* **Import en masse**

Build a web-based interface using React (frontend) and Java (Spring Boot or similar) as backend for importing users via CSV files.

* **Features:**

1. **Tabbed UI** with two tabs:
   * **Student Migration**
   * **Employee Migration**
2. In the **Student Migration** tab:
   * A file input (type="file") that accepts only .csv.
   * A styled button labeled “Upload File”.
   * A right-hand instruction panel that describes the required CSV format:
     + Columns: **Last Name**, **First Name**, **Gender**, **Birth Date (format: dd-mm-yyyy)**, **Code**.
   * Show localized instructions and labels based on language selection.
3. **CSV validation logic** (React-side):
   * Validate that uploaded file is a CSV.
   * Check header row matches expected column names.
   * Ensure required fields are present and non-empty.
   * Validate that date format is dd-mm-yyyy (22-12-2014).
   * Show error messages if any row fails validation.
4. **Localization (i18n)**:
   * Support **English**, **French**, and **Haitian Creole**.
   * Use React's i18n or ngx-translate for the frontend.
   * Backend messages (e.g., validation errors) should also be localizable.
   * Allow language switching via a dropdown or button group.
5. **Backend (Java)**:
   * Accept and process uploaded CSV.
   * Return validation errors or success message.
   * Optionally: return number of valid/invalid rows after processing.
6. **UI/UX styling**:
   * Clean and responsive design using React Material

**Gestion académique : Classes, Salles, Grille des Appréciations, Cours**

* **Classe**

Build a web-based Class Management Module using React (frontend) and Java (Spring Boot) backend.

### 1. Class Creation Form

Create a form with the following fields:

* **Class Label** – required text field (e.g., Sixième Année)
* **Class Short Name** – required text field (e.g., 6ème AF)
* **Previous Class**– dropdown populated from existing class labels, with a default option like “PAS DE CLASSE PRÉCÉDENTE”
* **Cycle** – dropdown (e.g.,1er Cycle, 2ème Cycle, 3ème Cycle)
* **Level** – dropdown (e.g., Fondamental, Secondaire)
* **Passing Grade** – number input (e.g., 50 or 60)
* Default display should resemble the following image structure (Include the passing grade field)

A white surface with a blue background

AI-generated content may be incorrect.

**Additional requirements:**

* Validate required fields.
* Prevent duplicate class names or short names.
* Allow creating and editing classes.
* Clear form after successful save.

### 2. Class Table Display

Below the form, display all classes in a paginated, sortable, and filterable table with the following columns:

* **Class Label**
* **Class Short Name**
* **Previous Class**
* **Cycle**
* **Level**
* **Passing Grade**
* **Actions**: Add, Edit, Delete

Table features:

* Sortable by any column.
* Filterable by Level, Cycle, or Class Label (optional).
* Pagination (e.g., 10–50 rows per page).
* Show total number of classes and range (e.g., “Afficher classe 1 à 9 (total de 9)”).
* Default display should resemble the following image structure.

A screenshot of a computer

AI-generated content may be incorrect.

### 3. Data Integration (Java Backend)

Create a REST API with the following endpoints:

* GET /api/classes – return list of all classes
* POST /api/classes – create a new class
* PUT /api/classes/{id} – update an existing class
* DELETE /api/classes/{id} – delete a class
* Validate that short name and label are unique

**Additional requirements**

* Confirmation dialog before delete
* Export class list to CSV
* Bulk edit/delete
* Audit trail or class creation history
* **Salles**

Create a web-based Room Management module using React (frontend) and Java (Spring Boot) backend.

**1. Room Creation Form:**  
Create a form with the following fields:

* Room Label\* – required text field (e.g., "Septième Année A")
* Room Short Name\* – required text field (e.g., "7e AF A")
* Associated Class – dropdown populated from existing classes
* Vacation dropdown (e.g., "Liaxin", "Seqdemo", "Nadismo")

A white surface with a blue background

AI-generated content may be incorrect.

Additional requirements:

* Validate required fields (marked with \*)
* Prevent duplicate room names or short names
* Allow creating and editing rooms
* Clear form after successful save

**2. Room Table Display:**  
Display all rooms in a paginated, sortable, and filterable table with columns:

* Room Label
* Room Short Name
* Associated Class
* Vacation
* Actions: Add, Edit, Delete

Table features:

* Sortable by any column
* Filterable by Vacation or Class
* Pagination (10-50 rows per page)
* Show total count (e.g., "Showing 1 to 9 of 9 rooms")

A screenshot of a computer

AI-generated content may be incorrect.

