|  |
| --- |
|  |
| Capstone Project Document |

**OHRM**

Report #2 – Architecture Design

|  |  |  |
| --- | --- | --- |
| **OHRM** | | |
| **Group Members** | Nguyễn Hữu Phước | SE61138 |
| Nguyễn Thị Xuân Mai | SE60910 |
| Ngô Trắc Kiện | SE61016 |
| Trần Quốc Hiệp | SE60928 |
| Đỗ Thanh Dương | SE61191 |
| **Supervisor** | Mr. Nguyễn Văn Sang | |
| **Project code** | OHRM | |

**- Hanoi, 09/2015 -**

# SIGNATURE PAGE

AUTHOR: Đỗ Thanh Dương 19/02/2016

Team member

REVIEWERS: Nguyễn Hữu Phước

Team leader

APPROVAL: Nguyễn Văn Sang

Supervisor

Record of change

\*A - Added M - Modified D – Deleted

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Effective Date | Changed Item | A,M,D | Change Description | Reason for Change | Rev. Number |
| 19/Sep/2015 | Create Architecture Design | A | First version | Create Architecture Design | 1.0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

TABLE CONTENTS

[SIGNATURE PAGE 1](#_Toc443945961)

[1 PROJECT OVERVIEW 3](#_Toc443945962)

[1.1 Purpose 3](#_Toc443945963)

[1.2 Scope 3](#_Toc443945964)

[1.3 Definitions, Acronyms, Abbreviations 3](#_Toc443945965)

[1.4 References 3](#_Toc443945966)

[1.5 Overview 3](#_Toc443945967)

[2 CHOICE OF ARCHITECTURE DESIGN 4](#_Toc443945968)

[2.1 MVC Model 4](#_Toc443945969)

[2.1.1 MVC Model Overview 4](#_Toc443945970)

[2.1.2 Advantages and disadvantages of MVC Model 5](#_Toc443945971)

[2.1.3 The reasons for choosing MVC Model 5](#_Toc443945972)

[2.2 Struts Framework 6](#_Toc443945973)

[2.2.1 Struts Framework Overview 6](#_Toc443945974)

[2.2.2 Advantages and disadvantages of Struts Framework 7](#_Toc443945975)

[2.2.3 The reasons for choosing .Net Framework 7](#_Toc443945976)

[2.3 EJB – Enterprise Java Bean 7](#_Toc443945977)

[2.3.1 EJB Overview 7](#_Toc443945978)

[2.3.2 Advantages and disadvantages of EJB 8](#_Toc443945979)

[2.3.3 The reason for choosing EJB 8](#_Toc443945980)

[3 ARCHITECTURAL REPRESENTATION 9](#_Toc443945981)

[4 ARCHITECTURAL GOALS AND CONSTRAINTS 10](#_Toc443945982)

[5 Use-case View 11](#_Toc443945983)

[5.1 Guest Group Function 11](#_Toc443945984)

[5.1.1 Login Module 11](#_Toc443945985)

[5.2 Authenticated User Group Function 12](#_Toc443945986)

[5.2.1 HR/ Resource Manager/Admin Group Function 12](#_Toc443945987)

[5.2.2 Resource Manager Group Function 14](#_Toc443945988)

[5.2.3 Admin Group Function 17](#_Toc443945989)

[6 Logical View 18](#_Toc443945990)

[6.1 Overview 18](#_Toc443945991)

[6.2 Architecturally Significant Design Packages 19](#_Toc443945992)

[7 Process view 22](#_Toc443945993)

[8 Deployment View 24](#_Toc443945994)

[9 Quality 25](#_Toc443945995)

# PROJECT OVERVIEW

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made on the system.

## Scope

The scope of this document is to depict the architecture of the OHRM website created by OHRM capstone project team.

## Definitions, Acronyms, Abbreviations

|  |  |  |
| --- | --- | --- |
| Acronym | Definition | Note |
| OHRM | Outsourcing Human Resource Management |  |
| DB | Database |  |
| MVC | Model view control |  |
| IDE | Integrated development environment |  |
| Q&A | Question and answer |  |
| GUI | Graphic user interface |  |
| EJB | Enterprise Java Bean |  |
| ORM | Object-Relational Mapping |  |
| Admin | Administrator |  |

Table 1‑1: Definitions and Acronyms

## References

* OHRM\_ Software Requirements Specification\_v1.0\_EN.docx
* OHRM\_Data Design\_v1.0\_EN.docx
* Software Architecture Design Illuminated Book

