

Kierunek: IST

**ZESPOŁOWE PRZEDSIĘWZIĘCIE
INŻYNIERSKIE**



Aplikacja mobilna wspomagająca rozwój umiejętności pisarskich wykorzystująca elementy sztucznej inteligencji do analizy i tworzenia tekstów

Mobile application for the development of writing skills using artificial intelligence for text creation and analysis

Agnieszka Kłobus
Aleksandra Koperwas
Aleksandra Stecka
Bogusława Tłołka

Opiekun pracy
dr inż. Michał Kędziora

Spis treści

DOKUMENTACJA PROJEKTOWA	2
1. Cel i zakres przedsięwzięcia	2
2. Słownik pojęć	2
3. Stan wiedzy w obszarze przedsięwzięcia	4
4. Założenia wstępne	6
4.1. Wysokopoziomowa lista wymagań	6
4.2. Wysokopoziomowa lista użytkowników	7
4.3. Dobór technologii	8
4.4. Przyjęte ograniczenia	9
5. Specyfikacja i analiza wymagań na produkt programowy	9
5.1. Definicja wymagań funkcjonalnych	9
5.2. Definicja wymagań niefunkcjonalnych	12
5.3. Tekstowa specyfikacja przypadków użycia	13
5.4. Diagram przypadków użycia	29
6. Projekt produktu programowego	30
6.1. Projekt architektury	30
6.2. Projekt bazy danych	32
6.3. Skrypt SQL DDL	34
6.4. Makiety interfejsu aplikacji	37
6.5. Reguły przewodnika stylu	47
6.6. Mockupy interfejsu	51
6.7. Badania UX na grupie docelowej	64
6.8. Wzorce projektowe	70
7. Implementacja	73
7.1. Aplikacja Android	73
7.2. API	88
7.3. Pozyskanie danych uczących	90
7.4. Moduł analizy tekstu	92
7.5. Moduł generowania tekstu	99
8. Przypadki testowe	103
9. Wyniki i analiza badań	115
10. Podsumowanie	119
DOKUMENTACJA UŻYTKOWNIKA	120
1. Wprowadzenie	120
2. Instalacja produktu programowego	120
2.1. Wymagania systemowe	120
2.2. Opis procesu instalacji	120
2.3. Opis realizacji typowych zadań z podziałem na aktorów	124
ANEKS	127
1. Kod projektu	127
2. Źródła	127
WYKRESY	130

DOKUMENTACJA PROJEKTOWA

1. Cel i zakres przedsięwzięcia

Celem pracy jest przygotowanie aplikacji mobilnej umożliwiającej użytkownikowi rozwój literacki. Będzie to przystępne narzędzie zachęcające do regularnego rozwijania hobby pisarskiego poprzez sugerowanie wyzwań oraz możliwość rywalizacji z innymi. Aplikacja powinna być atrakcyjna dla użytkowników, do czego przyczyni się wykorzystanie elementów sztucznej inteligencji.

Funkcjonalności aplikacji będą obejmować między innymi możliwość realizacji przez użytkownika zadań, polegających na pisaniu tekstów zawierających wylosowane słowa. Przesłane teksty będą analizowane z wykorzystaniem inteligentnych algorytmów pod kątem poprawności. W ramach rywalizacji, użytkownicy będą mogli porównywać ze sobą teksty pisane w ramach tych samych wyzwań. Dodatkowo, możliwa będzie rywalizacja z tekstami stworzonymi dla wyzwań z wykorzystaniem sztucznej inteligencji. Aplikacja będzie również zbierać statystyki użytkownika, obrazując jego stopień zaangażowania. Końcowe teksty będą dostępne do czytania dla wszystkich użytkowników w bibliotece aplikacji.

2. Słownik pojęć

<i>Id.</i>	<i>Termin (Tłumaczenie)</i>	<i>Znaczenie</i>
S1	Użytkownik (User)	Osoba zainteresowana rozwojem swoich umiejętności pisarskich, która utworzyła już indywidualne konto w aplikacji i jest na nie zalogowana.
S2	Gość (Guest)	Osoba zainteresowana rozwojem swoich umiejętności pisarskich, która nie utworzyła jeszcze indywidualnego konta w aplikacji lub nie jest na nie zalogowana.
S3	Opowiadanie (Story)	Tekst opublikowany przez użytkownika, zawierający wylosowane słowo lub słowa i należący do wylosowanego gatunku.
S4	Wyzwanie (Challenge)	Wylosowany gatunek literacki i wybrana przez użytkownika liczba słów.
S5	Gatunek (Genre)	Forma tekstu literackiego o określonej budowie i stylistyce.
S6	Zachęta (Prompt)	Lista wylosowanych słów w liczbie wybranej przez użytkownika.
S7	Słowo (Word)	Losowane słowa w języku angielskim, które użytkownik musi użyć w napisanej historii.

S8	Ulubione (Favourites)	Zbiór polubionych przez danego użytkownika opowiadań napisanych przez innych użytkowników.
S9	Ocena (Score)	Wartość procentowa obliczana przez algorytm określającą w jakim stopniu użytkownik poprawnie napisał historię.
S10	Społeczność (Community)	Grupa innych użytkowników mająca możliwość publikacji, komentowania i oceniania utworów.
S11	Login (Login)	Login identyfikujący użytkownika i wykorzystywany do logowania. Każdy użytkownik ma unikalny login i nie może go zmienić.
S12	Nazwa użytkownika (Username)	Nazwa użytkownika wyświetlana w aplikacji. Użytkownik może zmienić ją w dowolnym momencie. Nie jest ona wykorzystywana do logowania do aplikacji.
S13	Statystyki (User statistics)	Statystyki użytkowania aplikacji przez użytkownika – liczba opublikowanych opowiadań, liczba opublikowanych komentarzy, liczba komentarzy pod opowiadaniami użytkownika, liczba opowiadań dodanych do ulubionych, liczba razy, ile opowiadania użytkownika zostały dodane do ulubionych przez innych użytkowników, liczba razy, ile opowiadania użytkownika zostały ocenione przez innych użytkowników, średnia długość publikowanego opowiadania, liczba słów we wszystkich opowiadaniach w sumie, liczba opowiadań opublikowana dla każdego gatunku, najdłuższy “streak” – liczba dni z rzędu, w których publikowane były opowiadania.
S14	Minigra (Minigame)	Minigra dostępna w aplikacji, w której użytkownik próbuje zgadnąć, które z pary opowiadań napisanych pod to samo wyzwanie (jedno napisane przez człowieka, a drugie wygenerowane przy użyciu elementów sztucznej inteligencji) otrzymało lepszy wynik analizy przy użyciu elementów sztucznej inteligencji.

3. Stan wiedzy w obszarze przedsięwzięcia

Na rynku nie znajduje się aplikacja udostępniająca funkcjonalności ze wszystkich modułów opracowywanej aplikacji – moduł publikowania, czytania, oceny i komentowania tekstów, moduł generowania tekstu i moduł analizy tekstu. Istnieje jednak wiele aplikacji spełniających jedną z funkcjonalności opracowywanej aplikacji. W celu utworzenia konkurencyjnej aplikacji zastosowane zostanie połączenie trzech poniższych rozwiązań. Istniejące na rynku aplikacje zostały przedstawione jako inspirację przy planowaniu funkcjonalności i wyglądu aplikacji. Nie zostaną wykorzystane przy implementacji.

Sprawdzanie gramatyki:

- AI Grammar Checker for English [\[1\]](#)
- Grammarly [\[2\]](#)

Aplikacje sprawdzają wprowadzany tekst i umożliwiają zamienie błędnych sformułowań na poprawne możliwości podane przez aplikację. Tworzona aplikacja wykorzystuje sprawdzanie poprawności gramatycznej tekstu do uwzględniania tego wyniku przy obliczaniu końcowej oceny tekstu.

Publikacja, czytanie i ocena tekstów:

- Wattpad [\[3\]](#)
- archiveofourown.org [\[4\]](#)

Aplikacje do publikacji autorskich tekstów. Użytkownik ma także możliwość czytania już opublikowanych przez innych użytkowników tekstów, komentowania, oceniania lub dodawania do listy swoich ulubionych. Te funkcjonalności zostaną uwzględnione w aplikacji przy jednoczesnym poprawieniu przejrzystości interfejsu.

Generowanie tekstu:

- plot-generator.org.uk [\[5\]](#)

Aplikacja umożliwia generowanie tekstu po zaznaczeniu podanych opcji. W opracowywanej aplikacji tekst będzie generowany na podstawie gatunku literackiego oraz słów wylosowanych do zachęty. W wygenerowanym tekście powinny znajdować się wszystkie słowa z zachęty.

Zapoznanie z dostępnymi modelami generowania tekstu:

- Generowanie historii
 - **Progressive Generation of Long Text** [\[6\]](#)
Model stworzony do generowania tekstów większej długości (przekraczającej 2000 słów). Przedstawiony w pracy Progressive Generation of Long Text with Pretrained Language Models z 2020. Do generacji tekstu wykorzystuje modele BART oraz GPT-2.
 - **CBART (Parallel Refinements for Lexically Constrained Text Generation with BART)** [\[7\]](#)
Model przedstawiony na konferencji EMNLP (Empirical Methods in Natural Language Processing), która jest wiodącą konferencją naukową z obszaru przetwarzania języka naturalnego. Constrained Text Generation ma na celu wykorzystanie w generowanym fragmencie tekstu zdefiniowanych wcześniej kluczy. Wpisuje się więc

perfekcyjnie w zakres zadania generowania historii, które jest przedmiotem działania aplikacji. Do trenowania modelu wykorzystane zostały zdania pochodzące głównie z artykułów rzadowych oraz informacyjnych.

- **MVP (Multi-task Supervised Pre-training for Natural Language Generation) [8]**

Obszerny model umożliwiający generację tekstu w 11 różnych zadaniach między innymi generacja historii, odpowiadanie na pytania, tworzenie streszczeń. Na podstawie prompta (tutaj tytuł tekstu, ale można zmodyfikować model, żeby musiał wykorzystać podane słowa) generuje historię.

- **JointGT (Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs) [9]**

Model prezentowany w artykule naukowym ACL (Association for Computational Linguistics) w 2021 roku. Opera się na reprezentacji grafowej bazy wiedzy. Wykorzystuje pre-trenowane modele generacji tekstu, wzbogacając je o wykorzystanie grafowej bazy wiedzy.

- **PETGEN (Personalized Text Generation Attack on Deep Sequence Embedding-based Classification Models) [10]**

Model generujący teksty naśladujące styl testów uczących, prezentowany na konferencji ACM SIGKDD (Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining). Jest wykorzystywany do ataków "podsywania się".

- **GP-VAE (Variational Encoder-Decoder Models with Gaussian Process Priors) [11]**

Model przedstawiony w artykule Diverse Text Generation via Variational Encoder-Decoder Models with Gaussian Process Priors ACL (Association for Computing Machinery) w 2022 roku. Jest to metoda stochastyczna, wykorzystująca funkcję gaussowską do zwiększenia różnorodności w generowanym tekście.

- Generowanie zdań słowo po słowie

- **GPT-2 [12]**

Model przedstawiony w artykule Language Models are Unsupervised Multitask Learners. To model utworzony w 2019 roku, który stanowi podstawę dla większości metod wykorzystujących pre-trenowane modele cząstkowe. W 2020 ulepszony do standardu GPT-3, który jest dużo rzadziej wykorzystywany przez jego znaczną wielkość, niedostępną dla użytkowników bez specjalistycznego sprzętu.

- **GPT-Neo [13]**

Model opublikowany w Proceedings of BigScience Episode #5. Jest modelem autoregresyjny będący replikacją architektury GPT-3 i opracowany przez grupę badawczą EleutherAI. Od modelu GPT-3 różni się zmianami tokenizera, zastosowaniem RoPE (Rotary Positional Embeddings), inną metodą inicjalizacji, innymi hyperparametrami oraz równoległym obliczaniem warstw uwagi (attention layer) i sprzężenia w przód (feed-forward layer).

- **Markov Chain, LSTM neural network [14]**

Model przewiduje kolejne słowo mające wystąpić w tekście na podstawie poprzedniego zdania lub poprzednio wykorzystanych słów. Wykorzystywane w przykładzie do generowania bajek. Markov Chain jest modelem stochastycznym, bazującym na prawdopodobieństwie wystąpienia słowa w tekście.

Zdecydowano wykorzystać model CBART [7] – model generuje jedno zdanie zawierające podane słowa kluczowe. Jest to zadanie analogiczne do zachęty generowanej dla użytkowników przez proponowaną aplikację. Model CBART nie umożliwia generowania tekstu w określonym gatunku literackim, ale został wybrany ponieważ jest specjalizowany do zadania występującego w projekcie. Do wygenerowania opowiadania na podstawie jednego zdania wyprodukowanego przez model CBART zastosowany zostanie model GPT-Neo [13].

4. Założenia wstępne

4.1. Wysokopoziomowa lista wymagań

Wysokopoziomowa lista wymagań w formie listy potrzeb użytkowników:

<i>Id.</i>	<i>Opis</i>
P1	Użytkownik może otrzymać wyzwania motywujące go do pisania krótkich opowiadań.
P2	Użytkownik może publikować krótkie opowiadania pisane w ramach wyzwań.
P3	Użytkownik może czytać napisane przez siebie wcześniej opowiadania.
P4	Użytkownik może usunąć napisane przez siebie wcześniej opowiadania.
P5	Użytkownik może czytać opowiadania napisane przez innych użytkowników aplikacji.
P6	Użytkownik może widzieć ocenę, którą jego opowiadania otrzymały od innych użytkowników.

P7	Użytkownik może wystawiać oceny opowiadaniom innych użytkowników.
P8	Użytkownik może czytać komentarze, które inni użytkownicy dodali do jego opowiadania.
P9	Użytkownik może komentować opowiadania innych użytkowników.
P10	Użytkownik może usunąć komentarze dodane do opowiadań innych użytkowników.
P11	Użytkownik może dodać opowiadanie innego użytkownika do swoich ulubionych.
P12	Użytkownik może przeglądać statystyki dotyczące swojego użycia aplikacji.
P13	Użytkownik może edytować dane dotyczące swojego konta – zdjęcie profilowe, nazwę użytkownika, hasło, adres e-mail.
P14	Użytkownik może zobaczyć ocenę swojego opowiadania przygotowaną z wykorzystaniem elementów sztucznej inteligencji.
P15	Użytkownik może przeczytać opowiadania wygenerowane z wykorzystaniem elementów sztucznej inteligencji.
P16	Użytkownik może wziąć udział w grze, w której użytkownik próbuje zgadnąć, które z pary opowiadań napisanych pod to samo wyzwanie (jedno było napisane przez człowieka, a drugie wygenerowane przez model generowania tekstu) otrzymało lepszy wynik.

4.2. Wysokopoziomowa lista użytkowników

Nazwa	Opis	Zakres odpowiedzialności
Użytkownik	Osoba zainteresowana rozwojem swoich umiejętności pisarskich, która utworzyła już indywidualne konto w aplikacji i jest na nie zalogowana.	Pisanie krótkich opowiadań w ramach wyzwań Usunięcie wcześniej napisanych opowiadań Czytanie opowiadań napisanych przez siebie i innych użytkowników Ocena opowiadań innych użytkowników i dostęp do oceny swoich opowiadań przez innych użytkowników

		<p>Komentowanie opowiadań innych użytkowników i dostęp do komentarzy swoich opowiadań od innych użytkowników</p> <p>Dodanie opowiadania innego użytkownika do ulubionych</p> <p>Przeglądanie swoich statystyk</p> <p>Edycja swoich danych (zdjęcie profilowe, nazwa użytkownika, hasło do konta, adres e-mail)</p> <p>Dostęp do oceny swojego opowiadania przygotowanej z wykorzystaniem elementów sztucznej inteligencji</p> <p>Dostęp do przykładowych opowiadań wygenerowanych z wykorzystaniem elementów sztucznej inteligencji</p>
Gość	Osoba zainteresowana rozwojem swoich umiejętności pisarskich, która nie utworzyła jeszcze indywidualnego konta w aplikacji lub nie jest na nie zalogowana.	<p>Zakładanie indywidualnego konta użytkownika</p> <p>Logowanie na indywidualne konto użytkownika</p>

4.3. Dobór technologii

Podstawą produktu proponowanego w ramach przedsięwzięcia jest aplikacja mobilna. Zdecydowano o napisaniu aplikacji natywnej na system operacyjny Android w języku Kotlin. Perspektywą rozwoju projektu jest udostępnienie aplikacji również na inne systemy operacyjne, np. iOS.

Aby umożliwić udostępnianie opowiadań między użytkownikami konieczne jest utworzenie serwera bazy danych w chmurze. W tym celu wykorzystano usługi chmurowe Heroku [15], które umożliwiają darmowe hostowanie bazy danych PostgreSQL oraz aplikacji. Usługi chmurowe Heroku [15] udostępniają system zarządzania relacyjnymi bazami danych PostgreSQL.

W celu zapewnienia prostej komunikacji z bazą danych z aplikacji mobilnej zdecydowano o napisaniu API, które również można wdrożyć w chmurze Heroku [15]. Ze względu na ograniczenia serwerów Heroku [15] zdecydowano o napisaniu API w mikro-frameworku Flask w języku Python.

Moduł wykorzystujący elementy sztucznej inteligencji, który zapewnia aplikacji konkurencyjność na rynku i dodaje przedsięwzięciu aspekt badawczy, zdecydowano napisać w języku Python oraz wdrożyć w chmurze Heroku [15] jako mikroserwis.

Architektura aplikacji mobilnej oparta jest o wzorzec architektoniczny Model View ViewModel (MVVM). We wzorcu MVVM model odpowiada za połączenie aplikacji z bazą danych i dostęp do danych, ViewModel jest odpowiedzialny za odczytanie danych z modelu i przekazanie ich do widoku (view), a widok (view) jest warstwą udostępniającą użytkownikowi interfejs graficzny.

4.4. Przyjęte ograniczenia

Przyjęto ograniczenie dotyczące loginów użytkowników, które są używane w celu logowania do aplikacji – loginu wykorzystywanego do logowania do aplikacji nie ma możliwości zmienić. Zamiast tego wprowadzono nazwę użytkownika, która nie jest używana do logowania i jest wyświetlana w aplikacji. Użytkownik może zmienić swoją nazwę użytkownika w dowolnym momencie. Podobne rozwiązanie zastosowano w aplikacji Snapchat.

Przyjęto ograniczenie dotyczące zdjęć profilowych użytkowników. Aby nie przechowywać zdjęć profilowych w bazie danych z czym wiąże się nakładanie ograniczeń na rozmiar obrazów, zdecydowano udostępnić użytkownikom kilka zdjęć profilowych do wyboru. Użytkownik wciąż może wybrać zdjęcie profilowe, ale będzie to jedno ze zdjęć dostępnych w aplikacji. W ten sposób unika się również sytuacji, w której któryś użytkownik ustawi na swoje zdjęcie profilowe coś nieprzyzwoitego. Podobne rozwiązanie zastosowano w aplikacji Netflix.

Naturalnym ograniczeniem wynikającym z charakteru aplikacji jest ograniczenie długości opowiadań do krótkich tekstów (kilka akapitów). Aplikacja ma funkcjonować jako gra pozwalająca na rozwój zainteresowań pisarskich i kreatywności. Opowiadania pisane przez użytkowników są pisane pod różne wyzwania, dzięki czemu użytkownicy trenują swoją kreatywność poprzez pisanie tekstów na tematy, które inaczej być może nie przyszłyby im do głowy. Wprowadzono więc ograniczenie na długość pisanych przez użytkowników tekstów.

5. Specyfikacja i analiza wymagań na produkt programowy

5.1. Definicja wymagań funkcjonalnych

<i>Id.</i>	<i>Nazwa</i>	<i>Opis</i>	<i>Realizowana potrzeba</i>
WF1	Rejestracja	System umożliwia zakładanie indywidualnego konta.	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14
WF2	Logowanie	System umożliwia zalogowanie się na indywidualne konto.	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14

WF3	Wylogowanie	System umożliwia wylogowanie się z indywidualnego konta.	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14
WF4	Odzyskanie danych logowania	System umożliwia wygenerowanie nowego hasła i odzyskanie loginu, jeżeli użytkownik zapomni danych logowania do indywidualnego konta.	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14
WF5	Losowanie wyzwania	System generuje losowe wyzwanie na żądanie użytkownika.	P1
WF6	Pisanie opowiadań	System umożliwia pisanie i publikowanie opowiadań w ramach wylosowanych wyzwań.	P1, P2
WF7	Wyświetlanie własnego opowiadania	System umożliwia wyświetlanie opowiadań napisanych przez użytkownika razem z ocenami i komentarzami, które otrzymały one od innych użytkowników.	P3, P6, P8
WF8	Usunięcie własnego opowiadania	System umożliwia usunięcie opowiadania napisanego wcześniej przez użytkownika.	P4
WF9	Wyświetlanie opowiadań innych użytkowników	System umożliwia wyświetlanie opowiadań napisanych przez innych użytkowników.	P5
WF10	Filtrowanie opowiadań	System umożliwia filtrowanie listy opowiadań napisanych przez innych użytkowników po przypisanych gatunkach.	P5
WF11	Wyszukiwanie opowiadań	System umożliwia wyszukiwanie po słowach kluczowych opowiadań napisanych zarówno przez użytkownika, jak i innych użytkowników.	P5
WF12	Ocena opowiadań	System umożliwia wystawienie oceny opowiadaniom napisanym przez innych użytkowników.	P7

WF13	Komentowanie opowiadań	System umożliwia dodanie komentarza do opowiadań napisanych przez innych użytkowników.	P9
WF14	Usuwanie komentarzy	System umożliwia usunięcie komentarza dodanego pod opowiadaniem napisanym przez innego użytkownika.	P10
WF15	Dodawanie do ulubionych	System umożliwia dodanie opowiadań innych użytkowników do swoich ulubionych.	P11
WF16	Usuwanie z ulubionych	System umożliwia usuwanie opowiadań innych użytkowników ze swoich ulubionych.	P11
WF17	Ulubione	System przechowuje listę ulubionych opowiadań użytkownika i umożliwia jej wyświetlanie	P11
WF18	Statystyki	System oblicza statystyki użycia aplikacji przez użytkownika.	P12
WF19	Edycja konta	System umożliwia użytkownikowi zmianę zdjęcia profilowego, wyświetlnej nazwy użytkownika, hasła i adresu e-mail.	P13
WF20	Odzyskanie danych logowania	System umożliwia użytkownikowi odzyskanie danych logowania do swojego konta wykorzystując adres e-mail	P14
WF21	Analiza tekstu	System wykorzystuje elementy sztucznej inteligencji do analizy opowiadań napisanych przez użytkowników i wyświetla je użytkownikom.	P15
WF22	Generowanie tekstu	System wykorzystuje elementy sztucznej inteligencji do generowania przykładowych opowiadań na podstawie wyzwań i wyświetla wygenerowane opowiadania użytkownikom.	P16

WF23	Zagranie w minigrę	System umożliwia zagranie w minigrę, w której użytkownik próbuje zgadnąć, które z pary opowiadań napisanych pod to samo wyzwanie (jedno było napisane przez człowieka, a drugie wygenerowane przez model generowania tekstu) otrzymało lepszy wynik.	P17
------	--------------------	--	-----

5.2. Definicja wymagań niefunkcjonalnych

<i>Id.</i>	<i>Opis</i>
WNf1	Aplikacje są dostępne w systemie 24/7/365.
WNf2	Aplikacje są dostępne w języku angielskim.
WNf3	Z aplikacji może na raz korzystać 500 użytkowników.
WNf4	Przekleństwa i niestosowne słowa nie mogą być używane bez cenzury.
WNf5	Użytkownik ma możliwość dostosowania wyglądu aplikacji poprzez zastosowanie jasnego i ciemnego stylu.
WNf6	Hasła użytkowników są chronione za pomocą kryptograficznej funkcji skrótu MD5.
WNf7	Hasła użytkowników muszą mieć długość co najmniej 8 znaków, zawierać małe i duże litery, cyfry oraz znaki specjalne.
WNf8	Żadne wrażliwe dane użytkowników (imię i nazwisko, adres zamieszkania, numer karty kredytowej) nie są przechowywane.
WNf9	Aplikacja mobilna działa na urządzeniach z systemem operacyjnym Android.
WNf10	Aplikacja mobilna odpowiada na zapytania użytkownika w czasie do 2 sekund.

5.3. Tekstowa specyfikacja przypadków użycia

<i>Id.</i>	PU1
<i>Nazwa</i>	Rejestracja
<i>Aktorzy</i>	Gość
<i>Warunki wstępne</i>	Gość posiada zainstalowaną aplikację na telefonie
<i>Warunki końcowe</i>	Gość zostaje zarejestrowany w aplikacji oraz przeniesiony na ekran główny
<i>Realizowane wymagania</i>	WF1
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Gość uruchamia aplikację	Wyświetlenie ekranu logowania
Gość wybiera przycisk “I don’t have an account yet”	Wyświetlenie ekranu rejestracji
Gość wybiera sposób rejestracji: a. wpisuje login i hasło i naciska przycisk “Register” b. naciska przycisk “Sign up with Google”	Sprawdzenie poprawności danych: a. dane poprawne - zarejestrowanie w aplikacji oraz wyświetlenie ekranu głównego b. dane niepoprawne - wyświetlenie komunikatu o błędzie

<i>Id.</i>	PU2
<i>Nazwa</i>	Logowanie
<i>Aktorzy</i>	Gość
<i>Warunki wstępne</i>	Gość posiada zainstalowaną aplikację na telefonie
<i>Warunki końcowe</i>	Gość zostaje zalogowany do aplikacji i przeniesiony na ekran główny
<i>Realizowane wymagania</i>	WF2

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Gość uruchamia aplikację	Wyświetlenie ekranu logowania
Gość wybiera sposób logowania: a. wpisuje login i hasło i naciska przycisk "Log in" b. naciska przycisk "Sign in with Google"	Sprawdzenie poprawności danych: a. dane poprawne – wyświetlenie głównego ekranu aplikacji b. dane niepoprawne – wyświetlenie komunikatu "Incorrect login or password"

<i>Id.</i>	PU3
<i>Nazwa</i>	Wylogowanie
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i znajduje się na ekranie profilu
<i>Warunki końcowe</i>	Użytkownik zostaje wylogowany z aplikacji i przeniesiony na ekran logowania
<i>Realizowane wymagania</i>	WF3

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik naciska przycisk "Log out"	Wylogowanie z aplikacji i wyświetlenie ekranu logowania

<i>Id.</i>	PU4
<i>Nazwa</i>	Odzyskanie hasła
<i>Aktorzy</i>	Gość
<i>Warunki wstępne</i>	Gość posiada zainstalowaną aplikację na telefonie i posiada konto w aplikacji
<i>Warunki końcowe</i>	Przesłanie na adres e-mail użytkownika nowego hasła
<i>Realizowane wymagania</i>	WF4
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Gość uruchamia aplikację	Wyświetlenie ekranu logowania
Gość wybiera opcję “I've forgotten my password”	Wyświetlenie okna dialogowego umożliwiającego wpisanie swojego loginu
a. Gość wprowadza swój login i naciska przycisk “Confirm” b. Gość naciska przycisk “Cancel”	a. Sprawdzenie poprawności danych: i. dane poprawne – wysłanie na adres e-mail powiązany z kontem gościa link do generowania nowego hasła ii. dane niepoprawne – wyświetlenie komunikatu o błędzie b. Zamknięcie okna dialogowego
<i>Id.</i>	PU5
<i>Nazwa</i>	Odzyskanie loginu
<i>Aktorzy</i>	Gość
<i>Warunki wstępne</i>	Gość posiada zainstalowaną aplikację na telefonie i posiada konto w aplikacji
<i>Warunki końcowe</i>	Przesłanie na adres e-mail użytkownika jego loginu
<i>Realizowane wymagania</i>	WF4
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Gość uruchamia aplikację	Wyświetlenie ekranu logowania

Gość wybiera opcję “I've forgotten my login”	Wyświetlenie okna dialogowego umożliwiającego wpisanie swojego adresu e-mail
a. Gość wprowadza swój adres e-mail i naciska przycisk “Confirm” b. Gość naciska przycisk “Cancel”	a. Sprawdzenie poprawności danych: i. dane poprawne – wysłanie na adres e-mail powiązany z kontem gościa login użytkownika ii. dane niepoprawne – wyświetlenie komunikatu o błędzie b. Zamknięcie okna dialogowego
<i>Id.</i>	PU6
<i>Nazwa</i>	Pisanie opowiadania
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlili ekran główny
<i>Warunki końcowe</i>	Opublikowanie nowego opowiadania i powrót do ekranu głównego
<i>Realizowane wymagania</i>	WF5, WF6
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik naciska przycisk z ikoną plusa	Wyświetlenie przycisków z ikonami 1, 3, 5
Użytkownik wybiera jeden z przycisków	Wyświetlenie ekranu pisania opowiadania i wylosowanie wyzwania z wybraną liczbą słów do zachęty oraz losowym gatunkiem
Użytkownik wpisuje opowiadanie i naciska przycisk “Publish”	Sprawdzenie poprawności danych: a. dane poprawne – zapis opowiadania w bazie danych i wyświetlenie ekranu głównego b. dane niepoprawne – wyświetlenie komunikatu o błędzie

<i>Id.</i>	PU7
<i>Nazwa</i>	Przeglądanie napisanych opowiadań
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran główny
<i>Warunki końcowe</i>	Wyświetlenie listy napisanych przez użytkownika opowiadań
<i>Realizowane wymagania</i>	WF7

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie głównym	Wyświetlenie listy napisanych przez użytkownika opowiadań
<i>Id.</i>	PU8
<i>Nazwa</i>	Wyświetlanie opowiadania
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran główny
<i>Warunki końcowe</i>	Wyświetlenie wybranego opowiadania
<i>Realizowane wymagania</i>	WF7

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie głównym	Wyświetlenie listy napisanych przez użytkownika opowiadań
Użytkownik wybiera opowiadanie z listy	Wyświetlenie wybranego opowiadania

<i>Id.</i>	PU9
<i>Nazwa</i>	Usunięcie opowiadania
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik wybrał opowiadanie do przeczytania
<i>Warunki końcowe</i>	Opowiadanie zostaje usunięte z bazy danych, a użytkownik zostaje przeniesiony na ekran główny
<i>Realizowane wymagania</i>	WF8

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie ze szczegółami wybranego opowiadania	Wyświetlenie wybranego opowiadania
Użytkownik naciska ikonę kosza na śmieci	Wyświetlenie okna dialogowego z prośbą o potwierdzenie chęci usunięcia opowiadania
a. Użytkownik potwierdza chęć usunięcia opowiadania b. Użytkownik anuluje usunięcie opowiadania	a. Usunięcie opowiadania i powrót do ekranu głównego b. Zamknięcie okna dialogowego i wyświetlenie wybranego opowiadania

<i>Id.</i>	PU10
<i>Nazwa</i>	Przeglądanie opowiadań innych użytkowników
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran społeczności
<i>Warunki końcowe</i>	Wyświetlenie opowiadań innych użytkowników
<i>Realizowane wymagania</i>	WF9

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie społeczności	Wyświetlenie opowiadań innych użytkowników

<i>Id.</i>	PU11
<i>Nazwa</i>	Wyświetlanie opowiadania innego użytkownika
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran społeczności
<i>Warunki końcowe</i>	Wyświetlenie wybranego opowiadania
<i>Realizowane wymagania</i>	WF9

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie społeczności	Wyświetlenie opowiadań innych użytkowników
Użytkownik wybiera opowiadanie z listy	Wyświetlenie wybranego opowiadania
<i>Id.</i>	PU12
<i>Nazwa</i>	Filtrowanie opowiadań
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran społeczności
<i>Warunki końcowe</i>	Wyświetlenie opowiadań z wybranego gatunku lub kilku wybranych gatunków
<i>Realizowane wymagania</i>	WF10

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie społeczności	Wyświetlenie opowiadań innych użytkowników
Użytkownik naciska przycisk z ikoną filtrowania i wybiera gatunek	Wyświetlenie opowiadań z wybranego gatunku lub kilku wybranych gatunków

<i>Id.</i>	PU13
<i>Nazwa</i>	Wyszukiwanie opowiadań
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran społeczności lub ekran główny
<i>Warunki końcowe</i>	Wyświetlenie opowiadań spełniających wyniki wyszukiwania
<i>Realizowane wymagania</i>	WF11
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie społeczności lub ekranie głównym	Wyświetlenie opowiadań innych użytkowników lub swoich opowiadań
Użytkownik naciska ikonę lupy i wpisuje szukaną frazę	Wyświetlenie opowiadań spełniających wyniki wyszukiwania
<i>Id.</i>	PU14
<i>Nazwa</i>	Ocena opowiadania
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik wybrał opowiadanie innego użytkownika do przeczytania
<i>Warunki końcowe</i>	Ocena zostaje zapisana i wyświetlona zostaje odpowiednia liczba gwiazdek oraz aktualizowana jest średnia ocena opowiadania
<i>Realizowane wymagania</i>	WF12
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie ze szczegółami wybranego opowiadania	Wyświetlenie wybranego opowiadania
Użytkownik ocenia opowiadanie innego użytkownika	Ocena zostaje zapisana i wyświetlona zostaje odpowiednia liczba gwiazdek oraz aktualizowana jest średnia ocena opowiadania

<i>Id.</i>	PU15
<i>Nazwa</i>	Dodanie komentarza
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik wybrał opowiadanie innego użytkownika do przeczytania
<i>Warunki końcowe</i>	Komentarz zostaje zapisany w bazie danych i wyświetlony na liście komentarzy pod opowiadaniem
<i>Realizowane wymagania</i>	WF13
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie ze szczegółami wybranego opowiadania	Wyświetlenie wybranego opowiadania
Użytkownik naciska przycisk “Add comment”	Wyświetlenie okna dialogowego umożliwiającego wpisanie komentarza
a. Użytkownik naciska przycisk “Add” b. Użytkownik naciska przycisk “Cancel”	a. Sprawdzenie poprawności danych: i. dane poprawne – komentarz zostaje zapisany i wyświetlony ii. dane niepoprawne – wyświetlenie komunikatu o błędzie b. Zamknięcie okna dialogowego i wyświetlenie wybranego opowiadania

<i>Id.</i>	PU16
<i>Nazwa</i>	Usunięcie komentarza
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik wybrał opowiadanie innego użytkownika do przeczytania i znajduje się pod nim wcześniej dodany przez niego komentarz
<i>Warunki końcowe</i>	Komentarz zostaje usunięty z bazy danych i nie wyświetla się na liście komentarzy pod opowiadaniem
<i>Realizowane wymagania</i>	WF14
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie ze szczegółami wybranego opowiadania	Wyświetlenie wybranego opowiadania
Użytkownik naciska ikonę kosza na śmieci przy swoim komentarzu	Wyświetlenie okna dialogowego z prośbą o potwierdzenie chęci usunięcia komentarza
a. Użytkownik potwierdza chęć usunięcia komentarza b. Użytkownik anuluje usunięcie komentarza	a. Usunięcie komentarza i wyświetlenie wybranego opowiadania b. Zamknięcie okna dialogowego i wyświetlenie wybranego opowiadania

<i>Id.</i>	PU17
<i>Nazwa</i>	Dodanie do ulubionych
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik wybrał opowiadanie innego użytkownika do przeczytania i nie dodał go wcześniej do ulubionych
<i>Warunki końcowe</i>	Dodanie opowiadania do ulubionych użytkownika i wyświetlenie ikony pełnego serduszka
<i>Realizowane wymagania</i>	WF15
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie ze szczegółami wybranego opowiadania	Wyświetlenie wybranego opowiadania z ikoną pustego serduszka
Użytkownik naciska ikonę serduszka	Dodanie opowiadania do ulubionych użytkownika i wyświetlenie ikony pełnego serduszka
<i>Id.</i>	PU18
<i>Nazwa</i>	Usunięcie z ulubionych
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik wybrał opowiadanie innego użytkownika do przeczytania i dodał je wcześniej do ulubionych
<i>Warunki końcowe</i>	Usunięcie opowiadania z listy ulubionych i wyświetlenie ikony pustego serduszka
<i>Realizowane wymagania</i>	WF16
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie ze szczegółami wybranego opowiadania	Wyświetlenie wybranego opowiadania z ikoną pełnego serduszka
Użytkownik naciska ikonę serduszka	Usunięcie opowiadania z ulubionych użytkownika i wyświetlenie ikony pustego serduszka

<i>Id.</i>	PU19
<i>Nazwa</i>	Przeglądanie ulubionych
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran profilu
<i>Warunki końcowe</i>	Wyświetlenie listy ulubionych opowiadań
<i>Realizowane wymagania</i>	WF17
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie profilu	Wyświetlenie dostępnych opcji – ulubione, statystyki, zmiana zdjęcia profilowego, zmiana nazwy, zmiana hasła, zmiana adresu e-mail, wylogowanie
Użytkownik wybiera opcję “Favourite”	Wyświetlenie listy ulubionych opowiadań
<i>Id.</i>	PU20
<i>Nazwa</i>	Wyświetlenie statystyk
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran profilu
<i>Warunki końcowe</i>	Wyświetlenie statystyk korzystania z aplikacji
<i>Realizowane wymagania</i>	WF18
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie profilu	Wyświetlenie dostępnych opcji – ulubione, statystyki, zmiana zdjęcia profilowego, zmiana nazwy, zmiana hasła, zmiana adresu e-mail, wylogowanie
Użytkownik wybiera opcję “Statistics”	Wyświetlenie statystyk korzystania z aplikacji

<i>Id.</i>	PU21
<i>Nazwa</i>	Zmiana zdjęcia profilowego
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran profilu
<i>Warunki końcowe</i>	Zmiana zdjęcia profilowego w bazie danych i wyświetlenie go w profilu
<i>Realizowane wymagania</i>	WF19

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie profilu	Wyświetlenie dostępnych opcji – ulubione, statystyki, zmiana zdjęcia profilowego, zmiana nazwy, zmiana hasła, wylogowanie
Użytkownik wybiera opcję “Change profile picture”	Wyświetlenie ekranu zmiany zdjęcia profilowego
Użytkownik wybiera nowe zdjęcie profilowe	Zapisanie nowego zdjęcia profilowego i wyświetlenie nowego zdjęcia profilowego

<i>Id.</i>	PU22
<i>Nazwa</i>	Zmiana nazwy
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran profilu
<i>Warunki końcowe</i>	Zmiana nazwy w bazie danych i wyświetlenie jej w profilu
<i>Realizowane wymagania</i>	WF19

Scenariusz

<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie profilu	Wyświetlenie dostępnych opcji – ulubione, statystyki, zmiana zdjęcia profilowego, zmiana nazwy, zmiana hasła, zmiana adresu e-mail, wylogowanie
Użytkownik wybiera opcję “Change name”	Wyświetlenie okna dialogowego umożliwiającego wpisanie nowej nazwy

