|  |
| --- |
|  |
|  |
| **HRMS- Project Api analysis V4** |
|  |



**Process Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ver. No** | **Date** | **Author** | **Approved by** | **Description** |
| 1 | 23-Jun-2020 | Vishwajeet Velapurkar | Jayanth Chincholi | Writing technical documentation on the Project API micros services analysis. |
| 2 | 25-Jun-2020 | Vishwajeet Velapurkar | Jayanth Chincholi | Cosmic Changes in document. |
| 3 | 08-July-2020 | Vishwajeet Velapurkar | Jayanth Chincholi | Added Analysis for SOW tab. |
| 4 | 15-July-2020 | Vishwajeet Velapurkar | Jayanth Chincholi | Added Analysis for Client billing roles tab. |

Table of Contents

[1 PURPOSE 5](#_Toc45819731)

[2 SCOPE 5](#_Toc45819732)

[3 ACRONYMS AND DEFINITIONS 5](#_Toc45819733)

[4. Project 5](#_Toc45819734)

[4.1. Project Dashboard: 5](#_Toc45819735)

[4.1.1. Project/GetProjectsList: 5](#_Toc45819736)

[4.2. Project Add/Update: 7](#_Toc45819737)

[4.2.1. Project Detail Tab: 7](#_Toc45819738)

[4.2.1.1. Project/GetProjectTypes 7](#_Toc45819739)

[4.2.1.2. Project/GetCustomers 8](#_Toc45819740)

[4.2.1.3. UserDepartment/GetUserDepartmentDetails 8](#_Toc45819741)

[4.2.1.4. Project/AssignProgramMangerToProject 9](#_Toc45819742)

[4.2.1.5. Project/PostProject 11](#_Toc45819743)

[4.2.1.6. Project/PutProject 12](#_Toc45819744)

[4.2.2. SOW Tab: 14](#_Toc45819745)

[4.2.2.1. Project/GetProject 14](#_Toc45819746)

[4.2.2.2. Project/GetSowsByProjectId 16](#_Toc45819747)

[4.2.2.3. Project/GetSOWDetails 17](#_Toc45819748)

[4.2.2.4. Project/DeleteSOW 18](#_Toc45819749)

[4.2.2.5. Project/PostSOW 19](#_Toc45819750)

[4.2.2.6. Project/UpdateSOWAndAddendumDetails 21](#_Toc45819751)

[4.2.2.7. Project/GetAddendums 22](#_Toc45819752)

[4.2.2.8. Project/GetAddendumDetailsById 24](#_Toc45819753)

[4.2.3. Client Billing Roles Tab: 25](#_Toc45819754)

[4.2.3.1. Common/GetAllocationPercentages 25](#_Toc45819755)

[4.2.3.2. Reports/GetResourceByProject 26](#_Toc45819756)

[4.2.3.3. ClientBillingRole/GetClientBillingRolesByProjectId 28](#_Toc45819757)

[4.2.3.4. ClientBillingRole/CreateClientBillingRole 29](#_Toc45819758)

[4.2.3.5. ClientBillingRole/UpdateClientBillingRole 30](#_Toc45819759)

[4.2.3.6. ClientBillingRole/DeleteClientBillingRole 32](#_Toc45819760)

[4.2.3.7. ClientBillingRole/CloseClientBillingRole 33](#_Toc45819761)

[ REFERENCES 34](#_Toc45819762)

# 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 one of the project screens.

# ACRONYMS AND DEFINITIONS

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

## Project

## Project Dashboard:

## Project/GetProjectsList:

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

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

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| userRole | string |
| employeeId | int |
| dashboard | string |

* + - This Method used for fetching project detail based on assigned roles, employee Id and dashboard.
    - Call usp\_GetProjectDetail with parameters roleName and employeeId
    - usp\_GetProjectDetail:
      * Purpose: This stored procedure is to fetch a project record.
      * Tables: Projects, ProjectManagers, Clients, PracticeArea, ProjectType, Employee, Status, Users, Roles and CategoryMaster
      * Decision: Move the stored procedure to LINQ.
      * Rules:
* This allows user with role program Manager and department head to access project data.
* Project manager can fetch all active project which belong to him and which are not closed.
* Department head can fetch all which are not closed and in drafted. Department head can pull drafted project which are created by him.
  + - * Return: all project related details including client details.
    - Project Manager Dashboard and Department Head Dashboard show project which are send for approval.
    - Project Manager Dashboard show project which are sent not sent for approval.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

