# System Design Documentation

## Management Webpage API

### Overview

System Objective: Provide functionality for managing employees, projects, and associated technologies.

**Key Features:**

* Full CRUD operations for Employees, Projects, and Technologies
* Employee data import via spreadsheet (CSV/XLSX)
* Model associations with validation logic
* JSON-based API for frontend integration (SPA or mobile)

### Data Modeling

#### Entities & Relationships

Project

* has\_many :employees — A Project can have multiple Employees.
* has\_and\_belongs\_to\_many :technologies — Represents the required skills for the project.

Employee

* belongs\_to :project — Each Employee belongs to a single Project.
* has\_and\_belongs\_to\_many :technologies — Represents the skills the employee has.
* Business Rule: An Employee can only be assigned to a Project if they share at least one Technology with it. A class method enforces this constraint.

Technology

* has\_and\_belongs\_to\_many :projects
* has\_and\_belongs\_to\_many :employees

**Association Rules Summary:**

|  |  |
| --- | --- |
| Relationship | Description |
| Project ↔ Employees | One-to-many. An employee is linked to only one project. |
| Project ↔ Technologies | Many-to-many. Projects define required skills. |
| Employee ↔ Project | Conditional: only allowed if a shared Technology exists. |
| Employee ↔ Technology | Many-to-many. Defines employee expertise. |

### Employee Import Process

* Endpoint: POST /employees/import  
  - Accepts: file (multipart-form with 3 columns: user\_name, project, technology\_name)  
  - Example file: spec/fixtures/files/employees.xlsx
* Temporary Storage  
  - File is saved to tmp/uploads/  
  - Prevents processing of incomplete or corrupted files for security/integrity.
* Async Job Execution (Employees::ImportFileJob)  
  - Reads file using Roo::Spreadsheet  
  - Maps headers to model attributes  
  - For each row:  
   - Creates an Employee  
   - Splits technologies  
   - Uses find\_or\_create\_by  
   - Associates technologies with employee  
   - Validates compatibility with project
* Cleanup: Deletes the file after processing
* Response: Returns HTTP 202 Accepted

### Architecture

Architecture Style: Monolithic — all logic resides within a single Ruby on Rails application.

Pattern: MVC (Model-View-Controller)

* Model: core business logic and relationships
* Controller: HTTP handling and coordination
* View: not applicable (API-only)

**Technologies Used:**

* Ruby 3.1.3, Rails 6.1.7
* SQLite3 (dev/test), PostgreSQL (prod)
* Puma web server
* Sidekiq + Redis for background processing
* Foreman for process management
* rack-cors for CORS handling
* active\_model\_serializers for serialization

Diagrams: Architecture diagram available on draw.io / Miro

### Database Schema

**Entity-Relationship Tables:**

|  |  |
| --- | --- |
| Table | Fields |
| employees | id, user\_name, project\_id |
| projects | id, title |
| technologies | id, name |
| employees\_technologies | employee\_id, technology\_id |
| projects\_technologies | project\_id, technology\_id |

**Indexes:**

* index\_employees\_on\_project\_id
* Join table indexes: employee\_id, project\_id, technology\_id

### TODO

* Update import header keys: rename user\_name to "User name" and adjust header mapping.
* Integrate with frontend: GitHub - Management Webpage Frontend.
* Implement frontend logic to support employee file import.