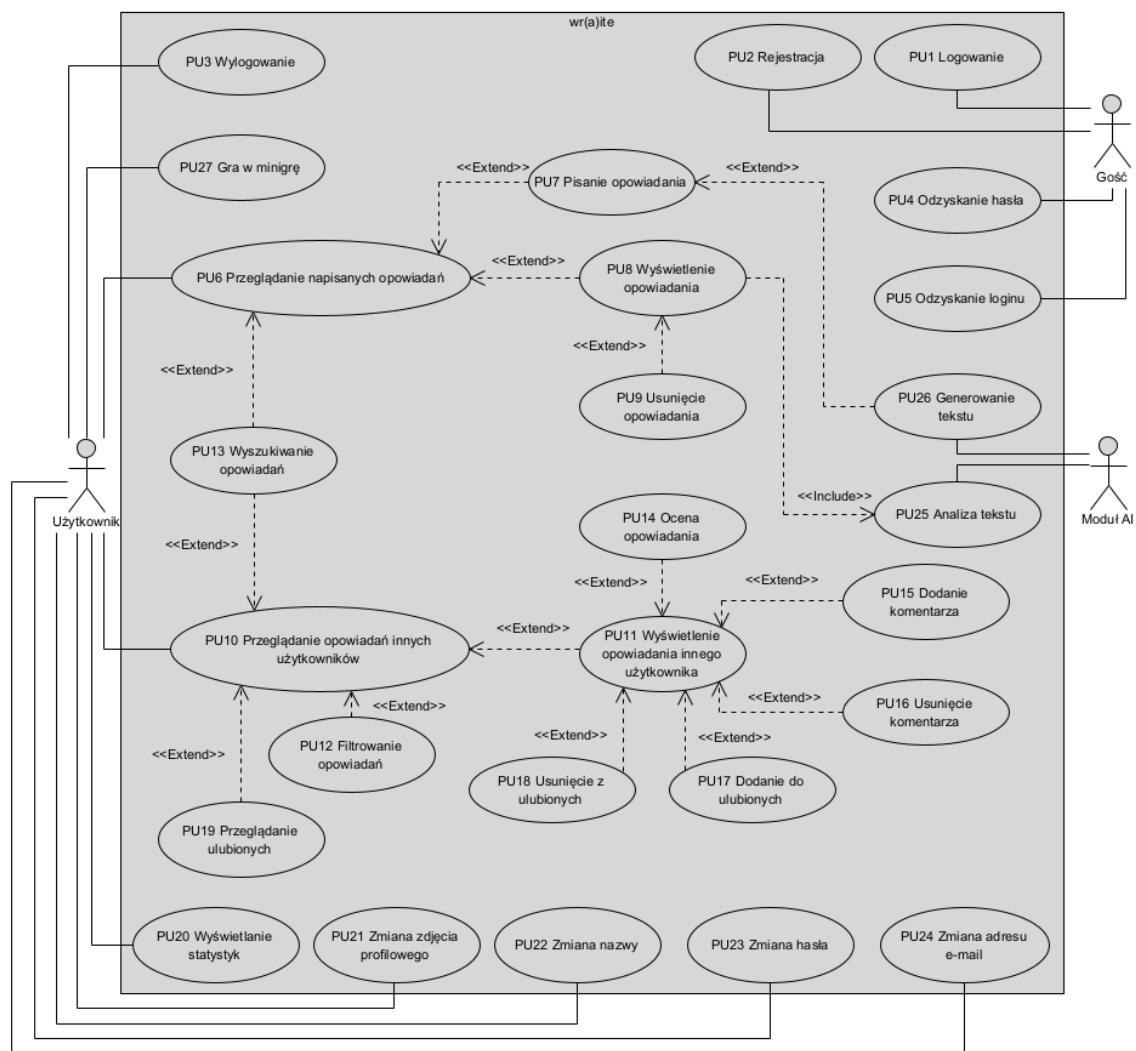
a. Użytkownik naciska przycisk "Save"	a. Sprawdzenie poprawności danych: i. dane poprawne – nowa nazwa użytkownika zostaje zapisana i wyświetlona na ekranie profilu ii. dane niepoprawne – wyświetlenie komunikatu o błędzie
b. Użytkownik naciska przycisk "Cancel"	b. Zamknięcie okna dialogowego
<i>Id.</i>	PU23
<i>Nazwa</i>	Zmiana hasła
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran profilu
<i>Warunki końcowe</i>	Zmiana hasła w bazie danych
<i>Realizowane wymagania</i>	WF19
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie profilu	Wyświetlenie dostępnych opcji – ulubione, statystyki, zmiana zdjęcia profilowego, zmiana nazwy, zmiana hasła, zmiana adresu e-mail, wylogowanie
Użytkownik wybiera opcję "Change password"	Wyświetlenie okna dialogowego umożliwiającego wpisanie nowego hasła
a. Użytkownik naciska przycisk "Save" b. Użytkownik naciska przycisk "Cancel"	a. Sprawdzenie poprawności danych: i. dane poprawne – nowe hasło użytkownika zostaje zapisane ii. dane niepoprawne – wyświetlenie komunikatu o błędzie b. Zamknięcie okna dialogowego

<i>Id.</i>	PU24
<i>Nazwa</i>	Zmiana adresu e-mail
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran profilu
<i>Warunki końcowe</i>	Zmiana adresu email w bazie danych
<i>Realizowane wymagania</i>	WF19
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik jest na ekranie profilu	Wyświetlenie dostępnych opcji – ulubione, statystyki, zmiana zdjęcia profilowego, zmiana nazwy, zmiana hasła, zmiana adresu e-mail, wylogowanie
Użytkownik wybiera opcję “Change e-mail address”	Wyświetlenie okna dialogowego umożliwiającego wpisanie nowego adresu e-mail
a. Użytkownik naciska przycisk “Save” b. Użytkownik naciska przycisk “Cancel”	a. Sprawdzenie poprawności danych: i. dane poprawne – nowy adres e-mail użytkownika zostaje zapisany ii. dane niepoprawne – wyświetlenie komunikatu o błędzie b. Zamknięcie okna dialogowego
<i>Id.</i>	PU25
<i>Nazwa</i>	Analiza tekstu
<i>Aktorzy</i>	Użytkownik, Moduł AI
<i>Warunki wstępne</i>	Użytkownik jest na ekranie publikowania opowiadania i napisał opowiadanie gotowe do publikacji
<i>Warunki końcowe</i>	Zapisanie analizy opowiadania
<i>Realizowane wymagania</i>	WF20
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik lub moduł AI publikuje opowiadanie	Zapisanie opowiadania użytkownika lub wygenerowanego przez moduł AI i żądanie od modułu AI wygenerowania analizy dla opowiadania

Moduł AI analizuje opowiadanie użytkownika	Analiza opowiadania użytkownika zostaje zapisana i wyświetlona pod tekstem w momencie zakończenia analizy
<i>Id.</i>	PU26
<i>Nazwa</i>	Generowanie tekstu
<i>Aktorzy</i>	Użytkownik, Moduł AI
<i>Warunki wstępne</i>	Użytkownik jest na ekranie publikowania opowiadania i napisał opowiadanie gotowe do publikacji
<i>Warunki końcowe</i>	Zapisanie wygenerowanego opowiadania
<i>Realizowane wymagania</i>	WF21
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik publikuje opowiadanie	Zapisanie opowiadania użytkownika i żądanie od modułu AI wygenerowania opowiadania dla opowiadania użytkownika
Moduł AI generuje opowiadanie	Wygenerowane opowiadanie zostaje zapisane i jest dostępne w minigrze w momencie zakończenia generowania
<i>Id.</i>	PU27
<i>Nazwa</i>	Gra w minigrę
<i>Aktorzy</i>	Użytkownik
<i>Warunki wstępne</i>	Użytkownik jest zalogowany i wyświetlił ekran minigry
<i>Warunki końcowe</i>	Wyświetlenie komunikatu o poprawności odpowiedzi
<i>Realizowane wymagania</i>	WF22
<i>Scenariusz</i>	
<i>Akcja aktora</i>	<i>Akcja systemu</i>
Użytkownik wybiera gatunek	<p>Sprawdzenie dostępności danych:</p> <ul style="list-style-type: none"> a. nie ma dostępnego opowiadania w tym gatunku – wyświetlenie komunikatu o braku opowiadania, koniec przypadku użycia b. jest dostępne opowiadanie w tym gatunku – wyświetlenie pary opowiadań (opowiadanie napisane przez użytkownika i opowiadanie wygenerowane przez moduł AI)

Użytkownik wybiera, które opowiadanie z pary jego zdaniem otrzymało wyższy wynik w module analizy AI	<p>Sprawdzenie poprawności danych:</p> <ol style="list-style-type: none"> odpowiedź poprawna – wyświetlenie komunikatu o poprawności odpowiedzi, przyznanie jednego punktu i zapisanie odpowiedzi odpowiedź niepoprawna – wyświetlenie komunikatu o niepoprawności odpowiedzi, przyznanie zera punktów i zapisanie odpowiedzi oba opowiadania z pary otrzymały taki sam wynik w module analizy AI – wyświetlenie komunikatu o remisie, przyznanie pół punkta i zapisanie odpowiedzi
--	--

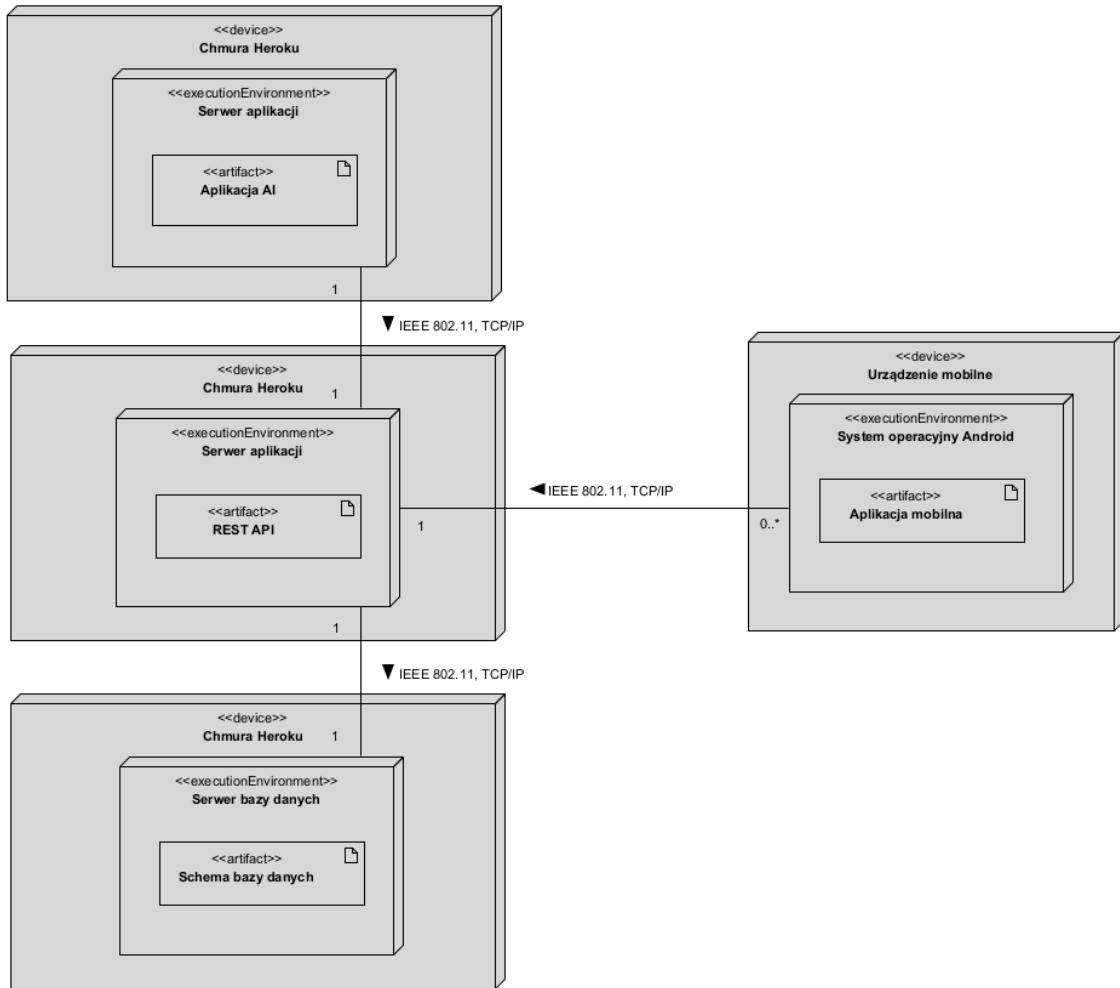
5.4. Diagram przypadków użycia



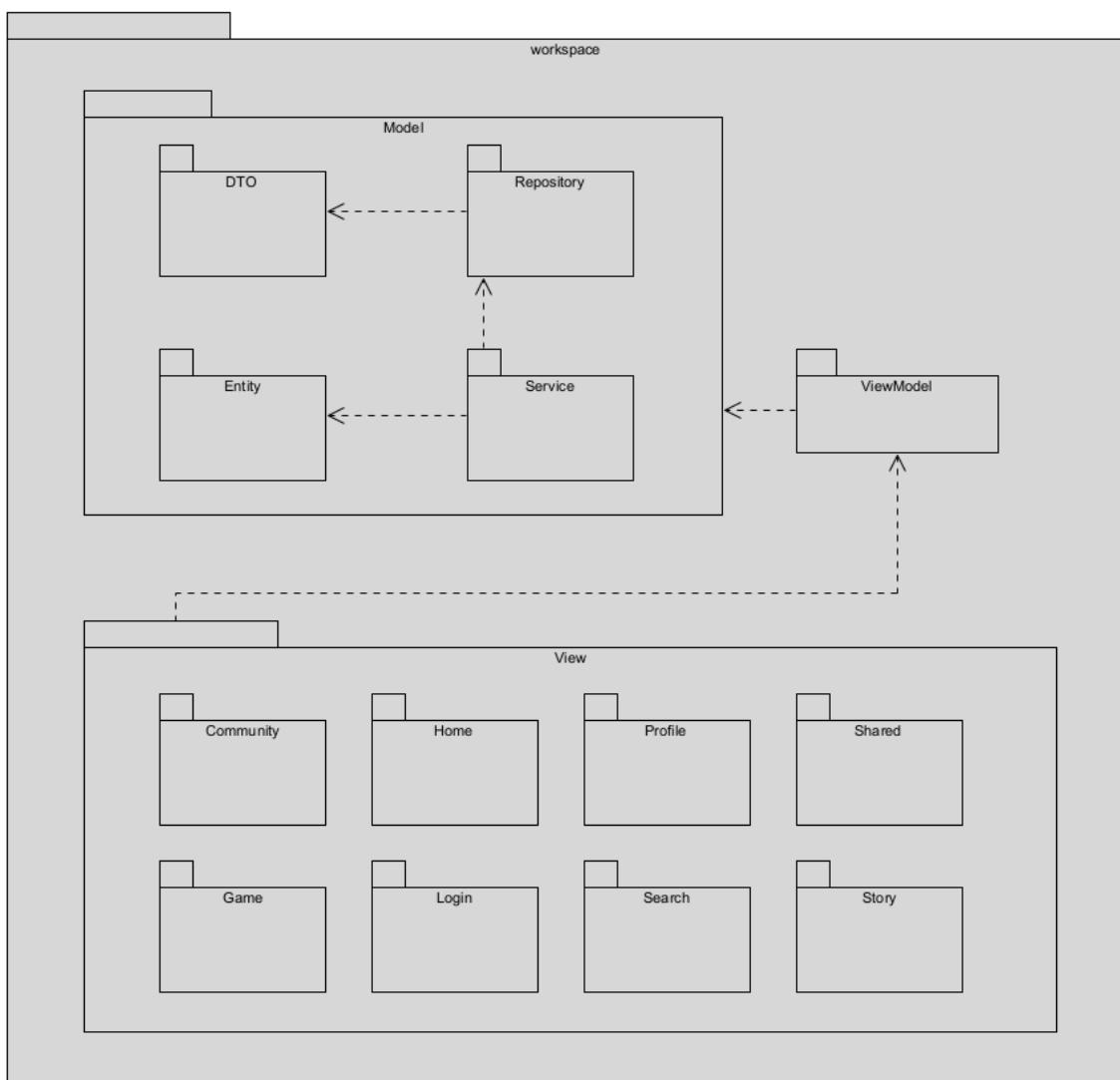
6. Projekt produktu programowego

6.1. Projekt architektury

Projekt architektury fizycznej systemu w postaci diagramu rozmieszczenia:



Projekt architektury logicznej systemu w postaci diagramu pakietów:

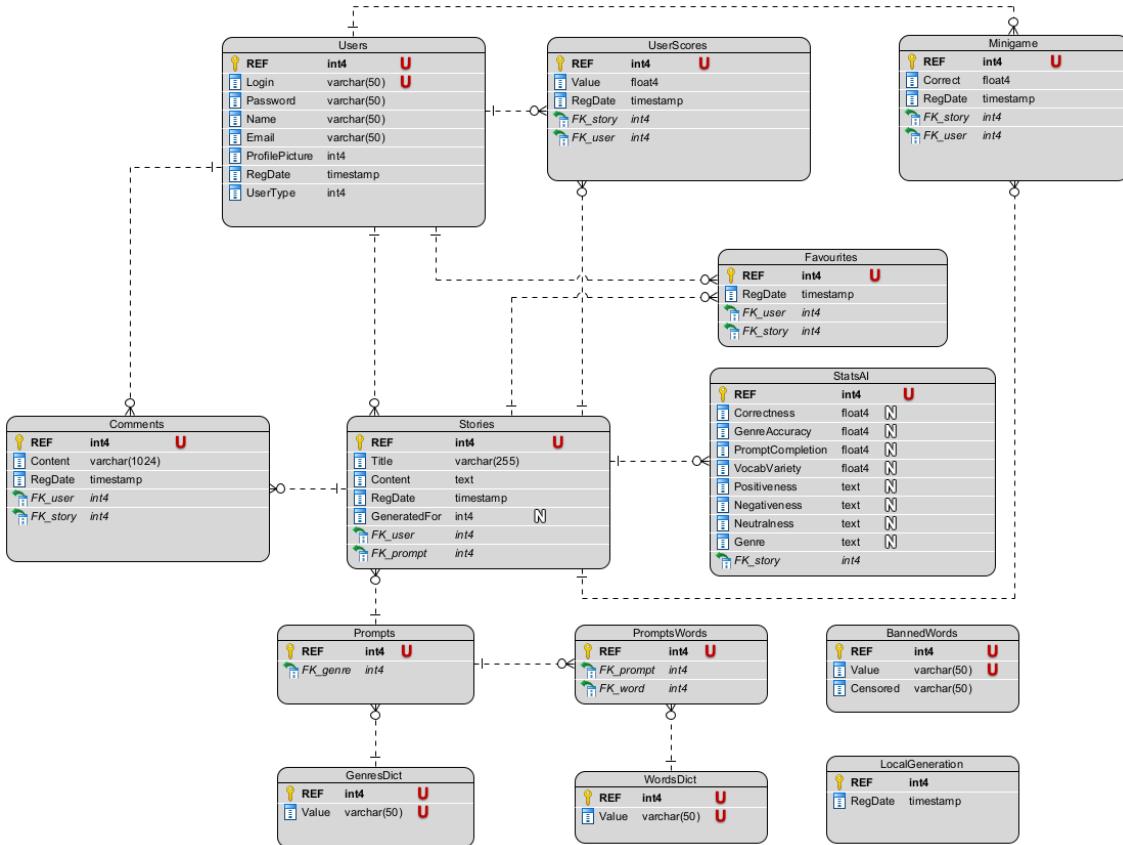


6.2. Projekt bazy danych

Typ bazy danych: relacyjna

System zarządzania bazą danych: PostgreSQL

Struktura bazy danych:



Znaczenie encji i ich atrybutów:

- **Users** – tabela reprezentująca encję użytkownik i przechowującą informacje o użytkowniku takie jak jego login, hasło, nazwę wyświetlaną w aplikacji, adres e-mail służący do resetowania hasła użytkownika, indeks zdjęcia profilowego, datę rejestracji oraz rodzaj użytkownika (wartość 0 oznacza standardowego użytkownika, wartość 1 oznacza użytkownika zalogowanego za pomocą Google – hasła i adresy e-mail użytkowników zalogowanych za pomocą Google nie są przechowywane).
- **Stories** – tabela reprezentująca encję opowiadanie i przechowującą informacje o opowiadaniu takie jak jego tytuł i treść, datę publikacji i wyzwanie, pod które napisane zostało opowiadanie, jak również klucze obce do autora opowiadania i do opowiadania, dla którego opowiadanie zostało wygenerowane (pole GeneratedFor ma wartość null dla opowiadań napisanych przez użytkowników i wartość różną od null dla opowiadań wygenerowanych przez moduł AI).
- **Comments** – tabela reprezentująca encję komentarz i przechowującą informacje o komentarzu takie jak jego treść i datę publikacji oraz klucze obce do autora komentarza i do opowiadania, którego dotyczy komentarz.
- **Prompts** – tabela reprezentująca encję wyzwanie i przechowującą informację o gatunku przypisany do wyzwania.

- **PromptsWords** – tabela asocjacyjna łącząca tabelę Prompts ze słownikiem WordsDict, dzięki czemu możliwe jest przechowywanie informacji o słowach przypisanych do wyzwania.
- **Favourites** – tabela asocjacyjna łącząca tabelę Stories z tabelą Users i reprezentująca dodanie opowiadania do ulubionych przez użytkownika. Tabela przechowuje dodatkowo informację o dacie dodania opowiadania do ulubionych. Użytkownik może dodawać do ulubionych tylko opowiadania innych użytkowników.
- **UserScores** – tabela reprezentująca encję oceny użytkownika i przechowująca informacje o ocenie użytkownika takie jak wartość oceny, data wystawienia oceny oraz klucze obce do użytkownika, który wystawił ocenę i do opowiadania, którego dotyczy ocena.
- **Minigame** – tabela reprezentująca encję wyników minigry i przechowująca informacje o wynikach użytkowników grających w minigrę takie jak wartość punktowa odpowiedzi, data wystawienia odpowiedzi oraz klucze obce do użytkownika, który zagrał w minigrę i do opowiadania, które zostało wylosowane do minigry. Do minigry losowane są opowiadania wygenerowane przez moduł AI i opowiadania napisane przez użytkowników, dla których zostały wygenerowane. Tabela Minigame posiada klucz obcy do opowiadania wygenerowanego przez moduł AI.
- **StatsAI** – tabela reprezentująca encję oceny modułu AI i przechowująca informacje o ocenie wystawionej przez moduł AI takie jak poziom poprawności językowej opowiadania (Correctness), poziom zgodności z gatunkiem wylosowanym w wyzwaniu (GenreAccuracy), poziom wykorzystania słów wylosowanych w wyzwaniu (PromptCompletion), poziom zróżnicowania słów wykorzystanych w opowiadaniu (VocabVariety), ton, w jakim napisane jest opowiadanie (pozytywny – Positiveness, negatywny – Negativeness lub neutralny – Neutralness), gatunek przypisany opowiadaniu przez moduł AI (Genre) oraz klucz obcy do opowiadania, którego dotyczy ocena modułu AI.
- **GenresDict, WordsDict, BannedWords** – słowniki odpowiednio gatunków i słów do wyzwań oraz słów uznawanych za obraźliwe lub niestosowne, których nie można wykorzystywać w aplikacji.
- **LocalGeneration** – tabela wykorzystywana do zaznaczenia daty ostatniego generowania opowiadań przez moduł generowania tekstu. W związku z problemem znalezienia darmowego serwisu hostowania, który udostępnia wystarczająco zasobów, aby moduł generowania był w stanie funkcjonować, teksty generowane są lokalnie. Lokalne generowanie opowiadania następują opowiadania wygenerowane w chmurze za pośrednictwem API.

6.3. Skrypt SQL DDL

Skrypt został wygenerowany przez Visual Paradigm na podstawie diagramu encji.

```
CREATE TABLE Users (
    REF      SERIAL NOT NULL,
    Login    varchar(50) NOT NULL UNIQUE,
    Password varchar(50) NOT NULL,
    Name     varchar(50) NOT NULL,
    Email    varchar(50) NOT NULL,
    ProfilePicture int4 NOT NULL,
    RegDate   timestamp NOT NULL,
    UserType  int4 NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE Stories (
    REF      SERIAL NOT NULL,
    Title    varchar(255) NOT NULL,
    Content   text NOT NULL,
    RegDate   timestamp NOT NULL,
    GeneratedFor int4 NOT NULL,
    FK_user   int4 NOT NULL,
    FK_prompt  int4 NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE WordsDict (
    REF      SERIAL NOT NULL,
    Value   varchar(50) NOT NULL UNIQUE,
    PRIMARY KEY (REF));

CREATE TABLE GenresDict (
    REF      SERIAL NOT NULL,
    Value   varchar(50) NOT NULL UNIQUE,
    PRIMARY KEY (REF));

CREATE TABLE Prompts (
    REF      SERIAL NOT NULL,
    FK_genre int4 NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE Comments (
    REF      SERIAL NOT NULL,
    Content  varchar(1024) NOT NULL,
    RegDate   timestamp NOT NULL,
    FK_user   int4 NOT NULL,
    FK_story  int4 NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE UserScores (
    REF      SERIAL NOT NULL,
    Value   float4 NOT NULL,
    RegDate   timestamp NOT NULL,
    FK_story  int4 NOT NULL,
    FK_user   int4 NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE Favourites (
    REF      SERIAL NOT NULL,
    RegDate   timestamp NOT NULL,
    FK_user   int4 NOT NULL,
    FK_story  int4 NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE PromptsWords (
    REF      SERIAL NOT NULL,
    FK_prompt int4 NOT NULL,
    FK_word   int4 NOT NULL,
    PRIMARY KEY (REF));
```

```

CREATE TABLE Minigame (
    REF      SERIAL NOT NULL,
    Correct  float4 NOT NULL,
    RegDate  timestamp NOT NULL,
    FK_story int4 NOT NULL,
    FK_user  int4 NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE StatsAI (
    REF          SERIAL NOT NULL,
    Correctness  float4,
    GenreAccuracy float4,
    PromptCompletion float4,
    VocabVariety float4,
    Positiveness  text,
    Negativeness  text,
    Neutralness   text,
    Genre         text,
    FK_story      int4 NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE BannedWords (
    REF      SERIAL NOT NULL,
    Value    varchar(50) NOT NULL UNIQUE,
    Censored varchar(50) NOT NULL,
    PRIMARY KEY (REF));

CREATE TABLE LocalGeneration (
    REF      SERIAL NOT NULL,
    RegDate  timestamp NOT NULL,
    PRIMARY KEY (REF));

ALTER TABLE Favourites
    ADD UNIQUE (FK_user, FK_story);

ALTER TABLE UserScores
    ADD UNIQUE (FK_user, FK_story);

ALTER TABLE Minigame
    ADD UNIQUE (FK_user, FK_story);

ALTER TABLE Prompts
    ADD CONSTRAINT FKPrompts333671
        FOREIGN KEY (FK_genre) REFERENCES GenresDict (REF);

ALTER TABLE Stories
    ADD CONSTRAINT FKStories397261
        FOREIGN KEY (FK_prompt) REFERENCES Prompts (REF);

ALTER TABLE Stories
    ADD CONSTRAINT FKStories238003
        FOREIGN KEY (FK_user) REFERENCES Users (REF);

ALTER TABLE Comments
    ADD CONSTRAINT FKComments176874
        FOREIGN KEY (FK_story) REFERENCES Stories (REF);

ALTER TABLE Comments
    ADD CONSTRAINT FKComments396688
        FOREIGN KEY (FK_user) REFERENCES Users (REF);

ALTER TABLE UserScores
    ADD CONSTRAINT FKUserScores780211
        FOREIGN KEY (FK_user) REFERENCES Users (REF);

ALTER TABLE UserScores
    ADD CONSTRAINT FKUserScores793350
        FOREIGN KEY (FK_story) REFERENCES Stories (REF);

```

```
ALTER TABLE Favourites
    ADD CONSTRAINT FKFavourites497606
        FOREIGN KEY (FK_story) REFERENCES Stories (REF);

ALTER TABLE Favourites
    ADD CONSTRAINT FKFavourites75956
        FOREIGN KEY (FK_user) REFERENCES Users (REF);

ALTER TABLE PromptsWords
    ADD CONSTRAINT FKPromptsWor704523
        FOREIGN KEY (FK_prompt) REFERENCES Prompts (REF);

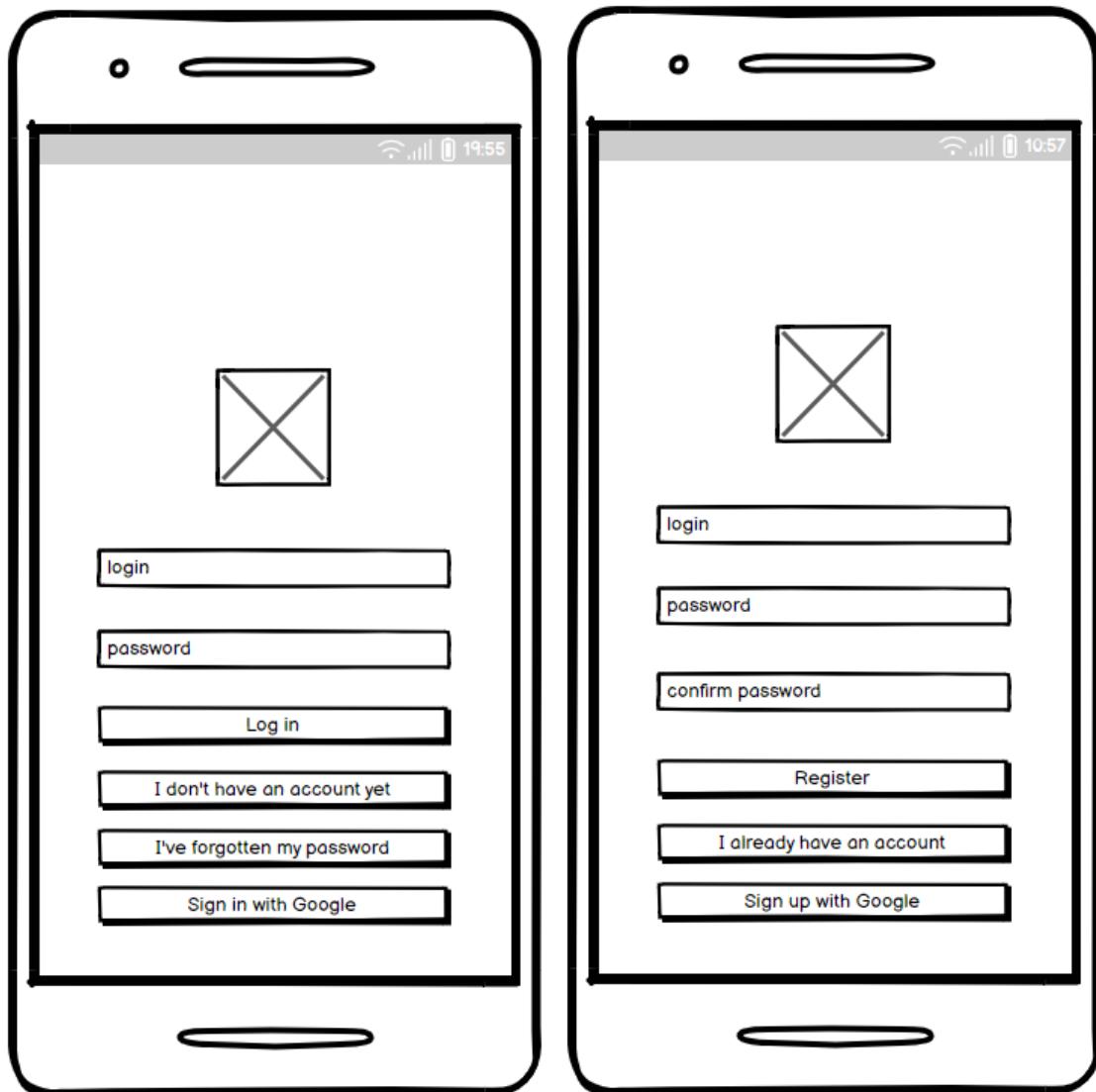
ALTER TABLE PromptsWords
    ADD CONSTRAINT FKPromptsWor358901
        FOREIGN KEY (FK_word) REFERENCES WordsDict (REF);

ALTER TABLE Minigame
    ADD CONSTRAINT FKMinigame236144
        FOREIGN KEY (FK_user) REFERENCES Users (REF);

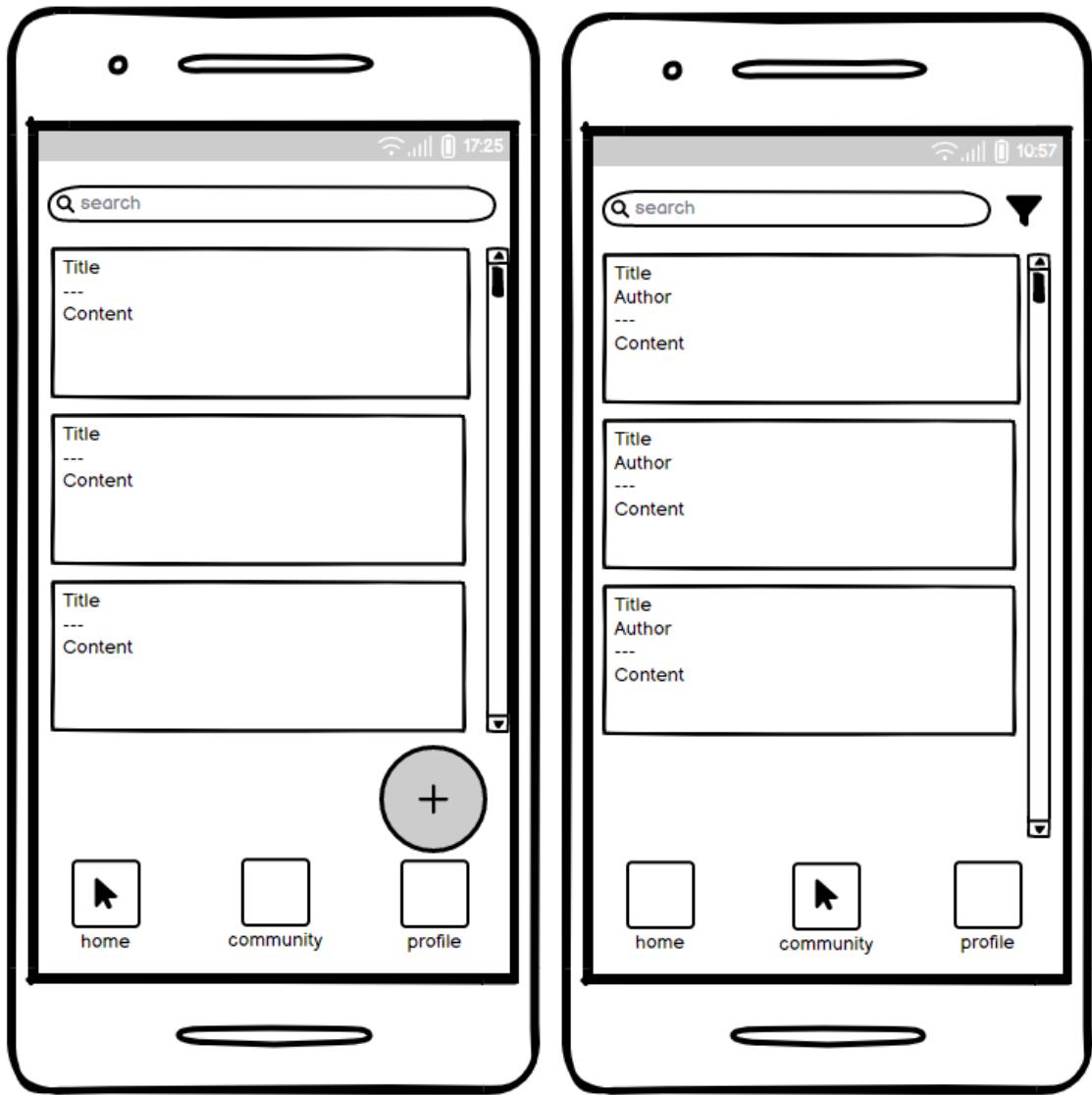
ALTER TABLE Minigame
    ADD CONSTRAINT FKMinigame337418
        FOREIGN KEY (FK_story) REFERENCES Stories (REF);
```

6.4. Makietы интерфейса приложения

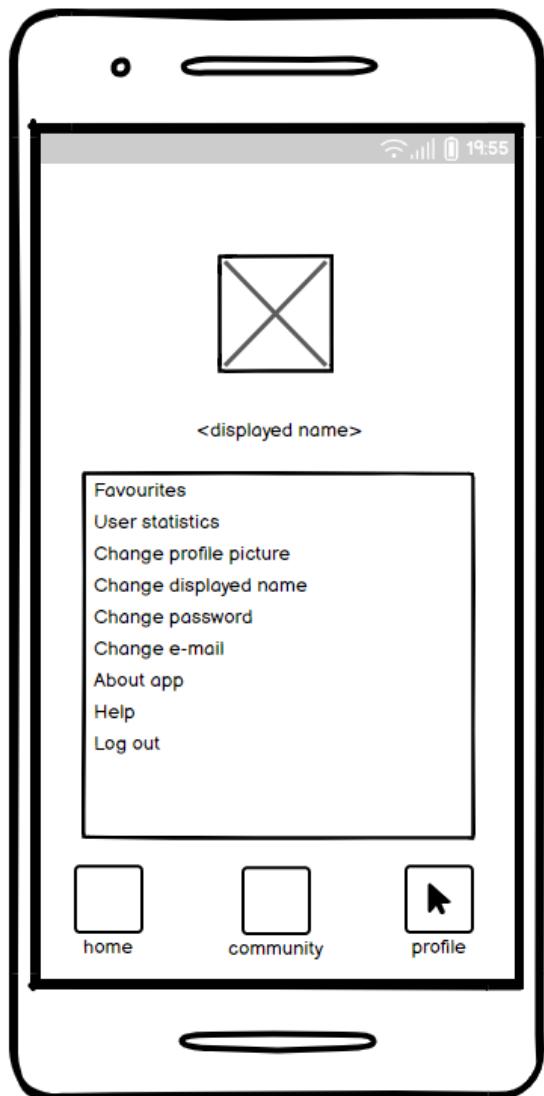
Макеты интерфейса не являются окончательным видом приложения. Были подготовлены как элемент процесса wireframing для последующего планирования дизайна и структуры приложения. После реализации видов необходимо обновление макетов.



