|  |
| --- |
|  |
|  |
| **HRMS- Employee Api analysis V1** |
|  |



**Process Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ver. No** | **Date** | **Author** | **Approved by** | **Description** |
| 1 | 26-Jun-2020 | Manasa Kothapalli |  | Writing technical documentation on the Employee API micro services analysis. |

Table of Contents

[1 PURPOSE 4](#_Toc45792373)

[2 SCOPE 4](#_Toc45792374)

[3 ACRONYMS AND DEFINITIONS 4](#_Toc45792375)

[4. Employee 4](#_Toc45792376)

[4.1. Prospective Associate Screen: 4](#_Toc45792377)

[4.1.1. ProspectiveAssociate/GetPADetails: 4](#_Toc45792378)

[4.2. Add Prospective Associate Screen: 5](#_Toc45792379)

[4.2.1. ProspectiveAssociate/GetEmpTypes: 5](#_Toc45792380)

[4.2.2. ProspectiveAssociate/GetHRAAdvisors: 6](#_Toc45792381)

[4.2.3. PracticeArea/GetPracticeAreas: 7](#_Toc45792382)

[4.2.4. UserDepartment/GetDepartmentDetails: 8](#_Toc45792383)

[4.2.5. MasterData/GetManagersandCompetencyLeads: 9](#_Toc45792384)

[4.2.6. ProspectiveAssociate/AddPADetails: 10](#_Toc45792385)

[4.2.7. MasterData/GetDesignations: 11](#_Toc45792386)

[4.2.8. ProspectiveAssociate/GetGradeByDesignation: 12](#_Toc45792387)

[4.2.9. ProspectiveAssociate/GetPADetailsByID: 14](#_Toc45792388)

[4.2.10. ProspectiveAssociate/UpdatePADetails: 15](#_Toc45792389)

[4.3. Associate Joining Screen: 16](#_Toc45792390)

[4.3.1. ProspectiveAssociate/GetJoinedAssociates: 16](#_Toc45792391)

[4.4. Associate Information Screen: 17](#_Toc45792392)

[4.4.1. AssociatePersonalDetails/GetAssociateDetails: 17](#_Toc45792393)

[4.5. Education Tab: 17](#_Toc45792392)

[4.5.1. AssociateEducationDetail/GetEducationDetailsbyID: 17](#_Toc45792393)

[4.5.2. AssociateEducationDetail/SaveEducationDetails: 17](#_Toc45792393)

[4.5.3. Common/GetBusinessValues: 17](#_Toc45792393)

[4.6. Employment Tab: 17](#_Toc45792392)

[4.5.1. AssociateEmployment/GetEmploymentDetailsbyID: 17](#_Toc45792393)

[4.5.2. AssociateEmployment/GetProfReferenceDetailsbyID: 17](#_Toc45792393)

[4.5.3. AssociateEmployment/SaveEmploymentDetails: 17](#_Toc45792393)

[REFERENCES 18](#_Toc45792394)

# PURPOSE

The purpose of this documentation is to:

* List out the business rules of the HRMS existing application.
* List out the database and other dependencies of the existing application.
* Helps other developers in the team to understand the existing business rules.

# SCOPE

The scope of this document is to provide technical document of Employee screens.

# ACRONYMS AND DEFINITIONS

|  |  |
| --- | --- |
| **Definition** | **Description** |
|  |  |

## Employee

## Prospective Associate Screen:

## ProspectiveAssociate/GetPADetails:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: ProspectiveAssociateController.cs
    - **Method:** GetPADetails
    - This method does not have any input.
    - This Method is used for getting Prospective Associate details.
    - Linq query is used in this method for getting Prospective Associate Details. The query works as follows
      * Purpose: This Query is to fetch Prospective Associate Details.
      * Tables: ProspectiveAssociates, Departments, Designations, and PracticeArea.
      * Decision: Retain LINQ.
      * Rules:
* Fetch all active Prospective Associate details.
  + - * Return: all prospective Associate related details including department, designation and Technology details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: To get Prospective Associate details
      * Database Tables:
      * **HRMSOrganization:**
        + Department
        + Designation
        + PracticeArea
      * **HRMSEmployee:** 
        + ProspectiveAssociate
    - Cross communication: Cross communication required for fetching data Department, Designations and user Practice Area.
    - Return: Prospective Associate related details.
  + API Details:
    - MicroService: Employee
    - Controller: ProspectiveAssociateController.cs
    - Method: GetAll
    - Path: {{HostName}}/Employee/api/v1/prospectiveAssociate/GetAll

