

# ELEWA TECHNICAL ASSESSMENT

Assessment Window: 14/11/2025 – 19/11/2025

## Employee Management System (EMS)

### Full-Stack Developer Assessment

Elewa Company operates 24 hours a day and requires a modern system to manage employee data efficiently, securely, and with high availability. As part of your assessment, you are required to design and implement a simplified version of an Employee Management System (EMS) using the MERN stack (MongoDB, Express, React, Node.js).

This assignment evaluates your ability to design systems, build scalable APIs, create functional UIs, and apply best practices in security, optimization, and modern full-stack development.

### • Assignment Overview

- Your task is to design and implement an EMS application that:
- Manages employee records
- Provides secure authentication
- Allows CRUD operations
- Supports search and pagination
- Has a clean and responsive UI
- Uses best practices in backend and frontend development
- The system should be functional, well-structured, and scalable.

### • **System Requirements (What You Must Build)**

#### • **A. Backend Requirements (Node.js + Express + MongoDB)**

##### • **1. Authentication & Security**

- You must implement:
- User **registration** and **login** endpoints
- Password hashing using **bcryptjs**
- JWT-based authentication
- Middleware to protect private routes
- **Explain or show:**
- How you secure routes
- How tokens are handled
- How passwords are protected

- **2. Employee Management API**

- You must create API endpoints that support:

- Add employee
- Get all employees
- Update employee
- Delete employee
- Search employees by:
  - Name
  - Email
  - Department
  - Role

- Pagination for large employee collections

- **Explain or show:**

- How your search works
- How pagination is implemented
- Your API route structure

- **3. Database Design (MongoDB + Mongoose)**

- Your database must include:

- **Employee model**

- Optional: **Department**, **Role** models

- Each employee should have fields such as:

- name
- email
- department
- role
- dateOfJoining
- createdAt / updatedAt

- **Explain or show:**

- Why you chose certain fields
- Any indexes used to optimize search

- **4. Environment Configuration**

- Your project should include a .env file storing:

- PORT
- MONGO\_URI
- JWT\_SECRET

- Do **not** upload your real secrets—use placeholders.

- **B. Frontend Requirements (React + TypeScript)**

- **1. Authentication UI**

- Build pages to:

- Login
- Register

- These must:

- Store JWT securely
- Block unauthorized access
- Redirect users appropriately

- **2. Employee Dashboard**

- Your dashboard must:
- Display a list of employees
- Allow adding new employees
- Allow editing employees
- Allow deleting employees
- Include search and pagination
- Be responsive on mobile & desktop

- **3. Components**

- At minimum, include:
- EmployeeList
- EmployeeForm
- DashboardStats (even if basic)

- **4. State Management**

- Use:
- AuthContext for authentication state
- EmployeeContext for managing employee data
- Custom hooks where helpful

- **C. API Integration**

- Using Axios, you must:
- Create a configured Axios instance
- Implement Axios interceptors for JWT authorization
- Handle API errors gracefully
- Show loading indicators during requests

- **D. Optimization Requirements**

- You should implement:
- Pagination (backend + frontend)
- Database indexes for fast searching
- Optimized React rendering
- Clean and reusable components

- **E. Testing & Deployment**

- **Testing (required)**

- Test all API endpoints using Postman
- Provide a Postman collection (optional but recommended)

- **Deployment (optional but extra points)**

- Deploy backend (Render/Heroku)
- Deploy frontend (Vercel/Netlify)

- **F. Optional Bonus Features**

- These are **not required**, but will earn bonus points:
- Role-Based Access Control (Admin, HR, Manager)
- Export employees as CSV or Excel
- Dashboard charts (e.g., using Recharts or Chart.js)
- Toast notifications (success/error)

- Dark mode

## • **Deliverables (What You Must Submit)**

- Please submit the following:
- **1. GitHub Repository Link**
- Containing:
- /backend folder
- /frontend folder
- README.md with instructions
- **2. Setup Instructions**
- How to run backend
- How to run frontend
- How to set up .env
- **3. Technical Explanation**
- A short document (in README or separate file) explaining:
- Your architecture
- API structure
- Database design
- Authentication logic
- Pagination & search approach
- **4. (Optional) Deployment Links**
- If you deploy, include URLs.

## • **Evaluation Criteria**

- Candidates will be assessed based on:
- ✓ Functionality
- Does the system work as required?
- ✓ Code Quality
- Is the code clean, modular, and readable?
- ✓ Architecture & Design
- Is the system well-structured and scalable?
- ✓ UI/UX
- Is the interface intuitive, responsive, and visually clean?
- ✓ Security Practices
- Are JWT, hashing, and protected routes implemented correctly?
- ✓ Problem-Solving Skills
- How well did you implement search, pagination, and integration?
- ✓ Documentation
- Is everything easy to set up and understand?
- ✓ Bonus Features (Optional)
- Any extra effort adds value.

## • **Submission Instructions**

- Submit your GitHub repository link

- Ensure both backend and frontend run without errors
- Include clear documentation
- Optional: Submit hosted/live links
- vents