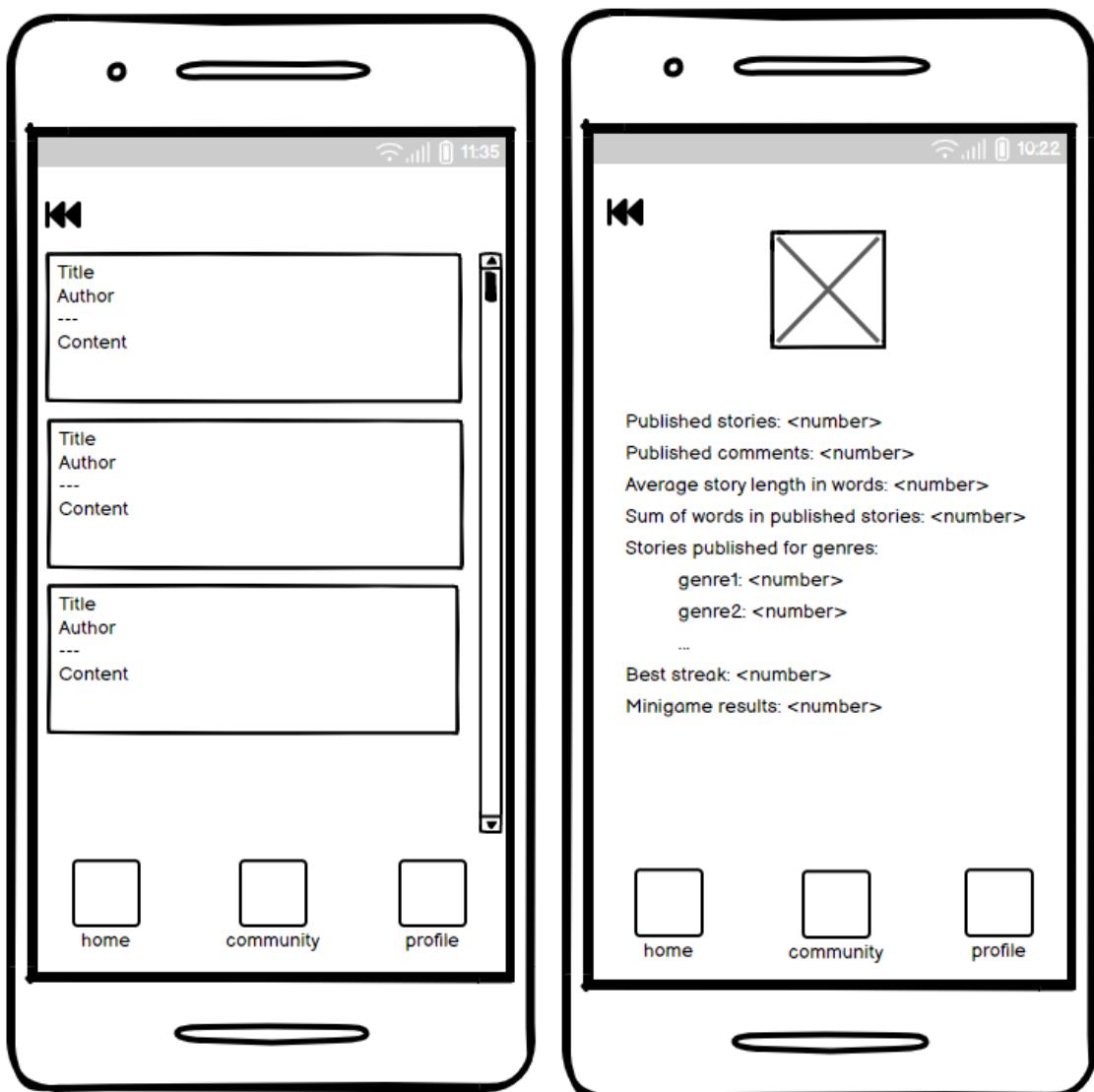
Макеты 1 – 2. Экран логования и регистрации



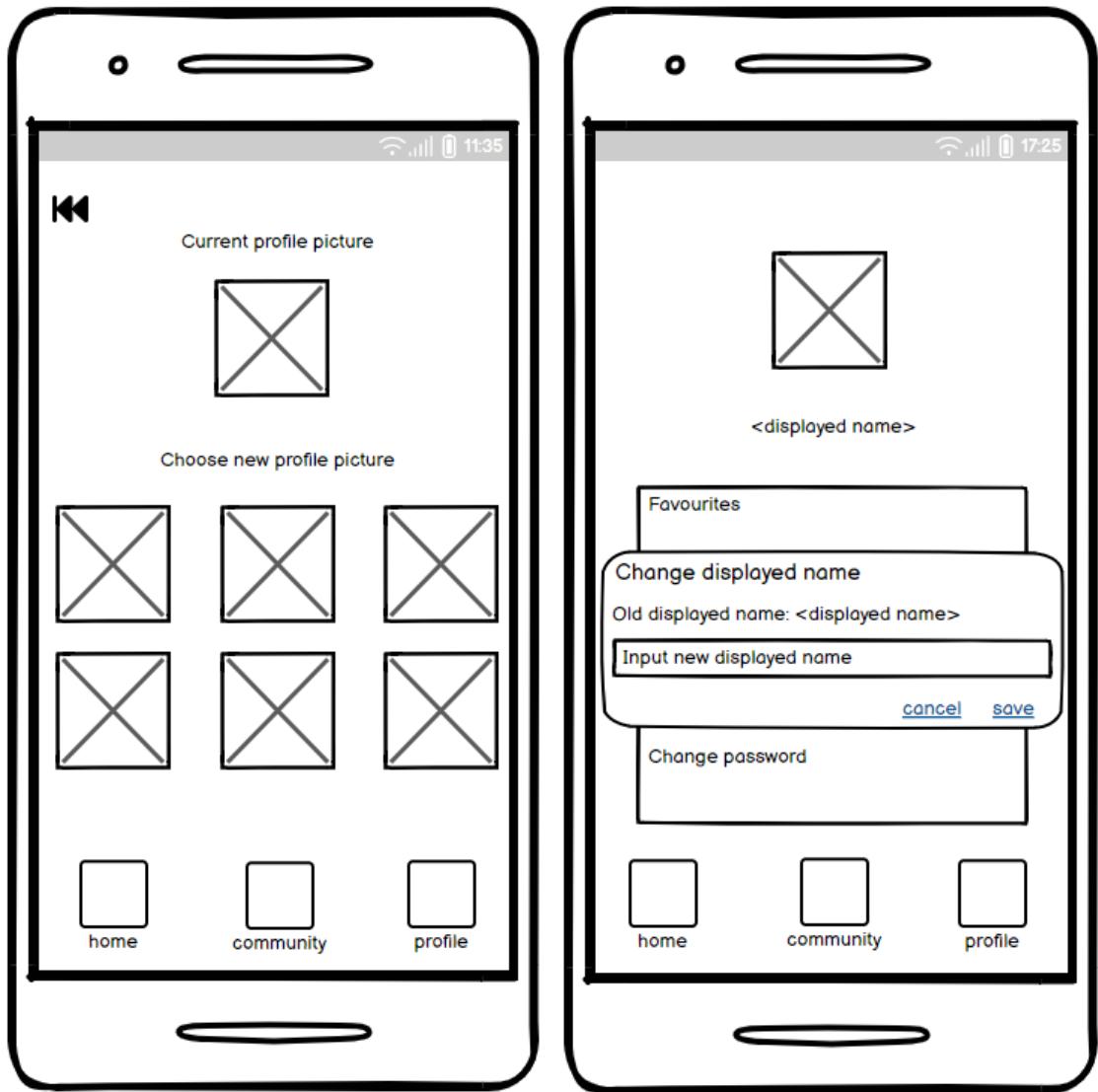
Makiety 3 – 4. Widok zakładki głównej (lista opowiadań użytkownika) oraz zakładki społeczności (lista opowiadań innych użytkowników)



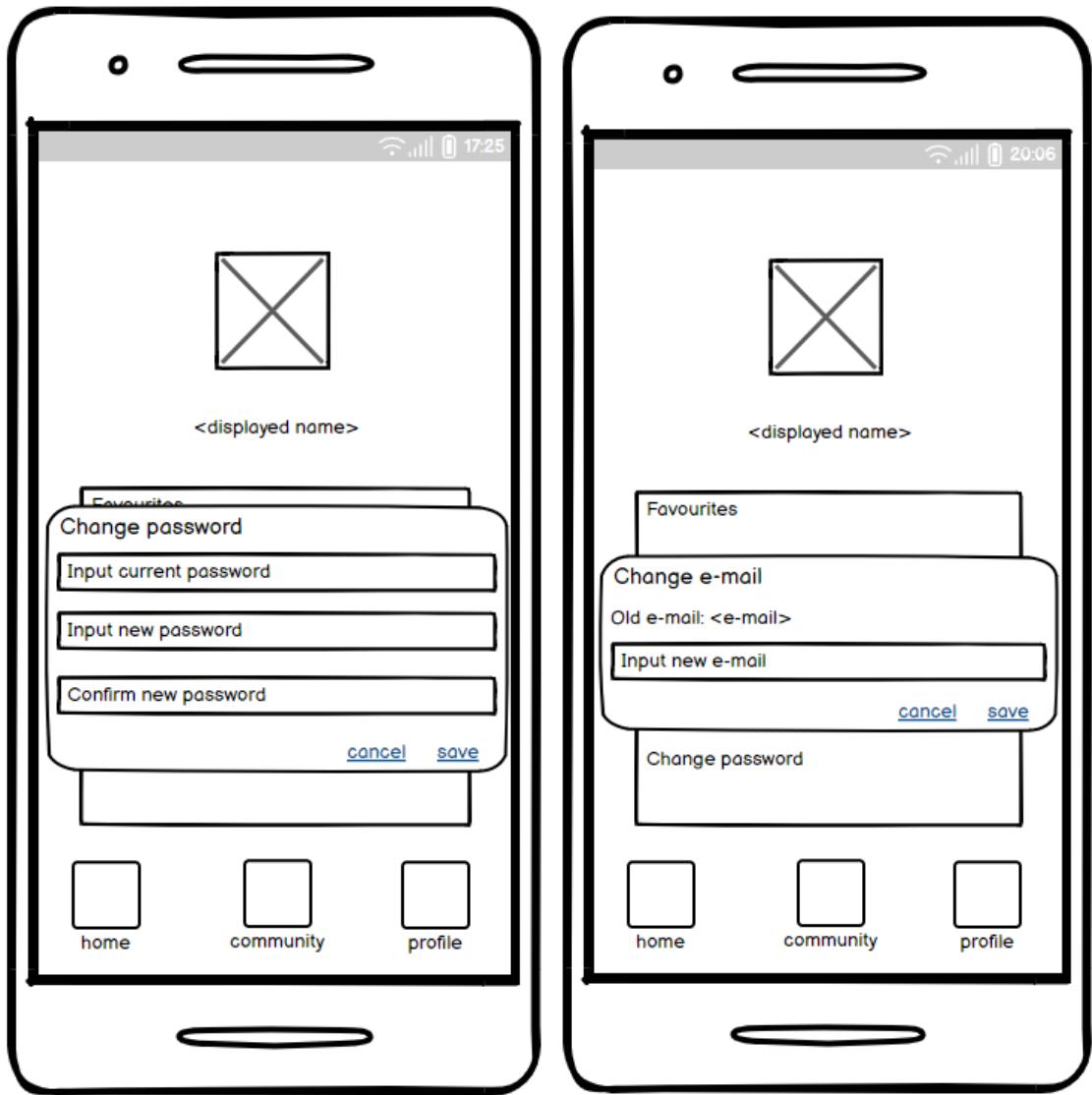
Makieta 5. Widok zakładki profilu użytkownika



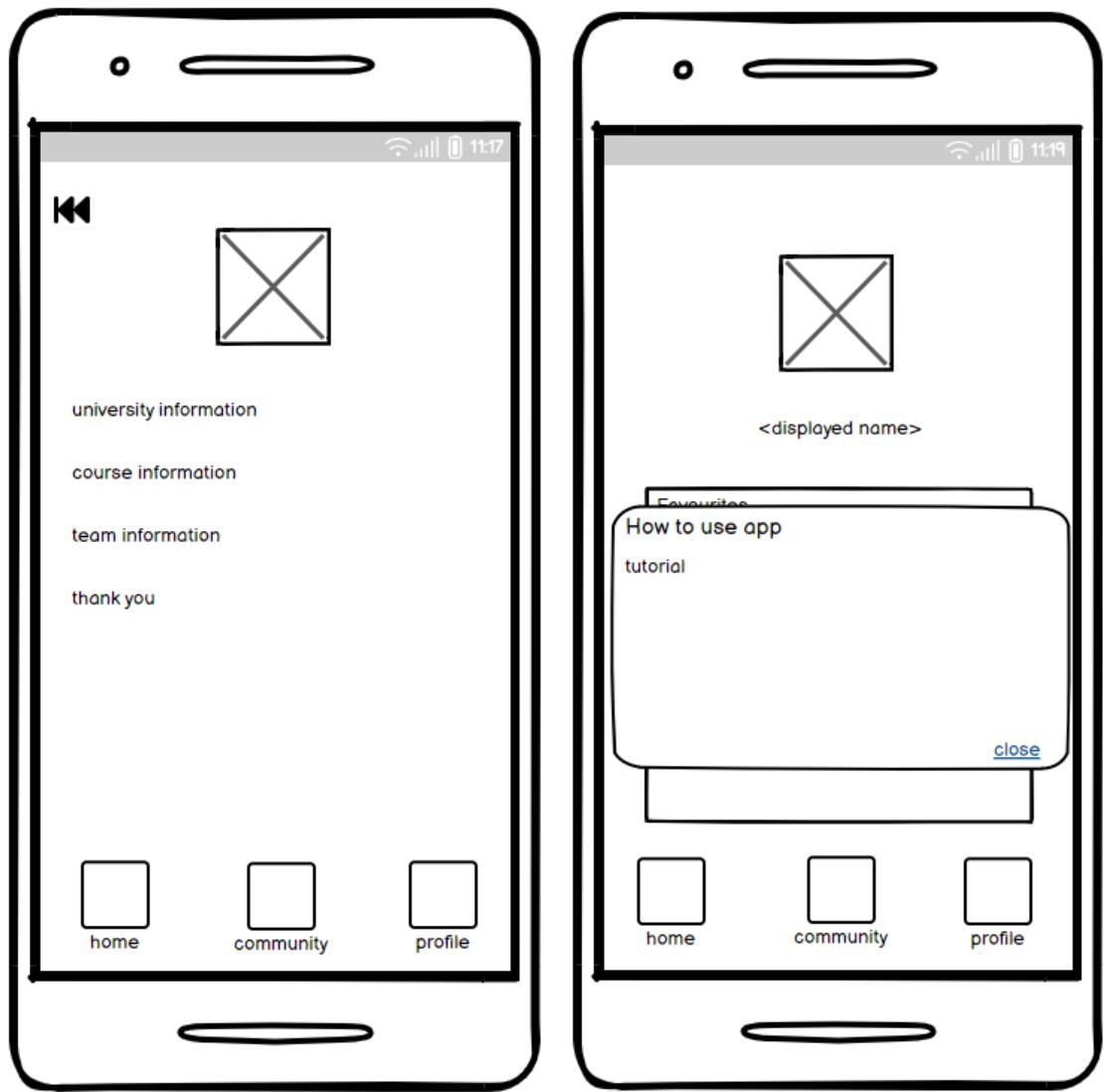
Makieta 6 – 7. Widok opowiadań, które użytkownik dodał do swoich ulubionych i widok statystyk użytkownika



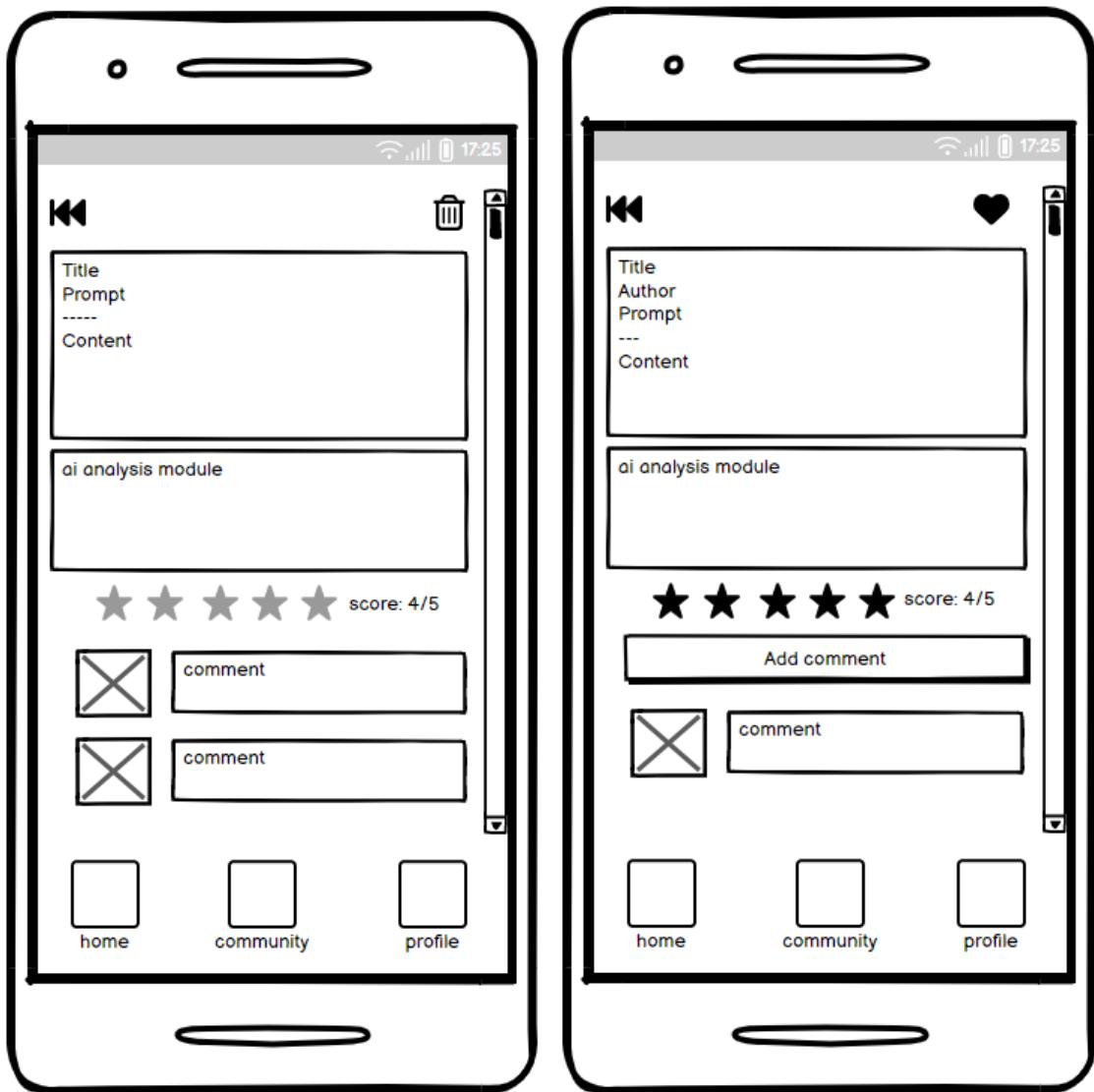
Makieta 8 – 9. Widok zmiany zdjęcia profilowego i widok zmiany wyświetlanej nazwy użytkownika



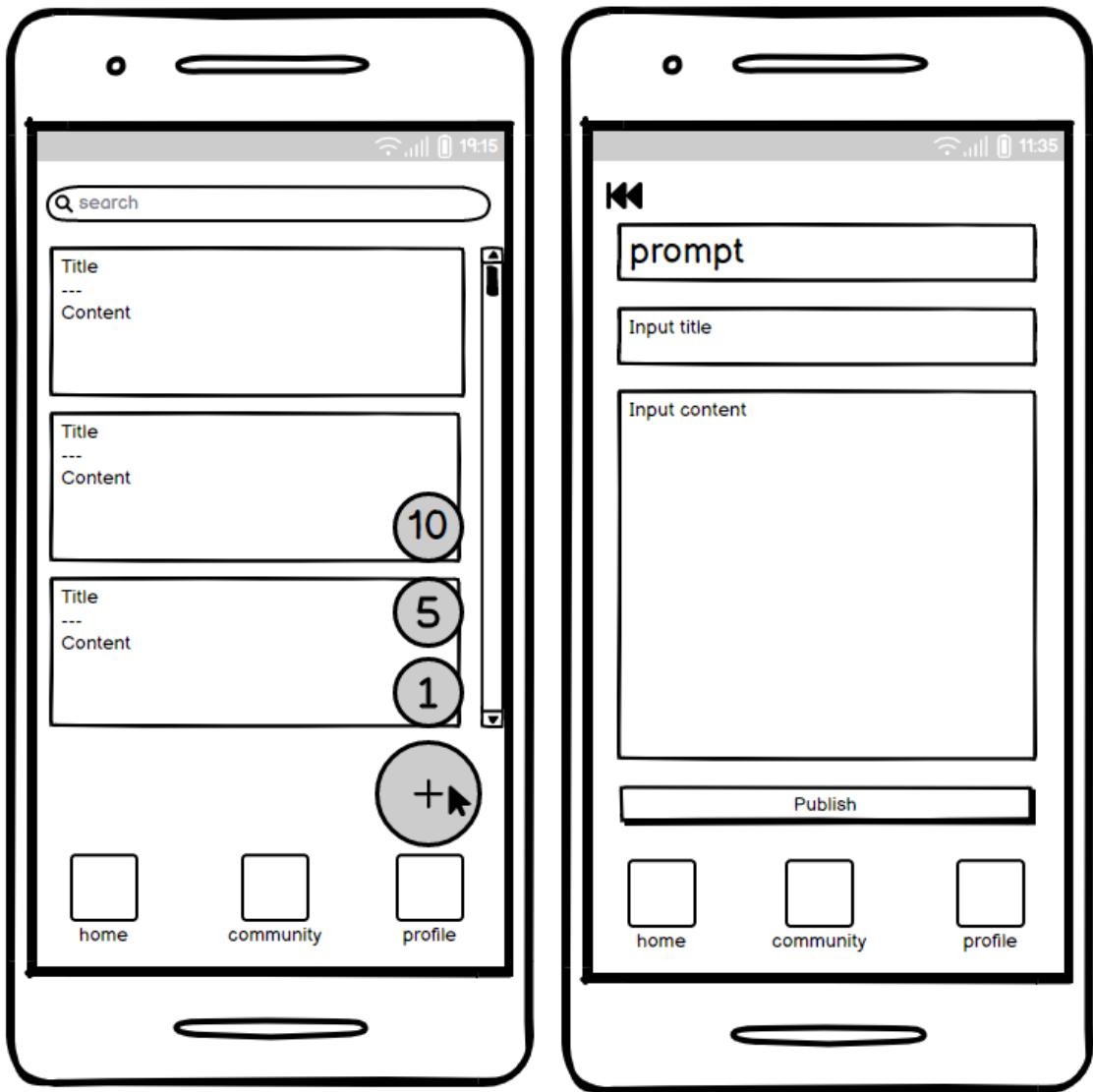
Makieta 10 – 11. Widok zmiany hasła użytkownika i widok zmiany adresu e-mail użytkownika



Makieta 12 – 13. Widok sekcji o aplikacji i widok sekcji pomoc

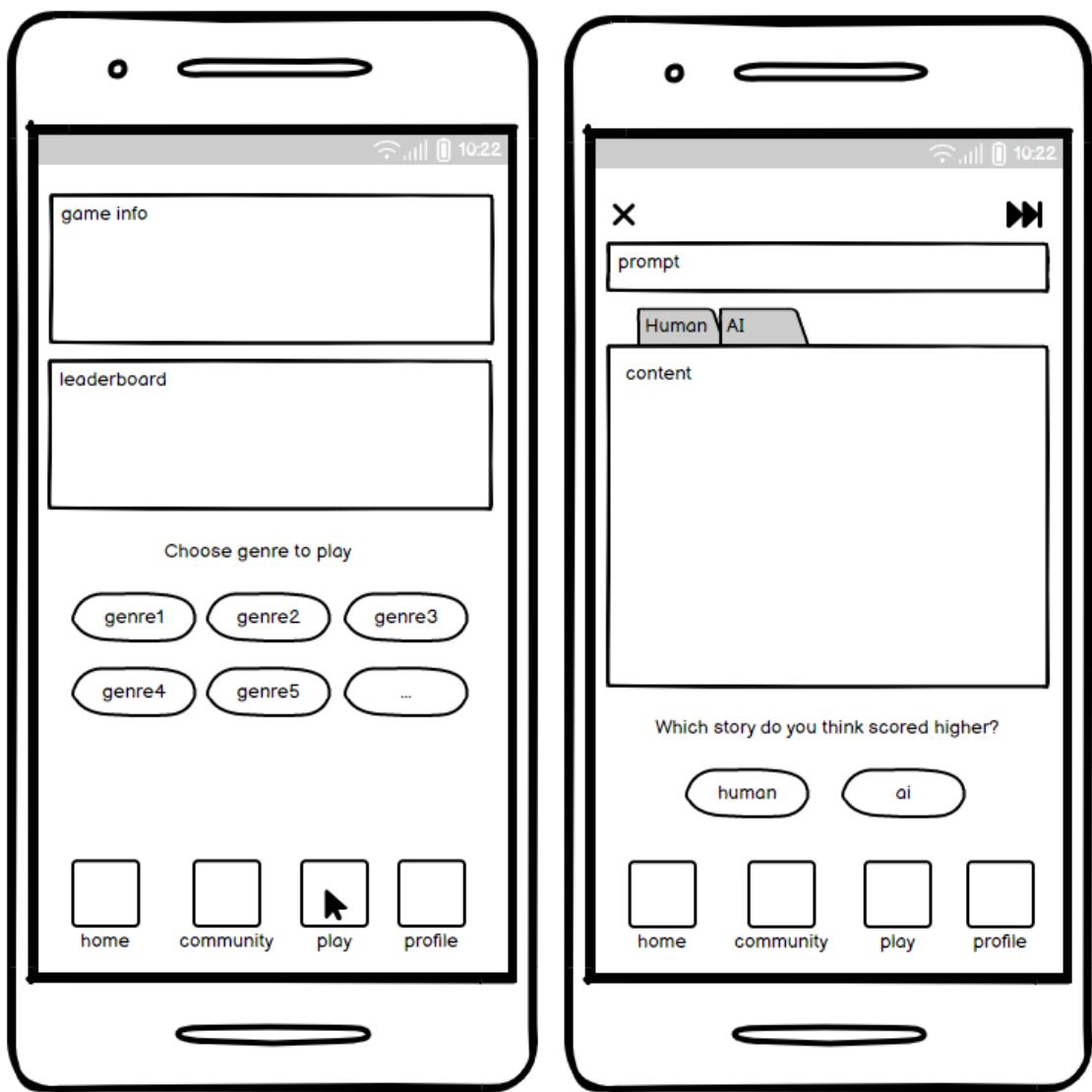


Makieta 14 – 15. Widok opowiadania napisanego przez użytkownika oraz opowiadania napisanego przez innego użytkownika



Makieta 16 – 17. Widok wyboru rodzaju nowego wyzwania oraz pisania nowego opowiadania dla wylosowanego wyzwania

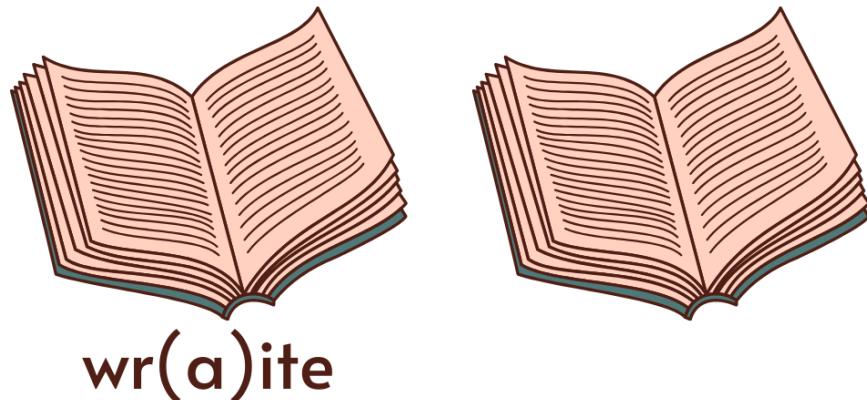
W ramach projektowania modułu generowania tekstu utworzono dodatkowe makiety interfejsów. Nie była konieczna modyfikacja poprzednich makiet, ponieważ na tym etapie funkcjonalności obejmowane przez nie były już całkowicie zaprojektowane i wstępnie zaimplementowane – zatem makiety spełniły swoją funkcję.



Makieta 18 – 19. Widok minigry, w której użytkownik próbuje zgadnąć, które z pary opowiadań napisanych pod to samo wyzwanie (jedno było napisane przez człowieka, a drugie wygenerowane przez model generowania tekstu) otrzymało lepszy wynik

6.5. Reguły przewodnika stylu

1. Logo aplikacji w wersji z podpisem i bez podpisu:



Logo aplikacji wykonano przy użyciu portalu Canva [\[16\]](#).

2. Wykorzystywane kolory:

- a. #191919
- b. #000000
- c. #FEFEFE
- d. #FFFFFF
- e. #BABABA
- f. #727272
- g. #373737
- h. #76B7AA
- i. #4D9385
- j. #21403A
- k. #A84551
- l. #722F37
- m. #3C191D
- n. #FFECE6
- o. #FFD1C1
- p. #B27763
- q. #4F1F15

Kolory inspirowane są paletą kolorów Montecristo [\[17\]](#) zaprojektowaną przez użytkownika cersin7.

3. Wykorzystywana czcionka:

a. Alata regular [18]

ABCDEFGHIJKLMNOPRSTUVXYZĄĘŁÓŞŻŻ
abcdefghijklmnoprstuvxyząęłóśżż
1234567890!@#\$%^&*()

b. Alata bold [18]

ABCDEFGHIJKLMNOPRSTUVXYZĄĘŁÓŞŻŻ
abcdefghijklmnoprstuvxyząęłóśżż
1234567890!@#\$%^&*()

c. na przyciskach “Sign in with Google” oraz “Sign up with Google”
Roboto bold [19]

ABCDEFGHIJKLMNOPRSTUVXYZĄĘŁÓŞŻŻ
abcdefghijklmnoprstuvxyząęłóśżż
1234567890!@#\$%^&*()

4. Wykorzystane ikony:



Nazwa użytkownika



Statystyki



Zdjęcie profilowe



Zakładka profil (Profile)



Hasło



Wyloguj



Zakładka główna (Your stories)



Zakładka społeczności (Community)



Dodaj do ulubionych



Ulubione / Usuń z ulubionych



Usuń



Powrót



Następne



Zamknij



Pokaż więcej



Pokaż mniej



Minigra



Dodaj opowiadanie



Wyszukaj



Pokaż filtry



Informacje o aplikacji



Wyświetl pomoc



Adres e-mail skojarzony z kontem



Ikona Google



Wygeneruj zachętą z jednym losowym słowem



Wygeneruj zachętą z trzema losowymi słowami



Wygeneruj zachętą z pięcioma losowymi słowami

Ikony (poza ikonami z cyframi oraz ikoną Google) pochodzą ze zbioru Google Material Icons [20]. Wszystkie ikony ze zbioru Google Material Icons wykonane są w stylu rounded. Ikona Google pochodzi z udostępnionego przez Google dokumentu Sign-In Branding Guidelines [21]. Przyciski “Sign in with Google” i “Sign up with Google” wykonano zgodnie z wytycznymi dotyczącymi marki przedstawionymi w tym dokumencie. Ikony z cyframi wykonano przy użyciu portalu Canva [16]. Wykorzystano czcionkę Alata regular [18] i Alata bold [18]. Kolory ikon są dostosowane do wyglądu aplikacji.

5. Wykorzystane awatary:



Wykorzystane awatary pochodzą ze zbioru Freepik i zostały wykonane przez użytkownika pikisuperstar [\[22\]](#), [\[23\]](#).

6. Aplikacja jest dostępna w stylu jasnym i ciemnym. Styl jasny jest domyślny. Styl ciemny włącza się dla użytkowników, którzy włączyli tryb ciemny w ustawieniach systemu operacyjnego Android.

6.6. Mockupu interfejsu



wr(a)ite

Login

Password

Log in

I've forgotten my password

I've forgotten my login

I don't have an account yet

 Sign in with Google



wr(a)ite

Please create a unique login. Your login will only be used to log you into your account. You can change your displayed name later. Your email will only be used if you forget your password.

Login

Password

Confirm password

E-mail address

Register

I already have an account

 Sign up with Google

Interfejs 1 – 2. Ekran logowania i rejestracji



Out of the Blue

test_user

Liv was tired. Very, very tired.
Another day on the lines and her eyes were aching like sagging balloons, ready to burst from the strain required to just stay employed at that dump. She laced her fingers together and pushed her arms above her head as she came to the end of her daily slog to the bus stop. The movement rewarded her with a series of satisfying pops along her cramped joints.
The bench at the bus stop was hard and damp, mildew curled around the edges. But, her aching legs demanded r...

Midsommar

test_user

For the final ceremony, the commune leaders explain that the commune must offer nine human sacrifices to purge it of its evil. The first four victims are outsiders lured to them by Pelle and Ingemar, while the next four victims must be from the commune. As May Queen, Dani must choose either Christian or a commune member to be the final sacrificial victim. She chooses Christian, who is stuffed into a disemboweled brown bear's body and placed in a triangular wooden temple alongside other sacrifice...

The Day from Hell

api_test

I actually used to work as a cashier and the following scenarios actually happened, along with many others. These two customers were so...eccentric that they are hard to forget. I remember working long hours and trying to not be rude to some people who acted in really bizarre ways. One lady asked if there was anyone in my life who loved me. The nerve...I know. I politely responded that there was. She donated to the charity we were fundraising for soon after. Another customer told me that my chak...

Child of fog

Aga

It was the first day of summer, the start of holidays. Sun was high in the sky when they came. Chemical cloud covered the world.
No one was save. Few were brave enough to go out of hiding and fight the monsters, that stared coming out front the fog. In that time I made a mistake. I took under my care a stray child i found on the streets. Child that, without my knowledge, was going out every night to run with the wolves.
It was months before I found out. Kid that I treated as mine was changed b...

There are No Saints Here

test

Cordelia had yet to figure out when she realised she first



Interfejs 3 – 4. Widok zakładki głównej (lista opowiadań użytkownika) oraz zakładki społeczności (lista opowiadań innych użytkowników)



test_user

♥ Favourites

⚡ Statistics

👤 Change profile picture

Ⓣ Change name

🔑 Change password

✉ Change e-mail address

ℹ About wr(a)ite

❓ Help

🔒 Log out



Profile

Interfejs 5. Widok zakładki profilu użytkownika

← Your favourites

There are No Saints Here

test

Cordelia had yet to figure out when she realised she first hated Joseph.

Perhaps it was because of her fierce competitiveness, her need to win. And unlike others, Joseph would never yield, constantly grabbing the title of the best dancer. Or perhaps it was because he bullied her, calling her talentless and useless. As he had proved multiple times, he did not believe her worthy of being a dancer.

While she only truly despised him for his callous attitude towards her and his immaculate dance tech...

Child of fog

Aga

It was the first day of summer, the start of holidays. Sun was high in the sky when they came. Chemical cloud covered the world.

No one was safe. Few were brave enough to go out of hiding and fight the monsters, that stared coming out front the fog. In that time I made a mistake. I took under my care a stray child i found on the streets. Child that, without my knowledge, was going out every night to run with the wolves.

It was months before I found out. Kid that I treated as mine was changed b...

← Your statistics



Published stories: 6

Published comments: 0

Comments published under your stories: 2

Stories added to favourites: 2

Your stories have been added to favourites: 5 times

Your stories have been rated: 4 times

Average story length in words: 71.33

Sum of words in published stories: 428

Best streak: 2 days

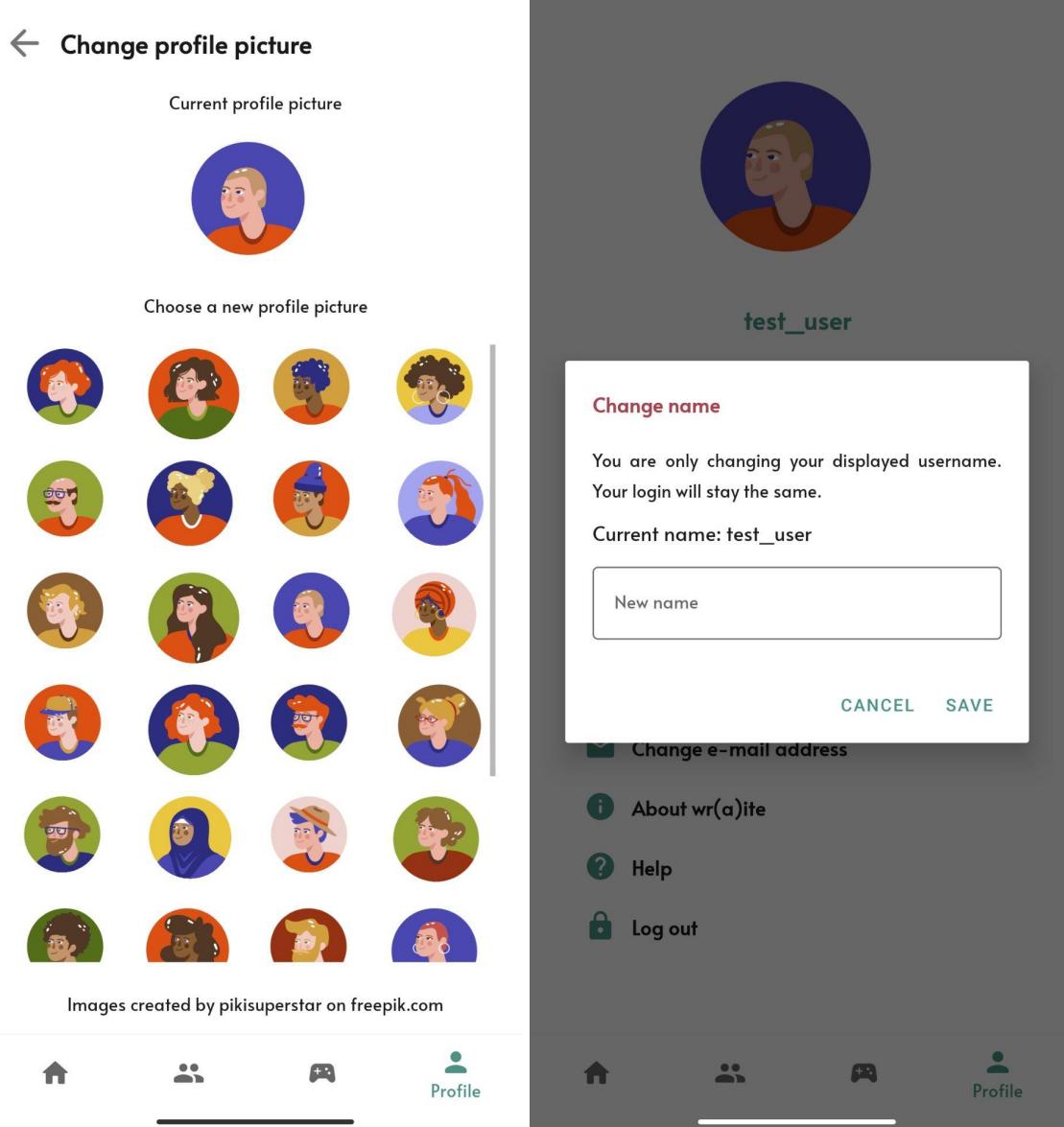
Minigame score: 2

Stories published per genre:

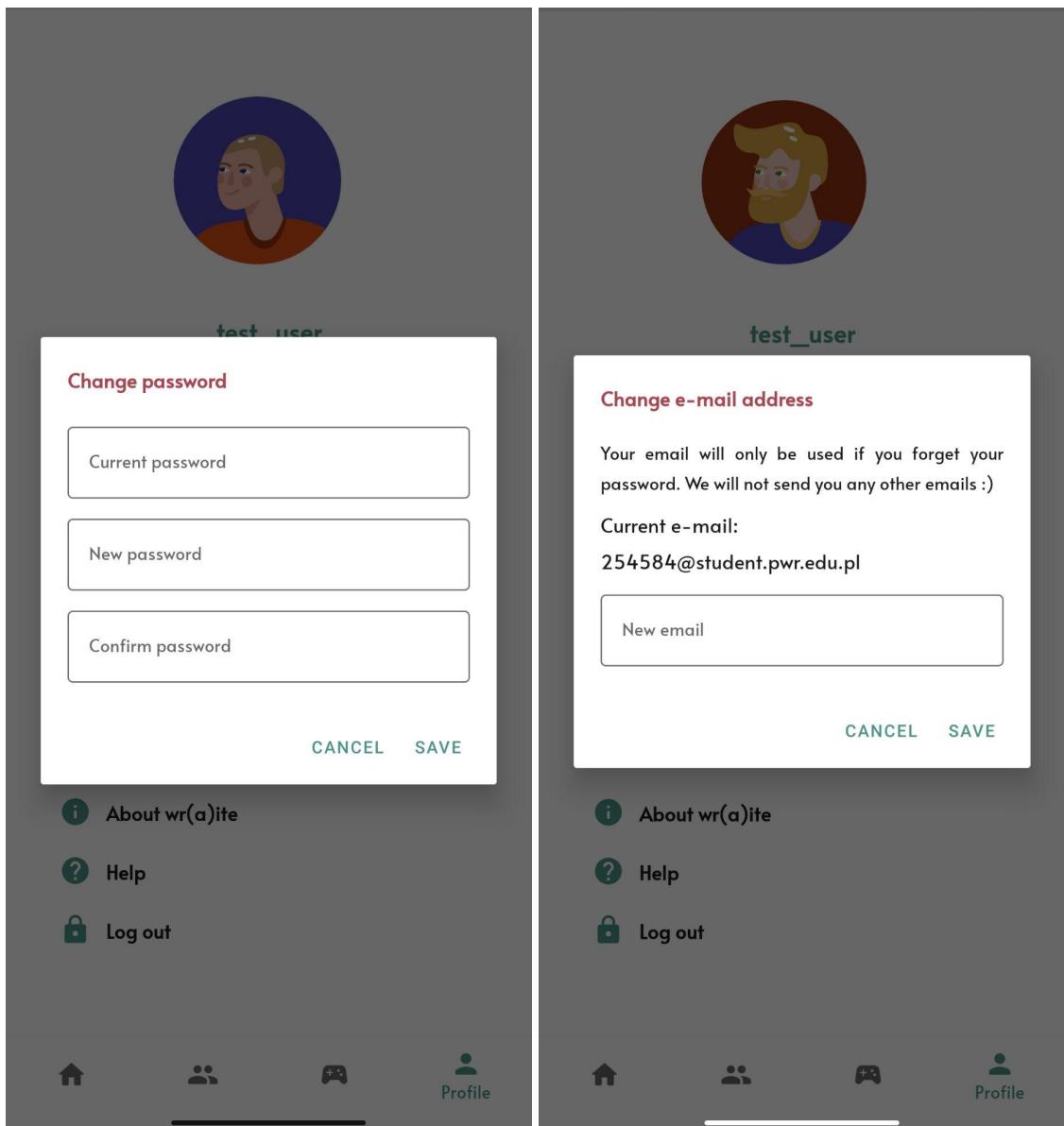
- historical 2
- comedy 1
- horror 2
- adventure 1



Interfejs 6 – 7. Widok opowiadań, które użytkownik dodał do swoich ulubionych i widok statystyk użytkownika



Interfejs 8 – 9. Widok zmiany zdjęcia profilowego i widok zmiany wyświetlanej nazwy użytkownika



Interfejs 10 – 11. Widok zmiany hasła użytkownika i widok zmiany adresu e-mail użytkownika

← About wr(a)ite



We are a team of young programmers studying Applied Computer Science at Wrocław University of Technology. This app was born out of our mutual interest in mobile applications, artificial intelligence and writing.