## Add Prospective Associate Screen:

## ProspectiveAssociate/GetEmpTypes:

* The above method is for Roles = "System Admin, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ProspectiveAssociateController.cs
    - **Method:** GetEmpTypes
    - This method does not have any input.
    - This Method used for fetching Employment Type list for dropdown.
    - Linq query is used in this method for getting Employee TypeDetails. The query works as follows.
* Purpose: To fetch Employment type list.
  + - * Tables: EmployeeTypes
      * Decision: Retain LINQ.
      * Rules:
* Fetch all active Employee types
  + - * Return: Employee type detail list.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write LINQ with below details:
      * Purpose: To fetch employee type details.
      * Database Tables:
        + **HRMSEmployee:**

EmployeeTypes

* + - * Cross communication: NA.
      * Return: The Employee Type related details.
    - API Details:
      * MicroService: Employee
      * Controller: EmployeeController.cs
      * Method: GetEmpTypes
      * Path: {{HostName}}/employee /api/v1/Employee/GetEmpTypes

## ProspectiveAssociate/GetHRAAdvisors:

* The above method is for Roles = "System Admin, HRA, HRM".

**Existing Implementation:**

* + - This method does not have any input.
    - **Controller**: ProspectiveAssociateController.cs
    - **Method:** GetHRAAdvisors
    - This Method used for fetching HRAdvisor list for dropdown.
    - Linq query is used in this method for getting Employee TypeDetails. The query works as follows.
* Purpose: To fetch HRAAdvisors list.
  + - * Tables: UserRoles, Roles and Users.
      * Decision: Retain LINQ.
      * Rules: This method fetches all active HRA users.
      * Return: HRA Advisor list details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write LINQ with below details:
      * Purpose: To fetch HRA Advisor details.
      * Database Tables:
        + **HRMSOrganization:**

UserRoles

Roles

Users

* + - * Cross communication: No
    - Return: The HRA Advisor related details
    - API Details:
      * MicroService: Admin
      * Controller: UserRoleController.cs
      * Method: GetHRAAdvisors
      * Path: {{HostName}}/admin/api/v1/UserRole/GetHRAAdvisors

## PracticeArea/GetPracticeAreas:

* The above method is for Roles = "System Admin, HRA".

**Existing Implementation:**

* + - This method does not have any input.
    - **Controller**: PracticeAreaController.cs
    - **Method:** GetPracticeAreas
    - This Method used for fetching Technology list for dropdown.
    - Call usp\_GetPracticeAreas.
    - usp\_GetPracticeAreas:
* Purpose: To fetch Technology list.
  + - * Tables: PracticeArea.
      * Decision: Convert SP to LinQ.
      * Rules: This SP fetches all active Practice Areas.
      * Return: PracticeArea list details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Convert the SP to LinQ:
      * Purpose: To fetch Technology details.
      * Database Tables:
        + **HRMSOrganization:**

PracticeArea

* + - * Cross communication:NA.
    - Return: The PracticeArea related details
    - API Details:
      * MicroService: Admin
      * Controller: PracticeAreaController.cs
      * Method: GetAll
      * Path: {{HostName}}/admin/api/v1/PracticeArea /GetAll

## UserDepartment/GetDepartmentDetails:

* The above method is for Roles = "System Admin, HRA, HRM".

**Existing Implementation:**

* + - This method have isActive flag as input.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| isActive | Bool |

* + - **Controller**: UserDepartmentController.cs
    - **Method:** GetDepartmentDetails
    - This Method used for fetching Department list for dropdown.
    - Linq query is used in this method for getting Department Details. The query works as follows.
* Purpose: To fetch Department list.
  + - * Tables: Department, DepartmentType and Employee.
      * Decision: Retain LINQ.
      * Rules: This method fetches all active Departments.
      * Return: Department list details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write LINQ with below details:
      * Purpose: To fetch Department details.
      * Database Tables:
        + **HRMSOrganization:**

Department

DepartmentType

* + - * + **HRMSEmployee:**

Employee

* + - * Cross communication: NA.
    - Return: The Department related details
    - API Details:
      * MicroService: Admin
      * Controller: DepartmentController.cs
      * Method: GetAll
      * Path: {{HostName}}/admin/api/v1/Department/GetAll
    - This method have isActive flag as input.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| isActive | Bool |

## MasterData/GetManagersandCompetencyLeads:

* The above method is for Roles = "System Admin, HRA and HRM".

