# Low Level Design for Employee Manager

# General Information

## Purpose and Scope

The document includes a solution on implementation of Employee Manager Microservice to address CRUD operations and holds employee specific data.

The solution is described in such detail that is possible to understand the business value, functionality and make high level effort estimation.

Any part of the proposed design may subject to change implementation.

## Input

This document is prepared based on the HLD referenced and attached below.



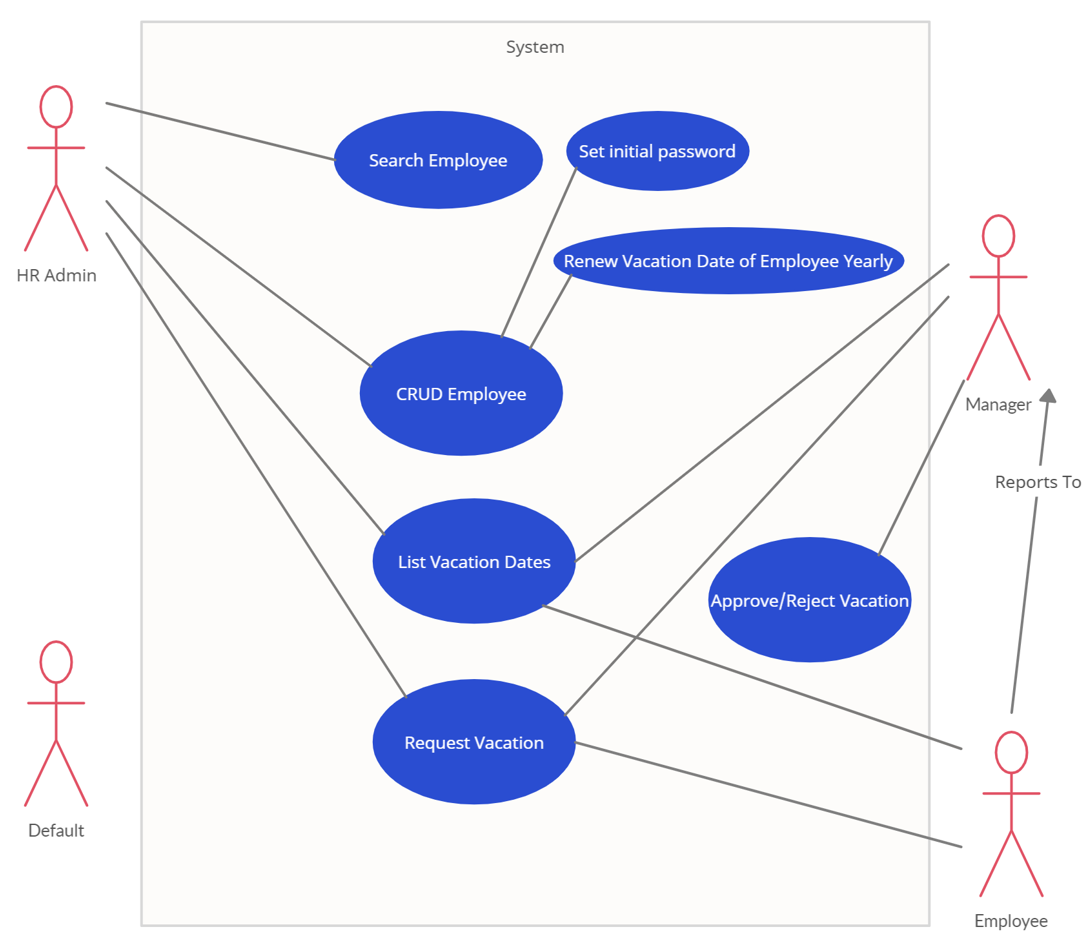
## Terminology

|  |  |  |
| --- | --- | --- |
| Term/Concept | Abbreviation | Definition |
|  |  |  |

## Revision Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Signature** | **Comments** |
| V1 | 11.12.2022 | IAKTAS | Initial Document has been created |
|  |  |  |  |
|  |  |  |  |

## Use Case Diagram



## Patent Aspects

The solution does not include any patentable part.