Projects

ProjectManagers

* + - * + **HRMSOrganization:**

Users

Status

Roles

CategoryMaster

Clients

PracticeArea

* + - * + **HRMSEmployee:**

Employee

* + - * Cross communication: Cross communication required for fetching data Status, User roles, Employee, and user API’s.
      * Return: The project related details.
    - API Details:
      * MicroService: Project
      * Controller: ProjectController.cs
      * Method: GetAll
      * Path: {{HostName}}/project/api/v1/Project/ GetAll
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| userRole | string |
| employeeId | int |
| dashboard | string |

## Project Add/Update:

## Project Detail Tab:

## Project/GetProjectTypes

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** GetProjectTypes
    - This method does not have any input.
    - This Method used for fetching project type list for dropdown.
    - Call usp\_GetProjectTypes without any parameter.
    - usp\_GetProjectTypes:
* Purpose: This SP fetches project type list.
  + - * Tables: ProjectTypes
      * Decision: Move the stored procedure to LINQ.
      * Rules:
* Fetch all active project types
  + - * Return: project type detail list.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSOrganization:**

ProjectTypes

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

## Project/GetCustomers

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - This method does not have any input.
    - **Controller**: ProjectController.cs
    - **Method:** GetCustomers
    - This Method used for fetching client list for dropdown.
    - Call usp\_GetClients without any parameter.
    - usp\_GetClients:
* Purpose: This SP fetches client list.
  + - * Tables: Clients
      * Decision: Move the stored procedure to LINQ.
      * Rules: This method fetches all active clients.
      * Return: client details

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Make use of already existing API by sending IsActive parameter as true.
    - API Details:
      * MicroService: Admin
      * Controller: ClientController.cs
      * Method: GetAll
      * Path: {{HostName}}/admin/api/v1/Client/GetAll
      * Inputs: isActive (Boolean)

## UserDepartment/GetUserDepartmentDetails

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - This method does not have any input.
    - **Controller**: UserDepartmentController.cs
    - **Method:** GetUserDepartmentDetails
    - This Method used for fetching department list for dropdown.
    - Call usp\_GetUserDepartmentDetails without any parameter.
    - usp\_GetUserDepartmentDetails:
* Purpose: This SP fetches department list.
  + - * Tables: Departments, and Employee
      * Decision: Move the stored procedure to LINQ.
      * Rules: This method fetches all active department.
      * Return: department list along with department head info.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Tables: Department, and Employee
      * Database Tables:
        + **HRMSOrganization:**

Departments

* + - * + **HRMSEmployee:**

Employee

* + - * Cross communication: Cross communication required for fetching data from employee microservice.
      * Return: Department list.
    - API Details:
      * MicroService: Admin
      * Controller: DepartmentController.cs
      * Method: GetAllWithDepartmentHead
      * Path: {{HostName}}/admin/api/v1/Department/GetAllWithDepartmentHead
      * Inputs: NA.

## Project/AssignProgramMangerToProject

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| userRole | string |
| employeeId | int |

* + - **Controller**: ProjectController.cs
    - **Method:** AssignProgramMangerToProject
    - This method used for fetching program manager details based on assigned roles, and employee Id.
    - This method used for fetching program manager list for dropdown.
    - Call usp\_AssignProgramManagerToProject with userRole and employeeId.
    - usp\_AssignProgramManagerToProject:
* Purpose: This SP fetches program manager list.
  + - * Tables: UserRoles, Roles, and Employee.
      * Decision: Move the stored procedure to LINQ.
      * Rules:
* For program manager, need to fetch data for himself only.
* For Department Head, need to fetch data for all active program manager.
  + - * Return: program manager list

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSOrganization:**

UserRoles

Roles

* + - * + **HRMSEmployee:**

Employee

* + - * Cross communication: Cross communication required for fetching data from employee microservice.
      * Return: Program manager list.
    - API Details:
      * MicroService: Admin
      * Controller: UserRoleController.cs
      * Method: GetProgramManagers
      * Path: {{HostName}}/admin/api/v1/UserRole/GetProgramManagers
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| userRole | string |
| employeeId | int |

