



POLSKO-JAPONSKA
AKADEMIA TECHNIK
KOMPUTEROWYCH

Gdańsk, 3 czerwca 2024

Kierunek studiów: Informatyka

Rodzaj studiów: Zaoczne

Praca dyplomowa

Temat pracy:

Fishki - multiplatformowa aplikacja służąca do nauki pamięciowej

Temat w języku angielskim:

Fishki - a multiplatform application for memory learning

Opiekun pracy:

dr Puźniakowski Tadeusz

Wykonawcy:

Nazwisko, imię	Nr albumu
Kossak Oliwier	s22018
Klimowski Daniel	s18504
Krieger Wiktor	s23638
Żurawski Jakub	s23047

Streszczenie:

TBC



POLSKO-JAPONSKA
AKADEMIA TECHNIK
KOMPUTEROWYCH

Karta projektu

Temat projektu: Fishki - multiplatformowa aplikacja służąca do nauki pamięciowej		Akronim: Fishki Data ustalenia tematu 11.10.2023
Promotor: dr Puźniakowski Tadeusz		Konsultanci: 1. — brak —
Cele projektu: Dostarczenie systemu do efektywnej nauki z wykorzystaniem techniki zapamiętywania w postaci fiszek. Cel projektu odpowiada na problem rozumiany jako trudności w organizacji oraz korzystania z materiałów służących do opanowywania pojęć.		
Rezultaty i zakres	Oczekiwane produkty/usługi: Strona internetowa, aplikacja mobilna, serwer obsługujący utrzymanie i żywotność produktu Główne funkcjonalności i/lub cechy: Nauka metodą fiszek, obsługa talii tj. zestawów fiszek, obsługa talii fiszek za pomocą mowy, tworzenie fiszek z pomocą sztucznej inteligencji.	
Miary sukcesu: Wytworzenie działającej strony internetowej i aplikacji mobilnej, opracowanie techniczne projektu z wykorzystaniem infrastruktury serwerowej, zaimplementowanie silnika sztucznej inteligencji, zrealizowanie wymagań systemowych na poziomie ‘wymagane’.		
Ograniczenia: Zdalny charakter pracy nad projektem, budżet, brak doświadczenia w pracy nad projektem o zadanej złożoności, nauka i wykorzystanie nowych technologii.		

Wykonawcy	Numer albu-mu	Specjalizacja	Tryb studiów
Kossak Oliwier	s22018	Sztuczna Inteligencja	Niestacjonarny
Klimowski Daniel	s18504	Sztuczna Inteligencja	Niestacjonarny
Krieger Wiktor	s23638	Sztuczna Inteligencja	Niestacjonarny
Żurawski Jakub	s23047	Sztuczna Inteligencja	Niestacjonarny

Oświadczenie autorów pracy dyplomowej

Świadomi odpowiedzialności prawnej oświadczamy, że niniejszą pracę dyplomową w zakresie przedstawionym przez nasz zespół projektowy wykonaliśmy samodzielnie i nie zawiera ona treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczamy również, że praca w przedstawionym przez nas zakresie nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu ukończenia studiów wyższych.

Oświadczamy ponadto, że niniejsza wersja pracy dyplomowej jest identyczna z załączoną wersją elektroniczną.

Spis treści

1 Wstęp	7
1.1 O projekcie	7
2 Omówienie problemu	8
2.1 Czym są fiszki?	8
2.2 Przedstawienie problemu	9
2.3 Zalety nauki z wykorzystaniem metody fiszek	10
2.3.1 Krzywa zapomnienia Hermanna Ebbinghausa	10
2.3.2 Przykłady praktycznych zastosowań fiszek	12
2.3.3 Personalizacja fiszek	13
2.4 Słownik pojęć	13
3 Projekt w kontekście problemu	14
3.1 Omówienie zakresu projektu	14
3.2 Proponowane rozwiązanie	14
3.3 Grupa docelowa	15
3.4 Analiza rozwiązań konkurencji	15
3.4.1 Quizlet	15
3.4.2 Fiszkoteka	17
3.4.3 Gizmo	18
3.4.4 Flashcards	19
3.5 Analiza szans oraz zalet względem konkurencyjnych rozwiązań	21
3.5.1 Szanse/Zalety względem konkurencyjnych rozwiązań	21
3.5.2 Wady względem konkurencyjnych rozwiązań	21
3.6 Analiza ryzyka	21
4 Społeczny aspekt projektu	24
4.1 Gamifikacja w kontekście produktu	24
4.2 Ryzyko związane z nauką przy pomocy systemu Fishki	24
5 Organizacja pracy	26
5.1 Harmonogram pracy	26
5.2 Wybrana metodyka	30
5.3 Zespół i podział obowiązków	30

6 Analiza wymagań	32
6.1 Diagramy	32
6.2 Wymagania ogólne	36
6.2.1 Moduł autoryzacji	36
6.2.2 Moduł zarządzania talią	36
6.2.3 Moduł uczenia	37
6.2.4 Moduł udogodnień	37
6.2.5 Moduł wspomagania tworzenia talii	38
6.3 Wymagania funkcjonalne	38
6.3.1 Rejestracja konta użytkownika	38
6.3.2 Logowanie do systemu	39
6.3.3 Wylogowanie z systemu	39
6.3.4 Edycja danych użytkownika	40
6.3.5 Usunięcie konta użytkownika	41
6.3.6 Tworzenie talii fiszek	41
6.3.7 Usunięcie talii fiszek	42
6.3.8 Edycja talii fiszek	42
6.3.9 Tryb uczenia się z talii fiszek	43
6.3.10 Sterowanie talią przy użyciu mowy	44
6.3.11 Import talii innych użytkowników	45
6.3.12 Tryb dark mode i light mode	45
6.3.13 Kalendarz śledzący aktywność	46
6.3.14 Weryfikacja konta użytkownika poprzez email	46
6.3.15 Tworzenie talii fiszek poprzez zeskanowanie dokumentu	47
6.3.16 Generowanie treści fiszki na podstawie słów kluczowych	47
6.3.17 Resetowanie hasła na konta	48
6.4 Interfejs z otoczeniem	48
6.5 Wymagania pozafunkcjonalne	49
6.5.1 Instrukcja korzystania z trybu sterowania głosem	49
6.5.2 Limit błędów dotyczących logowania	49
6.5.3 Dostępność systemu	50
6.5.4 Responsywność	50
6.5.5 Kompatybilność	50
6.5.6 Hashowanie haseł	51
6.6 Wymagania na środowisko docelowe	51
6.6.1 Przeglądarka	51
6.6.2 System Android 10 i iOS 16	51
6.6.3 Kontenery dockerowe	52
6.6.4 Baza danych	52
6.6.5 Python	52
6.6.6 Node.js	53
6.6.7 System operacyjny	53
7 Architektura projektu i użyte technologie	54
7.1 Narzędzia organizacji pracy	54
7.2 Aplikacja webowa	57
7.3 Aplikacja mobilna	58
7.4 Backend aplikacji	58
7.5 Baza danych	58

7.6	Infrastruktura serwerowa	59
7.7	Narzędzia programistyczne	59
8	Implementacja	60
8.1	Faza Planowania	60
8.2	Faza Implementacji	61
8.2.1	Sprint 1	61
8.2.2	Sprint 2	62
8.2.3	Sprint 3	62
8.2.4	Sprint 4	63
8.2.5	Sprint 5	64
8.2.6	Sprint 6	65
8.3	Szczegóły implementacji	67
9	Testy	68
9.1	Testy integracyjne	68
9.2	Testy akceptacyjne	69
	Bibliografia	70

Rozdział 1

Wstęp

1.1 O projekcie

TBC

Rozdział 2

Omówienie problemu

2.1 Czym są fiszki?

Nauka z wykorzystaniem metody fiszek opiera się na systemie regularnych powtórek rozłożonych w czasie. Jest jednym z najpopularniejszych i najbardziej efektywnych sposobów pozwalających przyswoić i utrważyć wiedzę. Technika ta w klasycznej wersji polega na regularnym dobieraniu kart z talii, w której wszystkie są zapisane z dwóch stron – na jednej stronie każdej fiszki znajduje się zagadnienie, na drugiej, natomiast przeznaczone do zapamiętania klucz odpowiedzi lub definicja. Karty są dobierane z talii kolejno w sposób losowy, po każdorazowym dobraniu należy odczytać zagadnienie lub kłopotliwy termin, a następnie spróbować odpowiedzieć zgodnie z kluczem zapisanym na drugiej stronie. Wtedy dopiero można dokonać porównania i zestawić swoją odpowiedź z prawidłową, zapisaną na odwrocie fiszki. Następnie karta powinna wrócić do talii, tj. zbioru.

Rozwój technologiczny oraz dobiegająca zewsząd cyfryzacja powoli wypierają z użytku klasyczną formę stosowania fiszek na rzecz jej organizacji poprzez aplikacje mobilne oraz strony internetowe. Zmiana ta niesie wiele szans oraz możliwości udoskonalenia metody fiszek w jej skuteczności oraz dostępności. Fundamentalną zmianą jest samo przyśpieszenie całego procesu sporządzania talii oraz ich przeglądania – przy wykorzystaniu komputera lub smartfonu staje się to znacznie prostsze i szybsze niż ręczne zapisywanie kart, wycinanie ich i gromadzenie.

2.2 Przedstawienie problemu

Celem projektu jest zwiększenie efektywności uczenia się metodą fiszek poprzez jej integrację z interfejsem bezdotykowym. Pomimo dużego wkładu w przystępność i zwiększenie dostępności tej metody przez cyfrowe rozwiązań, istniejące aplikacje do tworzenia i przeglądania fiszek wymagają zasadniczo aktywnego zaangażowania użytkownika poprzez klikanie lub wpisywanie na klawiaturze – zarówno przy użyciu smartfonów, jak i komputerów osobistych. Ta konieczność bezpośredniej interakcji ogranicza możliwości nauki do określonych sytuacji, co stanowi barierę w momentach, gdy użytkownik wykonuje inne czynności, na przykład proste prace domowe, a urządzenie obsługujące aplikacje nie może być efektywnie wykorzystane.

Dodatkowo proces tworzenia fiszek, który wymaga od użytkowników manualnego redagowania, przepisywania lub kopowania definicji, znaczco opóźnia przygotowanie materiałów do nauki i wydłuża czas potrzebny na stworzenie talii kart. Ta czasochłonna procedura potrafi stanowić istotną przeszkodę, ograniczając proces uczenia się. Czas, który mógłby być przeznaczonym na utrwalanie wiedzy, jest poświęcany na przygotowanie zasobów służących do nauki. Sama potrzeba redagowania fiszek może odebrać użytkownikowi chęci by w ogóle korzystać z tej metody. Celem projektu jest częściowa automatyzacja procesu tworzenia talii poprzez wzbogacenie funkcjonalności o narzędzia sztucznej inteligencji. Skutkiem tego jest możliwość generowania kluczy odpowiedzi fiszek odpowiednio do podanych zagadnień.

W odpowiedzi na te wyzwania, opisany w niniejszej pracy system dąży do zminimalizowania ograniczeń związanych z tradycyjną obsługą aplikacji oferujących możliwość nauki metodą fiszek. Podjęty projekt proponuje rozwiązanie, które integruje proces przyswajania wiedzy z codziennymi czynnościami użytkownika oraz umożliwia jednocześnie szybsze i bardziej intuicyjne tworzenie materiałów do nauki. Celem realizowanego projektu jest stworzenie aplikacji, w której nauka jest nie tylko bardziej efektywna, ale również dostępna w szerokim zakresie sytuacji z pomocą bezdotykowej obsługi.

Poniżej znajduje się graficzna reprezentacja problemu ujętego w formie Rich Picture:



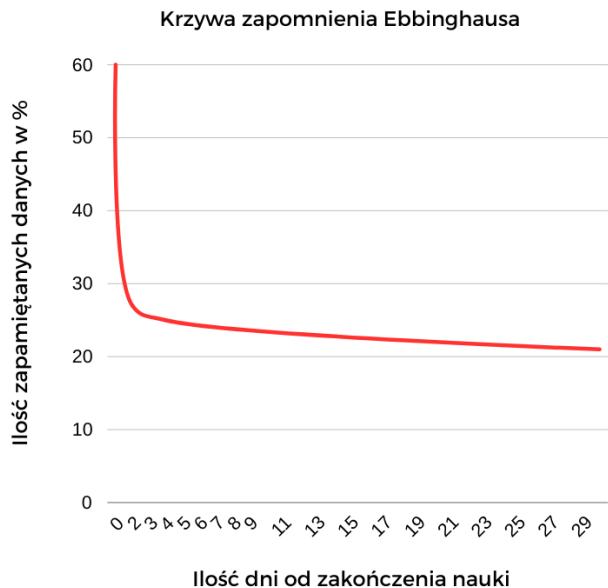
Rysunek 2.1: Problem przedstawiony w rich picture.

