Lista zagadnień i architektura systemu dla projektu "Lateral Thinking Game"

Lista zagadnień

Zagadnienia funkcjonalne

- **Interfejs użytkownika**:
  - Intuicyjny i przyjazny UI, umożliwiający zadawanie pytań.
  - Pasek postępu wizualizujący postęp w odkrywaniu historii.
  - Wyświetlanie historii i podpowiedzi.
  - *Opcjonalnie:* wsparcie wersji językowych:
    - Możliwość przełączenia między wersją anglojęzyczną i polską.
- **Obsługa sesji gry**:
  - Przechowywanie stanu gry dla każdej sesji.
  - **Typy sesji**:
    - Anonimowa sesja gry (domyślna).
    - *Opcjonalnie:* logowanie użytkownika — typy uwierzytelniania:
      - **JWT** lub **session-based** (frontend: account session page, backend: auth service).
  - Informacje o sesji przechowywane w backendzie (np. postęp, ID sesji).
- **System podpowiedzi**:
  - Kontekstowe podpowiedzi na podstawie zadanych pytań i postępu.
- **Analiza języka naturalnego (NLP)**:
  - Rozumienie pytań użytkownika w języku naturalnym.
  - Generowanie logicznych odpowiedzi typu "tak", "nie".
  - Identyfikacja kluczowych punktów historii.
- **Leaderboard**:
  - **Obsługa rankingu graczy**:
    - Zapytanie do endpointu na podstawie ID sesji.
    - ID sesji używane do przechowywania informacji o wynikach w systemie leaderboard.

Zagadnienia techniczne

- **Integracja modelu językowego**:
  - Wykorzystanie zaawansowanego LLM przez API (OpenAI GPT-4)
  - Optymalizacja komunikacji między aplikacją a modelem (API).
- **Baza danych**:
  - Przechowywanie zagadek, punktów kluczowych, podpowiedzi oraz wyników leaderboard.
  - Logowanie sesji gry dla analizy i ewentualnych poprawek.
- **Skalowalność**:
  - Obsługa wielu użytkowników jednocześnie.
  - Skalowanie backendu aplikacji w zależności od ruchu.
- **Bezpieczeństwo**:
  - Ochrona danych użytkownika i sesji gry.
  - Bezpieczna komunikacja między serwerem a klientem.
- **Frontend przechowuje informacje o sesji**:
  - Żądania do backendu są wysyłane z ID sesji użytkownika w celu synchronizacji danych.

Zagadnienia niefunkcjonalne

- **Wydajność**:
  - Szybka odpowiedź systemu na zadane pytania.
- **Dostępność**:
  - Responsywny design — aplikacja powinna działać na desktopach i *możliwie* urządzeniach mobilnych.
- **Elastyczność**:
  - Możliwość łatwego dodawania nowych zagadek.
  - *Opcjonalnie*: obsługa różnych języków (angielski i polski).
- **Testowanie**:
  - Walidacja odpowiedzi modelu.
  - Testy funkcjonalne i UX.

