EPI-USE Employee Hierarchy Platform

Overview

A cloud-hosted web application for managing EPI-USE Africa's employee hierarchy. The platform supports full CRUD for employee data, reporting structure management, visual org chart, Gravatar integration, profile picture upload, data export, and robust security. It is designed for scalability, maintainability, and a modern user experience.

Features

- **Cloud Hosted & Accessible via URL**: Deployed on Vercel and Railway.
- **Employee CRUD**: Create, read, update, delete employees with all required fields (name, surname, birth date, employee number, salary, role/position, manager).
- **Reporting Structure**: Assign managers, prevent self-management, support CEO with no manager.
- **Org Chart**: Visual tree/graph of hierarchy.
- **Employee Table**: Paginated, sortable, filterable.
- **Search**: Find employees by any field.
- **Profile Pictures**: Gravatar by default, optional upload.
- **Data Export**: CSV/JSON export for admins.
- **Role-Based Permissions**: Centralized logic for admin, manager, employee.
- **User Guide & Technical Document**: Included below.

Architecture

- **Frontend**: Next.js (React) SPA (`apps/frontend`)
- **Backend**: Node.js (Express) REST API (`apps/backend`)
- **Database**: PostgreSQL (Railway)
- **Deployment**: Vercel, Railway
- **Monorepo**: Shared types, utilities, and scripts

High-Level Diagram

Design Patterns & Technology Choices

- **Monorepo**: Consistency and maintainability
- **RESTful API**: Clean separation of concerns
- **Prisma ORM**: Type-safe DB access, migrations
- **React Query**: Efficient data fetching/caching
- **Tailwind CSS**: Rapid UI development
- **Docker**: Containerized deployment
- **Gravatar**: Standardized avatar solution
- **Multer**: Secure profile picture uploads

Security & Data Integrity

- **JWT Authentication**: Secure API endpoints
- **Password Reset**: Email-based reset flow
- **Duplicate Checks**: Backend enforces unique email/employee number
- **Validation**: Both frontend and backend schemas
- **Centralized Error Handling**: Middleware and UI propagation
- **Rate Limiting & Helmet**: Prevent abuse and secure HTTP headers

---

Why This Approach?

- **Scalability**: Cloud DB, stateless API, paginated frontend
- **Maintainability**: Monorepo, shared types, modular codebase
- **User Experience**: Fast, responsive UI
- **Extensibility**: Easy to add new features
- **Security**: Modern best practices
