Task: Build an HR Management Backend

Objective

Create a small HR management RESTful API. HR users can authenticate, then perform CRUD (Create, Read, Update, Delete) operations on employees. HR can record daily attendance. Provide a monthly report showing days present and late arrivals (late = check-in after 9:45 AM).

Requirements

1. Environment Setup

- Initialize a new **Node.js** project with **TypeScript with OOP.**
- Use **Express** as the web framework.
- Use **Knex** as the query builder.
- Use **PostgreSQL** as the database (preferred) or MySQL (if you choose to) But you must have a good knowledge about SQL.
- Use **Joi** (or Express Validator) for input validation.
- Configure **ESLint** and **Prettier** for code quality and formatting.
- Use a `.env` file for environment variables (e.g., DB credentials, JWT secret, upload path, server port) and put the file in gitignore, also create .env.example without gitignore.
- Use **Multer** (or similar) for handling photo uploads to local storage.

2. Database Schema

- Create a schema for the database with the following tables:
  - **hr_users**
    - `id` (primary key, auto-increment)
    - `email` (string, unique, required)
    - `password_hash` (string, required)
    - `name` (string, required)
    - `created_at` (timestamp, default now)
    - `updated_at` (timestamp, default now)
  - **employees**
    - `id` (primary key, auto-increment)
    - `name` (string, required)

        - `age` (integer, required)
        - `designation` (string, required)
        - `hiring_date` (date, required)
        - `date_of_birth` (date, required)
        - `salary` (numeric/decimal, required)
        - `photo_path` (string, optional) — local file path/filename
        - `created_at` (timestamp, default now)
        - `updated_at` (timestamp, default now)

    - **attendance**
        - `id` (primary key, auto-increment)
        - `employee_id` (foreign key → employees.id, required)
        - `date` (date, required)
        - `check_in_time` (time or timestamp, required)
        - **Unique constraint**: (`employee_id`, `date`) — ensures a single record per employee per day

Note: Use foreign keys properly.

3. Authentication

- **POST /auth/login**: HR login with email/password, returns JWT.
- Protect all `/employees` and `/attendance` routes with JWT middleware (Bearer token).

4. Endpoints

Auth

- `POST /auth/login` — HR user login.

Employees

- `GET /employees` — List all employees (with optional pagination & filters).
- `GET /employees/:id` — Get a single employee by ID.
- `POST /employees` — Create a new employee (supports multipart form-data for `photo`).
- `PUT /employees/:id` — Update employee details (also allow updating/replacing `photo`).
- `DELETE /employees/:id` — Delete an employee.

Attendance

- `GET /attendance` — List attendance entries (filterable by `employee_id`, `date`, range).
- `GET /attendance/:id` — Get a single attendance entry.
- `POST /attendance` — Create or **upsert** attendance:
  - Body: `employee_id`, `date`, `check_in_time`
  - If (`employee_id`, `date`) exists, **update** `check_in_time` instead of creating a duplicate.
- `PUT /attendance/:id` — Update an attendance entry.
- `DELETE /attendance/:id` — Delete an attendance entry.

Reports

- `GET /reports/attendance` — Monthly attendance summary.
  - Query: `month=YYYY-MM` (required), optional `employee_id`
  - Response per employee: `employee_id`, `name`, `days_present`, `times_late`
  - **Late rule**: `check_in_time > 09:45:00` counts as late.

Queries (Examples)

- `GET /employees?search=rahim` — Search by employee name (partial).
- `GET /attendance?employee_id=12&from=2025-08-01&to=2025-08-31`
- `GET /reports/attendance?month=2025-08`

5. TypeScript

- Type all request/response handlers.
- Must use all the payloads, request body and every method return type.
- Augment Express `Request` with `user` (decoded JWT payload) type.

6. Database Connection

- Use `pg` (PostgreSQL) or `mysql2` (if MySQL).
- Create a **Knex** connection pool for efficiency.
- Read DB credentials and pool settings from environment variables.
- Provide Knex migrations & seeds.

Bonus (Optional)

- **Pagination** for `GET /employees` and `GET /attendance`.
- **Search**: filter employees by `name` (ILIKE), filter attendance by date ranges.
- **Soft delete** for employees (with `deleted_at`) and exclude from lists.

Deliverables

- Source code in a public Git repository (e.g., GitHub).
- README file with instructions on how to set up and run the project.
- env.example File
- SQL script or migration files for setting up the database schema.