**Payroll Management System**

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20/12/2019 | 1.0 | Final | Quản Trọng Tú |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 2

1.1 Purpose 2

1.2 Scope 2

1.3 Definitions, Acronyms, and Abbreviations 2

1.4 References 2

1.5 Overview 2

2. Architectural Representation 2

3. Architectural Goals and Constraints 2

4. Use-Case View 2

4.1 Use-Case Realizations 2

5. Logical View 2

5.1 Overview 2

5.2 Architecturally Significant Design Packages 2

6. Process View 2

7. Deployment View 2

8. Implementation View 2

8.1 Overview 2

8.2 Layers 2

9. Data View (optional) 2

10. Size and Performance 2

11. Quality 2

# Introduction

## 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 which have been made on the system.

## Scope

This document applies to the Payroll Management System which will be developed by JavaTech Group.

## Definitions, Acronyms, and Abbreviations

* Employee – a person who work for the IT company.
* Accountant – an employee who manage salary.
* Administrator – a person who manage system.

## References

None.

## Overview

In the following section, architectural design of the Payroll Management System is provided in detail. First, the primary software architecture of the system will be defined. Then, there are further discussion about the goals and constraints that will be imposed upon the quality of the final product, which including but not limited to security, distribution and reuse. In the precedence sections, the key views of the system are demonstrated to depict different aspects of the system. Lastly, criteria concerning with size, performance and quality of the system will be proposed.

# Architectural Representation

This document presents the architectural as a series of mandatory views: Use-Case View, Logical View and Data View. These views are presented as Visual Paradigm Community Edition Models and use the Unified Modeling Language (UML).

**Use-Case View**

* **Audience:** all the stakeholders of the system, including the end-users.
* **Area:** describes the set of scenarios and/or use cases that represent significant, central functionality to the system.
* **Related artifacts:** Use-Case Model, Analysis Model, Use-Case-Realization documents.

**Logical View**

* **Audience:** designers, programmers.
* **Area:** functional requirements: describes the design’s object model.
* **Related artifacts:** Design Model.

**Data View**

* **Audience:** data specialists, database administrators.
* **Area:** persistence: describes the architecturally significant persistent elements in the data model.
* **Related artifacts**: Data Model.

# Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

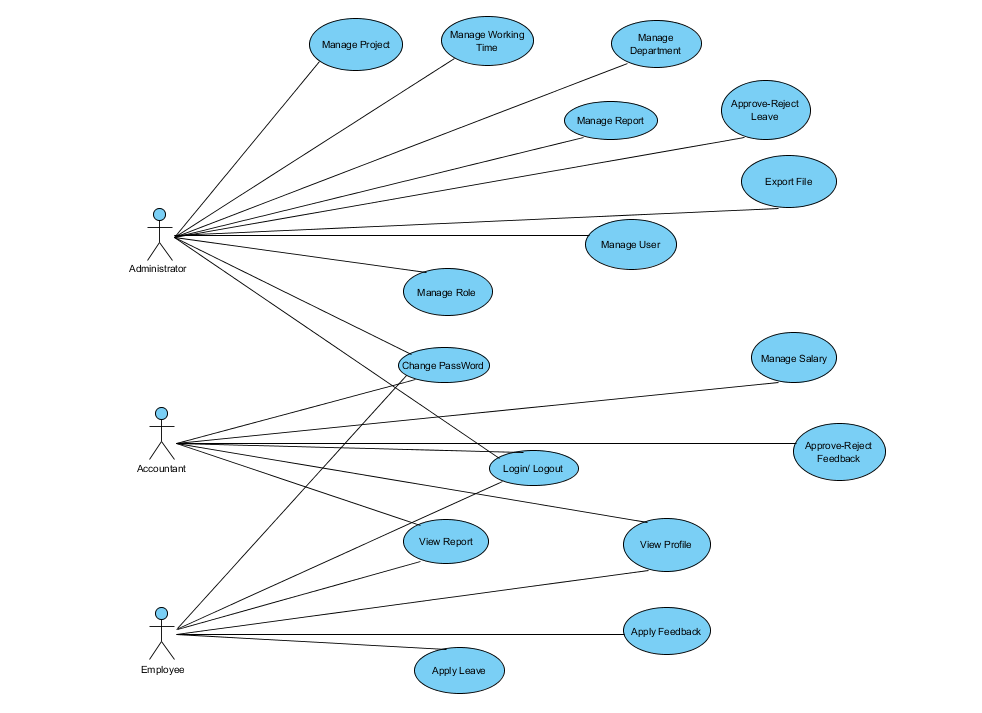
* The Payroll Management System must be designed to fulfills all system requirements specified in requirements definition.
* The Payroll Management design must be structured to be robust, easy to change if and when functional requirements change.
* The Payroll Management System must be designed to allow the re-use of business logic across applications; therefore, the design separate the three components: model, view and controller.
* The separation of the three components: model, view and controller are also necessary to provide a convenient cooperation between different development teams.
* The Payroll Management System will run on a dedicated platform with access to a database.
* The Payroll Management website provides most of the content display. An interface to this system must be capable of handling large traffic volumes.

# Use-Case View

A description of the Use-Case View of the system architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

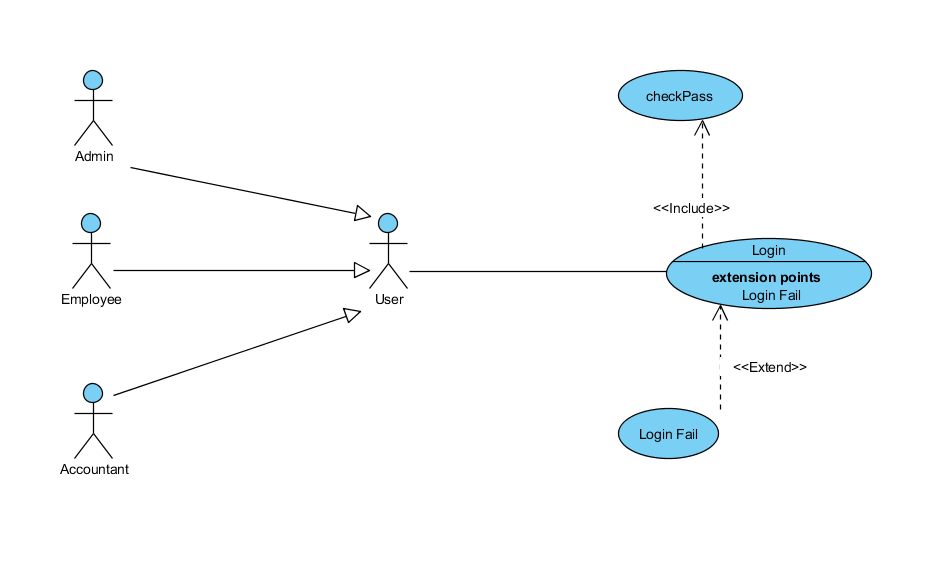
The significant use cases in this system are listed below:

* Login/Logout
* Change Password
* Manage Report
* View Report
* Manage Department
* Manage Role
* View Profile
* Export File
* Manage User
* Approve-Reject Leave
* Apply Feedback
* Approve-Reject Feedback
* Manage Salary
* Apply Leave
* Manage Project
* Manage Working Time

