Backend Homework

I. Problem Description

Contoso Ltd (a company) needs a way for employees and employers to track spending ("报销"). Specifically, two groups of users, employees and employers, are present in the system, and:

- **Employees**:
  - Creates "tickets" - representing a purchase record containing when, who, how much money did one spend, and field for relevant links to the purchase
  - Cannot see tickets created by others
- **Employers**:
  - Can see everyone's ticket as a long list
  - Can approve tickets or deny tickets
  - Can suspend an employee such that the employee cannot login again under the same email, and the tickets belonging to that employee will be hidden (but still present in the database, soft-deleted, feel free to add a flag of sorts to achieve this soft-deletion behavior)

Page designs:

- Login Page
  - The user is uniquely identified by **Email**
  - User is prompted to enter email and password. For the only action button "Login / Register"
    - In case of existing user, deny login if wrong password
    - in case of new user, hash the password, ask the user which role the user wants to be as well as username. Once the user is created with the information above, login
- Ticket List Page
  - Employee:
    - elects to add new entries to the ticket list, and the list should only show the tickets belonging to the current employee
  - Employers:

        - See everyone's tickets.
        - Can approve/deny tickets
- Employee List Page
    - Only visible to Employers
    - Employers elect to suspend and re-activate employees. Banned employees cannot login, nor can **anyone** see their tickets on the Ticket List Page.

II. Tech Stack Requirement

- Backend: written in **Python**
    - **Environment Setup**: Astral UV, here is a tutorial
    - **Python** : FastAPI + SQLAlchemy + asyncpg
        - FastAPI: https://fastapi.tiangolo.com/
        - Yes, please use PostgreSQL for the DB
    - **Must be able to handle multiple requests sent to the same endpoint concurrently/in parallel. We therefore require the use of asyncio via FastAPI.**

III. Deliverable

A github or gitee repository containing both the frontend and backend like this:

```
README.md
backend/
  uv.lock
  pyproject.toml
  src/
    (backend source files)
```

Where README.md should contain:

- How to setup DB
- How to install packages and run your project
- (see below for bonus tasks, some content also goes to README.md)

IV. Bonus Tasks!

The following tasks are completely optional but will add value to the mini-project, which in turn allow us to better understand your coding style

- **Unit Tests or Integration Tests**: for the backend, write some tests using unittest.
    - Minimally, set up the testing scaffolding and explain how to run your tests in README.md.
    - Try to have some chained integration tests where you make a GET request on a certain resource, then make POST request to somehow update it, and use GET to verify whether the transaction is successful.
- **Styling**
    - **Backend**: Setup python formatter black. Provide a brief tutorial for how to use it in README.md
- Authentication
- Docker Deployment