new
    o.year = 2008
    assert !o.valid?
    assert o.errors[:year].any?

    assert_equal ['Overtime rate for year 2008 already exist'], o.errors[:year]
  end
end

```

E:\workspace\payroll_rails\test\unit\pay_rate_test.rb

```

require 'test_helper'

class PayRateTest < ActiveSupport::TestCase

  test 'should create pay_rate' do
    o = PayRate.new
    o.id = 3
    o.staff_id = 'S0009'
    o.month = 8
    o.year = 2009
    o.hourly_pay_rate = 67

    assert o.save
  end

```

```

test 'should find pay_rate' do
  id = pay_rate(:one).id
  assert_nothing_raised { PayRate.find(id) }
end

test 'should update pay_rate' do
  o = pay_rate(:two)
  assert o.update_attributes(:month => 7, :year => 2006, :hourly_pay_rate => 44)

  o.month = 6
  assert o.save
  o = PayRate.find(o.id)
  assert_equal 6, o.month
end

test 'should destroy pay_rate' do
  o = pay_rate(:one)
  o.destroy
  assert_raise(ActiveRecord::RecordNotFound) { PayRate.find(o.id) }
end

test 'should not create a pay_rate without required fields' do
  o = PayRate.new
  assert !o.valid?
  assert o.errors[:staff_id].any?
  assert o.errors[:month].any?
  assert o.errors[:year].any?

  assert_equal ['Staff ID is required'], o.errors[:staff_id]
  assert_equal ['Month is required', 'Month is invalid', 'Month is invalid'], o.errors[:month]
  assert_equal ['Year is required', 'Year is invalid'], o.errors[:year]

  o.month = 0
  o.year = 0
  o.hourly_pay_rate = 0
  assert !o.valid?

  assert_equal ['Month is invalid'], o.errors[:month]
  assert_equal ['Year is invalid'], o.errors[:year]
  assert_equal ['Hourly pay rate is invalid'], o.errors[:hourly_pay_rate]
end
end

```

```

E:\workspace\payroll_rails\test\unit\route_test.rb

require 'test_helper'

class RouteTest < ActionController::TestCase

  test 'should route to auth' do
    assert_routing '/login', { :controller => 'application', :action => 'new' }
    assert_routing '/auth', { :controller => 'application', :action => 'create' }
    assert_routing '/logout', { :controller => 'application', :action => 'destroy' }
  end

  test 'should route to admin' do
    assert_routing '/', { :controller => 'admin/admin', :action => 'index' }

    assert_routing '/admin/user', { :controller => 'admin/user', :action => 'index' }
    assert_routing '/admin/user/list', { :controller => 'admin/user', :action => 'list' }
    assert_routing '/admin/user/new', { :controller => 'admin/user', :action => 'new' }
    assert_routing({ :method => :post, :path => '/admin/user/create' },
                  { :controller => 'admin/user', :action => 'create' })
    assert_routing '/admin/user/edit/1', { :controller => 'admin/user', :action => 'edit', :id => '1' }
    assert_routing({ :method => :post, :path => '/admin/user/update' },
                  { :controller => 'admin/user', :action => 'update' })
    assert_routing({ :method => :post, :path => '/admin/user/delete' },
                  { :controller => 'admin/user', :action => 'destroy' })

    assert_routing '/admin/employee', { :controller => 'admin/employee', :action => 'index' }
    assert_routing '/admin/employee/list', { :controller => 'admin/employee', :action => 'list' }
    assert_routing '/admin/employee/new', { :controller => 'admin/employee', :action => 'new' }
  end

  assert_routing({ :method => :post, :path => '/admin/employee/create' },

```

```

        { :controller => 'admin/employee', :action => 'create' })
    assert_routing '/admin/employee/edit/1', { :controller => 'admin/employee', :action =>
'edit', :id => '1' }
    assert_routing({ :method => :post, :path => '/admin/employee/update' },
                  { :controller => 'admin/employee', :action => 'update' })
    assert_routing({ :method => :post, :path => '/admin/employee/delete'},
                  { :controller => 'admin/employee', :action => 'destroy' })

    assert_routing '/admin/designation', { :controller => 'admin/designation', :action =>
'index' }
    assert_routing '/admin/designation/list', { :controller => 'admin/designation', :action =>
'list' }
    assert_routing '/admin/designation/new', { :controller => 'admin/designation', :action =>
'new' }
    assert_routing({ :method => :post, :path => '/admin/designation/create' },
                  { :controller => 'admin/designation', :action => 'create' })
    assert_routing '/admin/designation/edit/1', { :controller => 'admin/designation', :action =>
'edit', :id => '1' }
    assert_routing({ :method => :post, :path => '/admin/designation/update' },
                  { :controller => 'admin/designation', :action => 'update' })
    assert_routing({ :method => :post, :path => '/admin/designation/delete'},
                  { :controller => 'admin/designation', :action => 'destroy' })

    assert_routing '/admin/empstatus', { :controller => 'admin/employment_status', :action =>
'index' }
    assert_routing '/admin/empstatus/list', { :controller => 'admin/employment_status',
:action => 'list' }
    assert_routing '/admin/empstatus/new', { :controller => 'admin/employment_status', :action =>
'new' }
    assert_routing({ :method => :post, :path => '/admin/empstatus/create' },
                  { :controller => 'admin/employment_status', :action => 'create' })
    assert_routing '/admin/empstatus/edit/1', { :controller => 'admin/employment_status',
:action => 'edit', :id => '1' }
    assert_routing({ :method => :post, :path => '/admin/empstatus/update' },
                  { :controller => 'admin/employment_status', :action => 'update' })
    assert_routing({ :method => :post, :path => '/admin/empstatus/delete'},
                  { :controller => 'admin/employment_status', :action => 'destroy' })

    assert_routing '/admin/jobcat', { :controller => 'admin/job_category', :action => 'index'
}
    assert_routing '/admin/jobcat/list', { :controller => 'admin/job_category', :action =>
'list' }
    assert_routing '/admin/jobcat/new', { :controller => 'admin/job_category', :action =>
'new' }
    assert_routing({ :method => :post, :path => '/admin/jobcat/create' },
                  { :controller => 'admin/job_category', :action => 'create' })
    assert_routing '/admin/jobcat/edit/1', { :controller => 'admin/job_category', :action =>
'edit', :id => '1' }
    assert_routing({ :method => :post, :path => '/admin/jobcat/update' },
                  { :controller => 'admin/job_category', :action => 'update' })
    assert_routing({ :method => :post, :path => '/admin/jobcat/delete'},
                  { :controller => 'admin/job_category', :action => 'destroy' })

    assert_routing '/admin/dept', { :controller => 'admin/department', :action => 'index' }
    assert_routing '/admin/dept/list', { :controller => 'admin/department', :action => 'list'
}
    assert_routing '/admin/dept/new', { :controller => 'admin/department', :action => 'new' }
    assert_routing({ :method => :post, :path => '/admin/dept/create' },
                  { :controller => 'admin/department', :action => 'create' })
    assert_routing '/admin/dept/edit/1', { :controller => 'admin/department', :action =>
'edit', :id => '1' }
    assert_routing({ :method => :post, :path => '/admin/dept/update' },
                  { :controller => 'admin/department', :action => 'update' })
    assert_routing({ :method => :post, :path => '/admin/dept/delete'},
                  { :controller => 'admin/department', :action => 'destroy' })

    assert_routing '/admin/payrate', { :controller => 'admin/pay_rate', :action => 'index' }
    assert_routing '/admin/payrate/list', { :controller => 'admin/pay_rate', :action => 'list'
}
    assert_routing '/admin/payrate/new', { :controller => 'admin/pay_rate', :action => 'new' }
    assert_routing({ :method => :post, :path => '/admin/payrate/create' },
                  { :controller => 'admin/pay_rate', :action => 'create' })
    assert_routing '/admin/payrate/edit/1', { :controller => 'admin/pay_rate', :action =>
'edit', :id => '1' }
    assert_routing({ :method => :post, :path => '/admin/payrate/update' },
                  { :controller => 'admin/pay_rate', :action => 'update' })
    assert_routing({ :method => :post, :path => '/admin/payrate/delete'},
                  { :controller => 'admin/pay_rate', :action => 'destroy' })

```