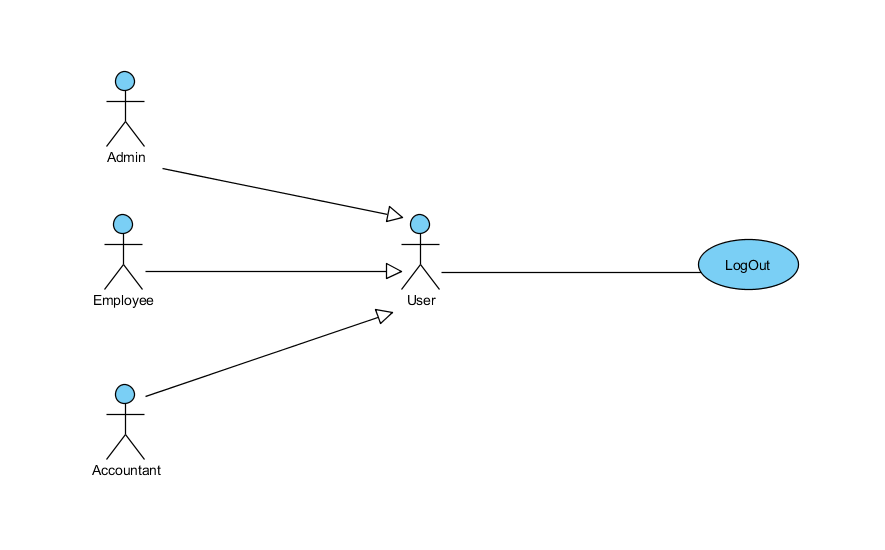


## Use-Case Realizations

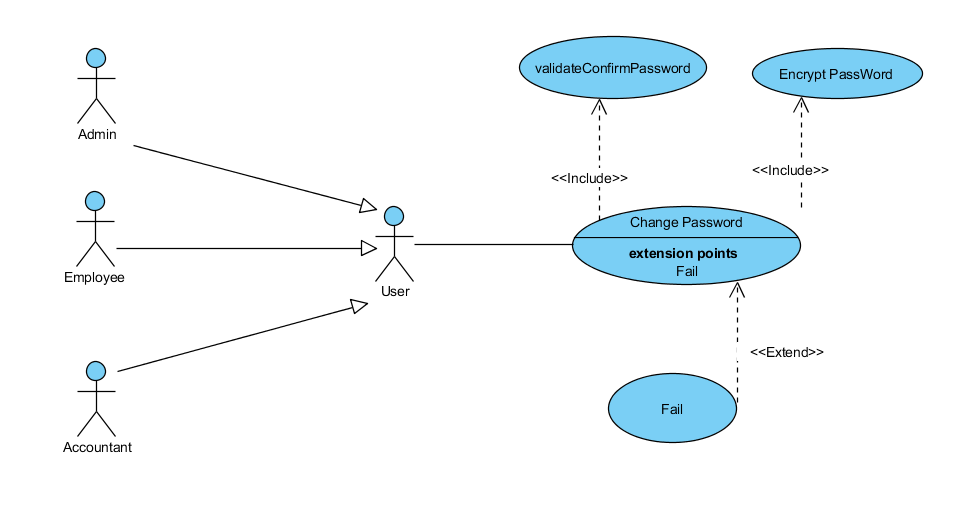
* **Login:**



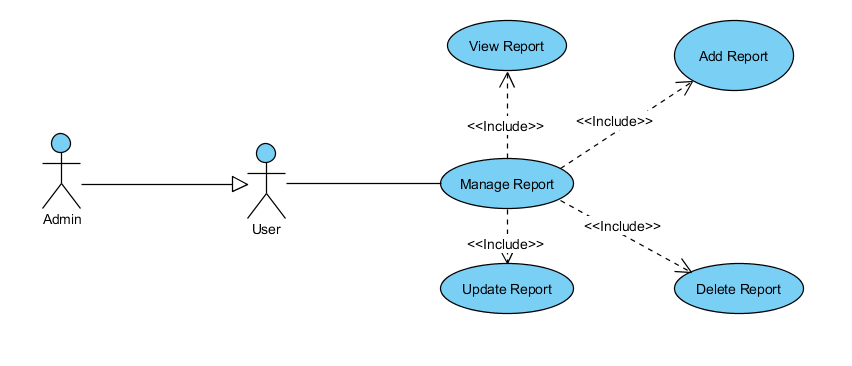
* + - **Brief Description:** A user login into System.
    - **Specification:** See Use-Case-Realization Specification: Login.
* **Log out:**

****

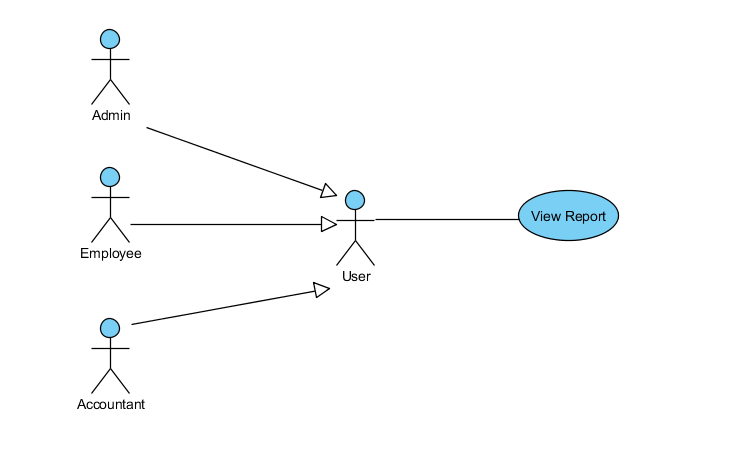
* + - **Brief Description:** A user logout to the system.
    - **Specification:** See Use-Case-Realization Specification: Log out
* **Change Password:**

****

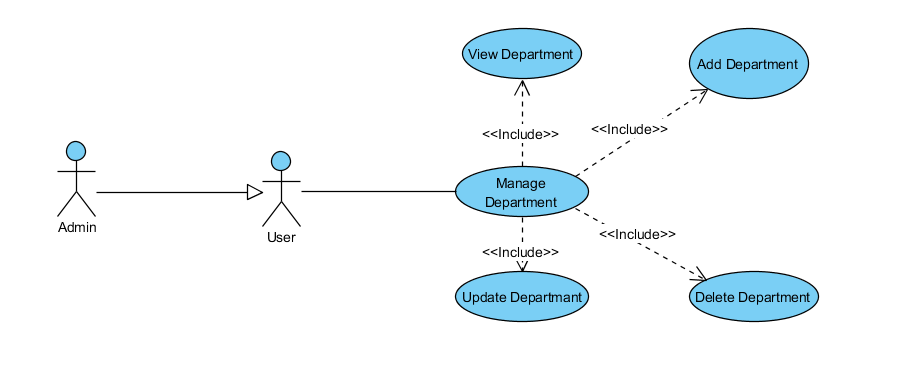
* **Brief Description:** A user change Password
* **Specification:** See Use-Case-Realization Specification: Change Password
* **Manage Report**

****

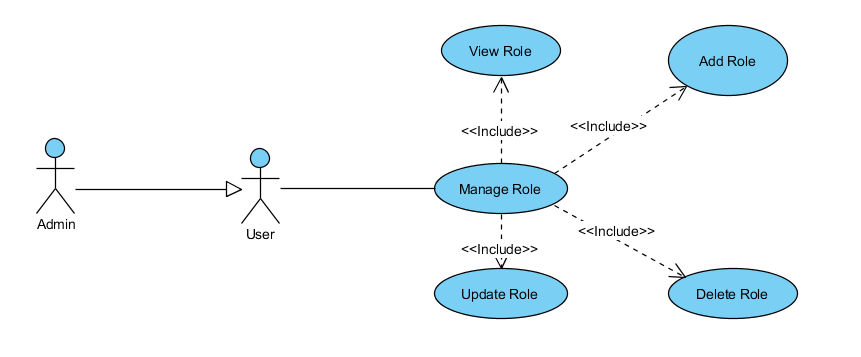
* + **Brief Description:** Admin manages report Salary as Notification
  + **Specification:** See Use-Case-Realization Specification: Manage Report
* **View Report:**

