Dokument planu testów dla projektu "Video-Sent"

1 Plan Testów

Przykładowa struktura testów:

1. Dziennik zmian

Modyfikujący	**Wykonane** zmiany	**Data**	**Komentarz**
Jakub Cendalski Julia Ruszer Kacper Przybylski Kacper Witek Kacper Stasiak Dawid Frontczak	Stworzenie dokumentu	22.10.2025	Finalna wersja
Julia Ruszer	Przypadki Testowe	2.12.2025	Dodanie przypadków testowych

Tabela 1. Dziennik zmian

2. Wstęp

Niniejszy dokument przedstawia plan testów dla aplikacji Video-Sent składającej się z części UI oraz API. Zgodnie z wymaganiami, aktywnością związaną z realizacją przedmiotu jest dokument przedstawiający plan testów dla aplikacji, co prezentuje niniejszy dokument.

W kolejnych rozdziałach dokumentu opisano wyniki prac:

- Teoretycznych, mających określić kontekst testowania oraz ryzyka z niego wynikające
- Koncepcyjnych, mających określić optymalne podejście do testowania aplikacji Video-Sent

3. Terminologia i definicje

Termin/skrót	Opis
UI	Interfejs użytkownika, część systemu z która wchodzi w interakcje użytkownik
API	interfejs umożliwiający komunikację między aplikacją a ui
Analiza sentymentu	Technika przetwarzania języka, polegająca na rozpoznawaniu emocji i opinii.

Tabela 2. Wyjaśnienie użytej w dokumencie

4. Zakres dokumentu

Niniejszy dokument opisuje strategię oraz organizację procesu testowego dla aplikacji webowej “Video-sent”, obejmującego testy jednostkowe i integracyjne oraz funkcjonalne i niefunkcjonalne.

5. Opis testowanego systemu

Aplikacja Video-Sent polega na analizy sentymentu recenzji telefonów komórkowych (z podziałem na cechy, tzn. bateria, wyświetlacz itd.). Użytkownik ma możliwość wpisania linku z platform takich jak YouTube, TikTok, Instagram i dostaje wynik w postaci wyników poszczególnych cech.

Główne moduły systemu to:

- Transkrypcji
- Pobierania danych
- Analizy sentymentu
- Bazy danych
- Wizualizacji
- Wejściowy (interfejs użytkownika)

6. Zakres planowanych testów

W ramach planu, przewidziane są testy dla aplikacji Vide-Sent na poziomie jednostkowym oraz integracyjnym. Zostaną wykorzystane następujące typy testów: funkcjonalne, niefunkcjonalne.

7. Założenia i ograniczenia

Założenia:

- Projekt „Video-Sent" realizuje 6-osobowa grupa w ramach zajęć *Projekt Kompetencyjny*.
- Celem jest stworzenie prototypu systemu analizującego recenzje wideo telefonów z YouTube, TikToka i Instagrama.
- System wykorzystuje gotowe narzędzia i modele AI do transkrypcji i analizy sentymentu.
- Projekt tworzony jest w języku Python (backend) i React (frontend).
- Aplikacja działa w środowisku testowym, lokalnym oraz konteneryzowanym.

Ograniczenia:

- Ograniczony czas realizacji (jeden semestr).
- Modele AI mogą mieć błędy w analizie (szczególnie w języku polskim).
- Zależność od zewnętrznych narzędzi i modeli AI

8. Interesariusze i komunikacja

Interesariuszami dla systemu Video-Sent są:

- Julia Ruszer
- Jakub Cendalski
- Kacper Przybylski
- Kacper Stasiak
- Kacper Witek
- Dawid Frontczak

Komunikacja w ramach realizacji zadania odbywać się będzie za pośrednictwem kanału komunikacyjnego ***Discord***, w formie cotygodniowych spotkań zespołu developerskiego.