## Project/PostProject

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| project | ProjectData |

* + - **Controller**: ProjectController.cs
    - **Method:** PostProject
    - This method used for creating new project record.
    - If project parameter is null, then throw “Details must be provided.” error message.
    - Verify project code is not null or empty. If not exist, throw error “Project Code Should not be null”.
    - Call usp\_CreateProject by sending required parameters like project code and etc.
    - usp\_CreateProject:
* Purpose: This SP creates new entry for project table.
  + - * Tables: Projects, and ProjectManagers
      * Decision: Move the stored procedure to LINQ.
      * Rules:
* Create entry for project record.
* Create entry for associated project manager record by using programManagerId property.
  + - * Return: true/false

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Tables: Project, and project manager
      * Database Tables:
        + **HRMSProjectDB:**

Projects

ProjectManagers

* + - * + **HRMSOrganization:**

ProjectTypes

Domain

Clients

Departments

PracticeArea

Status

* + - * Cross communication: Cross communication is required for verifying entries exist for client, practice area, and other table with sent Id’s.
      * Return: Project
    - Write functionality of creating project manager separately.
    - API Details:
      * Project API.
* MicroService: Project
* Controller: ProjectController.cs
* Method: Create
* Path: {{HostName}}/project/api/v1/Project/Create
* Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| projectIn | Project |

* + - * Project Manager API.
* MicroService: Project
* Controller: ProjectManagerController.cs
* Method: Create
* Path: {{HostName}}/project/api/v1/ProjectManager/Create
* Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| projectIn | Project |

## Project/PutProject

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - Input parameter for this method is as below

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| project | ProjectData |

* + - **Controller**: ProjectController.cs
    - **Method:** PutProject
    - This method used for updating existing project record.
    - If project parameter is null, then throw “Details must be provided.” error message.
    - Call usp\_UpdateProject by sending required parameters like project code and etc.
    - usp\_UpdateProject:
* Purpose: This SP updates existing project table entries.
  + - * Tables: Projects, AssociateAllocation, ProjectManagers, Status, and CategoryMaster
      * Decision: Move the stored procedure to LINQ.
      * Rules:
* Project State is sent as a closed then verify AssociateAllocation exists for particular project. If exits, then don’t allow updating project record.
* If allocation does not exist, then project status to closed and inactivate project manager record.
* For project state other than closed, update project records as per the sent parameter.
* Project Manager record update are performed only for user with Department head.
  + - * Return: true/false

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

Projects

ProjectManagers

AssociateAllocation

* + - * + **HRMSOrganization:**

ProjectTypes

Domain

Clients

Departments

PracticeArea

Status

* + - * Cross communication: Cross communication is required for fetching associate allocation and for verifying master data exists or not like client.
      * Return: Project
    - Write functionality of updating project manager separately.
    - API Details:
      * Project API.
* MicroService: Project
* Controller: ProjectController.cs
* Method: Update
* Path: {{HostName}}/project/api/v1/Project/Update
* Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| projectIn | Project |

* + - * Project Manager API.
* MicroService: Project
* Controller: ProjectManagerController.cs
* Method: Update
* Path: {{HostName}}/project/api/v1/ProjectManager/Update
* Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| projectIn | Project |

## SOW Tab:

## Project/GetProject

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** GetProject
    - This method has below input parameter.

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

* + - This Method used for fetching project for dropdown.
    - Call usp\_GetProjectById with projectId parameter.
    - usp\_GetProjectById:
* Purpose: This SP fetches project record by using project Id.
  + - * Tables: Project, Project Managers, Status, Domain, Client, Project Type, Practice area, Departments, Employee
      * Decision: Move the stored procedure to LINQ.
      * Rules:
* Fetch active project.
* Fetch not closed project.
* Fetch project by using ProjectId
  + - * Return: project details

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

Projects

ProjectManagers

* + - * + **HRMSOrganization:**

ProjectTypes

Domain

Clients

Departments

PracticeArea

Status

* + - * + **HRMSEmployeeDB:**

Employee

* + - * Cross communication: required for fetching project type and other data.
      * Return: The project related details.
    - API Details:
      * MicroService: Project
      * Controller: ProjectController.cs
      * Method: GetByProjectId
      * Path: {{HostName}}/project/api/v1/Project/GetByProjectId
      * Inputs: projectId (int)

## Project/GetSowsByProjectId

* The above method is for Roles = "Program Manager, Department Head ".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** GetSowsByProjectId
    - This method has below input parameter.