****

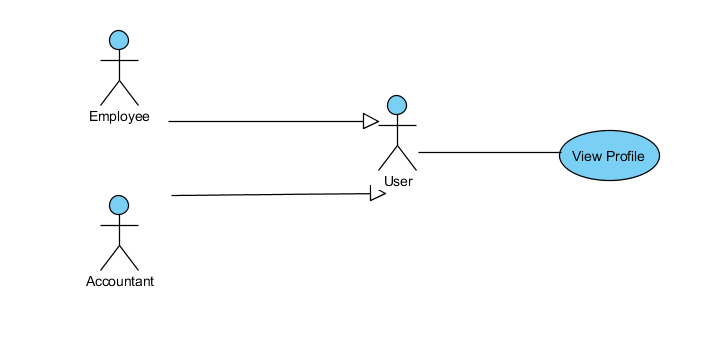
* **Brief Description:** A user can view Report to check Salary
* **Specification:** See Use-Case-Realization Specification: View Report
* **Manage Department:**

****

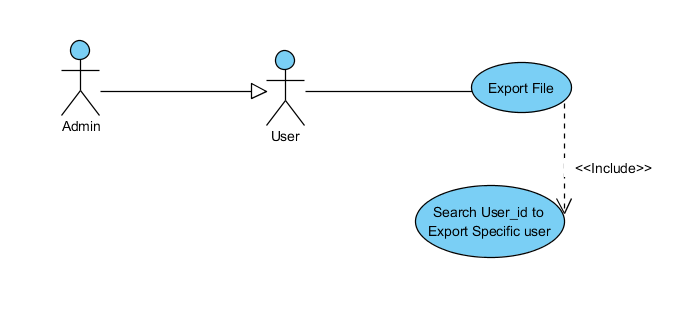
* **Brief Description:** Admin can manage department of company.
* **Specification:** See Use-Case-Realization Specification: Manage Department
* **Manage Role**

****

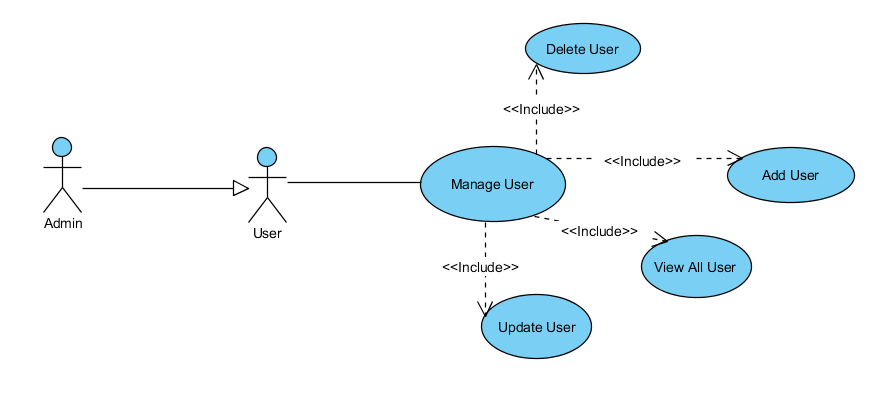
* **Brief Description:** Admin can manage Role of company.
* **Specification:** See Use-Case-Realization Specification: Manage Role
* **View Profile:**

****

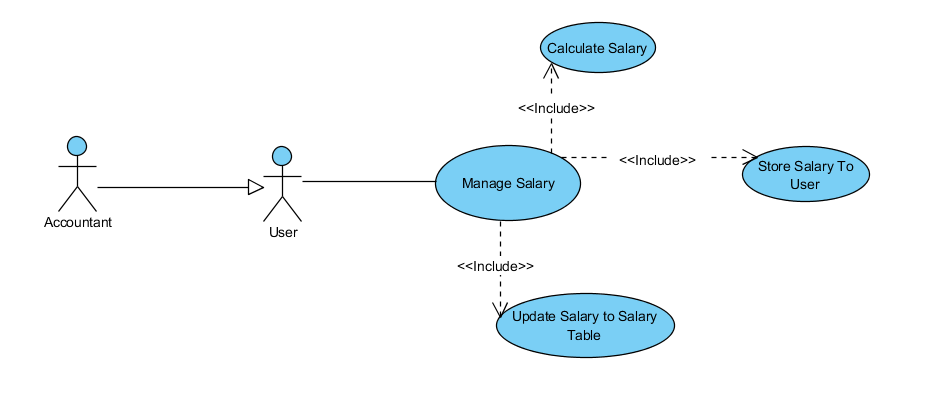
* **Brief Description:** Employee and Accountant can view profile, Admin just view profile through manage user.
* **Specification:** See Use-Case-Realization Specification: View Profile
* **Export File:**

****

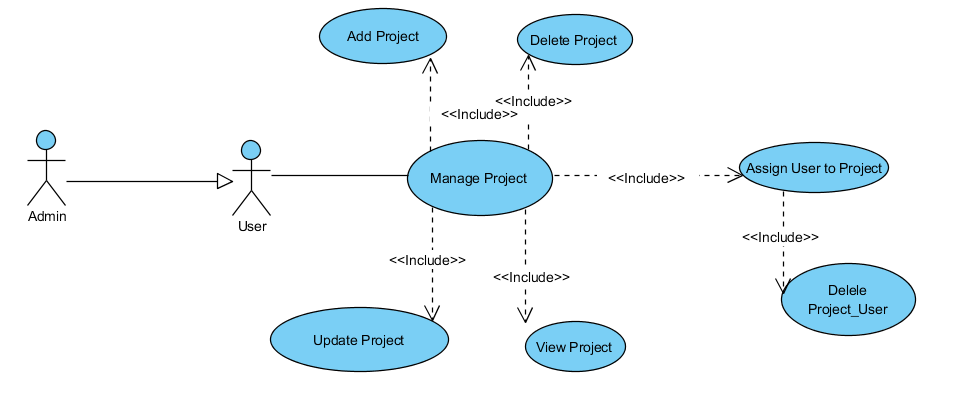
* + **Brief Description:** Admin export user’s profile if it exists.
  + **Specification:** See Use-Case-Realization Specification: Export profile
* **Manage User:**

****

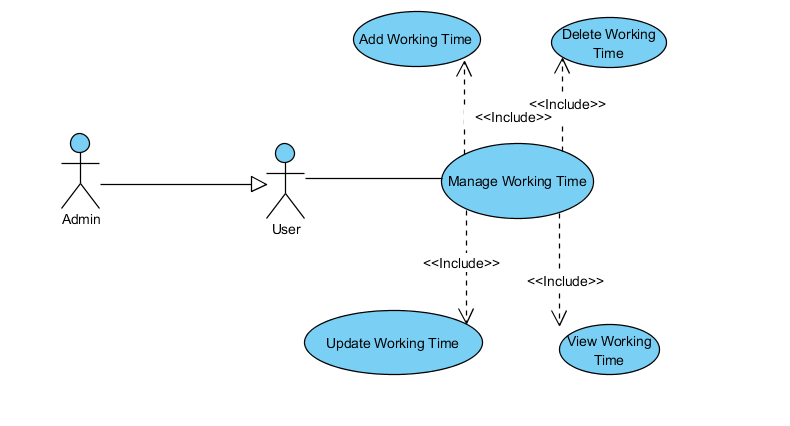
* **Brief Description:** Admin can manage User by sub-function.
* **Specification:** See Use-Case-Realization Specification: Manage User.
* **Manage Salary:**

