RMS architecture design

|  |
| --- |
| Project name: Resource Management System  Version: 0.1 |

# Signature page

***MP Software:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AUTHOR  Project Mangement  (Dd/mm/yy) |  | REVIEWER  Quality Assurance  (Dd/mm/yy) |  | APPROVAL  Project Director  (Dd/mm/yy) |
|  |  |  |  |  |

***Customer:***

|  |  |  |  |
| --- | --- | --- | --- |
|  | REVIEWER  Position  (Dd/mm/yy) |  | APPROVAL  Position  (Dd/mm/yy) |
|  |  |  |  |
|  |

REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 21/06/2017 | 0.1 | Create new | Dao Ngoc Son, Ngo Tri Te, Nguyen Thai Phuong |
| *23/06/2017* | *0.2* | *Update* | *Ngo Tri Te, Nguyen Thai Phuong* |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[REVISION HISTORY 3](#_Toc485885921)

[TABLE OF CONTENT 4](#_Toc485885922)

[Definitions and Acronyms 5](#_Toc485885923)

[I. OVERVIEW 6](#_Toc485885924)

[1. Purpose of this document 6](#_Toc485885925)

[2. Scope 6](#_Toc485885926)

[3. Reference 6](#_Toc485885927)

[II. Content 7](#_Toc485885928)

[1. Application architecture 7](#_Toc485885929)

[1.1. Liferay Architecture 7](#_Toc485885930)

[1.2. Functional model 8](#_Toc485885931)

[2. Database Design 10](#_Toc485885932)

[3. Infrastructure architecture 10](#_Toc485885933)

[4. Other architectural solutions 12](#_Toc485885934)

[4.1. Security architecture 12](#_Toc485885935)

[4.2. Backup architecture and data recovery 13](#_Toc485885936)

[5. Performance requirements 13](#_Toc485885937)

[6. Requirements that affect the architecture 14](#_Toc485885938)

# Definitions and Acronyms

Help: Define, or provide references to documents or annexes containing the definition of all terms and acronyms required to properly understand this Plan.

| Acronym | Definition | Note |
| --- | --- | --- |
| RMS | Human Resource Management |  |
| DB | Database |  |
| HR | Human Resource |  |
| JVM | Java Virtual Machine |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# OVeRVIEW

## Purpose of this document

This document describes the proposed prototyping approach for designing the human resource management system. It is based on the user requirement. It will form the basis of the planning of the development phase and will be updated using the results from the various development iterations to come to complete architectural design. It is intended for review by members of the project, notably the system architects of the development team.

## Scope

This document specifies other architectures of RMS projects including:

- Application Architecture

- Database Architecture

- Infrastructure Architecture

- Other Architecture

- Performance requirement

- Architecture requirement

## Reference

| STT | Name | Source |
| --- | --- | --- |
|  |  |  |

# Content

## Infrastructure architecture

The production environment for a typical Liferay deployment consists of a number of HTTP/web servers, application servers or servlet containers, and database servers playing specific roles. For example, the HTTP server’s main role is to load balance the entire system. It also serves as a proxy and a single entry point to the system shielding it from the Internet. The web server is the front-end of the system and its main role is to fulfill web requests. All the business components will reside in application servers. The database server is responsible for storing all data. An example configuration of these servers is captured in the diagram below.



Hardware platforms

|  |  |
| --- | --- |
| Name | Information |
| Server | Operation System : Ubuntu 16.04, JVM 1.8, Oracle 12g Express |
| Chip | Xeon 2.4 Gh 4 Core |
| Ram | 16GB/10000 rpm |
| HDD | 1TB x 2 |
| Network speed | 2Mbs/s |
| Network Card | 1GB x 2 |

Software platforms

|  |  |
| --- | --- |
| Name | Information |
| Browser | Google Chorme 40 - 58.0.3 , Firefox , IE , Safari |
| Liferay | Liferay Community Edition 6.2 |
| Webserver | Tomcat 7.0.62 |
| Oracle | 12g Express |
| Operation System | Ubuntu 16.04 |

