NHPC INTERNSHIP PROJECT

🔷 **Project: Instructor Training Portal**

💡 **Key Use Cases**

- New instructors can register → get unique login ID.
- Instructors log in → view dashboard.
- View list of **current trainees** with training duration.
- **View trainee history** (past trainees).
- **Mark daily attendance**.
- Click a trainee to see:
  - Assigned **Project ID & Details**
  - Attendance record
  - **Remaining days** in training

✅ **Refined Functional Blueprint**

🔐 **1. Instructor Registration & Login**

🔷 **Register**

- Form with name, email, password
- On submit → generate a unique ID (e.g., `INS2025_001`)
- Save to DB

🔷 **Login**

- Instructor enters unique ID + password
- Authenticated with session or JWT

- Redirect to dashboard

🏠 2. Instructor Dashboard

Section	Description
👤 Current Trainees	List with name, training start/end, days remaining
📜 Trainee History	Past trainees with end date and rating
📅 Attendance Tracker	Link to mark today's attendance
🔍 View Trainee Profile	Click on trainee to open full info (project + logs)

🧾 3. Current Trainees Page

Column	Details
Name	Trainee full name
Training Period	Start date → End date
Days Left	`(end_date - today)`
Actions	View Profile / Mark Attendance

🔁 4. Mark Attendance (Daily)

- A list of all current trainees
- Each row has
  - Trainee id
  - Today's date
  - Dropdown: Present / Absent / Leave
- On submit → POST to `/attendance`

📁 5. Trainee Profile Page

Accessed when instructor clicks on a trainee

Section	Info Shown
👨‍🎓 Name & Email	Basic details
📅 Training Period	Start date, End date, Days remaining
🆔 Project Info	Project ID, Title, Description
📊 Attendance	Calendar/table showing P/A/L per date

✅ 1. Frontend – Add Trainee Page

- Instructor clicks **"Add New Trainee"** on dashboard
- A form appears with:
  - Trainee Name
  - Training Start Date
  - Training End Date
- Instructor fills the form and submits

✅ 2. Backend – Handling Registration

After form submission:

1. Backend **receives form data** (name, start date, end date)
2. It checks the database for the **last trainee ID** registered (e.g., `TRN004`)
3. It generates the **next unique trainee ID** (e.g., `TRN005`)
4. It stores the new trainee in the database along with:
   - Generated `trainee_id`
   - Associated `instructor_id` (who registered this trainee)
   - Start and end date
5. Returns a success message to the frontend

✅ 3. Database Design – Trainees Table

Trainee table should include:

- Auto-increment `id`
- Unique `trainee_id` (e.g., `TRN003`)
- `name`
- `start_date`
- `end_date`
- `instructor_id` (foreign key to instructors)

✅ 4. Post-registration Behavior

After successful trainee registration:

- The instructor is shown a **confirmation message**
- Optionally, the list of current trainees is **refreshed** to show the newly added one

📦 Optional Enhancements

- Check for **duplicate trainee names + dates** under the same instructor
- Allow updating trainee details later
- Automatically calculate **training duration**
- Auto-show **countdown of days left** (based on end date)

🔁 Recap: Key Responsibilities

Component	Role
HTML Form	Collects trainee name, start date, end date
JS Script	Submits form to backend using fetch() or POST form
Express.js	Receives data, generates trainee ID, saves to database
MySQL	Stores trainee info with unique ID and instructor ID

Response	Sends back success/failure message

📚 6. Past Trainees (History)

- List all past trainees with:
  - Name
  - Training duration
  - Final rating (1-5 stars)
  - Feedback (optional)

🧱 Backend (Express + Sequelize + MySQL)

📂 Tables:

`instructors`

- `id`, `name`, `email`, `password`, `unique_login_id`

`trainees`

- `id`, `name`, `email`, `start_date`, `end_date`, `instructor_id`, `project_id`

`projects`

- project_`id`, `title`, `description`

`attendance`

- `id`, `trainee_id`, `date`, `status`

`ratings`

- `id`, `trainee_id`, `rating`, `feedback`

🧠 Calculations

🕒 Remaining Days of Training

```
const remainingDays = Math.ceil(
  (new Date(trainee.end_date) - new Date()) / (1000 * 60
* 60 * 24)
);
```