## Overview

The Software Architecture Document contains the following subsections:

* **Section 1**: Provide an overview of entire Software Architecture Document.
* **Section 2**: Choice of Architecture Design
* **Section 3**: Architectural Representation
* **Section 4**: Architectural Goals and Constraints
* **Section 5**: Use-Case view
* **Section 6**: Logical View
* **Section 7**: Process View
* **Section 8**: Deployment view
* **Section 9**: Quality

# CHOICE OF ARCHITECTURE DESIGN

## MVC Model

The purpose of OHRM is developing as a crowdfunding website, where people can start their ideas and make it real. The system of OHRM is structured based on MVC combined with layered architecture.

### MVC Model Overview

The **model-view-controller or MVC** is software architecture commonly used for creating web applications or software. In other words, it's a structure for web applications to follow in order to ensure efficiency and consistency. Many of the most popular frameworks use the MVC architecture, including ASP.NET, CodeIgniter, Zend, Django, and Ruby on Rails. At the same time, there are many web developers who don't use a coding framework yet still set up their applications to follow the MVC structure.

The Model-View-Controller (MVC) design pattern assigns objects in an application one of  
three roles: model, view, or controller. The pattern defines not only the roles objects play in  
the application, it defines the way objects communicate with each other. Each of the three  
types of objects is separated from the others by abstract boundaries and communicates with  
objects of the other types across those boundaries. The collection of objects of a certain MVC  
type in an application is sometimes referred to as a layer—for example, model layer.



Figure 2‑1: MVC Model

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

* **Controller:** The controller can be considered the "middle man" of the application. It works with the user, taking in data, and then working with the model to get the appropriate data or calculation, and then working with the view to show the response to the user.
* **Model:** A model is simply a representation of something we need to deal within our application. It is a "model" for something we must represent in code, such as a book, user, bank account, or whatever. The model is responsible for holding the functions and variables that are involved with whatever it's representing. You can think of a model's logic as the core concept to object oriented programming — models are just our "classes". However, don't let this confuse you as controllers are technically structured as classes as well.
* **View:** Finally, after the controller requests information from the model it sends it to a view. A view is just like the application's templating system — there might be a view for a certain type of page layout *(profile page)*, a mobile view, or a view for a particular theme/skin. A view will contain all of the markup, CSS, and etc. that you traditionally use with creating a static web page.

### Advantages and disadvantages of MVC Model

* Advantages:
* MVC model divides system into components, which can be developed, maintained and upgraded individually without stopping the system.
* Develop tools is useful and easy to use.
* Large of documentary sources.
* Disadvantages:
* For small projects that apply MVC model caused cumbersome, time consuming in development process.
* Time consuming to transits data between components.
* Not suitable for agent-oriented applications such as interactive mobile and robotics applications.
* Multiple pairs of controllers and views based on the same data model make data model change expensive.
* The division between the View and the Controller is not clear in some cases.

### The reasons for choosing MVC Model

* MVC makes parts of system can be developed individually and simultaneously to reduce developing time.
* Better support for test-driven development.
* Tools is useful and documentary source is large makes MVC is easy to develop.
* OHRM system is not complete system, now. We built the system that towards extensibility and maintainability in the future.

## Struts Framework

### Struts Framework Overview

Struts Framework is a software framework developed by Craig Mc Clanahan and sponsored by a group name Apache in Jakarta, that is used to develop Java Web application. By using MVC model, Struts solves lots of problems involving web applications built with Java servlet, JSP, etc that requires high performance. Struts, basically, redefining how Web Programmers think of a structure of their Web Application.

Struts is a set of custom JSP tag libraries. It provides custom JSP tag libraries for demonstrating the properties of beans, managing HTML forms, repeating the data structure types and providing conditional HTML.

A basic Struts process is divided into 5 steps:

* A request is sent from View
* ActionServlet will receive this request, then analyse and inspect it. Then assign the corresponding Action to handle the request, calculate necessary tasks. In this step, ActionServlet has the role of Controller.
* Action will manipulate and process on application’s Model.
* When Action finish its job, it will return control to ActionServlet with a key attached to the returned results. ActionServlet will rely on this key to determine how the results will be displayed.
* ActionServlet replies by sending back a request to view, that is a link to the above results. Then, View will do its job.



