API.py

api.py contains the API routes that interact with the HR tracking system. It is built based on flask and interacts with a postgreSQL database through a file called db.py

Imports

```
from flask import Flask, request, jsonify
import datetime
import db
from db import *
from flask_jwt_extended import JWTManager, get_jwt_identity, jwt_required,
create_access_token
```

- Flask is used to create the api server
- JWT manager from flask_jwt_extended is used for authentication with JWT (JSON Web token)
- Db is the database module used to interact with the PostgreSQL database

API Routes

- /login post method takes an email and checks whether it is in the db and the role is admin
    - Generates a token to to use
    - Unfortunately due to my lack of experience as a data analyst in setting up APIs i couldn't find use this token again for the rest of the routes
- /employee/<employee_id> get method
    - Gives data about employee based on id
- /add-employee post method
    - Adds either one employee or multiple employees to the db
        - Format: email, name, assigned office id, and role
- /checkin and /checkout post method
    - Allows employees to check and collects the office id, employee id and date//time
        - Can be adjusted to take the date/time when it is entered
    - Format example for one row
        - Id:1, check_in_date: Fri, 20 Dec 2024 08:00:00 GMT, check_out_date: Fri, 20 Dec 2024 17:00:00 GMT, employee_id:1, check_in_office_id:1, check_out_office_id:1
    - /attendance get method
        - Returns the attendance
    - /update-employee-role/<int:employee_id> put methods
        - Updates the employee role based on employee_id
    - /get-data?table=tablename get methods
        - Returns the full table asked for

DB.py

db.py contains the code that builds the database and makes the api to interact with it using pg8000 to establish the connection and is basically made of different functions that uses sql queries to interact with the database.
It intallizes the table if the table are not made and also there are functions which the tables can be updated, or inserted into

Reports.py

The reports.py script fetches data from a local server for employees, offices, and check-in/check-out records. It processes the data into pandas DataFrames, adds a new column (moved_offices) to track if employees moved offices, and calculates the working_hours by converting check-in and check-out times into datetime objects. The script then visualizes the number of employees who moved offices versus those who didn’t with a bar chart.