✅ Pages and Routing Summary

Page	Route	Method	Description
Register Instructor	`/register`	POST	Signup new instructor
Login Page	`/login`	POST	Login and generate token/ session
Dashboard Page	`/dashboard`	GET	Show overview
Current Trainees	`/trainees/current`	GET	List trainees with time left
Past Trainees	`/trainees/history`	GET	Show history and ratings
Mark Attendance	`/attendance`	POST	Submit attendance records
Get Attendance by Trainee	`/attendance/:id`	GET	For profile view
Trainee Profile	`/trainee/:id`	GET	Show trainee + project + logs

🔷 Big Picture: How the Project Runs

Your project is a **full-stack web application**. That means it has:

1. **Frontend** – What users see and interact with (HTML, CSS, JS)
2. **Backend** – The server logic that handles requests, processes data, and interacts with the database (Node.js + Express)
3. **Database** – Stores data persistently (like instructors, trainees, attendance)
4. **Middleware/API** – The communication bridge between frontend and backend

🔁 Flow of the Project

Let's break it into **phases** of how it works from the moment someone visits your portal.

🔷 1. Instructor Registration & Login

🔶 **Frontend:**

- A registration page where a **new instructor** enters name, email, and password.
- A login page where instructor enters their **unique login ID** and password.

🔶 **Backend:**

- Checks if instructor exists.
- Generates and stores a unique login ID (e.g., `INS2025_003`).
- Verifies credentials when logging in.

🔶 **Middleware/API:**

- Handles form submissions and sends the data to backend.
- If login is successful, backend sends a **session ID** or **JWT token**.
- Frontend stores it (in cookies or localStorage) to keep the user logged in.

🔷 2. Instructor Dashboard

After login, instructor sees a dashboard.

🔶 **Frontend:**

- Clean page with buttons/links:
  - Register new Trainees
  - View Current Trainees
  - Mark Attendance

    - View History
    - Give Ratings

🔶 **Backend:**

- Fetches instructor's data and related trainees from the database.

🔶 **Middleware/API:**

- Makes GET requests like:
    - `/trainees/current`
    - `/trainees/history`

🔷 **3. Current Trainees List**

🔶 **Frontend:**

- Displays a table or list:
    - Trainee Name
    - Training Start Date
    - End Date
    - Days Remaining
    - "View Profile" button

🔶 **Backend:**

- Queries the database for:
    - All trainees where instructor_id = logged-in instructor's ID
    - Filters based on `end_date >= today` (current trainees)

🔶 **Middleware/API:**

- Sends GET request to `/trainees/current`

- Receives JSON data and displays it dynamically

🔷 4. Mark Attendance (Daily)

🔶 **Frontend:**

- A page shows all trainees for today
- Dropdowns or buttons to mark Present / Absent / Leave
- "Submit" button

🔶 **Backend:**

- Receives list of attendance entries
- Saves them in `attendance` table with `trainee_id`, `date`, and `status`

🔶 **Middleware/API:**

- Sends POST request to `/attendance` with JSON body like:
  `json`

  `[`
- `  { "trainee_id": 1, "date": "2025-06-11", "status": "Present" },`
- `  { "trainee_id": 2, "date": "2025-06-11", "status": "Absent" }`
-

🔷 5. View Trainee Profile

When instructor clicks on a trainee:

🔶 **Frontend:**

- New page opens showing:
  - Trainee details
  - Project details (ID, description)
  - Attendance history (P/A/L chart or table)

    - Days remaining in training

🔶 **Backend:**

- Fetches trainee info from `trainees` table
- Joins with `projects` and `attendance` tables
- Calculates training days remaining

🔶 **Middleware/API:**

- Sends GET request to `/trainee/:id`
- Receives a structured JSON with everything needed to show

🔷 **6. Past Trainees & Ratings**

After a trainee completes training:

🔶 **Frontend:**

- History page lists all past trainees
- Instructor can select a trainee and submit a rating & optional feedback

🔶 **Backend:**

- Fetches all trainees with `end_date < today`
- Stores ratings in a `ratings` table

🔶 **Middleware/API:**

- GET `/trainees/history` → shows past list
- POST `/ratings` → saves rating for a trainee

🔄 How Everything Connects

Here's a simple **diagram-style explanation**:

```
User (Instructor)
    |
    v
Frontend (HTML, CSS, JS)
    |
    v
API Requests (Fetch/Axios)
    |
    v
Backend (Node.js + Express)
    |
    v
Middleware (Authentication, Validation, Logging)
    |
    v
Database (MySQL/PostgreSQL via Sequelize)
```

Each layer plays its role:

- **Frontend**: Looks good and collects input
- **API**: Sends/receives data between frontend & backend
- **Backend**: Processes requests, enforces logic
- **Database**: Stores everything permanently