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

* + - This method used for fetching SOW list for dropdown.
    - This method is also used for fetching SOW for grid.
    - Call usp\_GetSowsByProjectId with projectId parameter.
    - usp\_GetSowsByProjectId:
* Purpose: This SP fetches SOW list.
  + - * Tables: SOW
      * Decision: Move the stored procedure to LINQ.
      * Rules: This method fetches SOW which belongs to project.
      * Return: SOW details

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

SOW

* + - * Cross communication: N.A.
      * Return: The SOW details.
    - API Details:
      * MicroService: Project
      * Controller: SOWController.cs
      * Method: GetByProjectId
      * Path: {{HostName}}/project/api/v1/SOW/GetAllByProjectId
      * Inputs: projectId (int)

## Project/GetSOWDetails

* The above method is for Roles = "Program Manager, Department Head ".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** GetSOWDetails
    - This method has below input parameter.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| id | Int |
| projectId | Int |
| roleName | string |

* + - This method used for fetching SOW for update.
    - Call usp\_GetSOWDetails with id, projectId, and roleName parameter.
    - usp\_GetSOWDetails:
* Purpose: This SP fetches SOW.
  + - * Tables: SOW
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Fetch SOW only if user role is Program Manager or Department head.
        + Fetch SOW based on Id and projectId.
      * Return: SOW details

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

SOW

* + - * Cross communication: N.A.
      * Return: The SOW details.
    - API Details:
      * MicroService: Project
      * Controller: SOWController.cs
      * Method: GetByIdAndProjectId
      * Path: {{HostName}}/project/api/v1/SOW/ GetByIdAndProjectId
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| id | Int |
| projectId | Int |
| roleName | string |

## Project/DeleteSOW

* The above method is for Roles = "Program Manager, Department Head ".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** DeleteSOW
    - This method has below input parameter.

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

* + - This method used for deleting SOW.
    - Call usp\_DeleteSOW with id.
    - usp\_GetSOWDetails:
* Purpose: This SP deletes SOW.
  + - * Tables: SOW
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Deletes SOW based on Id
      * Return: true/false

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

SOW

* + - * Cross communication: N.A.
      * Return: true/false
    - API Details:
      * MicroService: Project
      * Controller: SOWController.cs
      * Method: Delete
      * Path: {{HostName}}/project/api/v1/SOW/Delete
      * Inputs:

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

## Project/PostSOW

* The above method is for Roles = "Program Manager, Department Head ".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** PostSOW
    - This method has below input parameter.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| sowSign | SOWSignData |

* + - This method used for creating SOW and Addendum.
    - If SOW information is sent, then call usp\_CreateSOW with SOW related data.
    - usp\_CreateSOW:
* Purpose: This SP creates SOW.
  + - * Tables: SOW
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Create new SOW based sent information.
      * Return: true/false

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

SOW

* + - * Cross communication: N.A.
      * Return: true/false
    - API Details:
      * MicroService: Project
      * Controller: SOWController.cs
      * Method: Create
      * Path: {{HostName}}/project/api/v1/SOW/Create
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| sowIn | SOW |

* + - If Addenduminformation is sent, then call usp\_CreateAddendum with Addendum related data.
    - usp\_CreateAddendum
* Purpose: This SP create new addendum.
  + - * Tables: Addendum
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Create new Addendum based sent information.
      * Return: true/false

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

Addendum

* + - * Cross communication: N.A.
      * Return: true/false
    - API Details:
      * MicroService: Project
      * Controller: AddendumController.cs
      * Method: Create
      * Path: {{HostName}}/project/api/v1/Addendum/Create
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| addendumIn | Addendum |

## Project/UpdateSOWAndAddendumDetails

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** UpdateSOWAndAddendumDetails
    - This method has below input parameter.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| sowAndAddendumData | SOWSignData |