****

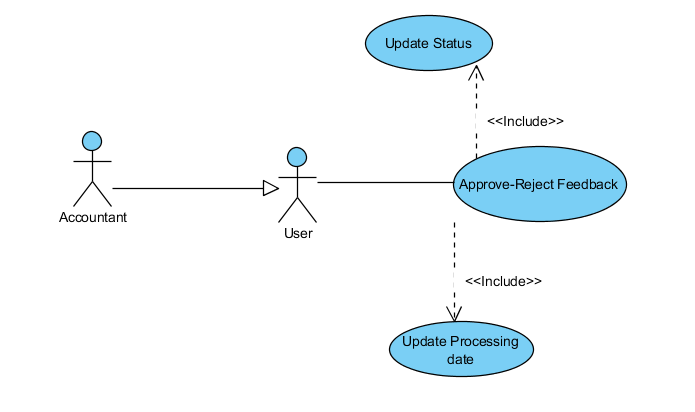
* **Brief Description:** Accountant can manage Salary by sub-function.
* **Specification:** See Use-Case-Realization Specification: Manage Salary
* **Manage Project:**

****

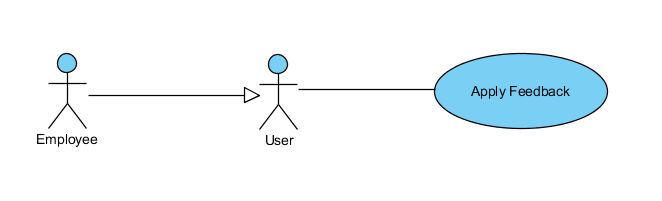
* **Brief Description:** Admin can manage Project by sub-function.
* **Specification:** See Use-Case-Realization Specification: Manage Project.
* **Manage Working Time:**

****

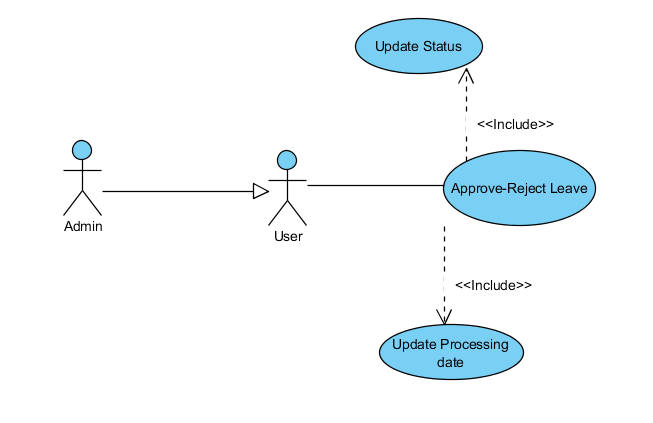
* **Brief Description:** Accountant can manage Working Time by sub-function.
* **Specification:** See Use-Case-Realization Specification: Manage Working Time.
* **Approve-Reject FeedBack:**

****

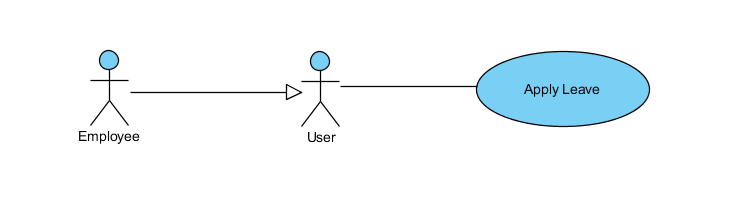
* **Brief Description:** Accountant can Approve-Reject Leave which are belong to user by sub-function.
* **Specification:** See Use-Case-Realization Specification: Approve-Reject Feedback.
* **Apply FeedBack**

****

* **Brief Description:** Employee can Apply Feedback.
* **Specification:** See Use-Case-Realization Specification: Apply Feedback.
* **Approve – Reject Leave:**

****

* **Brief Description:** Admin can Approve-Reject Leave which are belong to user by sub-function.
* **Specification:** See Use-Case-Realization Specification: Approve-Reject Leave.
* **Apply Leave:**

****

* **Brief Description:** Employee can Apply Leave.
* **Specification:** See Use-Case-Realization Specification: Apply Leave.

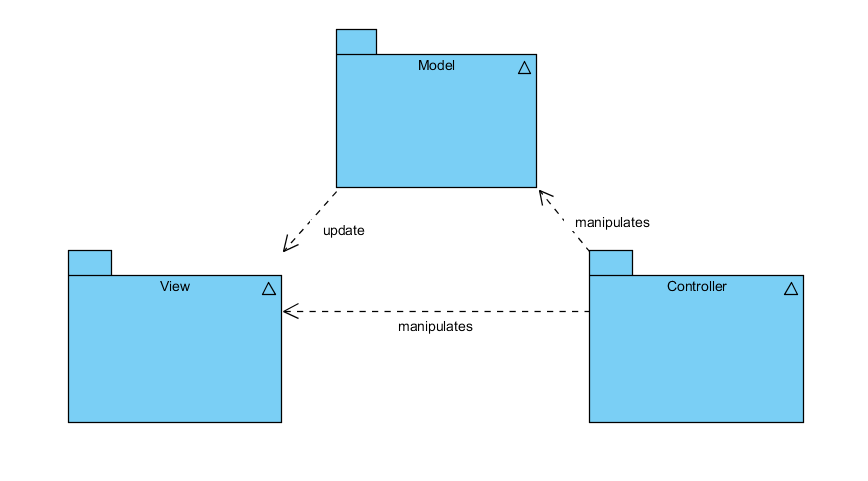
# Logical View

## Overview

A description of the logical view of the architecture. Describes the overall decomposition of the design model in terms of package hierarchy and layers.

The logical view of the Payroll Management System is comprised of 3 significant packages:

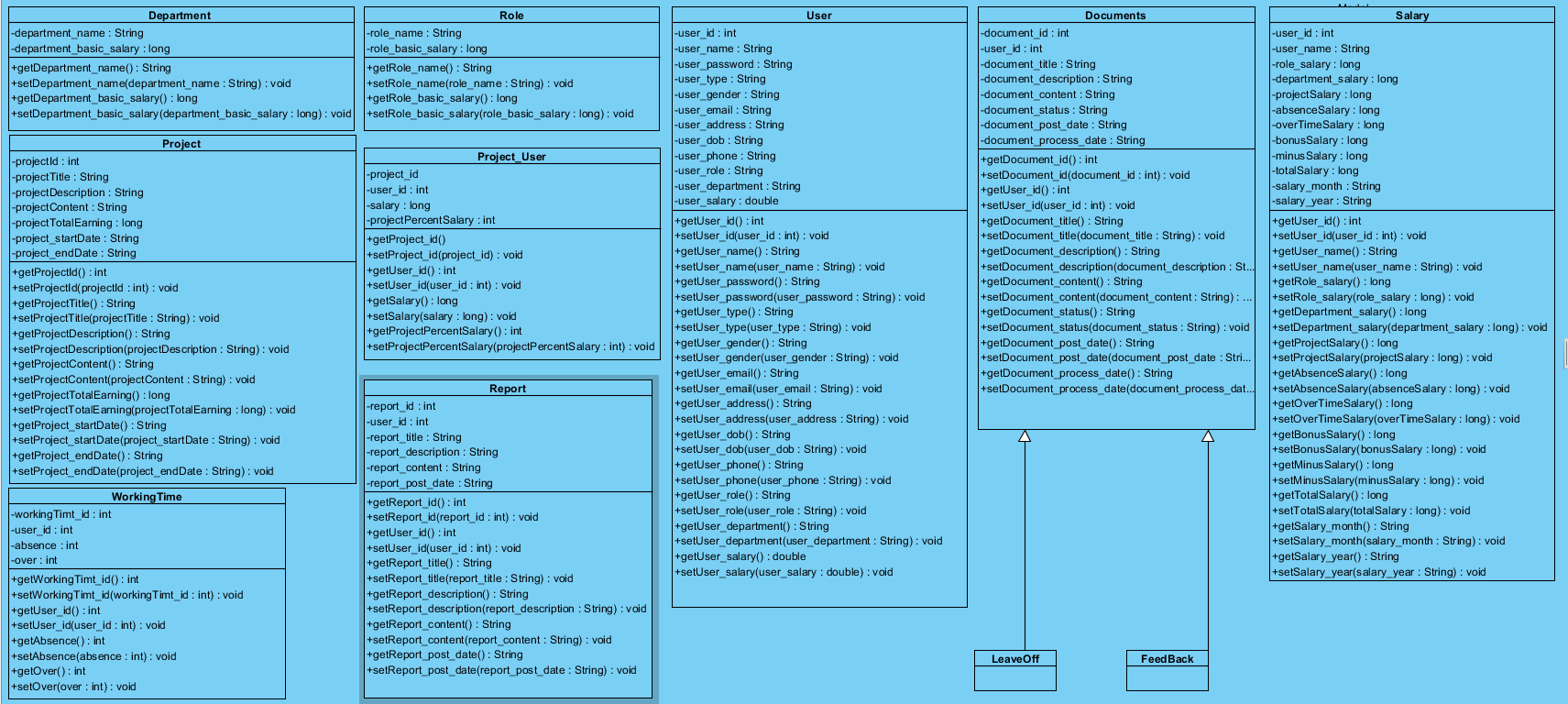
* + - **model:** contains classes that directly manages the data, logic and rules of the Payroll Management System and displayed in the view.
    - **view:** contains classes that generates output representation of information to the user based on changes in the model. View return the text, button,….
    - **controller:** contains classes that can send commands to the model to update the model’s state (e.g., add a new computer); it can also send commands to its associated view to change the view’s presentation of the model (e.g., scrolling through computer’s reviews). Specially, Payroll Management System was built in Java Swing following ORM and OPP design. Therefore, Controller execute event of view.