**Existing Implementation:**

* + - This method does not have any input.
    - **Controller**: MasterDataController.cs
    - **Method:** GetManagerandCompetencyLeads
    - This Method used for fetching manager list for dropdown.
    - Call usp\_GetManagersandCompetencyLeads.
    - usp\_GetManagersandCompetencyLeads:
* Purpose: To fetch Managers list.
  + - * Tables: UserRoles, Roles, Projects, ProjectManagers, Employee, Department.
      * Decision: Convert SP to LinQ.
      * Rules: This SP will work as follows:
* Retrieves Program managers using projects and ProjectManagers tables(calls usp\_GetEmployeeFullName function to get employee full name).
* Retrieves Reporting Mangers using Projects and ProjectManagers tables(calls usp\_GetEmployeeFullName function to get employee full name).
* Retrieves Competency Leads using UserRoles, Roles and Employee tables(calls usp\_GetEmployeeFullName function to get employee full name)..
* Retrieves Department Heads using Employee and Department tables(calls usp\_GetEmployeeFullName function to get employee full name)..
* Retrieves Leads using Projects and ProjectManagers table(calls usp\_GetEmployeeFullName function to get employee full name)..
* Performs Union operation between all the above five results and gets the managers details.
  + - * Return: Manager list details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Convert the SP to LinQ:
      * Purpose: To fetch manager details.
      * Database Tables:
        + **HRMSOrganization:**

UserRoles

Roles

Department

* + - * + **HRMSEmployee:**

Employee

* + - * + **HRMSProject:**

Project

ProjectManagers

* + - * Cross communication: need to communicate with Admin Micro service and Project micro service.
    - Return: The manager related details
    - API Details:
      * MicroService: Employee
      * Controller: ProspectiveAssociateController.cs
      * Method: GetManagersAndCompetencyLeads
      * Path: {{HostName}}/employee/api/v1/ProspectiveAssociate / GetManagersAndCompetencyLeads

## ProspectiveAssociate/AddPADetails:

* The above method is for Roles = "System Admin, HRA, HRM".

**Existing Implementation:**

* + - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| UserDetails | UserDetails |

* + - **Controller**: ProspectiveAssociateController.cs
    - **Method:** AddPADetails
    - This Method used for Creating new Prospective Associate Record.
    - Linq query is used in this method for creating prospective associate. The query works as follows.
* Purpose: To create Prospective Associate.
  + - * Tables: Departments, PracticeAreas and ProspectiveAssociate.
      * Decision: Retain LINQ.
      * Rules:
* Checks whether the email id already exists.
* Checks whether the User already Exists.
* Checks whether the department is training department, if yes assigns practice area to training.
* Adds the record into Prospective Associate table.
  + - * Return: Boolean.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write LINQ with below details:
      * Purpose: To create prospective Associate.
      * Database Tables:
        + **HRMSOrganization:**

Department

PracticeAreas

* + - * + **HRMSEmployee:**

ProspectiveAssociate

* + - * Cross communication: need to communicate with Admin Micro service.
    - Return: The HRA Advisor related details
    - API Details:
      * MicroService: Employee
      * Controller: ProspectiveAssociateController.cs
      * Method: Create
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| ProspectiveAssociate | ProspectiveAssociate |

* + - * Path: {{HostName}}/employee /api/v1/ProspectiveAssociate/Create

## MasterData/GetDesignations:

* The above method is for Roles = "System Admin, HRA and HRM".