2.3 Zalety nauki z wykorzystaniem metody fiszek

2.3.1 Krzywa zapomnienia Hermanna Ebbinghausa

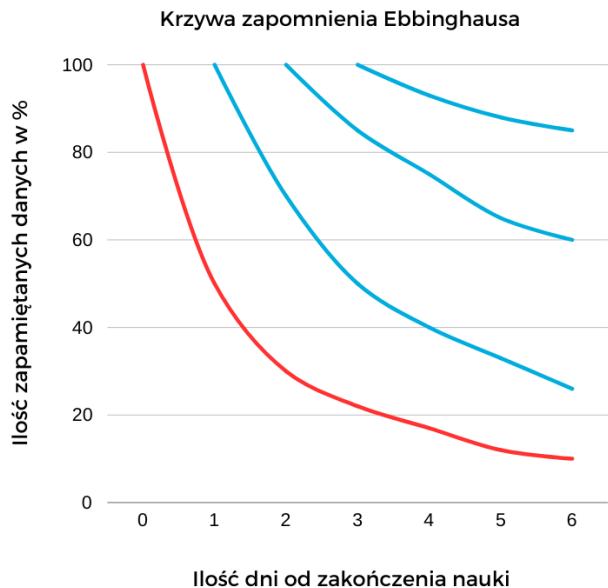
Opierając się na systemie regularnych powtórek, praktykowana metoda fiszek pozwala na utrzymanie stanu wiedzy na wysokim poziomie zgodnie z zasadami krzywej zapomnienia Ebbinghausa. Mowa o zależności opisanej przez niemieckiego psychologa, Hermanna Ebbinghausa. Według wyników badań opublikowanych w 1885 roku w pracy "Über das Gedächtnis"¹ (w tłumaczeniu „O pamięci”), najlepszym sposobem na zapamiętanie jest rozłożenie nauki w czasie. Praktyka ta jest znacznie skuteczniejsza od skupienia nauki w jednym okresie. Istotnym elementem zagadnienia jest wykres omawianej krzywej:

¹1.



Rysunek 2.2: Wizualizacja wyniku badań Ebbinghausa.

Powyższy reprezentuje tempo, w którym następuje zapominanie nauczonych treści po zakończeniu nauki. W przeciągu pierwszych kilku dni od jej zakończenia następuje gwałtowny spadek zapamiętanych informacji, nawet do poziomu 25% względem całkowitej ilości powtarzanych informacji. Dopiero po upływie tego czasu krzywa się stabilizuje, a tempo zapominania znacznie zwalnia. Najskuteczniejszym według badania sposobem by zapobiec temu procesowi jest regularne powtarzanie materiału. Poniższy wykres reprezentuje ilość przyswojonej wiedzy względem każdej kolejnej sesji nauki:



Rysunek 2.3: Wizualizacja wyniku badań Ebbinghausa.

W związku z powyższym przyjmuje się, że metoda nauki z regularnymi powtórkami, jaką oferuję metoda fiszek, pozwala na skuteczne utrwalenie wiedzy i znaczne spowolnienie zapominania jej. Użyte w badaniu informacje, które zostały podjęte nauce były danymi „bezużytecznymi”. Hermann Ebbinghaus celowo używał losowo utworzonych sylab, aby zminimalizować zależności między danymi a osobami które się uczyły. Powszechnie akceptowane w psychologii edukacyjnej jest twierdzenie, że przyswojenie informacji, które indywidualnie uznaje się za jakikolwiek interesujące dodatkowo ułatwia proces ich zapamiętania. Idące za tym wnioski plasują metodę regularnych powtórek tym bardziej przystępna, ponieważ znajduje zastosowanie nawet w przypadku zapamiętywania informacji, które nie koniecznie muszą być dla odbiorcy atrakcyjne.

2.3.2 Przykłady praktycznych zastosowań fiszek

Metoda nauki z wykorzystaniem fiszek znajduje szerokie zastosowanie w różnych dziedzinach nauki i życia codziennego, jej przystępna forma odpowiada poniższym przykładom:

- nauka języków obcych: fiszki są powszechnie stosowane w celu utrwalenia słownictwa, zwrotów lub zasad gramatycznych języków obcych; szybkie przypominanie i testowanie wiedzy jest w tym przypadku kluczowe w procesie opanowywania języka innego niż ojczysty;

- nauki techniczne: ze względu na dużą ilość szczegółowych informacji, które istotnie trzeba spamiętać, fiszki są odpowiednim narzędziem dla studentów lub pracowników branży technicznych. Pomagają one w efektywnym przyswajaniu skomplikowanych terminów;
- przygotowanie do egzaminów: Uczniowie i studenci przygotowujący się do egzaminów mogą wykorzystać fiszki w celu opanowania kluczowych pojęć, definicji czy zestawu odpowiedzi.

2.3.3 Personalizacja fiszek

Do najważniejszych zalet nauki metodą fiszek zalicza się możliwość personalizacji materiałów naukowych. Osoby korzystające z fiszek mogą tworzyć własne talie, dostosowane do ich indywidualnych potrzeb i preferencji. Dodatkowo fiszki mogą być łatwo dostosowane do różnych sposobów uczenia się, przykładowo poprzez stosowanie trybu obsługi głosowej.

2.4 Słownik pojęć

1. Fiszka - karta zapisana z dwóch stron, na jednej stronie znajduje się pytanie/zagadnienie, na drugiej odpowiedź/definicja.
2. Talia - zbiór, pula fiszek.
3. System - przedmiot projektu, ogólne określenie wytworzonej aplikacji mobilnej, webowej.
4. Gamifikacja - wykorzystanie elementów gier w niezwiązanym z grami kontekście².

²2, tutaj tekst?

Rozdział 3

Projekt w kontekście problemu

3.1 Omówienie zakresu projektu

W odpowiedzi na zdefiniowany i omówiony w Rozdziale II problem, niniejszy projekt zakłada wytworzenie aplikacji edukacyjnej, która umożliwi naukę metodą wykorzystującą fiszki. Projekt obejmuje dostarczenie aplikacji webowej dostępnej z poziomu przeglądarki internetowej oraz aplikacji mobilnej dla urządzeń z systemem Android lub iOS. W zakres prac wlicza się także zbudowanie pełnej infrastruktury wspierającej aplikację - backend oraz serwer utrzymujący cały system.

3.2 Proponowane rozwiązanie

Projekt bazuje na kilku kluczowych założeniach, które mają na celu bezpośrednie rozwiązanie problemów opisanych w Rozdziale II:

- obsługa głosowa: integracja obsługi głosowej aplikacji ma ułatwić korzystanie z niej w sytuacjach, w których użytkownik ma ograniczone możliwości fizycznej obsługi urządzenia. Dzięki temu nauka jest możliwa w każdych warunkach, co odpowiada na problem ograniczonej dostępności tradycyjnych metod nauki;
- narzędzie AI do tworzenia fiszek: udostępnienie narzędzia umożliwiającego automatyczne generowanie i dobieranie definicji/odpowiedzi do zagadnień w fiszkach. Pozwala przyśpieszyć i uprzystępnić proces redagowania fiszek.

3.3 Grupa docelowa

Główną grupę użytkowników stanowią uczniowie i studenci, a także osoby, które nie mają czasu na tradycyjną formę przyswajania wiedzy, na przykład czytanie lub naukę przy biurku w domowym zaciszu. Generowanie definicji przy użyciu programu ChatGPT pozwala na szybkie tworzenie zestawów fiszek, co może być bardzo wygodne dla osób ceniących sobie czas. Głosowa obsługa talii fiszek pozwala na wykonywanie sesji nauki w warunkach niesprzyjających innym formom uczenia się. Użytkownik ma możliwość powtórzenia materiału, wykonując inne obowiązki, takie jak gotowanie, sprzątanie lub inne czynności, którym nie trzeba poświęcać większej uwagi.

3.4 Analiza rozwiązań konkurencji

Ważnym aspektem tworzenia nowego produktu lub usługi jest analiza konkurencji. Dostrzeżenie wad i zalet konkurencyjnych rozwiązań pozwala na ulepszenie produktów dostępnych na rynku i dotarcie do nowej grupy klientów.

3.4.1 Quizlet

Jedna z pierwszych dostępnych na rynku aplikacja do fiszek. Utworzona w 2005 roku narzędzie do nauki zgromadziło ogromną bazę użytkowników szacowaną na około 50 milionów. Quizlet posiada stronę internetową i aplikacje mobilną dostępne na Android i IOS.



Rysunek 3.1: Aplikacja mobilna Quizlet.

Zalety:

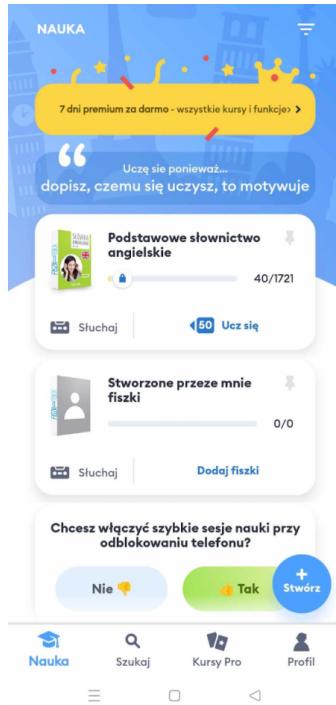
- Możliwość tworzenia testów w różnych trybach: prawda/fałsz, wielokrotny wybór, pisemny
- Odczytanie przez lektora treści fiszki poprzez kliknięcie przycisku
- Dostęp do materiałów, które zostały zweryfikowane przez ekspertów
- Personalizacja fiszek za pomocą obrazów i dźwięków
- Pozwala na wyszukiwanie tali u innych użytkowników

Wady:

- Reklamy w aplikacji
- Pełny dostęp do wszystkich funkcjonalności tylko za opłatą
- Brak rankingu popularności tali u użytkowników, co pozwalałoby na łatwy dostęp do innych tali
- Brak trybu sterowania tali głosem

3.4.2 Fiszkontakteka

Polski portal do nauki metodą fiszek ukierunkowany na naukę języków obcych. Oprócz strony internetowej aplikacja dostępna na system Android lub iOS.



Rysunek 3.2: Aplikacja mobilna Fiszkontakteka.

Zalety:

- Dostępne talie językowe o różnych poziomach trudności
- Odczytanie przez lektora treści fiszki po zobaczeniu odpowiedzi
- Tryb słuchania, tytuł i treść fiszek są odczytywane po kolej przekształcania przez lektora
- Dostęp do publicznych talii
- Można pobrać dokument PDF lub Mp3 z fiszek

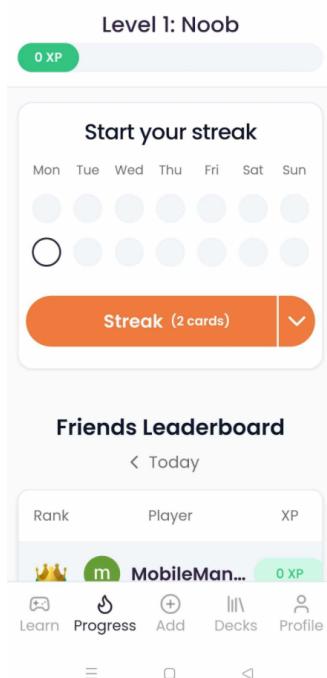
Wady:

- Pełny dostęp do wszystkich funkcjonalności tylko za opłatą
- Dla bezpłatnej wersji ograniczenie liczby utworzonych fiszek do 300

- Reklamy w aplikacji
- Lektor w bezpłatnej wersji ograniczony do 20 dźwięków dziennie
- Brak trybu sterowania talii głosem
- Aplikacja ukierunkowana na uczenie się słówek

3.4.3 Gizmo

Aplikacja wykorzystuje narzędzia sztucznej inteligencji, które pozwalają na tworzenie zestawów fiszek na podstawie plików PDF, CSV lub filmów na platformie YouTube. Gizmo posiada zarówno stronę internetową, jak i aplikację mobilną na Androida i iOS.



Rysunek 3.3: Aplikacja mobilna Gizmo.

Zalety:

- Brak reklam, brak opłat
- Zaawansowane zastosowanie AI do tworzenia fiszek
- Automatyczne tłumaczenie skanu na język angielski

- Szybki i prosty import fiszek z różnych źródeł
- Kalendarz passy (streak) zachęcający do nauki
- Przejrzysty interfejs
- Dostęp do publicznych talii

Wady:

- Nieumiejętnie użycie AI przy tworzeniu, prowadzi do utworzenia bezsensownych talii
- Dużo niezagospodarowanej przestrzeni na ekranie w trybie uczenia
- Brak rankingu popularności talii użytkowników
- Domyslnie irytująca częstotliwość powiadomień
- Brak trybu sterowania talii głosem

3.4.4 Flashcards

Aplikacja dostępna tylko w wersji mobilnej. Uorientowana na uczenie się ze słuchu i tworzenie talii z użyciem mowy. Nieintuicyjny interfejs sprawia, że aplikacja jest nieprzyjazna dla nowych użytkowników.



Rysunek 3.4: Aplikacja mobilna Flashcards.

Zalety:

- W trybie uczenia pozwala na korzystanie bez dotykania
- Możliwość tworzenia fiszek poprzez dyktowanie
- Brak reklam, brak opłat

Wady:

- Bardzo archaiczny interfejs
- Nieintuicyjny interfejs
- Skromna instrukcja korzystania
- W trybie uczenia można tylko słuchać i czekać na timer, nie ma nasłuchiwanego hasła/komend
- Brak publicznych talii

3.5 Analiza szans oraz zalet względem konkurencyjnych rozwiązań

Podrozdział zawiera spis najważniejszych funkcjonalności, które odróżniają aplikację fiszki od innych produktów dostępnych na rynku.

