Employee Records Management Backend - Documentation

This documentation provides instructions for setting up, running, and testing the Employee Records Management Backend system built with Node.js, Express.js, and MongoDB.

System Overview

The Employee Records Management Backend is a RESTful API that allows for managing employee records. It supports the following operations:

- Adding new employees
- Retrieving all employees
- Retrieving specific employees by ID
- Updating employee details
- Deleting employee records

Prerequisites

Before you begin, ensure you have the following installed:

- Node.js (v14.0.0 or later)
- npm (v6.0.0 or later)
- MongoDB (v4.0.0 or later)

Installation

1. Navigate to the project directory in your terminal
2. Install the required dependencies:

npm install

This will install Express.js, Mongoose, dotenv, and other dependencies defined in the package.json file.

Configuration

1. Create a .env file in the root directory of the project
2. Add the following environment variables:

PORT=5000

MONGO_URI=mongodb://localhost:27017/employeedb

Running the Application

To start the server:

node server.js

If everything is configured correctly, you should see the following output:

Server running on port 5000

MongoDB Connected: localhost

Directory Structure

```
W20/
  ├── config/
  │   └── db.js
  ├── models/
  │   └── Employee.js
  ├── routes/
  │   └── employees.js
  ├── .env        <-- Make sure this exists
  ├── package.json
  └── server.js
```

Endpoints

1. Add a New Employee

- **URL**: http://localhost:5000/api/employees
- **Method**: POST
- **Request Body**:
- { "name": "John Doe", "department": "Engineering", "designation": "Software Developer", "salary": 75000, "joiningDate": "2023-04-15"}
- **Success Response**: Status 200 OK
- { "name": "John Doe", "department": "Engineering", "designation": "Software Developer", "salary": 75000, "joiningDate": "2023-04-15T00:00:00.000Z", "_id": "60d21b4667d0d8992e610c85", "createdAt": "2023-04-28T10:15:30.000Z"}

2. Get All Employees

- **URL**: http://localhost:5000/api/employees
- **Method**: GET
- **Success Response**: Status 200 OK
- [{ "_id": "60d21b4667d0d8992e610c85", "name": "John Doe", "department": "Engineering", "designation": "Software Developer", "salary": 75000, "joiningDate": "2023-04-15T00:00:00.000Z", "createdAt": "2023-04-28T10:15:30.000Z" }, { "_id": "60d21b4667d0d8992e610c86", "name": "Jane Smith", "department": "Marketing", "designation": "Marketing Manager", "salary": 85000, "joiningDate": "2023-03-10T00:00:00.000Z", "createdAt": "2023-04-28T10:17:30.000Z" }]

3. Get Employee by ID

- **URL**: http://localhost:5000/api/employees/:id
- **Method**: GET

- **Success Response**: Status 200 OK
- { "_id": "60d21b4667d0d8992e610c85", "name": "John Doe", "department": "Engineering", "designation": "Software Developer", "salary": 75000, "joiningDate": "2023-04-15T00:00:00.000Z", "createdAt": "2023-04-28T10:15:30.000Z"}
- **Error Response**: Status 404 Not Found
- { "msg": "Employee not found"}

4. Update Employee

- **URL**: http://localhost:5000/api/employees/:id
- **Method**: PUT
- **Request Body**: (Include only fields to update)
- { "department": "Research & Development", "salary": 80000}
- **Success Response**: Status 200 OK
- { "_id": "60d21b4667d0d8992e610c85", "name": "John Doe", "department": "Research & Development", "designation": "Software Developer", "salary": 80000, "joiningDate": "2023-04-15T00:00:00.000Z", "createdAt": "2023-04-28T10:15:30.000Z"}
- **Error Response**: Status 404 Not Found
- { "msg": "Employee not found"}

5. Delete Employee

- **URL**: http://localhost:5000/api/employees/:id
- **Method**: DELETE
- **Success Response**: Status 200 OK
- { "msg": "Employee removed"}
- **Error Response**: Status 404 Not Found
- { "msg": "Employee not found"}

Testing

You can test the API using various tools such as Postman, cURL, or any HTTP client. Below are examples for using Postman.

Using Postman

1. **Setup Postman**:
    - Download and install Postman from https://www.postman.com/downloads/
    - Create a new collection for "Employee Management API"
2. **Testing Add a New Employee**:
    - Method: POST

- URL: http://localhost:5000/api/employees
- Headers: Content-Type: application/json
- Body (raw JSON):
- { "name": "John Doe", "department": "Engineering", "designation": "Software Developer", "salary": 75000, "joiningDate": "2023-04-15"}
- Click "Send" and verify you receive a 200 OK response

3. **Testing Get All Employees**:
   - Method: GET
   - URL: http://localhost:5000/api/employees
   - Click "Send" and verify you receive a 200 OK response with the list of employees
4. **Testing Get Employee by ID**:
   - First, get the ID of an employee from the previous request
   - Method: GET
   - URL: http://localhost:5000/api/employees/[employee_id] (replace [employee_id] with an actual ID)
   - Click "Send" and verify you receive a 200 OK response with the specific employee details
5. **Testing Update Employee**:
   - Method: PUT
   - URL: http://localhost:5000/api/employees/[employee_id] (replace [employee_id] with an actual ID)
   - Headers: Content-Type: application/json
   - Body (raw JSON):
   - { "salary": 80000, "department": "Research & Development"}
   - Click "Send" and verify you receive a 200 OK response with updated employee details
6. **Testing Delete Employee**:
   - Method: DELETE
   - URL: http://localhost:5000/api/employees/[employee_id] (replace [employee_id] with an actual ID)
   - Click "Send" and verify you receive a 200 OK response