**Figure 2-2:** Struts basic process

### Advantages and disadvantages of Struts Framework

* Advantages
* Because Struts (particularly Struts 2) is based on MVC model, so it will be fully leveraged the advantages of MVC
* Easy to customize request life cycles for each Action
* Solve effectively the problems of internalization and localization in web applications
* Provide tag libraries, themes and templates for implementing GUI easily, quickly and increase reusability
* Increase extensibility via supporting plug-in
* Support Portal, Ajax
* Easily intergrate with Spring and Hibernate
* Disadvantages
* Need deeper understanding of the framework and system’s core.
* Not suitable for small system, system that requires small amount of time
* Not suitable for inexperienced developers

### The reasons for choosing .Net Framework

* Consistent with OHRM system.
* There are many plugins and resources which support creating a website using MVC model.
* Most of the members can use and have experience with the framework.

## EJB – Enterprise Java Bean

### EJB Overview

EJB is a componet architecture to develop and deploy component-based distributed applications. EJB is one of J2EE ( Java 2 Platform, Enterprise Edition) components, it is a server-side software component that encapsulates the business logic of an application. The EJB specification is a subset of the Java EE specification. An EJB web container provides a runtime environment for web related software components, including computer security, Java servlet lifecycle management, transaction processing, and other web services.

An EJB will have the following characteristics:

* Contain business logic to manipulate the data
* Is created and managed by a Container
* Handle clients’ access
* Contain metadata, such as transaction and security attributes, separate from beans
* Provide transaction management, state management, resource pooling and security services.

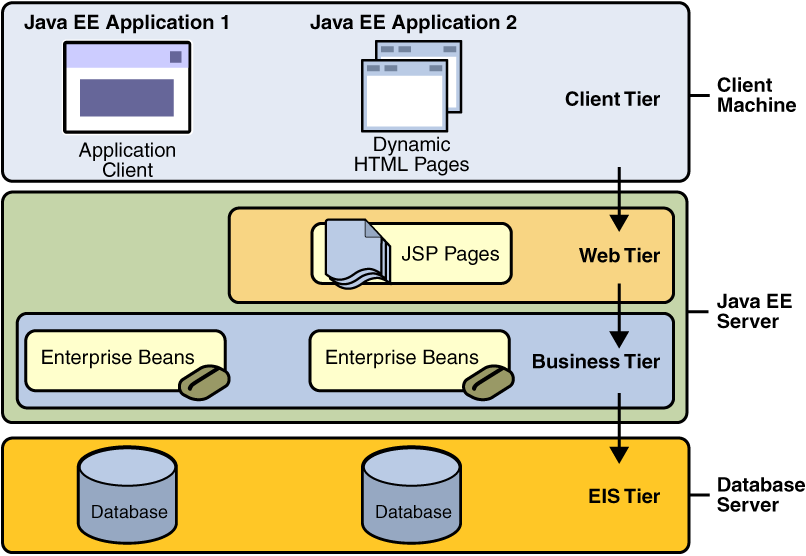


Figure 2‑3: Basic J2EE’s layer

### Advantages and disadvantages of EJB

* Advantages:
* Easy to implement because EJB is just a normal Java Class
* High reusability
* High extensibility
* Provide security service
* Disadvantages:
* Not suitable for small application
* Not suitable for application that does not require transaction or security

### The reason for choosing EJB

* xxxxxxxxxxxxxxxxx
* xxxxxxxxxxxxxxxxxx
* xxxxxxxxxxxxxxxxxxx

## Hibernate SQL

### Hibernate Overview

Hibernate is one of the ORM Framework. Hibernate framework is a framework for persistence layer. Thanks to Hibernate framework that now when you develop applications, you’ll only have to focus on the other layers without having to bother too much about the persistence layer. Hibernate helps store and query data and powerful relationship. Hibernate allows you to query data using Hibernate SQL (HQL) or using original SQL language.

A Hibernate configuration files will have the following characteristics:

* Each table in the database is an object in Hibernate. Therefore, you need to have a Java bean for each table in the database. The Java bean will have getters / setters function and some unconventional exception of Hibernate.
* Each mapping file takes the form of “\*\*\*. Hbm.xml”, has the task of specifying the relationship between the properties of objects and fields in a database table
* A Hibernate.cfg.xml file. This is the first file that is loaded when the app launches Hibernate.