Our team:

- Agnieszka Kłobus
- Aleksandra Koperwas
- Aleksandra Stecka
- Bogusława Tłońska

We created this app in our 7th semester — the last semester of our engineering degree. wr(a)ite was created in 2022 as part of our Team Engineering Project course. The official title of the project is "Mobile application for the development of writing skills using artificial intelligence for text creation and analysis".

Thank you for using wr(a)ite!

The screenshot shows the 'About' section of the app. At the top is a circular profile picture of a person with short blonde hair. Below it is a white rectangular box containing the text 'How to use?'. Underneath is another white box with the text 'Welcome to wr(a)ite — the app to help you develop your writing skills.' followed by 'Did you know?'. Below that is another white box with the text 'The name wr(a)ite is a combination of the words "write" and "AI". We know it's cheesy :)' followed by 'Before you begin let us show you how to use our app.' At the bottom right of the main screen is a 'NEXT' button. Below the main screen are two smaller navigation bars, each with icons for Home, People, Games, and Profile, with the 'Profile' icon being the active one.

Interfejs 12 – 13. Widok sekcji o aplikacji i widok sekcji pomocy

← Midsommar

Delete

Genre: horror

Prompt: certain, send, general

ORIGINAL STORY

Hide

For the final ceremony, the commune leaders explain that the commune must offer nine human sacrifices to purge it of its evil. The first four victims are outsiders lured to them by Pelle and Ingemar, while the next four victims must be from the commune. As May Queen, Dani must choose either Christian or a commune member to be the final sacrificial victim. She chooses Christian, who is stuffed into a disemboweled brown bear's body and placed in a triangular wooden temple alongside other sacrifices. The commune members to be sacrificed are given drugs and told it will prevent them from feeling fear or pain, but Christian is not and remains unable to move. The structure is set on fire, and the commune members mimic the screams and wails of those being burned alive. Dani initially sobs in horror and grief, but then she gradually begins to smile.

AI GENERATED STORY

Show

AI ANALYSIS

Show info

Genre of text: romance

Compatibility with generated genre: 50%

Prompt completion: 50%

Correctness: 50%

Vocabulary variety: 59%

For the final ceremony, the commune leaders explain that the commune must offer nine human sacrifices to purge it of its evil. The first four victims are outsiders lured to them by Pelle and Ingemar, while the next four victims must be from the commune. As May Queen, Dani must choose either Christian or a commune member to be the final sacrificial victim. She chooses Christian, who is stuffed into a disemboweled brown bear's body and placed in a triangular wooden temple alongside other sacrifices. The commune members to be sacrificed are given drugs and told it will prevent them from feeling fear or pain, but Christian is not and remains unable to move. The structure is set on fire, and the commune members mimic the screams and wails of those being burned alive. Dani initially sobs in horror and grief, but then she gradually begins to smile.

AI GENERATED STORY

Show

AI ANALYSIS

Show info

Genre of text: romance

Compatibility with generated genre: 50%

Prompt completion: 50%

Correctness: 50%

Vocabulary variety: 59%

Positiveness: 34%

Negativeness: 24%

Neutralness: 41%

Average score: 4.5/5.0



test
omg

Your stories

Your stories

Interfejs 14. Widok opowiadania napisanego przez użytkownika

← There are No Saints Here

Unfavourite 

Genre: romance

Prompt: send, within, composed, variety, train

Written by: test

ORIGINAL STORY

Hide 

Cordelia had yet to figure out when she realised she first hated Joseph.

Perhaps it was because of her fierce competitiveness, her need to win. And unlike others, Joseph would never yield, constantly grabbing the title of the best dancer. Or perhaps it was because he bullied her, calling her talentless and useless. As he had proved multiple times, he did not believe her worthy of being a dancer.

While she only truly despised him for his callous attitude towards her and his immaculate dance technique, Cordelia could not decipher what she felt around him. When she gazed upon his face, she didn't understand whether she wanted to kiss him or strangle him. She considered it highly frustrating, however; she tried not to dwell too much upon it. He was a pompous, arrogant, brainless idiot who was only good at using his body to express. As if he could ever express anything with that face of his, she scoffed at her own thoughts.

Pulling herself out of her thoughts, Cordelia stretched her muscles, preparing herself for the barre. She moved her head left and right, hoping that she could get some practice done in time before the rest of the class arrived.

Pushing her shoulders back and assuming her usual posture, Cordelia began her routine.

The music started slow, a thrumming, peaceful melody that

filled the air and silenced the birds from their usual morning chipper. She bent her knees into a graceful plié, quickly springing onto her toes as she positioned her arms into a graceful arch. Taking a deep breath, she began her spins. Round and round until her view turned glassy, but she made sure to never take her eye off the mirror. She needed to be perfect, needed to finally be chosen for a lead role in the next ballet.

It was normally uncommon for first-year ballerinas to get lead roles in Clarice. Somehow, though, Joseph got his role before even auditioning. The others, who had worked so desperately hard to gain a role, received nothing. Most considered it humiliating, evidence that one needed to be impossibly talented to achieve what Joseph had. Cordelia took it as a learning curve, an experience. She would beat Joseph, she was sure of it. It would just take time.

AI GENERATED STORY

Show 

AI ANALYSIS

Show info 

AI analysis in progress. Results will be available soon.

Average score: 4.0/5.0



Add comment



api_test

2/10



Community



Community



Interfejs 15. Widok opowiadania napisanego przez innego użytkownika



← Write story

Write a short story in the given genre. Your short story should include the words listed in the prompt field.

Genre: horror

Prompt: driving

Title

Content

Publish

Your stories

...

...

...

Your stories

...

...

...

Interfejs 16 – 17. Widok wyboru rodzaju nowego wyzwania oraz pisania nowego opowiadania dla wylosowanego wyzwania

Human vs AI? - minigame

A pair of stories will be presented to you - they will both have the same genre and prompt, but one will have been written by another user and the other will have been artificially generated. The goal of the game is to guess which story scored higher in the AI analysis module. You get a point for each correct guess.

Good luck!

Leaderboard

Place	User	Score
1	wunsz	4.0
2	test	3.0
3	test_user	2.0
4	bogusia	1.0
5	Aga	1.0

Choose a genre to play:

horror

adventure

comedy

crime and mystery

fantasy

historical

romance

science fiction

X Exit minigame

Next pair in this genre →

Swipe left and right to read the other story in this pair.

Genre: horror

Prompt: certain, general, send

Written by: test_user

For the final ceremony, the commune leaders explain that the commune must offer nine human sacrifices to purge it of its evil. The first four victims are outsiders lured to them by Pelle and Ingemar, while the next four victims must be from the commune. As May Queen, Dani must choose either Christian or a commune member to be the final sacrificial victim. She chooses Christian, who is stuffed into a disemboweled brown bear's body and placed in a triangular wooden temple alongside other sacrifices. The commune members to be sacrificed are given drugs and told it will prevent them from feeling fear or pain, but Christian is not and remains unable to move. The structure is set on fire, and the commune members mimic the screams and wails of those being burned alive. Dani initially sobs in horror and grief, but then she gradually begins to smile.

Which story do you think scored higher in the AI analysis module?

AI (CBART)

Human



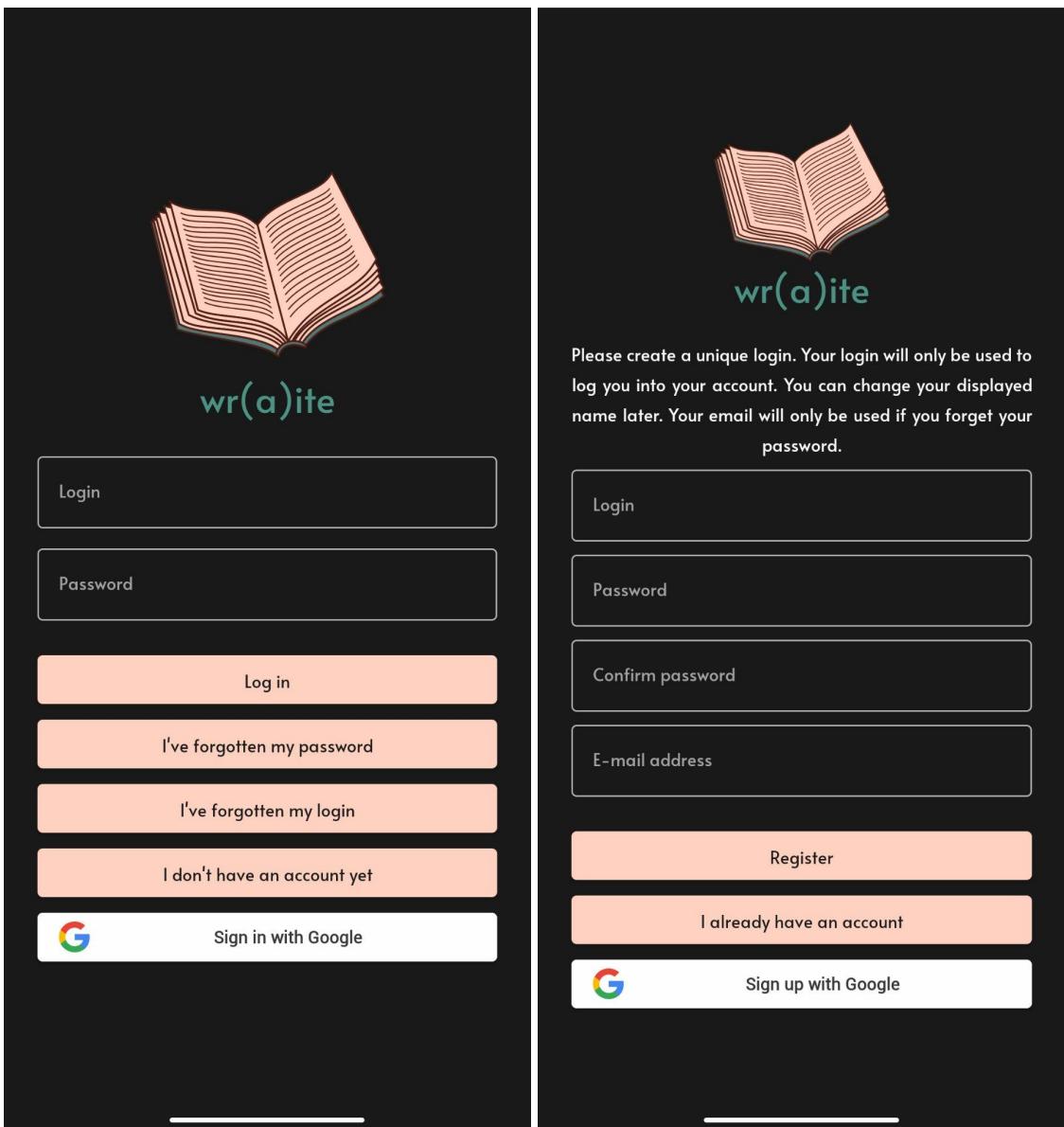
Minigame



Minigame



Interfejs 18 – 19. Widok minigry, w której użytkownik próbuje zgadnąć, które z pary opowiadań napisanych pod to samo wyzwanie (jedno było napisane przez człowieka, a drugie wygenerowane przez model generowania tekstu) otrzymało lepszy wynik



Interfejs 20 – 21. Ekran logowania i rejestracji w stylu ciemnym

The image displays two side-by-side screenshots of a mobile application's main feed screen, likely a social media or story-sharing platform.

Left Screenshot:

- Title:** Out of the Blue
- Author:** test_user
- Text Preview:** Liv was tired. Very, very tired. Another day on the lines and her eyes were aching like sagging balloons, ready to burst from the strain required to just stay employed at that dump. She laced her fingers together and pushed her arms above her head as she came to the end of her daily slog to the bus stop. The movement rewarded her with a series of satisfying pops along her cramped joints.
- Text Continuation:** The bench at the bus stop was hard and damp, mildew curled around the edges. But, her aching legs demanded r...
- Title:** Midsommar
- Author:** test_user
- Text Preview:** For the final ceremony, the commune leaders explain that the commune must offer nine human sacrifices to purge it of its evil. The first four victims are outsiders lured to them by Pelle and Ingemar, while the next four victims must be from the commune. As May Queen, Dani must choose either Christian or a commune member to be the final sacrificial victim. She chooses Christian, who is stuffed into a disemboweled brown bear's body and placed in a triangular wooden temple alongside other sacrifice...

Right Screenshot:

- Title:** The Day from Hell
- Author:** api_test
- Text Preview:** I actually used to work as a cashier and the following scenarios actually happened, along with many others. These two customers were so...eccentric that they are hard to forget. I remember working long hours and trying to not be rude to some people who acted in really bizarre ways. One lady asked if there was anyone in my life who loved me. The nerve...I know. I politely responded that there was. She donated to the charity we were fundraising for soon after.
- Text Continuation:** Another customer told me that my chak...
- Title:** Child of fog
- Author:** Aga
- Text Preview:** It was the first day of summer, the start of holidays. Sun was high in the sky when they came. Chemical cloud covered the world.
- Text Continuation:** No one was save. Few were brave enough to go out of hiding and fight the monsters, that stared coming out front the fog. In that time I made a mistake. I took under my care a stray child i found on the streets. Child that, without my knowledge, was going out every night to run with the wolves.
- Text Continuation:** It was months before I found out. Kid that I treated as mine was changed b...
- Title:** There are No Saints Here
- Author:** test
- Text Preview:** Cordelia had yet to figure out when she realised she first

Interfejs 22 – 23. Widok zakładki głównej (lista opowiadań użytkownika) oraz zakładki społeczności (lista opowiadań innych użytkowników) w stylu ciemnym

Human vs AI? - minigame

A pair of stories will be presented to you – they will both have the same genre and prompt, but one will have been written by another user and the other will have been artificially generated. The goal of the game is to guess which story scored higher in the AI analysis module. You get a point for each correct guess.

Good luck!

Leaderboard

Place	User	Score
1	wunsz	4.0
2	test	3.0
3	test_user	2.0
4	Aga	1.0
5	bogusia	1.0

Choose a genre to play:

- horror
- adventure
- comedy
- crime and mystery
- fantasy
- historical
- romance
- science fiction

test_user

- Favourites
- Statistics
- Change profile picture
- Change name
- Change password
- Change e-mail address
- About wr(a)ite
- Help
- Log out

Interfejs 24 – 25. Widok zakładki minigry i zakładki profilu użytkownika w stylu ciemnym

6.7. Badania UX na grupie docelowej

Docelową grupą użytkowników aplikacji są osoby między 20 – 29 rokiem życia – osoby, które statystycznie najwięcej korzystają z mediów społecznościowych [24]. Aplikacja nie jest medium społecznościowym, ale zawiera element społeczności i kontaktu z innymi użytkownikami.

Dodatkowo, do korzystania z aplikacji wymagana jest znajomość języka angielskiego. Prawdopodobne jest więc, że użytkownicy aplikacji pochodzący z krajów nieanglojęzycznych będą średnio starsi niż użytkownicy aplikacji pochodzący z krajów anglojęzycznych ze względu na konieczność poznania języka.

Przeprowadzono badania “user experience” na grupie docelowej. Zebrano opinie dotyczące wyglądu interfejsu aplikacji (zarówno w stylu jasnym, jak i ciemnym), przejrzystości interfejsu aplikacji oraz czy użycie aplikacji było dla użytkowników skomplikowane, czy proste i zrozumiałe. Uzyskano odpowiedzi od 15 osób respondenckich z grupy docelowej.

Jak oceniasz estetykę aplikacji w stylu jasnym? *

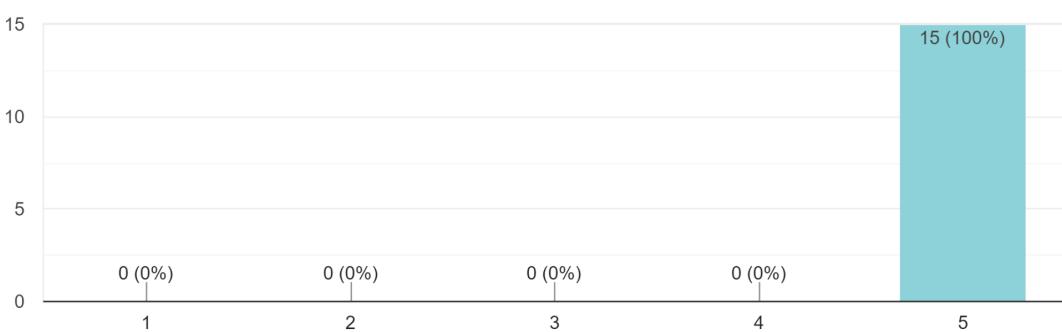
1 2 3 4 5

całkowicie nieestetyczna bardzo estetyczna

Pytanie 1. Pytanie dotyczące estetyki skórki aplikacji w stylu jasnym

Jak oceniasz estetykę aplikacji w stylu jasnym?

15 odpowiedzi



Wykres 1. Odpowiedzi na pytanie 1 – 100% osób respondenckich oceniło estetykę aplikacji w stylu jasnym jako bardzo estetyczną (odpowiedź 5)

Jak oceniasz estetykę aplikacji w stylu ciemnym? *

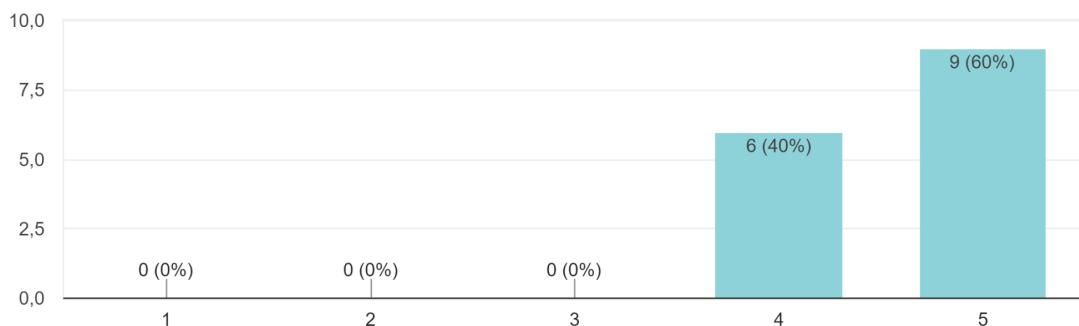
1 2 3 4 5

całkowicie nieestetyczna bardzo estetyczna

Pytanie 2. Pytanie dotyczące estetyki skórki aplikacji w stylu ciemnym

Jak oceniasz estetykę aplikacji w stylu ciemnym?

15 odpowiedzi



Wykres 2. Odpowiedzi na pytanie 2 – 60% osób respondentek oceniło estetykę aplikacji w stylu jasnym jako bardzo estetyczną (odpowiedź 5), a pozostałe 40% jako estetyczną (odpowiedź 4)

Jak oceniasz przejrzystość interfejsu i prostotę obsługi aplikacji? *

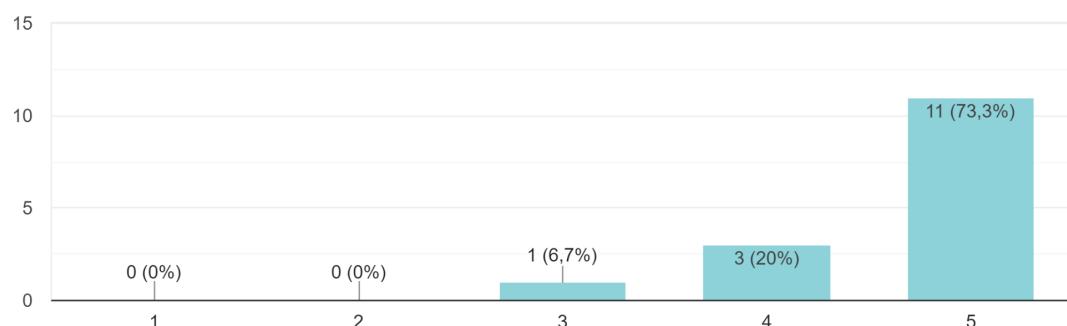
1 2 3 4 5

skomplikowany,
niezrozumiały prosty, całkowicie zrozumiały

Pytanie 3. Pytanie dotyczące przejrzystości interfejsu aplikacji i prostoty obsługi aplikacji

Jak oceniasz przejrzystość interfejsu i prostotę obsługi aplikacji?

15 odpowiedzi



Wykres 3. Odpowiedzi na pytanie 3 – 73,3% osób respondentek oceniło przejrzystość interfejsu i prostotę obsługi aplikacji jako bardzo proste i całkowicie zrozumiałe (odpowiedź 5), 20% jako proste i zrozumiałe (odpowiedź 4) i tylko jedna osoba jako średnio proste i średnio zrozumiałe (odpowiedź 3)

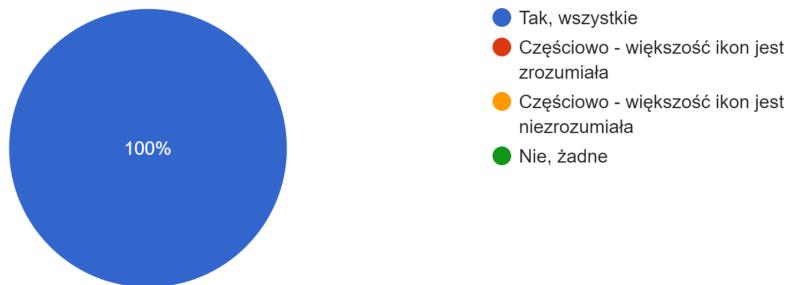
Czy ikony są dla Ciebie zrozumiałe? *

- Tak, wszystkie
- Częściowo - większość ikon jest zrozumiała
- Częściowo - większość ikon jest niezrozumiała
- Nie, żadne

Pytanie 4. Pytanie dotyczące intuicyjności wykorzystanych ikon

Czy ikony są dla Ciebie zrozumiałe?

15 odpowiedzi



Wykres 4. Odpowiedzi na pytanie 4 – 100% osób respondenckich oceniło ikony wykorzystywane w aplikacji jako całkowicie zrozumiałe

Czy rozumiesz do czego służą poszczególne zakładki? *

- Tak, wszystkie
- Częściowo - rozumiem do czego służy większość zakładek
- Częściowo - rozumiem do czego służy mniejsza część zakładek
- Nie, żadne

Pytanie 5. Pytanie dotyczące zrozumienia zakładek

Czy rozumiesz do czego służą poszczególne zakładki?

15 odpowiedzi



Wykres 5. Odpowiedzi na pytanie 5 – 93,3% osób respondenckich oceniło wszystkie zakładki jako całkowicie zrozumiałe i tylko jedna osoba odpowiedziała, że rozumie do czego służy większość zakładek

Czy wyświetlony tutorial rozwija Twoje wątpliwości co do użycia aplikacji? *

- Tak, wszystkie
- Częściowo - rozwija większość moich wątpliwości
- Częściowo - rozwija mniejszą część moich wątpliwości
- Nie, żadne

Pytanie 6. Pytanie dotyczące samouczka wyświetlanego przy pierwszym logowaniu

Czy wyświetlony tutorial rozwija Twoje wątpliwości co do użycia aplikacji?

15 odpowiedzi



Wykres 6. Odpowiedzi na pytanie 6 – 100% osób respondenckich odpowiedziało, że wyświetlony przy pierwszym logowaniu samouczek rozwija wszystkie ich wątpliwości co do użycia aplikacji

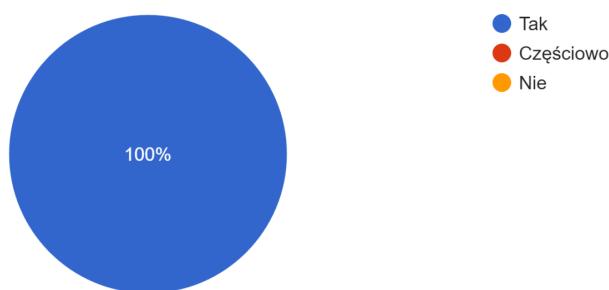
Czy masz poczucie że potrafisz korzystać z aplikacji? *

- Tak
- Częściowo
- Nie

Pytanie 7. Pytanie dotyczące ogólnego poczucia sprawności w korzystaniu z aplikacji

Czy masz poczucie że potrafisz korzystać z aplikacji?

15 odpowiedzi



Wykres 7. Odpowiedzi na pytanie 7 – 100% osób respondenckich odpowiedziało, że czuje, że potrafi korzystać z aplikacji

Jakie dostrzegasz wady i zalety interfejsu aplikacji? Co chciał_byś zmienić?

Twoja odpowiedź

Pytanie 8. Pytanie nieobowiązkowe dotyczące indywidualnej opinii osoby respondenckiej na temat interfejsu aplikacji

Na pytanie 8. odpowiedziało 8 osób respondenckich. Ocenili one interfejs aplikacji jako “bardzo profesjonalny”, “przejrzysty”, “intuicyjny” i “nowoczesny”. Kolory interfejsu w stylu jasnym bardzo podobały się dwóm osobom respondenckim. Jedna osoba respondencka jako zaletę aplikacji wskazała opcję włączenia stylu ciemnego. Jedna osoba respondencka jako zaletę aplikacji wskazała samouczek wyświetlany przy pierwszym logowaniu i oceniła, że jest on przydatny dla nowych użytkowników.

Zgodnie z sugestiami użytkowników z grupy docelowej wprowadzono następujące modyfikacje – do tutorialu wyświetlanego przy pierwszym logowaniu dopisano wytlumaczenie nazwy aplikacji, zwiększeno odległość pomiędzy zdjęciem i nazwą użytkownika a opcjami w zakładce profilu, zwiększeno odległość między liniami, a także zmodyfikowano kolory w stylu ciemnym, który użytkownikom mniej się podobał.

Zgodnie z sugestiami prowadzącego zwiększoną czcionką, wyjustowano tekst oraz dodano opisy przy ikonach, samouczek wyświetlany przy pierwszym logowaniu oraz wizualne oddzielenie opowiadań na liście opowiadań, a także na ekranie indywidualnego opowiadania.

6.8. Wzorce projektowe

Twórczy wzorzec projektowy Builder:

- Do utworzenia obiektu klasy nie jest wykorzystywany konstruktor. W przeciwieństwie do konstruktora wewnętrzna klasa Builder pozwala na utworzenie instancji klasy bez podania wartości dla wszystkich pól.
- Przykładem zastosowania wzorca projektowego w projekcie jest tworzenie okien dialogowych do komunikacji z użytkownikiem – AlertDialog. Dzięki użyciu wzorca można wywołać tylko istotne w danym kontekście metody. W ten sposób utworzono okna dialogowe z jednym przyciskiem i z dwoma przyciskami.

```
val dialog: AlertDialog = with(builder) { this: AlertDialog.Builder  
    setCustomTitle(dialogTitle)  
    setPositiveButton("Yes") { _, _ ->  
        findNavController().navigate(R.id.action_writeFragment_to_navigation_home)  
    }  
    setNegativeButton("No") { _, _ -> }  
    setView(dialogLayout)  
    create() ^with  
}  
  
dialog.setCanceledOnTouchOutside(false)  
dialog.show()
```

Kod 1. Wykorzystanie wzorca w kodzie – utworzenie okna dialogowego do potwierdzenia wyjścia z pisania opowiadania

Twórczy wzorzec projektowy wstrzykiwania zależności:

- Dla obiektów klas zależnych od innych klas (jednym z pól klasy jest obiekt innej klasy) nie należy tworzyć obiektów klas wewnątrz klasy zależnej. Nie jest wtedy możliwe utworzenie zależności kilku obiektów od jednego obiektu (każdy obiekt klasy zależnej posiada własną instancję klasy, od której zależy).
- Przykładem zastosowania wzorca projektowego w projekcie jest zależność obiektów StoryListItemAdapter od obiektów implementujących interfejs OnStorySelectedListener. Obiekty implementujące interfejs nie są tworzone wewnątrz klasy StoryListItemAdapter, tylko przekazywane jako argument konstruktora.

```
class StoryListItemAdapter(private val onStorySelectedListener: OnStorySelectedListener? = null) :  
    ListAdapter<Story, StoryListAdapter.StoryListViewHolder>(StoryItemDiff()) {
```

Kod 2. Wykorzystanie wzorca w kodzie – konstruktor klasy StoryListItemAdapter

Behawioralny wzorzec projektowy Obserwator:

- Pewne obiekty powinny być zależne od zmian w innym obiekcie, na przykład zmiany wartości atrybutów obiektu lub wystąpienia jakiegoś zdarzenia. Obserwatorzy (inaczej słuchacze) powiadomiani są o zmianie stanu obiektu.
- Przykładem zastosowania wzorca projektowego w projekcie są słuchacze zdarzeń, na przykład kliknięcia przycisku. Słuchacz powiadomiany jest o zmianie stanu obiektu, którego jest słuchaczem. Zmiana stanu obiektu występuje w wyniku zdarzenia, najczęściej pochodzącego od użytkownika.
- Kolejnym przykładem zastosowania wzorca projektowego w projekcie są obserwatorzy obiektów typu generycznego MutableLiveData. Obiekty są wykorzystywane aby przechowywać dane pobierane z bazy danych przez API. Obserwatorzy powiadomiani są o zmianie stanu obiektu MutableLiveData, dzięki czemu po otrzymaniu odpowiedzi z API można wyświetlić otrzymane dane na ekranie.

```
binding.fragmentWritePublishButton.setOnClickListener { it: View!  
    if (validate()) {  
        writeViewModel.publishStory(binding.fragmentWriteInputTitle.text.toString(),  
            binding.fragmentWriteInputContent.text.toString(), ::challengeCallback)  
    }  
}
```

Kod 3. Wykorzystanie wzorca w kodzie – słuchacz nasłuchuje naciśnięcia przycisku “Publish”

```
statsViewModel.stats.observe(viewLifecycleOwner) { changedStats ->  
    if (changedStats.stories != -10) {  
        if (changedStats.stories != -1) {  
            binding.fragmentStatsStories.text = changedStats.stories.toString()  
            binding.fragmentStatsComments.text = changedStats.comments.toString()  
            binding.fragmentStatsAvgWords.text = round(changedStats.avgWords).toString()  
            binding.fragmentStatsWordsCount.text = changedStats.wordsCount.toString()  
            binding.fragmentStatsStreak.text =  
                if (changedStats.streak == 1)  
                    "{changedStats.streak.toString()} day"  
                else  
                    "{changedStats.streak.toString()} days"  
            binding.fragmentStatsMinigame.text = changedStats.minigameScore.toString()  
        }  
    } else {  
        Toast  
            .makeText(  
                requireContext(),  
                "Cannot connect to server - please check your internet c...",  
                Toast.LENGTH_LONG)  
            .show()  
    }  
}
```

Kod 4. Wykorzystanie wzorca w kodzie – obserwator nasłuchuje zmiany stanu obiektu MutableLiveData<Statistics> i reaguje na odpowiedzi z API

Wzorzec projektowy DTO:

- Obiekty DTO to obiekty służące do przenoszenia danych między serwerem a aplikacją w celu zmniejszenia liczby wywołań metod. Głównym celem zastosowania obiektów DTO jest zgrupowanie kilku parametrów w jeden obiekt, dzięki czemu można przesłać je do serwera jednym wywołaniem.
- Obiekty DTO wykorzystane są w projekcie do definicji typów przyjmowanych i zwracanych przez API. Są również wykorzystywane z uwagi na zastosowanie klienta REST Retrofit [26] do komunikacji z API.

```
data class UserDTO(  
    @SerializedName(value: "ref") var ref: Int? = null,  
    @SerializedName(value: "login") var login: String? = null,  
    @SerializedName(value: "password") var password: String? = null,  
    @SerializedName(value: "name") var name: String? = null,  
    @SerializedName(value: "profile_pic") var profilePicture: Int? = null,  
    @SerializedName(value: "reg_date") var regDate: String? = null,  
    @SerializedName(value: "user_type") var userType: Int? = null,  
    @SerializedName(value: "email") var email: String? = null  
)
```

Kod 5. Wykorzystanie wzorca w kodzie – definicja typu UserDTO

Wzorzec projektowy Repozytorium:

- Repozytorium wprowadza warstwę abstrakcji nad warstwą dostępu do danych. Repozytorium pozwala na uniezależnienie aplikacji od źródła danych, dzięki czemu możliwa jest zmiana źródła danych bez konieczności zmiany kodu aplikacji.
- Repozytoria wykorzystywane są w projekcie do nawiązania połączenia z API. Wykorzystują obiekty DTO jako typy przyjmowane i zwracane przez API.

```
interface CommentRepository {  
  
    @GET(" /getStoryComments/{storyREF}")  
    fun getStoryComments(@Path("storyREF") storyREF: Int?): Call<List<CommentDTO>>?  
  
    @POST(" /publishComment")  
    fun publishComment(@Body comment: PublishCommentDTO): Call<RefDTO>?  
  
    @GET(" /removeComment/{commentREF}")  
    fun removeComment(@Path("commentREF") commentREF: Int?): Call<MessageDTO>?  
  
    companion object {  
        var BASE_URL = "https://zpi-aaab.herokuapp.com/"  
  
        fun create(apiInterface: Class<CommentRepository>): CommentRepository {  
            val retrofit = Retrofit.Builder()  
                .addConverterFactory(GsonConverterFactory.create())  
                .baseUrl(BASE_URL)  
                .build()  
            return retrofit.create(apiInterface)  
        }  
    }  
}
```

Kod 6. Wykorzystanie wzorca w kodzie – repozytorium dla komentarzy

Wzorzec projektowy Konwerter:

- Konwerter służy do udostępnienia generycznej metody konwersji pomiędzy odpowiadającymi typami. Jest najczęściej wykorzystywany razem ze wzorcem DTO, ponieważ obiekty DTO muszą być mapowane na obiekty encji modelu i odwrotnie.
- Konwerter wykorzystywany jest w projekcie do tłumaczenia obiektów encji modelu na obiekty DTO i odwrotnie. Wykorzystywany jest w repozytoriach.

```
class Converter {  
  
    companion object {  
  
        @JvmStatic  
        fun convertToStoryDTO(story: Story): PublishStoryDTO {  
            return PublishStoryDTO(  
                story.title,  
                story.content,  
                story.regDate  
                    .toInstant()  
                    .atZone(ZoneId.systemDefault())  
                    .toLocalDateTime()  
                    .format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS")),  
                story_fkUser, story.prompt.ref  
            )  
        }  
    }  
}
```

Kod 7. Wykorzystanie wzorca w kodzie – klasa konwertująca między klasami DTO a klasami modelu z metodą konwertującą klasę Story do klasy StoryDTO

7. Implementacja

7.1. Aplikacja Android

Opracowywana aplikacja napisana jest w języku Kotlin z elementami XML. Aplikacje natywne na system Android najczęściej napisane są w języku Java lub Kotlin, a XML jest w nich wykorzystywany do definicji zasobów aplikacji, czyli między innymi wykorzystanych ikon i obrazów, tekstów, wymiarów, kolorów, stylów aplikacji oraz wyglądu poszczególnych ekranów. W aplikacji wykorzystano takie bardziej zaawansowane elementy jak: Recycler View z odpowiednim adapterem, View Pager z odpowiednim adapterem, Search View, Navigation Graph oraz Bottom Navigation View. Do pobierania danych od użytkownika wykorzystano widoki TextInput Layout oraz TextInput Edit Text z pakietu Google Material [25].

Do asynchronicznej komunikacji z API wykorzystano klasę generyczną MutableLiveData i mechanizm callbacków. Dzięki ich zastosowaniu możliwe jest wysłanie zapytania do API, a następnie reakcja na otrzymanie odpowiedzi po jakimś czasie. Klasa generyczna MutableLiveData przechowuje pewną wartość i pozwala reagować na jej zmiany. Funkcja callback (wywołanie zwrotne) jest przekazywana jako argument metody i może być wywołana wewnątrz tej metody, dzięki czemu możliwa jest reakcja na zdarzenie, gdy ono nastąpi. Nie można w tym celu zastosować wyjątków, ponieważ odpowiedź z API nie przychodzi od razu i trzeba na nią zareagować w momencie jej otrzymania – wykorzystanie w tym celu wyjątków kończy się występowaniem nie przechwyconego wyjątku i błędem aplikacji. Obiekty MutableLiveData są przechowywane i obsługiwane w klasach z warstwy ViewModel. Reakcja na zmianę danych przechowywanych w obiekcie MutableLiveData oraz wykorzystanie wywołań zwrotnych zaprezentowano w kodzie 8 – 10.

```

    readStoryViewModel.getAverageScore(storyRef!!)

    readStoryViewModel.avgScore.observe(viewLifecycleOwner) { scoreChanged ->
        when (scoreChanged) {
            -10.0f -> binding.fragmentReadAvgScore.text = ""
            -1.0f -> binding.fragmentReadAvgScore.text = "Nobody has rated this story yet"
            -100.0f ->
                Toast.makeText(requireContext(),
                    "Cannot connect to server - please check your internet c...",
                    Toast.LENGTH_LONG)
                .show()
            else -> {
                binding.apply { this: FragmentReadStoryBinding
                    fragmentReadAvgScore.text = "Average score: ${scoreChanged.toString()}/5.0"
                    fragmentReadRatingBar.rating = scoreChanged
                }
            }
        }
    }
}

```

Kod 8. Pobranie z API średniej oceny opowiadania i reakcja na wartość otrzymaną z API razem z obsługą błędu – Score zainicjalizowany jako -10.0f, -1.0f jest wartością zwracaną przez API jeżeli jeszcze nie oceniono opowiadania, -100.0f oznacza błąd połączenia z API

```

fun getAverageScore(storyRef: Int) {
    scoreService.getAverageScore(storyRef, ::setAvgScore)
}

private fun setAvgScore(value: Float) {
    _avgScore.value = value
}

```

Kod 9. Wywoływana metoda w ReadStoryViewModel (obiekt readStoryViewModel) oraz wywoływany callback

```

fun getAverageScore(storyREF: Int, callback: (score: Float) -> Unit) {
    apiService.getAverageScore(storyREF)?.enqueue(object : Callback<AverageDTO?> {

        override fun onResponse(call: Call<AverageDTO?>, response: Response<AverageDTO?>) {
            Log.i("Get average score succeeded", msg: "")
            if (response.body() != null)
                callback.invoke(response.body()!!.averageScore!!)
            else
                callback.invoke(score: -100.0f)
        }

        override fun onFailure(call: Call<AverageDTO?>, t: Throwable) {
            Log.i("Get score failed", msg: "")
            callback.invoke(score: -100.0f)
        }
    })
}

```

Kod 10. Wywoywana metoda w ScoreService (obiekt scoreService)

Do komunikacji z API wykorzystano klienta REST Retrofit [26]. Jest on jednym z najpopularniejszych rozwiązań wykorzystywanych w branży w obszarze aplikacji natywnych na system Android. Retrofit jest klientem HTTP zorientowanym na typowanie obiektów zapytań – zdefiniowane są typy danych otrzymywane z API i

wysyłane do API. Retrofit wykorzystuje do obsługi żądań HTTP bibliotekę OkHttp oraz wymaga zastosowania konwertera Gson [27] do serializacji.

