**EMPLOYEE MAINTENANCE SYSTEM**

Table of Contents

1. Introduction ------------------------------------------------------------------------------------
2. Overview ----------------------------------------------------------------------------------------
3. Epic & Stories ----------------------------------------------------------------------------------
4. System Architecture -------------------------------------------------------------------------
5. Class Diagram ----------------------------------------------------------------------------------
6. Use Cases ---------------------------------------------------------------------------------------
   * 1. Login service
     2. Employee management service
     3. Leave Service
7. Database Tables
8. System Requirements

1. Introduction

This project is aimed at developing an Online Employee Maintenance System. This is an Intranet based application that can be accessed throughout the organization and this is a web-based application that can be accessed over the web. This system can be used to search for Employees based on search conditions, add individual employees, 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 the manager. This is an integrated system that contains both the user (Employee) component and the Admin component.

2. Overview

Following is a list of functionalities of the system. Wherever, the description of functionality is not adequate, you can make appropriate assumptions and proceed.

There are two categories of people who would access the system viz. employee & Admin. Each one of them would have some exclusive privileges (for e.g. Employees can just search an employee details based on search condition.)

1. Admin should be able to
   * 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 message.
     + Employee ID has to be a 6-digit number (with no decimals) and cannot start with a ZERO (Cannot be empty)
     + First and Last name should be alphabets, with initial uppercase only (Cannot be empty)
     + Date of joining and Date of birth need to undergo regular date validation (DD-MMM-YY) - (Cannot be empty)
     + The date of joining should be > Date of birth and the employee’s age has to be >=18 and <= 58 as an eligibility criterion for employment
     + Department name values has to be populated from database into a dropdown list (Cannot be empty)
     + Grade has to be one of the following values: (Cannot be empty)
       1. M1
       2. M2
       3. M3
       4. M4
       5. M5
       6. M6
       7. M7

This should be in a dropdown list

* + - Designation can be alphabets limited to 50 characters: (Can be empty)
    - Gender has to be one of the following values: (Cannot be empty)
      1. Male
      2. Female
    - Basic should be numeric only and the basic has to be validated against a salary band for each grade
    - Marital Status can be one of the following values: (Cannot be empty)
      1. Single
      2. Married
      3. Divorced
      4. Separated
      5. Widowed
    - Home Address (Can be alphanumeric and optional)
    - Personal Contact Number (Can be alphanumeric and optional)
    - The employee ID of manager should be stored against each employee. Admin can assign or modify the Manager for each employee.
  + Modify Employee details:
    - Based on existing employee id, display the following fields First Name, Last Name, Department, Grade, Designation, Basic Salary, Marital Status, Home Address, Personal Contact Number in an editable mode.(validation should be taken care for the change in basic salary if there is change in the grade)
  + Display all employee details:
    - Employee details either display in single page or easy to use provisions (like previous, next, first, last) should be provided to navigate between records. Display should be limited to 10 records per page
    - Display details in the below given tabular format

**<<First>> <<Next>> <<Previous>> <<Last>>**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| <<ID>> | <First Name> | <Last Name> | <Department> | <Grade> | <Designation> |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. Employees/User should be able to
   * Login to the system using his/her credentials
   * Search an employee details based on any of the fields - ID, First Name, Last Name, department, Grade, Marital Status. The following fields are provided to enable the user specify the search conditions:
     1. ID (Can provide wild card search)
     2. "First Name" (Can provide wild card search)
     3. "Last Name" (Can provide wild card search)
     4. Wild card search (? / \*)
        + ? for a character
        + \* for n characters
     5. Department (Can choose one or more department at a time – Multiple choice)
     6. Grade (Multiple choice)
     7. Marital Status (Multiple choice)

The results will be filtered and displayed following the same display rules as the display all employee details requirement.

* + Apply for leave:

1. An Employee is eligible for 12 casual leaves per year.
2. This can be availed at any time.
3. When an Employee is logged in, he or she should be able to apply for leave.
4. The status of leave should be 'applied'.
5. Manager of the employee can approve or reject the leave and appropriately status should be updated as 'approved' or 'rejected'.
6. Once approved, the balance should be updated in db.
7. If leave is not approved, it gets automatically approved after 3 days