```

        { :controller => 'admin/pay_rate', :action => 'destroy' })

      assert_routing '/admin/hourly/chart', { :controller => 'admin/hourly_payroll_chart',
:action => 'index' }
      assert_routing '/admin/hourly/chart/data', { :controller => 'admin/hourly_payroll_chart',
:action => 'data' }

      assert_routing 'admin/payslip', { :controller => 'admin/payslip', :action => 'index' }
      assert_routing 'admin/payslip/list', { :controller => 'admin/payslip', :action => 'list' }
      assert_routing 'admin/payslip/slip/1/2/2009', { :controller => 'admin/payslip', :action =>
'payslip',
                                         :id => '1', :month => '2', :year => '2009'
}

assert_routing 'admin/att', { :controller => 'admin/attendance', :action => 'index' }
assert_routing 'admin/att/list', { :controller => 'admin/attendance', :action => 'list' }

      assert_routing '/admin/overtime/rate', { :controller => 'admin/overtime_rate', :action =>
'index' }
      assert_routing '/admin/overtime/rate/list', { :controller => 'admin/overtime_rate',
:action => 'list' }
      assert_routing '/admin/overtime/rate/new', { :controller => 'admin/overtime_rate',
:action => 'new' }
      assert_routing({ :method => :post, :path => '/admin/overtime/rate/create' },
{ :controller => 'admin/overtime_rate', :action => 'create' })
      assert_routing '/admin/overtime/rate/edit/1', { :controller => 'admin/overtime_rate',
:action => 'edit', :id => '1' }
      assert_routing({ :method => :post, :path => '/admin/overtime/rate/update' },
{ :controller => 'admin/overtime_rate', :action => 'update' })
      assert_routing({ :method => :post, :path => '/admin/overtime/rate/delete' },
{ :controller => 'admin/overtime_rate', :action => 'destroy' })

      assert_routing '/admin/overtime/chart', { :controller => 'admin/overtime_chart',
:action => 'index' }
      assert_routing '/admin/overtime/chart/data', { :controller => 'admin/overtime_chart',
:action => 'data' }

      assert_routing '/admin/workhours/chart', { :controller => 'admin/total_work_hours_chart',
:action => 'index' }
      assert_routing '/admin/workhours/chart/data', { :controller =>
'admin/total_work_hours_chart', :action => 'data' }

      assert_routing '/admin/salaryadj', { :controller => 'admin/salary_adjustment',
:action => 'index' }
      assert_routing '/admin/salaryadj/list', { :controller => 'admin/salary_adjustment',
:action => 'list' }
      assert_routing '/admin/salaryadj/new', { :controller => 'admin/salary_adjustment',
:action => 'new' }
      assert_routing({ :method => :post, :path => '/admin/salaryadj/create' },
{ :controller => 'admin/salary_adjustment', :action => 'create' })
      assert_routing '/admin/salaryadj/edit/1', { :controller => 'admin/salary_adjustment',
:action => 'edit', :id => '1' }
      assert_routing({ :method => :post, :path => '/admin/salaryadj/update' },
{ :controller => 'admin/salary_adjustment', :action => 'update' })
      assert_routing({ :method => :post, :path => '/admin/salaryadj/delete' },
{ :controller => 'admin/salary_adjustment', :action => 'destroy' })
end

test 'should route to user' do
  assert_routing '/user/index', { :controller => 'user/user', :action => 'index' }

  assert_routing '/user/info', { :controller => 'user/info', :action => 'index' }
  assert_routing({ :method => :post, :path => '/user/info/update' },
{ :controller => 'user/info', :action => 'update' })

  assert_routing '/user/contact', { :controller => 'user/contact', :action => 'index' }
  assert_routing({ :method => :post, :path => '/user/contact/update' },
{ :controller => 'user/contact', :action => 'update' })

  assert_routing '/user/job', { :controller => 'user/job', :action => 'index' }

  assert_routing '/user/salary', { :controller => 'user/salary', :action => 'index' }

  assert_routing '/user/qualification', { :controller => 'user/qualification',
:action => 'index' }
  assert_routing({ :method => :post, :path => '/user/qualification/update' },
{ :controller => 'user/qualification', :action => 'update' })

```

```

    assert_routing '/user/overtime/chart', { :controller => 'user/overtime_chart', :action =>
'index' }
    assert_routing '/user/overtime/chart/data', { :controller => 'user/overtime_chart',
:action => 'data' }

    assert_routing '/user/workhours/chart', { :controller => 'user/total_work_hours_chart',
:action => 'index' }
    assert_routing '/user/workhours/chart/data', { :controller =>
'user/total_work_hours_chart', :action => 'data' }

    assert_routing '/user/hourly/chart', { :controller => 'user/hourly_payroll_chart', :action =>
'index' }
    assert_routing '/user/hourly/chart/data', { :controller => 'user/hourly_payroll_chart',
:action => 'data' }

    assert_routing '/user/payslip', { :controller => 'user/payslip', :action => 'index' }
    assert_routing '/user/payslip/slip/1/2010', { :controller => 'user/payslip', :action =>
'payslip',
                                         :month => '1', :year => '2010' }

end
end

E:\workspace\payroll_rails\test\unit\salary_adjustment_test.rb

require 'test_helper'

class SalaryAdjustmentTest < ActiveSupport::TestCase

test 'should create salary_adjustment' do
  o = SalaryAdjustment.new
  o.id = 5
  o.staff_id = 'S0010'
  o.inc = 60
  o.month = 9
  o.year = 2009

  assert o.save
end

test 'should find salary_adjustment' do
  id = salary_adjustment(:one).id
  assert_nothing_raised { SalaryAdjustment.find(id) }
end

test 'should update salary_adjustment' do
  o = salary_adjustment(:two)
  assert o.update_attributes(:inc => 70)

  o.month = 8
  assert o.save
  o = SalaryAdjustment.find(o.id)
  assert_equal 8, o.month
end

test 'should destroy salary_adjustment' do
  o = salary_adjustment(:one)
  o.destroy
  assert_raise(ActiveRecord::RecordNotFound) { SalaryAdjustment.find(o.id) }
end

test 'should not create a salary_adjustment without required fields' do
  o = SalaryAdjustment.new
  assert !o.valid?
  assert o.errors[:staff_id].any?
  assert o.errors[:inc].any?
  assert o.errors[:month].any?
  assert o.errors[:year].any?

  assert_equal ['Staff ID is required'], o.errors[:staff_id]
  assert_equal ['Month is required', 'Month is invalid', 'Month is invalid'],
o.errors[:month]
  assert_equal ['Year is required', 'Year is invalid'], o.errors[:year]

  o.inc = -9
  o.month = 13
  o.year = 0

```

```

assert !o.valid?

assert_equal ['Increment is invalid'], o.errors[:inc]
assert_equal ['Month is invalid'], o.errors[:month]
assert_equal ['Year is invalid'], o.errors[:year]
end
end

E:\workspace\payroll_rails\test\unit\user_test.rb

require 'test_helper'

class UserTest < ActiveSupport::TestCase

  test 'should create user' do
    o = User.new
    o.role = 1
    o.username = 'beny'
    o.status = true
    o.pwd = 'benben'

    assert o.save
  end

  test 'should find user' do
    id = user(:ben).id
    assert_nothing_raised { User.find(id) }
  end

  test 'should update user' do
    o = user(:ken)
    assert o.update_attributes(:role => 2, :status => false)

    o.password = 'bbb'
    assert o.save
    o = User.find(o.id)
    assert_equal 'bbb', o.password
  end

  test 'should destroy user' do
    o = user(:ben)
    o.destroy
    assert_raise(ActiveRecord::RecordNotFound) { User.find(o.id) }
  end

  test 'should not create a user without required fields' do
    o = User.new
    assert !o.valid?
    assert o.errors[:username].any?
    assert o.errors[:pwd].any?

    assert_equal ['Minimum is 3 characters'], o.errors[:username]
    assert_equal ['Minimum is 4 characters', 'Password is required'], o.errors[:pwd]
  end

  test 'should not create a user with duplicate username' do
    o = User.new
    o.username = 'admin'
    assert !o.valid?
    assert o.errors[:username].any?

    assert_equal ['Username admin already exist'], o.errors[:username]
  end

  test 'should not create/update user with wrong password confirmation' do
    o = User.new(:username => 'Ken', :pwd => 'hhhh', :pwd_confirmation => 'aaaa')
    assert !o.valid?
    assert o.errors[:pwd].any?

    assert_equal ["Password doesn't match confirmation"], o.errors[:pwd]

    o = user(:ken)
    assert o.update_attributes(:pwd => 'jjjj', :pwd_confirmation => 'mmmm')
    assert o.errors[:pwd].any?

    assert_equal ["Password doesn't match confirmation"], o.errors[:pwd]
  end

```

```

end

E:\workspace\payroll_rails\test\functional\admin\admin_controller_test.rb

require 'test_helper'

class Admin::AdminControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
  end
end

E:\workspace\payroll_rails\test\functional\admin\attendance_controller_test.rb

require 'test_helper'

class Admin::AttendanceControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
  end

  test 'should get list' do
    login_as :admin
    get :list, { :work_date => '12-01-2013', :employee => 'ben' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end
end

E:\workspace\payroll_rails\test\functional\admin\department_controller_test.rb

require 'test_helper'

class Admin::DepartmentControllerTest < ActionController::TestCase
  setup do
    @dept = department(:one)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
  end

  test 'should get list' do
    login_as :admin
    get :list, { :keyword => 'r' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  test 'should get new' do
    login_as :admin
    get :new
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:dept)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)
  end

  test 'should create department' do
    login_as :admin
    assert_difference('Department.count') do
      post :create, { :name => 'Sales' }
    end
  end
end

```

```

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should get edit' do
    login_as :admin
    get :edit, { :id => @dept.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:dept)
    assert_not_nil assigns(:form_id)
    assert_equal 'edit-form', assigns(:form_id)
  end

  test 'should update department' do
    login_as :admin
    post :update, { :id => @dept.id, :name => 'Admin' }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should destroy department' do
    login_as :admin
    assert_difference('Department.count', -1) do
      post :destroy, { :id => [@dept.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 1 of 1', data['itemscount']
  end
end

E:\workspace\payroll_rails\test\functional\admin\designation_controller_test.rb

require 'test_helper'

class Admin::DesignationControllerTest < ActionController::TestCase
  setup do
    @des = designation(:one)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
  end

  test 'should get list' do
    login_as :admin
    get :list, { :keyword => 'rammer' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  test 'should get new' do
    login_as :admin
    get :new
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:designation)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)
  end

  test 'should create designation' do
    login_as :admin
    assert_difference('Designation.count') do
      post :create, { :title => 'Account', :desc => 'Accounting',
                    :note => 'account related tasks' }
    end
  end

```