3.5.1 Szanse/Zalety względem konkurencyjnych rozwiązań

Zalety:

- Sterowanie głosowe talii
- Generowanie treści fiszki, wykorzystując ChatGPT
- Ranking najpopularniejszych talii

3.5.2 Wady względem konkurencyjnych rozwiązań

Wady:

- Aplikacja dostępna tylko w języku angielskim uniemożliwia uczenie się języków obcych
- Brak narzędzi generowania zestawów talii z różnych formatów plików takich jak: CSV, mp4, PDF
- Brak możliwości generowania testów wielokrotnego wyboru lub prawda/fałsz
- Nie można eksportować talii do innych formatów plików takich jak PDF lub CSV

3.6 Analiza ryzyka

Podrozdział przedstawia analizę ryzyka, która ma na celu określenie i zrozumienie potencjalnych zagrożeń, które mogą wpływać na proces tworzenia projektu.

Zidentyfikowane ryzyko [20]	Symptomy	Środki / Działania zapobiegawcze i szacowany poziom trudności ich wdrożenia	Środki / Działania minimalizujące wpływ na projekt – już po jego wystąpieniu i szacowany poziom trudności ich wdrożenia (1-10)	Ranga ryzyka (imi niższa, tym mniejszy negatywny wpływ na projekt)	Prawdopodobieństwo wystąpienia (1-100%)
Błędy w kodzie [O]	Błędy w komplikacji, wyniki testów nie pokrywają się z oczekiwany rezultatem	Szybka identyfikacja błędów, wykonywanie testów oprogramowania	Poprawa kodu, testowanie na bieżąco nowo implementowanych funkcjonalności (4)	10	80%
Awaria sprzętu deweloperskiego [S]	Sprzęt przestaje działać, brak możliwości odzyskania danych	Częste commits na repozytorium	Działanie na ostatniej wersji z repozytorium (1)	7	10%
Niedobór umiejętności w zespole [L]	Problemy jednostek w poszczególnych zadaniach	Uzupełnianie wiedzy	Pomoc innych członków zespołu (2)	7	70%

Niezgodność czasowa zespołu [C]	Osoba blokuje postęp nad projektem poprzez odpowiedzialność nad kluczowym elementem	Wcześniejszego zaplanowanie pracy	Wspólna praca nad kluczowym elementem zajęcie się zadaniami niepowiązanymi (5)	6	80%
Wybór nieodpowiednich technologii [T]	Planowane rozwiązania nie są osiągalne przez wykorzystywane technologie	Dogłębną analizą potrzebnych technologii i ich możliwości/ograniczeń	Dopasowanie alternatywnych możliwości rozwiązań (8)	5	50%
Niedoszacowanie budżetu potrzebnego do utrzymania infrastruktury [B]	Budżet zbliża się do wyczerpania przed zaplanowanym terminem	Analiza cenników wykorzystywanych produktów	Próba pozyksania inwestorów (10)	10	80%
ChatGPT 3.5 generuje bezsensowną treść dotyczącą zagadnienia o której został zapytany przez użytkownika [F]	Generowane treści nie nawiążują do zagadnień, o które chat został zapytany	Próba sprawdziania zapytania, które zostało przesłane do chatu w celu wygenerowania konkretniejszej treści	Użytkownik może zaakceptować lub odrzucić wygenerowaną treść. (4)	7	50%

Rozdział 4

Społeczny aspekt projektu

4.1 Gamifikacja w kontekście produktu

Została przeprowadzona analiza związana z wpływem rywalizacji na aktywność użytkownika, wynika z niej, że elementy rankingowe mają wpływ na zwiększoną aktywność użytkowników, z tego powodu aplikacja zawiera ranking popularności użytkowników i talii¹. Pozycja w rankingu użytkownika jest wyliczana na podstawie sumy wszystkich pobrań talii upublicznionych dla innych odbiorców systemu. Ranking talii to zestawy fiszek o największej liczbie pobrań. System rankingowy ma na celu wzbudzenie rywalizacji u użytkowników, co ma ich sprowokować do większej aktywności. System punktów zachęca także odbiorców systemu do tworzenia nowych zestawów talii i dzielenia się z nimi w celu zwiększenia swojej szansy na poprawienie swojej pozycji rankingowej.

4.2 Ryzyko związane z nauką przy pomocy systemu Fishki

System fiszki ma możliwość generowania definicji na podstawie słowa, wykorzystując Chat GPT, taka funkcjonalność niesie ze sobą różne zagrożenia związane z rozwojem umysłowym odbiorcy. Korzystanie z generowania treści może doprowadzić do rozleniwienia użytkownika. Odbiorca, wykorzystując Chat GPT do tworzenia definicji może nie podejmować żadnej refleksji, czy dostarczona treść ma jakikolwiek sens. W przypadku, w którym użytkownik nie jest zaznajomiony z tematem lub zagadnieniami może doprowadzić do sytuacji, w której wygenerowana

¹3, ciekawe.

treść będzie niepoprawna, ale odbiorca ze względu na brak wiedzy nie będzie tego świadomy. Chat GPT potrafi generować nieprawdziwe informacje co może spowodować nauczenie się przez użytkownika zmyślonych treści. Sytuacja, gdy użytkownik jest zaznajomiony z materiałem do nauki i jest w stanie ocenić jakość generowanych treści też niekoniecznie musi być lepsza, ponieważ odbiorca, tworząc samemu treści fiszek utrwała sobie zagadnienia. Bezmyślne zapamiętywanie informacji to kolejne zagrożenie, jakie może pojawić się w czasie nauki nowego materiału. Uczenie się na pamięć treści bez dogłębniego zrozumienia ich, sprawia, że zagadnienie jest krócej pamiętałe przez użytkownika, a także doprowadza do sytuacji, w której użytkownik nie rozumie i nie wie jak zastosować przyswojone informacje. Powyższe problemy występują także w momencie, w którym użytkownik korzysta z zestawu fiszek utworzonych przez kogoś innego.

Rozdział 5

Organizacja pracy

5.1 Harmonogram pracy

Październik 2023

Jakub Żurawski	Daniel Klimowski	Oliwier Kossak	Wiktor Krieger
Omawianie z członkami zaspołu schematu budowy aplikacji np. jakich narzędzi użyć, czy gdzie monitorować postęp	Omawianie z członkami zaspołu schematu budowy aplikacji np. jakich narzędzi użyć, czy gdzie monitorować postęp	Omawianie z członkami zaspołu schematu budowy aplikacji np. jakich narzędzi użyć, czy gdzie monitorować postęp	Omawianie z członkami zaspołu schematu budowy aplikacji np. jakich narzędzi użyć, czy gdzie monitorować postęp

Listopad-Grudzień 2023

Jakub Żurawski	Daniel Klimowski	Oliwier Kossak	Wiktor Krieger

Spotakania organizacyjne	Spotakania organizacyjne	Spotakania organizacyjne	Spotakania organizacyjne
Wypełnianie dokumentacji technicznej	Wypełnianie dokumentacji technicznej i utworzenie diagram architektury	Wypełnianie dokumentacji technicznej	Wypełnianie dokumentacji technicznej
Tworzenie diagramów	Analiza kosztów i doboru odpowiedniego dostawcy usług chmurowych	Tworzenie mocków aplikacji mobilnej	Tworzenie mocków aplikacji aplikacji

Styczeń 2024

Jakub Żurawski	Daniel Klimowski	Oliwier Kossak	Wiktor Krieger
Spotakania organizacyjne	Spotakania organizacyjne	Spotakania organizacyjne	Spotakania organizacyjne
Poprawa i aktualizacja dokumentacji technicznej			
Poprawa diagramów	Nauka Javascriptu i Reacta	Tworzenie mocków aplikacji mobilnej	Tworzenie mocków aplikacji aplikacji

Utworzenie wstępnego uruchamiającego się api, opakowanie api w konterze dockerowym	Konfiguracja środowiska iOS	Dodanie modeli dla api i utworzenie connectora do bazy danych	Przegląd artykułów naukowych
--	-----------------------------	---	------------------------------

Luty-Marzec 2024

Jakub Żurawski	Daniel Klimowski	Oliwier Kossak	Wiktor Krieger
Spotkania organizacyjne	Spotkania organizacyjne	Spotkania organizacyjne	Spotkania organizacyjne
Implementacji komend terminalowych (cli) i poprawa dependencji RoleChecker'a	Tworzenie ekranów aplikacji mobilnej (logowania, rejestracji i przypomnienia hasła)	Naprawa środowiska FastAPI	Konfiguracja środowiska
Poprawa bezpieczeństwa api, poprawa middleware'ów, dodanie loggera i endpointów autoryzujących	Konspekt pracy dyplomowej	Poprawa działania token autoryzującego i blacklist tokens	Utworzenie okno logowania (webówka)

Stawianie i konfiguracja środowiska Android	Utworzenie strony domowej dla aplikacji mobilnej	Dodanie endpointów dla flash cards	Projektowanie ikon
Naprawa aplikacji mobilnej, poprawa doboru wersji potrzebnych paczek	Piszanie rodzaju drugiego 'Omówienie problemu'	Poprawki w modelach bazy danych, dodanie crudów do endpointu deck	Poprawa mocków aplikacji webowej
Implementacja metody request z interfejsem, i dodanie paneli użytkownika	Czyszczenie kodu, przerzucenie regex'ów do katalogi validator	Utworzenie kontenera dla aplikacji webowej	Utworzenie tła aplikacji webowej

Kwiecień-Maj 2024

Jakub Żurawski	Daniel Klimowski	Oliwier Kossak	Wiktor Krieger
Pisanie działu 5	Pisanie działów 2, 3, 4	Pisanie działów 3, 4, 5, 10 pracy	Tworzenie ikon

Utworzenie widoku dla aktualizacji danych użytkownika	Utworzenie widoku my decks	Utworzenie strony tworzenia fiszek, utworzenie dla nich endpointów i połączenia jej całości z api	Utworzenie widoku profilu użytkownika dla aplikacji webowej
To be continue...	To be continue...	To be continue...	To be continue...

5.2 Wybrana metodyka

Zastosowaną w projekcie metodyką jest **model przyrostowo-ewolucyjny**. Jest to połączeniem dwóch modeli, przyrostowego i ewolucyjnego. W podjętym projekcie implementacja kodu odbywała się w trakcie sprintu, a każdy kolejny sprint był planowany po zakończeniu aktualnie prowadzonego (przerwa między sprintami), razem z wypełnianiem dokumentacji. Dzięki temu projekt był rozwijany i ewoluował cały czas, nie tylko w sprintach ale, i w trakcie przerw. Zalety takiego rozwiązania to przede wszystkim łatwe podejmowanie działań nad rozwojem aplikacji, wcześnie i stopniową dystrybucją oprogramowania, stały wgląd nad wyglądem oraz funkcjonowaniem aplikacji, czy możliwość nanoszenia poprawek w trakcie produkcji w związku ze zmianami wymagań.

5.3 Zespół i podział obowiązków

Daniel Klimowski:

- Budowanie aplikacji mobilnej;
- Konfiguracja Azure;
- Wypełnianie dokumentacji technicznej.

Oliwier Kossak:

- Budowa aplikacji webowej;
- Budowa aplikacji backendowej;
- Tworzenie modeli sql;
- Wypełnianie dokumentacji technicznej;
- Mockupy aplikacji mobilnej.

Wiktor Krieger:

- Budowa aplikacji webowej;
- Mockupy aplikacji webowej i mobilnej;
- Wypełnianie dokumentacji technicznej;
- Tworzenie ikon dla aplikacji.

Jakub Żurawski:

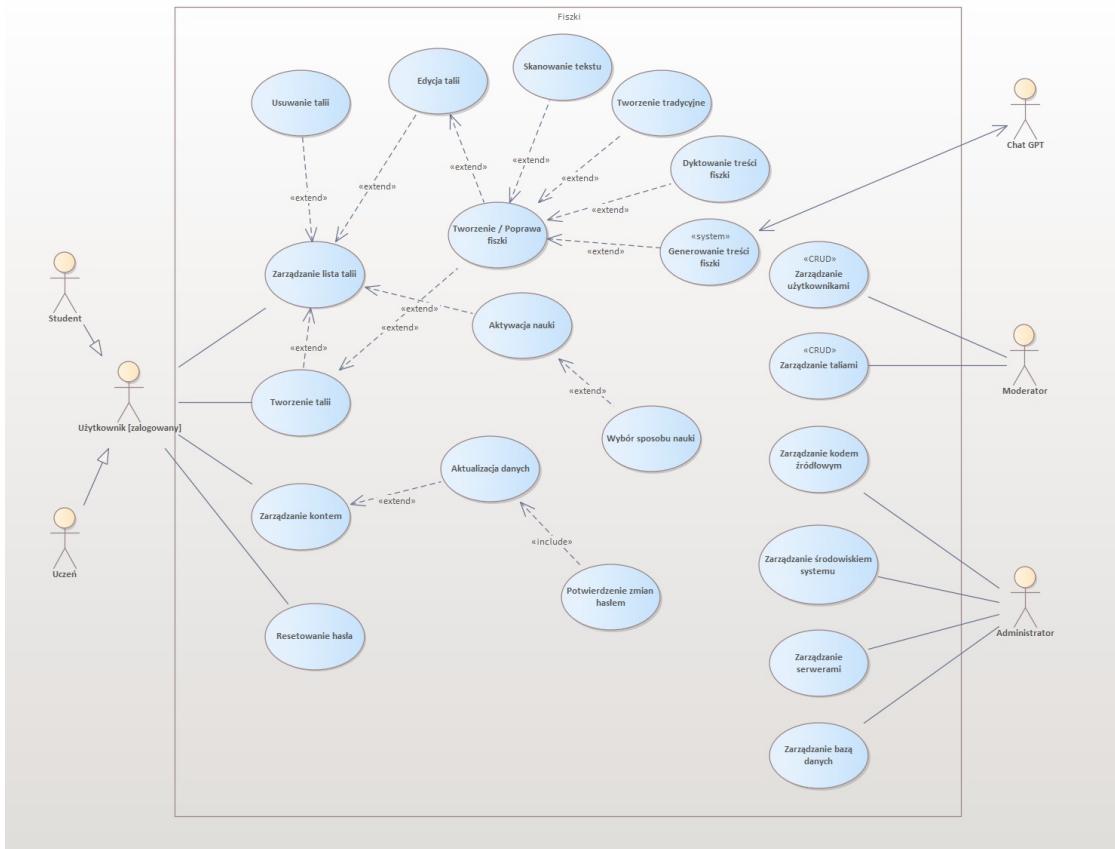
- Budowa aplikacji mobilnej;
- Budowa aplikacji backendowej;
- Wypełnianie dokumentacji technicznej;
- Tworzenie diagramów;
- Tworzenie modeli sql.

