**Employee Maintenance System**



|  |
| --- |
| **Team Members** |
| 1. **Sumit Chavan** |
| 1. **Rannvijay Kumar** |
| 1. **Manish Jawage** |
| 1. **Kaustubh Raigaonkar** |

1. Introduction

This document outlines a case study for Sprint 2 project. The project is to develop an Employee Maintenance System as integration of all independent micro-services. This document contains the work flow of the system and gives guidelines on how to build the functionality gradually in each of the course modules.

2. Overview

This project is aimed at developing an online Employee Maintenance System (EMS)for employees and Company administrator. EMS can be used to search for Employees based on search condition, add individual employee, modify an existing employee details and display all employee details across locations within an organization. Employees can apply for leave and the leave record will be updated and can be approved/rejected by manager.

The model followed is an agile model. Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations.

The project had to go under 2 sprints. First sprint had design implementation by core java and collections implementations. The assumed data was made static by using collections to enhance and have a practical knowledge on the collection’s framework. In second sprint we had to convert our monolithic architecture to micro-service architecture. Implementation of sprint 2 is with spring rest services along with Logger implementation and h2/MySQL database, the front end of this project is designed in AngularJS. Some of DevOps tools are also used such as Jenkins and SonarQube. Herein the entire application is ready to serve as an Employee Maintenance System with all major functionalities.

Following is a list of functionalities of the system

There are two types of users who would access the system viz. Company Administrator and Employees. Each one of them would have some exclusive privileges as follows.

1. Company Administrator

A set of administrators are assigned for managing the system. An admin has been assigned a set of privileges to manage the system. An admin can perform the following functionalities:

* + Login to the system using his/her credentials.
  + Add individual employee details by accepting all the field values from end user as listed below and inject the values into database table if data are valid else display an appropriate error messages.
  + Modify Employee details.
  + Display all employee details.

1. Employees
   * Login to the system using his/her credentials.
   * Search an employee details based on any of the fields - ID, First Name, etc.
   * Apply for leave

3. User and Epic Stories

In a sense, stories and epics in agile are similar to stories and epics in film or literature. A story is one simple narrative; a series of related and interdependent stories makes up an epic. The same is true for your work management, where the completion of related stories leads to the completion of an epic. The stories tell the arc of the work completed while the epic shares a high-level view of the unifying objective.

* [**Epics**](https://www.atlassian.com/agile/project-management/epics) are large bodies of work that can be broken down into a number of smaller tasks (called stories).
* [**Stories**](https://www.atlassian.com/agile/project-management/user-stories), also called “user stories,” are short requirements or requests written from the perspective of an end user.
* Table 1: User and Epic Stories of Employee Maintenance System

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| Login | Login | Admin / User | Validate the entered username and password | Check whether the entered password and username is correct or not |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| Employee CRUD Module | Add Employee | Admin | Enter Employee details | A new User can be added to the database |
| Search Employee | Admin / Employee | Search Employee by various fields and wild card searches(elastic search) | A particular Employee can be viewed from Employee database |
| Delete Employee | Admin | Delete an Employee by Employee Id or Delete all Employees at once | Employee/s will be deleted from Employee database |
| Modify Employee | Admin | Modify Employee details | Employee details of selected Employees will be modified |
| View all the Employees | Admin / | View all the Employees from database | All the Employees can be viewed from Employee database |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| Leave Module | Apply for Leave | Employee | Apply leave to the respective manager. | A new leave application entry will be created in the database. |
| Check leave status | Employee | Check the personal leave status and personal leave history. | A particular status of leave can be checked by Employee. |
| Approve / Reject leave | Employee(Sr.) | Approve / Reject the leave if having the authority of manager | Leaves can be granted if applicable. |
|  |  |  |  |  |

4. Use Case and UML Diagrams

1. **Pre-requisites**

User must be logged in as an Admin to perform add, update, view all, search or delete Employees.