```

    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should get edit' do
    login_as :admin
    get :edit, { :id => @des.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:designation)
    assert_not_nil assigns(:form_id)
    assert_equal 'edit-form', assigns(:form_id)
  end

  test 'should update designation' do
    login_as :admin
    post :update, { :id => @des.id, :title => 'Audit', :desc => 'Audit job',
                  :note => 'Audit task' }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should destroy designation' do
    login_as :admin
    assert_difference('Designation.count', -1) do
      post :destroy, { :id => [@des.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 1 of 1', data['itemscount']
  end
end

E:\workspace\payroll_rails\test\functional\admin\employee_controller_test.rb

require 'test_helper'

class Admin::EmployeeControllerTest < ActionController::TestCase
  setup do
    @employee = employee(:one)
    @employee_contact = employee_contact(:one)
    @employee_job = employee_job(:one)
    @employee_salary = employee_salary(:one)
    @employee_qualification = employee_qualification(:one)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
    assert_not_nil assigns(:employmentstatus)
    assert_not_nil assigns(:designation)
    assert_not_nil assigns(:dept)
  end

  test 'should get list' do
    login_as :admin
    get :list, { :staff_id => 'C0001' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  test 'should get new' do
    login_as :admin
    get :new
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:employee)
  end

```

```

    assert_not_nil assigns(:employee_contact)
    assert_not_nil assigns(:employee_job)
    assert_not_nil assigns(:employee_salary)
    assert_not_nil assigns(:employee_qualification)
    assert_not_nil assigns(:form_id)
    assert_not_nil assigns(:users)
    assert_not_nil assigns(:designations)
    assert_not_nil assigns(:employment_statuses)
    assert_not_nil assigns(:job_categories)
    assert_not_nil assigns(:departments)
    assert_equal 'add-form', assigns(:form_id)
  end

  test 'should create employee' do
    login as :admin
    assert_difference('Employee.count') do
      post :create, {
        :employee => {
          :staff_id => 'C0003', :first_name => 'wong', :middle_name => 'yan',
          :last_name => 'kin', :new_ic => '098455673', :old_ic => '88744532',
          :passport_no => @employee.passport_no, :gender => @employee.gender, :marital_status
=> @employee.marital_status,
          :nationality => @employee.nationality, :dob => '15-08-1977', :place_of_birth =>
@employee.place_of_birth,
          :race => @employee.race, :religion => @employee.religion, :is_bumi =>
@employee.is_bumi, :user_id => @employee.user_id
        },
        :employee_contact => {
          :id => @employee_contact.id, :address_1 => @employee_contact.address_1, :address_2
=> @employee_contact.address_2,
          :address_3 => @employee_contact.address_3, :city => @employee_contact.city, :state
=> @employee_contact.state,
          :postcode => @employee_contact.postcode, :country => @employee_contact.country,
          :home_phone => @employee_contact.home_phone,
          :mobile_phone => @employee_contact.mobile_phone, :work_email =>
@employee_contact.work_email,
          :other_email => @employee_contact.other_email
        },
        :employee_job => {
          :id => @employee_job.id, :designation_id => @employee_job.designation_id,
          :department_id => @employee_job.department_id,
          :employment_status_id => @employee_job.employment_status_id, :job_category_id =>
@employee_job.job_category_id,
          :join_date => '09-03-2011', :confirm_date => '09-06-2011'
        },
        :employee_salary => {
          :id => @employee_salary.id, :salary => @employee_salary.salary, :allowance =>
@employee_salary.allowance,
          :epf => @employee_salary.epf, :socso => @employee_salary.socso, :income_tax =>
@employee_salary.income_tax,
          :bank_name => @employee_salary.bank_name, :bank_acc_no =>
@employee_salary.bank_acc_no,
          :bank_acc_type => @employee_salary.bank_acc_type, :bank_address =>
@employee_salary.bank_address,
          :epf_no => @employee_salary.epf_no, :socso_no => @employee_salary.socso_no,
          :income_tax_no => @employee_salary.income_tax_no,
          :pay_type => @employee_salary.pay_type
        },
        :employee_qualification => {
          :id => @employee_qualification.id, :level => @employee_qualification.level,
          :institute => @employee_qualification.institute,
          :major => @employee_qualification.major, :year => 2010, :gpa =>
@employee_qualification.gpa,
          :start_date => '03-04-2006', :end_date => '02-04-2010'
        }
      }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

```

```

test 'should get edit' do
  login_as :admin
  get :edit, { :id => @employee.id }
  assert_response :success
  assert_template 'form'
  assert_not_nil assigns(:employee)
  assert_not_nil assigns(:employee_contact)
  assert_not_nil assigns(:employee_job)
  assert_not_nil assigns(:employee_salary)
  assert_not_nil assigns(:employee_qualification)
  assert_not_nil assigns(:form_id)
  assert_not_nil assigns(:users)
  assert_not_nil assigns(:designations)
  assert_not_nil assigns(:employment_statuses)
  assert_not_nil assigns(:job_categories)
  assert_not_nil assigns(:designations)
  assert_equal 'edit-form', assigns(:form_id)
end

test 'should update employee' do
  login_as :admin
  post :update,
  {
    :id => @employee.id,
    :employee => {
      :staff_id => 'C0005', :first_name => @employee.first_name, :middle_name =>
      @employee.middle_name,
      :last_name => @employee.last_name, :new_ic => @employee.new_ic, :old_ic =>
      @employee.old_ic,
      :passport_no => @employee.passport_no, :gender => @employee.gender, :marital_status =>
      @employee.marital_status,
      :nationality => @employee.nationality, :dob => '15-08-1977', :place_of_birth =>
      @employee.place_of_birth,
      :race => @employee.race, :religion => @employee.religion, :is_bumi =>
      @employee.is_bumi, :user_id => @employee.user_id
    },
    :employee_contact => {
      :id => @employee_contact.id, :address_1 => @employee_contact.address_1, :address_2 =>
      @employee_contact.address_2,
      :address_3 => @employee_contact.address_3, :city => @employee_contact.city, :state =>
      @employee_contact.state,
      :postcode => @employee_contact.postcode, :country => @employee_contact.country,
      :home_phone => @employee_contact.home_phone,
      :mobile_phone => @employee_contact.mobile_phone, :work_email =>
      @employee_contact.work_email,
      :other_email => @employee_contact.other_email
    },
    :employee_job => {
      :id => @employee_job.id, :designation_id => @employee_job.designation_id,
      :department_id => @employee_job.department_id,
      :employment_status_id => @employee_job.employment_status_id, :job_category_id =>
      @employee_job.job_category_id,
      :join_date => '09-03-2011', :confirm_date => '09-06-2011'
    },
    :employee_salary => {
      :id => @employee_salary.id, :salary => @employee_salary.salary, :allowance =>
      @employee_salary.allowance,
      :epf => @employee_salary.epf, :socso => @employee_salary.socso, :income_tax =>
      @employee_salary.income_tax,
      :bank_name => @employee_salary.bank_name, :bank_acc_no =>
      @employee_salary.bank_acc_no,
      :bank_acc_type => @employee_salary.bank_acc_type, :bank_address =>
      @employee_salary.bank_address,
      :epf_no => @employee_salary.epf_no, :socso_no => @employee_salary.socso_no,
      :income_tax_no => @employee_salary.income_tax_no,
      :pay_type => @employee_salary.pay_type
    },
    :employee_qualification => {
      :id => @employee_qualification.id, :level => @employee_qualification.level,
      :institute => @employee_qualification.institute,
      :major => @employee_qualification.major, :year => 2010, :gpa =>
      @employee_qualification.gpa,
      :start_date => '03-04-2006', :end_date => '02-04-2010'
    }
  }
end

```

```

        }
    }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
end

test 'should destroy employee' do
  login_as :admin
  assert_difference('Employee.count', -1) do
    post :destroy, { :id => [@employee.id] }
  end

  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
  assert_equal '1 to 1 of 1', data['itemscount']
end
end

E:\workspace\payroll_rails\test\functional\admin\employment_status_controller_test.rb

require 'test_helper'

class Admin::EmploymentStatusControllerTest < ActionController::TestCase
  setup do
    @employment_status = employment_status(:one)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
  end

  test 'should get list' do
    login_as :admin
    get :list, { :keyword => 'bat' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  test 'should get new' do
    login_as :admin
    get :new
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:empstatus)
  end

  test 'should create employment status' do
    login_as :admin
    assert_difference('EmploymentStatus.count') do
      post :create, { :name => 'Contract' }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should get edit' do
    login_as :admin
    get :edit, { :id => @employment_status.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:empstatus)
  end

  test 'should update employment status' do
    login_as :admin
    post :update, { :id => @employment_status.id, :name => 'Part time' }
    assert_response :success
    data = JSON.parse(@response.body)

```

```

    assert_equal 1, data['success']
  end

  test 'should destroy employment status' do
    login_as :admin
    assert_difference('EmploymentStatus.count', -1) do
      post :destroy, { :id => [@employment_status.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 1 of 1', data['itemscount']
  end
end

E:\workspace\payroll_rails\test\functional\admin\hourly_payroll_chart_controller_test.rb

require 'test_helper'

class Admin::HourlyPayrollChartControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:month_hash)
  end

  test 'should get data' do
    login_as :admin
    get :data
    assert_response :success
    assert_not_nil assigns(:data)
  end
end