Rozdział 6

Analiza wymagań

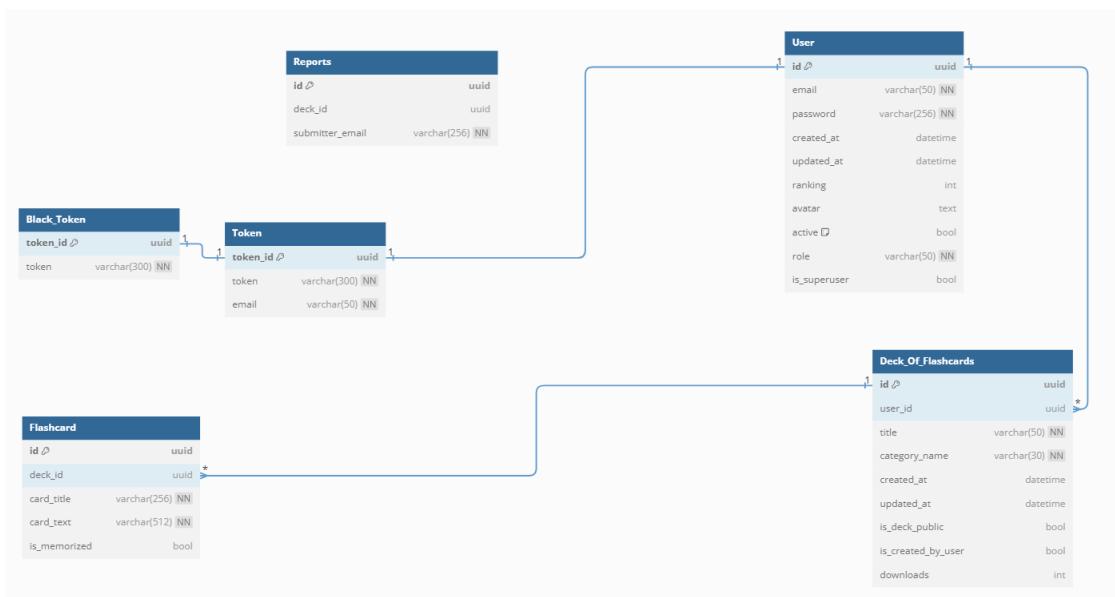
6.1 Diagramy

Diagram przypadków użycia jest narzędziem służącym do przedstawienia funkcjonalności systemu z punktu widzenia użytkownika.



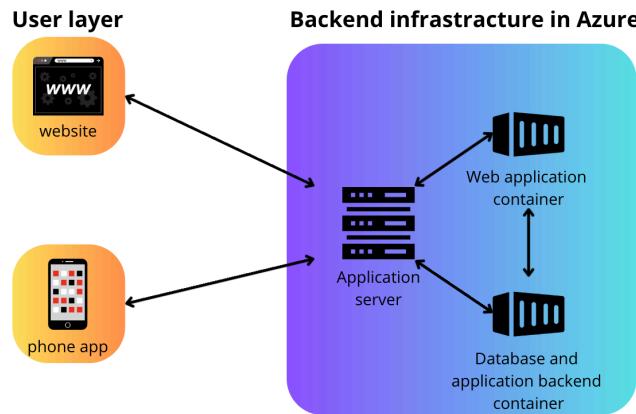
Rysunek 6.1: Diagram przypadków użycia.

Diagram ERD (Entity-Relationship Diagram) to graficzne przedstawienie struktury bazy danych używane w projektowaniu i modelowaniu baz danych.



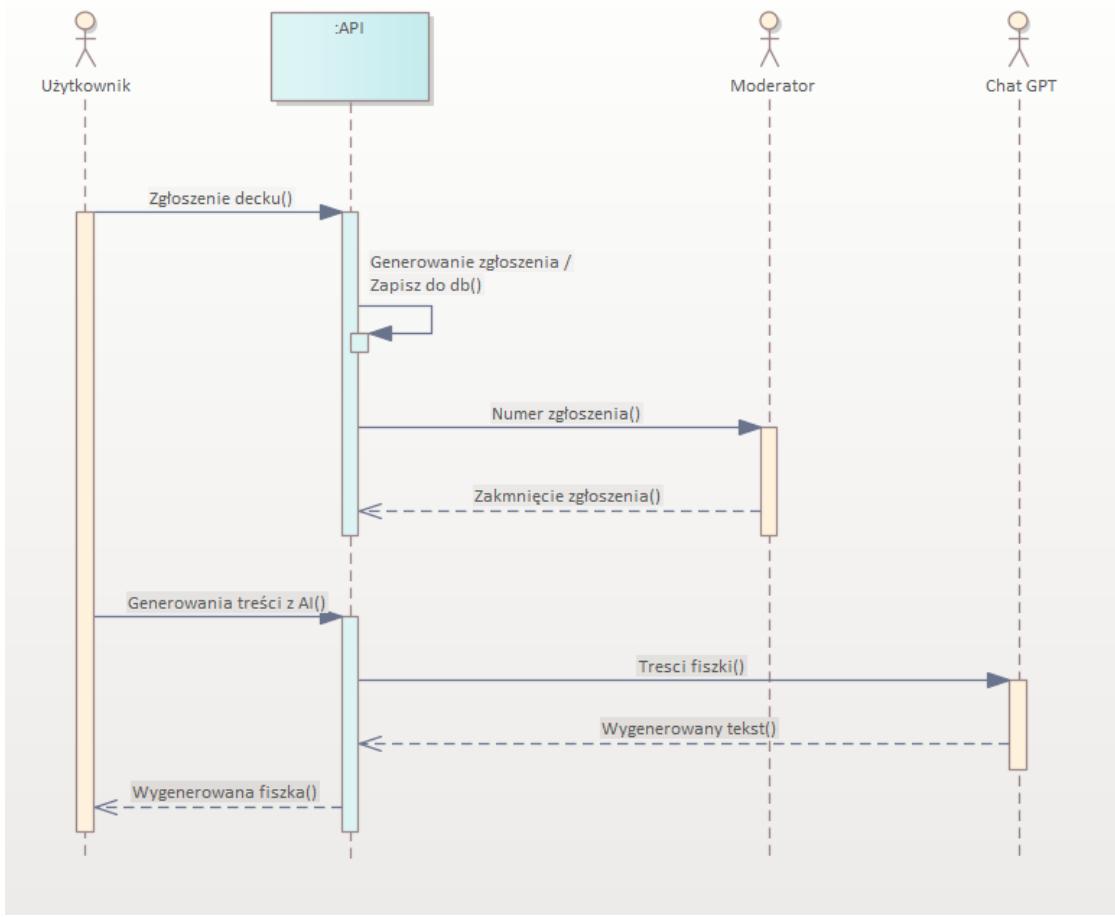
Rysunek 6.2: Diagram ERD.

Diagram architektury stanowi reprezentację organizacji systemu informatycznego w oparciu o który został zrealizowany podjęty projekt.



Rysunek 6.3: Diagram architektury.

Diagram sekwencji przedstawia interakcję między obiektami w porządku czasowym.



Rysunek 6.4: Diagram sekwencji.

6.2 Wymagania ogólne

6.2.1 Moduł autoryzacji

KARTA WYMAGANIA			
Identyfikator:	WO1	Priorytet:	M – must
Nazwa:	Moduł autoryzacji		
Opis:	Rejestracja konta nowego użytkownika. Możliwość usunięcia konta użytkownika. Logowanie do systemu. Wylogowanie użytkownika z systemu. Edycja danych użytkownika.		
Udziałowiec:	Zespół projektowy (UOB 01) Użytkownik systemu (UOB 03)		
Wymagania powiązane:	Rejestracja konta użytkownika (WF01) Logowanie do systemu (WF02) Wylogowanie z systemu (WF03) Edycja danych użytkownika (WF04) Usunięcie konta użytkownika (WF05) Weryfikacja konta użytkownika poprzez email (WF14)		

Tabela 6.1: Przykładowe wymaganie ogólne lub dziedzinowe

6.2.2 Moduł zarządzania talią

KARTA WYMAGANIA			
Identyfikator:	WO2	Priorytet:	M – must
Nazwa:	Moduł zarządzania talią		
Opis:	Tworzenie talii fiszek. Pobranie talii utworzonej przez innych użytkowników. Edycja utworzonych talii fiszek lub edycja talii pobranej przez innych użytkowników. Usunięcie talii fiszek.		
Udziałowiec:	Zespół projektowy (UOB 01) Użytkownik systemu (UOB 03)		
Wymagania powiązane:	Tworzenie talii fiszek (WF06) Usunięcie talii fiszek (WF07) Edycja talii fiszek (WF08) Import talii innych użytkowników (WF11)		

Tabela 6.2: Wymaganie ogólne dla modułu zarządzania talią

6.2.3 Moduł uczenia

KARTA WYMAGANIA			
Identyfikator:	W03	Priorytet:	M – must
Nazwa:	Moduł uczenia		
Opis:	Tryb sterowania głosem pozwala na uruchomienie talii fiszek w specjalnym trybie, który pozwala na sterowanie talią fiszek przy wykorzystaniu komend głosowych. Zwykły tryb uczenia uruchamia talie w pełnym ekranie i pozwala na podzielenie talii na fiszki, które użytkownik zapamiętał i na te nie zapamiętane.		
Udziałowiec:	Zespół projektowy (UOB 01) Użytkownik systemu (UOB 03)		
Wymagania powiązane:	Tryb uczenia się z talii fiszek (WF09) Sterowanie talią przy użyciu mowy (WF10)		

Tabela 6.3: Wymaganie ogólne dla modułu uczenia

6.2.4 Moduł udogodnień

KARTA WYMAGANIA			
Identyfikator:	W04	Priorytet:	S – should)
Nazwa:	Moduł udogodnień		
Opis:	Tryb dark mode i light mode pozwala użytkownikowi na zmianę kolorystyki systemu w celu uniknięcia przemęczenia wzroku. Kalendarz oznacza dni, w których użytkownik korzysta z aplikacji, może to spowodować większą motywację u użytkownika aby regularnie korzystał z systemu.		
Udziałowiec:	Zespół projektowy (UOB 01) Użytkownik systemu (UOB 03)		
Wymagania powiązane:	Tryb dark mode i light mode (WF12) Kalendarz śledzący aktywność (WF13)		

Tabela 6.4: Wymaganie ogólne dla modułu udogodnień

6.2.5 Moduł wspomagania tworzenia talii

KARTA WYMAGANIA		
Identyfikator:	W05	Priorytet: W – won't
Nazwa:	Moduł wspomagania tworzenia talii	
Opis:	Analiza dokumentów poprzez wykorzystanie sztucznej inteligencji pozwala na wygenerowanie zestawu fiszek poprzez wgranie do systemu dokumentu w formacie csv lub pdf. Użytkownik może wygenerować treść fiszki, podając kilka słów kluczowych na podstawie których sztuczna inteligencja przeszukuje bazę definicji i na podstawie wyszukanych definicji generuje treść.	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Tworzenie talii fiszek przez analizę dokumentu (WF15) Generowanie treści fiszki na podstawie słów kluczowych (WF16)	

Tabela 6.5: Wymaganie ogólne dla modułu wspomagania tworzenia talii

6.3 Wymagania funkcjonalne

6.3.1 Rejestracja konta użytkownika

KARTA WYMAGANIA		
Identyfikator:	WF01	Priorytet: M – must
Nazwa:	Rejestracja nowego użytkownika w systemie	
Opis:	Użytkownik musi utworzyć konto aby móc korzystać z aplikacji.	
Kryteria akceptacji:	Nowy użytkownik dodany do systemu	
Dane wejściowe:	nickname, email, hasło	
Warunki początkowe:	Użytkownik musi posiadać adres email	
Warunki końcowe:	Utworzenie konta użytkownika	
Sytuacje wyjątkowe:	Brak łączności z bazą danych. Wprowadzenie dane przez użytkownika istnieją już w bazie danych.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Zespół projektowy (UOB 01) Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Logowanie do systemu (WF02) Wylogowanie z systemu (WF03) Edycja danych użytkownika (WF04) Usunięcie konta użytkownika (WF05) Weryfikacja konta użytkownika poprzez email (WF14)	

Tabela 6.6: Wymagania na interfejs z otoczeniem dla procesu rejestracji użytkownika