**Existing Implementation:**

* + - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| searchString | String |

* + - **Controller**: MasterDataController.cs
    - **Method:** GetDesignations
    - This Method used for fetching Designation list for dropdown by searchstring.
    - Call usp\_GetDesignations with parameter SearchString.
    - usp\_GetDesignations:
* Purpose: To fetch Designations list by search string.
  + - * Tables: Designation.
      * Decision: Convert SP to LinQ.
      * Rules: This SP fetches all active Designations which matches the SearchString.
      * Return: Designations list details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Convert the SP to LinQ:
      * Purpose: To fetch Designation list by searchstring.
      * Database Tables:
        + **HRMSOrganization:**

Designations

* + - * Cross communication: NA
    - Return: The Designation related details
    - API Details:
      * MicroService: Admin
      * Controller: DesignationController.cs
      * Method: GetDesignationBySearchString
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| searchString | String |

* + - * Path:{{HostName}}/admin/api/v1/Designation/GetDesignationBySearchString

## ProspectiveAssociate/GetGradeByDesignation:

* The above method is for Roles = "System Admin, HRA and HRM".

**Existing Implementation:**

* + - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| designationId | int |

* + - **Controller**: ProspectiveAssociateController.cs
    - **Method:** GetGradeByDesignation
    - This Method used for getting Grade by designationId.
    - Call usp\_GetGradeByDesignation with parameter designationId.
    - usp\_GetGradeByDesignation:
* Purpose: To fetch Grade by given DesignationId.
  + - * Tables: Designations, Grades.
      * Decision: Convert SP to LinQ.
      * Rules: This SP fetches the Active Grade by given Designation id.
      * Return: Grade Related details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Convert the SP to LinQ:
      * Purpose: To fetch Grade by Designation Id.
      * Database Tables:
        + **HRMSOrganization:**

Designations

Grades

* + - * Cross communication: NA.
    - Return: The Grade related details
    - API Details:
      * MicroService: Admin
      * Controller: GradeController.cs
      * Method: GetGradeByDesignationId
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| DesignationId | int |

* + - * Path:{{HostName}}/admin/api/v1/Grade/GetGradeByDesignationId

## ProspectiveAssociate/GetPADetailsByID:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: ProspectiveAssociateController.cs
    - **Method:** GetPADetailsByID
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| Id | int |

* + - This Method is used for getting Prospective Associate details by id.
    - Linq query is used in this method for getting Prospective Associate Details by id. The query works as follows
      * Purpose: This Query is to fetch Prospective Associate Details by id.
      * Tables: ProspectiveAssociates, Departments, Designations, and Grades.
      * Decision: Retain LINQ.
      * Rules:
* Fetch all active Prospective Associate details by id.
  + - * Return: prospective Associate related details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: To get Prospective Associate details
      * Database Tables:
      * **HRMSOrganization:**
        + Department
        + Designation
        + Grades
      * **HRMSEmployee:** 
        + ProspectiveAssociate
    - Cross communication: Cross communication required for fetching data Department, Designations and user Practice Area.
    - Return: Prospective Associate related details.
  + API Details:
    - MicroService: Employee
    - Controller: ProspectiveAssociateController.cs
    - Method: GetById
    - Path: {{HostName}}/Employee/api/v1/prospectiveAssociate/GetById

## ProspectiveAssociate/UpdatePADetails:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: ProspectiveAssociateController.cs
    - **Method:** UpdatePADetails
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| userDetails | UserDetails |

* + - This Method is used for updating Prospective Associate details.
    - Linq query is used in this method for updating Prospective Associate Details. The query works as follows
      * Purpose: This Query is to update Prospective Associate Details.
      * Tables: Departments, PracticeAreas and ProspectiveAssociate.
      * Decision: Retain LINQ.
      * Rules:
* Checks whether the email id already exists.
* Checks whether the User already Exists.
* Checks whether the department is training department, if yes assigns practice area to training.
* Updates the record in Prospective Associate table.
* Return: Boolean

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: To Update Prospective Associate details
      * Database Tables:
      * **HRMSOrganization:**
        + Department
        + PracticeArea
      * **HRMSEmployee:** 
        + ProspectiveAssociate
    - Cross communication: Cross communication required for fetching data Department and user Practice Area.
    - Return: Updated Prospective Associate related details.
  + API Details:
    - MicroService: Employee
    - Controller: ProspectiveAssociateController.cs
    - Method: Update
    - Path: {{HostName}}/Employee/api/v1/prospectiveAssociate/Update