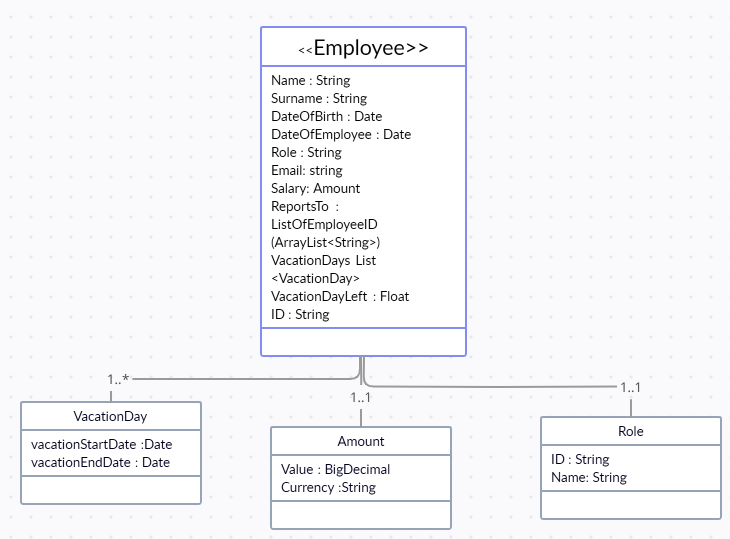
# Technical Solution

This solution proposes the design of Employee Manager Microservice which exposes basic APIs (CRUD), list vacation date API and holds employee related data.

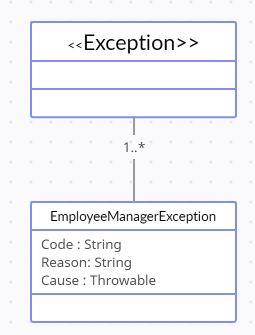
The base API path should be **/employeeManager/v1/employee**

Employee Manager is responsible for create, update, patch, delete employee and write employee info in Mongo DB.

Domain Model contains Employee and Exception Model. A view of the domain model can be seen in the following logical View – Domain Model Diagram.



**Figure 1 Employee Manager Logical View, Domain Model**



**Figure 2 Employee Manager Logical View, Domain Model – Exception**

HR Admin Front-End is going to use Keycloak to store username, password and group/role definitions. When user log-in to system the username and password validation will be done by Keycloak.

In Every CRUD operation request in Employee Manager microservice, validation of username, password from Keycloak adapter will be done first. Keycloak adapter should produce access token if user is valid for the operation else returns unauthorized response in response message. In request header username and password will be sent to request get token.

**Keycloak stub will be implemented.**

Not to wait for Front End Keycloak configuration, stub should be created.

Stub will get keycloak user which is created in create an employee scenario.

Stub will request for token to validate system user to perform specific action.

“client\_id” will store action which system user will perform.

“group” is user group which a system user is assigned to.

Stub will validate weather group – client-id relation is correct or not by comparing data in configMap. If not validate, stub will return 403-Forbidden Response.

GET employee-manager-api/v1/realm\_hr\_stub/users/{{user\_name}}

GET employee-manager-api/v1/realm\_hr\_stub/protocol/openid-connect/token

Sample Keycloak User Json:

{

    "username": "isabah",

    "firstName": "isa",

    "lastName": "sabah",

    "groups": [

        "EMPLOYEE"

    ],

    "credentials": [

        {

            "id": null,

            "createdDate": "2022-12-13T18:17:44.486Z",

            "type": "password",

            "value": "pwd\_hr\_123",

            "temporary": true

        }

    ],

    "email": "isabah@xx.com",

    "enabled": true,

    "client\_id": null,

    "lastAction": null,

    "lastActionDate": null

}

**Sample Token Response:**

{

    "access\_token": "eyJhbGciOiJSUzI1NiIsINVSHGhepnDu13SwRBL-v-y-04\_6e6IJbMzreZwPI-epwdVPQe-ENhpvms2WdGM\_DmgMLZ8YQFS4LDl9R7ZHT8AgXe-WCFV6OFkA7zvdeFwQ4kVVZE0HlNgHgoi4DrgMfwwz\_ku1yJNJP3ztTY1nEqmA",

    "refresh\_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRRnB5YlloMGVEektIdlhOb3JvaFUxdlRvWVdjdP3vbfvk7O0zvppK9N4-oaUqZSr0smHv5LkuLDQYdPuxA",