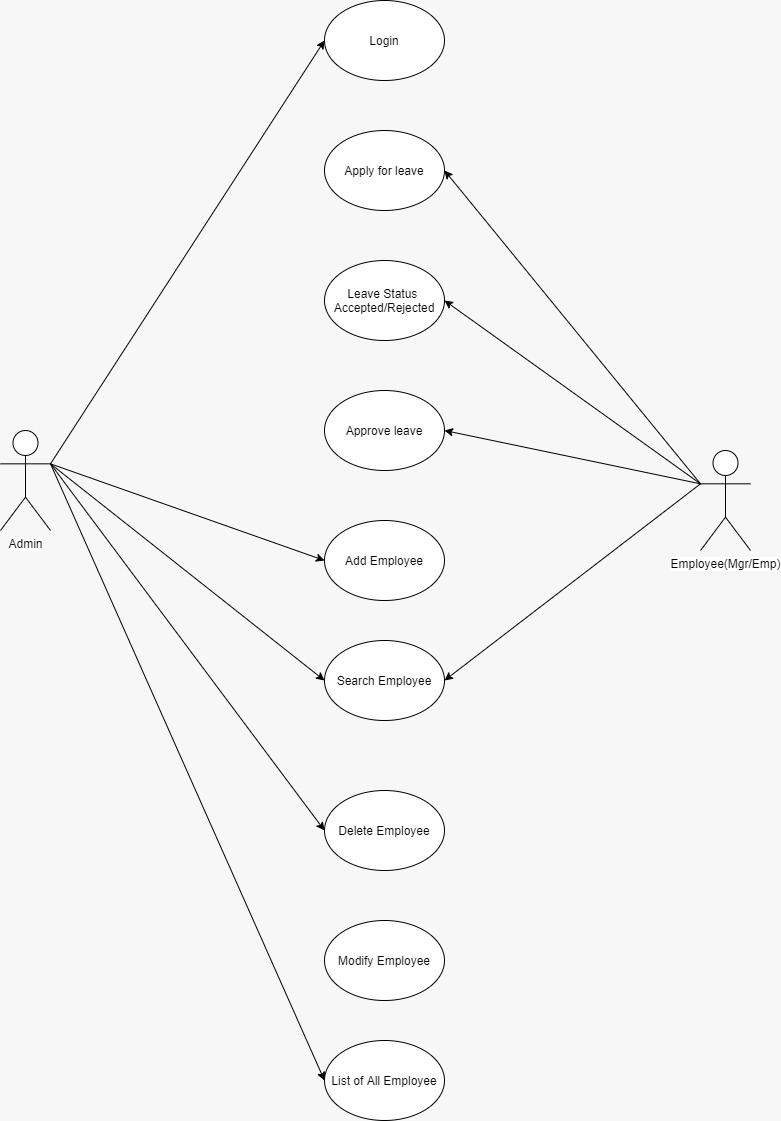
Employee must be logged in to perform the elastic search for employees and apply for leave and check the status for the already applied leaves.

1. **Non-functional requirement**

Proper form validations are maintained. To make it more user-friendly proper alert messages are shown wherever required. Any changes in database will only be performed if user gives a confirmation to those pop-up warnings.

* 1. **Employee Maintenance System Diagram:**
* Use case diagram

Following is the use case diagram of Employee Maintenance System.