## Associate Joining Screen:

## ProspectiveAssociate/GetJoinedAssociates:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: ProspectiveAssociateController.cs
    - **Method:** GetJoinedAssociates
    - This method does not have any input.
    - This Method is used for getting Joined Associate details.
    - Call SP usp\_GetJoinedAssociates. This SP works as follows
      * Purpose: This Query is to fetch Joined Associate Details.
      * Tables: Employee, Departments, Designations, and PracticeArea.
      * Decision: Convert SP to LINQ.
      * Rules:
* Fetch all active Joined Associate details.
  + - * Return: all Active joined Associate related details including department, designation and Technology details.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: To get Active Joined Associate details
      * Database Tables:
      * **HRMSOrganization:**
        + Department
        + Designation
        + PracticeArea
      * **HRMSEmployee:** 
        + Employee
    - Cross communication: Cross communication required for fetching data Department, Designations and user Practice Area.
    - Return: All active joined Associate related details.
  + API Details:
    - MicroService: Employee
    - Controller: EmployeeController.cs
    - Method: GetJoinedAssociates
    - Path:{{HostName}}/Employee/api/v1/EmployeeController/GetJoinedAssociates

## Associate Information Screen:

## AssociatePersonalDetails/GetAssociateDetails:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: AssociatePersonalDetailsController.cs
    - **Method:** GetAssociateDetails
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| departmentId | int |

.

* + - This Method is used for getting Associate Information details by departmentId.
    - Linq query is used in this method for getting Associate Information details. The query works as follows
      * Purpose: This Query is to getting Associate Information details by department id.
      * Tables: Status, Employees and Departments.
      * Decision: Retain LINQ.
      * Rules:
* Checks whether the employee is having approved status.
* Return: returns all the active associates by given departmentId

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: To get Active Associate details by department Id
      * Database Tables:
      * **HRMSOrganization:**
        + Department
        + status
      * **HRMSEmployee:** 
        + Employee
    - Cross communication: Cross communication required for fetching data from Department and Status.
* Return: returns all the active associates by given departmentId.
  + API Details:
    - MicroService: Employee
    - Controller: EmployeeController.cs
    - Method: GetEmployeeInformation
    - Path:{{HostName}}/Employee/api/v1/EmployeeController/ GetEmployeeInformation/departmentId

## Education Tab:

## AssociateEducationDetail/SaveEducationDetails:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: AssociateEducationDetailController.cs
    - **Method:** SaveEducationDetails
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| details | UserDetails |

.

* + - This Method is used for creating or updating Education Details.
    - Linq query is used in this method for creating or updating Education Details. The query works as follows
      * Purpose: This Query is for creating or updating Education Details.
      * Tables: EducationDetails.
      * Decision: Retain LINQ.
      * Rules:
* Checks whether there are any duplicate education details in the education details table for the employee. If yes, then make those records InActive.
* Checks whether the record exists in the education details table by id of input parameter details, if yes then update the record with new details, if no then create new education record.
* Return: returns boolean.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: To create or update Education Details
      * Database Tables:
      * **HRMSEmployee:** 
        + EducationDetails
    - Cross communication: No
* Return: returns EducationDetails related details.
  + API Details:
    - MicroService: Employee
    - Controller: EmployeeEducationController.cs
    - Method: Save
    - Path:{{HostName}}/Employee/api/v1/EmployeeEducationController/Save

## AssociateEducationDetail/GetEducationDetailsByID:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: AssociateEducationDetailController.cs
    - **Method:** GetEducationDetailsByID
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| empID | Int |

.

* + - This Method is used for getting Education Details by empID.
    - Linq query is used in this method for getting Education Details by empID. The query works as follows
      * Purpose: This Query is for getting Education Details by empID.
      * Tables: EducationDetails.
      * Decision: Retain LINQ.
      * Rules:
* Checks whether the record exists by input parameter empID.
* Return: returns EducationDetail related data.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: getting Education Details by empID.
      * Database Tables:
      * **HRMSEmployee:** 
        + EducationDetails
    - Cross communication: No
* Return: returns EducationDetails related details.
  + API Details:
    - MicroService: Employee
    - Controller: EmployeeEducationController.cs
    - Method: GetById
    - Path:{{HostName}}/Employee/api/v1/EmployeeEducationController/GetById

## Common/GetBusinessValues:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: CommonController.cs
    - **Method:** GetBusinessValues
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| ValueKey | String |

.