Figure 2‑3: Basic Hibernate Structure

### Advantages and disadvantages of Hibernate

* Advantages:
* Search and sort quickly
* Work effectively with big data, group data
* Support Joining, Aggregating.
* Support handling Transaction.
* Support for multiple applications.
* Share with multiple users.
* Ensure integrity
* Disadvantages:
* “updating”

### The reason for choosing Hibernate

* xxxxxxxxxxxxxxxxx
* xxxxxxxxxxxxxxxxxx
* xxxxxxxxxxxxxxxxxxx

# ARCHITECTURAL REPRESENTATION

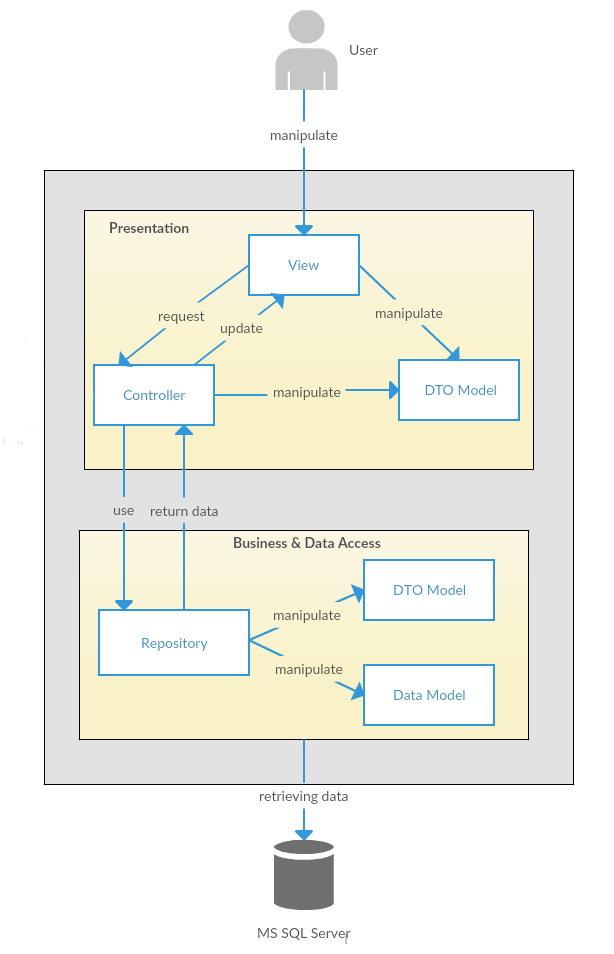


Figure 3‑1: System Overview (chưa chắc lắm, có thể sẽ sửa lại)

We follow MVC architecture to implement the OHRM Project. MVC offers architectural benefits over standard Jquery and AngularJS — it helps us write better-organized and therefore more maintainable code.

* **DTO Model (data transfer object model)** is an object that defines how the data will be sent over the network to remove circular references from data model, Hide particular properties that clients are not supposed to view, omit some properties in order to reduce payload size, flatten object graphs that contain nested objects, to make them more convenient for clients, avoid “over-posting” vulnerabilities and decouple your service layer from your database layer.
* **View** is what is presented to the users and how users interact with the system. The view is expected to render the model in a meaningful way to the user. In OHRM, the view is made with .cshtml file including css, AngularJS and jQuery, it sends user gestures to controller and allows controller to select view.
* **Controller** is the decision maker and the glue between the model and view; it handles user actions and gestures, and responds to user events. For example, in CMS, when a user clicks the “Create” button to create a new contract, the controller for that action is invoked. The controller will then make changes to the contract model. The view will then render the modified contract model to the display so that user can view the new contract he added in the contract list.
* **Data Model** is where the application’s data objects are stored. A model object is in charge of encapsulating application state and one object could be related to other objects establishing a one-to-one or one-to-many relationship.
* **Repository** is intermediate layer which used to separate the controller and the data access layer (database context). It queries the data source for data, maps it to DTO models, processes data and returns data to controller.

# ARCHITECTURAL GOALS AND CONSTRAINTS

* **Availability:**
* The application must be available 95% of time. Users can access to it everywhere from there .Web browser with internet connection.
* **Maintainability:**
  + Coding standards and naming conventions:
    - Output of the project must include coding standards and naming conventions documentations. Implementation code must be easy to maintain.
    - All code must be clearly commented, including class, method documentations.
    - If some components are reused, the documentations of those components must also be included.
* Design:
  + - The design of the system must be loosely coupled that chances on some module will not affect others.
* Logging:
  + - All the errors should be logged, supporting for bug fixing and maintenance.
    - All strange or sensitive situations should also be logged.
* **Usability:**
  + Intuitiveness: all help/error messages are simple to understand; user can know exactly how to do each feature after one time using it.
* **Capacity and scalability:**
  + Throughput, storage and growth requirements.