Do utworzenia klienta Retrofit wykorzystywany jest interfejs definiujący możliwe operacje na danych. Retrofit wewnątrz wykorzystuje bibliotekę OkHttp do obsługi żądań HTTP, ale w kodzie klienta Retrofit używa się tak samo jak obiektów i ich metod. Retrofit przyjmuje interfejs jako argument metody create(...), tworzącej obiekt klienta. Kod 11 jest definicją jednego interfejsów Retrofit wykorzystywanego w aplikacji.

Interfejs Retrofit wykorzystuje obiekty warstwy transferu danych. Pozwala to na grupowanie wielu parametrów w jeden obiekt i wysłanie jednego zapytania, a także uniezależnia kod od reprezentacji danych przyjmowanych i zwracanych przez API. Obiekty warstwy transferu danych są parametrami przyjmowanymi i zwracanymi przez metody interfejsu Retrofit i istnieją niezależnie od encji modelu. Ich zastosowanie pozwala na określenie nazw pól przy serializacji obiektu, dzięki czemu możliwe jest stosowanie odpowiedniej konwencji nazewnictwa. Do konwersji pomiędzy obiektami warstwy transferu danych i obiektami encji modelu wykorzystano wzorzec projektowy Konwerter. Kod 12 jest definicją jednej z klas warstwy transferu danych wykorzystywanej w aplikacji.

Metody interfejsów wywoływanie są w obiektach klas z przyrostkiem Service. Metoda enqueue(...) służy do asynchronicznego wysyłania żądań HTTP i pozwala zdefiniować własną reakcję na sukces odpowiedź i brak odpowiedzi od API poprzez przesłonięcie odpowiednio metod onResponse(...) i onFailure(...). Obiekty klas z przyrostkiem Service są zaś wykorzystywane w klasach z warstwy ViewModel. Dane otrzymane z API często przechowywane są w obiektach MutableLiveData, aby można było reagować na otrzymanie danych z API w komunikacji asynchronicznej. W kodzie 13 zaprezentowana jest definicja jednej z klas z przyrostkiem Service, a w kodzie 14 zaprezentowane jest wykorzystanie obiektu jednej z klas z przyrostkiem Service.

```
interface CommentRepository {  
  
    @GET("value: "/getStoryComments/{storyREF}")  
    fun getStoryComments(@Path("value: "storyREF")storyREF: Int?): Call<List<CommentDTO>>?  
  
    @POST("value: "/publishComment")  
    fun publishComment(@Body comment: PublishCommentDTO): Call<RefDTO>?  
  
    @GET("value: "/removeComment/{commentREF}")  
    fun removeComment(@Path("value: "commentREF")commentREF: Int?): Call<MessageDTO>?  
  
    companion object {  
        var BASE_URL = "https://zpi-aaab.herokuapp.com/"  
  
        fun create(apiInterface: Class<CommentRepository>): CommentRepository {  
            val retrofit = Retrofit.Builder()  
                .addConverterFactory(GsonConverterFactory.create())  
                .baseUrl(BASE_URL)  
                .build()  
            return retrofit.create(apiInterface)  
        }  
    }  
}
```

Kod 11. Interfejs Retrofit będący najniższą warstwą obsługującą komunikację z API

```

data class CommentDTO(
    @SerializedName("ref") var commentRef: Int? = null,
    @SerializedName("content") var content: String? = null,
    @SerializedName("reg_date") var regdate: String? = null,
    @SerializedName("fk_user") var fkUser: Int? = null,
    @SerializedName("name") var username: String? = null,
    @SerializedName("profile_pic") var profilePicture: Int? = null
)

```

Kod 12. Klasa warstwy transferu danych wykorzystywana w interfejsie Retrofit
CommentRepository

```

class CommentService {
    private var apiService: CommentRepository =
        CommentRepository.create(CommentRepository::class.java)

    fun getStoryComments(storyREF: Int, callback: (comments: List<Comment>) -> Unit) {
        apiService.getStoryComments(storyREF)?.enqueue(object : Callback<List<CommentDTO>?> {
            override fun onResponse(call: Call<List<CommentDTO>?>,
                                  response: Response<List<CommentDTO>?>) {
                Log.i( tag: "Get story comments called for story", storyREF.toString())
                if (response.body() != null) {
                    val comments: MutableList<Comment> =
                        Converter.convertToCommentList(response.body())
                    Log.i( tag: "Comments service returned", response.body().toString())
                    callback.invoke(comments)
                } else {
                    callback.invoke(
                        listOf(Comment( REF: -1, content: "", LocalDateTime.now(), username: "",
                                      profilePicture: 0, fkUser: 0, fkStory: 0))
                    )
                }
            }

            override fun onFailure(call: Call<List<CommentDTO>?>, t: Throwable) {
                callback.invoke(
                    listOf(Comment( REF: -1, content: "", LocalDateTime.now(), username: "",
                                  profilePicture: 0, fkUser: 0, fkStory: 0))
                )
            }
        })
    }
}

```

Kod 13. Klasa z przyrostkiem Service wykorzystująca interfejs Retrofit
CommentRepository

```

fun getComments() {
    commentService.getStoryComments(storyREF!!, ::getComments)
}

private fun getComments(comments: List<Comment>) {
    _commentList.value = comments
}

```

Kod 14. Wywołanie metody z klasy serwisu w klasie warstwy ViewModel
ReadStoryViewModel

Hasła użytkowników aplikacji są chronione kryptograficzną funkcją skrótu MD5. Jest to funkcja jednokierunkowa, co oznacza, że łatwo ją obliczyć z wartości wejściowej, ale trudno obliczyć wartość wejściową na podstawie jej skrótu. W rzeczywistości funkcja skrótu MD5 nie jest bezpieczna (na przykład istnieją bazy danych przechowujące słowa i ich funkcje skrótu), dlatego na hasła użytkowników nałożone są ograniczenia: hasła muszą mieć co najmniej 8 znaków oraz zawierać duże i

małe litery, cyfry i znaki specjalne. W bazie danych przechowywane są skróty haseł użytkowników. Hasła użytkowników plaintext nie są przechowywane aby zapewnić poufność danych użytkowników. Przy logowaniu użytkownika wartość skrótu przechowywana w bazie danych porównywana jest z wartością skrótu wprowadzonego hasła. Algorytm obliczania funkcji skrótu MD5 zaprezentowano w kodzie 15.

```
class MD5 {  
  
    companion object {  
  
        @JvmStatic  
        fun md5(input: String): String {  
            val md = MessageDigest.getInstance(algorithm: "MD5")  
            return BigInteger(signum: 1, md.digest(input.toByteArray()))  
                .toString(radix: 16).padStart(length: 32, padChar: '0')  
        }  
    }  
}
```

Kod 15. Obliczanie kryptograficznej funkcji skrótu MD5 dla ciągu wejściowego

Ponieważ aplikacja przyjmuje dane od użytkownika należy uniemożliwić użytkownikowi wprowadzenie danych niepoprawnych, na przykład pustego hasła lub błędного adresu e-mail. Dodatkowo w ramach walidacji danych wprowadzanych przez użytkownika sprawdzane jest, czy wprowadzony ciąg znaków nie zawiera żadnego słowa ze słownika słów wymagających cenzury (dalej nazywanych słowami zbanowanymi).

Sprawdzane są wszystkie dane wprowadzane przez użytkownika: login, hasło, wyświetlana nazwa, adres e-mail, komentarz, tytuł opowiadania oraz treść opowiadania. Wszystkie wartości muszą mieć co najmniej jeden znak oraz być krótsze niż zdefiniowana dla nich maksymalna długość – dla loginu, wyświetlonej nazwy i hasła jest to 25 znaków, dla adresu e-mail i tytułu opowiadania jest to 50 znaków, dla komentarza jest to 250 znaków i dla opowiadania jest to 3000 znaków. Dla hasła wprowadzono dodatkowo dolne ograniczenie długości – hasło musi być dłuższe niż 8 znaków. Limity długości wprowadzanych danych znajdują się w kodzie 16.

Hasło i adres e-mail walidowane są przy użyciu wyrażeń regularnych – hasło musi zawierać co najmniej jedną małą i dużą literę, cyfrę oraz znak specjalny ze zbioru { . , _ - ! @ # \$ % ^ & * + }, a adres e-mail musi być poprawny. Wszystkie dane wyświetlane w aplikacji (login, wyświetlana nazwa, komentarz, tytuł i treść opowiadania) są sprawdzane pod kątem zawierania słów zbanowanych – nie ma potrzeby sprawdzania haseł ani adresów e-mail użytkowników, ponieważ nie są one widoczne dla innych użytkowników.

Wszystkie metody walidujące znajdują się w klasie Validator, która jest odpowiedzialna za sprawdzanie poprawności danych wprowadzonych przez użytkownika i wyświetlanie stosownych komunikatów w przypadku niepoprawności danych. Przykłady metod walidujących znajdują się w kodzie 17 – 19.

```
private val minPasswordLength = 8  
private val maxLengthShort = 25  
private val maxLengthMedium = 50  
private val maxLengthLong = 250  
private val maxLengthVeryLong = 3000
```

Kod 16. Limity długości wprowadzanych przez użytkownika danych

```

fun ifValidName(name: String): Boolean {
    if (name == "") {
        Toast.makeText(
            context,
            "New name cannot be empty",
            Toast.LENGTH_SHORT)
            .show()
        return false
    }
    if (name.length > maxLengthShort) {
        Toast.makeText(
            context,
            "{New name} cannot be longer than {25} characters",
            Toast.LENGTH_SHORT)
            .show()
        return false
    }
    return checkForBannedWords(name)
}

```

Kod 17. Walidacja wyświetlanej nazwy użytkownika – nie może ona być pusta, dłuższa niż 25 znaków ani zawierać żadnego ze słów zbanowanych

```

fun ifValidEmail(email: String): Boolean {
    if (email == "") {
        Toast.makeText(
            context,
            "New e-mail address cannot be empty",
            Toast.LENGTH_SHORT)
            .show()
        return false
    }
    if (email.length > maxLengthMedium) {
        Toast.makeText(
            context,
            "{New e-mail address} cannot be longer than {50} characters",
            Toast.LENGTH_SHORT)
            .show()
        return false
    }
    if (!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        Toast.makeText(
            context,
            "This e-mail address is invalid",
            Toast.LENGTH_LONG)
            .show()
        return false
    }
    return true
}

```

Kod 18. Walidacja adresu e-mail użytkownika – nie może być on pusty ani dłuższy niż 50 znaków, a także musi być poprawnym adresem e-mail. Ponieważ adres e-mail nie jest widoczny dla innych użytkowników nie jest sprawdzany pod kątem zawierania słów zbanowanych

```

private fun checkForBannedWords(text: String): Boolean {
    var message = ""
    for (word in bannedWords)
        if (word.value in text)
            message += word.censored + ", "
    message = message.trimEnd()
    message = message.trimEnd(...chars: ',', ')
    if (message == "")
        return true
    Toast.makeText(
        context,
        "Please censor the following words: {message}.",
        Toast.LENGTH_LONG)
    .show()
    return false
}

```

Kod 19. Wywoływana funkcja checkForBannedWords(...)

Aby umożliwić użytkownikom logowanie przy użyciu konta Google należy utworzyć dane uwierzytelniające dla aplikacji w serwerze Google OAuth 2.0 w usłudze Google Cloud [28]. Uwierzytelnienie aplikacji w serwerze Google OAuth 2.0 jest konieczne do działania integracji z API Google. Po utworzeniu danych uwierzytelniających należy utworzyć przycisk logowania z Google zgodny z wytycznymi dotyczącymi marki Sign-In Branding Guidelines [21].

Logowanie z Google wykorzystuje mechanizm startActivityForResult(...). Umożliwia on uruchomienie aktywności (może być to aktywność zewnętrznej aplikacji, tak jak w tym przypadku) i reakcję na rezultat otrzymany po jej zamknięciu. Utworzenie klienta GoogleSignIn oraz uruchomienie aktywności z oczekiwaniem na rezultat zaprezentowano w kodzie 20.

Reakcję na rezultat obsługuje się przesłanając metodę onActivityResult(...) aktywności. Nazwę zalogowanego użytkownika można pobrać z klienta GoogleSignIn za pomocą metody getSignedInAccountFromIntent(...) Za pomocą uzyskanych danych logowania można następnie zalogować użytkownika do aplikacji wr(a)ite. Użytkownicy logujący się swoim kontem Google są w bazie danych przechowywani jako użytkownicy z polem UserType o wartości 1. Ich adresy e-mail oraz hasła nie są przechowywane. Obsługę rezultatu aktywności oraz logowanie do aplikacji wr(a)ite za pomocą danych logowania Google zaprezentowano w kodzie 21 – 22.

Przy próbie zalogowania do aplikacji API zwraca numer REF użytkownika, który próbuje się zalogować oraz jego hasło do autentykacji (przy logowaniu z Google hasło nie jest wykorzystywane). Jeżeli użytkownik o danym loginie nie istnieje, API zwraca numer REF -1. W przypadku logowania z Google oznacza to, że osoba próbująca zalogować się swoim kontem Google jeszcze nie logowała się do aplikacji i należy zarejestrować jej konto. Konta użytkowników, którzy pierwszy raz korzystają ze swojego konta Google logując się do aplikacji rejestrowane są automatycznie. W kodzie 22 – 24 zaprezentowano logowanie z Google dla użytkowników, którzy korzystali wcześniej ze swojego konta Google logując się do aplikacji. W kodzie 24 – 27 zaprezentowano logowanie z Google dla użytkowników, którzy jeszcze nie logowali się do aplikacji przy użyciu swojego konta Google.

```

val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken("1074516094425-bc8pg37o0t1dve948n9dkdbin25c2f77.apps.goo...")
    .requestEmail()
    .build()
val mGoogleSignInClient = GoogleSignIn.getClient(requireActivity(), gso)
binding.signInButtonGoogle.setOnClickListener { it: View ->
    val signInIntent = mGoogleSignInClient.signInIntent
    startActivityForResult(signInIntent, requestCode: 100)
}

```

Kod 20. Utworzenie i konfiguracja klienta GoogleSignIn oraz uruchomienie aktywności zewnętrznej aplikacji z oczekiwaniem na rezultat

```

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    val task: Task<GoogleSignInAccount> = GoogleSignIn.getSignedInAccountFromIntent(data)
    handleSignInResult(task)
}

private fun handleSignInResult(completedTask: Task<GoogleSignInAccount>) {
    try {
        val account = completedTask.getResult(ApiException::class.java)
        loginViewModel.loginWithGoogle(account.displayName!!, ::loginCallback)
    } catch (e: ApiException) {
        Log.w(tag: "LOGIN", msg: "signInResult:failed code=" + e.statusCode)
    }
}

```

Kod 21. Obsługa rezultatu uruchomienia aktywności zewnętrznej aplikacji i logowanie do aplikacji wr(a)ite loginem Google

```

fun loginWithGoogle(login: String,
                     callback: (success: Boolean, login: String, ref: Int) -> Unit) {
    userService.loginUserGoogle(login, callback)
}

```

Kod 22. Wywoływana metoda w LoginViewModel (obiekt loginViewModel) – logowanie użytkownika

```

fun loginUserGoogle(login: String,
                     callback: (success: Boolean, login: String, ref: Int) -> Unit) {
    apiService.getUserRef(login)?.enqueue(object : Callback<RefDTO?> {

        override fun onResponse(call: Call<RefDTO?>, response: Response<RefDTO?>) {
            Log.i(tag: "Login attempt response", response.body().toString())
            if (response.body() != null) {
                callback.invoke(success: true, login, response.body()!!._ref!!)
            } else {
                callback.invoke(success: true, login, ref: -1)
            }
        }

        override fun onFailure(call: Call<RefDTO?>, t: Throwable) {
            callback.invoke(success: false, login, ref: -1)
        }
    })
}

```

Kod 23. Wywoywana metoda w UserService (obiekt userService) – logowanie użytkownika

```
private fun loginCallback(success: Boolean, login: String, ref: Int) {
    if (success) {
        if (ref >= 0) {
            val intent = Intent(packageContext, MainActivity::class.java)
            intent.putExtra(name: "userREF", ref)
            intent.putExtra(name: "firstLogin", value: 1)
            this.startActivity(intent)
        } else {
            loginViewModel.registerUserGoogle(login, ::registerGoogleCallback)
        }
    } else {
        Toast.makeText(
            context: this,
            "Cannot connect to server - please check your internet c...",
            Toast.LENGTH_LONG
        ).show()
    }
}
```

Kod 24. Funkcja callback wykorzystywana przy logowaniu z Google. Jeżeli użytkownik wcześniej wykorzystywał do logowania swoje konto Google success ma wartość True, a ref wartość większą od 0 i użytkownik zostaje zalogowany do aplikacji i nie zostaje mu wyświetlony samouczek. Jeżeli użytkownik jeszcze nie wykorzystał do logowania swojego konta Google success ma wartość True, ale ref ma wartość -1 i użytkownik musi zostać zarejestrowany w aplikacji.

```
fun registerUserGoogle(login: String, callback: (success: Boolean, ref: Int) -> Unit) {
    val user = User(
        ref: -1,
        login,
        password: "",
        login,
        profilePicture: 1,
        LocalDateTime.now(),
        userType: 1,
        email: "")
    userService.registerUser(user, callback)
}
```

Kod 25. Wywoływana metoda w LoginViewModel (obiekt loginViewModel) – tworzony jest nowy użytkownik. Jego hasło i adres e-mail są puste – ten użytkownik będzie mógł zalogować się do aplikacji jedynie za pomocą swojego konta Google.

```

fun registerUser(user: User, callback: (success: Boolean, ref: Int) -> Unit) {
    val userDTO = Converter.convertToUserDTO(user, useMD5: true)
    apiService.registerUser(userDTO)?.enqueue(object : Callback<RefDTO?> {

        override fun onResponse(call: Call<RefDTO?>, response: Response<RefDTO?>) {
            Log.i( tag: "Register attempt response", response.body().toString())
            if (response.body() != null) {
                callback.invoke( success: true, response.body()!!._ref!!)
            } else {
                callback.invoke( success: true, ref: -1)
            }
        }

        override fun onFailure(call: Call<RefDTO?>, t: Throwable) {
            Log.i( tag: "Register attempt failed", msg: "")
            callback.invoke( success: false, ref: -1)
        }
    })
}

```

Kod 26. Wywoływana metoda w UserService (obiekt userService) – rejestracja użytkownika

```

private fun registerGoogleCallback(success: Boolean, ref: Int) {
    if (success) {
        if (ref >= 0) {
            val intent = Intent( packageContext: this, MainActivity::class.java)
            intent.putExtra( name: "userREF", ref)
            intent.putExtra( name: "firstLogin", value: 0)
            this.startActivity(intent)
        } else {
            Toast.makeText(
                context: this,
                "This login is already taken - please choose a different...",
                Toast.LENGTH_LONG)
            .show()
        }
    } else {
        Toast.makeText(
            context: this,
            "Cannot connect to server - please check your internet c...",
            Toast.LENGTH_LONG)
        .show()
    }
}

```

Kod 27. Funkcja callback wykorzystywana przy rejestracji konta Google w aplikacji.

Jeżeli udało się zarejestrować użytkownika success ma wartość True, a ref wartość większą od 0 i użytkownik zostaje zalogowany do aplikacji i zostaje mu wyświetlony samouczek. Jeżeli nie udało się zarejestrować użytkownika success ma wartość True, ale ref ma wartość -1 i wyświetlony zostaje stosowny komunikat – oznacza to, że ten login został już wykorzystany przez innego użytkownika.

Adresy e-mail użytkowników są przechowywane aby umożliwić użytkownikom reset hasła. Jeżeli użytkownik zapomnia swojego hasła może nacisnąć przycisk “I've forgotten my password” i zostanie poproszony o wprowadzenie swojego loginu. Na adres e-mail skojarzony z tym kontem zostanie następnie wysłany adres e-mail umożliwiający użytkownikowi wygenerowanie nowego hasła. Aby umożliwić użytkownikom bezpieczny reset swoich haseł konieczne jest wysłanie adresu e-mail przez aplikację w tle. W tym celu wykorzystano API JavaMail [29].

Działanie API JavaMail opiera się na dwóch klasach – GMailSender, który jest implementacją wysłania wiadomości e-mail i wykorzystuje elementy z pakietu javax.mail, oraz JSSEProvider, który definiuje ustawienia bezpiecznej komunikacji przez Internet i jest licencjonowany przez Apache Software Foundation (ASF) zgodnie z licencją 2.0 [30]. Obiekt klasy JSSEProvider jest wykorzystywany jako dostawca w konfiguracji ustawień bezpieczeństwa klasy GMailSender. Wykorzystano klasy GMailSender oraz JSSEProvider udostępnione przez użytkownika Vinayak Bevinakatti na portalu StackOverflow [31]. Kod klas przepisano z języka Java na język Kotlin, a kod klasy GMailSender zmodyfikowano do własnych potrzeb. Klasę JSSEProvider zaprezentowano w kodzie 28, a fragment klasy GMailSender w kodzie 29 – 30.

Aby umożliwić aplikacji logowanie do konta Google i wysłanie wiadomości e-mail z tego konta należy dla konta włączyć weryfikację dwuetapową oraz zdefiniować hasło do aplikacji, które będzie wykorzystywane w aplikacji do logowania do konta Google w celu wysłania wiadomości e-mail. Nie jest ono tożsame z hasłem do konta Google. Ten rodzaj weryfikacji został wprowadzony aby zwiększyć poziom bezpieczeństwa przy logowaniu do Google z innych aplikacji na przykład w celu wysłania wiadomości e-mail.

Obsługę naciśnięcia przycisku “I've forgotten my password”, obsługę okna dialogowego, pobranie adresu e-mail skojarzonego z kontem użytkownika oraz wysłanie wiadomości e-mail na innym wątku zaprezentowano w kodzie 31 – 36. Po naciśnięciu przycisku i wpisaniu poprawnego loginu użytkownik otrzymuje wiadomość e-mail o treści przedstawionej na obrazie 1. Po kliknięciu w link w tej wiadomości e-mail użytkownikowi wyświetlana jest strona zaprezentowana na obrazie 2 i jego hasło do aplikacji zostaje zresetowane.

```
class JSSEProvider : Provider( name: "HarmonyJSSE", version: 1.0, info: "Harmony JSSE Provider") {
    init {
        AccessController.doPrivileged(PrivilegedAction<Void?> {
            put("SSLContext.TLS", "org.apache.harmony.xnet.provider.jsse.SSLContextImpl")
            put("Alg.Alias.SSLContext.TLSv1", "TLS")
            put(
                "KeyManagerFactory.X509",
                "org.apache.harmony.xnet.provider.jsse.KeyManagerFactoryImpl"
            )
            put(
                "TrustManagerFactory.X509",
                "org.apache.harmony.xnet.provider.jsse.TrustManagerFactoryImpl"
            )
            null ^PrivilegedAction
        })
    }
}
```

Kod 28. Klasa JSSEProvider – dostawca Java Secure Socket Extension

```
class GMailSender(private val user: String, private val password: String) : Authenticator() {
    private val mailhost = "smtp.gmail.com"
    private val session: Session
    private val _multipart: Multipart

    companion object {
        init {
            Security.addProvider(JSSEProvider())
        }
    }
}
```

Kod 29. Fragment klasy GMailSender z konfiguracją ustawień bezpieczeństwa

```

@Synchronized
@Throws(Exception::class)
fun sendMail(subject: String?, body: String, sender: String?, recipients: String) {
    val message = MimeMessage(session)
    val handler = DataHandler(ByteArrayDataSource(body.toByteArray(), type = "text/plain"))
    message.sender = InternetAddress(sender)
    message.subject = subject
    message.dataHandler = handler
    if (recipients.indexOf(',') > 0)
        message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(recipients))
    else
        message.setRecipient(Message.RecipientType.TO, InternetAddress(recipients))
    Transport.send(message)
}

```

Kod 30. Metoda klasy GMailSender wysyłająca wiadomość e-mail w tle

```

binding.fragmentLoginForgotPasswordButton.setOnClickListener { it: View! -->
    showForgotPasswordDialog()
}

```

Kod 31. Słuchacz naciśnięcia przycisku “I’ve forgotten my password”

```

private fun showForgotPasswordDialog() {
    val builder = AlertDialog.Builder(requireContext())
    val inflater: LayoutInflater = layoutInflater
    val dialogLayout = inflater.inflate(R.layout.dialog_forgot_password, root: null)
    val dialogTitle = inflater.inflate(R.layout.dialog_title, root: null)
    dialogTitle.findViewById<TextView>(R.id.textView).text = "Forgot password?"

    val login = dialogLayout.findViewById<EditText>(R.id.dialog_forgot_password_input)

    val dialog: AlertDialog = with(builder) { this: AlertDialog.Builder!
        setCustomTitle(dialogTitle)
        setPositiveButton("Confirm") { _, _ -> }
        setNegativeButton("Cancel") { _, _ -> }
        setView(dialogLayout)
        create() ^with
    }

    dialog.setCancelable(false)

    dialog.setOnShowListener { it: DialogInterface! -->
        val confirmButton = dialog.getButton(AlertDialog.BUTTON_POSITIVE)
        confirmButton.setOnClickListener { it: View! -->
            if (login.text.toString() != "") {
                if (validator.isValidLogin(login.text.toString())) {
                    loginViewModel.getUserEmail(login.text.toString(), ::passwordEmailCallback)
                    dialog.dismiss()
                }
            } else {
                Toast.makeText(context, "{Login} cannot be empty", Toast.LENGTH_SHORT).show()
            }
        }
    }
    dialog.show()
}

```

Kod 32. Dialog “Forgot password?” umożliwiający użytkownikowi wpisanie swojego loginu i pobierający z bazy danych adres e-mail skojarzony z tym kontem użytkownika

```

fun getUserEmail(login: String, callback: (success: Boolean, ref: Int, email: String) -> Unit) {
    userService.getUserEmail(login, callback)
}

```

Kod 33. Wywoływana metoda w LoginViewModel (obiekt loginViewModel) – pobranie adresu e-mail skojarzonego z kontem użytkownika

```

fun getUserEmail(login: String, callback: (success: Boolean, ref: Int, email: String) -> Unit) {
    apiService.getUserEmail(login)?.enqueue(object : Callback<GetUserEmailDTO?> {

        override fun onResponse(call: Call<GetUserEmailDTO?>,
                               response: Response<GetUserEmailDTO?>) {
            Log.i( tag: "Forgot password attempt response", response.body().toString())
            if (response.body() != null) {
                if (response.body()!!.ref != -1) {
                    callback.invoke(
                        success: true,
                        response.body()!!.ref!!,
                        response.body()!!.email!!
                    )
                }
                else {
                    callback.invoke( success: false, ref: -1, email: "")
                }
            } else {
                callback.invoke( success: false, ref: 0, email: "")
            }
        }

        override fun onFailure(call: Call<GetUserEmailDTO?>, t: Throwable) {
            Log.i( tag: "Forgot password attempt failed", msg: "")
            callback.invoke( success: false, ref: -1, email: "")
        }
    })
}

```

Kod 34. Wywoływana metoda w UserService (obiekt userService) – pobranie adresu e-mail skojarzonego z kontem użytkownika

```

private fun passwordEmailCallback(success: Boolean, ref: Int, address: String) {
    if (success) {
        sendEmail(
            address,
            "wr(a)ite - reset password request",
            "Hello! You've recently sent a request to generate a ne...")
        Toast.makeText(context, "E-mail sent", Toast.LENGTH_LONG).show()
    } else {
        if (ref == -1)
            Toast.makeText(context, "This user doesn't exist", Toast.LENGTH_LONG).show()
        else
            Toast.makeText(
                context,
                "Cannot connect to server - please check your internet c...",
                Toast.LENGTH_LONG)
            .show()
    }
}

```

Kod 35. Obsługa błędów połączenia z API i w przypadku sukcesu wysłanie wiadomości e-mail

```

private fun sendEmail(address: String, title: String, content: String) {
    Thread {
        try {
            val sender = GMailSender("zpi.aaab.noreply@gmail.com", "cruvhftjigdsmrzh")
            sender.sendMail(title, content, getString(R.string.email_address), address)
        } catch (e: Exception) {
            Log.e(tag, "LoginFragment email error", e.stackTraceToString())
        }
    }.start()
}

```

Kod 36. Wysłanie wiadomości e-mail z adresu zpi.aaab.noreply@gmail.com

Hello!

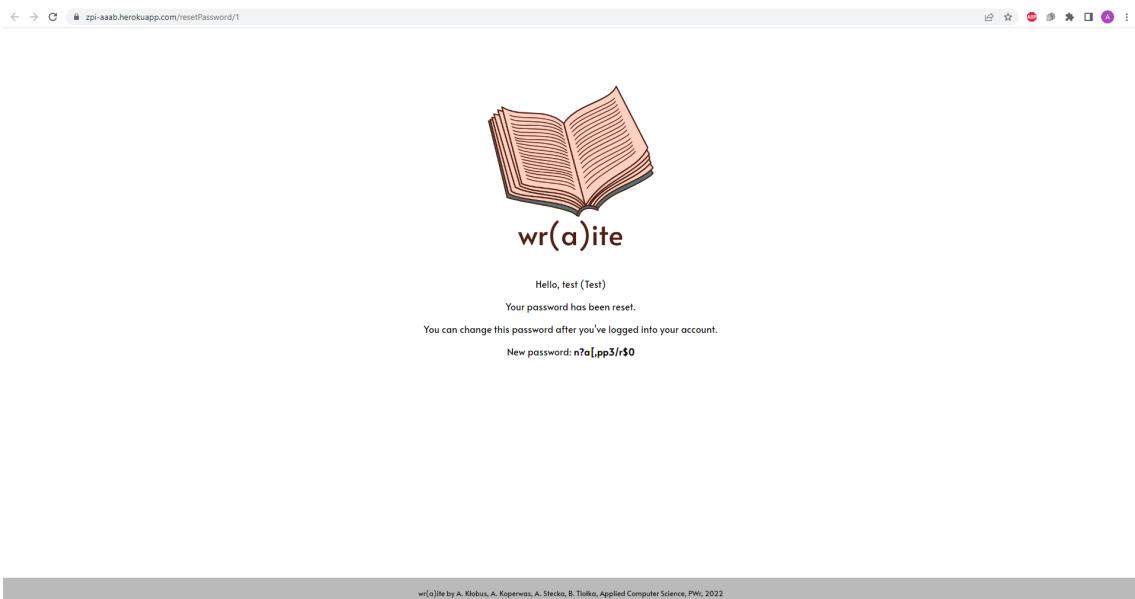
You've recently sent a request to generate a new password for your wr(a)ite account.
If this wasn't you - ignore this message. Someone must have accidentally written your login instead of theirs when requesting to generate a new password. Don't worry, your password has not been reset yet.
You can change your password after you've logged into your account.

To generate a new, random password click this link: <https://zpi-aaab.herokuapp.com/resetPassword/1>

Happy writing! - the wr(a)ite team

Please do not respond to this message.

Obraz 1. Treść wiadomości e-mail wysyłanej do użytkownika jeżeli zapomniał on hasła



Obraz 2. Wygląd strony informującej użytkownika o zresetowaniu jego hasła i prezentującej nowe hasło

Do obsługi wyszukiwania wykorzystywany jest element `SearchView` z biblioteki `android.view`. Za jego pomocą możliwe jest wprowadzanie zapytań do wyszukiwania. Rozpoczęcie wpisywania powoduje wyświetlenie listy proponowanych wyszukiwań, na podstawie wcześniej wprowadzonych. Wprowadzone wartości zapamiętywane są w liście przy pomocy `SharedPreferences` w momencie wyświetlania wyników, jak pokazano w kodzie 37.

Do wyświetlania rezultatów wykorzystywana jest osobna aktywność. Uruchomienie jej następuje w momencie wysłania zapytania dzięki implementacji słuchacza `OnQueryChangeListener`. Dzięki niemu możliwe jest również uzupełnianie listy podpowiedzi o wcześniejsze zapytania zgodne z wprowadzonym dzięki metodzie `onQueryTextChange`. Implementacja prezentowana jest w kodzie 38.

Aby umożliwić wybór podpowiedzi w elemencie wyszukiwania, konieczna była również implementacja słuchacza OnSuggestionListener pozwalającego na zamianę wyszukiwanego zapytania na wybrane, co prezentuje kod 39. Oba słuchacze są klasami wewnętrznymi opakowującą je wspólnej klasy QueryHandler, co pozwoliło na uniknięcie niepotrzebnego powielania kodu.

```
private fun handleIntent(intent: Intent) {
    intent.getStringExtra(SearchManager.QUERY)?.also { query ->
        viewModel.searchForStories(query)
        viewModel.setQuery(query)
        val sharedPref = getSharedPreferences(getString(R.string.prefs_for_all), Context.MODE_PRIVATE);
        val savedQueries: MutableSet<String>? = sharedPref.getStringSet(getString(R.string.saved_search_queries), setOf())
        with(sharedPref.edit()) { this: SharedPreferences.Editor!
            val queries = mutableSetOf<String>()
            if (savedQueries != null) {
                queries.addAll(savedQueries)
            }
            queries.add(query)
            this.putStringSet(getString(R.string.saved_search_queries), queries)
            this.apply()
        }
    }
}
```

Kod 37. Obsługa wyszukiwania wywołująca wyszukania wartości przy pomocy view modelu oraz zapamiętująca wyszukiwanie.

```
inner class OnQueryText : SearchView.OnQueryTextListener {
    override fun onQueryTextSubmit(query: String?): Boolean {
        if(searchActive){...}
        val intent = Intent(activity, SearchActivity::class.java)
        intent.action = Intent.ACTION_SEARCH
        Log.i("SEARCH", query.toString())
        intent.putExtra(SearchManager.QUERY, query.toString())
        startActivity(context, intent, options: null)
        return true
    }

    override fun onQueryTextChange(text: String?): Boolean {
        text?.let { populateAdapter(it) }
        return true
    }
}
```

Kod 38. Implementacja słuchacza OnQueryTextListener uruchamiająca aktywność wyszukiwania i obsługująca mechanizm podpowiedzi.

```

inner class OnQuerySuggestion : SearchView.OnSuggestionListener {
    override fun onSuggestionSelect(p0: Int): Boolean {
        return true
    }

    @SuppressLint( ...value: "Range" )
    override fun onSuggestionClick(position: Int): Boolean {
        val cursor: Cursor = adapter.getItem(position) as Cursor
        val txt: String = cursor.getString(cursor.getColumnIndex(tableName))
        searchView.setQuery(txt, submit: true)
        return true
    }
}

```

Kod 39. Implementacja słuchacza OnSuggestionListener pozwalająca na wybranie podpowiedzi jako tekstu wyszukiwania

7.2. API

Aby uniezależnić najniższą warstwę dostępu do bazy danych od aplikacji mobilnej zdecydowano o napisaniu API. API napisano w języku Python przy użyciu mikro-frameworku Flask, a wdrożono w chmurze Heroku. Dzięki temu aplikacja mobilna nie wykonuje bezpośrednio kwerend na bazie danych i w razie konieczności modyfikacji bazy danych lub operacji na niej nie jest wymagana modyfikacja aplikacji mobilnej, a jedynie API, będącego pośrednikiem między bazą danych a aplikacją mobilną.

API jest odpowiedzialne za wszystkie podstawowe operacje na danych przechowywanych w bazie danych – zwracanie danych do wyświetlenia w aplikacji (na przykład opowiadań użytkownika wyświetlanych w zakładce “Your stories” lub opowiadań innych użytkowników wyświetlanych w zakładce “Community”), usuwanie danych (na przykład opowiadania lub komentarza), dodawanie danych (na przykład opowiadania lub komentarza) oraz modyfikacji danych (na przykład hasła, wyświetlonej nazwy, zdjęcia profilowego). API wywołuje też metody z modułu wykorzystującego elementy sztucznej inteligencji, co jest wykorzystywane na przykład do tego, aby po publikacji opowiadania automatycznie rozpoczęła się jego analiza. Dane dotyczące opowiadania, które ma zostać przeanalizowane są wysyłane jako ciało żądania POST z uwagi na ciągłe trwanie transakcji zapisywania opowiadania w bazie danych. Kod 40 – 43 zawiera definicję przykładowych endpointów API, a kod 44 metodę wywołującą metodę z modułu wykorzystującego elementy sztucznej inteligencji przy publikowaniu opowiadania.

```

@app.route("/getUserStories/<user_ref>")
def get_user_stories(user_ref: str):
    records = _get_user_stories(user_ref)
    return json.dumps(records)

```

Kod 40. Endpoint API zwracający wszystkie opowiadania napisane przez użytkownika

```

def _get_user_stories(user_ref: str):
    query = "select s.ref, s.title, s.content, s.regdate as reg_date, s.fk_user, s.fk_prompt, u.name as username, " \
            "gd.value as genre, trim(trailing ' ' from string_agg(wd.value, ', ')) as words " \
            "from stories s " \
            "inner join users u on u.ref = s.fk_user " \
            "inner join prompts p on p.ref = s.fk_prompt " \
            "inner join genredict gd on gd.ref = p.fk_genre " \
            "inner join promptswords pw on pw.fk_prompt = p.ref " \
            "inner join wordsdict wd on wd.ref = pw.fk_word " \
            "where u.ref = " + user_ref + " and s.generatedfor is null " \
            "group by s.ref, s.title, s.content, u.name, gd.value " \
            "order by s.regdate desc"
    records = []
    with _get_connection() as conn:
        with conn.cursor() as cursor:
            cursor.execute(query)
            attributes = [key[0] for key in cursor.description]
            rows = cursor.fetchall()
            for row in rows:
                records.append(dict(zip(attributes, row)))
    return records

```

Kod 41. Wywoływana metoda `_get_user_stories(...)` – kod wydzielono do osobnej metody, ponieważ jest wykorzystywany w kilku endpointach

```

@app.route("/publishStory", methods=['POST'])
def publish_story():
    story = Story(request.get_json())
    query = "insert into stories (title, content, regdate, fk_user, fk_prompt) " \
            "values ('" + _escape_specials(story.title) + "', '" + _escape_specials(story.content) + "', '" + \
            str(story.reg_date) + "', " + str(story.fk_user) + ", " + str(story.fk_prompt) + ") " \
            "returning ref"
    with _get_connection() as conn:
        with conn.cursor() as cursor:
            cursor.execute(query)
            row = cursor.fetchone()
            start_analyse(str(row[0]), story.content, str(story.fk_prompt))
    return {"ref": row[0]}

```

Kod 42. Endpoint API umożliwiający opublikowanie opowiadania

```

def _escape_specials(text: str):
    return text.replace("'", "''")

```

Kod 43. Wywoływana metoda `_escape_specials(...)` zastępująca znak apostrofu podwójnym apostrofem, dzięki czemu może być on przechowywany w bazie danych jako wartość tekstowa

```

def start_analyse(story_ref: str, story_content: str, fk_prompt: str):
    genre = _get_prompt_genre(fk_prompt)
    prompt_words = _get_prompt_words(fk_prompt)
    url = "https://zpi-ai.herokuapp.com/analyse"
    payload = {
        "story_ref": str(story_ref),
        "story_content": str(story_content),
        "fk_prompt": str(fk_prompt),
        "prompt_genre": str(genre),
        "prompt_words": str(prompt_words)
    }
    headers = {
        "content-type": "application/json"
    }
    print(url)
    try:
        requests.request("POST", url, json=payload, headers=headers)
    except requests.exceptions.ReadTimeout:
        pass

```

Kod 44. Metoda API rozpoczynająca proces analizowania opowiadania przez moduł AI po publikacji opowiadania

7.3. Pozyskanie danych uczących

Do wyuczenia modelu sztucznej inteligencji wykorzystanego do analizy tekstu wykorzystano opowiadania pisane przez użytkowników portalu archiveofourown.org (AO3) [4]. Teksty opublikowane na AO3 posiadają przypisany gatunek – jest to istotne, ponieważ jednym z zadań modułu analizy tekstu jest przyporządkowanie gatunku do opowiadania. Teksty nie są jednak pisane przez profesjonalnych pisarzy, więc mogą nie spełniać kryteriów gatunkowych, a także nie być najlepiej językowo napisane. Zaletą jest ich darmowa dostępność w Internecie. Do pobrania opowiadań wykorzystano nieoficjalne API umożliwiające pobranie opowiadań z AO3 po podaniu ich numeru identyfikującego. API zostało napisane i uzupełnione przez użytkownika Armindo Flores na GitHubie [32].