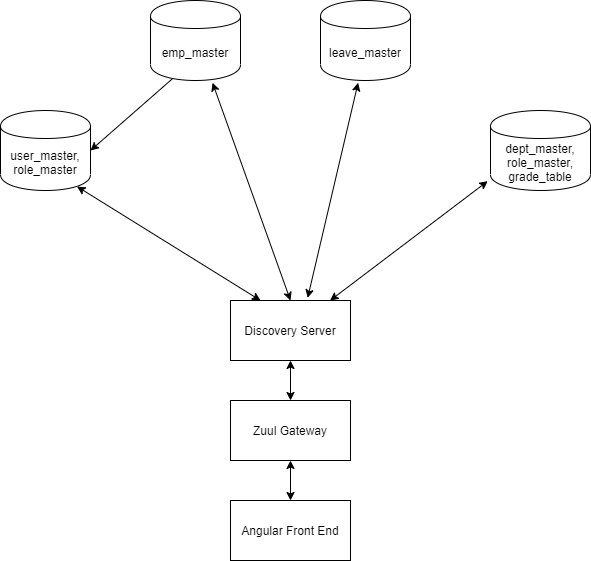
3. Epic & Stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| Login | Registration | User | Sign up as new user | A new user will be registered |
| Login | Admin, User, Manager | Validate the entered username and password | Check whether the entered password and username is correct or not |
| Change Password | User, Admin, Manager | Change existing password | A new password will be created for a particular Account |
| Change Role | Admin | Promote the Employee | Employee can access additional features |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| Employee Management Module | Add Employee | Admin | Enter employee info | A new employee will be added to the system |
| Update Employee | Admin | Update Employee Details | Previous incorrect data can be modified |
| Search by Category | User, Admin, Manager | Use Filtered search | Search for an Employee becomes clean |
| Search All | User, Admin, Manager | Search for an Existing employee details | An Employee details can be searched for |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that… |
| Leave Module | Create Leave | Manager, Admin, User | Request for a new Leave | A new request for leave will be generated |
| Approve Leave | Manager | Approve a requested leave | The leave request is approved |
| Reject Leave | Manager | Reject a requested leave | The leave request is rejected |
| Show my Leaves | Manager, Admin, User | Show my Leave History | A summary of users leave can be provided |
| Show Sub Employee Leaves | Manager | To see which Sub Employee has requested for leave | So, leaves can be updated from dropdown |
|  | Scheduled Approval | System | To automatically Approve Leave | So leaves can be approved if manager doesn’t take any action |

4. System Architecture

****

5. Class Diagram

****

6. Use Cases

6.1 Login Service

Overview

To carry out basic login and log out operation. A basic wall of security to access the confidential data. All Employees, Manager and Admin will use a common login system

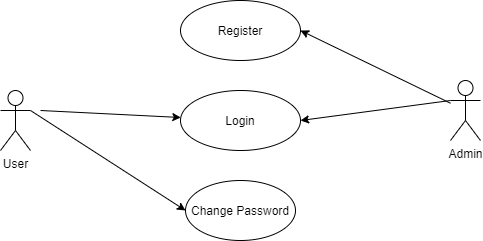
Prerequisite

A database containing authorized user credentials

Flowchart for Login Service



Use Case Diagram for Login Service

****

* 1. Register

A new user can be registered by the admin and he will be assigned a username and password.

* 1. Login

The user entered username and password is validated.

* 1. Change password

The existing password can be changed, and a new password will be created.

6.2 Employee Management Service

Overview

To Carry employee management operation such add, update, create, search employee using various filters

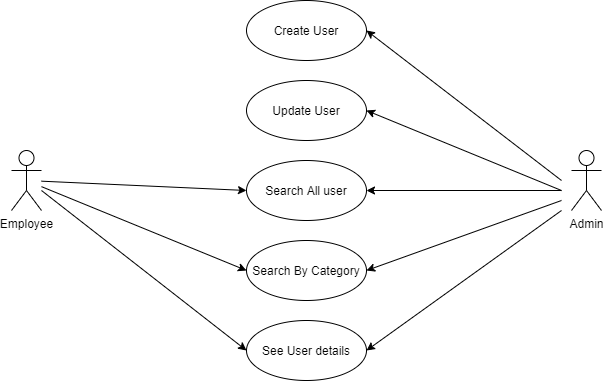
Prerequisite

An employee database containing employee details and mapping to other tables containing data related to employee

Flowchart for Employee Management Service

****

Use Case Diagram for Employee Management Service

****

1. Create User

A new user can be added to the system by the admin by entering user information.

1. Update User

An existing user can be modified by the admin if the data entered is incorrect.

1. Search All User

All users can be searched by both employee as well as admin.

1. Search by Category

Users can be searched by filtering on the basis of category.

1. See User details

User details can be viewed by the employee and the admin.

6.3 Leave Service

Overview

A microservice to manage Employee leaves. Leaves can be requested by employees. They can be approved/ rejected by a manager

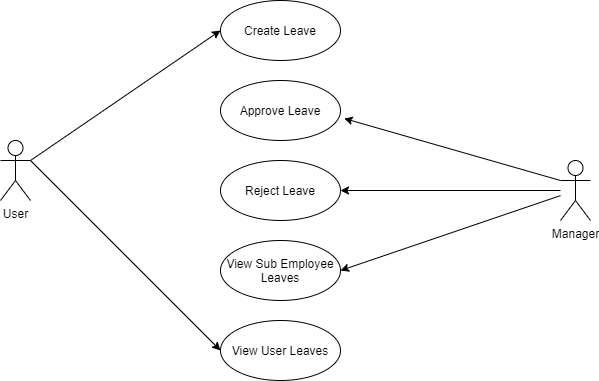
Prerequisite

A Leave table containing leave related data and mapped by employees and managers

Flowchart for Employee Management Service



Use Case Diagram for Employee Management Service

****

1. Create Leave

The user can request for a new leave and a new leave request will be generated.