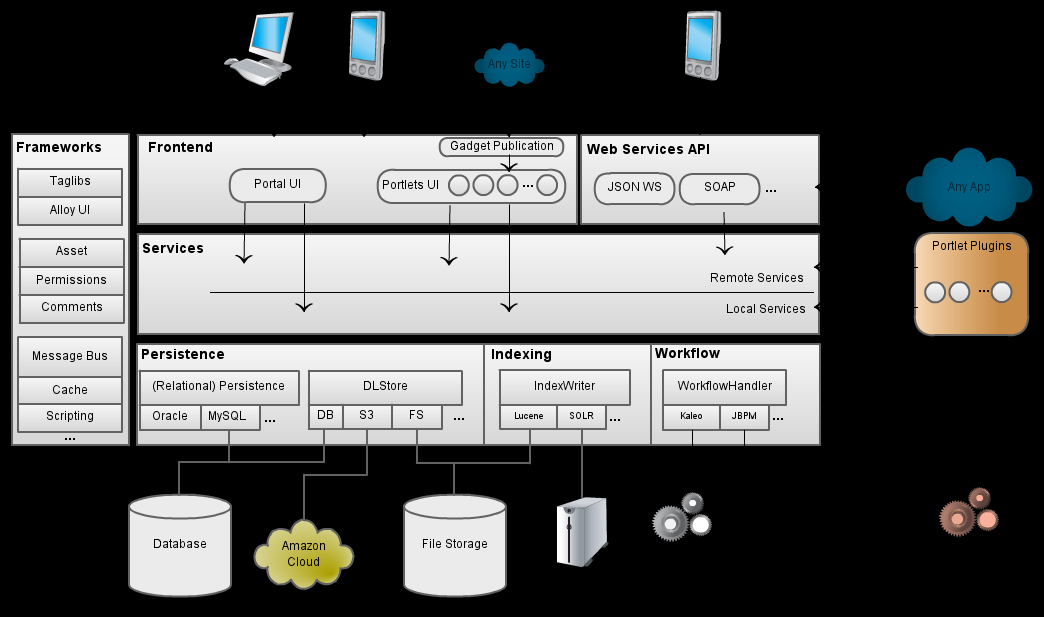
## Application architecture

### Liferay Architecture

Liferay is available in two main versions: Liferay Portal Enterprise Edition and the Liferay Portal Community Edition .While the Enterprise version is an enterprise-tested and refined version with a long-term support, both versions share the same architecture and features. This document will present the Liferay solution specialising in the Liferay 6.2. CE GA5 version.

Liferay is a an enterprise web platform that contains among others a robust collaboration suite. Being a part of Liferay’s development platform the collaboration suite inherits the strengths that Liferay offers in the areas of user management, security management and integration with other systems.

Liferay system is devided into 3 layers: Presentation, Services and Database access :



- Presentation & User interaction: deals with the presentation logic and page rendering.

- The Service layer is related to the business logic and manages the accesses to the resource layer (Spring 3.0.5 with POJO implementation as business service component, Liferay Service builder generated implementation classes).

- The data access integration component is responsible for the access to the enterprise information system (databases) (Hibernate for the implementation of the data access layer).

### Functional model

Base on user requirements, RMS system is devided into 7 main subsystems :

* Accounts management
* HR records management
* Timekeeping management
* Leave management
* Timesheet management
* Assets management
* Administration



|  |  |
| --- | --- |
| **Account management** | |
|  | *Login* |
|  | *Logout* |
| **HR records management** | |
|  | *Create new HR record* |
|  | *Update HR record* |
|  | *Delete HR record* |
|  | *Import HRs list* |
|  | *Export HRs list* |
|  | *Filter* |
|  | *Print* |
|  | *View an employee’s HR record* |
| **Timekeeping management** | |
|  | *Import data from excel file* |
|  | *Display timekeeping document* |
|  | *Filter* |
|  | *Print* |
| **Leave management** | |
|  | *Create new leave application* |
|  | *Update leave application* |
|  | *Delete leave application* |
|  | *Approve leave application* |
|  | *Filter* |
|  | *Print* |
| **Timesheet management** | |
|  | *Create new timesheet* |
|  | *Update timesheet* |
|  | *Delete timesheet* |
|  | *View an employee’s timesheet* |
|  | *Approve timesheet* |
|  | *Filter* |
|  | *Print* |
| **Assets management** | |
|  | *Create new asset information* |
|  | *Update asset information* |
|  | *Delete asset* |
|  | *Export assets list* |
|  | *Filter* |
|  | *Print* |
| **Administration** | |
|  | *Permissive* |

## Database Design

**RMS database**

**Category Database**

**Administration database**

***RMS database components***:

* Administration database: store data for Administration function, including information about users groups, permissions to access different functions.
* Category database: store data for different types of category which are used in main function
* RMS database: store data for main functions : HR records management, timekeeping management, leave management, timesheet management, assets management

## Sercurity architecture



The system is secured to each user, each user is assigned an account to access the program and the permission to access each function of the system and the functions and duties of each employee in the department.

