**HRMS API**

**Scope**: To develop HRMS (Human Resource Management System API) which helps us to automate the end-to-end process of the Enique HR Team starting from onboarding to resignation of an employee.

Since this is the initial draft/ basic idea of the project, scope will stand the same till the end but the approach/design to achieve the scope might differ. Any idea which seems relevant and useful will be incorporated as part of the design.

**RAML Design:**

RAML needs to be designed for all the APIs by following all the design standards.

Proper documentation should be done for all the API endpoints

Child ramls, resource Types, libraries concept must be used.

Publish the APIs to exchange.

**Database Design**:

The developer who is going to develop a particular API needs to design the database tables as well by adhering to all the DB constraint principles.

General fields for creating the table are mentioned below and also refer zoho portal and anypoint exchange for HR APIs to find additional fields. Insert 10 meaningful records to each and every table. Add all the additional fields included in the table below.

**Database (EniqueHrms)**

|  |  |
| --- | --- |
| **Table Name** | **Fields** |
| Applicant | Id, applicantName,DateOfApplication,JobID,ApplicantType,ApplicationStatus, OnlineTestStatus,OnlineTestEvaluation, Score, LocationPreference1,LocationPreference2,LocationPreference3 ,ContactNumber,EmailID, CurrentLocation, Qualification, CollegeName, PassOutYear,ProgrammingSkills,Experience, Designation, ExpInCC,NoticePeriod,OpenForRelocation,CurrentSalary,ExpectedSalary, statusOfApplication, InterviewPanelist,Comments |
| Fresher |  |
| Employee | Id, employeeId, employeeName, dateOfJoining, nickName, gender, country, state, language, email, mobileNumber, deptId, managerId |
| Manager | Id, managerId, deptId |
| Department | Id, deptId, deptName |
| Personal Details | Id, employeeId, name, education, maritalStatus, aadharId, panNumber, fatherName, fatherMobileNumber, email |
| Attendance | Id, employeeId, checkInTime, checkOutTime, date, totalHoursLogged |
| Compensation | Id, employeeId, salary, bonus, hike, variablePay, year, quarter |
| Leaves | Id, employeeId, leaveBalance, leaveType, noOfLeaves |
| Expenses | Id, employeeId, expenseType, Amount, approved, status |
| Office Readiness | Id, employeeId, workMode, workLocation |
| Project | Id, projectName, employeeId, projectDetails, clientName, technology |
| Announcements | Id, announcement Name, announcementDate, |
| Resignation Details | Id, employeeId, resignationDate, lastWorkingDate |
| Former Employee | Id, employeeId, yearsWorked |

Creation of tables will be done at once by one or two people after the raml is created and approved with all the fields. We would be creating a strategy to make local database in sync for all the developers.

Or

We will be using salesforce for creation of objects (tables are called as objects in salesforce terms), so that everyone can access the same data with single point of connection. This process also makes our deployment to Cloud Hub easy by connecting to salesforce objects.

**Project Implementation:**

Create project enique-hrms-sapi, enique-hrms-papi and enique-hrms-exp. The projects must be pushed to three different repositories on git by following the branching strategies. Code needs to be deployed to Cloud Hub by creating the CI/CD pipelines using Git Hub action workflows.

Reusability-

* The APIs that are built for this integration can be reused in future by another client or consumer with similar requirements.
* Create common request connections and use flow references to refer them

Security-

* All MuleSoft APIs are enabled with Client ID Enforcement policy using option “HTTP Basic Authentication Header as the credentials origin” to enable security on APIs.
* The Client ID Enforcement Policy restricts access to a protected resource, by only allowing HTTP requests if credentials of client applications already registered to (contracted with) the API being targeted are provided in each request. The set of Client ID and Client Secret are also known as credentials. The policy ensures that the client credentials sent on each request have been approved to consume the API.
* When a client application is registered in Anypoint Platform, a pair of credentials consisting of a client ID and client secret is generated. When the client application requests access to an API, a contract is created between the application and that API. An API that is protected with a Client ID Enforcement policy is accessible only to applications that have an approved contract.

Readability-

* Flows to be created should follow proper naming convention across the project
* Variables, transform messages, flow references etc. should be properly named

Unit Testing-

* Each and every API must be thoroughly tested
* Create necessary postman collections

The first step of our project is to identify and develop as many system APIs as possible. Endpoint starts with api/v1/system/{dbtablename}

**Below are the system APIs identified:**

1. Store the details of the candidate whoever applies for the job (DB Table Name: Applicant)
2. Perform all the crud (create, read, update, delete) operations on Applicant
3. Send an email invitation with proper template to a candidate interview
4. Store the details of onboarded employee (DB Table Name: Employee)
5. Perform all the crud operations on employee
6. Create API to store the details of the manager (DB Table Name: Manager)
7. Perform all the crud operations on Manager
8. Create API to store the details of the department (DB Table Name: Department)
9. Perform all the crud operations on Manager
10. Create API to store the personal details of the employee (DB Table Name: Personal Details)
11. Perform all the crud operations on Personal Details
12. Create API to store the details of the Attendance (DB Table Name: Attendance)
13. Perform all the crud operations on Attendance
14. Create API to store the details of the compensation (DB Table Name: Compensation)
15. Perform all the crud operations on compensation
16. Create API to store the details of the leaves (DB Table Name: Leaves)
17. Perform all the crud operations on leaves
18. Create API to store the details of the Expenses (DB Table Name: Expenses)
19. Perform all the crud operations on Expenses
20. Create API to store the details of the Office Readiness (DB Table Name: Office Readiness)
21. Perform all the crud operations on Office Readiness
22. Create API to crud the details of the Project (DB Table Name: Project)
23. Perform all the crud operations on Project
24. Create API to send and store announcements (DB Table Name: Announcements)
25. Create API to store the resignation details (DB Table Name: Resignation Details)
26. Create API to store the details of the Former Employee (DB Table Name: Former Employee)
27. Perform all the crud operations on Former Employee
28. Create API to read the details of policies.
29. Create API to send an email with the attachment whenever a policy is updated

After all the system APIs are developed, tested and deployed to Cloud Hub our second step is to develop Process APIs. Endpoint starts with api/v1/process/{processapiname}

**Below are the process APIs identified:**

1. Send email notifications to HR at 8 Am on birthdays, work, and marriage anniversary dates of the employees. In return, HR will feed a message to other API which will send wishes to the employee by keeping organization members in cc.
2. After the last working day of the employee, all the employee records must be removed from the employee, person details, department, etc. Those records need to be inserted into the former employee table.
3. Leave balance calculation and updation. When an employee applies for a leave, we need to update which leave was availed then reduce the count of that leave and finally update the total leaves available for the employee.
4. When an applicant got selected his record needs to be removed from the Applicant table and specific fields data needs to be used in creating the employee record.
5. Employee has to check-in and check-out on all the working days,
6. Optional holidays
7. Read the input of applicant (Java Freshers) from the google sheet, perform nullable validation and insert the record to Applicant object
8. Read the input of applicant (Java Experienced) from the google sheet, perform nullable validation and insert the record to Applicant object
9. Read the input of applicant (Salesforce) from the google sheet, perform nullable validation and insert the record to Applicant object
10. Read the input of applicant (Integrations) from the google sheet, perform nullable validation and insert the record to Applicant object

**End Goal:**

This project is initiated to learn more things, stay productive, and gain real-time experience in this rapidly growing world. Our final goal is by the end of this project, the organization should be benefited from our HRMS API.

All the best ...