Wykorzystywane API zwraca zawartość tekstu użytkownika dla jego numeru identyfikującego. Każde opowiadanie na AO3 posiada indywidualny identyfikator. Identyfikator jest wyświetlany na stronie HTML wyświetlonej jako wynik wyszukiwania. Pobieranie danych z AO3 działa więc następująco – pobierana jest strona HTML z wynikami wyszukiwania dla konkretnego gatunku i z niej wybierane są identyfikatory opowiadań. Identyfikatory wykorzystywane są jako parametr do wywołania metody API, pobierającej treść opowiadania dla danego numeru identyfikującego opowiadania. Po pobraniu treści opowiadania jest ona przygotowywana pod wyuczenie na niej modelu analizy tekstu.

```
def _save_all_texts():
    texts = []
    genres_from_text = []
    for i in range(len(GENRES)):
        texts_genre = _get_texts_for_genre(GENRES[i])
        texts.extend(texts_genre)
        for j in range(len(texts_genre)):
            genres_from_text.append(GENRES[i])

    with open(SCRAPED_TEXT_FILENAME, 'w', encoding="utf-16") as f:
        writer = csv.writer(f)
        writer.writerow(["label", "text"])
        for line in range(len(texts)):
            if len(texts[line]) > 10:
                writer.writerow([genres_from_text[line], texts[line]])
    f.close()
    with open(SCRAPED_TEXT_LABELS_FILENAME, 'w', encoding="utf-16") as f:
        for line in range(len(texts)):
            f.write(genres_from_text[line] + ",")
    f.close()
    with open(SCRAPED_GENRES_FILENAME, 'w', encoding="utf-16") as f:
        for genre in GENRES:
            f.write(genre + ",")
    f.close()
```

Kod 45. Pobranie opowiadań za pomocą API i zapisanie uzyskanych danych do trzech plików: pliku zawierającego opowiadania, pliku zawierającego etykiety opowiadań oraz pliku zawierającego gatunki przypisane do opowiadań

```

def _get_texts_for_genre(genre: str):
    ids = []
    for i in range(PAGE_START, PAGE_END):
        page_ids = _get_genre_work_ids_from_page(genre, str(i))
        print(i, end=" ")
        ids.extend(page_ids)
    texts = []
    for i in range(0, len(ids)):
        texts.extend(_get_texts_from_story_id(ids[i]))
    return texts

```

Kod 46. Wywoływana metoda `_get_texts_for_genre(...)` – pobranie identyfikatorów opowiadań dla konkretnego gatunku, a następnie dla wszystkich otrzymanych identyfikatorów pobranie treści opowiadań

```

@retry(delay=60, jitter=5, tries=10)
def _get_genre_work_ids_from_page(genre: str, page: str):
    if genre == 'science%20fiction':
        url = "https://archiveofourown.org/tags/Science%20Fiction/works?commit=Sort+and+Filter&" \
            "exclude_work_search%5Barchive_warning_ids%5D%5B%5D=19&" \
            "exclude_work_search%5Barchive_warning_ids%5D%5B%5D=20&" \
            "exclude_work_search%5Bfreeform_ids%5D%5B%5D=60&" \
            "exclude_work_search%5Bfreeform_ids%5D%5B%5D=110&" \
            "exclude_work_search%5Bfreeform_ids%5D%5B%5D=176&" \
            "exclude_work_search%5Bfreeform_ids%5D%5B%5D=6276&" \
            "exclude_work_search%5Brating_ids%5D%5B%5D=12&" \
            "exclude_work_search%5Brating_ids%5D%5B%5D=13&" \
            "page=" + page + "&work_search%5Bcomplete%5D=&" \
            "work_search%5Bcrossover%5D=&" \
            "work_search%5Bdate_from%5D=&" \
            "work_search%5Bdate_to%5D=&" \
            "work_search%5Bexcluded_tag_names%5D=" \
            "Horror%20Adventure%20Comedy%20Crimes%26+Criminals%20Fantasy%20Historical%20Romance&" \
            "work_search%5Blanguage_id%5D=en&" \
            "work_search%5Bother_tag_names%5D=&" \
            "work_search%5Bquery%5D=&" \
            "work_search%5Bsort_column%5D=bookmarks_count&" \
            "work_search%5Bwords_from%5D=5000&" \
            "work_search%5Bwords_to%5D=20000"
    elif genre == "horror":...
    elif genre == "adventure":...
    elif genre == "comedy":...
    elif genre == "crime and mystery":...
    elif genre == "fantasy":...
    elif genre == "historical":...
    else:...
        response = requests.request("GET", url)
        ids = _extract_work_ids(response.text)
        if len(ids) == 0:
            print("Get works from page exception")
            raise Exception
    return ids

```

Kod 47. Wywoywana metoda `_get_genre_work_ids_from_page(...)` – pobranie wyniku wyszukiwania opowiadań w interesującym gatunku za pomocą wyszukiwarki AO3

```

def _extract_work_ids(html: str):
    ids = []
    find = re.findall("work_\d+", html)
    for identifier in find:
        ids.append(int(str(identifier).strip("work_")))
    return ids

```

Kod 48. Wywoywana metoda `_extract_work_ids(...)` – pobranie identyfikatorów opowiadań z uzyskanej w wyniku wyszukiwania strony HTML

```
def _get_texts_from_story_id(identifier: int):
    try:
        work = _get_work(identifier)
        return _prepare_work(work)
    except TimeoutError:
        return []
```

Kod 49. Wywoływana metoda `_get_texts_from_story_id(...)` – pobranie treści opowiadań dla uzyskanych identyfikatorów i przygotowanie danych

```
@timeout(100.0)
@retry(delay=60, tries=3)
def _get_work(identifier: int):
    return AO3.Work(identifier)
```

Kod 50. Wywoływana metoda `_get_work(...)` – pobranie treści opowiadania o danym identyfikatorze przy użyciu API AO3

```
@timeout(120.0)
def _prepare_work(work):
    texts = []
    for i in range(work.nchapters):
        text = str(work.chapters[i].text).replace("!", " ! ").replace("?", " ? ") \
            .replace(".", " .").replace("\n", "").replace(" ", "").split(" ")
        if len(text) > 100:
            texts.append(' '.join(text[:len(text) - 1]))
    return texts
```

Kod 51. Wywoływana metoda `_prepare_work(...)` – wstępne przygotowanie opowiadania do wykorzystania do w module analizy tekstu

7.4. Moduł analizy tekstu

Moduł analizy tekstu wykorzystuje elementy sztucznej inteligencji do analizy opowiadań pisanych przez użytkowników oraz generowanych przez moduł generowania tekstu pod kątem określonych kryteriów. Analizowane są poziom poprawności językowej, poziom zgodności z gatunkiem wylosowanym w wyzwaniu, poziom wykorzystania słów zachęty, poziom zróżnicowania słów wykorzystanych w opowiadaniu, ton, w jakim napisane jest opowiadanie (pozytywny, negatywny lub neutralny) oraz gatunek przypisany do opowiadania w zadaniu klasyfikacji. Niektóre z tych elementów – poziom poprawności językowej, poziom zgodności z gatunkiem wylosowanym w wyzwaniu, poziom wykorzystania słów zachęty oraz poziom zróżnicowania słów wykorzystanych w opowiadaniu – są wykorzystywane w minigrze, w której użytkownik próbuje zgadnąć, które z pary opowiadań otrzymało wyższy wynik w module analizy AI. Ten wynik jest sumą wartości, które opowiadanie otrzymało w wymienionych elementach.

Moduł analizy tekstu napisany został w języku Python przy użyciu mikro-frameworku Flask i wdrożony w chmurze Heroku [15]. Dzięki temu działa niezależnie od aplikacji mobilnej i czasochłonne operacje, które wykonuje, nie powodują zamrożenia aplikacji mobilnej. Dane produkowane przez moduł analizy tekstu zapisywane są w bazie danych i są dostępne od momentu publikacji. Zanim zostaną zapisane użytkownikowi wyświetlna jest informacja, że moduł analizy tekstu jest w trakcie analizowania opowiadania i że dane zostaną wyświetcone, gdy analiza się zakończy. Uruchomienie analizy opowiadania z wykorzystaniem elementów sztucznej inteligencji oraz zapisanie uzyskanych wartości do bazy danych zaprezentowano w kodzie 52 – 53.

```

@app.route('/analyse', methods=['POST'])
def analyse_story():
    try:
        data = request.get_json()
        story_ref = data['story_ref']
        story = data['story_content']
        prompt = data['fk_prompt']
        genre = data['prompt_genre']
        words = str(data['prompt_words']).split(",")
        save_analysis(str(story), str(story_ref), str(prompt), str(genre), words)
        return {"message": "ok"}
    except Exception as e:
        print(e)

```

Kod 52. Uruchomienie analizy opowiadania z wykorzystaniem elementów sztucznej inteligencji

```

def save_analysis(story_text: str, story_ref: str, prompt_genre: str, words):
    try:
        create_empty(story_ref)
        save_sentiment_to_database(story_text, story_ref)
        save_correctness_to_database(story_text, story_ref)
        save_word_variety_to_database(story_text, story_ref)
        save_prompt_completeness(story_text, words, story_ref)
        save_genre_and_promp_genre_probability(story_text, story_ref, prompt_genre)
    except Exception as e:
        print(e)

```

Kod 53. Wywoływana metoda save_analysis(...) – zapis wartości uzyskanych w wyniku analizy do bazy danych

Do oceny poziomu poprawności językowej opowiadania wykorzystano API JSpell Checker [33][34]. JSpell Checker w wersji darmowej sprawdza poprawność pisowni wyrazów oraz proponuje słowa, na które zamienić niepoprawnie napisane wyrazy. Jeżeli dane słowo nie jest poprawne, JSpell Checker szuka słów podobnie brzmiących lub o podobnej strukturze i pisowni. JSpell Checker udostępnia API, które przyjmuje dokument w jednym z dostępnych języków i zwraca informację o liczbie znalezionych błędów pisowni. Na potrzeby projektu pisownia sprawdzana jest względem American English oraz wyliczana jest wartość procentowa liczby błędów w stosunku do liczby wszystkich słów w opowiadaniu – większa wartość poziomu poprawności językowej oznacza większą poprawność językową opowiadania, czyli mniej błędów. Połączenie z JSpellChecker API oraz zapis obliczonego poziomu poprawności językowej w bazie danych zaprezentowano w kodzie 54 – 55.

```

def save_correctness_to_database(story_text: str, story_ref: str):
    try:
        correctness = float(float(get_correctness_score(story_text)) / len(story_text.split()))
        correctness = 1.0 - correctness
        query = "update statsai set correctness = " + str(correctness) + " where fk_story = " + story_ref
        with _get_connection() as conn:
            with conn.cursor() as cursor:
                cursor.execute(query)
    except Exception as e:
        print(e)

```

Kod 54. Obliczenie liczby błędów w stosunku do długości opowiadania i zapis wartości w bazie danych

```

def get_correctness_score(story: str):
    url = "https://jspell-checker.p.rapidapi.com/check"
    payload = {
        "language": "enUS",
        "fieldvalues": story,
        "config": {
            "forceUpperCase": False,
            "ignoreIrregularCaps": True,
            "ignoreFirstCaps": False,
            "ignoreNumbers": True,
            "ignoreUpper": True,
            "ignoreDouble": True,
            "ignoreWordsWithNumbers": True
        }
    }
    headers = {
        "content-type": "application/json",
        "X-RapidAPI-Key": "2f62eb4641msh4888131718c480ep1b0785jsn1dcbf5ed4a10",
        "X-RapidAPI-Host": "jspell-checker.p.rapidapi.com"
    }

    response = requests.request("POST", url, json=payload, headers=headers)
    errors = json.loads(response.text)
    return errors['spellingErrorCount']

```

Kod 55. Wywoływana metoda `get_correctness_score(...)` – wysłanie zapytania do API JSpell Checker. Ignorowane są wyrazy z niregularnym użyciem wielkiej litery (pRzYKŁAd – takie słowa są często używane w slangu internetowym), wyrazy napisane caps lockiem (skróty, akronimy mogą być wymyślane przez użytkowników), powtarzonych słów (powtórzone słowa mogą być używane przez użytkowników w celu zwiększenia wydźwięku), cyfry i słowa zawierające cyfry. Błędy braku wielkiej litery po kropce nie są ignorowane.

Do oceny tonu opowiadania wykorzystano API Text Sentiment Analysis Method [35]. Analiza sentymentu opiera się na rozwiązańach wypracowanych w dziedzinie przetwarzania języka naturalnego (Natural Language Processing) i ma na celu określenie, jaki jest ton emocjonalny wypowiedzi – pozytywny, negatywny lub neutralny. Na potrzeby projektu sentyment opowiadań analizowany jest jako ciekawostka dla użytkowników i nie jest wliczany w całkowitą ocenę opowiadania, którą użytkownicy próbują oszacować w minigrze. Text Sentiment Analysis Method jest API, które przyjmuje dokument w języku angielskim i zwraca informację o liczbie linijek o pozytywnym, negatywnym i neutralnym tonie emocjonalnym a także stosunek pozytywnych, negatywnych i neutralnych linii do wszystkich linii w dokumencie. Jako ocenę tonu emocjonalnego opowiadania wykorzystano wartości procentowe zaokrąglone do części całkowitej. Połączenie z API Text Sentiment Analysis Method oraz zapis obliczonych wartości w bazie danych zaprezentowano w kodzie 56 – 57.

```

def save_sentiment_to_database(story_text: str, story_ref: str):
    try:
        sentiment = get_sentiment_analysis(story_text)
        positiveness = round(float(sentiment['pos_percent'])[:-1])
        negativeness = round(float(sentiment['neg_percent'])[:-1])
        neutralness = round(float(sentiment['mid_percent'])[:-1])
        query = "update statsai "
        query += "set positiveness = '" + str(positiveness) + "%', "
        query += "negativeness = '" + str(negativeness) + "%', "
        query += "neutralness = '" + str(neutralness) + "%' "
        query += "where fk_story = " + story_ref
        with _get_connection() as conn:
            with conn.cursor() as cursor:
                cursor.execute(query)
    except Exception as e:
        print(e)

```

Kod 56. Zaokrąglenie otrzymanych wartości do części całkowitej i zapis wartości w bazie danych

```

def get_sentiment_analysis(story: str):
    url = "https://text-sentiment.p.rapidapi.com/analyze"
    payload = "text=" + story
    headers = {
        "content-type": "application/x-www-form-urlencoded",
        "X-RapidAPI-Key": "2f62eb4641sh4888131718c480ep1b0785jsn1dcbf5ed4a10",
        "X-RapidAPI-Host": "text-sentiment.p.rapidapi.com"
    }
    response = requests.request("POST", url, data=payload, headers=headers)
    result = json.loads(response.text)
    return result

```

Kod 57. Wywoływana metoda `get_sentiment_analysis(...)` – wysłanie zapytania do API
Text Sentiment Analysis Method

Poziom zróżnicowania słów wykorzystanych w opowiadaniu obliczany jest jako stosunek liczby różnych słów użytych w tekście do liczby wszystkich słów w tekście. Im większy poziom zróżnicowania słów dla danego opowiadania, tym barwniejszym i mniej monotonnym językiem posługuje się autor opowiadania. Poziom zróżnicowania słów wykorzystanych w opowiadaniu zapisywany jest jako wartość procentowa zaokrąglona do części całkowitej. Obliczanie poziomu zróżnicowania słów oraz zapis wartości w bazie danych zaprezentowano w kodzie 58 – 59.

```

def save_word_variety_to_database(story_text: str, story_ref: str):
    try:
        word_variety: float = calculate_word_variety(story_text)
        query = "update statsai set vocabvariety = " + str(word_variety) + " where fk_story = " + story_ref
        with _get_connection() as conn:
            with conn.cursor() as cursor:
                cursor.execute(query)
    except Exception as e:
        print(e)

```

Kod 58. Zapis poziomu zróżnicowania słów w opowiadaniu w bazie danych

```

def calculate_word_variety(story: str):
    word_set: set = set()
    word_list = story.split()
    for word in word_list:
        word_set.add(word)
    return round(float(len(word_set)) / float(len(word_list)), 2)

```

Kod 59. Wywoływana metoda `calculate_word_variety(...)` – obliczenie poziomu zróżnicowania słów w opowiadaniu

Poziom wykorzystania słów zachęty jest obliczany przy wykorzystaniu lematyzacji. Nie wystarczy sprawdzić każdego słowa w opowiadaniu i porównać go do słów zachęty, ponieważ słowa mają różne formy odmienione. Użytkownik, który wykorzysta dane słowo na przykład w formie mnogiej, wciąż je wykorzystał i powinno być one wliczone do poziomu wykorzystana słów zachęty. W tym celu wykorzystano proces lematyzacji, która jest procesem grupowania odmiennych form słowa, aby mogły być one analizowane przez pojedynczy element. Lematyzacja polega na usuwaniu końcówek gramatycznych ze słów i pozostawieniu tylko podstawy słowotwórczej. Lematyzację wykonano za pomocą NLTK (Natural Language Toolkit) [36], czyli zestawu bibliotek w języku Python do statystycznego i symbolicznego przetwarzania języka naturalnego. Przed dokonaniem na tekście procesu lematyzacji należy odpowiednio go przygotować – zmienić wszystkie litery na małe, usunąć spacje z przodu i na końcu tekstu, powielone spacje, znaki interpunkcyjne oraz liczby. Proces lematyzacji zaprezentowano w kodzie 60 – 62.

```

def text_preprocessing(text: str):
    return lemmatizer(preprocess(text))

```

Kod 60. Wykonanie procesu lematyzacji

```

def lemmatizer(text):
    wl = WordNetLemmatizer()
    word_pos_tags = nltk.pos_tag(word_tokenize(text))
    a = [wl.lemmatize(tag[0], get_wordnet_pos(tag[1])) for idx, tag in enumerate(word_pos_tags)]
    return " ".join(a)

```

Kod 61. Wywoływana metoda lemmatizer(...) wykorzystująca WordNetLemmatizer z pakietu NLTK

```

def preprocess(text):
    text = text.lower()
    text = text.strip()
    text = re.compile('<.*?>').sub(' ', text)
    text = re.compile('[\s]+ % re.escape(string.punctuation)).sub(' ', text)
    text = re.sub(r'[0-9]', ' ', text)
    text = re.sub(r'^[\w\s]+', ' ', str(text).lower().strip())
    text = re.sub(r'\d+', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    return text

```

Kod 62. Wywoywana metoda preprocess(...) – metoda zamienia wszystkie litery w podanym tekście na małe, usuwa z tekstu spacje z przodu i na końcu, powielone spacje, znaki interpunkcyjne oraz liczby

Poziom zgodności opowiadania z gatunkiem wylosowanym w wyzwaniu oceniany jest z wykorzystaniem klasyfikatora Naive Bayes. Ten sam klasyfikator jest wykorzystywany do klasyfikowania opowiadania do określonego gatunku. Klasyfikator Naive Bayes (inaczej: naiwny klasyfikator bayesowski, naiwny klasyfikator Bayesa, model cech niezależnych) jest prostym klasyfikatorem probabilistycznym opartym na twierdzeniu Bayesa. Klasyfikator może zostać zastosowany do klasyfikacji tekstu do określonego gatunku na podstawie cech tego tekstu i jego podstawą jest założenie o wzajemnej niezależności cech, na podstawie których dokonywana jest klasyfikacja. Na podstawie cech wyznaczane jest prawdopodobieństwo, że dany tekst należy do danego gatunku.

Aby zastosować klasyfikator Naive Bayes do analizy języka naturalnego konieczne usunięcie znaków interpunkcyjnych i lematyzacja słów, co zostało opisane przy okazji omówienia poziomu wykorzystania słów zachęty. Dodatkowo, konieczne jest usunięcie z tekstów tak zwanych słów stopu, czyli słów, które bardzo często występują (na przykład w języku angielskim *not*, *the*, *a*, *are*, *is*, ...). Usuwanie z analizowanych tekstów słów stopu zaprezentowano w kodzie 63.

Klasyfikator Naive Bayes operuje na danych binarnych, zatem teksty muszą zostać zapisane jako sekwencje zer i jedynek. Dla każdego analizowanego gatunku wybrano najczęściej występujące słowa, a następnie dla każdego z najczęściej występujących słów we wszystkich gatunkach słowo oznaczane jest jako 1, jeżeli występuje w opowiadaniu lub jako 0, jeżeli nie występuje w opowiadaniu. W efekcie opowiadanie jest reprezentowane jako sekwencja zer i jedynek o długości N, gdzie N to liczba najczęściej występujących słów we wszystkich gatunkach. Transformację opowiadania do sekwencji zer i jedynek zaprezentowano w kodzie 64 – 68.

Zaimplementowano klasyfikator Naive Bayes. Został on wyuczony odpowiednio przygotowanymi danymi pobranymi z portalu AO3 [\[4\]](#). Model na podstawie wyuczonych danych potrafi dla konkretnego opowiadania zwrócić gatunek, dla którego prawdopodobieństwo, że dany tekst do niego należy, jest największe oraz dla określonego gatunku zwrócić prawdopodobieństwo, że dane opowiadanie do niego należy. Te dwie informacje zapisywane są w bazie danych jako gatunek przypisany do opowiadania w zadaniu klasyfikacji oraz poziom zgodności z gatunkiem wylosowanym w wyzwaniu. Obliczenie prawdopodobieństwa, że opowiadanie należy do podanego

gatunku oraz wyznaczenie gatunku, dla którego prawdopodobieństwo, że opowiadanie do niego należy jest największe oraz zapis tych wartości w bazie danych zaprezentowane są w kodzie 69 – 71.

```
def stop_words(text):
    a = [i for i in text.split() if i not in stopwords.words('english')]
    return ' '.join(a)
```

Kod 63. Element wstępnego przetwarzania danych – usuwanie z danych uczących tak zwanych słów stopu, czyli bardzo często występujących słów, metoda `stopwords.words(...)` pochodzi z pakietu NLTK

```
if __name__ == '__main__':
    data = load_normalized_data()
    word_in_genre = {}
    for genre in SELECTED_GENRES:
        word_in_genre[genre] = get_words_in_genre(genre)
        all_words = get_all_words(word_in_genre)

    results = []
    for text in data['text']:
        results.append(transform_text_to_binary_sequence(all_words, text))
    with open("../genre_ai_nave_bayes/training_data/test_zero_one_X20P.txt", 'w', encoding="utf-16") as fp:
        for result in results:
            for r in result:
                fp.write(str(r))
                fp.write("\n")
```

Kod 64. Uruchomienie transformacji opowiadań do sekwencji zer i jedynek

```
def load_normalized_data():
    return pd.read_csv("processing_files/cleaned_data_X20P.csv", delimiter=",")
```

Kod 65. Wywoływana metoda `load_normalized_data()` – ładowanie wstępnie przetworzonych danych do dalszego przetwarzania

```
def get_words_in_genre(genre):
    dataset = load_normalized_data()
    df = dataset[['label', 'text']]
    texts = ""

    for index, row in df.iterrows():
        if row["label"] == genre:
            texts += " " + str(row["text"])

    word_counter = Counter(texts.split())
    most_common_words_in_genre = []
    i = 0
    for word, count in word_counter.most_common():
        if count < ALL_WORD_APPEARANCES_IN_GENRE and i < NO_OF_WORDS_FOR_GENRE:
            most_common_words_in_genre.append(word)
            i = i + 1

    most_common_words_in_genre_dict = {word: 0 for word in most_common_words_in_genre}
    return most_common_words_in_genre_dict
```

Kod 66. Wywoływana metoda `get_words_in_genre(...)` – wybieranie najczęściej występujących w opowiadaniach słów dla każdego gatunku. Nie są wybierane najczęściej występujące słowa oraz liczba wybieranych słów określona jest parametrem `NO_OF_WORDS_FOR_GENRE`

```

def get_all_words(words_in_genre):
    all_words = []
    for genre in SELECTED_GENRES:
        all_words.extend(words_in_genre[genre])
    with open(ALL_WORDS_FILENAME_OUT, 'w', encoding="utf-16") as f:
        for word in all_words:
            f.write(word + ",")
    f.close()
    return all_words

```

Kod 67. Wywoływana metoda `get_all_words(...)` – zapisanie najczęściej występujących słów dla każdego słowa w postaci jednej listy słów oraz zapis danych do pliku

```

def transform_text_to_binary_sequence(all_words, text):
    text = str(text)
    contains = []
    for i, c in enumerate(all_words):
        if text.__contains__(c):
            contains.append("1,")
        else:
            contains.append("0,")
    return contains

```

Kod 68. Wywoływana metoda `transform_text_to_binary_sequence(...)` – transformacja opowiadania do sekwencji zer i jedynek, w taki sposób, że dla każdego z najczęściej występujących słów we wszystkich gatunkach słowo oznaczane jest jako 1, jeżeli występuje w opowiadaniu lub jako 0, jeżeli nie występuje w opowiadaniu

```

def save_genre_and_promp_genre_probability(story_text: str, story_ref: str, genre_prompt: str):
    try:
        genre, prob_for_genre = analyze_genre(story_text, genre_prompt)
        query = "update statsai set genreaccuracy = " + str(prob_for_genre) + ", " \
                "genre = '" + genre + "' " \
                "where fk_story = " + story_ref
        with _get_connection() as conn:
            with conn.cursor() as cursor:
                cursor.execute(query)
    except Exception as e:
        print(e)

```

Kod 69. Obliczenie prawdopodobieństwa, że opowiadanie należy do podanego gatunku oraz wyznaczenie gatunku, dla którego prawdopodobieństwo, że opowiadanie do niego należy jest największe i zapis tych wartości w bazie danych

```

def analyze_genre(story: str, prompt_genre: str):
    try:
        preprocessed = text_preprocessing(story)
        cleaned = text_checking(preprocessed)
        genre_index = GENRES.index(prompt_genre)
        genre, prop_for_genre = calculate_probability_for(cleaned, genre_index)
        return genre, prop_for_genre
    except Exception as e:
        print(e)

```

Kod 70. Wywoływana metoda `analyze_genre(...)` – wykorzystanie klasyfikatora Naive Bayes do obliczenia prawdopodobieństwa, że opowiadanie należy do podanego gatunku oraz wyznaczenia gatunku, dla którego prawdopodobieństwo, że opowiadanie do niego należy jest największe

```

def calculate_probability_for(text, genre_id):
    text = s.sparse.csc_matrix(text)
    p_x_y_nb_estimate = load_matrix(p_x_y_NB_estimate_file_in)
    estimate_a_priori = load_vector(estimate_a_priori_nb_file_in)
    result = p_y_x_nb(estimate_a_priori, p_x_y_nb_estimate, text)[0]
    max_index = 0
    for i in range(result[0]):
        if result[i] > result[max_index]:
            max_index = i
    return genres[max_index], float(result[genre_id])

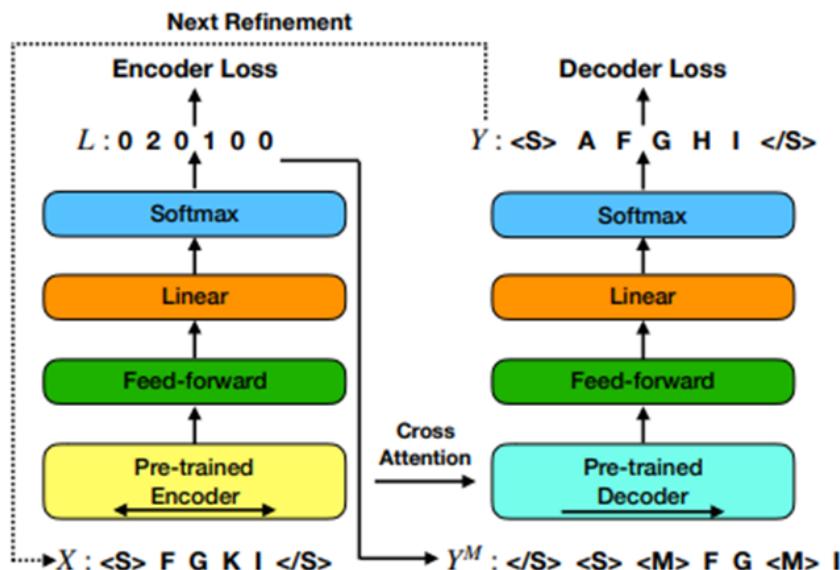
```

Kod 71. Wywoływana metoda `calculate_probability_for(...)` – obliczenie prawdopodobieństwa, że opowiadanie należy do podanego gatunku oraz wyznaczenie gatunku, dla którego prawdopodobieństwo, że opowiadanie do niego należy jest największe przez klasyfikator Naive Bayes na podstawie wyuczonych danych

7.5. Moduł generowania tekstu

Do generowania tekstu wykorzystano pretrenowany model CBART [7] oraz model GPT-Neo [13]. Model GPT-Neo jest udostępniony jako API [13], które przyjmuje początek zdania i zwraca wygenerowany z użyciem modelu ciąg dalszy.

Model CBART generuje tekst na podstawie słów kluczowych. Korzysta on z wytrenowanego wcześniej modelu BART i przekazuje część procesu generowania z dekodera do enkodera poprzez podział tego zadania na dwa podzadania co pozwala na poprawę jakości generowanych zdań. Model BART jest rozszerzony o klasyfikator na poziomie tokenów (token-level) dla enkodera, co wskazuje dekoderowi, na której pozycji w tekście powinna zostać wykonana jakaś akcja (wstawienie/usunięcie/edytacja). Dekoder później analizuje tokeny wejściowe - wstawia je przed określonymi pozycjami i następnie ponownie predykuje tokeny z niską pewnością (low confidence). Żeby uniknąć opóźnień we wnioskowaniu, dekoder predykuje wszystkie tokeny równolegle. Przeprowadzone przez autora tekstu eksperymenty wskazują na to, że CBART jest w stanie generować wiarygodny tekst o wysokiej jakości i różnorodności jednocześnie zwiększając szybkość wnioskowania.



Obraz 3. Schemat działania modelu

Wybrany model składa się z 2 modułów: enkodera i dekodera. Enkoder jest odpowiedzialny za dostarczenie informacji doskonalących o wysokiej ziarnistości (coarse-grained refinement information) do dekodera. Innymi słowy – ekoder ma wskazać dekoderowi gdzie zamienić oraz wstawić tekst. Jest tu wykorzystywana pełna warstwa przesyłania w przód. Na wejściu enkoder otrzymuje niekompletne zdanie, a na wyjście podaje odpowiadające mu zdanie w postaci sekwencji etykiet: 0 - kopia (zachowanie aktualnego tokenu), 1- zamiana (zamiana tokenu na inny, żeby tekst był bardziej spójny), 2 - wstawienie (wstawienie tokenu przed aktualnym, żeby uzupełnić tekst). Na podstawie tak utworzonej sekwencji dekoder będzie wiedział jak udoskonalić zdanie kandydujące. Dekoder przyjmuje wejście Y^M i ma za zadanie zrekonstruować oryginalny tekst Y , który będzie predykować na podstawie modelu BART. Optymalizacja dekodera następuje poprzez minimalizowanie błędu rekonstrukcji.

Trenowanie modelu polega na minimalizacji całkowitej straty dla enkodera i dekodera. Podczas trenowania wejście dekodera Y^M jest tworzone na podstawie sekwencji złotych etykiet enkodera, natomiast podczas wnioskowania jest tworzone na podstawie predykowanej sekwencji etykiet enkodera.

Model CBART wykorzystywany jest do tworzenia zdań wykorzystujących słowa podane jako wejście. W zastosowaniu generacji tekstu na potrzeby przedsięwzięcia jako wejście podawane są losowe słowo oznaczone jako cecha gatunku z wyzwania przez klasyfikator Naive Bayes oraz pierwsze słowo zachęty. Wygenerowane przez model CBART zdanie jest następnie wykorzystywane jako wejście do API GPT-Neo. Do wyniku generacji przez model GPT-Neo następnie doklejane są zdania wygenerowane przez model CBART wykorzystujące kolejne słowa zachęty oraz uzyskany tekst ponownie podawany jest jako wejście do API GPT-Neo. Generowanie tekstu przedstawione jest w kodzie 72.

```
def update_generate_local():
    rows = get_stories_for_generation()
    for row in rows:
        genre: str = _get_prompt_genre(row[3])
        words_string: str = str(_get_prompt_words(row[3]))
        words = words_string.split(',')
        max_len = 500 / len(words)
        text: str = generate_cbart("cbart-large", [get_random_word_from_genre(genre), words[0]], max_len=max_len) + " "
        text = generate_gpt_neo(text)
        for i in range(1, len(words)):
            text = text + generate_cbart("cbart-large", [words[i]], max_len=max_len) + " "
            text = generate_gpt_neo(text)
        existing = ref_for_generated(str(row[0]))
        if existing is None:
            save_generated(text, str(row[0]), str(row[2]), str(row[3]))
            print(text)
        else:
            update_if_already_generated(str(existing[0]), text, row[2], row[3])
            print(text)
            start_analyse(str(existing[0]), text, str(row[3]), genre, words_string)
    url = "https://zpi-aaab.herokuapp.com/updateDate"
    requests.request("GET", url)
```

Kod 72. Lokalne generowanie tekstu dla opowiadań użytkowników z wykorzystaniem modeli CBART oraz GPT-Neo

```

for i in range(0, length, batch_size):
    indicate_labels = indicate_labels_list[i:i + batch_size]
    encoder_inputs = encoder_inputs_list[i:i + batch_size]
    decoder_inputs = None
    predict_outputs, refinement_steps = generate(model, tokenizer, encoder_inputs, indicate_labels,
                                                encoder_loss_type,
                                                max_insert_label,
                                                device,
                                                decoder_inputs=decoder_inputs,
                                                stop_tokens_tensor=stop_tokens_tensor,
                                                sub_tokens_tensor=sub_tokens_tensor,
                                                temperature=temperature,
                                                do_sample=do_sample,
                                                top_k=top_k,
                                                top_p=top_p,
                                                refinement_steps=refinement_steps,
                                                max_refinement_steps=max_refinement_steps,
                                                adaptive=True,
                                                repetition_penalty=repetition_penalty,
                                                threshold=threshold,
                                                decoder_chain=decoder_chain,
                                                rank_lm=rank_lm,
                                                max_len=max_len)
batch_size = len(indicate_labels)
generated = ''
for b in range(batch_size):
    generated = tokenizer.decode(predict_outputs[b].tolist()[1:-1], clean_up_tokenization_spaces=False)
return generated

```

Kod 73. Fragment wywoływanej metody `generate_cbart(...)` – wywołanie pretrenowanego modelu CBART

```

def generate_gpt_neo(previous: str):
    try:
        payload = {"inputs": previous + " "}
        return query(payload)[0]["generated_text"]
    except Exception as e:
        print(e)
        return previous + " "

```

Kod 74. Wywoływana metoda `generate_gpt_neo(...)` – przygotowanie wejścia do API GPT-Neo

```

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload, timeout=60)
    return response.json()

```

Kod 75. Wywoływana metoda `query(...)` – wysłanie zapytania do API GPT-Neo

```

def ref_for_generated(story_ref: str):
    try:
        query = "select ref from stories where generatedfor = " + story_ref
        with _get_connection_local() as conn:
            with conn.cursor() as cursor:
                cursor.execute(query)
                row = cursor.fetchone()
                return row
    except Exception as e:
        print(e)

```

Kod 76. Wywoływana metoda `ref_for_generated(...)` – pobranie identyfikatora wygenerowanego opowiadania w bazie danych

```

def save_generated(text: str, story_ref: str, user_ref: str, fk_prompt: str):
    try:
        url = "https://zpi-aaab.herokuapp.com/publishGenerated"
        payload = {
            "content": str(text),
            "fk_prompt": str(fk_prompt),
            "fk_user": str(user_ref),
            "generatedfor": str(story_ref)
        }
        headers = {
            "content-type": "application/json"
        }
        response = requests.request("POST", url, json=payload, headers=headers)
        while not response.ok:
            response = requests.request("POST", url, json=payload, headers=headers)
            sleep(1)
    except Exception as e:
        print(e)

```

Kod 77. Wywoływana metoda `save_generated(...)` – zapisanie wygenerowanego opowiadania w bazie danych. Metoda wywoływana jest jeżeli dla opowiadania jeszcze nie wygenerowano opowiadania przy użyciu elementów sztucznej inteligencji. Po zapisaniu wygenerowanego opowiadania automatycznie uruchamiana jest jego analiza.

```

def update_if_already_generated(story_ref: str, content: str, fk_user: str, fk_prompt: str):
    try:
        url = "https://zpi-aaab.herokuapp.com/updateGenerated"
        payload = {
            "content": str(content),
            "fk_prompt": str(fk_prompt),
            "fk_user": str(fk_user),
            "generatedfor": str(story_ref)
        }
        headers = {
            "content-type": "application/json"
        }
        try:
            requests.request("POST", url, json=payload, headers=headers)
        except requests.exceptions.ReadTimeout:
            pass
    except Exception as e:
        print(e)

```

Kod 78. Wywoływana metoda `update_if_already_generated(...)` – aktualizowanie wygenerowanego opowiadania. Metoda wywoywana jest jeżeli dla opowiadania jeszcze nie wygenerowano opowiadania przy użyciu elementów sztucznej inteligencji. Wywołanie metody ma związek z ograniczoną możliwością hostowania programu w chmurze.

```

def start_analyse(story_ref: str, story_content: str, fk_prompt: str, genre: str, words_analyze: str):
    try:
        prompt_words = words_analyze
        url = "https://zpi-ai.herokuapp.com/analyse"
        payload = {
            "story_ref": str(story_ref),
            "story_content": str(story_content),
            "fk_prompt": str(fk_prompt),
            "prompt_genre": str(genre),
            "prompt_words": str(prompt_words)
        }
        headers = {
            "content-type": "application/json"
        }
        try:
            requests.request("POST", url, json=payload, headers=headers)
        except requests.exceptions.ReadTimeout:
            pass
    except Exception as e:
        print(e)

```

Kod 79. Wywoływana metoda `start_analyse(...)` – uruchomienie analizy dla zaktualizowanego opowiadania.