6.3.2 Logowanie do systemu

KARTA WYMAGANIA		
Identyfikator:	WF02	Priorytet: M – must
Nazwa:	Logowanie do systemu	
Opis:	Użytkownik musi zalogować się do systemu, aby mieć dostęp do systemu.	
Kryteria akceptacji:	Pomyślne zalogowanie do systemu	
Dane wejściowe:	email i hasło	
Warunki początkowe:	Posiadane konto użytkownika, konto musi być aktywne.	
Warunki końcowe:	Dostęp do systemu	
Sytuacje wyjątkowe:	Brak łączności z bazą danych. Wprowadzenie niepoprawnych danych logowania, co uniemożliwia zalogowanie do systemu.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:	Rejestracja konta użytkownika (WF01) Wylogowanie z systemu (WF03) Edycja danych użytkownika (WF04) Usunięcie konta użytkownika (WF05) Weryfikacja konta użytkownika poprzez email (WF14)	

Tabela 6.7: Wymagania na interfejs z otoczeniem dla procesu logowania do systemu

6.3.3 Wylogowanie z systemu

KARTA WYMAGANIA		
Identyfikator:	WF03	Priorytet: M – must
Nazwa:	Wylogowanie z systemu	
Opis:	Jako użytkownik chciałbym mieć możliwość wylogowania z systemu, aby inne osoby nie mogły korzystać z mojego konta.	
Kryteria akceptacji:	Wylogowanie użytkownika z systemu.	
Dane wejściowe:	Brak	
Warunki początkowe:	Użytkownik zalogowany do systemu	
Warunki końcowe:	Wylogowanie z systemu	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Rejestracja konta użytkownika (WF01) Logowanie do systemu (WF02) Edycja danych użytkownika (WF04) Usunięcie konta użytkownika (WF05) Weryfikacja konta użytkownika poprzez email (WF14)	

Tabela 6.8: Wymagania na interfejs z otoczeniem dla procesu wylogowania z systemu

6.3.4 Edycja danych użytkownika

KARTA WYMAGANIA		
Identyfikator:	WF04	Priorytet: M – must
Nazwa:	Edycja danych użytkownika	
Opis:	Jako użytkownik muszę mieć możliwość zmiany hasła lub email ponieważ mogę stracić dostęp do konta email lub moje hasło z różnych powodów może stać się jawne.	
Kryteria akceptacji:	Zmienione parametry logowania	
Dane wejściowe:	Użytkownik podaje nowy parametr wraz z hasłem, w celu edycji danych użytkownika	
Warunki początkowe:	Posiadane konto użytkownika, użytkownik zalogowany do systemu.	
Warunki końcowe:	Zmienione parametry logowania	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą. Wprowadzenie niepoprawnych danych co uniemożliwia zmianę parametrów użytkownika.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Rejestracja konta użytkownika (WF01) Logowanie do systemu (WF02) Wylogowanie z systemu (WF03) Usunięcie konta użytkownika (WF05) Weryfikacja konta użytkownika poprzez email (WF14)	

Tabela 6.9: Wymagania na interfejs z otoczeniem dla procesu edycji danych użytkownika

6.3.5 Usunięcie konta użytkownika

KARTA WYMAGANIA		
Identyfikator:	WF05	Priorytet: M – must
Nazwa:	Usunięcie konta użytkownika	
Opis:	Jako użytkownik chcę mieć możliwość usunięcia swojego konta, gdy przestanę korzystać z aplikacji.	
Kryteria akceptacji:	Usunięte konto użytkownika	
Dane wejściowe:	email i hasło	
Warunki początkowe:	Posiadane konto użytkownika, użytkownik zalogowany do systemu.	
Warunki końcowe:	Konto użytkownika usunięte z systemu	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą. Wprowadzenie niepoprawnych danych, co uniemożliwia usunięcie konta użytkownika.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Rejestracja konta użytkownika (WF01) Logowanie do systemu (WF02) Wylogowanie z systemu (WF03) Edycja danych użytkownika (WF04) Weryfikacja konta użytkownika poprzez email (WF14)	

Tabela 6.10: Wymagania na interfejs z otoczeniem dla procesu usunięcia konta użytkownika

6.3.6 Tworzenie talii fiszek

KARTA WYMAGANIA		
Identyfikator:	WF06	Priorytet: M – must
Nazwa:	Tworzenie talii fiszek	
Opis:	Jako użytkownik muszę mieć możliwość utworzenia własnej talii fiszek, w celu uczenia się zagadnień, które mają dla mnie znaczenie.	
Kryteria akceptacji:	Utworzona talia fiszek	
Dane wejściowe:	tytuł talii fiszek, kategoria talii, tytuł fiszki, treść fiszki	
Warunki początkowe:	Posiadane konto użytkownika, użytkownik zalogowany do systemu.	
Warunki końcowe:	Utworzona nowa talia fiszek.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą. Puste pole z nazwą talii uniemożliwia tworzenie talii. Pusty tytuł fiszki lub pusta treść uniemożliwia utworzenie fiszki.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Usunięcie talii fiszek (WF07) Edycja talii fiszek (WF08) Import talii innych użytkowników (WF11)	

Tabela 6.11: Wymagania na interfejs z otoczeniem dla procesu tworzenia talii fiszek

6.3.7 Usunięcie talii fiszek

KARTA WYMAGANIA		
Identyfikator:	WF07	Priorytet: M – must
Nazwa:	Usunięcie talii fiszek	
Opis:	Jako użytkownik chcę mieć możliwość usunięcia talii fiszek, z których nie będę już korzystał.	
Kryteria akceptacji:	Usunięta talia fiszek	
Dane wejściowe:	Brak	
Warunki początkowe:	Posiadane konto użytkownika, użytkownik zalogowany do systemu. Utworzona talia fiszek, którą można usunąć.	
Warunki końcowe:	Talia fiszek zostaje usunięta.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Tworzenie talii fiszek (WF06) Edycja talii fiszek (WF08) Import talii innych użytkowników (WF11)	

Tabela 6.12: Wymagania na interfejs z otoczeniem dla procesu usunięcia talii fiszek

6.3.8 Edycja talii fiszek

KARTA WYMAGANIA		
Identyfikator:	WF08	Priorytet: M – must
Nazwa:	Edycja talii fiszek	
Opis:	Jako użytkownik chcę mieć możliwość edycji talii fiszek, aby zaktualizować informacje dotyczące talii.	
Kryteria akceptacji:	Zaktualizowana zawartość talii	
Dane wejściowe:	Nowy tytuł talii, kategoria lub zmieniona treść fiszki.	
Warunki początkowe:	Talia fiszek, która będzie edytowana.	
Warunki końcowe:	Talia fiszek zostaje zaktualizowana o nowe informacje.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą. Puste pole dotyczące tytułu talii uniemożliwia zapisanie zmian.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Tworzenie talii fiszek (WF06) Usunięcie talii fiszek (WF07) Import talii innych użytkowników (WF11)	

Tabela 6.13: Wymagania na interfejs z otoczeniem dla procesu edycji talii fiszek

6.3.9 Tryb uczenia się z talii fiszek

KARTA WYMAGANIA		
Identyfikator:	WF09	Priorytet: M – must
Nazwa:	Tryb uczenia się z talii fiszek	
Opis:	System musi posiadać tryb uczenia się, który pozwoli użytkownikowi na naukę i zapamiętywanie zagadnień znajdujących się w talii fiszek. W trakcie nauki użytkownik dzieli talię na zestaw z pojęciami które już opanował i na te, których jeszcze nie pamięta.	
Kryteria akceptacji:	Talia fiszek zostaje podzielona na dwa zbiory, fiszki zapamiętane i nie zapamiętanego.	
Dane wejściowe:	Brak	
Warunki początkowe:	Utworzona talia fiszek lub pobrana talia fiszek, która zostanie wykorzystana do nauki.	
Warunki końcowe:	Talia fiszek zostaje podzielona na dwa zbiory, fiszki zapamiętane i nie zapamiętanego.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:	Sterowanie talią przy użyciu mowy (WF10)	

Tabela 6.14: Wymagania na interfejs z otoczeniem dla trybu uczenia się z talii fiszek

6.3.10 Sterowanie talią przy użyciu mowy

KARTA WYMAGANIA		
Identyfikator:	WF10	Priorytet: M – must
Nazwa:	Sterowanie talią przy użyciu mowy	
Opis:	Jako użytkownik chcę mieć możliwość uczenia się w warunkach, w których odczytywanie treści fiszki jest utrudnione, na przykład w trakcie prowadzenia samochodu. Sterowanie talią z przy użyciu mowy i odczytanie zawartości fiszki przez sztuczną inteligencję pozwala na naukę podczas prowadzenia pojazdu.	
Kryteria akceptacji:	Użytkownik steruje talią fiszek przy użyciu komend głosowych	
Dane wejściowe:	Brak	
Warunki początkowe:	Utworzona talia fiszek	
Warunki końcowe:	Talia fiszek pozostaje niezmieniona	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą danych. Hałas, który aplikacja może przechwytywać, przez co komendy głosowe mogą działać niepoprawnie. Fiszki utworzone w innym języku niż angielski co może spowodować trudności w odczytaniu ich zawartości przez syntezator mowy.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Tryb uczenia się z talią fiszek (WF09)	

Tabela 6.15: Wymagania na interfejs z otoczeniem dla sterowania talią przy użyciu mowy

6.3.11 Import talii innych użytkowników

KARTA WYMAGANIA		
Identyfikator:	WF11	Priorytet: M – must
Nazwa:	Import talii innych użytkowników	
Opis:	Jako użytkownik chcę mieć możliwość pobrania talii od innych użytkowników, aby mieć łatwy dostęp do treści i zagadnień, które mnie interesują.	
Kryteria akceptacji:	Talia dodana do zakładki talii pobranych od innych użytkowników.	
Dane wejściowe:	Brak	
Warunki początkowe:	Talia którą użytkownik może pobrać.	
Warunki końcowe:	Talia dodana do zakładki talii pobranych od innych użytkowników.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą danych.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Tworzenie talii fiszek (WF06) Usunięcie talii fiszek (WF07) Edycja talii fiszek (WF08)	

Tabela 6.16: Wymagania na interfejs z otoczeniem dla procesu importowania talii fiszek od innych użytkowników

6.3.12 Tryb dark mode i light mode

KARTA WYMAGANIA		
Identyfikator:	WF12	Priorytet: S – should
Nazwa:	Tryb dark mode i light mode	
Opis:	Jako użytkownik chciałbym mieć możliwość zmiany kontrolowania jasności systemu aby mój wzrok się nie zmęczał.	
Kryteria akceptacji:	Zmiana koloru interfejsu aplikacji.	
Dane wejściowe:	Brak	
Warunki początkowe:	Posiadane konto użytkownika, użytkownik zalogowany do systemu.	
Warunki końcowe:	Zmiana koloru interfejsu aplikacji.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą danych.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Kalendarz śledzący aktywność (WF13)	

Tabela 6.17: Wymagania na interfejs z otoczeniem dla trybu dark mode i light mode

6.3.13 Kalendarz śledzący aktywność

KARTA WYMAGANIA		
Identyfikator:	WF13	Priorytet: S – should
Nazwa:	Kalendarz śledzący aktywność	
Opis:	Kalendarz odznaczający dni, w których użytkownik korzystał z aplikacji, mógłby zwiększyć motywację użytkownika do regularnego korzystania z aplikacji.	
Kryteria akceptacji:	Kalendarz oznacza dzień w momencie uruchomienia przez użytkownika aplikacji.	
Dane wejściowe:	Brak	
Warunki początkowe:	Posiadane konto użytkownika, użytkownik zalogowany do systemu.	
Warunki końcowe:	Dzień oznaczony w kalendarzu.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą danych.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:	Tryb dark mode i light mode (WF12)	

Tabela 6.18: Wymagania na interfejs z otoczeniem dla kalendarza śledzącego aktywność użytkownika

6.3.14 Weryfikacja konta użytkownika poprzez email

KARTA WYMAGANIA		
Identyfikator:	WF14	Priorytet: C – could
Nazwa:	Weryfikacja konta użytkownika poprzez email	
Opis:	Użytkownik po pomyślniej rejestracji, otrzyma na email wiadomość z linkiem przekierowującym na stronę webową, gdzie nastąpi wysłanie żądania o aktywację konta użytkownika.	
Kryteria akceptacji:	Wiadomość dostarczona na wskazany email, poprawnie działająca aktywacja konta, poprawne przekierowanie.	
Dane wejściowe:	Token autoryzujący	
Warunki początkowe:	Użytkownik musi posiadać istniejący adres email, na który przyjdzie wiadomość z linkiem aktywującym konto.	
Warunki końcowe:	Użytkownik aktywuje konto.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą danych. Użytkownik nie ma dostępu do podanego przez siebie adresu email lub podany adres jest błędny. Wiadomość nie dotarła do użytkownika.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:	Rejestracja konta użytkownika (WF01) Logowanie do systemu (WF02)	

Tabela 6.19: Wymagania na interfejs z otoczeniem dla weryfikacji konta użytkownika poprzez email

6.3.15 Tworzenie talii fiszek poprzez zeskanowanie dokumentu

KARTA WYMAGANIA		
Identyfikator:	WF15	Priorytet: W – won't
Nazwa:	Tworzenie talii fiszek poprzez zeskanowanie dokumentu	
Opis:	Jako użytkownik chciałbym mieć możliwość szybkiego utworzenia zestawu talii fiszek poprzez wgranie gotowego dokumentu w formacie csv lub pdf z którego zostałaby utworzona talia fiszek na podstawie zagadnień znajdujących się w dokumencie.	
Kryteria akceptacji:	Talia fiszek utworzona po analizie dokumentu.	
Dane wejściowe:	Dokument do analizy w formacie csv lub pdf.	
Warunki początkowe:	Posiadane konto użytkownika, użytkownik zalogowany do systemu, dokument do analizy.	
Warunki końcowe:	Utworzona talia fiszek.	
Sytuacje wyjątkowe:	Awaria bazy danych lub brak połączenia z bazą danych. Dokument nie zdatny do odczytu.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Generowanie treści fiszki na podstawie słów kluczowych (WF16)	