Zagadnienia projektowe

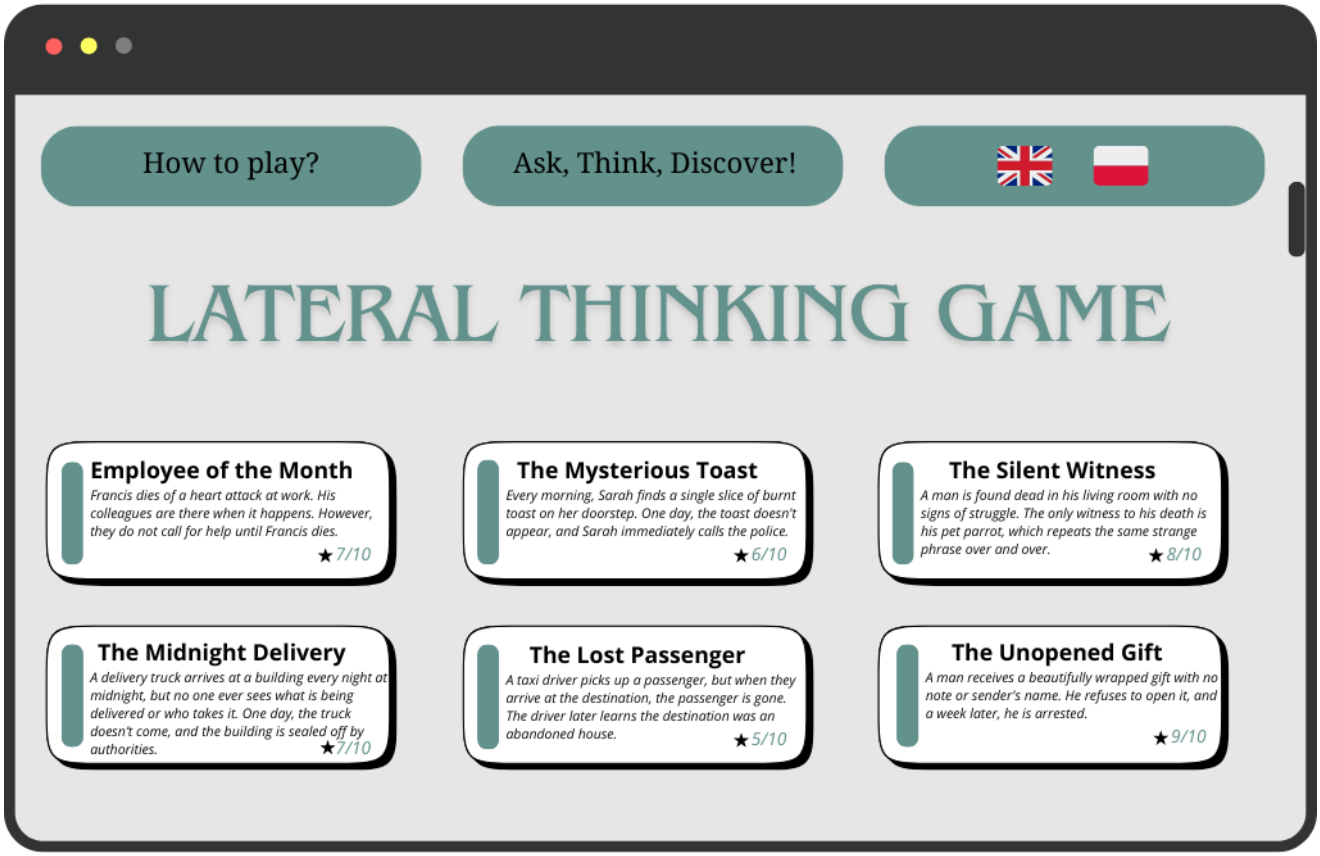
- **Tworzenie fabuł i zagadek**:
  - Unikalne, kreatywne historie dla każdej zagadki.

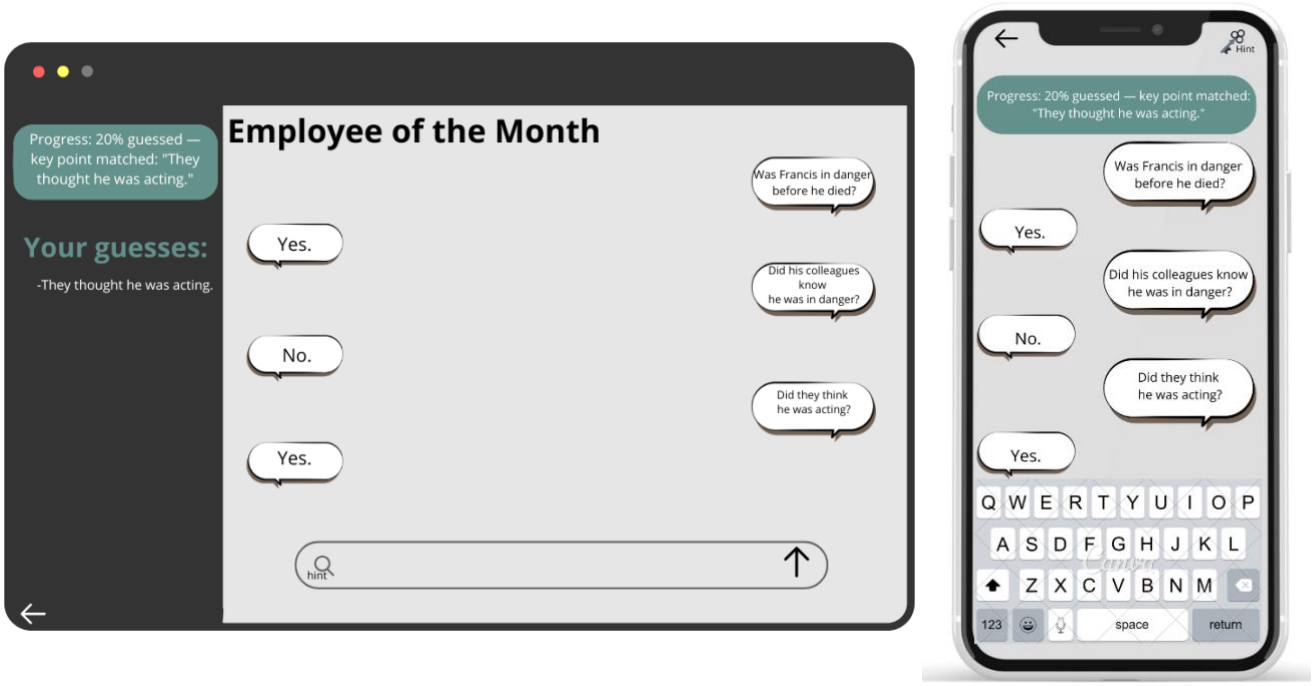
- **Mechanizmy sukcesu i porażki**:
  - Kryteria zakończenia gry.
- **Gamifikacja**:
  - *Opcjonalnie*: nagrody za rozwiązanie zagadek, np. odznaki lub punkty.