Ze względu na ograniczone możliwości hostowania programu wykorzystującego dużo pamięci RAM w chmurze, program uruchamiany jest lokalnie. Czytane są wszystkie opowiadania opublikowane od ostatniego lokalnego uruchomienia generacji tekstu i dla tych opowiadań uruchamiane jest generowanie. Po opublikowaniu opowiadania przez użytkownika uruchamiane jest uproszczony schemat generowania wykorzystujący API oparte na modelu GPT-Neo [37]. Opowiadania generowane są na podstawie losowego zdania z opowiadania użytkownika, losowego słowa oznaczonego jako cecha gatunku z wyzwania przez klasyfikator Naive Bayes oraz słów zachęty.

8. Przypadki testowe

<i>Id.</i>	PT1
<i>Idea testowa</i>	Sprawdzenie poprawności logowania
<i>Dane testowe</i>	Login użytkownika Hasło użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Podaj login użytkownika 2. Podaj hasło użytkownika 3. Kliknij przycisk “Log in”
<i>Oczekiwane wyniki</i>	Wyświetlenie ekranu głównego z opowiadaniami użytkownika
<i>Id.</i>	PT2
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji loginu użytkownika
<i>Dane testowe</i>	Niepoprawny login użytkownika – nieistniejący login użytkownika Hasło użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Podaj niepoprawny login użytkownika 2. Podaj hasło użytkownika 3. Kliknij przycisk “Log in”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o braku zarejestrowanego użytkownika o takim loginie
<i>Id.</i>	PT3
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji hasła użytkownika
<i>Dane testowe</i>	Login użytkownika Niepoprawne hasło użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Podaj login użytkownika 2. Podaj niepoprawne hasło użytkownika 3. Kliknij przycisk “Log in”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o błędnych danych logowania

<i>Id.</i>	PT4
<i>Idea testowa</i>	Sprawdzenie poprawności rejestracji
<i>Dane testowe</i>	Login użytkownika Hasło użytkownika Adres e-mail użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Podaj login użytkownika 2. Podaj hasło użytkownika 3. Powtórz hasło użytkownika 4. Podaj adres e-mail użytkownika 5. Kliknij przycisk “Register”
<i>Oczekiwane wyniki</i>	Wyświetlenie ekranu głównego z opowiadaniami użytkownika oraz samouczka obsługi aplikacji
<i>Id.</i>	PT5
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji loginu użytkownika przy rejestracji
<i>Dane testowe</i>	Niepoprawny login użytkownika – pusty, zbyt długi lub już wykorzystany login użytkownika Hasło użytkownika Adres e-mail użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Podaj niepoprawny login użytkownika 2. Podaj hasło użytkownika 3. Powtórz hasło użytkownika 4. Podaj adres e-mail użytkownika 5. Kliknij przycisk “Register”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnym loginie użytkownika

<i>Id.</i>	PT6
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji hasła użytkownika przy podaniu błędного hasła użytkownika przy rejestracji
<i>Dane testowe</i>	Login użytkownika Niepoprawne hasło użytkownika – puste, zbyt długie lub za słabe hasło login użytkownika Adres e-mail użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Podaj login użytkownika 2. Podaj niepoprawne hasło użytkownika 3. Powtórz hasło użytkownika 4. Podaj adres e-mail użytkownika 5. Kliknij przycisk “Register”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnym haśle użytkownika
<i>Id.</i>	PT7
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji hasła użytkownika przy podaniu różnych haseł użytkownika przy rejestracji
<i>Dane testowe</i>	Login użytkownika Hasło użytkownika Niepoprawne hasło użytkownika – różne od hasła użytkownika Adres e-mail użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Podaj login użytkownika 2. Podaj hasło użytkownika 3. Błędnie powtórz hasło użytkownika 4. Podaj adres e-mail użytkownika 5. Kliknij przycisk “Register”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o nieudanej rejestracji w wyniku różnych wartości hasła

<i>Id.</i>	PT8
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji adresu e-mail użytkownika przy rejestracji
<i>Dane testowe</i>	Login użytkownika Hasło użytkownika Niepoprawny adres e-mail użytkownika – pusty, zbyt długi lub niepoprawny adres e-mail użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Podaj login użytkownika 2. Podaj hasło użytkownika 3. Powtórz hasło użytkownika 4. Podaj niepoprawny adres e-mail użytkownika 5. Kliknij przycisk “Register”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnym adresie e-mail użytkownika
<i>Id.</i>	PT9
<i>Idea testowa</i>	Sprawdzenie poprawności resetu hasła
<i>Dane testowe</i>	Login użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Kliknij przycisk “I've forgotten my password” 2. Podaj login użytkownika 3. Kliknij przycisk “Confirm”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o wysłaniu wiadomości e-mail na adres e-mail powiązany z kontem użytkownika
<i>Id.</i>	PT10
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji loginu przy resecie hasła
<i>Dane testowe</i>	Niepoprawny login użytkownika – nieistniejący login użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Kliknij przycisk “I've forgotten my password” 2. Podaj błędny login użytkownika 3. Kliknij przycisk “Confirm”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o braku zarejestrowanego użytkownika o takim loginie

<i>Id.</i>	PT11
<i>Idea testowa</i>	Sprawdzenie poprawności odzyskania loginu
<i>Dane testowe</i>	Adres e-mail użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Kliknij przycisk “I’ve forgotten my login” 2. Podaj adres e-mail użytkownika 3. Kliknij przycisk “Confirm”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o wysłaniu wiadomości e-mail na podany adres e-mail
<i>Id.</i>	PT12
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji adresu e-mail przy odzyskiwaniu loginu
<i>Dane testowe</i>	Niepoprawny adres e-mail użytkownika – pusty, zbyt długi, niepoprawny lub nie powiązany z żadnym kontem adres e-mail użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Kliknij przycisk “I’ve forgotten my login” 2. Podaj niepoprawny adres e-mail użytkownika 3. Kliknij przycisk “Confirm”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnym adresie e-mail
<i>Id.</i>	PT13
<i>Idea testowa</i>	Sprawdzenie poprawności edycji nazwy użytkownika
<i>Dane testowe</i>	Nowa nazwa użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Profile” 2. Kliknij przycisk “Change name” 3. Podaj nową nazwę użytkownika 4. Kliknij przycisk “Save”
<i>Oczekiwane wyniki</i>	Wyświetlenie nowej nazwy użytkownika na profilu użytkownika, przy opowiadaniach użytkownika i przy komentarzach użytkownika

<i>Id.</i>	PT14
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji nazwy użytkownika
<i>Dane testowe</i>	Niepoprawna nazwa użytkownika – pusta lub zbyt dобра nazwa użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdz do zakładki “Profile” 2. Kliknij przycisk “Change name” 3. Podaj nową nazwę użytkownika 4. Kliknij przycisk “Save”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnej nazwie użytkownika
<i>Id.</i>	PT15
<i>Idea testowa</i>	Sprawdzenie poprawności edycji hasła użytkownika
<i>Dane testowe</i>	Aktualne hasło użytkownika Nowe hasło użytkownika Login użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdz do zakładki “Profile” 2. Kliknij przycisk “Change password” 3. Podaj aktualne hasło użytkownika 4. Podaj nowe hasło użytkownika 5. Powtorz nowe hasło użytkownika 6. Kliknij przycisk “Save” 7. Kliknij przycisk “Log out” 8. Zaloguj się do konta z użyciem nowego hasła
<i>Oczekiwane wyniki</i>	Udane logowanie przy użyciu nowego hasła użytkownika – wyświetlenie ekranu głównego po zalogowaniu
<i>Id.</i>	PT16
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji hasła użytkownika przy podaniu błędnego aktualnego hasła użytkownika
<i>Dane testowe</i>	Niepoprawne aktualne hasło użytkownika Nowe hasło użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdz do zakładki “Profile” 2. Kliknij przycisk “Change password” 3. Podaj niepoprawne aktualne hasło użytkownika 4. Podaj nowe hasło użytkownika 5. Powtorz nowe hasło użytkownika 6. Kliknij przycisk “Save”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnym aktualnym haśle użytkownika

<i>Id.</i>	PT17
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji hasła użytkownika przy podaniu błędnego nowego hasła użytkownika
<i>Dane testowe</i>	Aktualne hasło użytkownika Niepoprawne nowe hasło użytkownika – puste, zbyt długie lub zbyt słabe hasło użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdz do zakładki “Profile” 2. Kliknij przycisk “Change password” 3. Podaj aktualne hasło użytkownika 4. Podaj niepoprawne nowe hasło użytkownika 5. Powtórz nowe hasło użytkownika 6. Kliknij przycisk “Save”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnym nowym haśle użytkownika
<i>Id.</i>	PT18
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji hasła użytkownika przy podaniu różnych nowych haseł użytkownika
<i>Dane testowe</i>	Aktualne hasło użytkownika Nowe hasło użytkownika Niepoprawne nowe hasło użytkownika – różne od nowego hasła użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdz do zakładki “Profile” 2. Kliknij przycisk “Change password” 3. Podaj aktualne hasło użytkownika 4. Podaj nowe hasło użytkownika 5. Błędnie powtórz nowe hasło użytkownika 6. Kliknij przycisk “Save”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o nieudanej zmianie hasła w wyniku różnych wartości nowego hasła
<i>Id.</i>	PT19
<i>Idea testowa</i>	Sprawdzenie poprawności edycji zdjęcia profilowego
<i>Dane testowe</i>	Wybrane zdjęcie profilowe
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdz do zakładki “Profile” 2. Kliknij przycisk “Change profile picture” 3. Wybierz nowe zdjęcie profilowe z listy dostępnych zdjęć
<i>Oczekiwane wyniki</i>	Wyświetlenie nowego zdjęcia profilowego na ekranie edycji zdjęcia profilowego, ekranie profilu użytkownika oraz przy komentarzach użytkownika

<i>Id.</i>	PT20
<i>Idea testowa</i>	Sprawdzenie poprawności edycji adresu e-mail użytkownika
<i>Dane testowe</i>	Nowy adres e-mail
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Profile” 2. Kliknij przycisk “Change e-mail address” 3. Podaj nowy adres e-mail 4. Kliknij przycisk “Save”
<i>Oczekiwane wyniki</i>	Zmiana adresu e-mail użytkownika
<i>Id.</i>	PT21
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji adresu e-mail użytkownika
<i>Dane testowe</i>	Niepoprawny adres e-mail użytkownika – pusty, zbyt długi lub niepoprawny adres e-mail użytkownika
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Profile” 2. Kliknij przycisk “Change e-mail address” 3. Podaj niepoprawny nowy adres e-mail 4. Kliknij przycisk “Save”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnym adresie e-mail użytkownika
<i>Id.</i>	PT22
<i>Idea testowa</i>	Sprawdzenie poprawności wyświetlania opowiadania
<i>Dane testowe</i>	
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Wybierz opowiadanie z listy
<i>Oczekiwane wyniki</i>	Wyświetlenie ekranu opowiadania z informacjami o wyzwaniu i autorze, treścią opowiadania, treścią wygenerowanego opowiadania, analizą opowiadania, oceną opowiadania oraz komentarzami dotyczącymi opowiadania

<i>Id.</i>	PT23
<i>Idea testowa</i>	Sprawdzenie poprawności oceniania opowiadania napisanego przez innego użytkownika
<i>Dane testowe</i>	Wartość oceny opowiadania
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Wybierz opowiadanie z listy 2. Oceń opowiadanie w skali 0,5 – 5
<i>Oczekiwane wyniki</i>	Wyświetlenie oceny opowiadania i zaktualizowanie średniej oceny opowiadania
<i>Id.</i>	PT24
<i>Idea testowa</i>	Sprawdzenie poprawności dodania opowiadania innego użytkownika do listy ulubionych
<i>Dane testowe</i>	
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Community” 2. Wybierz opowiadanie z listy 3. Kliknij ikonę pustego serduszka lub napis “Favourite”
<i>Oczekiwane wyniki</i>	Wyświetlenie pełnego serduszka i napisu “Unfavourite” przy opowiadaniu Wyświetlenie danego opowiadania w zakładce “Favourites” w profilu użytkownika
<i>Id.</i>	PT25
<i>Idea testowa</i>	Sprawdzenie poprawności usunięcia opowiadania innego użytkownika z listy ulubionych
<i>Dane testowe</i>	
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Community” 2. Wybierz opowiadanie z listy 3. Kliknij ikonę pełnego serduszka lub napis “Unfavourite”
<i>Oczekiwane wyniki</i>	Wyświetlenie pustego serduszka i napisu “Favourite” przy opowiadaniu Zaprzestanie wyświetlania danego opowiadania w zakładce “Favourites” w profilu użytkownika

<i>Id.</i>	PT26
<i>Idea testowa</i>	Sprawdzenie poprawności dodawania komentarza pod opowiadaniem innego użytkownika
<i>Dane testowe</i>	
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Community” 2. Wybierz opowiadanie z listy 3. Kliknij przycisk “Add comment” 4. Wpisz treść komentarza 5. Kliknij przycisk “Add”
<i>Oczekiwane wyniki</i>	Wyświetlenie dodanego komentarza pod opowiadaniem
<i>Id.</i>	PT27
<i>Idea testowa</i>	Sprawdzenie poprawności usuwania komentarza spod opowiadania innego użytkownika
<i>Dane testowe</i>	
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Community” 2. Wybierz opowiadanie z listy 3. Kliknij ikonę kosza na śmieci lub napis “Delete przy komentarzu” 4. Potwierdź chęć usunięcia komentarza
<i>Oczekiwane wyniki</i>	Wyświetlenie okna dialogowego do potwierdzenia chęci usunięcia komentarza Zaprzestanie wyświetlania komentarza pod opowiadaniem
<i>Id.</i>	PT28
<i>Idea testowa</i>	Sprawdzenie poprawności losowania wyzwań
<i>Dane testowe</i>	Wybrana liczba słów zachęty
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Your stories” 2. Kliknij na pływający przycisk plusa w prawym dolnym rogu ekranu 3. Kliknij na przycisk z odpowiednią liczbą słów zachęty
<i>Oczekiwane wyniki</i>	Wyświetlenie ekranu dodawania opowiadania z wylosowanym gatunkiem i odpowiednią liczbą słów zachęty

<i>Id.</i>	PT29
<i>Idea testowa</i>	Sprawdzenie poprawności dodania nowego opowiadania
<i>Dane testowe</i>	Tytuł opowiadania Treść opowiadania
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Your stories” 2. Kliknij na pływający przycisk plusa w prawym dolnym rogu ekranu 3. Kliknij na przycisk z odpowiednią liczbą słów zachęty 4. Podaj tytuł opowiadania 5. Podaj treść opowiadania 6. Kliknij przycisk “Publish”
<i>Oczekiwane wyniki</i>	Wyświetlenie dodanego opowiadania w zakładce “Your Stories”
<i>Id.</i>	PT30
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji tytułu nowego opowiadania
<i>Dane testowe</i>	Niepoprawny tytuł opowiadania – pusty, za długi lub zawierający słowo niedozwolone Treść opowiadania
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Your stories” 2. Kliknij na pływający przycisk plusa w prawym dolnym rogu ekranu 3. Kliknij na przycisk z odpowiednią liczbą słów zachęty 4. Podaj niepoprawny tytuł opowiadania 5. Podaj treść opowiadania 6. Kliknij przycisk “Publish”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnym tytule opowiadania

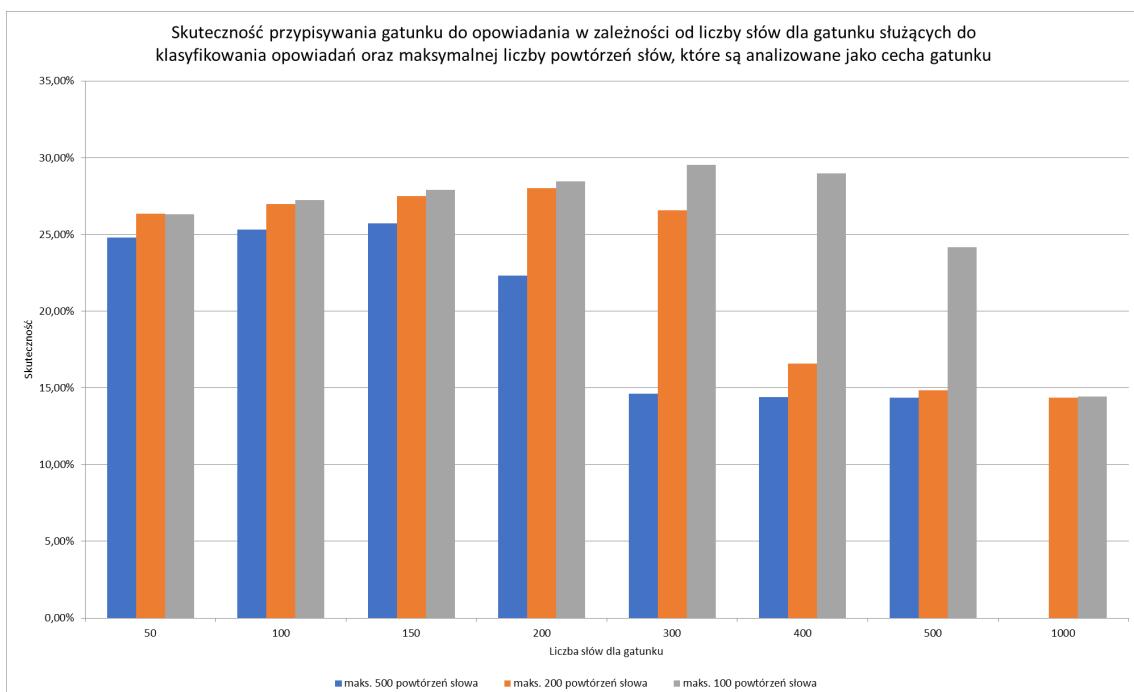
<i>Id.</i>	PT31
<i>Idea testowa</i>	Sprawdzenie poprawności walidacji treści nowego opowiadania
<i>Dane testowe</i>	Tytuł opowiadania Niepoprawna treść opowiadania – pusta, za długa lub zawierająca słowo niedozwolone
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do zakładki “Your stories” 2. Kliknij na płynący przycisk plusa w prawym dolnym rogu ekranu 3. Kliknij na przycisk z odpowiednią liczbą słów zachęty 4. Podaj tytuł opowiadania 5. Podaj niepoprawną treść opowiadania 6. Kliknij przycisk “Publish”
<i>Oczekiwane wyniki</i>	Wyświetlenie informacji o niepoprawnej treści opowiadania
<i>Id.</i>	PT32
<i>Idea testowa</i>	Sprawdzenie poprawności filtrowania
<i>Dane testowe</i>	Wybrany gatunek z listy możliwych filtrów
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Przejdź do ekranu “Community” 2. Kliknij ikonę filtrowania lub napis “Filter” 3. Wybierz gatunek z listy możliwych filtrów
<i>Oczekiwane wyniki</i>	Zaprzestanie wyświetlania opowiadań z wybranego gatunku lub wyświetlanie opowiadań z wybranego gatunku w zależności od koloru przycisku (mniej kontrastowy – zaprzestanie wyświetlania opowiadań, bardziej kontrastowy – wyświetlanie opowiadań)
<i>Id.</i>	PT33
<i>Idea testowa</i>	Sprawdzenie poprawności wyszukiwania
<i>Dane testowe</i>	Wyszukiwana fraza
<i>Kroki testu</i>	<ol style="list-style-type: none"> 1. Kliknij ikonę lupy 2. W polu wyszukiwania podaj wyszukiwaną frazę
<i>Oczekiwane wyniki</i>	Wyświetlanie opowiadań zawierających wyszukiwaną frazę w tytule, treści lub nazwie autora

9. Wyniki i analiza badań

Na skuteczność klasyfikatora Naive Bayes wpływają następujące parametry: liczba słów dla każdego gatunku, na podstawie których klasyfikator dokonuje klasyfikacji opowiadań oraz maksymalna liczba powtórzeń słów, które są analizowane jako cecha gatunku – zbyt często występujące z opowiadaniach słowa nie mogą być stosowane do skutecznego określania gatunku opowiadań. Na skuteczność klasyfikatora Naive Bayes wpływa również jakość i ilość danych uczących. Klasyfikator Naive Bayes musi zostać dostrojony na drodze badań empirycznych. W tym celu podzielono dane uczące na dwa podzbiory – 80% danych jako zbiór treningowy oraz 20% jako zbiór walidacyjny – i testowano wartości parametrów. Jako zbiór danych uczących wykorzystano zbiór około 10000 opowiadań z portalu AO3 [4].

Opowiadania przypisywane są do jednego z ośmiu gatunków (science-fiction, horror, adventure, comedy, crime and mystery, fantasy, historical, romance). Dla ośmiu gatunków metoda losowa osiąga skuteczność 12,5%.

Dla pierwszego zestawu danych uczących uzyskano bardzo niską skuteczność klasyfikatora Naive Bayes – maksymalnie około 30% (liczba słów dla każdego gatunku, na podstawie których klasyfikator dokonuje klasyfikacji opowiadań: 300, maksymalna liczba powtórzeń słów, które są analizowane jako cecha gatunku: 100). Zdecydowano w pierwszej kolejności zwiększyć jakość zbioru danych uczących, a dopiero następnie dostosować wartości parametrów. Drugi zestaw danych uczących posiada opowiadania lepiej reprezentujące wybrane gatunki. Wyniki badań dla pierwszego zestawu danych uczących zaprezentowano na wykresie 1. Wykres na stronie w orientacji poziomej znajduje się w sekcji [Wykresy](#).



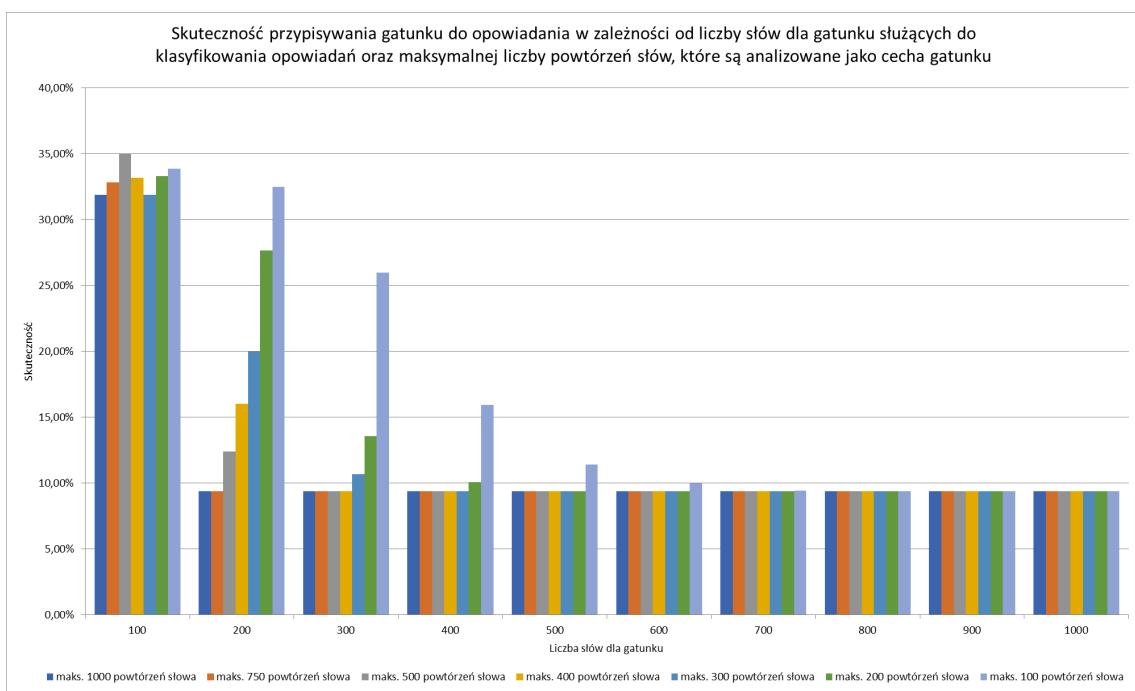
Wykres 1. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku – pierwszy zestaw danych uczących

Dla drugiego zestawu danych uczących uzyskano nieco wyższą skuteczność klasyfikatora Naive Bayes, z uwagi na sam fakt większej jakości zestawu danych uczących. Dostrojono więc klasyfikator Naive Bayes drogą empiryczną.

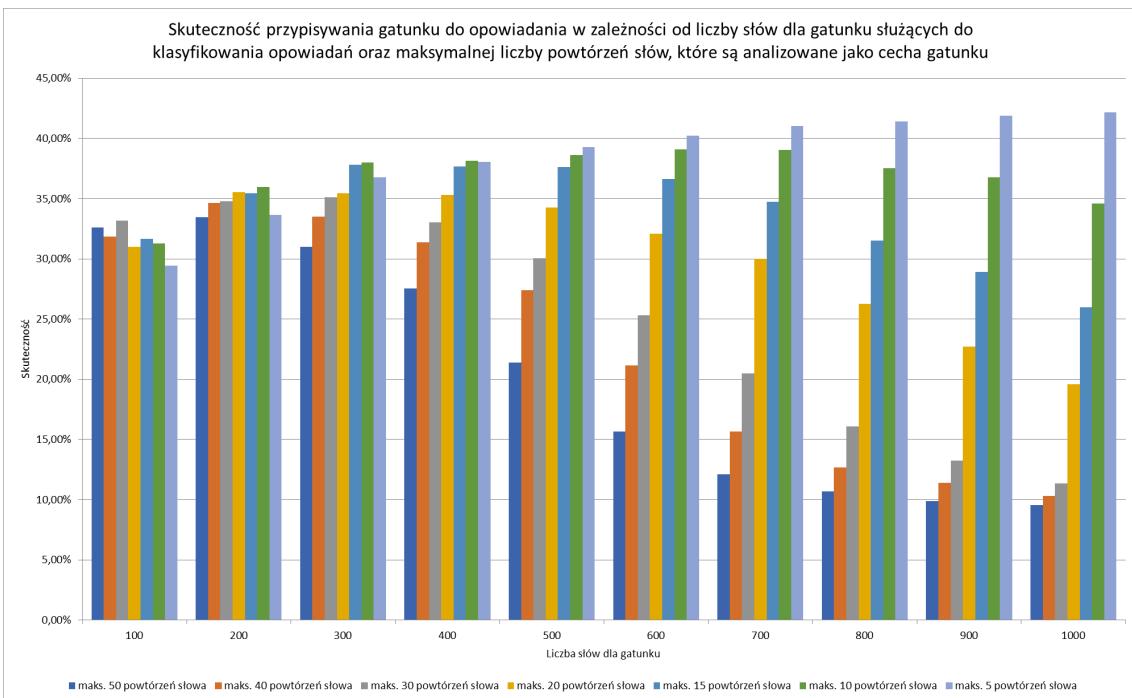
Przetestowano następujące wartości parametrów:

- liczba słów dla każdego gatunku, na podstawie których klasyfikator dokonuje klasyfikacji opowiadań: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 2500, 3000
- maksymalna liczba powtórzeń słów, które są analizowane jako cecha gatunku: 1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 50, 100, 200, 300, 400, 500, 750, 1000

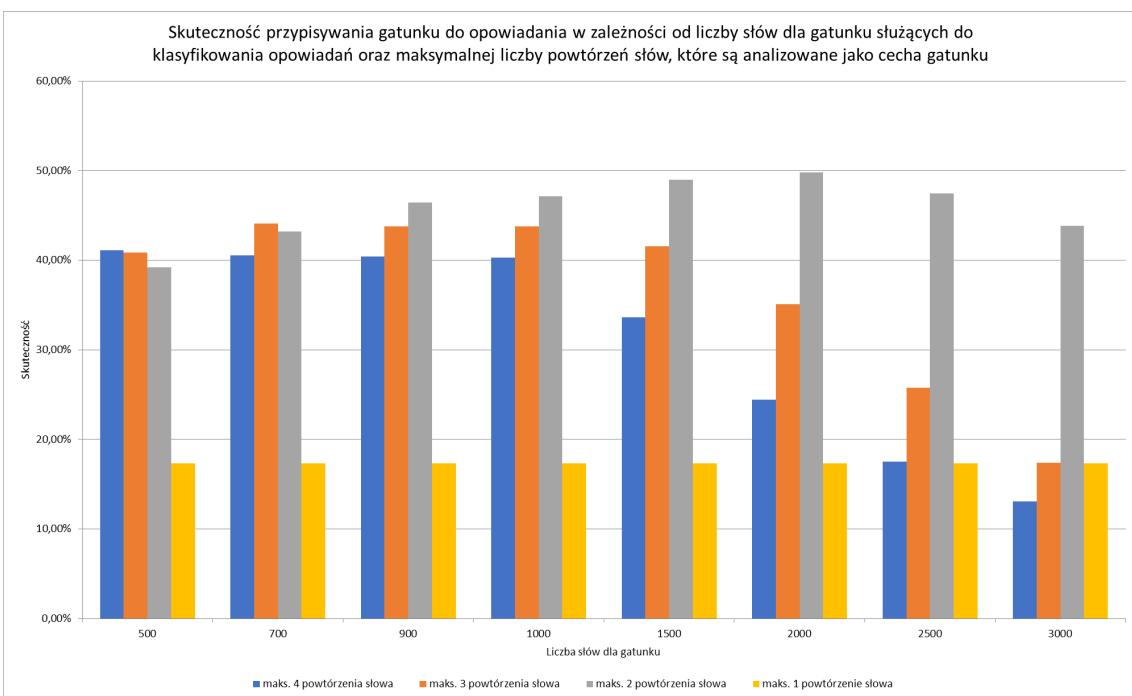
Uzyskano maksymalną skuteczność klasyfikatora około 50% (liczba słów dla każdego gatunku, na podstawie których klasyfikator dokonuje klasyfikacji opowiadań: 2000, maksymalna liczba powtórzeń słów, które są analizowane jako cecha gatunku: 2). Dla maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku o wartości 2 przetestowano więcej wartości drugiego parametru: 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300, 2400, 2500. Wyniki badań dla drugiego zestawu danych uczących zaprezentowano na wykresach 2 – 5. Wykresy na stronie w orientacji poziomej znajdują się w sekcji [Wykresy](#).



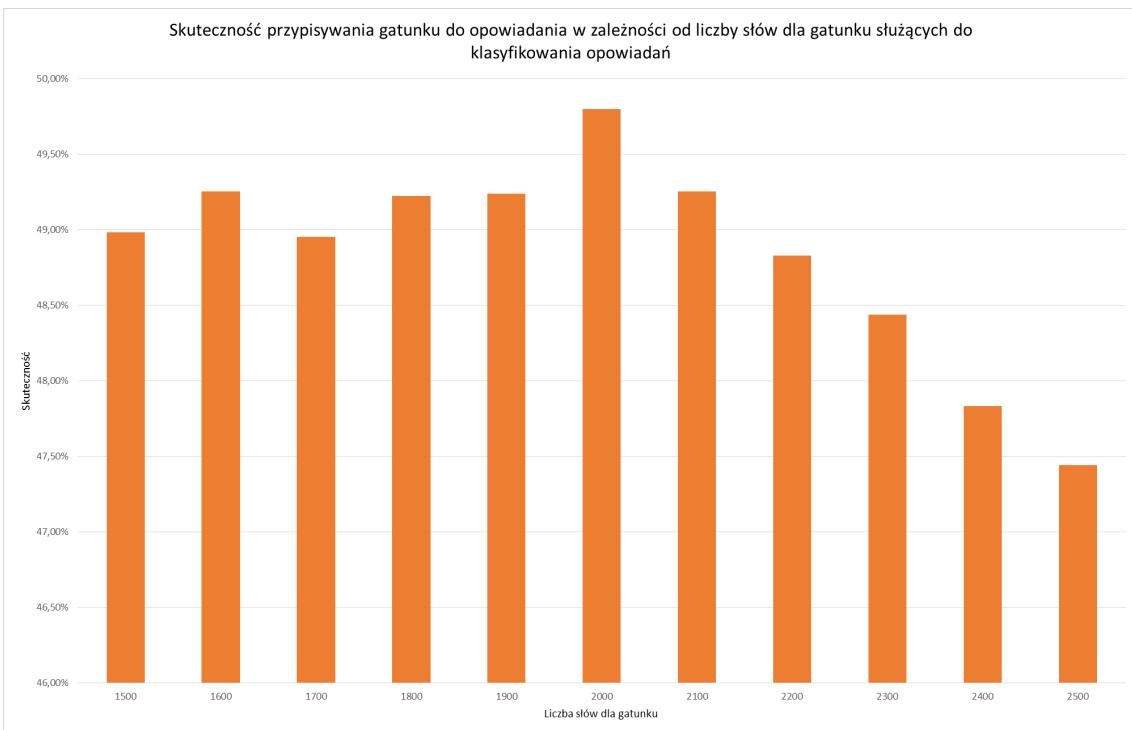
Wykres 2. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku – drugi zestaw danych uczących, maksymalne liczby powtórzeń słów: 1000, 750, 500, 400, 300, 200, 100



Wykres 3. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku – drugi zestaw danych uczących, maksymalne liczby powtórzeń słów: 50, 40, 30, 20, 15, 10, 5



Wykres 4. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku – drugi zestaw danych uczących, maksymalne liczby powtórzeń słów: 4, 3, 2, 1



Wykres 5. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań – drugi zestaw danych uczących, maksymalna liczba powtórzeń słów, które są analizowane jako cecha gatunku: 2

10. Podsumowanie

Główym celem zespołowego przedsięwzięcia inżynierskiego było stworzenie aplikacji mobilnej, która zachęca użytkowników do kreatywnego podejścia do pisania tekstów literackich oraz jest przestrzenią, w której użytkownicy o podobnych zainteresowaniach mogą rozwijać swoje umiejętności, dzielić się swoimi pomysłami oraz opiniami. Cel ten udało się zrealizować. W ramach przedsięwzięcia zaprojektowano i zaimplementowano aplikację mobilną natywną na systemy operacyjne Android. Zrealizowano wszystkie zakładane funkcjonalności aplikacji oraz przebadano zaprojektowane interfejsy na grupie docelowej. Do działania aplikacja mobilna wymaga zewnętrznej bazy danych, którą zaprojektowano oraz następnie wdrożono w chmurze Heroku [15]. Do komunikacji z bazą danych wykorzystano API napisane w Pythonie oraz w mikro-frameworku Flask, również wdrożone w chmurze Heroku [15].

Aplikację stworzoną w ramach przedsięwzięcia od konkurencji na rynku odróżnia wykorzystanie elementów sztucznej inteligencji do analizy i generowania tekstów. Moduł analizy tekstów pisanych przez użytkowników wykorzystuje lematyzację do określenia, ile słów zachęty zostało wykorzystanych oraz klasyfikację opowiadań do gatunków literackich za pomocą klasyfikatora Naive Bayes, który jest metodą sztucznej inteligencji. Moduł generowania tekstów wykorzystuje modele sztucznej inteligencji CBART [7] oraz GPT-Neo [13][37] do generowania opowiadań. Wygenerowane opowiadania oraz wygenerowana analiza opowiadań pisanych przez użytkownika są dostępne do oglądu przez użytkowników, a także są wykorzystywane w minigrze.

Elementem badawczym przedsięwzięcia było odpowiednie przygotowanie danych uczących oraz dostrojenie klasyfikatora Naive Bayes. Zbiór danych uczących pozyskano z portalu archiveofourown.org [4]. Uzyskano maksymalną skuteczność klasyfikatora około 50% dla liczby słów dla każdego gatunku, na podstawie których klasyfikator dokonuje klasyfikacji opowiadań równej 2000 oraz maksymalnej liczby powtórzeń słów, które są analizowane równej 2. Dla gatunków rozpoznawanych przez zaimplementowany klasyfikator Naive Bayes metoda losowa osiąga skuteczność 12,5%.

Jako narzędzia wspomagające pracę zespołową wykorzystano kalendarz Google [38], narzędzie do spotkań online Zoom [39] oraz przede wszystkim narzędzie Jira [40]. Kalendarz Google wykorzystano do koordynacji i planowania spotkań w zespole, z uwagi na różne przedmioty wybieralne, na które zapisane są różne osoby z zespołu. Wszystkie spotkania dotyczące projektu odbywały się na żywo lub za pośrednictwem narzędzia Zoom. Narzędzie Jira wykorzystano do planowania zadań, planowania sprintów, określania statusu zadań oraz przypisywania zadań do wykonujących je członków zespołu. Z narzędzi Zoom oraz Jira korzystano w ramach licencji studenckiej udostępnianej przez Politechnikę Wrocławską.

DOKUMENTACJA UŻYTKOWNIKA

1. Wprowadzenie

Aplikacja wr(a)ite jest aplikacją mobilną umożliwiającą użytkownikowi rozwój literacki. Zachęca ona do regularnego rozwijania hobby pisarskiego poprzez sugerowanie wyzwań oraz możliwość rywalizacji z innymi. W ramach ciekawostki, aplikacja udostępnia użytkownikom analizę napisanych przez nich opowiadań oraz wygenerowane opowiadania, do czego wykorzystuje elementy sztucznej inteligencji.

Podstawową funkcjonalnością aplikacji jest realizacja przez użytkownika losowych wyzwań składających się z gatunku oraz listy słów. Zadaniem użytkownika jest napisanie opowiadania w danym gatunku oraz zawierającego słowa z listy. Opublikowane teksty są analizowane i przetwarzane z wykorzystaniem intelligentnych algorytmów oraz są dostępne do przeczytania przez wszystkich użytkowników aplikacji. Użytkownicy mają możliwość oceny, komentowania oraz dodawania do ulubionych opowiadań innych użytkowników. Dodatkowo, aplikacja udostępnia minigrę, polegającą na zgadywaniu, które opowiadanie z pary, w której jedno zostało napisane przez użytkownika, a drugie wygenerowane przy użyciu intelligentnych algorytmów, otrzymało wyższą ocenę w wyniku analizy z wykorzystaniem elementów sztucznej inteligencji. Aplikacja umożliwia również personalizację swojego profilu oraz przeglądanie statystyk swojego użycia aplikacji.

2. Instalacja produktu programowego

2.1. Wymagania systemowe

System operacyjny: Android

Minimalna wersja systemu: 10 Q

Preferowana wersja systemu (zapewniająca najlepsze działanie aplikacji): 12 S

Dodatkowe wymagania:

- aplikacja do działania wymaga dostępu do Internetu,
- aplikacja jest dostępna tylko w wersji na telefon.

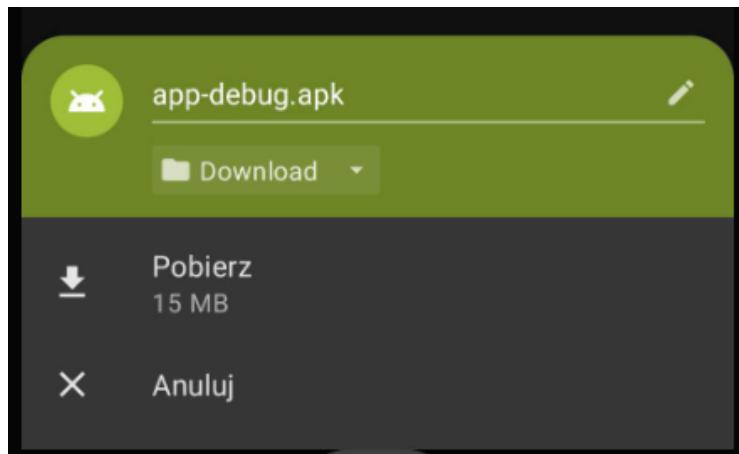
2.2. Opis procesu instalacji

Opis dotyczy telefonów posiadających Android 10 lub wyższy – mogą wystąpić drobne różnice w zależności od posiadanego urządzenia. Należy wtedy znaleźć odpowiednie informacje na stronie producenta pod hasłem “instalacja aplikacji z pliku apk” lub wyszukać w ustawieniach telefonu “Aplikacje ze specjalnym dostępem”.

Zainstalowanie aplikacji wymaga włączenia pozwolenia na instalację aplikacji ze źródeł innych niż Google Play Store.

