Project Report: Explain This with AI Chrome Extension

Goal:

I created a Chrome Extension that allows me to right-click on selected text in a browser and get a simplified explanation using OpenAIs GPT model. This explanation is retrieved by sending the text to a backend Flask API that communicates with OpenAI's API and returns a response.

Tools & Technologies Used:

- Chrome Extension APIs: To create context menus and access selected text
- JavaScript: To implement the extensions background script and handle user interactions
- Flask (Python): To create a simple web API for processing requests and interacting with OpenAI
- Render: To deploy and host the Flask backend
- Git & GitHub: For version control and to deploy code on Render
- OpenAI API: To generate simplified explanations from GPT-3.5-turbo

Application Flow:

1. I select text on any webpage.
2. I right-click and choose Explain this with AI from the context menu.
3. The Chrome Extension sends the selected text to my Flask backend via a POST request.
4. The Flask server sends the text to OpenAIs ChatCompletion API.
5. The simplified explanation is returned and displayed to me in an alert.

Chrome Extension Files:

- manifest.json: Declared extension permissions, background scripts, and the context menu.
- background.js: Added a context menu, captured the selected text, and handled sending it to the backend.

Flask Backend Files:

- app.py: The main Flask application. It receives the text from the Chrome Extension, uses OpenAI to get a response, and sends it back.
- requirements.txt: Lists the Python dependencies used (flask, flask-cors, openai).

Deployment:

I used Render to deploy the Flask backend. I connected my GitHub repository and configured the build to install dependencies and run app.py. The app runs on port 10000 and exposes a /explain route that accepts POST requests with the selected text.