**3. Data Integration (Java Backend)**  
Create REST API with these endpoints:

* GET /api/rooms – list all rooms
* POST /api/rooms – create new room
* PUT /api/rooms/{id} – update room
* DELETE /api/rooms/{id} – delete room
* Room model:

class Room {

Long id;

String label;

String shortName;

String associatedClass;

String vacation;

}

**4. Localization (i18n)**  
Full i18n support for English, French, and Haitian Creole  
Translate all labels, buttons, and messages  
Use ngx-translate or React i18n  
Include language switcher

**5. UI/UX Requirements**

* Use React Material components
* Responsive design
* Clear visual distinction for required fields
* Action buttons (Save, Cancel, Add New) matching the reference design

Optional Enhancements:

* Confirmation dialog before delete
* Export to CSV functionality
* Bulk operations
* Room assignment history
* **Grading scheme**

Create a web-based grading scheme management system with React frontend and Spring Boot backend.

1. Form Structure

**Level Selection Section**

Dropdown with:

* + "Fondamental"
  + "Secondaire"
  + "Préscolaire"

For each level, provide:

* Starting Value (0-100)
* Ending Value (0-100)
* Comment (e.g., "Très Bien")
* Short Code (e.g., "TB")

**A screenshot of a computer

AI-generated content may be incorrect.**

2. Data Display Table

Create a sortable table showing:

**A screenshot of a white screen

AI-generated content may be incorrect.**

3. Functional Requirements

* **CRUD Operations**:
  + Add new grade ranges
  + Edit existing entries
  + Delete entries
  + Bulk operations
* **Validation**:
  + Ensure no overlapping ranges
  + Validate score boundaries (0-100)
  + Prevent duplicate comments/codes
* **Buttons**:
  + Save
  + Cancel
  + Edit

4. Technical Specifications

**Frontend (React)**:

* + Use React Material components
  + Reactive forms with validation
  + Responsive layout

**Backend (Spring Boot):**

@Entity

class GradingScheme {

Long id;

String level;

Integer minScore;

Integer maxScore;

String comment;

String code;

Integer sq;

}

Endpoints:

* GET /api/grading-schemes
* POST /api/grading-schemes
* PUT /api/grading-schemes/{id}
* DELETE /api/grading-schemes/{id}

i18n Support:

* French (default)
* English
* Haitian Creole

5. UI/UX Requirements

* Clear visual hierarchy
* Highlight required fields
* Confirmation dialogs for deletions
* Success/error notifications

Additional requirement:

* Export to Excel/CSV
* Level-based filtering
* Historical versioning
* **Cours**

Create a web-based system for managing academic courses with React frontend and Spring Boot backend.

**1. Course Creation Form**

**Single Course Mode:**

* Room selection dropdown ("Huitième Année A", etc.)
* Dynamic form fields:
  + Subject dropdown ("Choisir matière svp")
  + Teacher dropdown ("Choisir un professeur svp")
  + Academic period (default: "2024-2025")
  + Coefficient (numeric input)
  + Exam coefficient (numeric input)
  + Core course checkbox: ☐ Cours de base

**Bulk Creation Mode:**

* CSV/Excel import capability
* Template download button
* Validation preview before import

**A screenshot of a computer

AI-generated content may be incorrect.**

**2. Course Table Display**

Display courses with:

* Pagination (25+ items per page)
* Sorting by all columns
* Filtering by:
  + Subject
  + Teacher
  + Room
  + Course type

Columns:  
| Subject | Coefficient | Passing Grade | Room | Core Course | Teacher | Actions |

**A screenshot of a computer

AI-generated content may be incorrect.**

**3. Functional Requirements**

CRUD Operations:

* Create (single/bulk)
* Edit existing courses
* Delete (with confirmation)
* Search functionality

Validation Rules:

* Prevent duplicate course-room assignments
* Validate coefficient ranges
* Require all dropdown selections

**4. Technical Specifications**

Frontend (React):

* React Material components
* Reactive forms with validation
* ExcelJS for bulk operations
* Responsive layout

**Backend (Spring Boot):**

@Entity

class Course {

Long id;

String subject;

String teacher;

String academicPeriod;

Integer coefficient;

Integer examCoefficient;

String room;

Boolean isCore;

Integer passingGrade;

}

Endpoints:

* GET /api/courses (with pagination)
* POST /api/courses (single)
* POST /api/courses/bulk (batch)
* PUT /api/courses/{id}
* DELETE /api/courses/{id}

**5. UI Components**

* Action Buttons:
* "Enregistrer" (Save)
* "Annuler" (Cancel)
* "Ajouter Cours" (Add Course)
* "Création par lot" (Bulk Create)
* "Retour" (Back)
* "Excel" (Export)

Display Features:

* "Showing 1 to 25 of number of pages" counter
* Pagination controls
* Visual indicator for core courses

6. i18n Support

* French (default)
* English
* Haitian Creole
* Additional Features:
* Teacher assignment validation
* Academic period management
* Course prerequisites system
* Schedule conflict detection