**Employee Management System – Back End**

This project includes functionality to track the number of departments and projects associated with a company and employees. It ensures that department counts for companies and project counts for employees are updated automatically when changes are made to the respective documents.

**Overview**

The Employee Management is designed to maintain and update counts for departments and projects linked to companies and employees.

**Key Features:**

* **Department Counts for Company**: Whenever a DepartmentEmp document is created or modified, the number of departments associated with the company is updated.
* **Project Counts for Employees**: The ProjectEmp document tracks employees assigned to projects, and their project count is updated whenever an assignment is added or changed.
* **Automatic Count Updates**: The counts are updated in real-time whenever relevant documents are validated.

**Implementation Details**

The project includes two primary classes and three key whitelisted functions:

**1. Class: DepartmentEmp**

* **Purpose**: Tracks departments within a company.
* **Method: before\_validate()**: Updates the company’s department count whenever a department document is validated.

**2. Class: ProjectEmp**

* **Purpose**: Tracks projects assigned to employees.
* **Method: validate()**: Updates the project counts for both the company and employee whenever a project document is validated.

**3. Whitelisted Functions:**

* **update\_company\_counts(company\_id)**: Updates the number of departments or projects associated with a given company.
* **update\_project\_count(employee\_id, project\_id)**: Updates the number of projects associated with a given employee.
* **update\_department\_counts(department\_id)**: Updates the number of projects associated with a given department.

Each function is designed to update counts efficiently by querying the database using frappe.db.count() to retrieve the relevant number of documents.

**Setup and Installation**

Follow these steps to set up and run the application:

**Prerequisites:**

* **Frappe Framework**: The system requires the Frappe Framework, which includes ERPNext.
* **Python**: Ensure Python is installed (preferably version 3.x).
* **Database**: A working MySQL or MariaDB database.

**Installation Steps:**

1. **Clone the Repository**:
   * Clone the repository to your local machine or server.
   * git clone https://github.com/Zero056/employee\_management.git
2. **Set Up the Frappe Environment**:
   * Install Frappe using the Frappe installation guide.
   * Set up the site and app:
   * bench init frappe-bench --frappe-branch version-15
   * bench new-site sitename
3. **Install the Custom App**:
   * Bench get-app employee\_management
   * Bench –site sitename install-app employee\_management
4. **Run the Server**:
   * After installation, run the Frappe server:
   * bench start

**Task Completion Checklist**

* **Created the DepartmentEmp Document** with a method to update department counts for companies.
* **Created the ProjectEmp Document** with a method to update project counts for employees.
* **Implemented update\_company\_counts()** to update department/project counts based on filters.
* **Implemented update\_project\_count()** to update the number of projects associated with employees.
* **Tested the logic** to ensure that counts are updated correctly.
* **Completed API functionality** for external access if needed (whitelisted functions).
* **Wrote test cases** to validate the functionality (optional, but recommended).

**Security Considerations**

The system uses role-based access control (RBAC) as implemented by the Frappe framework. Here are the security measures that were considered:

* **Access Control**: Only authorized users with specific roles can update project or department counts. You should configure roles and permissions as needed in the Frappe admin panel.
* **Whitelisted Functions**: The functions update\_company\_counts(), update\_project\_count(), and update\_department\_counts() are whitelisted, ensuring they can be called from external systems or API clients with proper access control.

Make sure to review and configure roles and permissions to restrict access to critical functions that modify project and department counts.

**API Endpoints and Access**

**API Endpoints:**

1. Login:
   * **Endpoint**: POST [{{url}}/api/method/employee\_management.api.login?username=za@gmail.com&password=123](mailto:%7b%7burl%7d%7d/api/method/employee_management.api.login?username=za@gmail.com&password=123)
   * **Parameters**:
     + company\_id (str): The ID of the company.
   * **Expected Response**: A JSON response indicating the success of the operation.
2. Get Company:
   * **Endpoint**: GET /{{url}}/api/method/employee\_management.api.get\_company
   * **Parameters**:
     + company\_id (str): The ID of the company.
   * **Expected Response**: A JSON response indicating the success of the operation.
3. **Get Department**:
   * **Endpoint**: GET {{url}}/api/method/employee\_management.api.get\_department
   * **Parameters**:
     + department\_id (str): The ID of the Department.
   * **Expected Response**: A JSON response indicating the success of the operation.
4. **Get Employee**:
   * **Endpoint**: GET {{url}}/api/method/employee\_management.api.get\_employee?employee\_id=tasbppqqk
   * **Parameters**:
     + employee\_id (str): The ID of the employee whose project count should be updated.
   * **Expected Response**: A JSON response indicating the success of the operation.
5. **Create Employee**:
   * **Endpoint**: POST {{url}}/api/method/employee\_management.api.create\_employee?email\_address={email} &company={ company \_id} &employee\_name=Galal&employee\_status={status} &department={ department\_id }&designation={ designation}&address={address} &mobile\_number={mob\_number}
   * **Parameters**:
     + employee\_id (str)

The ID of the employee whose project count should be updated.

* + - **email\_address** (str)

The email address of the employee. It is a unique identifier for the employee.

* + - **company** (str)

The company ID that the employee belongs to. This should be the unique identifier for the company in the system.

* + - **employee\_name** (str)

The name of the employee. A required field.

* + - **employee\_status** (str)

The status of the employee. Common values could be active, inactive, on\_leave, etc.

* + - **department** (str)

The department ID where the employee is assigned. This is the unique identifier for the department within the company.

* + - **designation** (str)

The designation of the employee, such as Software Engineer, Manager, etc.

* + - **address** (str)

The residential address of the employee.

* + - **mobile\_number** (str)

The mobile number of the employee. This field stores the phone number for communication.

* + **Expected Response**: A JSON response indicating the success of the operation.

1. **Update Employee**:
   * **Endpoint**: PATCH {{url}}/api/method/employee\_management.api.update\_employee?employee\_id=t1al91ed3f&designation=Backend
   * **Parameters**:
     + employee\_id (str)

The unique ID of the employee whose information is to be updated.

* + - designation (str)

The new designation of the employee (e.g., Backend Developer, Manager).

* + **Expected Response**: A JSON response indicating the success of the operation.

1. **Delete Employee**:
   * **Endpoint**: DELETE {{url}}/api/method/employee\_management.api.delete\_project?employee\_id
   * **Parameters**:
     + employee\_id (str)

The unique ID of the employee to be deleted.

* + **Expected Response**: A JSON response indicating the success of the operation.

**I live the postman json to download it and test if there is any wrong**

{{url}} the url of your site

Make sure you have the proper access permissions to invoke these API endpoints.

**Assumptions and Considerations**

* The system assumes that the DepartmentEmp, ProjectEmp, Employee Emp, and Company Emp documents are set up properly within Frappe and that the respective relationships (such as employee, company, and department) are correctly defined in the database.
* The counting logic works based on the current data in the database, and any changes to the structure of related documents may require adjustments to the counting functions.
* It's assumed that Frappe's default permissions and access control will suffice for the security needs of the application.