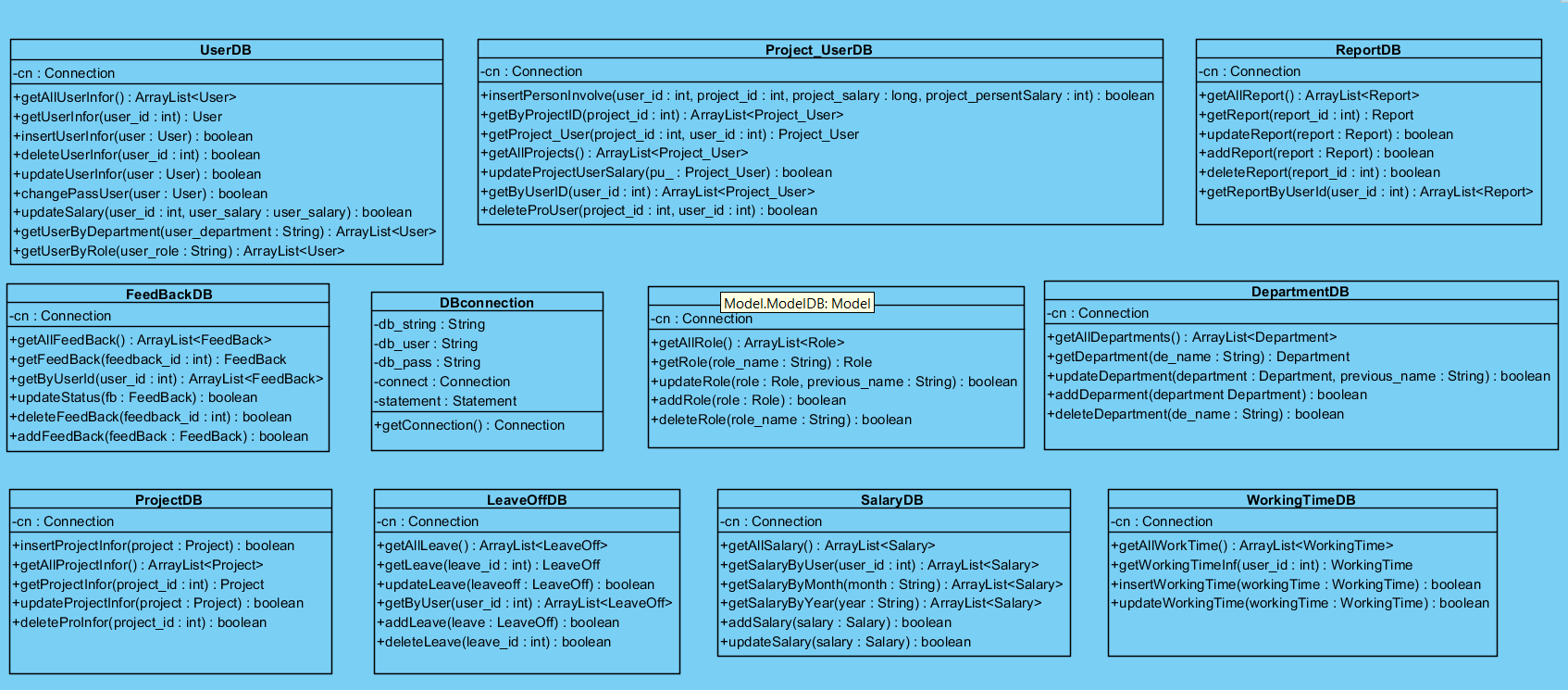
## Architecturally Significant Design Packages

### Package model:

* Entity Class



* Access Database class:

****

|  |  |
| --- | --- |
| **Name** | Model Entities |
| **Brief Description** | Contains classes that directly manages the data, logic and rules of the Payroll Management System and displayed in the view. |
| **Classes** | Department, Role, User, FeedBack, LeaveOff, Documents, Project, Project\_User, Report, Salary, WorkingTime. |

* Class: User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | User | | | | | | |
| **Brief Description** | Data model for User table in database. | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| user\_id | int | Private | True | False | N/A | 1 | N/A |
| user\_name | String | Private | True | False | N/A | N/A | N/A |
| user\_password | String | Private | True | False | N/A | N/A | N/A |
| user\_type | String | Private | True | False | N/A | N/A | N/A |
| user\_gender | String | Private | True | False | N/A | N/A | N/A |
| user\_email | String | Private | True | False | N/A | N/A | N/A |
| user\_address | String | Private | True | False | N/A | N/A | N/A |
| user\_dob | String | Private | True | False | N/A | N/A | N/A |
| user\_phone | String | Private | True | False | N/A | N/A | N/A |
| user\_role | String | Private | True | False | N/A | N/A | N/A |
| user\_department | String | Private | True | False | N/A | N/A | N/A |
| user\_salary | double | Private | True | False | N/A | N/A | N/A |

* Class: Documents

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | Documents | | | | | | |
| **Brief Description** | This is class for LeaveOff and Feedback extends | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| document\_id | int | Private | True | False | N/A | 1 | N/A |
| user\_id | int | Private | True | False | N/A | 1 | N/A |
| document\_title | String | Private | True | False | N/A | N/A | N/A |
| document\_description | String | Private | True | False | N/A | N/A | N/A |
| document\_content | String | Private | True | False | N/A | N/A | N/A |
| document\_status | String | Private | True | False | N/A | N/A | N/A |
| document\_post\_date | String | Private | True | False | N/A | N/A | N/A |
| document\_process\_date | String | Private | True | False | N/A | N/A | N/A |