Architektura systemu

1. Warstwa frontendowa

- **Technologie**: HTML, CSS, JavaScript (React).
- **Funkcjonalności**:
  - Wyświetlanie historii początkowej i paska postępu.
  - Pole tekstowe do zadawania pytań.
  - Wizualizacja odpowiedzi i podpowiedzi.
  - Obsługa nawigacji i responsywności.
  - **Obsługa sesji**:
    - Frontend przechowuje informacje o sesji i wysyła żądania do backendu z ID sesji.





2. Warstwa backendowa

- **Technologie**: Python (FastAPI), Docker, inne biblioteki Pythonowe
- **Funkcjonalności**:
    - Obsługa zapytań od frontendu.
    - Integracja z API modelu językowego.
    - Zarządzanie bazą danych (zagadki, stany gry, logi, ranking).
    - Algorytmy śledzenia postępu i generowania podpowiedzi.
    - **Autoryzacja** (opcjonalna):
        - Uwierzytelnianie przy użyciu JWT lub sesji.

3. Model językowy (LLM)

- **Rozwiązanie**: GPT-4.
- **Zadania**:
    - Analiza pytań użytkownika.
    - Generowanie odpowiedzi i wskazówek.
    - Dopasowanie kluczowych punktów do progresu w grze.

4. Baza danych

- **Technologia**: PostgreSQL, MongoDB lub DynamoDB.

5. Warstwa integracji

- **API**:
    - REST do komunikacji między frontendem i backendem.

    - API zewnętrzne dla modelu językowego.
- **Leaderboard**:
    - Endpoint obsługujący leaderboard na podstawie ID sesji.
    - Dane sesji wykorzystywane do zapisu i odczytu wyników.

6. Infrastruktura

- **Hosting**: AWS lub DigitalOcean.
- **CDN**: Dostarczanie statycznych treści.
- **Konteneryzacja**: Docker dla spójności środowiska.

7. Platformy

- **Główna platforma**: Strona internetowa.
- **Możliwość aplikacji mobilnej**:
    - Rozważana jest implementacja wersji natywnej lub hybrydowej w przyszłości.

Schemat przepływu

1. **Użytkownik**: Zadaje pytanie w interfejsie.
2. **Frontend**: Wysyła zapytanie do backendu z ID sesji.
3. **Backend**:
    - Przesyła pytanie do modelu językowego przez API.
    - Analizuje odpowiedź i dopasowuje ją do postępu w grze.
    - Aktualizuje stan gry i pasek postępu.
    - Wysyła dane do leaderboard, jeśli gra się zakończyła.
4. **Frontend**: Wyświetla odpowiedź oraz aktualizuje wizualizacje.
5. **Opcjonalnie**: Użytkownik może poprosić o podpowiedź.