E:\workspace\payroll_rails\test\functional\admin\job_category_controller_test.rb

require 'test_helper'

class Admin::JobCategoryControllerTest < ActionController::TestCase
  setup do
    @jobcat = job_category(:one)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
  end

  test 'should get list' do
    login_as :admin
    get :list, { :keyword => 'les' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  test 'should get new' do
    login_as :admin
    get :new
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:jobcat)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)
  end

  test 'should create job category' do
    login_as :admin
    assert_difference('JobCategory.count') do
      post :create, { :name => 'Management' }
    end
  end

```

```

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should get edit' do
    login_as :admin
    get :edit, { :id => @jobcat.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:jobcat)
    assert_not_nil assigns(:form_id)
    assert_equal 'edit-form', assigns(:form_id)
  end

  test 'should update job category' do
    login_as :admin
    post :update, { :id => @jobcat.id, :name => 'Accounting' }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should destroy job category' do
    login_as :admin
    assert_difference('JobCategory.count', -1) do
      post :destroy, { :id => [@jobcat.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 1 of 1', data['itemscount']
  end
end

E:\workspace\payroll_rails\test\functional\admin\overtime_chart_controller_test.rb

require 'test_helper'

class Admin::OvertimeChartControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:month_hash)
  end

  test 'should get data' do
    login_as :admin
    get :data
    assert_response :success
    assert_not_nil assigns(:data)
  end
end

E:\workspace\payroll_rails\test\functional\admin\overtime_rate_controller_test.rb

require 'test_helper'

class Admin::OvertimeRateControllerTest < ActionController::TestCase
  setup do
    @rate = overtime_rate(:one)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
  end

  test 'should get list' do
    login_as :admin

```

```

    get :list, { :year => 2008 }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  test 'should get new' do
    login_as :admin
    get :new
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:rate)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)
  end

  test 'should create overtime rate' do
    login_as :admin
    assert_difference('OvertimeRate.count') do
      post :create, { :duration => @rate.duration, :year => 2011, :pay_rate => @rate.pay_rate }
    end
  end

  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
end

test 'should get edit' do
  login_as :admin
  get :edit, { :id => @rate.id }
  assert_response :success
  assert_template 'form'
  assert_not_nil assigns(:rate)
  assert_not_nil assigns(:form_id)
  assert_equal 'edit-form', assigns(:form_id)
end

test 'should update overtime rate' do
  login_as :admin
  post :update, { :id => @rate.id, :duration => @rate.duration, :year => 2005,
                 :pay_rate => @rate.pay_rate }
  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
end

test 'should destroy overtime rate' do
  login_as :admin
  assert_difference('OvertimeRate.count', -1) do
    post :destroy, { :id => [@rate.id] }
  end

  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
  assert_equal '1 to 1 of 1', data['itemscount']
end
end

E:\workspace\payroll_rails\test\functional\admin\pay_rate_controller_test.rb

require 'test_helper'

class Admin::PayRateControllerTest < ActionController::TestCase
  setup do
    @rate = pay_rate(:one)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
  end

```

```

test 'should get list' do
  login_as :admin
  get :list, { :staff_id => 'C0001' }
  assert_response :success
  assert_template 'list'
  assert_not_nil assigns(:data)
end

test 'should get new' do
  login_as :admin
  get :new
  assert_response :success
  assert_template 'form'
  assert_not_nil assigns(:payrate)
  assert_not_nil assigns(:form_id)
  assert_equal 'add-form', assigns(:form_id)
end

test 'should create pay rate' do
  login_as :admin
  assert_difference('PayRate.count') do
    post :create, { :staff_id => 'C0007',
                   :month => @rate.month, :year => @rate.year,
                   :pay_rate => @rate.hourly_pay_rate }
  end

  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
end

test 'should get edit' do
  login_as :admin
  get :edit, { :id => @rate.id }
  assert_response :success
  assert_template 'form'
  assert_not_nil assigns(:payrate)
  assert_not_nil assigns(:form_id)
  assert_equal 'edit-form', assigns(:form_id)
end

test 'should update pay rate' do
  login_as :admin
  post :update, { :id => @rate.id, :staff_id => 'C0006', :month => 6,
                 :year => 2012, :pay_rate => 3.4 }
  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
end

test 'should destroy pay rate' do
  login_as :admin
  assert_difference('PayRate.count', -1) do
    post :destroy, { :id => [@rate.id] }
  end

  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
  assert_equal '1 to 1 of 1', data['itemscount']
end

```

E:\workspace\payroll_rails\test\functional\admin\payslip_controller_test.rb

```

require 'test_helper'

class Admin::PayslipControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
    assert_not_nil assigns(:employmentstatus)
    assert_not_nil assigns(:designation)
    assert_not_nil assigns(:dept)
  end

```

```

end

test 'should get list' do
  login_as :admin
  get :list, { :employee => 'ben' }
  assert_response :success
  assert_template 'list'
  assert_not_nil assigns(:data)
end

test 'should get monthly payslip' do
  login_as :admin
  get :payslip, { :id => 1 }
  assert_response :success
  assert_template 'payslip_monthly'
  assert_not_nil assigns(:employee)
  assert_not_nil assigns(:employee_salary)
  assert_not_nil assigns(:period)
  assert_not_nil assigns(:total_overtime)
  assert_not_nil assigns(:total_overtime_earnings)
  assert_not_nil assigns(:adjustment)
  assert_not_nil assigns(:total_earnings)
  assert_not_nil assigns(:total_deduct)
  assert_not_nil assigns(:nett_salary)
  assert_not_nil assigns(:basic_pay)
end

test 'should get hourly payslip' do
  login_as :admin
  get :payslip, { :id => 2 }
  assert_response :success
  assert_template 'payslip_hourly'
  assert_not_nil assigns(:employee)
  assert_not_nil assigns(:employee_salary)
  assert_not_nil assigns(:total_earnings)
  assert_not_nil assigns(:total_hours)
  assert_not_nil assigns(:hourly_pay_rate)
  assert_not_nil assigns(:total_deduct)
  assert_not_nil assigns(:nett_salary)
end
end

E:\workspace\payroll_rails\test\functional\admin\salary_adjustment_controller_test.rb

require 'test_helper'

class Admin::SalaryAdjustmentControllerTest < ActionController::TestCase
  setup do
    @adj = salary_adjustment(:one)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
  end

  test 'should get list' do
    login_as :admin
    get :list, { :staff_id => 'C0001' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  test 'should get new' do
    login_as :admin
    get :new
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:adj)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)
  end

```

```

test 'should create salary adjustment' do
  login_as :admin
  assert_difference('SalaryAdjustment.count') do
    post :create, { :staff_id => 'C0004', :inc => 300, :month => 8, :year => 2013 }
  end

  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
end

test 'should get edit' do
  login_as :admin
  get :edit, { :id => @adj.id }
  assert_response :success
  assert_template 'form'
  assert_not_nil assigns(:adj)
  assert_not_nil assigns(:form_id)
  assert_equal 'edit-form', assigns(:form_id)
end

test 'should update salary adjustment' do
  login_as :admin
  post :update, { :id => @adj.id, :staff_id => 'C0003', :inc => 250,
                 :month => 10, :year => 2013 }
  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
end

test 'should destroy salary adjustment' do
  login_as :admin
  assert_difference('SalaryAdjustment.count', -1) do
    post :destroy, { :id => [@adj.id] }
  end

  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
  assert_equal '1 to 1 of 1', data['itemscount']
end
end

E:\workspace\payroll_rails\test\functional\admin\total_work_hours_chart_controller_test.rb

require 'test_helper'

class Admin::TotalWorkHoursChartControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:month_hash)
  end

  test 'should get data' do
    login_as :admin
    get :data
    assert_response :success
    assert_not_nil assigns(:data)
  end
end

E:\workspace\payroll_rails\test\functional\admin\user_controller_test.rb

require 'test_helper'

class Admin::UserControllerTest < ActionController::TestCase
  setup do
    @user = user(:ben)
  end

  test 'should get index' do
    login_as :admin
    get :index
    assert_response :success
  end

```

```

    assert_template 'index'
    assert_not_nil assigns(:data)
  end

  test 'should get list' do
    login_as :admin
    get :list, { :username => 'en' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  test 'should get new' do
    login_as :admin
    get :new
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:user)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)
  end

  test 'should create user' do
    login_as :admin
    assert_difference('User.count') do
      post :create, { :role => 2, :username => 'mary', :status => '1',
                    :pwd => 'abc111', :pwdconfirm => 'abc111' }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should get edit' do
    login_as :admin
    get :edit, { :id => @user.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:user)
    assert_not_nil assigns(:form_id)
    assert_equal 'edit-form', assigns(:form_id)
  end

  test 'should update user' do
    login_as :admin
    post :update, { :id => @user.id, :role => 2, :username => 'mary',
                  :status => '1', :pwd => 'kim111', :pwdconfirm => 'kim111' }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  test 'should destroy user' do
    login_as :admin
    assert_difference('User.count', -1) do
      post :destroy, { :id => [@user.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 2 of 2', data['itemscount']
  end
end