Tabela 6.20: Wymagania na interfejs z otoczeniem dla tworzenia talii fiszek poprzez zeskanowanie dokumentu

6.3.16 Generowanie treści fiszki na podstawie słów kluczowych

KARTA WYMAGANIA		
Identyfikator:	WF16	Priorytet: M – must
Nazwa:	Generowanie treści fiszki na podstawie słów kluczowych	
Opis:	Jako użytkownik chcę, aby system potrafił generować definicje zagadnień, które wpisałem na pierwszą stronę karty, co ułatwi tworzenie talii.	
Kryteria akceptacji:	Zawartość fiszki wygenerowana przy pomocy zewnętrznego API na podstawie podanej treści na stronie polu dla przedniej strony fiszki.	
Dane wejściowe:	Brak	
Warunki początkowe:	Utworzone konto użytkownika.	
Warunki końcowe:	Treść fiszki wygenerowana na podstawie zawartości wpisanej w pole przedniej strony karty.	
Sytuacje wyjątkowe:	Awaria API lub połączenia z API. Podanie przez użytkownika zdania nie mającego sensu.	
Szczegóły implementacji:	uzupełniane w trakcie sprintu – opis sposobu realizacji	
Udziałowiec:	Użytkownik systemu (UOB 03)	
Wymagania powiązane:	Tworzenie talii fiszek przez analizę dokumentu (WF15)	

Tabela 6.21: Wymagania na interfejs z otoczeniem dla generowania treści fiszki na podstawie słów kluczowych

6.3.17 Resetowanie hasła na konta

KARTA WYMAGANIA		
Identyfikator:	WF017	Priorytet: S – should
Nazwa:	Resetowanie hasła na konta	
Opis:		Użytkownik po podaniu adresu mailowego połączonego z kontem, otrzyma na email wiadomość z linkiem przekierowującym na stronę webową, gdzie będzie mógł podać nowe hasło.
Kryteria akceptacji:		Pomyślna zmiana hasła.
Dane wejściowe:		Token autoryzujący, nowe hasło
Warunki początkowe:		Użytkownik musi posiadać istniejący adres email, na który przyjdzie wiadomość z linkiem przekierowującym na widok strony internetowej.
Warunki końcowe:		Użytkownik pomyślnie zmienia hasło.
Sytuacje wyjątkowe:		Awaria bazy danych lub brak połączenia z bazą danych. Użytkownik nie ma dostępu do podanego przez siebie adresu email lub podany adres jest błędny. Wiadomość nie dotarła do użytkownika.
Szczegóły implementacji:		uzupełniane w trakcie sprintu – opis sposobu realizacji
Udziałowiec:		Zespół projektowy (UOB 01)
Wymagania powiązane:		Logowanie do systemu (WF02)

Tabela 6.22: Wymagania na interfejs z otoczeniem dla resetowania hasła

6.4 Interfejs z otoczeniem

KARTA WYMAGANIA		
Identyfikator:	I01	Priorytet: S – should
Nazwa:	Chat GPT 3.5	
Opis:		Aplikacja zostaje połączona z API chatu GPT, aby użytkownik miał możliwość wygenerowania treści fiszki na podstawie podanego słowa.
Kryteria akceptacji:		Prawidłowo generuje definicje dla wysyłanych słów kluczowych.
Dane wejściowe:		Słowo podane przez użytkownika.
Warunki początkowe:		Aplikacja połączona z API chatu GPT.
Warunki końcowe:		Chat GPT zwraca definicję podanego słowa.
Sytuacje wyjątkowe:		Brak połączenia z Chat GPT.
Szczegóły implementacji:		uzupełniane w trakcie sprintu – opis sposobu realizacji
Udziałowiec:		Zespół projektowy - UOB 01
Wymagania powiązane:		

Tabela 6.23: Wymagania na interfejs z otoczeniem dla integracji z Chat GPT 3.5

6.5 Wymagania pozafunkcjonalne

6.5.1 Instrukcja korzystania z trybu sterowania głosem

KARTA WYMAGANIA		
Identyfikator:	WN01	Priorytet: M – must
Nazwa:	Instrukcja korzystania z trybu sterowania głosem	
Opis:	Instrukcja zawiera komendy, wraz z opisem, których użytkownik korzysta w momencie uruchomienia trybu sterowania głosem.	
Kryteria akceptacji:	Nowy użytkownik po zapoznaniu się z instrukcją, jest w stanie korzystać z trybu sterowania głosem.	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:	Sterowanie talią przy użyciu mowy (WF10)	

Tabela 6.24: Wymagania na interfejs z otoczeniem dla instrukcji korzystania z trybu sterowania głosem

6.5.2 Limit błędów dotyczących logowania

KARTA WYMAGANIA		
Identyfikator:	WN02	Priorytet: S – should
Nazwa:	Limit błędów dotyczących logowania	
Opis:	Po wpisaniu 3 razy niepoprawnych danych logowania w aplikacji pojawi się okienko informujące o blokadzie aplikacji na 1 minutę, okienko zawiera przycisk zmiany hasła. Przycisk przekierowuje do podstrony zawierającej pole do podania email na które przyjdzie wiadomość dotycząca zmiany hasła.	
Kryteria akceptacji:	Użytkownik otrzymuje wiadomość email, która przekierowuje go do formularza zmiany hasła.	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:	Resetowanie hasła (WF017)	

Tabela 6.25: Wymagania na interfejs z otoczeniem dla limitu błędów logowania

6.5.3 Dostępność systemu

KARTA WYMAGANIA			
Identyfikator:	WN03	Priorytet:	M – must
Nazwa:	Dostępność systemu		
Opis:	System powinien być dostępny 7 dni w tygodniu, 24 godziny na dobę.		
Kryteria akceptacji:	System działa przez 7 dni bez żadnych awarii. Co godzinę sprawdzane jest połączenie z systemem.		
Udziałowiec:	Zespół projektowy (UOB 01)		
Wymagania powiązane:			

Tabela 6.26: Wymagania na interfejs z otoczeniem dla dostępności systemu

6.5.4 Responsywność

KARTA WYMAGANIA			
Identyfikator:	WN04	Priorytet:	M – must
Nazwa:	Responsywność		
Opis:	System musi być responsywnym, dostosowując się do wielkości okienka lub urządzenia.		
Kryteria akceptacji:	System w pełni dostosuje się do dostępnej wielkości okienka.		
Udziałowiec:	Zespół projektowy (UOB 01)		
Wymagania powiązane:			

Tabela 6.27: Wymagania na interfejs z otoczeniem dla responsywności systemu

6.5.5 Kompatybilność

KARTA WYMAGANIA			
Identyfikator:	WN05	Priorytet:	M – must
Nazwa:	Kompatybilność		
Opis:	System musi być kompatybilny z różnymi przeglądarkami internetowymi.		
Kryteria akceptacji:	System w pełni działa i ukazuje się w wybranej przeglądarce.		
Udziałowiec:	Zespół projektowy (UOB 01)		
Wymagania powiązane:			

Tabela 6.28: Wymagania na interfejs z otoczeniem dla kompatybilności z przeglądarkami internetowymi

6.5.6 Hashowanie haseł

KARTA WYMAGANIA		
Identyfikator:	WN06	Priorytet: M – must
Nazwa:	Hashowanie haseł	
Opis:	Hasła użytkowników hashowane przy użyciu algorytmu sha256.	
Kryteria akceptacji:	Hasło w bazie danych zostanie zaszyfrowane.	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:	Rejestracja konta użytkownika (WF01) Logowanie do systemu (WF02)	

Tabela 6.29: Wymagania na interfejs z otoczeniem dla hashowania haseł

6.6 Wymagania na środowisko docelowe

6.6.1 Przeglądarka

KARTA WYMAGANIA		
Identyfikator:	ŚD01	Priorytet: M – must
Nazwa:	Przeglądarka	
Opis:	Chrome wersja 110.0.0.0, Firefox wersja 120.0, Microsoft Edge 115.0.0.0	
Kryteria akceptacji:	Strona internetowa uruchamia się.	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:		

Tabela 6.30: Wymagania na interfejs z otoczeniem dla przeglądarek internetowych

6.6.2 System Android 10 i iOS 16

KARTA WYMAGANIA		
Identyfikator:	ŚD02	Priorytet: M – must
Nazwa:	System Android 10 i iOS 16	
Opis:	Aplikacja działa na urządzeniach mobilnych z systemem Android w wersji 10 i wyższych, w przypadku iOS w wersji 16 i wyższych.	
Kryteria akceptacji:	Aplikacja uruchamia się na urządzeniach Android w wersji 10 i wyższych, w przypadku iOS w wersji 16 i wyższych.	
Udziałowiec:	Zespół projektowy (UOB 01)	
Wymagania powiązane:		

Tabela 6.31: Wymagania na interfejs z otoczeniem dla systemów mobilnych

6.6.3 Kontenery dockerowe

KARTA WYMAGANIA			
Identyfikator:	ŚD03	Priorytet:	M – must
Nazwa:	Kontenery dockerowe		
Opis:	Dwa kontenery: jeden zawierający backend wraz z bazą danych, drugi zawierający aplikację webową.		
Kryteria akceptacji:	Łączność pomiędzy kontenerami.		
Udziałowiec:	Zespół projektowy (UOB 01)		
Wymagania powiązane:			

Tabela 6.32: Wymagania na interfejs z otoczeniem dla kontenerów dockerowych

6.6.4 Baza danych

KARTA WYMAGANIA			
Identyfikator:	ŚD04	Priorytet:	M – must
Nazwa:	Baza danych		
Opis:	Mariadb - relacyjna baza danych w wersji 11.0. Będzie przechowywana w kontenerze razem z backend'em.		
Kryteria akceptacji:	Prawidłowo tworzące się obiekty modeli, stabilna łączność z backendem.		
Udziałowiec:	Zespół projektowy (UOB 01)		
Wymagania powiązane:			

Tabela 6.33: Wymagania na interfejs z otoczeniem dla bazy danych

6.6.5 Python

KARTA WYMAGANIA			
Identyfikator:	ŚD05	Priorytet:	M – must
Nazwa:	Python		
Opis:	Python w wersji 3.10 będzie odpowiedzialny za poprawne działanie backendu.		
Kryteria akceptacji:	Backend reaguje na otrzymywane requesty.		
Udziałowiec:	Zespół projektowy (UOB 01)		
Wymagania powiązane:			

Tabela 6.34: Wymagania na interfejs z otoczeniem dla Pythona

6.6.6 Node.js

KARTA WYMAGANIA			
Identyfikator:	ŚD06	Priorytet:	M – must
Nazwa:	Node.js		
Opis:	Minimalna wersja Node.js dla aplikacji webowej oraz mobilnej to wersja 16. Będzie on odpowiedzialny za działania aplikacji webowej, jak i mobilnej.		
Kryteria akceptacji:	Poprawne uruchomienie aplikacji webowej i mobilnej.		
Udziałowiec:	Zespół projektowy (UOB 01)		
Wymagania powiązane:			

Tabela 6.35: Wymagania na interfejs z otoczeniem dla Node.js

6.6.7 System operacyjny

KARTA WYMAGANIA			
Identyfikator:	ŚD07	Priorytet:	M – must
Nazwa:	System operacyjny		
Opis:	System operacyjny Ubuntu w wersji 20, w którym zainicjowane będą kontenery dockerowe oraz środowisko z załączem technicznym projektu.		
Kryteria akceptacji:	System stabilnie utrzymuje połączenie między użytkownikami a aplikacją, obsługuje środowisko techniczne projektu z zoptymalizowanym zużyciem zasobów oraz spełnia podstawowe wymagania zabezpieczeń bezpieczeństwa sieciowego.		
Udziałowiec:	Zespół projektowy (UOB 01)		
Wymagania powiązane:	ŚD03 ŚD04 ŚD05 ŚD06		

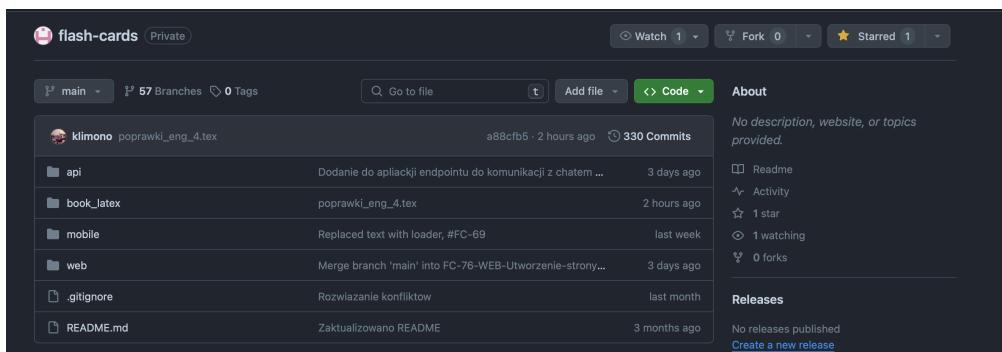
Tabela 6.36: Wymagania na interfejs z otoczeniem dla systemu operacyjnego

Rozdział 7

Architektura projektu i użyte technologie

7.1 Narzędzia organizacji pracy

Github Platforma dostarczająca usługi zdalnego przechowywania oraz kontroli wersji projektów informatycznych. Na potrzeby projektu GitHub został wykorzystany w celu utworzenia repozytorium przechowującego kod źródłowy: aplikacji webowej, aplikacji mobilnej, backendu aplikacji oraz niniejszej pracy wraz ze wszystkimi niezbędnymi assetami i dokumentacją. Repozytorium odgrywa kluczową rolę stanowiąc ważne narzędzie przy organizacji projektu o zadanej złożoności nad którym cały zespół pracuje jednocześnie.



