RESOURCE MANAGEMENT SYSTEM – PROJECT REQUIREMENTS (FULL DETAILED SPECIFICATION)

1. PROJECT OVERVIEW

An IT company wants to track:

- Employees (developers, testers, managers)
- Their skills
- Their allocation to projects
- Bench availability (unassigned employees)
- Project capacity vs assigned headcount
- Resource utilization reports

This system must allow HR and PMOs to assign and track people efficiently.

2. ACTORS

Admin (HR / PMO)

- Manages employees
- Manages projects
- Assigns employees to projects

System (Backend)

- Validates seat availability
- Prevents duplicate assignment
- Calculates bench employees
- Generates utilization reports

3. CORE MODULES & DETAILED REQUIREMENTS

MODULE 1 – Employee Management

Functional Requirements

Admin should be able to:

1. Add new employees
2. Edit employee details

3. View list of all employees
4. Activate/Deactivate employee (IsActive)
5. Manage employee skills

Employee Fields

- EmployeeID (INT, Identity, PK)
- FullName
- Email (unique)
- Phone
- Gender (check constraint: M, F, O)
- DateOfJoining (default = GETDATE())
- Designation (Developer, Tester, PM, etc.)
- ExperienceInYears
- IsActive (BIT, default = 1)

SQL Requirements

- **Identity column** for EmployeeID
- **Unique key** on Email
- **Default constraints** on DateOfJoining & IsActive
- **Check constraint** on Gender & Experience (>=0)
- **Indexes** on Email, Designation

Skills Table

- SkillID (Identity)
- SkillName (Unique)
- Description

EmployeeSkill Mapping Table

- EmployeeID, SkillID
- Use **composite primary key**
- Use a **foreign key with cascading delete**

MODULE 2 – Project Management

Functional Requirements

Admin should be able to:

1. Create a new project
2. Edit project details
3. View active/inactive projects
4. Set project capacity
5. Set start and end dates

Project Fields

- ProjectID (Identity PK)
- ProjectName (Unique)
- ClientName
- Capacity (number of employees allowed)
- StartDate
- EndDate
- ProjectStatus (Active/Inactive)

SQL Requirements

- Unique constraint on ProjectName
- Check: StartDate < EndDate
- Check: Capacity > 0
- Add **index** on ProjectStatus

MODULE 3 – Resource Allocation (Employees → Projects)

Functional Requirements

Admin can assign employees to projects:

- Fields:
    - EmployeeID
    - ProjectID
    - AllocationStartDate
    - AllocationEndDate

- AllocationPercentage (default 100%)

Business Rules

1. An employee **cannot be assigned twice** to the same project.
2. An employee **cannot be assigned to multiple projects** overlapping dates.
3. A project cannot exceed its **capacity**.
4. AllocationEndDate must be ≥ AllocationStartDate.

SQL Logic Required

Use a **Stored Procedure** to assign employees:

Must include:

- **Transaction (BEGIN TRAN → COMMIT/ROLLBACK)**
- **TRY/CATCH error handling**
- **Output parameter** to return messages like:
  - 0 = success
  - 1 = employee already assigned
  - 2 = project capacity full
  - 3 = overlapping assignment
  - 4 = invalid dates

Sample SQL Features Used

- JOIN & **Self JOIN** to detect overlapping allocations
- COALESCE to handle null end dates
- **Temp tables** for quick reporting
- **Views** for project utilization summary

MODULE 4 – Bench Management

Definition:

Employees who are active but **not assigned** to any project.

Functional Requirements

System should calculate:

- List of bench employees

- % of workforce on bench
- Skills of bench employees

SQL Queries Required

- Use **LEFT JOIN** between Employees and Allocations
- WHERE Allocation is NULL = Bench
- Use **COALESCE** to replace nulls in experience/skills
- Use GROUP BY to show:
    - Count of employees by designation
    - Skill-wise bench list

MODULE 5 – Reports & Analytics

Reports Required

1. Project Utilization Report

In Reports Display Bar graphs and Pie charts

Show:

- Project name
- Capacity
- Assigned count
- Available slots
- List of assigned employees

Use:

- **Aggregation (GROUP BY)**

2. Skill Availability Report

- Skill → Count of available employees
- Skill → Count of employees currently allocated
- Use **JOIN**, GROUP BY, COALESCE

3. Employee Allocation History

- For a given employee, show all past assignments
- Use ORDER BY Date

4. Project Overlap Check (Advanced SQL)

- Use **Self-join** to detect employees assigned to multiple projects at same time

---

4. API REQUIREMENTS (ADO.NET Web API)

Controllers Required

1. EmployeeController
2. SkillController
3. ProjectController
4. AllocationController
5. ReportsController

API Endpoints

- POST /employees
- POST /employees/{id}
- POST /employees
- POST /employees/{id}
- POST /employees/{id} (soft delete)
- POST /allocations (calls stored proc)
- POST /reports/bench
- POST /reports/project-utilization
- POST /reports/skills

Backend Coding Requirements

- Use **Repository Pattern**
- Use **Dependency Injection (AddScoped)**
- Use **Error Handling Middleware**
- Use **Async/Await** for database calls
- Use ADO.NET: SqlConnection, SqlCommand, SqlDataReader

---

5. FRONTEND REQUIREMENTS (HTML, CSS, JavaScript, jQuery)

Pages Required

1. Employee Management Page
2. Project Management Page
3. Skill Management Page
4. Allocation Page
5. Reports Page

UI Behaviors

- jQuery form validation
- AJAX calls to Web API for CRUD operations
- Dynamic tables updated with retrieved data
- Dropdowns for employee, project, skill selection
- Use jQuery events: click, change, submit
- Use jQuery animations for success/error messages

---

6. PYTHON BONUS MODULE

Add a Python Flask microservice for:

Bench Prediction API (Optional)

Given employee skill + experience → Predict bench probability
(Not AI-based—just a rule: If experience < 2yrs & skill rare → lower bench probability.)

Use:

- Flask route
- JSON input/output