E:\workspace\payroll_rails\test\functional\user\contact_controller_test.rb

require 'test_helper'

class User::ContactControllerTest < ActionController::TestCase
  setup do
    @employee_contact = employee_contact(:two)
  end

  test 'should get index' do

```

```

    login_as :ben
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:employee_contact)
  end

  test 'should update' do
    login_as :ben
    post :update, {
      :employee_contact => {
        :address_1 => @employee_contact.address_1, :address_2 => @employee_contact.address_2,
        :address_3 => @employee_contact.address_3, :city => @employee_contact.city, :state =>
        @employee_contact.state,
        :postcode => @employee_contact.postcode, :country => @employee_contact.country,
        :home_phone => @employee_contact.home_phone,
        :mobile_phone => @employee_contact.mobile_phone, :work_email =>
        @employee_contact.work_email,
        :other_email => @employee_contact.other_email
      },
    }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end
end

E:\workspace\payroll_rails\test\functional\user\hourly_payroll_chart_controller_test.rb

require 'test_helper'

class User::HourlyPayrollChartControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :ben
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:month_hash)
  end

  test 'should get data' do
    login_as :ben
    get :data
    assert_response :success
    assert_not_nil assigns(:data)
  end
end

E:\workspace\payroll_rails\test\functional\user\info_controller_test.rb

require 'test_helper'

class User::InfoControllerTest < ActionController::TestCase
  setup do
    @employee = employee(:one)
  end

  test 'should get index' do
    login_as :ben
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:employee)
    assert_not_nil assigns(:user)
  end

  test 'should update' do
    login_as :ben
    post :update, {
      :employee => {
        :first_name => 'wong', :middle_name => 'yan', :last_name => 'kin', :new_ic =>
        '098455673', :old_ic => '88744532',
        :passport_no => @employee.passport_no, :gender => @employee.gender, :marital_status =>
        @employee.marital_status,
        :nationality => @employee.nationality, :dob => '15-08-1977', :place_of_birth =>
        @employee.place_of_birth,
      }
    }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end
end

```

```

        :race => @employee.race, :religion => @employee.religion, :is_bumi =>
@employee.is_bumi
    }
}
assert_response :success
data = JSON.parse(@response.body)
assert_equal 1, data['success']
end
end

E:\workspace\payroll_rails\test\functional\user\job_controller_test.rb

require 'test_helper'

class User::JobControllerTest < ActionController::TestCase
test 'should get index' do
  login_as :ben
  get :index
  assert_response :success
  assert_template 'index'
  assert_not_nil assigns(:employee_job)
end
end

E:\workspace\payroll_rails\test\functional\user\overtime_chart_controller_test.rb

require 'test_helper'

class User::OvertimeChartControllerTest < ActionController::TestCase
test 'should get index' do
  login_as :ben
  get :index
  assert_response :success
  assert_template 'index'
  assert_not_nil assigns(:month_hash)
end

test 'should get data' do
  login_as :ben
  get :data
  assert_response :success
  assert_not_nil assigns(:data)
end
end

E:\workspace\payroll_rails\test\functional\user\payslip_controller_test.rb

require 'test_helper'

class User::PayslipControllerTest < ActionController::TestCase
test 'should get index' do
  login_as :ben
  get :index
  assert_response :success
  assert_template 'index'
end

test 'should get monthly payslip' do
  login_as :ben
  get :payslip, { :id => 2 }
  assert_response :success
  assert_template 'admin/payslip/payslip_monthly'
  assert_not_nil assigns(:employee)
  assert_not_nil assigns(:employee_salary)
  assert_not_nil assigns(:period)
  assert_not_nil assigns(:total_overtime)
  assert_not_nil assigns(:total_overtime_earnings)
  assert_not_nil assigns(:adjustment)
  assert_not_nil assigns(:total_earnings)
  assert_not_nil assigns(:total_deduct)
  assert_not_nil assigns(:nett_salary)
  assert_not_nil assigns(:basic_pay)
end

test 'should get hourly payslip' do
  login_as :ken
  get :payslip, { :id => 3 }

```

```

    assert_response :success
    assert_template 'admin/payslip/payslip_hourly'
    assert_not_nil assigns(:employee)
    assert_not_nil assigns(:employee_salary)
    assert_not_nil assigns(:total_earnings)
    assert_not_nil assigns(:total_hours)
    assert_not_nil assigns(:hourly_pay_rate)
    assert_not_nil assigns(:total_deduct)
    assert_not_nil assigns(:nett_salary)
  end
end

```

E:\workspace\payroll_rails\test\functional\user\qualification_controller_test.rb

```

require 'test_helper'

class User::QualificationControllerTest < ActionController::TestCase
  setup do
    @employee_qualification = employee_qualification(:one)
  end

  test 'should get index' do
    login_as :ben
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:employee_qualification)
  end

  test 'should update' do
    login_as :ben
    post :update, {
      :employee_qualification => {
        :level => @employee_qualification.level, :institute =>
@employee_qualification.institute,
        :major => @employee_qualification.major, :year => 2010, :gpa =>
@employee_qualification.gpa,
        :start_date => '03-04-2006', :end_date => '02-04-2010'
      },
    }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end
end

```

E:\workspace\payroll_rails\test\functional\user\salary_controller_test.rb

```

require 'test_helper'

class User::SalaryControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :ben
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:employee)
    assert_not_nil assigns(:employee_salary)
    assert_not_nil assigns(:adjustment)
    assert_not_nil assigns(:basic_pay)
  end
end

```

E:\workspace\payroll_rails\test\functional\user\total_work_hours_chart_controller_test.rb

```

require 'test_helper'

class User::TotalWorkHoursChartControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :ken
    get :index
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:month_hash)
  end

```

```
  test 'should get data' do
```

```

    login_as :ken
    get :data
    assert_response :success
    assert_not_nil assigns(:data)
  end
end

E:\workspace\payroll_rails\test\functional\user\user_controller_test.rb

require 'test_helper'

class UserControllerTest < ActionController::TestCase
  test 'should get index' do
    login_as :ben
    get :index
    assert_response :success
  end
end

E:\workspace\payroll_rails\test\integration\admin_attendance_flows_test.rb

require 'test_helper'

class AdminAttendanceFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
  end

  test 'browse attendance' do
    a = admin_user(AdminAttendanceFlows)
    a.logs_in @admin.username, 'secret'
    a.browse
    a.logs_out
  end

  private

  module AdminAttendanceFlows
    def browse
      get admin_att_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:data)

      get admin_att_list_path, { :work_date => '12-01-2013', :employee => 'ben' }
      assert_response :success
      assert_template 'list'
      assert_not_nil assigns(:data)
    end
  end
end

E:\workspace\payroll_rails\test\integration\admin_department_flows_test.rb

require 'test_helper'

class AdminDepartmentFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
    @dept = department(:one)
  end

  test 'login, search, create, update, and delete department' do
    a = admin_user(AdminDepartmentFlows)
    a.logs_in @admin.username, 'secret'
    a.search
    a.create :name => 'Sales'
    a.update @dept, { :id => @dept.id, :name => 'Admin' }
    a.destroy @dept
    a.logs_out
  end

  private

  module AdminDepartmentFlows
    def search
      get admin_dept_path

```

```

    assert_template 'index'
    assert_not_nil assigns(:data)

    get admin_dept_list_path, { :keyword => 'r' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  def create(obj_hash)
    get admin_dept_new_path
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:dept)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)

    assert_difference('Department.count') do
      post admin_dept_create_path, obj_hash
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def update(o, obj_hash)
    get admin_dept_edit_path, { :id => o.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:dept)
    assert_not_nil assigns(:form_id)
    assert_equal 'edit-form', assigns(:form_id)

    post admin_dept_update_path, obj_hash

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def destroy(o)
    assert_difference('Department.count', -1) do
      post admin_dept_delete_path, { :id => [o.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 2 of 2', data['itemscount']
  end
end

```

E:\workspace\payroll_rails\test\integration\admin_designation_flows_test.rb

```

require 'test_helper'

class AdminDesignationFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
    @des = designation(:one)
  end

  test 'login, search, create, update, and delete designation' do
    a = admin_user(AdminDesignationFlows)
    a.logs_in @admin.username, 'secret'
    a.search
    a.create :title => 'Account', :desc => 'Accounting',
             :note => 'account related tasks'
    a.update @des, { :id => @des.id, :title => 'Audit', :desc => 'Audit job',
                  :note => 'Audit task' }
    a.destroy @des
    a.logs_out
  end

  private

```

```

module AdminDesignationFlows
  def search
    get admin_designation_path
    assert_template 'index'
    assert_not_nil assigns(:data)

    get admin_designation_list_path, { :keyword => 'rammer' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  def create(obj_hash)
    get admin_designation_new_path
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:designation)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)

    assert_difference('Designation.count') do
      post admin_designation_create_path, obj_hash
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def update(o, obj_hash)
    get admin_designation_edit_path, { :id => o.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:designation)
    assert_not_nil assigns(:form_id)
    assert_equal 'edit-form', assigns(:form_id)

    post admin_designation_update_path, obj_hash

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def destroy(o)
    assert_difference('Designation.count', -1) do
      post admin_designation_delete_path, { :id => [o.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 2 of 2', data['itemscount']
  end
end

```

E:\workspace\payroll_rails\test\integration\admin_employee_flows_test.rb

```

require 'test_helper'

class AdminEmployeeFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:admin)
    @employee = employee(:one)
    @employee_contact = employee_contact(:one)
    @employee_job = employee_job(:one)
    @employee_salary = employee_salary(:one)
    @employee_qualification = employee_qualification(:one)
  end

  test '' do
    a = admin_user(AdminEmployeeFlows)
    a.logs_in @user.username, 'secret'
  end
end

```

