**Dokumentacja inżynierii wymagań**

**Członkowie zespołu:**

1.Macierz kompetencji zespołu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kompetencje** | **Artur Donimirski** | **Szymon Burliga** | **Maciej Jurczyga** | **Piotr Zapiór** |
| js | Posiada(podstawy) | Posiada | Posiada | Posiada |
| react | Posiada(podstawy) | Posiada | Posiada | Posiada |
| postgresql | Posiada | Posiada | Posiada | Posiada |
| backend | Posiada(podstawy) | Posiada(podstawy) | Posiada | Posiada |
| testy | Nie posiada | Nie posiada | Posiada | Posiada |
| llm/api ai | Posiada(podstawy) | Posiada(podstawy) | Posiada(podstawy) | Posiada(podstawy) |
| java-spring | Nie posiada | Posiada(podstawy) | Posiada | Posiada |
| golang | Nie posiada | Nie posiada | Nie posiada | Posiada |
| python | Posiada | Posiada(podstawy) | Nie posiada | Posiada(podstawy) |
| soft skills | Posiada | Nie posiada | Posiada | Nie posiada |
| nosql | Posiada(podstawy) | Nie posiada | Posiada | Posiada(podstawy) |

2. Tabela pytań, które zostały sformułowane w celu uszczegółowienia zdanego projektu

|  |  |  |
| --- | --- | --- |
| Pytanie | Odpowiedź | Uwagi |
| Co ma robić aplikacja? | Aplikacja służy do nauki języka angielskiego,hiszpańskiego lub polskiego. |  |
| Jaka jest forma aplikacji? | Aplikacja webowa |  |
| Co może robić użytkownik? | Rozmawiać z botem w obcym języku | Język może zostać zmieniony z jednego na drugi |
| W jaki sposób użytkownik wprowadza dane? | Przez mowę lub wpisane tekstu |  |
| Czy jest możliwość korzystania z COTS? | tak |  |
| Czy aplikacja poprawia błędy użytkownika? | Tak, poprawia błędy | Wyświetlanie błędów użytnownika będzie na jego własne życzenie |
| Jakie cechy musi posiadać GUI aplikacji? | Przejrzystość, prostota oraz intuicyjność |  |
| W jaki sposób aplikacja zwraca dane użytkownikowi | Przez mowę lub wypisanie tekstu |  |
| Ile ma wynosić czas opowiedzi | Czas odpowiedzi nie powinien wynosić więcej jak 10s |  |

Czy wszystkie wymagania kliena są możliwe do spełnienia? NIE – ze względu na ograniczenia technologiczne.

Niektóre przeglądarki nie obsługują wymaganych bibliotek.

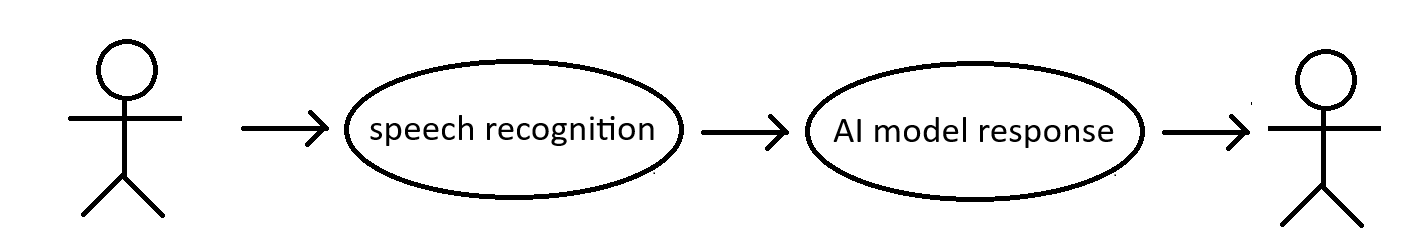
3. Ustalony format wejściowych.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Input | Lang | Response | Response\_type | timestamp |
| Int(unique) | String – tekst/mowa | String- język zapytania | String – odpowiedz od API | String – typ odpowiedzi | Epoch – czas wysyłania zapytania |

4.1 Modelowanego systemu za pomocą tabeli

|  |  |
| --- | --- |
| **Aktorzy** | Użytkownik, system API OpenAI |
| **Opis** | Celem działania aplikacji jest przyjęcie tekstu lub mowy od użytkownika w wybranym języku, przesłanie jej do modelu OpenAI API, a następnie zwrócenie odpowiedzi w formie tekstowej lub głosowej. Użytkownik wybiera preferowany typ odpowiedzi (tekst lub mowa). |
| **Dane** | Tekst lub dźwięk wejściowy, parametry językowe, model API OpenAI |
| **Wyzwalacz** | Po otrzymaniu tekstu lub mowy, aplikacja przesyła dane do modelu OpenAI API |
| **Odpowiedź** | Zwracana jest odpowiedź w formie tekstowej lub głosowej zgodnie z preferencjami użytkownika |
| **Uwagi** | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Obsługiwane języki muszą być zgodne z możliwościami modelu API OpenAI. W przypadku mowy wymagana jest konwersja mowa-tekst lub tekst-mowa w zależności od wybranego trybu odpowiedzi. | | |

4.2 Przedstawienie modelowanego systemu za pomocą diagramów UML



Rys. 1 Diagram przypadków użycia

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Rys. 2 Diagram przepływu danych

5. Diagram sekwencyjny UML

Obraz zawierający tekst, diagram, zrzut ekranu, linia

Opis wygenerowany automatycznie

6. Projekt architektury opracowanego systemu

Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

Opis wygenerowany automatycznie

7. Sugerowany język implementacji: JavaScript (React), GO

**Uzasadnienie:**

JavaScript z Reactem na frontendzie oraz Go na backendzie to świetne połączenie dla aplikacji webowej, która przetwarza tekst lub mowę i korzysta z OpenAI API. React, jako nowoczesna biblioteka frontendowa, umożliwia tworzenie dynamicznych i responsywnych interfejsów użytkownika, które są intuicyjne i szybkie. Dzięki komponentom Reacta rozwój interfejsu jest zwinny i łatwy w utrzymaniu, co przyspiesza tworzenie oraz aktualizację aplikacji.

Z kolei Go na backendzie zapewnia wysoką wydajność i skalowalność, co jest kluczowe przy przetwarzaniu większej liczby żądań do API OpenAI. Go, znany ze swojej efektywności i szybkości, świetnie nadaje się do aplikacji serwerowych, które muszą obsługiwać wiele jednoczesnych zapytań bez obciążenia serwera. Połączenie frontendu w JavaScript/React z backendem w Go tworzy solidną architekturę: frontend jest responsywny i interaktywny, a backend wydajny i bezpieczny, co gwarantuje płynne przetwarzanie danych użytkowników oraz bezproblemową integrację z API. Taki zestaw technologiczny sprawia, że aplikacja działa szybko, niezawodnie i jest gotowa do obsługi dużego ruchu w czasie rzeczywistym.