# 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-wfsiewapp.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 there 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 a 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.

1. **rails new**

It creates a new Rails application.

1. **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.



Figure x: rails server command.

1. **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.

1. **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.

1. **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).

### 5.3.3: Technologies

Todo

### 5.3.4: Integrated Development Environment

Todo