```

:employee => {
  :staff_id => 'C0003', :first_name => 'wong', :middle_name => 'yan',
  :last_name => 'kin', :new_ic => '098455673', :old_ic => '88744532',
  :passport_no => @employee.passport_no, :gender => @employee.gender,
  :marital_status => @employee.marital_status, :nationality => @employee.nationality,
  :dob => '15-08-1977', :place_of_birth => @employee.place_of_birth,
  :race => @employee.race, :religion => @employee.religion,
  :is_bumi => @employee.is_bumi, :user_id => @employee.user_id
},
:employee_contact => {
  :id => @employee_contact.id, :address_1 => @employee_contact.address_1,
  :address_2 => @employee_contact.address_2,
  :address_3 => @employee_contact.address_3, :city => @employee_contact.city,
  :state => @employee_contact.state, :postcode => @employee_contact.postcode,
  :country => @employee_contact.country, :home_phone => @employee_contact.home_phone,
  :mobile_phone => @employee_contact.mobile_phone,
  :work_email => @employee_contact.work_email,
  :other_email => @employee_contact.other_email
},
:employee_job => {
  :id => @employee_job.id, :designation_id => @employee_job.designation_id,
  :department_id => @employee_job.department_id,
  :employment_status_id => @employee_job.employment_status_id,
  :job_category_id => @employee_job.job_category_id, :join_date => '09-03-2011',
  :confirm_date => '09-06-2011'
},
:employee_salary => {
  :id => @employee_salary.id, :salary => @employee_salary.salary,
  :allowance => @employee_salary.allowance, :epf => @employee_salary.epf,
  :socso => @employee_salary.socso, :income_tax => @employee_salary.income_tax,
  :bank_name => @employee_salary.bank_name,
  :bank_acc_no => @employee_salary.bank_acc_no,
  :bank_acc_type => @employee_salary.bank_acc_type,
  :bank_address => @employee_salary.bank_address,
  :epf_no => @employee_salary.epf_no, :socso_no => @employee_salary.socso_no,
  :income_tax_no => @employee_salary.income_tax_no,
  :pay_type => @employee_salary.pay_type
},
:employee_qualification => {
  :id => @employee_qualification.id, :level => @employee_qualification.level,
  :institute => @employee_qualification.institute,
  :major => @employee_qualification.major, :year => '2010',
  :gpa => @employee_qualification.gpa, :start_date => '03-04-2006',
  :end_date => '02-04-2010'
}
),
a.create d

a.update @employee,
{
  :id => @employee.id,
  :employee => {
    :staff_id => 'C0005', :first_name => @employee.first_name, :middle_name =>
@employee.middle_name,
    :last_name => @employee.last_name, :new_ic => @employee.new_ic, :old_ic =>
@employee.old_ic,
    :passport_no => @employee.passport_no, :gender => @employee.gender, :marital_status
=> @employee.marital_status,
    :nationality => @employee.nationality, :dob => '15-08-1977', :place_of_birth =>
@employee.place_of_birth,
    :race => @employee.race, :religion => @employee.religion, :is_bumi =>
@employee.is_bumi, :user_id => @employee.user_id
  },
  :employee_contact => {
    :id => @employee_contact.id, :address_1 => @employee_contact.address_1, :address_2
=> @employee_contact.address_2,
    :address_3 => @employee_contact.address_3, :city => @employee_contact.city, :state
=> @employee_contact.state,
    :postcode => @employee_contact.postcode, :country => @employee_contact.country,
    :home_phone => @employee_contact.home_phone,
    :mobile_phone => @employee_contact.mobile_phone, :work_email =>
@employee_contact.work_email,
  }
}

```

```

        :other_email => @employee_contact.other_email
    },
    :employee_job => {
        :id => @employee_job.id, :designation_id => @employee_job.designation_id,
        :department_id => @employee_job.department_id,
        :employment_status_id => @employee_job.employment_status_id, :job_category_id =>
        @employee_job.job_category_id,
        :join_date => '09-03-2011', :confirm_date => '09-06-2011'
    },
    :employee_salary => {
        :id => @employee_salary.id, :salary => @employee_salary.salary, :allowance =>
        @employee_salary.allowance,
        :epf => @employee_salary.epf, :socso => @employee_salary.socso, :income_tax =>
        @employee_salary.income_tax,
        :bank_name => @employee_salary.bank_name, :bank_acc_no =>
        @employee_salary.bank_acc_no,
        :bank_acc_type => @employee_salary.bank_acc_type, :bank_address =>
        @employee_salary.bank_address,
        :epf_no => @employee_salary.epf_no, :socso_no => @employee_salary.socso_no,
        :income_tax_no => @employee_salary.income_tax_no,
        :pay_type => @employee_salary.pay_type
    },
    :employee_qualification => {
        :id => @employee_qualification.id, :level => @employee_qualification.level,
        :institute => @employee_qualification.institute,
        :major => @employee_qualification.major, :year => 2010, :gpa =>
        @employee_qualification.gpa,
        :start_date => '03-04-2006', :end_date => '02-04-2010'
    }
}
}

a.destroy @employee
a.logs_out
end

private

module AdminEmployeeFlows
  def search
    get admin_employee_path
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:data)
    assert_not_nil assigns(:employmentstatus)
    assert_not_nil assigns(:designation)
    assert_not_nil assigns(:dept)

    get admin_employee_list_path, { :staff_id => 'C0001' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)
  end

  def create(obj_hash)
    get admin_employee_new_path
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:employee)
    assert_not_nil assigns(:employee_contact)
    assert_not_nil assigns(:employee_job)
    assert_not_nil assigns(:employee_salary)
    assert_not_nil assigns(:employee_qualification)
    assert_not_nil assigns(:form_id)
    assert_not_nil assigns(:users)
    assert_not_nil assigns(:designations)
    assert_not_nil assigns(:employment_statuses)
    assert_not_nil assigns(:job_categories)
    assert_not_nil assigns(:departments)
    assert_equal 'add-form', assigns(:form_id)

    assert_difference('Employee.count') do
      post admin_employee_create_path, obj_hash
    end
  end
end

```

```

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def update(o, obj_hash)
    get admin_employee_edit_path, { :id => o.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:employee)
    assert_not_nil assigns(:employee_contact)
    assert_not_nil assigns(:employee_job)
    assert_not_nil assigns(:employee_salary)
    assert_not_nil assigns(:employee_qualification)
    assert_not_nil assigns(:form_id)
    assert_not_nil assigns(:users)
    assert_not_nil assigns(:designations)
    assert_not_nil assigns(:employment_statuses)
    assert_not_nil assigns(:job_categories)
    assert_not_nil assigns(:designations)
    assert_equal 'edit-form', assigns(:form_id)

    post admin_employee_update_path, obj_hash
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def destroy(o)
    assert_difference('Employee.count', -1) do
      post admin_employee_delete_path, { :id => [o.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 2 of 2', data['itemscount']
  end
end
end

```

E:\workspace\payroll_rails\test\integration\admin_employment_status_flows_test.rb

```

require 'test_helper'

class AdminEmploymentStatusFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
    @employment_status = employment_status(:one)
  end

  test 'login, search, create, update, and delete employment status' do
    a = admin_user(AdminEmploymentStatusFlows)
    a.logs_in @admin.username, 'secret'
    a.search
    a.create :name => 'Contract'
    a.update @employment_status, { :id => @employment_status.id, :name => 'Part time' }
    a.destroy @employment_status
    a.logs_out
  end

  private

  module AdminEmploymentStatusFlows
    def search
      get admin_empstatus_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:data)
    end

    def create(obj_hash)

```

```

get admin_empstatus_new_path
assert_response :success
assert_template 'form'
assert_not_nil assigns(:empstatus)

assert_difference('EmploymentStatus.count') do
  post admin_empstatus_create_path, obj_hash
end

assert_response :success
data = JSON.parse(@response.body)
assert_equal 1, data['success']
end

def update(o, obj_hash)
  get admin_empstatus_edit_path, { :id => o.id }
  assert_response :success
  assert_template 'form'
  assert_not_nil assigns(:empstatus)

  post admin_empstatus_update_path, obj_hash
  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
end

def destroy(o)
  assert_difference('EmploymentStatus.count', -1) do
    post admin_empstatus_delete_path, { :id => [o.id] }
  end

  assert_response :success
  data = JSON.parse(@response.body)
  assert_equal 1, data['success']
  assert_equal '1 to 2 of 2', data['itemscount']
end
end
end

E:\workspace\payroll_rails\test\integration\admin_hourly_payroll_chart_flows_test.rb

require 'test_helper'

class AdminHourlyPayrollChartFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:admin)
  end

  test 'browse chart' do
    a = admin_user(AdminHourlyPayrollChartFlows)
    a.logs_in @user.username, 'secret'
    a.browse
    a.logs_out
  end

  private

  module AdminHourlyPayrollChartFlows
    def browse
      get admin_hourly_chart_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:month_hash)

      get admin_hourly_chart_data_path
      assert_response :success
      assert_not_nil assigns(:data)
    end
  end
end

```

```

E:\workspace\payroll_rails\test\integration\admin_job_category_flows_test.rb

require 'test_helper'

class AdminJobCategoryFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
    @jobcat = job_category(:one)
  end

  test 'login, search, create, update, and delete job category' do
    a = admin_user(AdminJobCategoryFlows)
    a.logs_in @admin.username, 'secret'
    a.search
    a.create :name => 'Management'
    a.update @jobcat, { :id => @jobcat.id, :name => 'Accounting' }
    a.destroy @jobcat
    a.logs_out
  end

  private

  module AdminJobCategoryFlows
    def search
      get admin_jobcat_path
      assert_template 'index'
      assert_not_nil assigns(:data)

      get admin_jobcat_list_path, { :keyword => 'les' }
      assert_response :success
      assert_template 'list'
      assert_not_nil assigns(:data)
    end

    def create(obj_hash)
      get admin_jobcat_new_path
      assert_response :success
      assert_template 'form'
      assert_not_nil assigns(:jobcat)
      assert_not_nil assigns(:form_id)
      assert_equal 'add-form', assigns(:form_id)

      assert_difference('JobCategory.count') do
        post admin_jobcat_create_path, obj_hash
      end

