Sprawozdanie z projektu

Programowanie Użytkowe

Python Chat Bot

Kamil Surlas

Michał Wilkosz

Cel projektu

Głównym celem projektu jest stworzenie interaktywnego asystenta (czatbota), który ułatwi proces nauki języka Python osobom początkującym.

Cele szczegółowe:

- Wyjaśnianie pojęć: Tłumaczenie skomplikowanych zagadnień (np. dekoratory, programowanie obiektowe) w przystępny sposób.
- Analiza błędów: Pomoc w zrozumieniu komunikatów błędów (Traceback) i sugerowanie poprawek w kodzie użytkownika.
- Objaśnianie składni języka Python.
- Nauka dobrych praktyk programistycznych.

Zespół i podział ról

Kamil Surlas:

- Odpowiedzialny za proces pozyskiwania i przygotowania danych edukacyjnych.
- Budowa logiki serwerowej w języku Python, odpowiedzialnej za przetwarzanie zapytań i integrację z modelem LLM (Gemini - Flash 2.5).
- Opracowanie systemu przeszukiwania stron internetowych w celu znalezienia jak najlepszej odpowiedzi dla użytkownika.
- Mechanizm przywracania haseł.
- Implementacja wsparcia dla wielu języków (PL – ENG)

Michał Wilkosz:

- Stworzenie API (backend)
- Stworzenie responsywnego i intuicyjnego interfejsu użytkownika w bibliotece React, zorientowanego na wygodę nauki (UX/UI).
- Integracja frontend'u z API.
- Implementacja uwierzytelniania.
- Implementacja formularza logowania / rejestracji / edycji użytkownika

Planowany stos technologiczny

1. Backend: Python
   Framework: FastAPI
2. Baza danych: SQLAlchemy, Alembic
3. Frontend: React, TypeScript
4. Ai: Gemini - Flash 2.5

5. Uwierzytelnianie: tokeny JWT

Architektura systemu

System został zaprojektowany w oparciu o klasyczną, trójwarstwową architekturę (Client-Server). Aplikacja działa w modelu SPA (Single Page Application), gdzie warstwa prezentacji jest wyraźnie odseparowana od logiki biznesowej, komunikując się z oddzielnym serwerem backend REST API.

Aplikacja typu SPA to nowoczesne podejście do tworzenia stron internetowych. W przeciwieństwie do tradycyjnych serwisów, które przy każdym kliknięciu przeładowują całą stronę, SPA ładuje tylko jeden główny dokument HTML. Zamiast pobierać nowe strony z serwera, aplikacja dynamicznie podmienia zawartość widoczną dla użytkownika, korzystając z danych pobranych w tle. Główne zalety tego rozwiązania w naszym systemie to:

1. Płynność działania: Interfejs reaguje natychmiastowo, przypominając w działaniu aplikację desktopową lub mobilną.
2. Optymalizacja: Mniejsze obciążenie sieci, ponieważ przesyłane są tylko czyste dane, a nie struktura graficzna strony.
3. Separacja: Niezależność frontend'u od backendu ułatwia rozwój i utrzymanie kodu.

Podział warstw

1. Warstwa prezentacji – odpowiada za interfejs użytkownika oraz interakcję z systemem.
    a. Technologia: Projekt zbudowany w oparciu o bibliotekę React.
    b. Komunikacja: Warstwa ta komunikuje się z serwerem wyłącznie poprzez protokół HTTP, wykorzystując standard REST API.
    c. Zarządzanie Stanem: Stan sesji użytkownika (access tokens) przechowywany jest bezpiecznie w mechanizmie przeglądarki localStorage.
2. Warstwa aplikacji – Zawiera logikę biznesową aplikacji oraz udostępnia API dla warstwy prezentacji.
    a. Technologia: Serwer napisany w języku Python przy użyciu nowoczesnego frameworka FastAPI.
    b. Rola: Działa jako centralny punkt logiki, udostępniając endpointy REST.
    c. Bezpieczeństwo i uwierzytelnianie:
        i. Wykorzystuje schemat OAuth2 z HTTPBearer do ekstrakcji tokenów.
        ii. Dedykowany Middleware get_current_user dekoduje token JWT i weryfikuje tożsamość użytkownika w bazie danych przy każdym chronionym zapytaniu.
        iii. Ochrona danych: Hasła użytkowników są haszowane algorytmem bcrypt. Klucze API do zewnętrznych usług (Gemini API) są przechowywane w bazie w formie zaszyfrowanej (AES/Fernet) i odszyfrowywane w locie wyłącznie na czas trwania zapytania do AI.
3. Warstwa zewnętrznych usług (Ai) - Moduł odpowiedzialny za interakcje z modelem językowym.