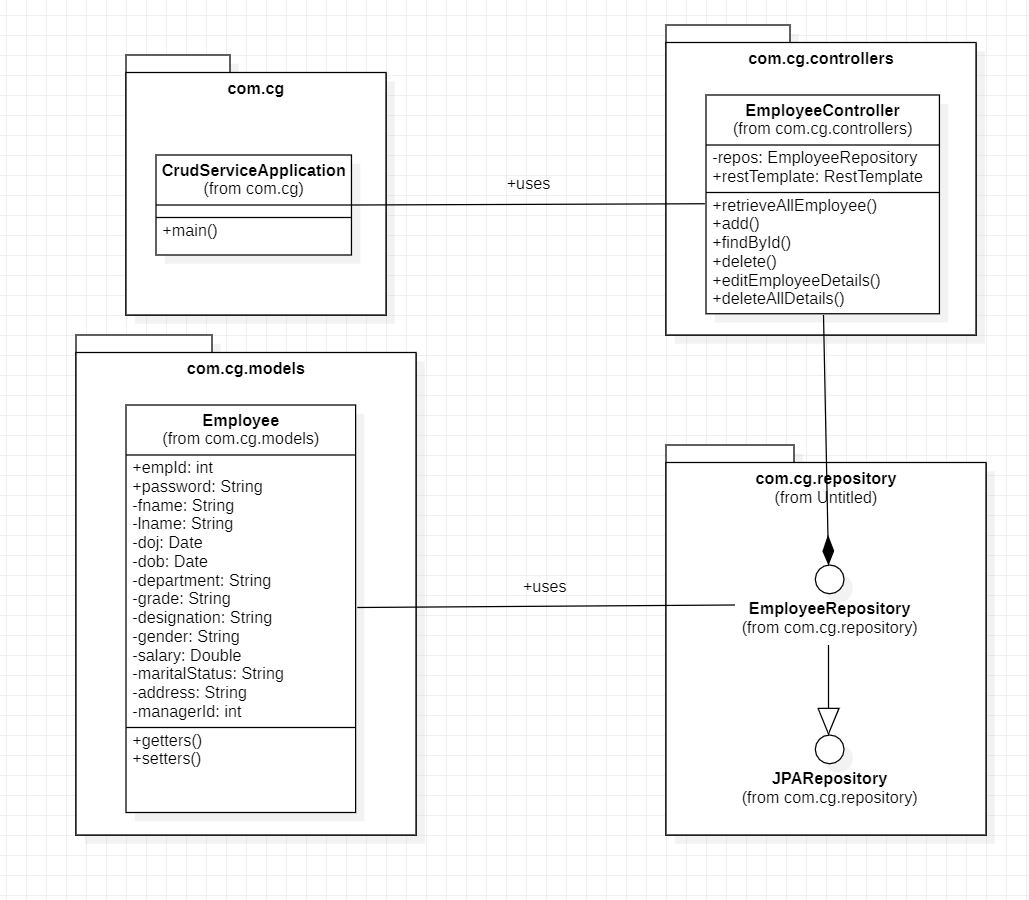


**Figure 4.1 Use Case Diagram for Employee Maintenance System**

**4.4 Employee CRUD Module:**

Class Diagram:

Following is the class diagram of Employee CRUD Module:



**Figure 4.4 Class Diagram for Employee CRUD Module**

* **Employee CRUD Module:**

This micro-service is used to perform all the CRUD operations related to the Employees which will be managed by the Admin. The respective operations will store the corresponding data in the Employee database.