      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
    end

    def update(o, obj_hash)
      get admin_jobcat_edit_path, { :id => o.id }
      assert_response :success
      assert_template 'form'
      assert_not_nil assigns(:jobcat)
      assert_not_nil assigns(:form_id)
      assert_equal 'edit-form', assigns(:form_id)

      post admin_jobcat_update_path, obj_hash

      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
    end

    def destroy(o)
      assert_difference('JobCategory.count', -1) do
        post admin_jobcat_delete_path, { :id => [o.id] }
      end

      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
      assert_equal '1 to 2 of 2', data['itemscount']
    end
  end
end

```

```

end

E:\workspace\payroll_rails\test\integration\admin_overtime_chart_flows_test.rb

require 'test_helper'

class AdminOvertimeChartFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:admin)
  end

  test 'browse chart' do
    a = admin_user(AdminOvertimeChartFlows)
    a.logs_in @user.username, 'secret'
    a.browse
    a.logs_out
  end

  private

  module AdminOvertimeChartFlows
    def browse
      get admin_overtime_chart_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:month_hash)

      get admin_overtime_chart_data_path
      assert_response :success
      assert_not_nil assigns(:data)
    end
  end
end

```



```

E:\workspace\payroll_rails\test\integration\admin_overtime_rate_flows_test.rb

require 'test_helper'

class AdminOvertimeRateFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
    @rate = overtime_rate(:one)
  end

  test 'login, search, create, update, and delete overtime rate' do
    a = admin_user(AdminOvertimeRateFlows)
    a.logs_in @admin.username, 'secret'
    a.search
    a.create :duration => @rate.duration, :year => 2011, :pay_rate => @rate.pay_rate
    a.update @rate, { :id => @rate.id, :duration => @rate.duration, :year => 2005,
                     :pay_rate => @rate.pay_rate }
    a.destroy @rate
    a.logs_out
  end

  private

  module AdminOvertimeRateFlows
    def search
      get admin_overtime_rate_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:data)

      get admin_overtime_rate_list_path, { :year => 2008 }
      assert_response :success
      assert_template 'list'
      assert_not_nil assigns(:data)
    end

    def create(obj_hash)
      get admin_overtime_rate_new_path
      assert_response :success
      assert_template 'form'
      assert_not_nil assigns(:rate)
      assert_not_nil assigns(:form_id)
      assert_equal 'add-form', assigns(:form_id)
    end
  end
end

```

```

    assert_difference('OvertimeRate.count') do
      post admin_overtime_rate_create_path, obj_hash
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def update(o, obj_hash)
    get admin_overtime_rate_edit_path, { :id => o.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:rate)
    assert_not_nil assigns(:form_id)
    assert_equal 'edit-form', assigns(:form_id)

    post admin_overtime_rate_update_path, obj_hash
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def destroy(o)
    assert_difference('OvertimeRate.count', -1) do
      post admin_overtime_rate_delete_path, { :id => [o.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 2 of 2', data['itemscount']
  end
end

```

E:\workspace\payroll_rails\test\integration\admin_pay_rate_flows_test.rb

```

require 'test_helper'

class AdminPayRateFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
    @rate = pay_rate(:one)
  end

  test 'login, search, create, update, and delete pay rate' do
    a = admin_user(AdminPayRateFlows)
    a.logs_in @admin.username, 'secret'
    a.search
    a.create :staff_id => 'C0007',
             :month => @rate.month, :year => @rate.year,
             :pay_rate => @rate.hourly_pay_rate
    a.update @rate, { :id => @rate.id, :staff_id => 'C0006', :month => 6,
                    :year => 2012, :pay_rate => 3.4 }
    a.destroy @rate
    a.logs_out
  end

  private

  module AdminPayRateFlows
    def search
      get admin_payrate_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:data)
    end

    def create(obj_hash)
      get admin_payrate_new_path

```

```

    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:payrate)
    assert_not_nil assigns(:form_id)
    assert_equal 'add-form', assigns(:form_id)

    assert_difference('PayRate.count') do
      post admin_payrate_create_path, obj_hash
    end

    assert response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def update(o, obj_hash)
    get admin_payrate_edit_path, { :id => o.id }
    assert_response :success
    assert_template 'form'
    assert_not_nil assigns(:payrate)
    assert_not_nil assigns(:form_id)
    assert_equal 'edit-form', assigns(:form_id)

    post admin_payrate_update_path, obj_hash
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end

  def destroy(o)
    assert_difference('PayRate.count', -1) do
      post admin_payrate_delete_path, { :id => [o.id] }
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
    assert_equal '1 to 2 of 2', data['itemscount']
  end
end

```

```

E:\workspace\payroll_rails\test\integration\admin_payslip_flows_test.rb

require 'test_helper'

class AdminPayslipFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:admin)
  end

  test 'browse and view payslip' do
    a = admin_user(AdminPayslipFlows)
    a.logs_in @user.username, 'secret'
    a.browse
    a.logs_out
  end

  private

  module AdminPayslipFlows
    def browse
      get admin_payslip_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:data)
      assert_not_nil assigns(:employmentstatus)
      assert_not_nil assigns(:designation)
      assert_not_nil assigns(:dept)
    end

    get admin_payslip_list_path, { :employee => 'ben' }
    assert_response :success
    assert_template 'list'
    assert_not_nil assigns(:data)

    get admin_payslip_slip_path, { :id => 1 }
    assert_response :success
  end
end

```

```

    assert_template 'payslip_monthly'
    assert_not_nil assigns(:employee)
    assert_not_nil assigns(:employee_salary)
    assert_not_nil assigns(:period)
    assert_not_nil assigns(:total_overtime)
    assert_not_nil assigns(:total_overtime_earnings)
    assert_not_nil assigns(:adjustment)
    assert_not_nil assigns(:total_earnings)
    assert_not_nil assigns(:total_deduct)
    assert_not_nil assigns(:nett_salary)
    assert_not_nil assigns(:basic_pay)

    get admin_payslip_slip_path, { :id => 2 }
    assert_response :success
    assert_template 'payslip_hourly'
    assert_not_nil assigns(:employee)
    assert_not_nil assigns(:employee_salary)
    assert_not_nil assigns(:total_earnings)
    assert_not_nil assigns(:total_hours)
    assert_not_nil assigns(:hourly_pay_rate)
    assert_not_nil assigns(:total_deduct)
    assert_not_nil assigns(:nett_salary)
  end
end
end

E:\workspace\payroll_rails\test\integration\admin_salary_adjustment_flows_test.rb

require 'test_helper'

class AdminSalaryAdjustmentFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
    @adj = salary_adjustment(:one)
  end

  test 'login, search, create, update, and delete salary adjustment' do
    a = admin_user(AdminSalaryAdjustmentFlows)
    a.logs_in @admin.username, 'secret'
    a.search
    a.create :staff_id => 'C0003', :inc => 300, :month => 8, :year => 2013
    a.update @adj, { :id => @adj.id, :staff_id => 'C0003', :inc => 250,
      :month => 10, :year => 2013 }
    a.destroy @adj
    a.logs_out
  end

  private

  module AdminSalaryAdjustmentFlows
    def search
      get admin_salaryadj_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:data)

      get admin_salaryadj_list_path, { :staff_id => 'C0001' }
      assert_response :success
      assert_template 'list'
      assert_not_nil assigns(:data)
    end

    def create(obj_hash)
      get admin_salaryadj_new_path
      assert_response :success
      assert_template 'form'
      assert_not_nil assigns(:adj)
      assert_not_nil assigns(:form_id)
      assert_equal 'add-form', assigns(:form_id)

      assert_difference('SalaryAdjustment.count') do
        post admin_salaryadj_create_path, obj_hash
      end
    end

    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end
end

```

```

    end

    def update(o, obj_hash)
      get admin_salaryadj_edit_path, { :id => o.id }
      assert_response :success
      assert_template 'form'
      assert_not_nil assigns(:adj)
      assert_not_nil assigns(:form_id)
      assert_equal 'edit-form', assigns(:form_id)

      post admin_salaryadj_update_path, obj_hash
      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
    end

    def destroy(o)
      assert_difference('SalaryAdjustment.count', -1) do
        post admin_salaryadj_delete_path, { :id => [o.id] }
      end

      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
      assert_equal '1 to 2 of 2', data['itemscount']
    end
  end
end

```

E:\workspace\payroll_rails\test\integration\admin_total_work_hours_chart_flows_test.rb

```

require 'test_helper'

class AdminTotalWorkHoursChartFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:admin)
  end

  test 'browse chart' do
    a = admin_user(AdminTotalWorkHoursChartFlows)
    a.logs_in @user.username, 'secret'
    a.browse
    a.logs_out
  end

  private

  module AdminTotalWorkHoursChartFlows
    def browse
      get admin_workhours_chart_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:month_hash)

      get admin_workhours_chart_data_path
      assert_response :success
      assert_not_nil assigns(:data)
    end
  end
end

```

E:\workspace\payroll_rails\test\integration\admin_user_flows_test.rb