Rysunek 7.1: Repozytorium projektu umieszone na GitHub.

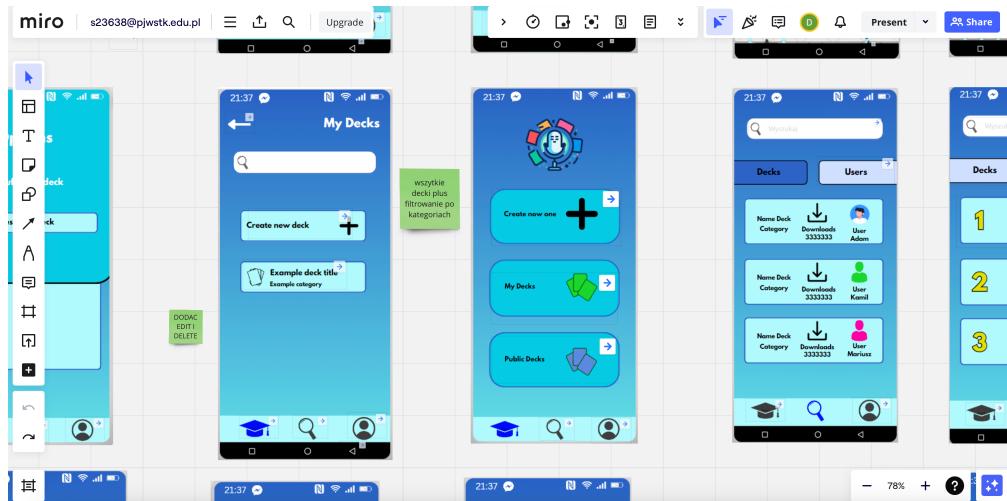
Jira Platforma służąca głównie do zarządzania projektami oraz do śledzenia powiązanych z nimi błędów. Jira została wykorzystana w celu dokumentacji, organizacji oraz planowania rozwoju systemu Fishki. Najważniejszą użytką w projekcie funkcją jest możliwość zarządzania sprintami i taskami, tj. zadaniami, bez których realizacja pracy w wybranym modelu byłaby niemożliwa.

The screenshot shows the Jira interface with the following details:

- Project Path:** Projekty / flash-cards
- Section:** Backlog
- Sprint:** FC Sprint 5 4 maj – 12 maj (20 zgłoszeń)
- Status Legend:**
 - DO ZROBienia (Red): 5 tasks
 - W TOKU (Orange): 0 tasks
 - DO ZROBienia (Green): 0 tasks
 - DO ZROBienia (Yellow): 0 tasks
 - DO ZROBienia (Blue): 0 tasks
 - DO ZROBienia (Grey): 0 tasks
 - GOTOWE (Green): 1 task
- Tasks:**
 - FC-35 [BACKEND] wstępne testy integracyjne (Status: DO ZROBienia)
 - FC-50 [Web] Profil użytkownika (Status: W TOKU)
 - FC-61 [WEB] Usunięcie konta użytkownika (Status: DO ZROBienia)
 - FC-63 [WEB] Zmiana email użytkownika (Status: DO ZROBienia)
 - FC-66 [WEB] Utworzenie strony public decks (Status: DO ZROBienia)
 - FC-68 [WEB] Utworzenie widoku dla statystyk użytkownika (Status: DO ZROBienia)
 - FC-64 [WEB] Zmiana nicku użytkownika (Status: W TOKU)
 - FC-62 [WEB] Zmiana hasła użytkownika (Status: DO ZROBienia)
 - FC-76 [WEB] Utworzenie strony do sterowania głosem talia fiszek (Status: GOTOWE)

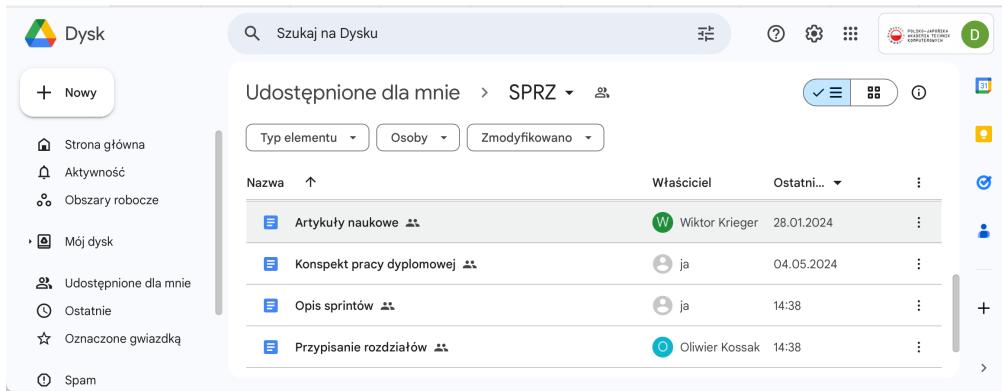
Rysunek 7.2: Sprint rozpisany przy pomocy Jira.

Miro Wirtualna tablica umożliwiająca współpracę zespołową w czasie rzeczywistym. Miro zostało użyte do połączenia ze sobą mockupów aplikacji mobilnej utworzonych w canvie.



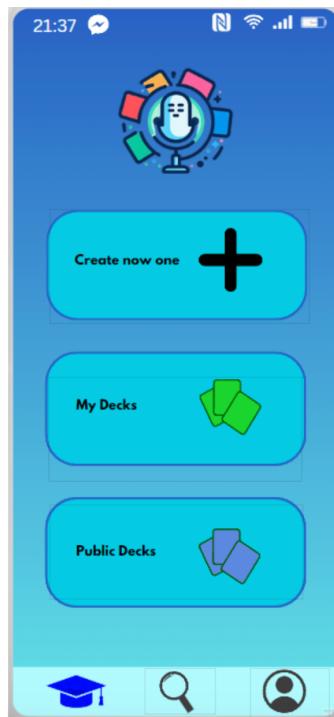
Rysunek 7.3: Tablica w Miro z interaktywnymi mockupami projektu.

Google Drive Usługa pozwalająca na przechowywanie plików wirtualnie w chmurze. Wykorzystana w projekcie do zarządzania i przechowywania niektórych dokumentów związanych z projektem.



Rysunek 7.4: Dokumenty przechowywane na Google Drive.

Canva Bezpłatne narzędzie służące do tworzenia grafik i stron internetowych. Program został użyty w celu utworzenia mockapów aplikacji moblinej oraz diagramu architektury.



Rysunek 7.5: Grafika mockupu wykonana w Canva.

Discord Bezpłatny program służący do komunikacji głosowej i tekstowej. Wykorzystywany do spotkań organizacyjnych i komunikacji członków zespołu o problemach związanych z projektem.

Enterprise Architect Program służący do modelowania diagramów. Został wykorzystany do przygotowania diagramu przypadków użycia.

dbdiagram.io Narzędzie online wykorzystywane przy projektowaniu diagramów.. W projekcie zostało użyte do stworzenia diagramu ERD.

LaTeX System składu tekstu. Wykorzystany w projekcie do organizacji oraz tworzenia książki projektu.

Git Rozproszony system kontroli wersji, wykorzystany w projekcie do zarządzania kodem.

7.2 Aplikacja webowa

React Biblioteka JavaScript wykorzystywana do tworzenia interfejsów użytkownika, pozwala ona na szybkie i łatwe kreowanie uniwersalnych komponentów, umożliwia to wykorzystanie

jednego komponentu w wielu miejscach w kodzie. Biblioteka została wykorzystana do tworzenia strony internetowej.

TypeScript Rozszerza JavaScript o możliwość typowania zmiennych. TypeScript został wykorzystany w celu szybszego wykrywania błędów w kodzie i łatwiejszym zarządzaniu go.

Material UI Popularna biblioteka UI, zawiera zestaw komponentów do tworzenia interfejsów internetowych. Używana w połączeniu z biblioteką React.js.

Node JS Środowisko, które pozwala na uruchomienie javascriptu na serwerze jako samodzielna aplikacji.

7.3 Aplikacja mobilna

React Native Framework pozwalający na tworzenie natywnych aplikacji mobilnych dla iOS i Android przy użyciu JavaScript i Reacta. Umożliwia programistom wykorzystanie jednego kodu źródłowego do budowy aplikacji na obie wymienione platformy.

Expo Framework i platforma służąca do tworzenia aplikacji na iOS i Android za pomocą React Native, która ułatwia rozwój aplikacji mobilnych

TypeScript Jw. w sekcji “Aplikacja webowa”.

7.4 Backend aplikacji

Python Wysokopoziomowy język programowania. Wykorzystany w projekcie ze względu na swoją prostotę, wszechstronność i wieloplatformowość.

FastAPI Framework do tworzenia interfejsów API w języku Python. Wykorzystany w projekcie ze względu na szybkość co pozwala na szybką komunikację z bazą danych. Framework wykorzystuje asynchroniczną obsługę żądań co pozwala aplikacji osiągnąć wysoką wydajność.

7.5 Baza danych

MariaDB Relacyjna baza danych, wykorzystana w projekcie ze względu na wysoką wydajność, a także otwartoźródłowy charakter systemu.

7.6 Infrastruktura serwerowa

Microsoft Azure Platforma chmurowa pozwalająca wdrożyć wirtualną infrastrukturę IT. Wykorzystana na rzecz utrzymania środowiska projektowego tj. kontenera z Api, backend aplikacji webowej i bazę danych w oparciu o wirtualną maszynę z systemem operacyjnym Ubuntu server.

7.7 Narzędzia programistyczne

Jetbrains PyCharm Środowisko programistyczne zaprojektowane dla języka Python. Wykorzystane w projekcie podczas projektowania i tworzenia backendu aplikacji oraz do komplikacji książki w LaTeX.

Jetbrains WebStorm Środowisko programistyczne zaprojektowane głównie dla języka JavaScript oraz technologii wykorzystywanych w budowie aplikacji webowej/mobilnej. Użyte w projektowaniu oraz tworzeniu frontenu.

Docker Narzędzie programistyczne, które umożliwia tworzenie, wdrażanie i uruchamianie aplikacji w kontenerach. Docker wykorzystany został do umieszczenia aplikacji webowej, api i bazy danych w kontenerach, aby ułatwić łatwe uruchamiania aplikacji.

Visual Studio Code Środowisko programistyczne wszechstronnego przeznaczenia. Wykorzystane na etapie tworzenia aplikacji ze względu na dodatek Thunder Client, który zastąpił popularnego Postmana, do testowania api.

Android Studio Oficjalne zintegrowane środowisko programistyczne dla systemu operacyjnego Google Android. Umożliwia ściąganie różnych wersji emulatorów, ściągania wymaganych paczek do uruchomienia danego emulatora, a także do celów testowania wytwarzanej aplikacji mobilnej.

Xcode Oficjalne środowisko programistyczne Apple udostępniające narzędzia do tworzenia aplikacji oraz oprogramowania dla systemów z rodziny iOS i macOS. Wykorzystane w projekcie w celu testowania aplikacji mobilnej w symulatorach urządzeń z zainstalowanym systemem iOS.

Adobe Illustrator Rozbudowany program graficzny przeznaczony do tworzenia i edycji grafiki wektorowej. Utworzone zostały za jego pomocą ikony oraz tło do naszych aplikacji.

Rozdział 8

Implementacja

Implementacja projektu odbyła się metodą przyrostowo-ewolucyjną. Metoda ta pozwala na szybką adaptację i elastyczność, która jest kluczowa w przypadku projektów, które mogą ulegać zmianom w czasie realizacji. Tworzenie systemu odbyło się w dwóch etapach, fazie planowanie i fazie implementacji. Faza planowania dotyczyła określenia celu projektu, funkcjonalności systemu i wypełnienia dokumentacji. Faza implementacji odbywała się w sprintach, które zazwyczaj trwały od 2 do 3 tygodni.

8.1 Faza Planowania

Na samym początku była burza mózgów w celu wymyślenia tematu naszego projektu. Po analizie potencjalnych tematów zespół zdecydował się na aplikację do uczenia się z wykorzystaniem metody fiszek. Następnym krokiem było wypełnienie karty projektu, zespół musiał określić cele projektu, miary sukcesu i główne funkcjonalności. Po określeniu ogólnych założeń dotyczących projektu, zespół zajął się wypełnieniem dokumentu założeń wstępnych, który dotyczył opisu naszego problemu, analizy konkurencji i ogólnej wizji konstrukcyjnej. Ostatnim krokiem w fazie planowania było wypełnienie specyfikacji wymagań systemowych, która dotyczyła szczegółowego opisu wymagań dotyczących naszego projektu. Po ukończeniu planowania zespół był gotowy do przejścia w fazę implementacji.

8.2 Faza Implementacji

Implementacja projektu odbyła się w sprintach, które trwało od 2 do 4 tygodni. Członkowie zespołu w ramach każdego sprintu mieli do wykonania zadania, które były przypisane do nich w narzędziu do zarządzania projektem jira.