    "token\_type": "bearer",

    "session\_state": "bb1c586a-e880-4b96-ac16-30e42c0f46dc",

    "expires\_in": 300,

    "refresh\_expires\_in": 1800,

    "not\_before\_policy": null

}

**ConfigMap Cache will store user goup and its actions list.**

Below config map will be cached in microservice:

{  
 "configManagerMaps" : [  
 {  
 "groupName": "HR\_ADMIN",  
 "actionList": [  
 "manageEmployee",  
 "checkVacationList",  
 "requestVacation",  
 "getEmployee"  
 ]  
 },  
 {  
 "groupName": "MANAGER",  
 "actionList": [  
 "manageVacation",  
 "requestVacation",  
 "checkVacationList",  
 "getEmployee"  
 ]  
 },  
 {  
 "groupName": "EMPLOYEE",  
 "actionList": [  
 "checkVacationList",  
 "requestVacation",  
 "getEmployee"  
 ]  
 }  
 ]  
  
 }

**Create an employee:**

Initial password will be assigned by the system. With that username and password user will be created in Keycloak and employee data except password will be stored in MongoDB.

Employee initial remaining vacation day will be set as 18 days.

vacationRenewalDate should be set as todaysDate for the first creation.

Stub Keycloak URL will be created.

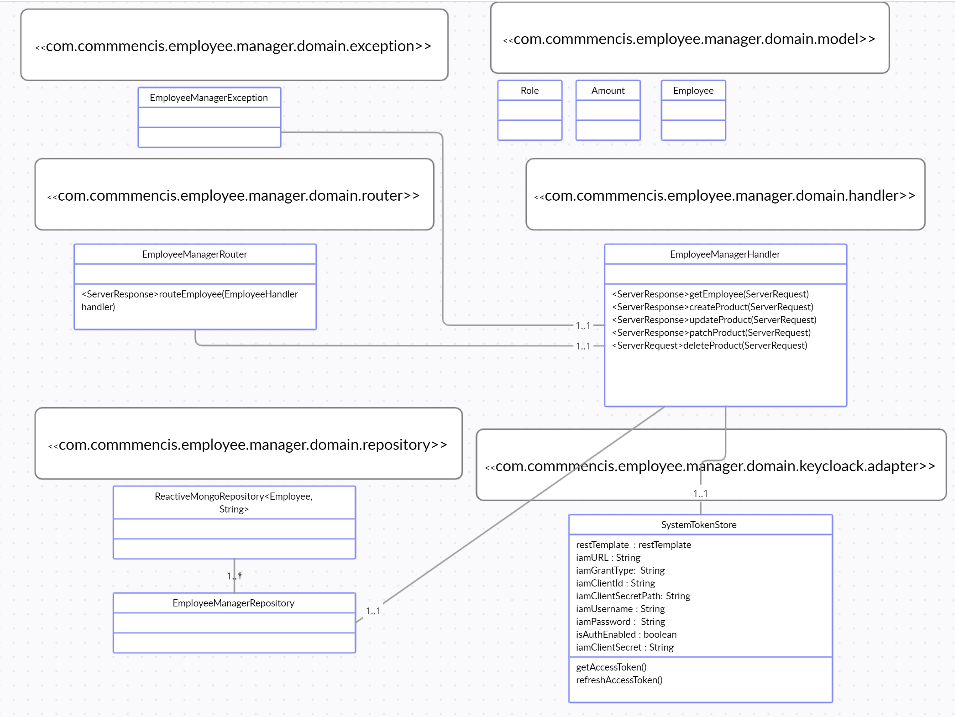
Created user can be retrieved by using below API.

employee-manager-api/v1/realm\_hr\_stub/users/{{user\_name}}

**Update/Patch and Employee:** There will be no request sent to Keycloak for change, only Employee data in repository will be updated. If there is no such Employee, EmployeeManagerException will be thrown.