* Class: Salary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | Salary | | | | | | |
| **Brief Description** | Data model for department table in database. | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| user\_id | int | Private | True | False | N/A | N/A | N/A |
| user\_name | String | Private | True | False | N/A | N/A | N/A |
| role\_salary | long | Private | True | False |  |  |  |
| department\_salary | long | Private | True | False |  |  |  |
| projectSalary | long | Private | True | False |  |  |  |
| absenceSalary | long | Private | True | False |  |  |  |
| overTimeSalary | long | Private | True | False |  |  |  |
| bonusSalary | long | Private | True | False |  |  |  |
| minusSalary | long | Private | True | False |  |  |  |
| totalSalary | long | Private | True | False |  |  |  |
| salary\_month | String | Private | True | False |  |  |  |
| salary\_year | String | Private | True | False |  |  |  |

|  |  |
| --- | --- |
| **Name** | Model Access Database |
| **Brief Description** | Contains classes that directly manages the data, logic and rules of the Payroll Management System and displayed in the view. |
| **Classes** | DBConnection, DepartmentDB, FeedBackDB, LeaveOffDB, ProjectDB, Project\_UserDB, ReportDB, RoleDB, SalaryDB, UserDB, WorkingTimeDB |

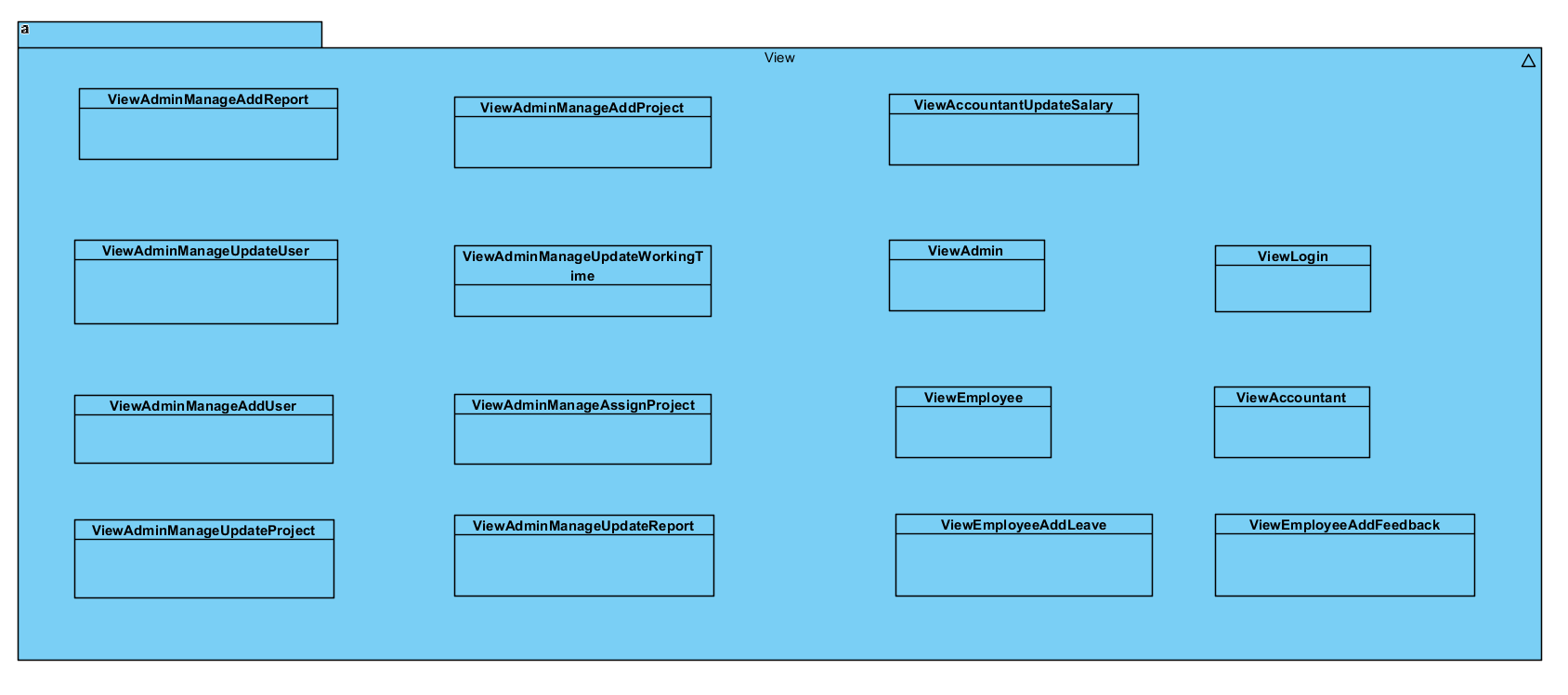
* Class DBConnection:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | DBConnection | | | | | | |
| **Brief Description** | To Connect Database | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| db\_string | String | Private | False | False |  |  |  |
| db\_user | String | Private | False | False |  |  |  |
| db\_pass | String | Private | False | False |  |  |  |
| connect | Connection | Private | False | False |  |  |  |
| statement | Statement | Private | False | False |  |  |  |
| **Operations** | | | | | | | |
| Header | **Return Type** | **Access** | **Scope** | **Specification** | | | |
| getConnection() | Connection | Public | Instance | Connect to database | | | |

* Class UserDB:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | UserDB | | | |
| **Brief Description** | Get Database and store it to User entity | | | |
| **Operations** | | | | |
| Header | **Return Type** | **Access** | **Scope** | **Specification** |
| getAllUserInfor() | ArrayList<User> | Public | Instance | Return all user in database |
| getUserInfor() | User | Public | Instance | Return specific user |
| insertUserInfor() | boolean | Public | Instance | add user to database |
| deleteUserInfor() | boolean | Public | Instance | delete specific user from database |
| updateUserInfor() | boolean | Public | Instance | update user information in database |
| changePassUser() | boolean | Public | Instance | change user password |
| updateSalary() | boolean | Public | Instance | update User salary |
| getUserByRoles() | ArrayList<User> | Public | Instance | Return Users by role |
| getUserByDepartment() | ArrayList<User> | Public | Instance | Return Users by department |