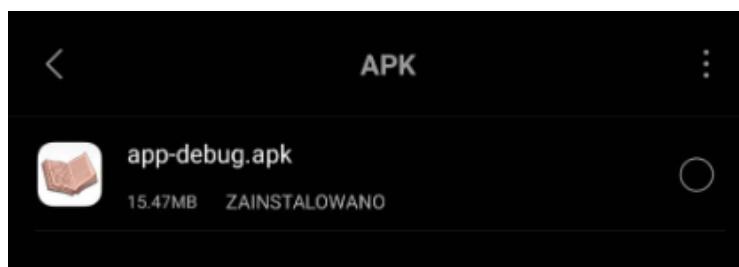
Proces instalacji:

1. Pobierz plik instalacyjny aplikacji z rozszerzeniem .apk:



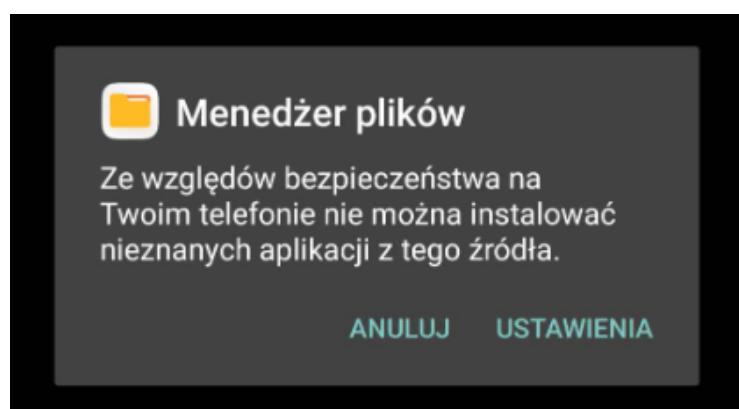
Screen 1. Widok pobierania aplikacji

2. Znajdź plik w managerze plików na urządzeniu:



Screen 2. Widok aplikacji w managerze plików

3. Po pojawienniu się komunikatu wybierz “USTAWIENIA”:



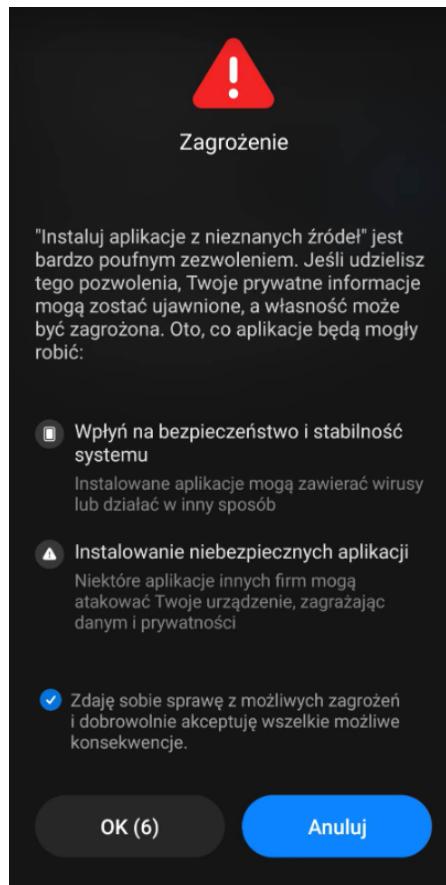
Screen 3. Widok komunikatu o braku uprawnień

4. Po uruchomieniu ekranu ustawień zaznacz wymagane uprawnienie “Zezwól z tego źródła”:



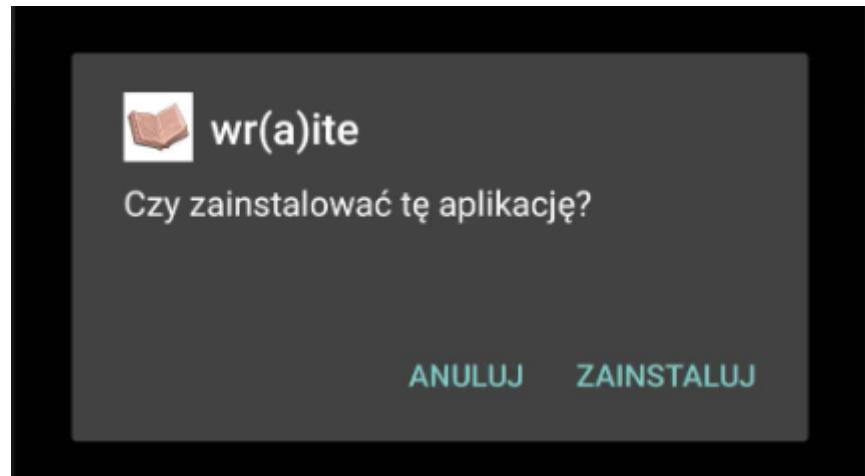
Screen 4. Widok wymaganych uprawnień w ustawieniach urządzenia

5. Zostanie wyświetcone ostrzeżenie, należy wybrać “OK”:



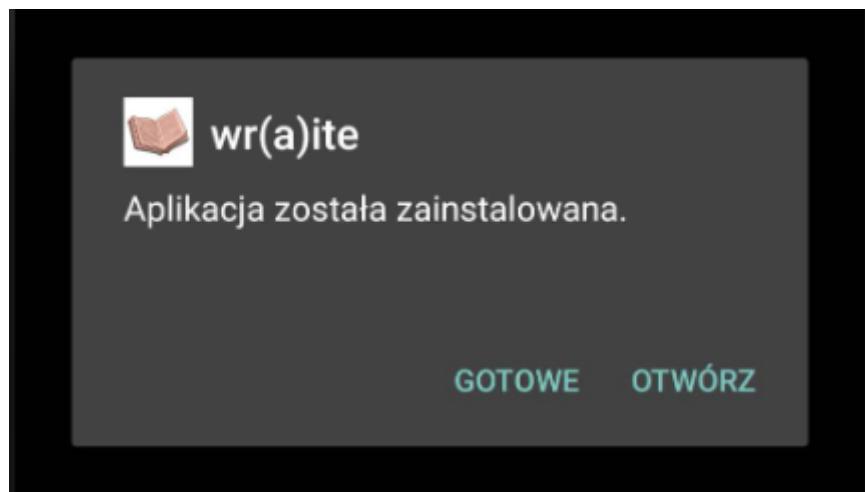
Screen 5. Widok ostrzeżenia przy próbie nadania uprawnień

6. Po zaznaczeniu wejdź ponownie w manager plików i wybierz pobraną aplikację, po pojawienniu się komunikatu wybierz “ZAINSTALUJ”:



Screen 6. Widok pytania o potwierdzenie chęci instalacji

7. Po zakończeniu instalacji i wyświetleniu komunikatu wybierz “OTWÓRZ” lub kliknij “GOTOWE” i wybierz aplikację z listy aplikacji:



Screen 7. Widok komunikatu o pomyślnej instalacji

2.3. Opis realizacji typowych zadań z podziałem na aktorów

Gość – osoba niezalogowana:

1. Rejestracja w systemie:
 - a. ekran rejestracji / logowania
 - b. przycisk “I don’t have an account yet”
 - c. przycisk “Sign up with Google” lub po uzupełnieniu danych (login, hasło, powtórzenie hasła oraz adres e-mail) przycisk “Register”
2. Logowanie
 - a. ekran rejestracji / logowania
 - b. przycisk “Sign in with Google” lub po uzupełnieniu danych (login oraz hasło) przycisk “Log in”

Użytkownik – osoba zalogowana:

1. Wylogowanie:
 - a. ekran profilu
 - b. przycisk “Log out”
2. Odzyskiwanie hasła:
 - c. ekran rejestracji / logowania
 - d. przycisk “I’ve forgotten my password” (nowe hasło zostanie wysłane na adres e-mail powiązany z kontem o podanym loginie)
3. Odzyskiwanie loginu:
 - a. ekran rejestracji / logowania,
 - b. przycisk “I’ve forgotten my login” (login zostanie wysłany na podany adres e-mail jeżeli istnieje konto powiązane z tym adresem e-mail)
4. Pisanie opowiadania:
 - a. ekran główny
 - b. przycisk ‘+’ → przyciski ‘1’, ‘3’, ‘5’ (wybór liczby słów, które należy użyć w opowiadaniu)
5. Przeglądanie napisanych opowiadań:
 - a. ekran główny
 - b. lista zawierająca napisane opowiadania
6. Wyświetlanie opowiadania:
 - a. ekran główny → wybór opowiadania z listy → ekran czytania opowiadania
7. Usunięcie opowiadania:
 - a. ekran czytania opowiadania
 - b. przycisk “Delete” (ikona kosza na śmieci opisana jako “Delete”)
8. Przeglądanie opowiadań innych użytkowników:
 - a. ekran społeczności
 - b. lista zawierająca opowiadania innych użytkowników
9. Wyświetlanie opowiadania innego użytkownika:
 - a. ekran społeczności → wybór opowiadania z listy → ekran czytania opowiadania
10. Filtrowanie opowiadań:
 - a. ekran społeczności
 - b. przycisk “Filter” (ikona filtrowania opisana jako “Filter”) → wybór gatunku

11. Wyszukiwanie opowiadań:
 - a. ekran główny / społeczności
 - b. przycisk wyszukiwania (ikona lupy)
 - c. wpisanie szukanej frazy
12. Ocena opowiadania:
 - a. ekran czytania opowiadania innego użytkownika
 - b. wybór odpowiedniej liczby gwiazdek
13. Dodanie komentarza:
 - a. ekran czytania opowiadania innego użytkownika
 - b. przycisk "Add comment"
14. Usunięcie komentarza:
 - a. ekran czytania opowiadania innego użytkownika
 - b. przycisk "Delete" w prawym górnym rogu komentarza (ikona kosza na śmieci opisana jako "Delete")
15. Dodanie do ulubionych:
 - a. ekran czytania opowiadania innego użytkownika
 - b. przycisk "Favourite" (ikona pustego serduszka opisana jako "Favourite")
16. Usunięcie z ulubionych:
 - a. ekran czytania opowiadania innego użytkownika
 - b. przycisk "Unfavourite" (ikona pełnego serduszka opisana jako "Unfavourite")
17. Przeglądanie ulubionych:
 - a. ekran profilu
 - b. przycisk "Favourites"
18. Wyświetlanie statystyk:
 - a. ekran profilu
 - b. przycisk "Statistics"
19. Zmiana zdjęcia profilowego:
 - a. ekran profilu
 - b. przycisk "Change profile picture"
20. Zmiana nazwy:
 - a. ekran profilu
 - b. przycisk "Change name"
21. Zmiana hasła:
 - a. ekran profilu
 - b. przycisk "Change password"
22. Zmiana adresu e-mail:
 - a. ekran profilu
 - b. przycisk "Change e-mail address"
23. Analiza tekstu:
 - a. ekran czytania opowiadania (wynik analizy jest dostępny po jej zakończeniu)
24. Generowanie tekstu:
 - a. ekran minigry – po wyborze gatunku (wynik dostępny po zakończeniu generacji)
 - b. ekran czytania opowiadania (wynik dostępny po zakończeniu generacji)
25. Gra w minigrę:
 - a. ekran minigry

Szczegółowy opis wybranych zadań:

- Pisanie opowiadania:
 - o zadanie polega na napisaniu opowiadania o długości max 3000 słów, które będzie zawierało wylosowane przez aplikację słowa oraz będzie należało do wylosowanego przez aplikację gatunku.
- Wyświetlanie statystyk:
 - o aplikacja zbiera dane na temat jej użytkowania,
 - o dostępne statystyki obejmują: liczbę opublikowanych historii, liczbę opublikowanych komentarzy, średnią długość opublikowanego opowiadania (wyrażoną w słowach), sumę słów w opublikowanych opowiadaniach, liczbę napisanych opowiadań z podziałem na gatunki, najlepszy "streak" (liczbę dni z rzędu, w których była używana aplikacja), liczbę punktów zdobytych w minigrze.
- Analiza tekstu:
 - o po napisaniu opowiadania zostaje ono przesłane do modułu sztucznej inteligencji, która dokonuje analizy pod względem: poprawności gramatycznej, wydźwięku opowiadania (pozytywne, negatywne, neutralne), bogactwa użytych słów, zgodności z wyzwaniem (wylosowanymi słowami i gatunkiem),
 - o na podstawie uzyskanych wyników zostanie obliczona ogólna ocena opowiadania.
- Gra w minigrę:
 - o minigra polega na ocenie, które z dwóch wyświetlonych opowiadań z wybranego gatunku (jedno napisane przez użytkownika, drugie wygenerowane przez moduł sztucznej inteligencji) uzyskało wyższy wynik w module analizy,
 - o za poprawną odpowiedź przyznawany jest 1pkt, za błędą odpowiedź przyznawane jest 0pkt, a jeżeli oba opowiadania z pary otrzymały taki sam wynik przyznawane jest 0,5 pkt,
 - o wyniki uzyskane w poszczególnych podejściach są sumowane i wyświetlane na ekranie minigry w formie rankingu użytkowników.

ANEKS

1. Kod projektu

Aplikacja mobilna

<https://github.com/AleksandraStecka/zpi-app>

REST API

<https://github.com/AgnieszkaKlobus12/zpi-api>

Moduł analizy tekstu

<https://github.com/AgnieszkaKlobus12/zpi-ai>

Moduł generowania tekstu

<https://github.com/AgnieszkaKlobus12/zpi-gai>

2. Źródła

[1] –

Ai grammar checker for English - apps on Google Play. Google. Accessed 22.11.2022. Available at: <https://play.google.com/store/apps/details?id=com.hellotalk.aigrammar&hl=en&gl=US&pli=1>

[2] –

Write your best with grammarly. Grammarly. Accessed 22.11.2022.

Available at: <https://www.grammarly.com/>

[3] –

Where stories live. Wattpad. Accessed 22.11.2022.

Available at: <https://www.wattpad.com/>

[4] –

Organization for Transformative Works. Archive of our own. Archive of Our Own. Accessed 22.11.2022.

Available at: <https://archiveofourown.org/>

[5] –

Plot generator. Plot Generator. Accessed 22.11.2022.

Available at: <https://www.plot-generator.org.uk>

[6] –

Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric Xing and Zhiting Hu. 2021. Progressive Generation of Long Text with Pretrained Language Models. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4313–4324, Online. Association for Computational Linguistics.

[7] –

Xingwei He. 2021. Parallel Refinements for Lexically Constrained Text Generation with BART. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8653–8666, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[8] –

Tianyi Tang, Junyi Li, Wayne Xin Zhao and Ji-rong Wen. 2022. MVP: Multi-task Supervised Pre-training for Natural Language Generation. *ArXiv* abs/2206.12131 (2022): n. pag.

[9] –

Pei Ke, Haozhe Ji, Yu Ran, Xin Cui, Liwei Wang, Linfeng Song, Xiaoyan Zhu and Minlie Huang. 2021. JointGT: Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2526–2538, Online. Association for Computational Linguistics.

- [10] –
 Bing He, Mustaque Ahamed and Srijan Kumar. 2021. PETGEN: Personalized Text Generation Attack on Deep Sequence Embedding-based Classification Models. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021): n. pag.
- [11] –
 Wanyu Du, Jianqiao Zhao, Liwei Wang and Yangfeng Ji. 2022 Diverse Text Generation via Variational Encoder-Decoder Models with Gaussian Process Priors. *ArXiv* abs/2204.01227 (2022): n. pag.
- [12] –
 Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.
- [13] –
 Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models* (2022): n. pag.
- [14] –
 Klaudia Nazarko. 2021. Practical text generation using GPT-2, LSTM and Markov chain. Accessed 26.10.2022. Available at: <https://towardsdatascience.com/text-generation-gpt-2-lstm-markov-chain-9ea371820ele>
- [15] –
 Heroku. Cloud Application Platform. Accessed 29.11.2022. Available at: <https://www.heroku.com/>
- [16] –
 Free design tool: Presentations, video, social media | CANVA. Canva. Accessed 26.10.2022. Available at: <https://www.canva.com/>
- [17] –
 cersin7. Montecristo color palette. Colox Hex. Accessed 26.10.2022. Available at: <https://www.color-hex.com/color-palette/1019071>
- [18] –
 Spyros Zevelakis and Eben Sorkin. 2022. Alata Font – Google Fonts. Free Fonts Vault. Accessed 26.10.2022. Available at: <https://freefontsvault.com/alata-font-download-free/>
- [19] –
 Christian Robertson. 2022. Roboto font family – Google Fonts. Free Fonts Vault. Accessed 26.10.2022. Available at: <https://freefontsvault.com/roboto-font-download-free/>
- [20] –
 Material symbols and icons – Google Fonts. Google. Accessed 26.10.2022. Available at: <https://fonts.google.com/icons?icon.style=Rounded&icon.set=Material%2BIIcons>
- [21] –
 Sign-in branding guidelines | Google Identity | Google Developers. Google. 2020. Accessed 26.10.2022. Available at: <https://developers.google.com/identity/branding-guidelines>
- [22] –
 pikisuperstar. 2021. Free vector: Hand drawn profile icons pack. Freepik. Accessed 26.10.2022. Available at: https://www.freepik.com/free-vector/hand-drawn-profile-icons-pack_17787078.htm
- [23] –
 pikisuperstar. 2021. Free vector: Hand drawn profile icons pack. Freepik. Accessed 26.10.2022. Available at: https://www.freepik.com/free-vector/hand-drawn-profile-icons-pack_17787077.htm
- [24] –
 Simon Kemp. 2022. Digital 2022: Global Overview Report - DataReportal – Global Digital Insights. DataReportal. Accessed 29.10.2022. Available at: <https://datareportal.com/reports/digital-2022-global-overview-report>

[25] –

Package Index | Android Developers. Android Developers. Accessed 13.11.2022.

Available at: <https://developer.android.com/reference/com/google/android/material/classes>

[26] –

Maven Repository: Com.squareup.retrofit2 » retrofit. Accessed 14.11.2022.

Available at: <https://mvnrepository.com/artifact/com.squareup.retrofit2/retrofit>

[27] –

Maven Repository: Com.squareup.retrofit2 » converter-gson. Accessed 14.11.2022.

Available at: <https://mvnrepository.com/artifact/com.squareup.retrofit2/converter-gson>

[28] –

Google cloud platform. Google. Accessed 15.11.2022.

Available at: <https://console.developers.google.com/apis/credentials>

[29] –

JavaMail. Accessed 15.11.2022.

Available at: <https://javaee.github.io/javamail/>

[30] –

Vinayak Bevinakatti. 09.01.2010. Sending email in Android using JavaMail API without using the default/built-in App. Stack Overflow. Accessed 15.11.2022.

Available at: <https://stackoverflow.com/a/2033124>

[31] –

Welcome to the Apache Software Foundation! Accessed 15.11.2022.

Available at: <http://www.apache.org/licenses/LICENSE-2.0>

[32] –

Armindo Flores, Armindoflores/ao3_api: An unofficial archiveofourown.org (AO3) API for python. GitHub. Published 11.02.2019. Last updated 26.10.2022. Accessed 22.11.2022.

Available at: https://github.com/ArmindoFlores/ao3_api

[33] –

Page Scholar Inc. JSpell Checker - Spelling & Grammar. Accessed 16.11.2022.

Available at: <https://www.jspell.com/>

[34] –

Page Scholar Inc. JSpell Checker Api Documentation: Rapidapi. Rapid. Accessed 16.11.2022.

Available at: <https://rapidapi.com/page-scholar-inc-page-scholar-inc-default/api/jspell-checker>

[35] –

fyhao. Text sentiment analysis method API documentation: Rapidapi. Rapid. Accessed 17.11.2022. Available at:

<https://rapidapi.com/fyhao/api/text-sentiment-analysis-method>

[36] –

NLTK. Accessed 17.11.2022.

Available at: <https://www.nltk.org/>

[37] –

FrostAura/GPT-neo-1.3B-fiction-novel-generation – hugging face. Hugging Face. Published 20.08.2022.

Last updated 26.10.2022. Accessed 27.11.2022.

Available at: <https://huggingface.co/FrostAura/gpt-neo-1.3B-fiction-novel-generation>

[38] –

Google calendar. Google. Accessed 29.11.2022. Available at: <https://calendar.google.com/>

[39] –

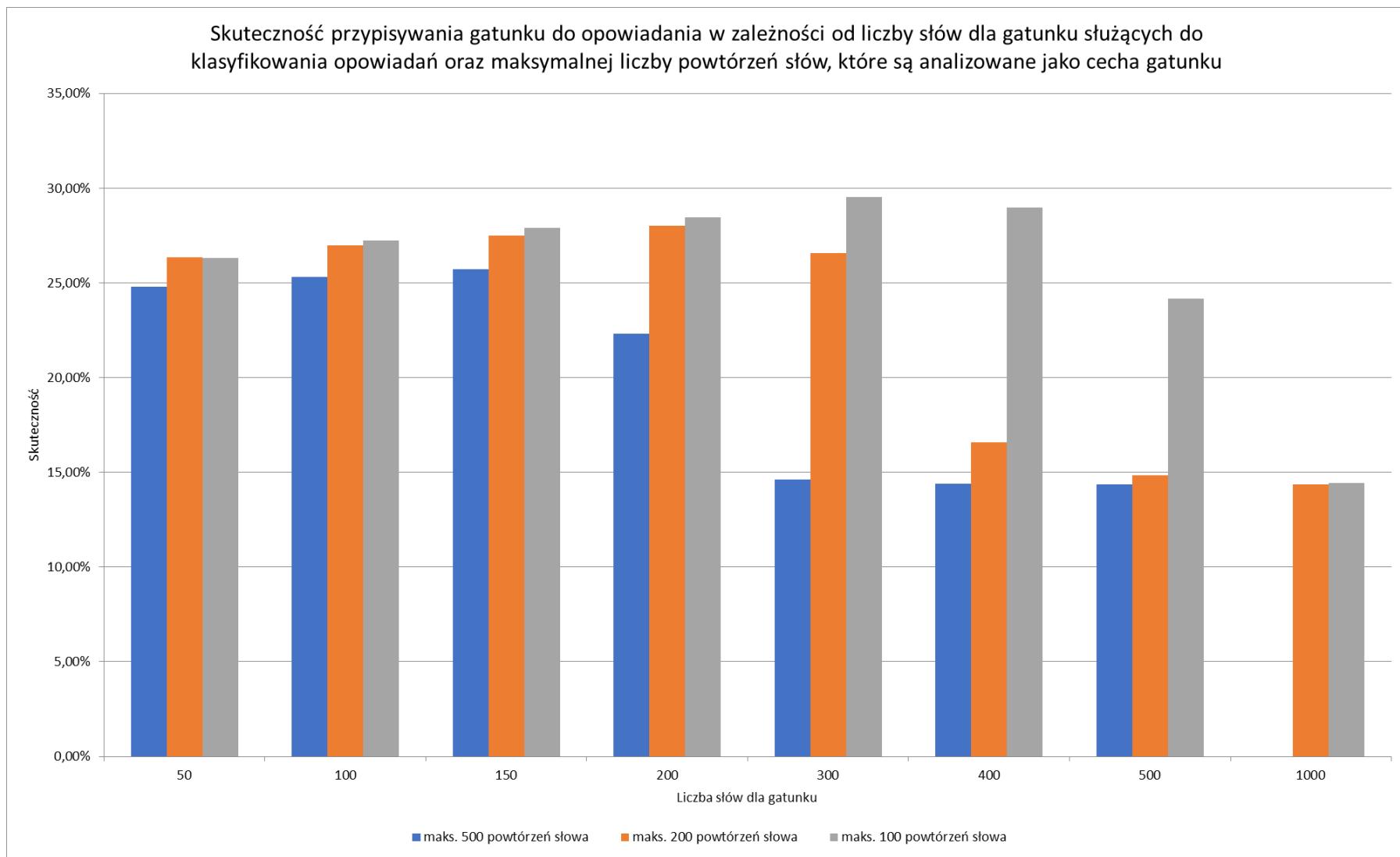
One platform to connect. Zoom. Accessed 29.11.2022. Available at: <https://zoom.us/>

[40] –

Jira: Issue & project tracking software. Atlassian. Accessed 29.11.2022.

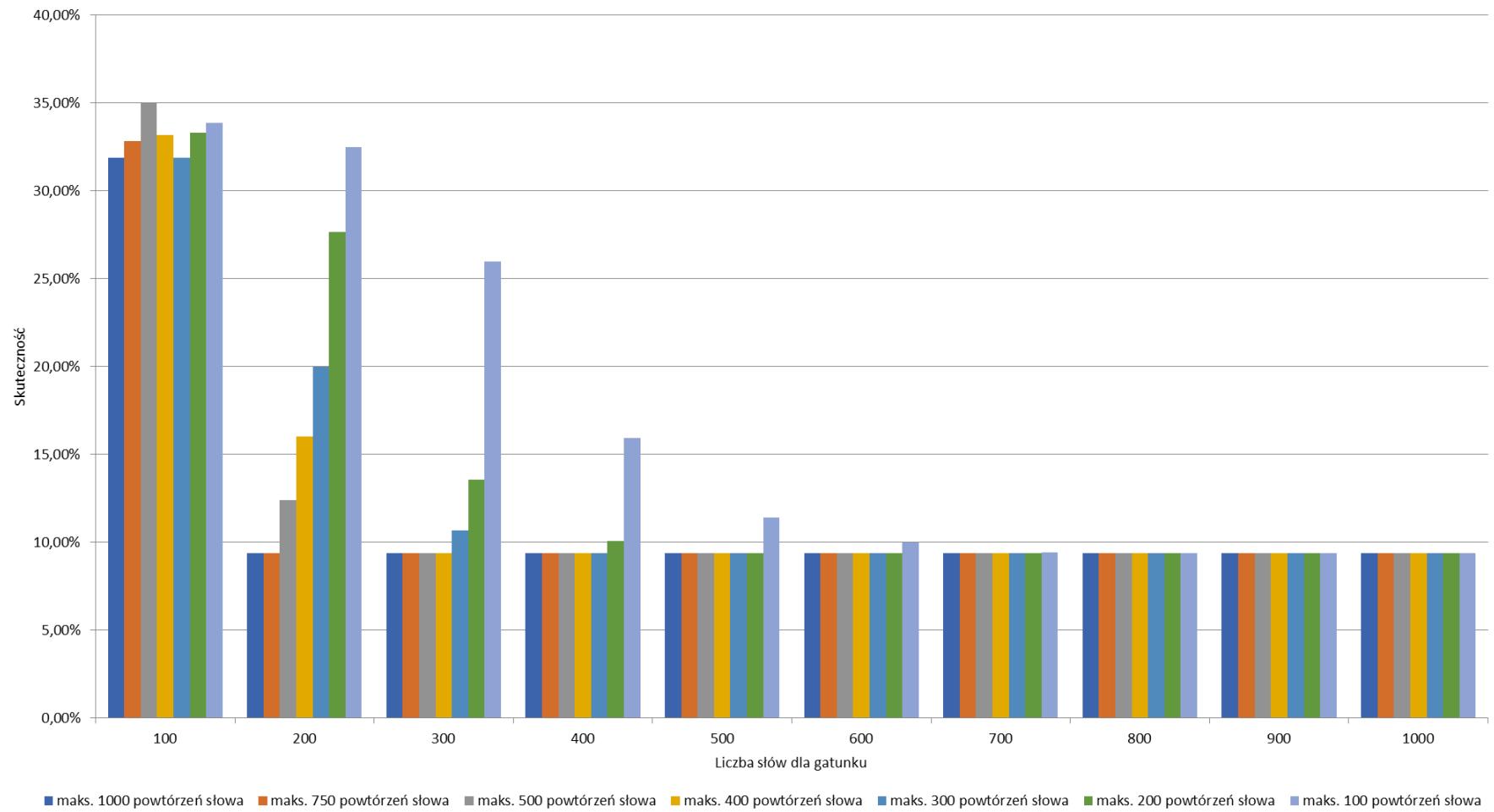
Available at: <https://www.atlassian.com/software/jira>

WYKRESY

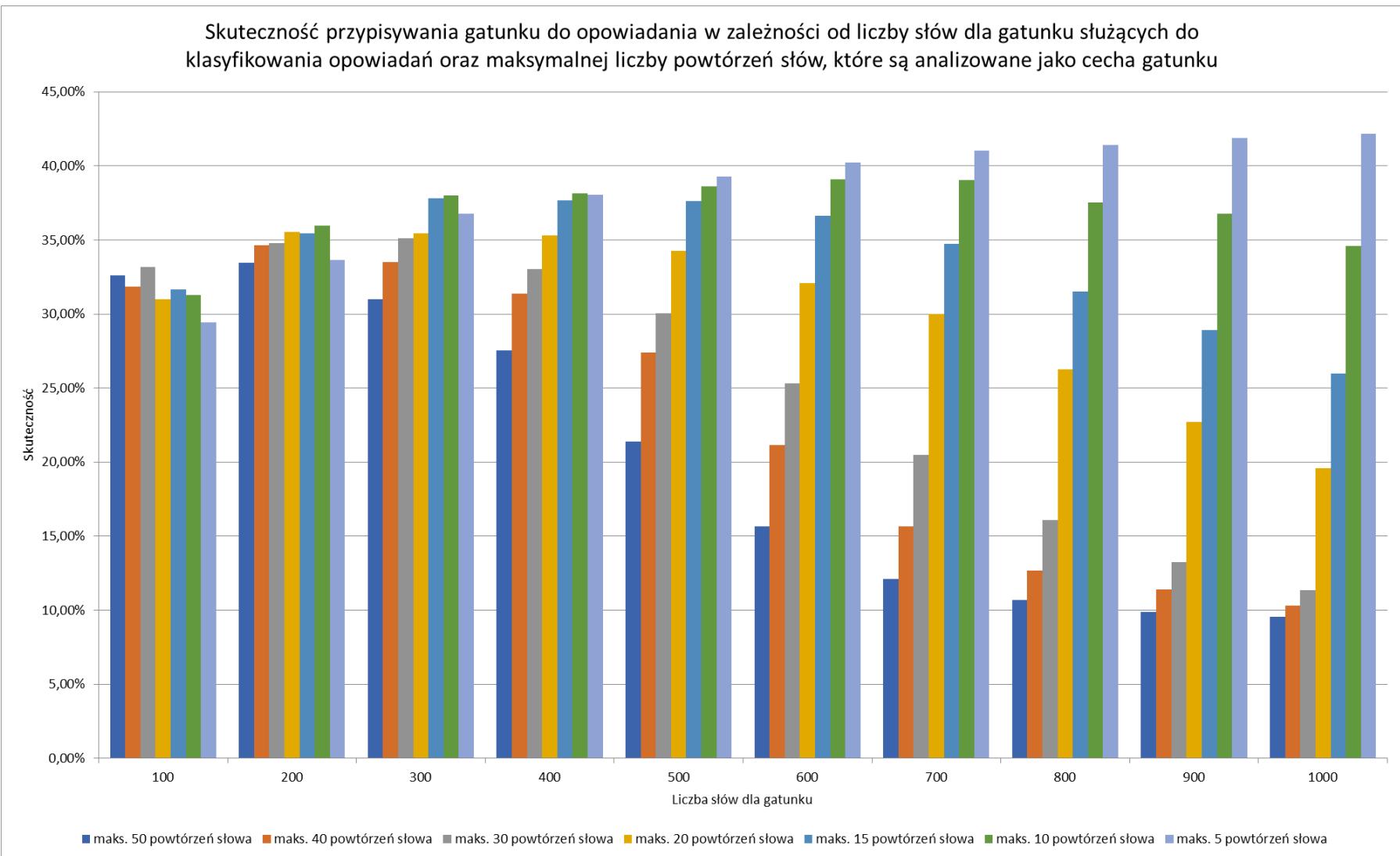


Wykres 1. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku – pierwszy zestaw danych uczących

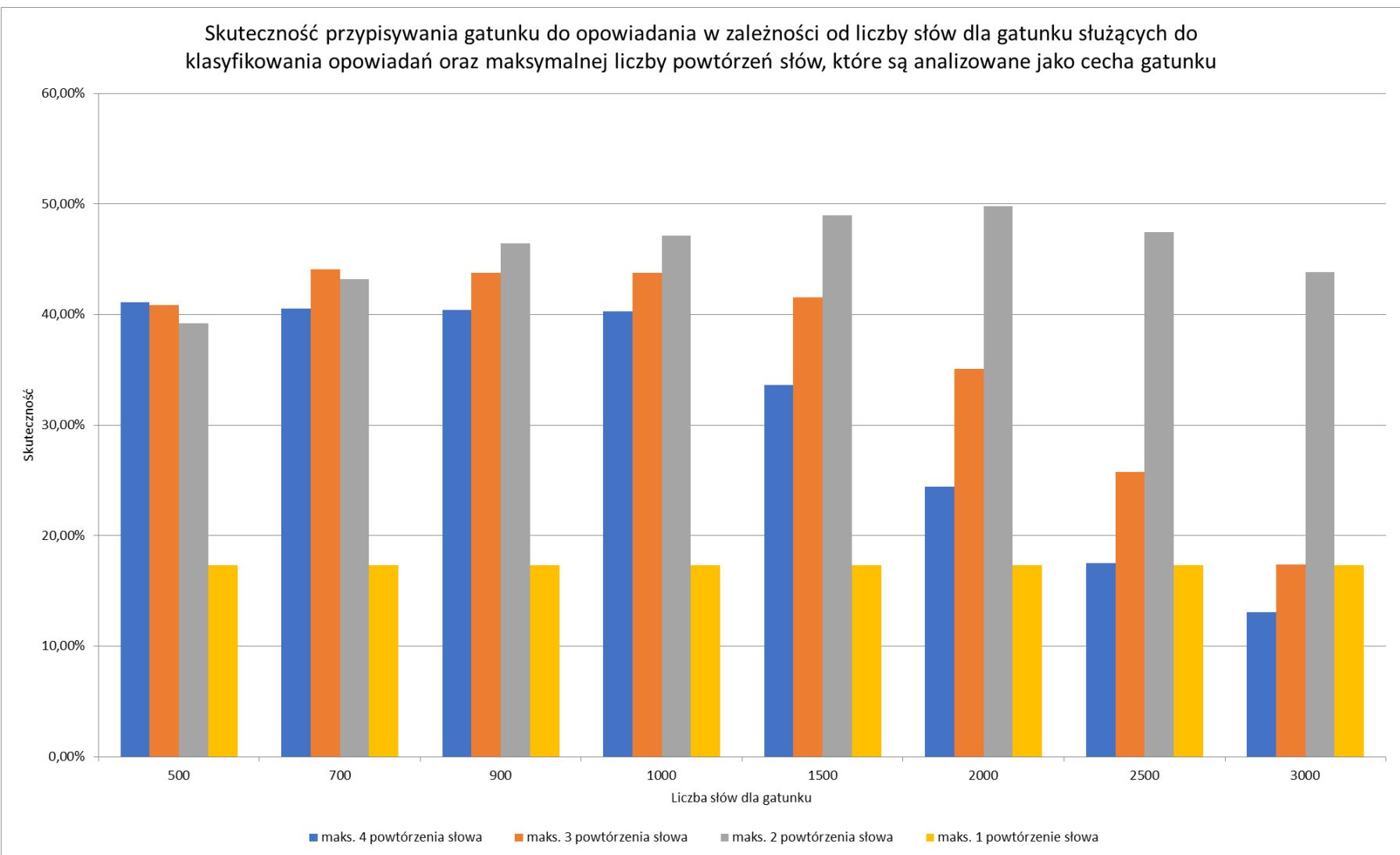
Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku



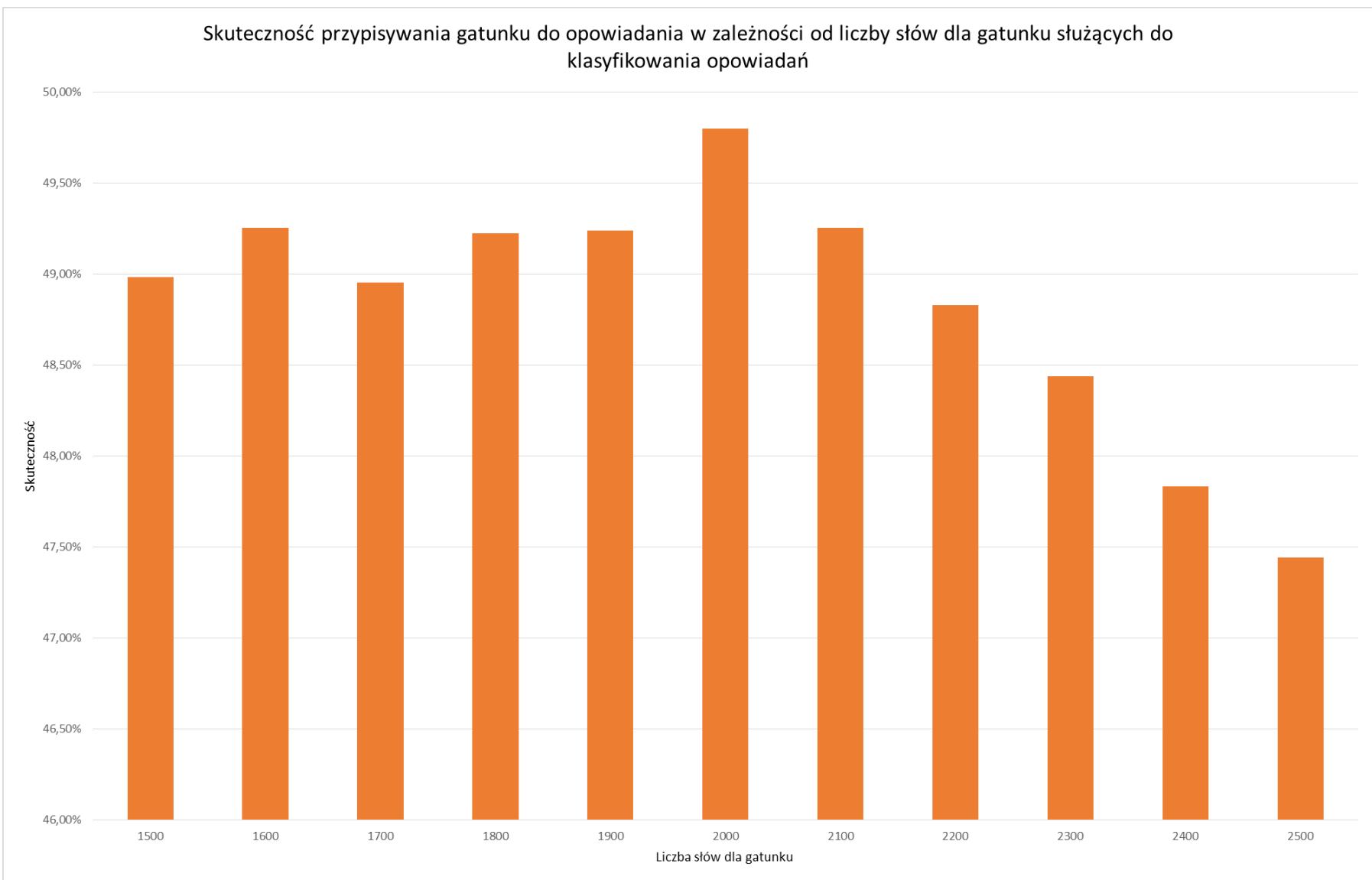
Wykres 2. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku – drugi zestaw danych uczących, maksymalne liczby powtórzeń słów: 1000, 750, 500, 400, 300, 200, 100



Wykres 3. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku – drugi zestaw danych uczących, maksymalne liczby powtórzeń słów: 50, 40, 30, 20, 15, 10, 5



Wykres 4. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań oraz maksymalnej liczby powtórzeń słów, które są analizowane jako cecha gatunku – drugi zestaw danych uczących, maksymalne liczby powtórzeń słów: 4, 3, 2, 1



Wykres 5. Skuteczność przypisywania gatunku do opowiadania w zależności od liczby słów dla gatunku służących do klasyfikowania opowiadań – drugi zestaw danych uczących, maksymalna liczba powtórzeń słów, które są analizowane jako cecha gatunku: 2