Assignment Title

“Patient Diagnostic Report Upload with Cascading Medical Selection”

Scenario

A clinic staff member registers a patient’s diagnostic document (Lab report / Scan / Prescription). While submitting, they must choose medical categories using **cascading dropdowns** and upload the report file. The system stores the metadata + file and shows a list of submissions.

Requirements (Must Do)

1) Cascading Dropdowns (3-level)

Create a form with these dropdowns:

1. **Department** (e.g., Cardiology, Neurology, Orthopedics, Radiology, Pathology)
2. **Test Category** (depends on Department)
3. **Test Name** (depends on Test Category)

Behavior

- Initially only **Department** is enabled.
- Selecting **Department** loads **Test Categories** (via API/AJAX).
- Selecting **Test Category** loads **Test Names** (via API/AJAX).
- Changing a parent dropdown resets the children (clear + disable until reloaded).

Example dataset (you can hardcode or seed)

- Cardiology → ECG Tests → ECG, TMT
- Cardiology → Blood Tests → Troponin, Lipid Profile
- Neurology → Imaging → MRI Brain, CT Brain
- Orthopedics → Imaging → X-Ray Knee, X-Ray Spine
- Pathology → Blood Tests → CBC, ESR
- Radiology → Ultrasound → USG Abdomen, USG Pelvis

2) Patient Report Upload Form

Form fields:

- Patient Name (text, required)
- Age (number, required)
- Mobile/Email (optional)
- Cascading dropdowns (required)
- Report Date (required)
- **File Upload** (required)

File upload rules

- Allowed types: **PDF, JPG, PNG**
- Max size: **5 MB**
- Save the uploaded file to:
    - Local server folder (e.g., /uploads) **OR**
    - Database as blob (optional, only if you want)
- Store in DB: patient details + selected dropdown values + file name/path + uploaded timestamp.

3) Display Screen

Create a page/table that lists uploaded reports with:

- Patient Name, Age
- Department, Category, Test Name
- Report Date
- File name
- Action: **View/Download** (open file)

API Endpoints (Common across .NET / Java / PHP)

Implement these endpoints (REST style):

1. GET /api/departments → list departments
2. GET /api/categories?departmentId=__ → list categories for department
3. GET /api/tests?categoryId=__ → list test names for category
4. POST /api/reports → submit form + file upload (multipart/form-data)
5. GET /api/reports → list uploaded reports
6. GET /api/reports/{id}/download → download/view file

Validation & Error Handling

- Required field validation (backend + basic frontend).
- Show meaningful errors:
    - “Invalid file type”, “File too large”, “Department not selected”, etc.
- If file upload fails, do not save incomplete record.

Deliverables (What candidate submits)

1. Source code project
2. DB script / migration (or seeded JSON data)
3. README with:
    - Setup steps
    - API list
    - Assumptions
    - Sample request/response (optional)

Evaluation Criteria (Quick Rubric)

- Cascading dropdown works correctly (reset + dynamic load)
- Clean API design + correct filtering logic
- Upload validation (type/size) + safe storage
- Report list + download works
- Code structure + naming + basic security hygiene

Optional (Only if time permits)

- Search/filter by patient name / department
- Pagination
- Store file checksum / unique naming
- Simple login (admin)