### Package View:

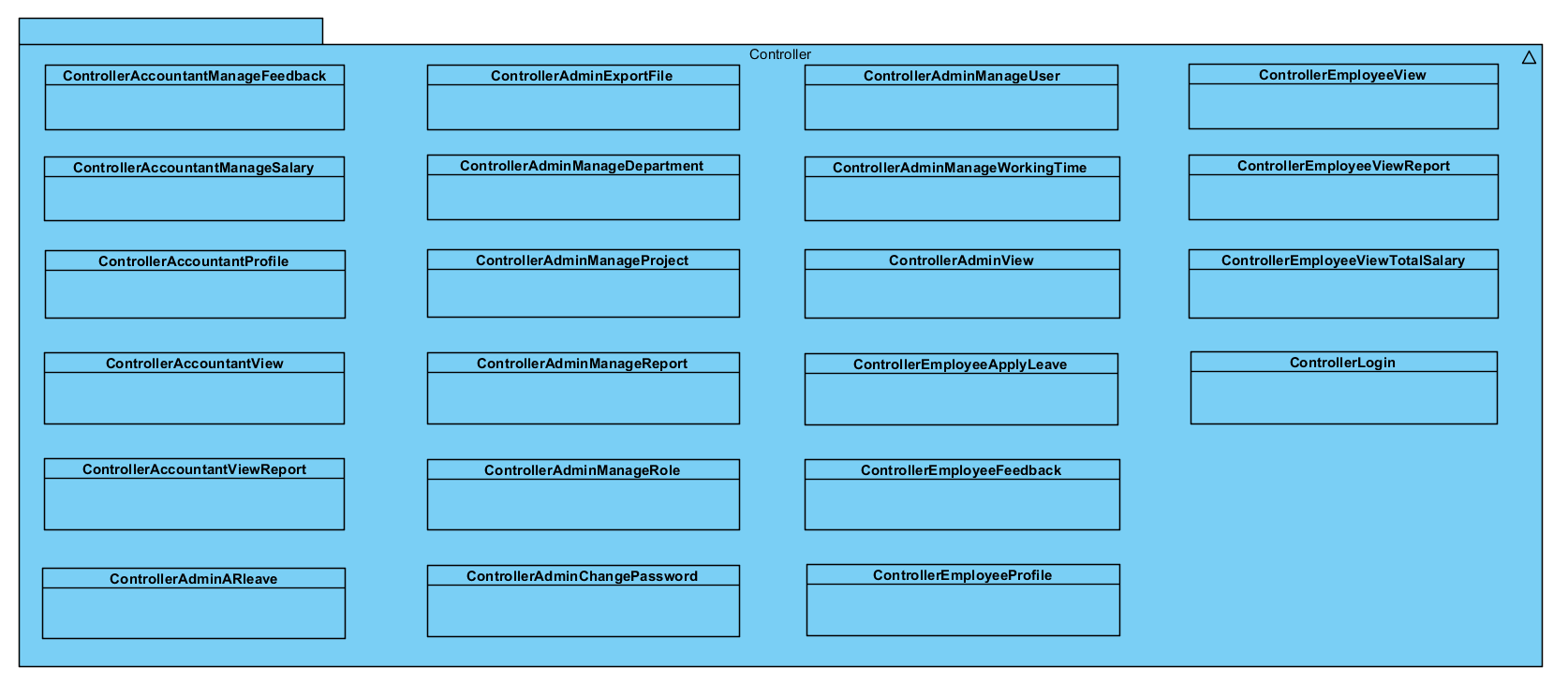


|  |  |
| --- | --- |
| **Name** | view |
| **Brief Description** | Contains classes that generates output representation of  information to the user based on changes in the model. |
| **Classes** | * ViewAdminManageAddReport * ViewAdminManageUpdateUser * ViewAdminManageAddUser * ViewAdminManageUpdateProject * ViewAdminManageAddProject * ViewAdminManageUpdateWorkingTime * ViewAdminManageAssignProject * ViewAdminManageUpdateReport * ViewAccountantUpdateSalary * ViewAdmin * ViewLogin * ViewEmployee * ViewAccountant * ViewEmployeeAddLeave * ViewEmployeeAddFeedback |

* Class: ViewAdminManageAddUser

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | ViewAdminManageAddUser | | | | | | |
| **Brief Description** | Represents the visualization of the data that model contains. | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | | **Mutable** | | **Optional** | **Length** |
| button\_add | **javax.swing.JButton** | **private** | | False | | True |  |
| get\_userAddress | **javax.swing.JTextField** | **private** | | False | | True |  |
| get\_userDepartment | **javax.swing.JComboBox** | **private** | | False | | True |  |
| get\_userDob | **com.toedter.calendar.JDateChooser** | **private** | | False | | True |  |
| get\_userEmail | **javax.swing.JTextField** | **private** | | False | | True |  |
| get\_userGender | **javax.swing.JTextField** | **private** | | False | | True |  |
| get\_userName | **javax.swing.JTextField** | **private** | | False | | True |  |
| get\_userPass | **javax.swing.JPasswordField** | **private** | | False | | True |  |
| get\_userPhone | **javax.swing.JTextField** | **private** | | False | | True |  |
| get\_userRole | **javax.swing.JComboBox** | **private** | | False | | True |  |
| get\_userSalary | **javax.swing.JTextField** | **private** | | False | | True |  |
| get\_userType | **javax.swing.JComboBox** | **private** | | False | | True |  |
| jLabel10 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel11 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel12 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel13 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel3 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel4 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel5 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel6 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel7 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel8 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jLabel9 | **javax.swing.JLabel** | **private** | | False | | True |  |
| jPanel1 | **javax.swing.JPanel** | **private** | | False | | True |  |
| jPanel2 | **javax.swing.JPanel** | **private** | | False | | True |  |
| **Operations** | | | | | | | |
| **Header** | **Return Type** | **Access** | **Scope** | | **Specification** | | | |
| getGet\_userDob() | JDateChooser | Public | Instance | | Return Date to choose | | | |
| formatDate() | void | Public | Instance | | to format JDateChooser | | | |
| buttonAdd() | JButton | Public | Instance | | return button in view | | | |
| getUserName() | JTextField | Public | Instance | | return Text field | | | |
| getUserPass() | JPasswordField | Public | Instance | | return password field | | | |
| getUserType() | JComboBox | Public | Instance | | return box to choose | | | |
| getUserGender() | JTextField | Public | Instance | | return Text field | | | |
| getUserEmail() | JTextField | Public | Instance | | return Text field | | | |
| getUserAddress() | JTextField | Public | Instance | | return Text field | | | |
| getUserPhone() | JTextField | Public | Instance | | return Text field | | | |
| getUserRole() | JComboBox | Public | Instance | | return box to choose | | | |
| getUserDepartment() | JComboBox | Public | Instance | | return box to choose | | | |
| getUserSalary() | JTextField | Public | Instance | | return Text field | | | |

### Package Controller



|  |  |
| --- | --- |
| **Name** | Controller |
| **Brief Description** | Contains classes that directly manages the data, logic and rules of  the Payroll Management System and displayed in the view. |
| **Classes** | * ControllerAccountantManageFeedback * ControllerAccountantManageSalary * ControllerAccountantProfile * ControllerAccountantView * ControllerAccountantViewReport * ControllerAdminARleave * ControllerAdminChangePassword * ControllerAdminExportFile * ControllerAdminManageProject * ControllerAdminManageReport * ControllerAdminManageRole * ControllerAdminManageUser * ControllerAdminManageWorkingTime * ControllerAdminView * ControllerEmployeeApplyLeave * ControllerEmployeeFeedback * ControllerEmployeeProfile * ControllerEmployeeView * ControllerEmployeeViewReport * ControllerEmployeeViewTotalSalary * ControllerLogin |