* + - This method used for updating SOW and Addendum.
    - If SOW information is sent, then call usp\_UpdateSOWDetails with SOW related data.
    - usp\_UpdateSOWDetails:
* Purpose: This SP updates SOW.
  + - * Tables: SOW, Roles
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Updates SOW based sent information.
        + Update is allowed only Program Manager and Department Head user.
      * Return: true/false

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

SOW, Roles

* + - * Cross communication: N.A.
      * Return: true/false
    - API Details:
      * MicroService: Project
      * Controller: SOWController.cs
      * Method: Update
      * Path: {{HostName}}/project/api/v1/SOW/Update
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| sowIn | SOW |

* + - If Addenduminformation is sent, then call usp\_UpdateAddendumDetails with Addendum related data.
    - usp\_UpdateAddendumDetails:
* Purpose: This SP updates existing addendum.
  + - * Tables: Addendum
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Update Addendum based sent information.
        + Only Program Managers can update Addendum.
      * Return: true/false

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

Addendum

* + - * Cross communication: N.A.
      * Return: true/false
    - API Details:
      * MicroService: Project
      * Controller: AddendumController.cs
      * Method: Update
      * Path: {{HostName}}/project/api/v1/Addendum/Update
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| addendumIn | Addendum |

## Project/GetAddendums

* The above method is for Roles = "Program Manager, Department Head".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** GetAddendums
    - This method has below input parameter.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| Id (SOWId) | int |
| projectId | int |

* + - This method used for fetching list addendums.
    - Call usp\_GetAddendums with id and projectId
    - usp\_UpdateSOWDetails:
* Purpose: This SP fetches list of addendums.
  + - * Tables: Addendum
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Fetch list of addendums-based SOW and product.
      * Return: List of Addendums

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

Addendum

* + - * Cross communication: N.A.
      * Return: List of Addendums
    - API Details:
      * MicroService: Project
      * Controller: AddendumController.cs
      * Method: GetBySOWIdAndProjectId
      * Path: {{HostName}}/project/api/v1/Addendum/GetAllBySOWIdAndProjectId
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| Id (SOWId) | int |
| projectId | int |

## Project/GetAddendumDetailsById

* The above method is for Roles = "Program Manager, Department Head".

**Existing Implementation:**

* + - **Controller**: ProjectController.cs
    - **Method:** GetAddendumDetailsById
    - This method has below input parameter.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| Id (SOWId) | int |
| projectId | int |
| roleName | string |

* + - This method used to fetch addendum for update.
    - Call usp\_GetAddendumsBySOWId with id, projectId, and roleName.
    - usp\_UpdateSOWDetails usp\_GetAddendumsBySOWId:
* Purpose: This SP fetches addendum.
  + - * Tables: Addendum, Roles
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Fetch addendums-based addendum id and product.
        + User with role Program Manager and Department head are allowed to fetch the record.
      * Return: Addendum

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

Addendum

Role

* + - * Cross communication: N.A.
      * Return: Addendum
    - API Details:
      * MicroService: Project
      * Controller: AddendumController.cs
      * Method: GetByIdandProjectId
      * Path: {{HostName}}/project/api/v1/Addendum/GetByIdandProjectId
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| Id (SOWId) | int |
| projectId | int |
| roleName | string |

## Client Billing Roles Tab:

## Common/GetAllocationPercentages

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: CommonController.cs
    - **Method:** GetAddendumDetailsById
    - This method does not have any parameters input parameter.
    - This method used to fetch allocation percentage in dropdown list.
    - There is LINQ query used to fetch the required data
      * Tables: AllocationPercentage
      * Return: List of allocation percentage details
    - Fetch all allocation percentage records.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - Continue with LINQ query
      * Database Tables:
        + **HRMSProjectDB:**

AllocationPercentage

* + - * Cross communication: N.A.
      * Return: List of allocation percentage details
    - API Details:
      * MicroService: Project
      * Controller: AllocationPercentageController.cs
      * Method: GetAll
      * Path: {{HostName}}/project/api/v1/AllocationPercentage/GetAll
      * Inputs: N.A.

## Reports/GetResourceByProject

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ReportsController.cs
    - **Method:** GetResourceByProject
    - This method has below input parameter.

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

* + - This method used to fetch resource allocation report.
    - This implementation is divided in 3 parts.