✅ Recap of Features & Responsibilities

Feature	Frontend	Backend	Database
Login/Register	Forms + Redirects	Validate + Save/Retrieve	Store user info
Dashboard	Buttons + Layout	Fetch summary data	
View Trainees	Table/List	Query current trainees	trainees table
Mark Attendance	Dropdowns + Submit	Save attendance for today	attendance table
View Profile	Info display	Join project + attendance	multiple tables
Trainee History	List past trainees	Filter based on date	trainees + ratings

Rating Trainees	Star input + feedback	Save rating and feedback	ratings table

🔧 PROJECT ARCHITECTURE USING ONLY:

MySQL + Node.js (Express) + HTML/CSS/JS

🔁 How Everything Connects

```
HTML/CSS/JS  <--->  Express.js Server  <--->  MySQL Database
(Frontend)          (Backend)                 (Data Storage)
```

💡 Explanation: Layer by Layer

1. Frontend (HTML, CSS, JS)

What It Does:

- Shows all web pages: login, register, dashboard, trainee list, attendance, etc.
- Sends form data to the backend (e.g., login info, attendance marks).
- Uses **vanilla JavaScript** to fetch or submit data (via `fetch()` or form POST).
- Gets back responses (like JSON or redirects) and updates the view accordingly.

Pages You Will Have:

- `index.html` → Login page
- `register.html` → Instructor signup
- `dashboard.html` → Main menu after login & List of current trainees
- `attendance.html` → Mark attendance form
- `traineeProfile.html` → Individual trainee details
- `history.html` → Past trainees + rating page

2. Backend (Node.js + Express)

What It Does:

- Creates a web server and handles routes (like `/login`, `/attendance`, `/trainees`)
- Accepts data from frontend via `POST`, `GET` routes
- Talks to the MySQL database (via raw SQL or with query builder like Sequelize or mysql2)
- Sends back responses to frontend

Example Routes:

- `POST /register` → Save instructor to DB
- `POST /login` → Check instructor credentials
- `GET /trainees/current` → Send all current trainees
- `POST /attendance` → Store attendance
- `GET /trainee/:id` → Send trainee details + attendance

3. Database (MySQL)

What It Does:

- Stores all persistent data:
  - Instructor info (login ID, name, password)
  - Trainees info
  - Attendance entries
  - Project details
  - Ratings/history

Example Tables:

- `instructors (id, name, login_id, password)`
- `trainees (id, name, start_date, end_date, instructor_id)`
- `attendance (id, trainee_id, date, status)`
- `projects (id, title, description, trainee_id)`
- `ratings (id, trainee_id, rating, feedback)`

🧠 How a Typical Action Works

Example: Instructor Logs In

1. Instructor opens `index.html`
2. Fills form and clicks "Login"
3. JS sends form data → `POST /login`
4. Express receives request
5. It queries MySQL to match login ID + password
6. If matched, send back success → redirect to dashboard.html
7. Dashboard uses JS to load trainee data via `GET /trainees/current`

📁 Folder Structure (Simple & Clean)

```
InstructorPortal/
├── public/
│   ├── css/
│   │   └── styles.css
│   ├── js/
│   │   └── scripts.js
│   ├── html/
│   │   ├── index.html
│   │   ├── register.html
│   │   ├── dashboard.html
│   │   └── attendance.html
```

```
├── server/
│   ├── app.js                # Main Express server
│   ├── db.js                 # MySQL connection config
│   ├── routes/
│   │   ├── auth.js           # Login/register routes
│   │   ├── trainees.js       # Get/add trainees
│   │   └── attendance.js     # Attendance routes
├── package.json
├── .env                     # DB credentials
```

🔐 Handling Login Without Frameworks

You'll use:

- **Sessions or tokens stored in JS (or cookies)**
- On login, backend sets a `loggedIn` variable in memory (or sends login ID back)
- JS will use this info to keep the user in session

✅ Summary: How It Runs (Your Tech Stack)

Feature	Handled By
UI Pages	HTML + CSS + JS
Button Clicks / Forms	JS fetch() / form POST to Express
Data Validation/Handling	Express.js server
Persistent Storage	MySQL
Login / Routing Protection	Express middleware or basic session

Training Logic	JS + server logic + MySQL joins

✅ You Are 100% Capable of Building This With:

- 💾 MySQL
- 🟢 Node.js + Express
- 👨‍🏫 HTML/CSS
- ⚙️ JavaScript (vanilla)