Deployment

Overview:

This project consists of three components:

- **Frontend** (Static HTML/CSS/JS)
- **Backend** (Flask Python API)
- **Database** (PostgreSQL)

This document provides recommended deployment options.

Local deployment:

1. Intended for **development, testing, and internal demonstration**.
2. It runs all components (Flask backend, PostgreSQL database, frontend files) on a developer's local machine.
3. We have "setup_env.bat" and "setup_env.sh" file for both windows and macOS/Linux/unix，these files can make local deployment easier.

Recommended cloud deployment：

Component	Recommended Platform	Description
Backend (Flask API)	**Vercel**, **Render**, or **Railway**	Easy to deploy Flask apps with auto-build from GitHub
Database(postgresSQL)	**Neon**, **Supabase**, or **Render PostgreSQL**	Cloud-hosted PostgreSQL with free tier
Frontend(HTML/css/js)	**Vercel**, **Netlify**, or **GitHub Pages**	For static files (HTML/CSS/Js)

Example:

1. Vercel + Neon (this is suitable for demostration)

+: a. Excellent for static sites — smooth deployment for HTML/CSS/JS and frontend frameworks.

b. Free tier available

c. Fast deployment

-: a. File system is **ephemeral**, cannot persist local files.

b. Serverless functions may have **cold start delay** on first requests.

2. Render（Web Service） + Render PostgreSQL (this is suitable when the application needs stable backend)

+: a. **Persistent backend** — Flask runs as a long-running process.

b. Unified environment for both app and database.

c. Built-in logs, metrics, and deployment dashboard.

-: a. Free tier may **sleep after inactivity**

b. Limited storage and compute resources in free tier.

3. Deploy app servers close to your users · Fly (Docker) + Neon/Supabase (Database) (Developers comfortable with DevOps who want fine-grained control.)

+: a. **Full control** — deploy real containers.

b. Global deployment and volume support.

c. Excellent for production-level control and scaling.

-: a. Requires Dockerfile and more configuration.

b. Slightly higher learning curve.

Summary:

For different project requirements, we can adopt different deployment approaches. For simple and fast demonstrations, **Vercel** and **Neon** are suitable choices. For slightly more complex **small-scale product deployments**, **Render** provides a more stable solution. If a **more professional or production-grade deployment** is required, we can use **Docker**.