# Use-case View

* **This application includes two parts:**
* The first part is User module. User module includes registered and guest.
* Next part is Administrator module.

## Guest Group Function

### Login Module



Figure 5‑1: Login module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Login | Guest | This function allows Guest to login into system |

Table 5‑9: Login module UC

## Authenticated User Group Function

### HR/ Resource Manager/Admin Group Function

#### HR/Resource Manager/Admin Common Module



Figure 5‑2: Common Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Edit profile | Authenticated users (HR, Resource Manager, Admin) | Edit personal profile on the system |
| 2 | Logout | Authenticated users (HR, Resource Manager, Admin) | Logout of current session. |

Table 5‑2: Common Module UC

#### HR/Resource Manager Common Module



Figure 5‑3: Recruitment Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Search recruitment | HR/Resource Manager | Search for needed recruitment |
| 2 | View recruitment details | HR/Resource Manager | View all information about the recruitment |
| 3 | Remove recruitment | HR/Resource Manager | Remove recruitment out of the system after finish or cancel |
| 4 | Edit recruitment | HR/Resource Manager | Edit information of recruitment |
| 5 | Import file | HR/Resource Manager | Import file contains list of engineer |
| 6 | Edit recruited engineer | HR/Resource Manager | Edit information of recruited engineer |
| 7 | Remove recruited engineer | HR/Resource Manager | Remove engineer out of recruitment |
| 8 | Add recruited engineer | HR/Resource Manager | Add engineer to recruitment list |

Table 5‑3: Recruitment Module UC

### Resource Manager Group Function

#### Project Management Module



Figure 5‑4: Registered Common Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Search project | Resource Manager | Search project by name or ID |
| 2 | Edit project | Resource Manager | Edit project’s information |
| 3 | Assign engineer(s) to project | Resource Manager | Assign engineer(s) to demanding project |
| 4 | Edit project member position | Resource Manager | Edit position of member(s) in project |

Table 5‑4: Registered Common Module UC

#### View Report Module



Figure 5‑5: View Report Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Generate busy rate statistics report | Resource Manager | Generate busy rate statistics report |
| 2 | Generate efficiency statistics report | Resource Manager | Generate efficiency statistics report |
| 3 | Generate engineer's efficiency statistics report | Resource Manager | Generate engineer's efficiency statistics report |
| 4 | Print report | Resource Manager | Print report |

Table 5‑5: View Report Module UC

#### Plan Management Module



Figure 5‑6: View Personal Page

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Search plan | Resource Manager | Search plan by name of ID |
| 2 | Create plan | Resource Manager | Create new plan |
| 3 | View plan detail | Resource Manager | View detail information of plan |
| 4 | Planning resource | Resource Manager | Plan, list all required resource |
| 5 | Edit plan detail | Resource Manager | Edit detail information of plan |
| 6 | Create project | Resource Manager | Create new project |
| 7 | Edit resource plan | Resource Manager | Edit resource plan |
| 8 | Create recruitment | Resource Manager | Create a recruitment |

Table 5‑6: View Personal Page UC

### Admin Group Function

#### Manage Account Module



Figure 5‑7: Admin Manage Account Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Search account | Admin | Search an account |
| 2 | Create account | Admin | Create an account to access the system |
| 3 | Edit account | Admin | Edit and update an account information |

Table 5‑7: Admin Manage Account Module UC

#### Edit system configuration



Figure 5‑8: Admin Edit Configuration Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Edit system configuration | Admin | Edit basic configuration of system |

Table 5‑8: Admin Edit Configuration Module UC

# Logical View

## Overview

Logical View includes Package diagram and Class diagram. Package diagram describes the organization of packages and elements. Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages; from modeling the domain-specific data structure to detailed design of the target system

* + Controller contain the interface between:
    - Associated models
    - Associated views
    - The input devices (e.g., keyboard, pointing device, time).
    - Send commands to the model to update the model's state.
  + Model is:
    - the domain-specific software simulation
    - Or implementation of the application's central structure.
  + View: deal with everything graphical:
    - Requests data from their model
    - Display the data.
  + Repository:
    - Create queries to DB.
    - Process data.
    - Return to controller.

## Architecturally Significant Design Packages (còn update)

* **Model:**
* **Controller:**
* **Repository:**
* **DTOs:**
* **View:**

# Process view (còn update)

# Deployment View (còn update)

# Quality

Reference to: OHRM\_Software requirement specification\_v1.0\_EN.docx