```

require 'test_helper'

class AdminUserFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @admin = user(:admin)
    @user = user(:ken)
  end

  test 'login, search, create, update, and delete user' do
    a = admin_user(AdminUserFlows)
    a.logs_in @admin.username, 'secret'
    a.search
    a.create :role => 2, :username => 'jane', :status => '1',
             :pwd => 'abc111', :pwdconfirm => 'abc111'
  end
end

```

```

    a.update @user, { :id => @user.id, :role => 2, :username => 'pauline', :status => '1',
                    :pwd => 'kim111', :pwdconfirm => 'kim111' }
    a.destroy @user
    a.logs_out
  end

  private

  module AdminUserFlows
    def search
      get admin_user_path
      assert_template 'index'
      assert_not_nil assigns(:data)

      get admin_user_list_path, { :username => 'en' }
      assert_response :success
      assert_template 'list'
      assert_not_nil assigns(:data)
    end

    def create(obj_hash)
      get admin_user_new_path
      assert_response :success
      assert_template 'form'
      assert_not_nil assigns(:user)
      assert_not_nil assigns(:form_id)
      assert_equal 'add-form', assigns(:form_id)

      assert_difference('User.count') do
        post admin_user_create_path, obj_hash
      end

      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
    end

    def update(o, obj_hash)
      get admin_user_edit_path, { :id => o.id }
      assert_response :success
      assert_template 'form'
      assert_not_nil assigns(:user)
      assert_not_nil assigns(:form_id)
      assert_equal 'edit-form', assigns(:form_id)

      post admin_user_update_path, obj_hash

      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
    end

    def destroy(o)
      assert_difference('User.count', -1) do
        post admin_user_delete_path, { :id => [o.id] }
      end

      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
      assert_equal '1 to 3 of 3', data['itemscount']
    end
  end
end

```

```

E:\workspace\payroll_rails\test\integration\user_contact_flows_test.rb

require 'test_helper'

class UserContactFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:ben)
    @employee_contact = employee_contact(:two)
  end

  test 'view and update contact' do
    a = normal_user(UserContactFlows)
  end

```

```

    a.logs_in @user.username, 'secret'
    a.update @employee_contact
    a.logs_out
  end

  private

  module UserContactFlows
    def update(o)
      get user_contact_path
      assert response :success
      assert_template 'index'
      assert_not_nil assigns(:employee_contact)

      post user_contact_update_path, {
        :employee_contact => {
          :address_1 => o.address_1, :address_2 => o.address_2,
          :address_3 => o.address_3, :city => o.city, :state => o.state,
          :postcode => o.postcode, :country => o.country, :home_phone => o.home_phone,
          :mobile_phone => o.mobile_phone, :work_email => o.work_email,
          :other_email => o.other_email
        },
      }
      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
    end
  end
end

E:\workspace\payroll_rails\test\integration\user_hourly_payroll_chart_flows_test.rb

require 'test_helper'

class UserHourlyPayrollChartFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:ken)
  end

  test 'browse chart' do
    a = normal_user(UserHourlyPayrollChartFlows)
    a.logs_in @user.username, 'secret'
    a.browse
    a.logs_out
  end

  private

  module UserHourlyPayrollChartFlows
    def browse
      get user_hourly_chart_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:month_hash)

      get user_hourly_chart_data_path
      assert_response :success
      assert_not_nil assigns(:data)
    end
  end
end

E:\workspace\payroll_rails\test\integration\user_info_flows_test.rb

require 'test_helper'

class UserInfoFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:ben)
    @employee = employee(:one)
  end

  test 'view and update info' do
    a = normal_user(UserInfoFlows)
    a.logs_in @user.username, 'secret'
    a.update @employee
    a.logs_out
  end

```

```

end

private

module UserInfoFlows
  def update(o)
    get user_info_path
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:employee)
    assert_not_nil assigns(:user)

    post user_info_update_path, {
      :employee => {
        :first_name => 'wong', :middle_name => 'yan', :last_name => 'kin', :new_ic =>
        '098455673', :old_ic => '88744532',
        :passport_no => o.passport_no, :gender => o.gender, :marital_status =>
        o.marital_status,
        :nationality => o.nationality, :dob => '15-08-1977', :place_of_birth =>
        o.place_of_birth,
        :race => o.race, :religion => o.religion, :is_bumi => o.is_bumi
      }
    }
    assert_response :success
    data = JSON.parse(@response.body)
    assert_equal 1, data['success']
  end
end
end

E:\workspace\payroll_rails\test\integration\user_job_flows_test.rb

require 'test_helper'

class UserJobFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:ben)
  end

  test 'view job' do
    a = normal_user(UserJobFlows)
    a.logs_in @user.username, 'secret'
    a.view
    a.logs_out
  end

private

module UserJobFlows
  def view
    get user_job_path
    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:employee_job)
  end
end
end

E:\workspace\payroll_rails\test\integration\user_overtime_chart_flows_test.rb

require 'test helper'

class UserOvertimeChartFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:ben)
  end

  test 'browse chart' do
    a = normal_user(UserOvertimeChartFlows)
    a.logs_in @user.username, 'secret'
    a.browse
    a.logs_out
  end

module UserOvertimeChartFlows
  def browse
    get user_overtime_chart_path

```

```

    assert_response :success
    assert_template 'index'
    assert_not_nil assigns(:month_hash)

    get user_overtime_chart_data_path
    assert_response :success
    assert_not_nil assigns(:data)
  end
end
end

E:\workspace\payroll_rails\test\integration\user_payslip_flows_test.rb

require 'test_helper'

class UserPayslipFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @ben = user(:ben)
    @ken = user(:ken)
  end

  test 'view payslip' do
    a = normal_user(UserPayslipFlows)
    b = normal_user(UserPayslipFlows)
    a.logs_in @ben.username, 'secret'
    b.logs_in @ken.username, 'secret'
    a.view_monthly
    b.view_hourly
    a.logs_out
    b.logs_out
  end

  module UserPayslipFlows
    def view_monthly
      get user_payslip_path
      assert_response :success
      assert_template 'index'

      get user_payslip_slip_path, { :id => 2 }
      assert_response :success
      assert_template 'admin/payslip/payslip_monthly'
      assert_not_nil assigns(:employee)
      assert_not_nil assigns(:employee_salary)
      assert_not_nil assigns(:period)
      assert_not_nil assigns(:total_overtime)
      assert_not_nil assigns(:total_overtime_earnings)
      assert_not_nil assigns(:adjustment)
      assert_not_nil assigns(:total_earnings)
      assert_not_nil assigns(:total_deduct)
      assert_not_nil assigns(:nett_salary)
      assert_not_nil assigns(:basic_pay)
    end

    def view_hourly
      get user_payslip_path
      assert_response :success
      assert_template 'index'

      get user_payslip_slip_path, { :id => 3 }
      assert_response :success
      assert_template 'admin/payslip/payslip_hourly'
      assert_not_nil assigns(:employee)
      assert_not_nil assigns(:employee_salary)
      assert_not_nil assigns(:total_earnings)
      assert_not_nil assigns(:total_hours)
      assert_not_nil assigns(:hourly_pay_rate)
      assert_not_nil assigns(:total_deduct)
      assert_not_nil assigns(:nett_salary)
    end
  end
end

```

```

E:\workspace\payroll_rails\test\integration\user_qualification_flows_test.rb

require 'test_helper'

class UserQualificationFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:ben)
    @employee_qualification = employee_qualification(:one)
  end

  test 'view and update qualification' do
    a = normal_user(UserQualificationFlows)
    a.logs_in @user.username, 'secret'
    a.update @employee_qualification
    a.logs_out
  end

  private

  module UserQualificationFlows
    def update(o)
      get user_qualification_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:employee_qualification)

      post user_qualification_update_path, {
        :employee_qualification => {
          :level => o.level, :institute => o.institute,
          :major => o.major, :year => 2010, :gpa => o.gpa,
          :start_date => '03-04-2006', :end_date => '02-04-2010'
        },
      }
      assert_response :success
      data = JSON.parse(@response.body)
      assert_equal 1, data['success']
    end
  end
end

```

```

E:\workspace\payroll_rails\test\integration\user_salary_flows_test.rb

require 'test_helper'

class UserSalaryFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:ben)
  end

  test 'view salary' do
    a = normal_user(UserSalaryFlows)
    a.logs_in @user.username, 'secret'
    a.view
    a.logs_out
  end

  private

  module UserSalaryFlows
    def view
      get user_salary_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:employee)
      assert_not_nil assigns(:employee_salary)
      assert_not_nil assigns(:adjustment)
      assert_not_nil assigns(:basic_pay)
    end
  end
end

```

```

E:\workspace\payroll_rails\test\integration\user_total_work_hours_chart_flows_test.rb

require 'test_helper'

class UserTotalWorkHoursChartFlowsTest < ActionDispatch::IntegrationTest
  setup do
    @user = user(:ben)
  end

  test 'browse chart' do
    a = normal_user(UserTotalWorkHoursChartFlows)
    a.logs_in @user.username, 'secret'
    a.browse
    a.logs_out
  end

  private

  module UserTotalWorkHoursChartFlows
    def browse
      get user_workhours_chart_path
      assert_response :success
      assert_template 'index'
      assert_not_nil assigns(:month_hash)

      get user_workhours_chart_data_path
      assert_response :success
      assert_not_nil assigns(:data)
    end
  end
end

```