1. Fetching count of billable and non-billable resources.
2. Fetching records of billable resources.
3. Fetching records of non-billable resources.
4. **Fetching count of billable and non-billable resources:**
   * + - Call usp\_GetResourceByProjectForReport with projectId
       - usp\_GetResourceByProjectForReport:
   * Purpose: This SP fetches count of billable and non-billable resources.
     + - * Tables: AssociateAllocation, and Projects.
         * Decision: Move the stored procedure to LINQ.
         * Rules:

Fetch count of billable and non-billable resources-based product id.

Billable resources count can be fetch from the associate allocation table where records are active, belong to project and billable flag is on.

Non-Billable resources count can be fetched from the associate allocation table where records are active, belong to project and billable flag is off.

* + - * + Return: project name, Count of billable and non-billable resources.

1. **Fetching records of billable resources:**
   * + - Call usp\_GetBillableResourcesByProjectForReport with projectId
       - usp\_GetBillableResourcesByProjectForReport:
   * Purpose: This SP fetches billable resources details.
     + - * Tables: AssociateAllocation, Employee, AllocationPercentage, InternalBillingRoles, and ClientBillingRoles
         * Decision: Move the stored procedure to LINQ.
         * Rules:

While fetching billable record need to verify below

Associate allocation project matches with sent project

Release date is null.

Employee and allocation are active.

* + - * + Return: Billable resource details

1. **Fetching records of non-billable resources:**
   * + - Call usp\_GetNonBillableResourcesByProjectForReport with projectId
       - usp\_GetNonBillableResourcesByProjectForReport:
   * Purpose: This SP fetches billable resources details.
     + - * Tables: AssociateAllocation, Employee, AllocationPercentage, InternalBillingRoles.
         * Decision: Move the stored procedure to LINQ.
         * Rules:

While fetching billable record need to verify below

Associate allocation project matches with sent project

Release date is null.

Employee and allocation are active.

* + - * + Return: Billable resource details

**New Implementation:**

* + - * From all above methods only Billable resource details are displayed in UI, so need to fetch billable resource detail
      * Make sure all older Business Rules are applied.
      * convert the SP to LINQ
        + Purpose:
        + Database Tables:

**HRMSProjectDB:**

AssociateAllocation

Employee

AllocationPercentage

InternalBillingRoles

ClientBillingRoles

* + - * + Cross communication: N.A.
        + Return: Billable resource details
    - API Details:
      * MicroService: Project
      * Controller: ClientBillingRolesController.cs
      * Method: GetBillableResouceReportByProjectId
      * Path: {{HostName}}/project/api/v1/ClientBillingRoles/GetBillableResouceInfoByProjectId
      * Inputs:

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

## ClientBillingRole/GetClientBillingRolesByProjectId

**Existing Implementation:**

* + - **Controller**: ClientBillingRoleController.cs
    - **Method:** GetClientBillingRolesByProjectId
    - This method has below input parameter.

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

* + - This method used to fetch client billing roles for grid.
    - Call usp\_GetClientBillingRolesByProjectId with projectId
    - usp\_GetClientBillingRolesByProjectId:
* Purpose: This SP fetches client billing roles.
  + - * Tables: ClientBillingRoles, Projects, AllocationPercentage, and AssociateAllocation
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Fetch client billing roles -based product id.
        + AllocationCount is calculated from Associate allocation table based on client billing role, project tables, and release date is null.
      * Return: Client Billing Role

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

ClientBillingRoles

Projects

AllocationPercentage

AssociateAllocation

* + - * Cross communication: N.A.
      * Return: Client Billing Role
    - API Details:
      * MicroService: Project
      * Controller: ClientBillingRolesController.cs
      * Method: GetByProjectId
      * Path: {{HostName}}/project/api/v1/ClientBillingRoles/GetByProjectId
      * Inputs:

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

## ClientBillingRole/CreateClientBillingRole

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ClientBillingRoleController.cs
    - **Method:** CreateClientBillingRole
    - This method has below input parameter.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| clientBillingRoleDetails | ClientBillingRoleDetails |

* + - This method used to create client billing roles.
    - Call usp\_CreateClientBillingRole with projectId
    - usp\_CreateClientBillingRole:
* Purpose: This SP creates client billing roles.
  + - * Tables: ClientBillingRoles
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Verify existence of client billing role based on number of position, start date, project, and client billing percentage.
      * Return: Client Billing Role