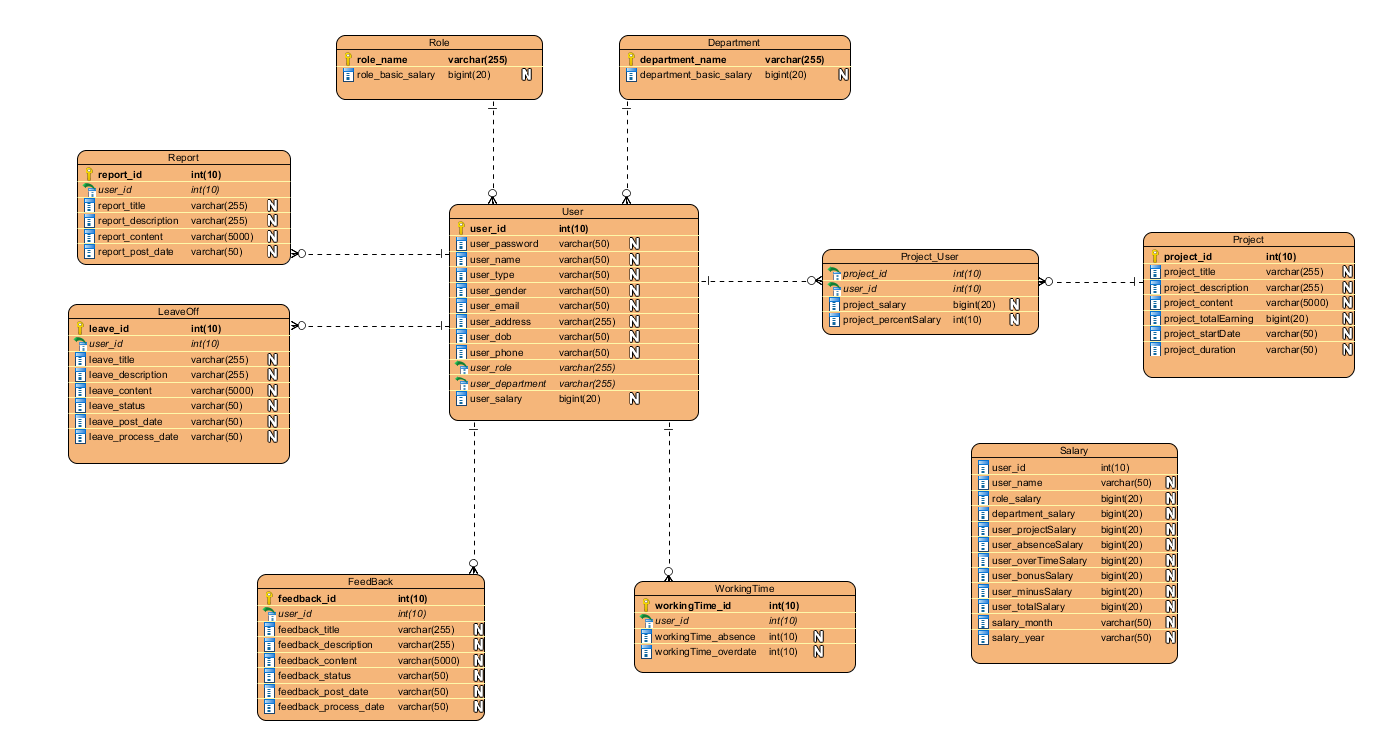
* Class: ControllerAdminView

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller to Manage all of admin’s function | | | | | | | | | |
| **Attributes** | | | | | | | | | | |
| **Name** | **Type** | | **Access** | | **Mutable** | | **Optional** | **Length** | **Min** | **Max** |
| userDB | UserDB | | Private | | False | | False | N/A | N/A | N/A |
| viewLogin | ViewLogin | | Private | | False | | False | N/A | N/A | N/A |
| viewAdmin | ViewAdmin | | Private | | False | | False |  |  |  |
| projectDB | ProjectDB | | Private | | False | | False |  |  |  |
| project\_UserDB | Project\_UserDB | | Private | | False | | False |  |  |  |
| reportDB | ReportDB | | Private | | False | | False |  |  |  |
| leaveOffDB | LeaveOffDB | | Private | | False | | False |  |  |  |
| wdb | WorkingTimeDB | | Private | | False | | False |  |  |  |
| roleDB | RoleDB | | Private | | False | | False |  |  |  |
| departmentDB | DepartmentDB | | Private | | False | | False |  |  |  |
| **Operations** | | | | | | | | | | |
| **Header** | **Return**  **Type** | **Access** | | **Scope** | | **Specification** | | | | |
| logOut() | void | Public | | Instance | | Handling log out request. | | | | |
| changeView() | void | Public | | Instance | | Handling change view request. | | | | |

# Implementation View

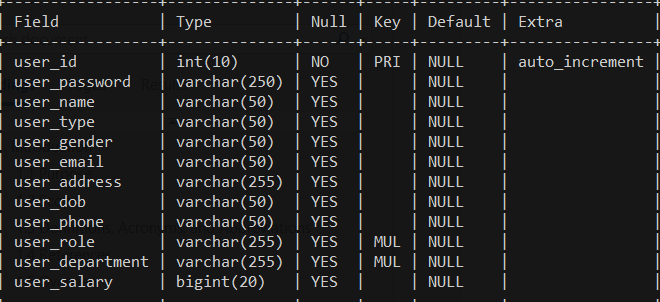
The implementation of the system is strictly driven from the design; therefore, the implementation view will not be considered in this document.

# Data View (optional)

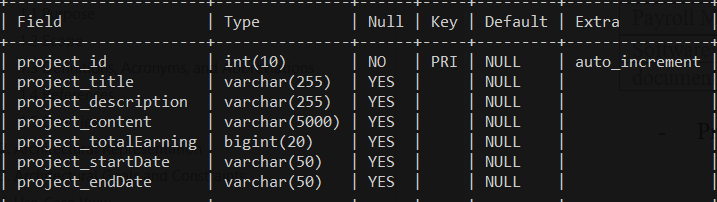


Data Dictionary:

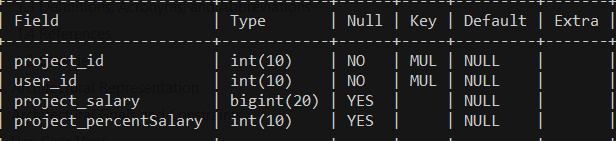
* User:



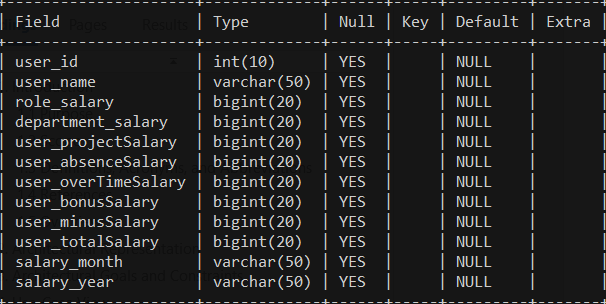
* Project:



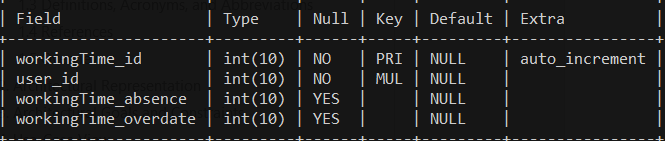
* Project\_User:



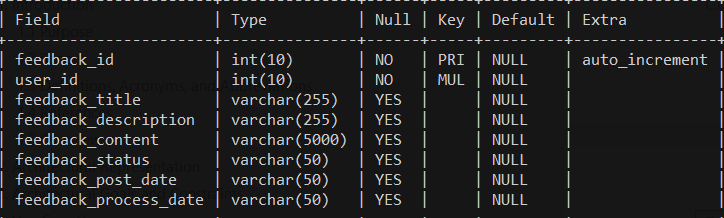
* Salary:



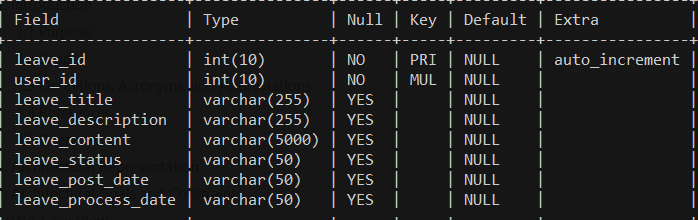
* Working Time:



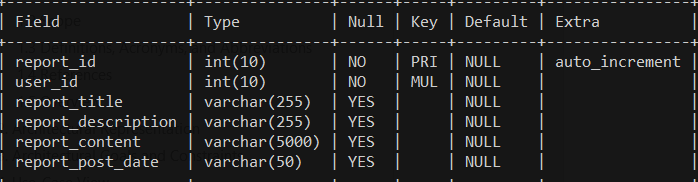
* Feedback:



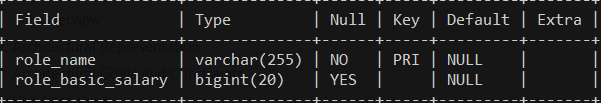
* LeaveOff:



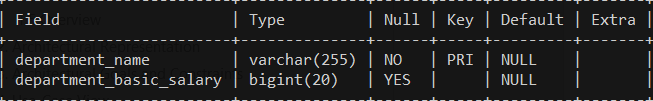
* Report:



* Role:



* Department:



# Size and Performance

The major dimensioning characteristics of the software that impact the architecture and performance constraints:

The system shall support up to 1000 concurrent users against the primary database at any given time, and up to 500 concurrent users against the local servers at any one time.

The system must perform all functions with minimal time delays. The system must also accurately save all information transactions

# Quality

The system architecture supports the quality requirements:

* In order to maintain the highest degree of system integrity, the system is capable of ensuring that all information transitions are saved.
* Databases will be backed up on a daily basis in concern with safety implications.
* This system can install in some Operating System: MacOS, Window, ….