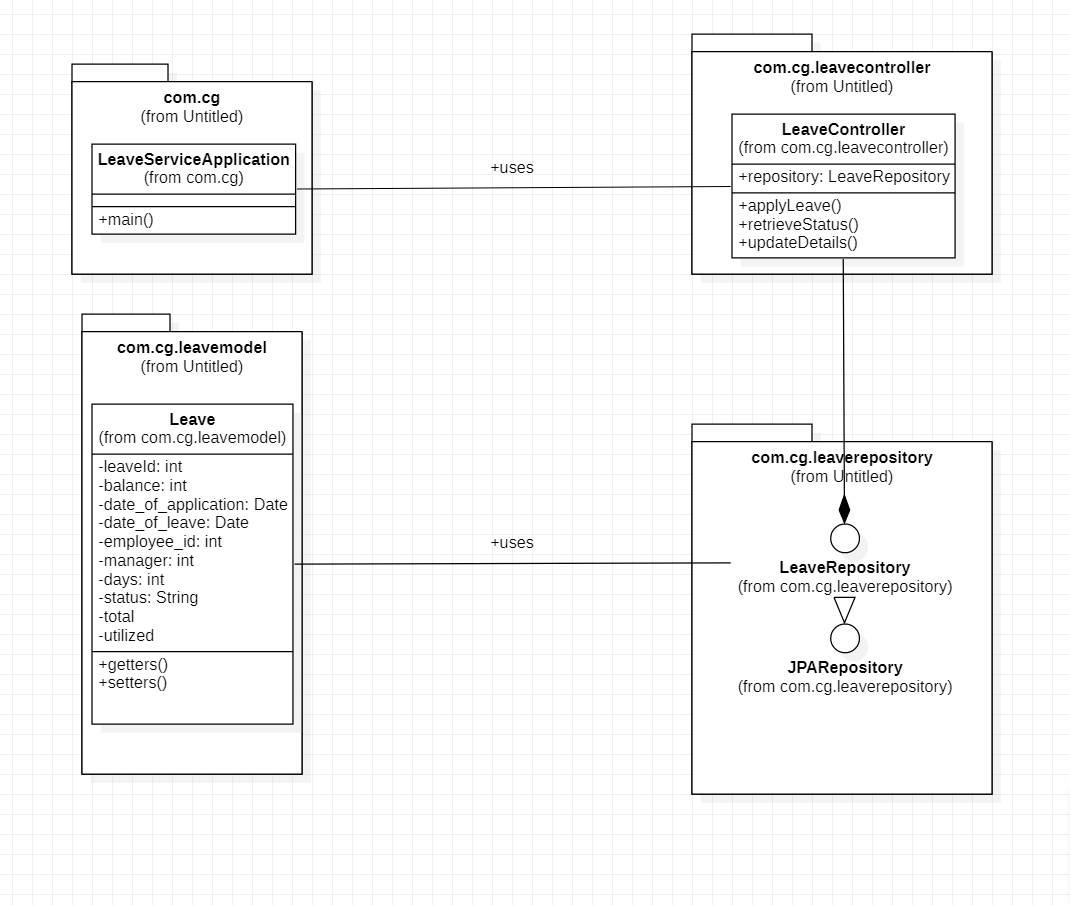
Following is the list of all the functionalities in this module:

1. **Add Employee:** This functionality has been designed to add an Employee in the database. This can be only accessed by Admin. The form should be designed with proper validation with all fields required.
2. **Delete Employee:** Admin needs to click a button to delete a User from the database. A confirmation warning will pop up. On clicking on the confirm message the Employee will be deleted.
3. **Update Employee:** It has been observed that during manual entry for an Employee detail any individual can often make mistakes while entering the data. Hence there should always be a scope to update the data which has been once entered. Thus, it is our admin’s requirement to introduce an update option. Hence, we have implemented a user-friendly update option for our Employee Maintenance system. Just on clicking an update button, the update form will appear. Either the Employee or the Admin can update the data.
4. **Search Employee:** This functionality has been designed to search a particular User on the basis of User Id specified. The required User will be fetched from the User database.
5. **View all Employees:** This functionality has been designed so the Admin will be able to view all the Users stored in the database.

**4.5 Leave Module**

* Class diagram

Following is the class diagram of Leave Module:



**Figure 4.6 Class Diagram for Leave Module**

* **Leave Module:**

This micro-service is used by Employees to apply for leaves and to check the status of already applied leaves. The Employee with the role of manager can approve / reject the leave.