* + - This Method is used for getting Business Values by ValueKey.
    - Linq query is used in this method for getting Business Values by ValueKey. The query works as follows
      * Purpose: This Query is for getting Business Values by ValueKey.
      * Tables: lkValue and ValueType.
      * Decision: Retain LINQ.
* Return: returns Business Values related data.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: getting Business Values by ValueKey.
      * Database Tables:
      * **HRMSEmployee:** 
        + lkValue
        + ValueType
    - Cross communication: No
* Return: returns Business Value related details.
  + API Details:
    - MicroService: Employee
    - Controller: EmployeeController.cs
    - Method: GetBusinessValues
    - Path:{{HostName}}/Employee/api/v1/EmployeeController/GetBusinessValues

## Employment Tab:

## AssociateEmployment/SaveEmploymentDetails:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: AssociateEmploymentController.cs
    - **Method:** SaveEmploymentDetails
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| details | UserDetails |

.

* + - This Method is used for creating or updating Employment Details.
    - Linq query is used in this method for creating or updating Employment Details. The query works as follows
      * Purpose: This Query is to creating or updating Employment Details.
      * Tables: PreviousEmploymentDetails, ProfessionalReference.
      * Decision: Retain LINQ.
      * Rules:
* Checks whether there are any duplicate previous Employment details in the PreviousEmploymentDetails table for the employee. If yes, then make those records InActive.
* Checks whether the record exists in the employment details table by id of input parameter details, if yes then update the record with new details, if no then create new employment record.
* Checks whether there are any duplicate professional Reference details in the ProfessionalReferences table for the employee. If yes, then make those records InActive.
* Checks whether the record exists in the ProfessionalReferences table by id of input parameter details, if yes then update the record with new details, if no then create new Professional Reference record.
* Return: returns boolean.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: To create or update Employment Details
      * Database Tables:
      * **HRMSEmployee:** 
        + PreviousEmploymentDetails
        + ProfessionalReferences
    - Cross communication: No
* Return: returns Employment related details.
  + API Details:
    - MicroService: Employee
    - Controller: EmployeeEmploymentController.cs
    - Method: Save
    - Path:{{HostName}}/Employee/api/v1/ EmployeeEmploymentController/Save

## AssociateEmployment/GetEmploymentDetailsByID:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: AssociateEmploymentController.cs
    - **Method:** GetEmploymentDetailsByID
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| empID | Int |

.

* + - This Method is used for getting Previous Employment Details by empID.
    - Linq query is used in this method for getting Previous Employment Details by empID. The query works as follows
      * Purpose: This Query is for getting Previous Employment Details by empID.
      * Tables: PreviousEmploymentDetails.
      * Decision: Retain LINQ.
      * Rules:
* Checks whether the record exists by input parameter empID.
* Return: returns Previous Employment related data.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: getting Previous Employment Details by empID.
      * Database Tables:
      * **HRMSEmployee:** 
        + PreviousEmploymentDetails
    - Cross communication: No
* Return returns Previous Employment related data.
  + API Details:
    - MicroService: Employee
    - Controller: EmployeeEmploymentController.cs
    - Method: GetEmploymentDetailsById
    - Path:{{HostName}}/Employee/api/v1/EmployeeEmploymentController/GetEmploymentDetailsById

## AssociateEmployment/GetProfReferenceDetailsByID:

* The above method is for Roles = "**System Admin**, **HRA**, **HRM**".

**Existing Implementation:**

* + - **Controller**: AssociateEmploymentController.cs
    - **Method:** GetProfReferenceDetailsByID
    - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| empID | int |

.

* + - This Method is used for getting Professional Reference Details by empID.
    - Linq query is used in this method for getting Professional Reference Details by empID. The query works as follows
      * Purpose: This Query is for getting Professional Reference Details by empID.
      * Tables: ProfessionalReferences.
      * Decision: Retain LINQ.
      * Rules:
* Checks whether the record exists by input parameter empID.
* Return: returns Professional References related data.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Write Linq with below details
      * Purpose: getting Professional References Details by empID.
      * Database Tables:
      * **HRMSEmployee:** 
        + ProfessionalReferences
    - Cross communication: No
* Return returns Professional References related data.
  + API Details:
    - MicroService: Employee
    - Controller: EmployeeEmploymentController.cs
    - Method: GetProfReferencesById
    - Path:{{HostName}}/Employee/api/v1/EmployeeEmploymentController/GetProfReferencesById

# REFERENCES