1. The login account is synchronized with the join domain account

Account join domain -> Check username + pass -> server (ubuntu)

2. Check the authorized functions

3. Database access rights on any database? Detailed permissions corresponding

### Backup architecture and data recovery

* Regular backups: The program is designed and configured to be able to run automatically without the system administrator.
* At a pre-setup time every day (eg 11h30 and 16h30), the program will automatically run the function Backup data to the storage device, this device must ensure the storage capacity. Storage size as well as storage time.
* Support for user-requested backup mechanisms: Scheduled automatic backups (daily / weekly / monthly), manual backups on demand, full backups or backups of changed data. The system allows to save backup versions, allowing the administrator to restore the system at any time.
* Rapid recovery mechanism, allowing data recovery at any time or based on the time of the incident.
* Store all backup versions of your system data.
* Back up and restore data with high reliability and availability

### Data growth architecture in the future

* These are the following ways to meet the expectation for data growth in the furture:

1. **Using tools to control all information about indexes and it distribution**. The optimizer uses of this information are used to determine the path with minimum cost. Not updating or losing statical information will make the optimization tools inffective that causes increasing response time.
2. **Create the optimized index**. SQL optimizer relies heavily on the index are defined for specific table. Index has 2 effects: no index would slow SELECT query speed, too many index will slow queries DML (Data Manipulation Language). Therefore, you need to select the right balance of indexes. Besides the index number, the fields and their order are also important. When creating the index, estimates the number of unique values of the column will be with each individual school
3. **Determine the expected growth rate**. As mentioned above, the index have a negative effect on DML queries. One way to minimize this effect is to determine the reasonable value for the fill factor when creating index. When the index is created, the data for the indexed column is stored on the hard drive. When a new row of data is added to the table or the value in the indexed column changes, the database reorganizes the storage area of the data to conserve space for the new row. This reorganization may take extra cost for DML queries. However, if you know the growth rate of the data, let's first define the index growth.
4. **Specifies optimization hints in SELECT**. Although in most cases the optimizer will select the appropriate index for the statistics table, but it would be better to occasionally specify the index name in the SELECT query..
5. **Using EXPLAIN**. Most databases return execution maps for all SELECT statements generated by the optimizer. This diagram is useful for optimizing SQL queries.
6. **Create job scheduler to aggregate the report data**
7. **Select limited data**. The less data you collect, the faster your query run. The better option of client-side filtering is to filter as much as possible to the server side. This will reduce the data sent on the transmission line and increase the speed.
8. **Delete the index before loading the data**. Considering delete index in the table before loading a large amounts of data. This makes the insert statement run faster. Each time the insert statement completes, you can create the index. If you are inserting thousands lines in an online system, use the clipboard to load the data. Make sure the clipboard does not have an index. Moving data from one table to another is much faster than downloading data from external sources, you can delete the index in the main table, move the data from the temporary table to the last table, and finally recreate the index.

## Performance requirements

System implementation time meets the following requirements:

* + Time of accessing data and displaying details of an object (e.g., view details of text, view profile, etc.) is no more than 05 seconds.
  + Simply searching time is no more than 05 seconds
  + Complicated search (join more than 3 tables to aggregate information), is no more than 08 seconds.
  + Daily report time is no more than 10 seconds
  + Reporting time on data of previous days is no more than 15 seconds.
* The system must ensure the number of simultaneous transactions:

## Requirements that affect the architecture

The most important factors that can affect the system architecture such as:

* Online / offline processing requirements: The software needs to be developed so that the system is capable of working 24 hours per day.
* Requirement environment: Oracle 12c Database, Tomcat WebServer, Linux Server Operating System