    a. LLM (Gemini – Flash 2.5): Backend pełni rolę pośrednika między użytkownikiem a API Google Gemini.
    b. Inteligentny chat_agent:
        i. Proces decyzyjny: Agent automatycznie decyduje o konieczności poszerzenia wiedzy. Wykorzystuje silnik *DuckDuckGo* do wyszukiwania informacji w sieci.
        ii. Przetwarzanie: System pobiera treść znalezionych stron (scraping), oczyszcza ją z elementów zbędnych i dołącza jako kontekst do *promptu* wysyłanego do modelu Gemini, co pozwala na generowanie odpowiedzi opartych o aktualne dane.
4. Warstwa danych.
    a. ORM: Do komunikacji z bazą danych wykorzystywany jest SQLAlchemy, co pozwala na operowanie na obiektach zamiast na surowym SQL.
    b. Zarządzanie Sesją: Wykorzystano wzorzec Dependency Injection wbudowany w FastAPI. Gwarantuje to utworzenie nowej, izolowanej sesji bazy danych dla każdego żądania HTTP i jej bezpieczne zamknięcie po zakończeniu obsługi.
    c. Wzorzec Repozytorium: Kod wykorzystuje warstwę abstrakcji w postaci repozytoriów. Oddziela to logikę API od bezpośrednich operacji na bazie danych, zwiększając testowalność i czytelność kodu.

Motywacja

Głównym celem projektu było zaprojektowanie i implementacja inteligentnego asystenta wspierającego naukę programowania w języku Python. Tematyka ta wynika z realnych barier, z jakimi mierzą się osoby początkujące w IT, takich jak wysoki próg wejścia, złożona składnia czy nieczytelne komunikaty błędów, często prowadzące do zniechęcenia na wczesnym etapie nauki. Opracowane narzędzie stanowi odpowiedź na te wyzwania, przekształcając proces edukacji z pasywnego w aktywną interakcję z inteligentnym systemem.

W przeciwieństwie do tradycyjnych form nauki, *Python Chat Bot* oferuje wsparcie, dostosowując się do problemów użytkownika. Pozwala to skupić się na logice i projektowaniu algorytmów, zamiast na żmudnym rozwiązywaniu błędów, co przyspiesza rozwój kompetencji inżynierskich. Projekt może pełnić również rolę całodobowego korepetytora, uczącego samodzielnej analizy i krytycznego myślenia.

Postępy realizacji projektu

20.01.2026

1. Utworzono bazę danych.
2. Utworzono modele tabeli.
3. Zaimplementowano warstwę API do logowania i rejestracji użytkowników.
4. Zaimplementowano klasę *Security* do szyfrowania kluczy.
5. Zaimplementowano sesje użytkowników.

27.01.2026

1. Zaimplementowano mechanizm przeszukujący strony internetowe w celu jak najlepszego dopasowania odpowiedzi.
2. Zaimplementowano szyfrowanie klucza Gemini API.
3. Zaimplementowano wsparcie dla historii konwersacji.
4. Nowe endpointy:
    a. /Chat/new - rozpoczęcie nowej konwersacji.
    b. /Chat/history - pobranie historii.
    c. /Chat - zaktualizowano w celu obsługi kontekstu obejmującego konwersację.
    d. /users/edit - edycja użytkownika.
5. Rozpoczęto implementowanie warstwy frontend

03.02.2026

1. Zaimplementowano edycję danych użytkownika.
2. Zaimplementowano możliwość przywrócenia hasła.
3. Zaimplementowano warstwę frontend, w tym:
    a. Rozmowa z chat'em.
    b. Panel logowania / rejestracji.
    c. Panel edycji profilu, w tym przywrócenia hasła.
    d. Możliwość zmiany języka strony na angielski.

Podsumowanie

Dzięki projektowi *Python Chat Bot* udało się stworzyć kompletną aplikację webową obejmującą zarówno warstwę frontendową, odpowiedzialną za interakcję z użytkownikiem, jak i backendową, realizującą logikę biznesową systemu. Aplikacja została zaprojektowana z myślą o wspomaganiu nauki programowania w języku Python.

System pełni rolę inteligentnego chatbota wykorzystującego mechanizmy sztucznej inteligencji, w szczególności model językowy Google Gemini, który umożliwia prowadzenie kontekstowych i spersonalizowanych dialogów z użytkownikiem. Historia rozmów jest zapisywana w bazie danych, co zapewnia ciągłość interakcji oraz umożliwia analizę postępów użytkownika w procesie nauki.

Dodatkowo aplikacja wykorzystuje mechanizmy przeszukiwania Internetu, dzięki czemu chatbot jest w stanie dostarczać aktualne, rzetelne informacje dotyczące języka Python, jego bibliotek oraz narzędzi. Zastosowane rozwiązania technologiczne pozwalają na stworzenie nowoczesnego, interaktywnego środowiska edukacyjnego, wspierającego rozwój umiejętności programistycznych w sposób bardziej efektywny niż tradycyjne formy nauki.

Dokumentacja

Dokumentacja projektu została umieszczona wraz z kodem źródłowym wewnątrz repozytorium na GitHubi'e.

Link do repozytorium

Docs

Wewnątrz folderu *Docs* znajduję się prezentacja, sprawozdanie (*report.pdf*) oraz krótki film przedstawiający działanie systemu.