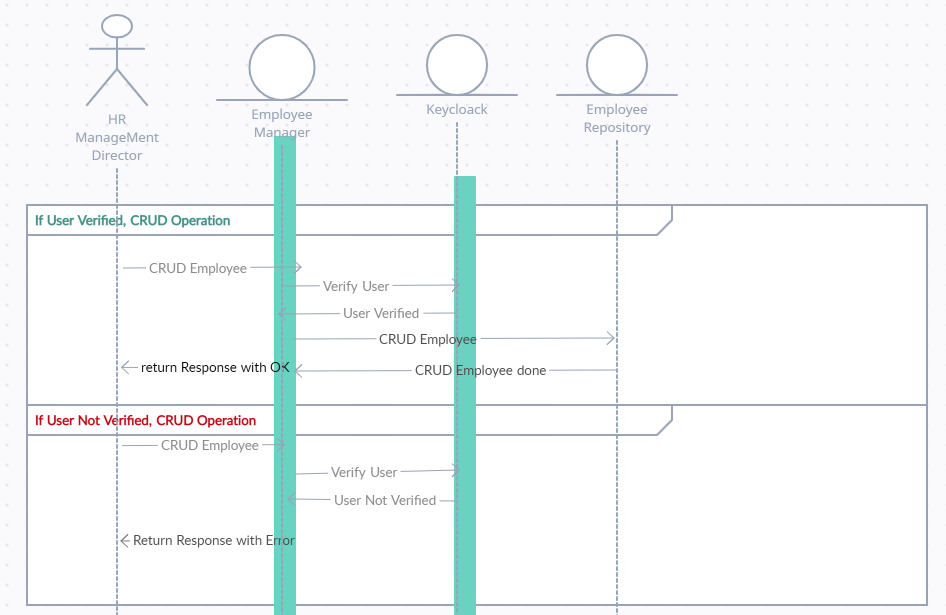
**Delete Employee:** There will be no delete request sent to Keycloak. Only Employee data in repository will be deleted.

The basic development view package diagram including dependencies in between classes can be seen in the following figure:

**Figure 3 Employee Manager Basic Development View – Package Diagram**

In Handler, employee related services can be called if required.

A simplified main flow of user story implementation can be seen in the following logical sequence diagram.



# Employee Json File should be attached

Sample Json representation of employee creation request is as below:

{

    "username": "iaktas",

    "name": "irem",

    "surname": "aktas",

    "role": {

        "id": "HR\_ADMIN",

        "value": "HR\_ADMIN"

    },

    "reportsTo": [

        "xx"

    ],

    "salary": {

        "value": 3,

        "unit": "TL"

    },

    "dateOfBirth": "1988-10-11T03:06:00.000Z",

    "dateOfEmployment": "2023-10-11T03:06:00.000Z",

    "email": "iaktas@xx.com",

    "groups" : [

        "HR\_ADMIN"

    ]

}

Sample Json representation of employee creation response is as below:

{

    "id": "63974bbda1b60c153e0a73cf",

    "username": "iaktas",

    "name": "irem",

    "surname": "aktas",

    "role": {

        "id": "HR\_ADMIN",

        "value": "HR\_ADMIN"

    },

    "reportsTo": [

        "xx"

    ],

    "vacationDayLeft": 18.0,

    "salary": {

        "unit": "TL",

        "value": 3.0

    },

    "dateOfBirth": "1988-10-11T03:06:00.000Z",

    "dateOfEmployment": "2023-10-11T03:06:00.000Z",

    "vacationRenewalDate": "2022-12-12T15:41:49.597Z",

    "vacations": null,

    "email": "iaktas@xx.com",

    "groups": [

        "HR\_ADMIN"

    ]

}

# Other Functionalities to Be Added

If team capacity is enough for this sprint we need to add below functionalities in the microservice.

Micrometer and Prometheus should be implemented for performance logging.

Resilience to make microservice fault tolerant and Spring retry can be added to recall operation when request gets unauthorized or unreachable operations.

Sleuth should be used for microservice logging.

Test Coverage should not be under %80.

Docker File should be prepared.

# Assumptions

Keycloak service will be added to the environment setup. In this sprint It will be only used for registering user and password and assigning to group. Role definitions, assigning functionalities to role and Assigning roles to user will be implemented in further sprint planning which will be added in project backlog.

Roles and its functionalities will be kept in Config Manager Service for this sprint.

# References

<https://www.keycloak.org/docs-api/12.0/rest-api/#_users_resource>