1. Approve Leave

The manager can approve leaves of the users registered to him.

1. Reject Leave

The manager can reject leaves of the users registered to him.

1. View Sub employee Leaves

The manager can view the leaves of the users registered to him.

1. View User Leaves

The user can view the leaves requested by him and its status.

7. Database Tables

1. **User Credential table**

|  |  |  |  |
| --- | --- | --- | --- |
| Database Table | user\_master |  | Stores User Credentials |
| user\_id | NUMBER | 6 | Primary Key Id |
| username | VARCHAR | 25 | Username |
| password | VARCHAR | 25 | Password |
| Role\_id(FK) | NUMBER | 6 | Role table mapping |

1. **Roles table**

|  |  |  |  |
| --- | --- | --- | --- |
| Database Table | role\_master |  | List of Available Roles |
| role\_id | NUMBER | 1 | Primary Key |
| role\_desc | VARCHAR | 25 | Available role |

1. **User details table**

|  |  |  |  |
| --- | --- | --- | --- |
| Database table | user\_details |  | To store additional details of user |
| user\_id(FK) | NUMBER | 6 | Mapping to user table |
| user\_firstname | VARCHAR | 25 | First name |
| user\_lastname | VARCHAR | 25 | Last name |
| user\_DOB | DATE | DD-MM-YYYY | Date of birth (age 18-58) |
| user\_DOJ | DATE | DD-MM-YYYY | Date of joining (age 18-58) |
| user\_email | VARCHAR | 20 | Email |
| user\_phoneno | VARCHAR | 10 | Phone No |
| user\_managerid - FK (user\_id) | NUMBER | 6 | Mapping with user id with manager role |
| dept\_id(FK) | NUMBER | 6 | Department Mapping |
| user\_marital\_status | VARCHAR | 25 | Marital Status (Single, Married, Widowed) |
| user\_gender | VARCHAR | 1 | Gender (M, F) |
| user\_designation | VARCHAR | 25 | Designation |

1. **Address Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Database table | address |  | To store addresses |
| user\_id (FK) | NUMBER | 6 | User foreign Key |
| city | VARCHAR | 20 | City |
| state | VARCHAR | 20 | State |
| area | VARCHAR | 20 | Area/Street |
| pincode | VARCHAR | 6 | Pin code |

1. **Grade table**

|  |  |  |  |
| --- | --- | --- | --- |
| Database table | grade\_table |  | Salary Validation |
| grade\_code | VARCHAR(PK) | 2 | Grade (M1-M7) |
| grade\_min\_sal | NUMBER | 6 | Minimum Salary |
| grade\_max\_sal | NUMBER | 6 | Max salary |
| grade\_desc | VARCHAR | 50 | Description |

1. **Leave History**

|  |  |  |  |
| --- | --- | --- | --- |
| Database table | leave\_history |  | To track employees leave History |
| leave\_id | NUMBER | 6 | Primary Key |
| FK(user\_id) | NUMBER | 6 | Mapping with User |
| leave\_balance | NUMBER | 2 | Max 13 leaves possible |
| leave\_start\_date | DATE | DD-MM-YYYY | Start Date |
| leave\_end\_date | DATE | DD-MM-YYYY | End date |
| leave\_no\_of\_days\_applied | NUMBER | 2 | End-Start date |
| leave\_approval\_status | VARCHAR | 20 | Status (Approved, Pending, Rejected) |

8. System Requirements

Below is a list of the minimum Hardware and Software requirements

**Operating System:**

* Windows 7 and above.
* Mac OSX 10.8, 10.9, 10.10 or 10.11
* Any OS that supports Chrome browser

**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)
* Safari 6+ (MacOS only)*Users using unsupported browsers may experience issues.*

**Browser Configuration:**

Your browser must be configured as follows:

* JavaScript must be enabled
* CORS must be configured properly
* Cookies must be enabled.
* Pop-up windows must be enabled.

**Software:**

* Java — to view and interact with all available blackboard applications.
* Spring Tool Suite — STS workbench was used to run JDK (write, compile and run the code).
* Visual Studio Code — for writing codes for frontend using Angular, VS Code was used as a workbench.
* Apache Tomcat — it was used as a server for hosting the website.

**DevOps Tools:**

* GitHub – Collaboration for Project development
* Docker – Project is deployed in the form of docker container

**Framework:**

* Swagger2 Docs
* Spring Sleuth for Logging
* Eureka Discovery Server
* Netflix Zuul Gateway