8.2.1 Sprint 1

Opis Sprintu

Pierwszy przyrost dotyczył podstawowej konfiguracji projektu z wykorzystaniem framework'a FastAPI. Na początku został utworzony projekt wraz z podstawową strukturą katalogów , następnie powstał plik readme, który zawierał instrukcję uruchomienia projektu. Kolejnym wykonanym krokiem było utworzenie modeli karty fiszek i talii. Ostatnim zadaniem wykonanym w przyroście było opakowanie projektu w kontener dockerowy.

Wykonane zadania

Zadanie	Wykonawca
Utworzenie kontenera backendu	Jakub
Aktualizacja Readme	Oliwier
[BACKEND] Dodanie 'connector' dla łączności z bazą danych	Jakub
[BACKEND] Utworzenie modelu autoryzacji i użytkownika	Jakub
[BACKEND] Utworzenie cli i funkcji do tworzenia modeli w bazie	Jakub
[BACKEND] Dodanie app_middleware'y	Jakub
[BACKEND] Utworzenie modelu tali oraz karty	Oliwier
[BACKEND] Naprawienie ścieżki importów projektu	Oliwier

Tabela 8.1: Zadania wykonane w sprintie 1

8.2.2 Sprint 2

Opis Sprintu

Drugi przyrost był skupiony na utworzeniu widoku logowania i rejestracji dla aplikacji mobilnej, a także na implementacji tokenu autoryzacji i przypisaniu ról użytkownikom systemu.

Wykonane zadania

Zadanie	Wykonawca
[BACKEND] Poprawienie middleware dla jwt	Oliwier
[MOBILE] utworzyć style scss i zainportować	Daniel
[MOBILE] Wstępny widok logowania i rejestracji bez funkcjonalności (mobilka)	Daniel
[MOBILE] Dodać serwis logowania i rejestracji	Jakub
[BACKEND] Utworzyć wstępne fixture	Jakub
[BACKEND] Utworzyć endpoint do resetowania hasła	Jakub
[MOBILE] Utworzyć wstępne pliki i konfiguracje dla aplikacji mobilnej	Jakub
[BACKEND] Utworzenie loggera	Jakub
[BACKEND] Poprawienie ścieżki dla api	Jakub
[BACKEND] Dodanie nowej dependencji dla roli	Jakub
[BACKEND] Dodanie tokenu na czarną listę	Jakub

Tabela 8.2: Zadania wykonane w sprintie 2

8.2.3 Sprint 3

Opis Sprintu

Trzeci przyrost był skupiony na zadaniach związanych z uporządkowaniem kodu aplikacji mobilnej. Ważnym krokiem dotyczącym strony webowej było utworzenie strony domowej. Backend został rozbudowany o nowe endpointy związane z talią, został utworzony role checker do sprawdzania roli użytkownika aplikacji.

Wykonane zadania

Zadanie	Wykonawca
[MOBILE] Dodać możliwość rejestracji	Jakub
[MOBILE] Dodać panel użytkownika	Jakub
[MOBILE] Zrobić porządek w kodzie	Jakub
[MOBILE] Dodać walidacje hasła	Jakub
[MOBILE] Zaktualizować serwisy z nową metodą request	Jakub
[WEB] Okno logowania	Wiktor
[BACKEND] Poprawić endpoint decs	Oliwier
[WEB] Okno rejestracji	Wiktor
[MOBILE] Dodać interface dla zwracanych danych w metodzie request	Jakub
[BACKEND] napisanie endpointów dla fiszek	Oliwier
[WEB] Utworzenie kontenera docker dla NodeJS	Oliwier
[BACKEND] Poprawić dependencje dla RoleCheckera	Jakub
[MOBILE] Utworzenie metody request	Jakub
[MOBILE] Utworzyć stronę domową po zalogowaniu	Daniel
[WEB] Utworzyć stronę domową po zalogowaniu	Oliwier
[MOBILE] Przerzucić regexy do katalogu validator	Daniel
[MOBILE] Poprawić widok logowania i rejestracji	Daniel
[WEB] Utworzenie strony do tworzenia decku	Oliwier

Tabela 8.3: Zadania wykonane w sprincie 3

8.2.4 Sprint 4

Opis Sprintu

W czwartym przyroście aplikacja webowa została w dużym stopniu rozbudowana o nowe widoki, a także została połączona z warstwą backend, umożliwiło to komunikację stron interne-towej z bazą danych w celu pobierania i tworzenia danych potrzebnych do rejestracji i logowania użytkownika. Rozbudowa aplikacji mobilnej była skupiona na profilu użytkownika, zostały dodane funkcjonalności związane ze zmianą i aktualizacją danych.

Wykonane zadania

Zadanie	Wykonawca
[MOBILE] Utworzyć loader	Jakub
[MOBILE] Dodać modal aby potwierdzić hasłem	Jakub
[BACKEND] Poprawa modeli talii i fiszki	Oliwier
[MOBILE] Dodać walidacje hasła	Jakub
[MOBILE] Widok dla zmiany email	Jakub
[MOBILE] Widok dla zmiany hasła	Jakub
[MOBILE] Widok dla zmiany nazwy użytkownika	Jakub
[BACKEND] Dodać endpointy na aktualizowanie danych użytkownika	Jakub
[MOBILE] Widok tworzenia decku	Daniel
[MOBILE] Bottom tab navigator	Daniel
[MOBILE] Widok my decks	Daniel
[WEB] Połączenie strony tworzenia fiszek z backendem	Oliwier
[WEB] Połączenie strony domowej z backendem	Oliwier
[WEB] Utworzenie strony my decks	Oliwier
[WEB] Wylogowanie użytkownika	Oliwier
[WEB] Utworzenie customowego okna dla alertów	Oliwier
[WEB] Utworzenie widoku strony do nauki z talii fiszek	Oliwier

Tabela 8.4: Zadania wykonane w sprincie 4

8.2.5 Sprint 5

Opis Sprintu

Do aplikacji został dodany endpoint, umożliwiający komunikację z czatem GPT. Pozwoliło to na dodanie do strony webowej funkcjonalności związanej z generowaniem treści. Zostały także zaimplementowany tryb uczenia się, który pozwala na dzielenie fiszek na zapamiętane i nie zapamiętane. Do aplikacji mobilnej zostało dodane usuwanie konta użytkownika oraz poprawiona została struktura kodu.

Wykonane zadania

Zadanie	Wykonawca
[MOBILE] Dodać usuwanie konta	Jakub
[BACKEND] Dodał podstawowe fixture dla decków	Jakub
[WEB] Zmiana hasła użytkownika	Wiktor
[WEB] Utworzenie strony do sterowania głosem talia fiszek	Oliwier
[BACKEND] Dodanie endpointu do obsługi czatu GPT	Oliwier
[WEB] Dodać generowanie treści przy użyciu chatu GPT	Oliwier
[BACKEND] Dodanie do usera kolumny dla avatara, poprawa dodanie w flashcard kolumny is memorized	Oliwier
[WEB] Dodanie widoku dla not memorized flashcards	Oliwier
[BACKEND] Dodanie endpointów do flashcards, które filtrują zapamiętane i nie zapamiętane karty	Oliwier
[WEB] Dodanie trybu uczenia	Oliwier
[WEB] Dodać opcję edycji fiszki i możliwość udostępnienia decku	Oliwier
[MOBILE] Poprawienie nazewnictwa w kodzie nawigacji	Daniel
[MOBILE] Naprawa struktury ekranów	Daniel

Tabela 8.5: Zadania wykonane w sprincie 5

8.2.6 Sprint 6

Opis Sprintu

Do aplikacji mobilnej został dodany ekran tworzenia talii fiszek, następne dodane widoki były związane z trybem uczenia się. Aplikacja webowa została rozbudowana o ranking talii i użytkowników. Na stronie webowej zostały dodane funkcjonalności związane z aktualizacją danych użytkownika. Utworzony został model nlp do rozumienia semantyki słów wykorzystany do sterowania talią fiszek przy użyciu komend głosowych. Dla backendu zostały napisany testy integracyjne uruchamiany przy użyciu biblioteki pytest.

Wykonane zadania

Zadanie	Wykonawca
[MOBILE] Dodać możliwość update'u avatara	Jakub
[MOBILE] Naprawić błąd z query w widoku decklist	Jakub
[WEB] Profil użytkownika	Wiktor
[BACKEND] Dodać celery do aplikacji	Jakub
[WEB] Usunięcie konta użytkownika	Wiktor
[WEB] Zmiana email użytkownika	Wiktor
[WEB] Zmiana nicku użytkownika	Wiktor
[BACKEND] Utworzyć metodę do wysyłania emaila	Jakub
[BACKEND] Utworzyć templatki dla maila	Jakub
[MOBILE] Detale usera z rankingu	Jakub
[MOBILE] Spiąć "My Decks" z API	Daniel
[MOBILE] Spiąć "Create Decks" z API	Daniel
[MOBILE] Dodać ekran podglądu fiszek	Daniel
[MOBILE] Dodać ekran podglądu decku	Daniel
[WEB] Ranking użytkowników	Oliwier
[MOBILE] Dodać listę udostępnionych talii	Jakub
[MOBILE] Spiąć podgląd decku z API	Daniel
[MOBILE] Dodać ekran tworzenia fiszki	Daniel
[MOBILE] Spiąć ekran tworzenia fiszki z API	Daniel
[MOBILE] Dodać service flashcards	Daniel
[MOBILE] Utworzyć component pobierania danych z API	Daniel
[MOBILE] Dodać edycję i usunięcie fiszki	Daniel
[MOBILE] Dodać ekran settings dla decku	Daniel
[WEB] Poprawa rankingów	Oliwier
[MOBILE] Dodać i spiąć memorized flashcards	Daniel
[BACKEND] Testy integracyjne	Oliwier
[WEB] Utworzenie strony public decks	Oliwier
[WEB] Dodanie obsługi złożonych komend głosowych	Oliwier

[BACKEND] Utworzenie modelu NLP do rozumienia semantyki słów	Oliwier
--	---------

Tabela 8.6: Zadania wykonane w sprincie 6

8.3 Szczegóły implementacji

Podrozdział opisuje jak zostały zaimplementowane poszczególne funkcjonalności systemu.

Rozdział 9

Testy

9.1 Testy integracyjne

Testy integracyjne mają na celu sprawdzenie, czy interfejsy pomiędzy komponentami współpracującymi ze sobą poprawnie¹. Do przeprowadzenia testów została użyta biblioteka "pytest", która pozwala na szybkie i proste uruchomienie testów. Do wykonywania zapytań HTTP została wykorzystana biblioteka "requests". Testy miały na celu sprawdzenie, czy status zwrocony przez wykonane żądanie pokrywa się z oczekiwany rezultatem.

Przetestowane zostały poniższe funkcjonalności systemu:

- Utworzenie fiszki
- Zwrócenie danych fiszki na podstawie id
- Aktualizacja danych fiszki
- Usunięcie fiszki
- Otrzymanie odpowiedzi od chatu GPT
- Tworzenie talii
- Zwrócenie danych talii na podstawie id
- Zwrócenie danych wszystkich talii należących do użytkownika

¹4, i co tera.

- Aktualizacja danych talii
- Usunięcie talii
- Zwrócenie rankingu talii
- Utworzenie konta użytkownika
- Logowanie do systemu
- Zwrócenie danych użytkownika

```
def test_delete_deck():
    body = {
        "user_id": f"{user['id']}",
        "title": "string",
        "deck_category": "string"
    }
    headers = {
        "Authorization": f"Bearer {token}",
    }

    deck = requests.post(ENDPOINT + "decks/create_deck", json=body , headers=headers)
    deck_id = deck.json()['id']
    response = requests.delete(ENDPOINT + f"decks/delete_deck/{deck_id}" , json=body , headers=headers)
    assert response.status_code == 200
```

Rysunek 9.1: Test sprawdzający działanie endpointu do usuwania talii użytkownika.

9.2 Testy akceptacyjne

Testy akceptacyjne pozwalają ocenić gotowość systemu do wdrożenia. Głównym ich założeniem jest sprawdzenie, czy system nadaje się do użytkowania. Zazwyczaj do testów akceptacyjnych wykorzystywana jest grupa użytkowników. Do przeprowadzenia testów zostały sporządzone scenariusze testowe.

TBC

¹czek 5, it out.

Bibliografia

- [1] Hermann Ebbinghaus. *Memory: A Contribution to Experimental Psychology*. Dostęp: 11 maja 2024, godz. 7:00. Classics in the History of Psychology. 1885. URL: <https://psychclassics.yorku.ca/Ebbinghaus/memory5.htm> (term. wiz. 11.05.2024).
- [2] Wikipedia. *Grywalizacja*. Dostęp: 11 maja 2024, godz. 8:40. 2024. URL: <https://pl.wikipedia.org/wiki/Grywalizacja> (term. wiz. 11.05.2024).
- [3] Ably. *Gamification: How to Increase Active Users (MAU and DAU)*. Dostęp: 11 maja 2024, godz. 8:30. 2024. URL: <https://ably.com/blog/how-to-increase-active-users> (term. wiz. 11.05.2024).
- [4] Tomasz Miernik. *Testy integracyjne*. Dostęp: 31 maja 2024, godz. 14:00. 2024. URL: <https://testerzy.pl/baza-wiedzy/artykuly/testy-integracyjne> (term. wiz. 31.05.2024).
- [5] Redakcja Testerzy.pl. *Testowanie akceptacyjne*. Dostęp: 31 maja 2024, godz. 15:00. 2024. URL: <https://testerzy.pl/baza-wiedzy/testowanie-akceptacyjne> (term. wiz. 31.05.2024).