Sprawozdanie z projektu zaliczeniowego

Przedmiot: Narzędzia do automatyzacji budowy oprogramowania
Temat projektu: CodeBeautifier – Webowy asystent AI do poprawy i analizy kodu Pythona
Grupa projektowa 3J:

- Kajetan Szymczak (95522)
- Mariusz Szmondrowski (96591)
- Adrian Mikołajczyk (96192)
- Jan Kuliński (95402)

1. Cel projektu

Celem projektu było stworzenie webowej aplikacji wspierającej analizę i automatyczne poprawianie kodu w języku Python przy pomocy sztucznej inteligencji. Projekt miał za zadanie połączyć różne elementy: frontend, backend, API OpenAI oraz hosting w celu stworzenia funkcjonalnego narzędzia edukacyjno-programistycznego.

2. Opis aplikacji CodeBeautifier

CodeBeautifier to interaktywna aplikacja webowa, która umożliwia użytkownikowi:

- pisanie kodu Pythona w edytorze z kolorowaniem składni,
- uruchamianie kodu bezpośrednio w przeglądarce,
- analizę kodu przez sztuczną inteligencję (GPT-4),
- automatyczne wprowadzanie sugerowanych poprawek,
- prowadzenie rozmowy z AI w celu wyjaśnienia działania kodu, refaktoryzacji lub rozwiązania błędów.

3. Funkcjonalności aplikacji

- Edytor kodu z kolorowaniem składni (CodeMirror)
- Wbudowane środowisko uruchomieniowe kodu Python
- Analiza i poprawki kodu generowane przez AI (GPT-4)

- Automatyczne zastosowanie poprawek (przycisk "Zastosuj poprawki AI")
- Czat z AI jako pomoc programistyczna
- Dostępność aplikacji online (hosting na Render.com)

4. Wykorzystane technologie

- **Frontend:** HTML, CSS, JavaScript, CodeMirror
- **Backend:** Python, Flask
- **AI:** OpenAI GPT-4 API
- **Inne biblioteki i narzędzia:**
  - `dotenv` – obsługa zmiennych środowiskowych
  - `subprocess` – uruchamianie kodu Pythona
  - `JSON` – komunikacja między frontendem a backendem
- **Hosting:** Render.com

5. Struktura projektu

Projekt został zorganizowany w następującej strukturze katalogów i plików:

- `backend.py` – plik główny serwera Flask obsługującego żądania API
- `static/index.html` – interfejs użytkownika aplikacji
- `.env` – plik zawierający klucz API do OpenAI (niewersjonowany)
- `requirements.txt` – lista zależności potrzebnych do uruchomienia aplikacji

6. Uruchomienie projektu

Lokalnie:

1. Zainstalowanie zależności:
   ```
   pip install flask openai python-dotenv
   ```
2. Utworzenie pliku `.env` z kluczem OpenAI:
   ```
   OPENAI_API_KEY=your_api_key
   ```
3. Uruchomienie aplikacji:
   ```
   python backend.py
   ```
4. Wejście w przeglądarce na:
   http://localhost:5000

Online (hostowane):
Aplikacja została również opublikowana na platformie **Render.com**, co umożliwia jej użycie bez lokalnej instalacji – wystarczy przeglądarka internetowa. Strona działa online 24/7 i jest dostępna z każdego miejsca z dostępem do internetu.

7. Efekty i rezultaty

Projekt został zrealizowany zgodnie z założeniami. Aplikacja działa stabilnie i oferuje funkcjonalności przewidziane w planie. Udało się zintegrować model AI GPT-4 z własnym interfejsem webowym i zapewnić płynną współpracę pomiędzy frontendem i backendem. Aplikacja może być z powodzeniem wykorzystana do nauki programowania, testowania algorytmów oraz jako punkt wyjściowy do bardziej zaawansowanych rozwiązań.

8. Wnioski i refleksje

Projekt CodeBeautifier pozwolił zespołowi zdobyć praktyczne umiejętności z zakresu:

- integracji API sztucznej inteligencji z aplikacją webową,
- budowy aplikacji z architekturą frontend–backend,
- obsługi i bezpieczeństwa danych (zmienne środowiskowe),
- deployowania aplikacji na platformie hostingowej.

Zespół projektowy pozytywnie ocenia realizację zadania, a aplikacja stanowi wartościowe narzędzie edukacyjne. Możliwym kierunkiem dalszego rozwoju projektu jest dodanie wsparcia dla innych języków programowania, analiza statyczna kodu lub wersjonowanie kodu użytkownika.