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

ClientBillingRoles

* + - * Cross communication: N.A.
      * Return: Client Billing Role
    - API Details:
      * MicroService: Project
      * Controller: ClientBillingRolesController.cs
      * Method: Create
      * Path: {{HostName}}/project/api/v1/ClientBillingRoles/Create
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| clientBillingRoleDetail | ClientBillingRoleDetails |

## ClientBillingRole/UpdateClientBillingRole

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ClientBillingRoleController.cs
    - **Method:** UpdateClientBillingRole
    - This method has below input parameter.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| clientBillingRoleDetails | ClientBillingRoleDetails |

* + - This method used to update client billing roles.
    - Call usp\_UpdateClientBillingRole with projectId
    - usp\_UpdateClientBillingRole:
* Purpose: This SP update client billing roles.
  + - * Tables: ClientBillingRoles, ClientBillingRolesHistory, AssociateAllocation, Projects, Status, and CategoryMaster.
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + When project state is Drafted or Create, and allocation count is greater than or equal to number of positions then updating client billing role is not allowed. Throw “Cannot decrease Positions” validation.
        + When client bill role like start date, number of positions, client billing percentage, and project id exists then throw “Client billing role already exist” validation.
        + Create Client Billing Roles History record and update client billing role table.
      * Return: Client Billing Role

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

ClientBillingRoles

ClientBillingRolesHistory

AssociateAllocation

Projects

* + - * + **HRMSOrganization:**

Status

CategoryMaster.

* + - * Cross communication: N.A.
      * Return: Client Billing Role
    - API Details:
      * MicroService: Project
      * Controller: ClientBillingRolesController.cs
      * Method: Update
      * Path: {{HostName}}/project/api/v1/ClientBillingRoles/Update
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| clientBillingRoleDetails | ClientBillingRoleDetails |

## ClientBillingRole/DeleteClientBillingRole

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ClientBillingRoleController.cs
    - **Method:** DeleteClientBillingRole
    - This method has below input parameter.

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

* + - This method used to delete client billing roles.
    - Call usp\_DeleteClientBillingRole with clientBillingRoleId
    - usp\_DeleteClientBillingRole:
* Purpose: This SP deletes client billing roles.
  + - * Tables: ClientBillingRoles
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Verify client billing role exists or not.
        + Delete client billing role.
      * Return: true/false

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

ClientBillingRoles

* + - * Cross communication: N.A.
      * Return: true/false
    - API Details:
      * MicroService: Project
      * Controller: ClientBillingRolesController.cs
      * Method: Delete
      * Path: {{HostName}}/project/api/v1/ClientBillingRoles/Delete
      * Inputs:

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

## ClientBillingRole/CloseClientBillingRole

* The above method is for Roles = "Program Manager, Department Head, HRA, HRM".

**Existing Implementation:**

* + - **Controller**: ClientBillingRoleController.cs
    - **Method:** CloseClientBillingRole
    - This method has below input parameter.

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| clientBillingRoleId | Int |
| endDate | DateTime |

* + - This method used to close client billing roles.
    - Call usp\_CloseClientBillingRole with clientBillingRoleId and end date
    - usp\_DeleteClientBillingRole:
* Purpose: This SP closes client billing roles.
  + - * Tables: ClientBillingRoles, AssocaiteAllocation
      * Decision: Move the stored procedure to LINQ.
      * Rules:
        + Closing of client billing role is only allowed only when there is no active allocation with client billing role
      * Return: count of update rows.

**New Implementation:**

* + - Make sure all older Business Rules are applied.
    - convert the SP to LINQ
      * Purpose:
      * Database Tables:
        + **HRMSProjectDB:**

ClientBillingRoles

AssocaiteAllocation

* + - * Cross communication: N.A.
      * Return: true/false
    - API Details:
      * MicroService: Project
      * Controller: ClientBillingRolesController.cs
      * Method: Close
      * Path: {{HostName}}/project/api/v1/ClientBillingRoles/Close
      * Inputs:

|  |  |
| --- | --- |
| **Parameter Name** | **Data Type** |
| closeClientBillingRoleRequest | closeClientBillingRoleRequest |

# REFERENCES