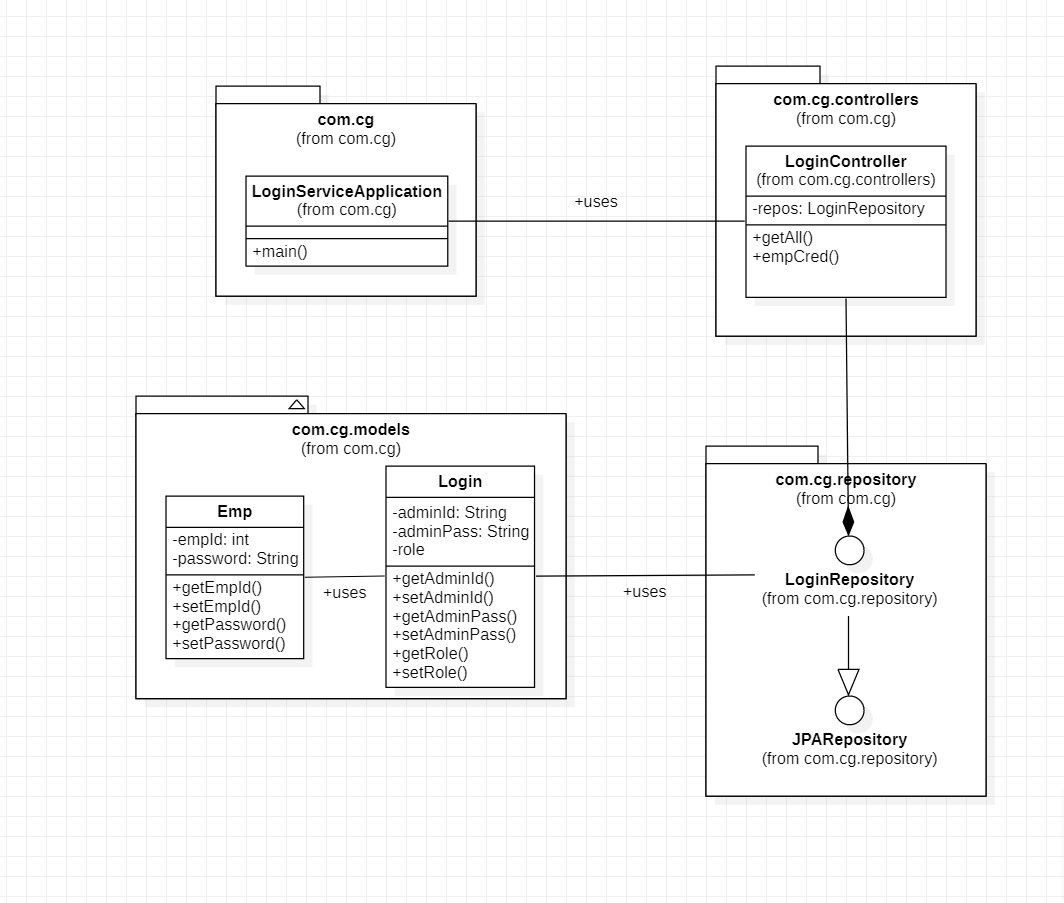
Following is the list of all the functionalities in this module:

* 1. **Apply for Leave:** This functionality has been designed to allow the Employee to apply for leaves.
  2. **Check Leave Status:** The Employee can check the status of his already applied leaves.
  3. **Approve / Reject Leave:** The Employee with the role of manager can approve / reject the leave.

**4.7 Login Module:**

* Class diagram

Following is the class case diagram of Login module.



**Figure 4.8 Use Case Diagram for Login Module**

* **Login Module:** This micro-service is used to allow either the Employee or the Admin to log into the system using appropriate credentials.

Following is the list of all the functionalities in this module:

1. **Login:** The Employee and the Admin can log in into the system by entering the respective Username and Password
2. System Requirements

Below is a list of the minimum Hardware and Software requirements to access Employee Maintenance System.

**Operating System:**

* Windows 7 and above.
* Mac OSX 10.8, 10.9, 10.10 or 10.11
* Android 3 and onwards.

**Hardware:**

* Processor (CPU) with 2 gigahertz (GHz) frequency or above
* A minimum of 4 GB of RAM
* Monitor Resolution 1024 X 768 or higher (For better view)
* A minimum of 5 GB of available space on the hard disk
* Internet Connection Broadband (high-speed) Internet connection with a speed of 2 Mbps or higher
* Keyboard and a Mouse or some other compatible pointing device

**Browsers:**

* Chrome\* 58+
* Microsoft Edge\* 20+
* Mozilla Firefox 40+
* Internet Explorer 11+ (Windows only)

*\**Google Chrome version 42+ and Microsoft Edge do not support NPAPI-type plug-ins, including Java plug-ins and many media browser plug-in.

*Users using unsupported browsers may experience issues submitting forms, placing orders, purchasing, updating details and transaction management threads.*