Employee Management REST API

A Flask-based REST API for managing employees, built as part of the **HabotConnect Python Backend Developer** hiring task.
This API supports **secure CRUD operations**, token-based authentication, and MongoDB storage.

Features

- Token-based authentication (login required)
- Create, read, update, and delete employees
- Email validation and uniqueness check
- Name validation (non-empty)
- Filter employees by department and role
- Pagination (10 employees per page)
- Proper HTTP status codes
- Console logging for backend actions

Tech Stack

- Python 3.10+
- Flask
- MongoDB (PyMongo)
- Flask-CORS

- Werkzeug (for password hashing)

Project Structure

```
employee-management-api/ ├── app.py # Main application file (runs the API) ├── login.py # Handles user authentication and token generation ├── insertdata.py # Optional: populate the database with sample employees └── README.md # Project documentation
```

API Endpoints

Authentication

Method

Endpoint

Description

POST

/api/auth/login

Login and get token

Employee APIs (Authenticated)

Method

Endpoint

Description

POST

/api/employees/

Create a new employee

GET

/api/employees/

List employees (supports filter & pagination)

GET

/api/employees/{id}

Get employee by ID

PUT

/api/employees/{id}

Update employee by ID

DELETE

/api/employees/{id}

Delete employee by ID

Employee Fields

Field

Type

Required

Notes

id

string

Auto

Auto-generated

name

string

Yes

Cannot be empty

email

string

Yes

Must be unique

department

string

No

Optional

role

string

No

Optional

date_joined

string

Auto

Auto-generated

How to Run Locally

1. Clone the repository:

```
git clone https://github.com/your-username/employee-management-api.git cd employee-management-api
```

2. Install dependencies:

```
pip install flask flask-cors pymongo werkzeug
```

3. Run the application:

```
python app.py
```

Server will start at:

```
http://localhost:5000
```

Optional Scripts

- `login.py` → Test login and generate token
- `insertdata.py` → Populate database with sample employees for testing

Testing the API

1. **Login** to get token (using `login.py` or `/api/auth/login`)
2. **Pass token** in the `Authorization` header for all employee endpoints:

```
Authorization: Bearer <your_token_here>
```

3. **Example: Create Employee**

```
curl -X POST http://localhost:5000/api/employees/ \ -H "Authorization: Bearer <TOKEN>" \ -H "Content-Type: application/json" \ -d '{"name": "John Doe", "email": "john@example.com", "department": "IT", "role": "Developer"}'
```

4. **Example: Get Employees (Filtered & Paginated)**

```
GET /api/employees/?department=IT&page=2
```

Demo Workflow for Video Submission

1. Run `python app.py` to start the server.
2. Open `login.py` or Postman to login with your credentials and get the token.
3. Copy the token and use it in the `Authorization` header for employee endpoints.
4. Demonstrate CRUD operations:
    - Create a new employee (`POST /api/employees/`)
    - List employees (`GET /api/employees/`)
    - Update an employee (`PUT /api/employees/{id}`)
    - Delete an employee (`DELETE /api/employees/{id}`)
5. Optionally, use `insertdata.py` to populate sample data for testing.

Notes

- All employee endpoints are protected using **token-based authentication**
- Proper validation and error handling are implemented as per task requirements

---

Author

Karishma Sandupatla
Python Backend Developer Candidate
GitHub: https://github.com/karishmasandupatla/employee-management-api

Git Commands

```
git init git add . git commit -m "Initial commit: Employee Management REST API" git branch -M main git remote add origin https://github.com/your-username/employee-management-api.git git push -u origin main
```