9. Strategia testowania

Testy zostaną przeprowadzone zgodnie z podejściem zorientowanym na ryzyko i wymagania. Testy są projektowane w oparciu o wymagania funkcjonalne, specyfikacje API oraz przypadku użycia systemu i będą one realizowane w dedykowanym środowisku testowym z użyciem odpowiednich narzędzi.

Klasyfikacja defektów:

Poziom krytyczności	Opis
Krytyczny	Błąd uniemożliwia działanie systemu lub kluczowej funkcji.
Wysoki	Błąd istotnie wpływa na działanie funkcji, istnieje obejście, ale utrudnia pracę
Średni	Błąd wpływa na mniej istotne funkcje lub powoduje błędy w danych pomocniczych
Niski	Błąd kosmetyczny, nie wpływa na funkcjonalność

Klasyfikacja testów:

Każdy przypadek testowy zostanie przypisany do jednej z kategorii priorytetu

Priorytet testu	Opis
Wysoki	Testuje kluczową funkcjonalność systemu lub ścieżkę krytyczną dla użytkownika
Średni	Testuje funkcjonalność często używaną, ale niekrytyczną
Niski	Testuje funkcję o niskim znaczeniu biznesowym lub rzadko używaną.

Warunkiem akceptacji oznaczamy gdy:

a. Brak defektów o poziomie krytycznym, bądź wysokim
b. Minimum 60% przypadków testowych zakończonych pozytywnie.
c. Defekty średnie i niskie muszą być zapisane.

Kategoria	**Opis**	**Przykład**
Wydajność testów	% wykonania przypadków testowych	91.6%
Skuteczność testów	% przypadków zakończonych pozytywnie	92.4%
Stabilność systemu	Średni czas naprawy defektu	1.5 dnia

10. Artefakty z procesu testowego

- Plan testów
- Raport z testów
- Testy aplikacji

11. Opis środowiska testowego

- Przeglądarka Chrome, Firefox
- System operacyjny Windows 11
- Docker

12. Harmonogram testowania

Proces testowania systemu odbywa się równolegle z pisaniem kodu aplikacji. Wynika z tego, że harmonogram testowania jest zgodny z harmonogramem pracy dla całego projektu.

13. Implementacja testów

Zaimplementowane zostaną testy funkcjonalne i niefunkcjonalne. Testy podzielone zostaną na cztery grupy.

Grupa testów	**Technologia**	L**okalizacja**	**Nazewnictwo**
Moduł transkrypcji	Pytest	backend/tests/transcription/	test_<nazwa_funkcji>.py
Moduł analizy sentymentu	Pytest	backend/tests/senticore/	test_<nazwa_funkcji>.py
Moduł pobierania danych	Pytest	backend/tests/download/	test_<nazwa_funkcji>.py
UI	Cypress	frontend/tests/	<nazwa_funkcji>.cy.js

14. Przypadki testowe

ID	Scenariusz testowy	ID scen.	Cel	Priorytet	Warunki wstępne	Dane testowe	Oczekiwany wynik	Przebieg testu
TC_01	Inicjowanie nowej analiz	SC_INIT_01	Sprawdzenie poprawności zlecenia nowej analizy dla poprawnego URL.	Wysoki	System działa, baza dostępna.	URL: "https://youtube.com/watch?v=test1"	Kod 202 Accepted, status ‘pending’, utworzone zadanie w DB.	1. Wyślij POST /api/analysis z URL. 2. Sprawdź kod odpowiedzi. 3. Sprawdź rekord w bazie.
TC_02	Ponowne zlecenie analizy (cache)	SC_INIT_02	Weryfikacja, czy system zwraca istniejący wynik dla znanego URL zamiast tworzyć nowe zadani	Średni	Film o danym URL jest już w bazie.	URL: "https://youtube.com/watch?v=test2	Kod 200/202, status ‘complete’, ID > 0.	1. Dodaj film do DB. 2. Wyślij POST /api/analysis z tym samym URL. 3. Sprawdź status i ID.
TC_03	Konflikt analizy (zadanie w toku)	SC_INIT_03	Sprawdzenie, czy system zwraca ID aktywnego zadania zamiast błędu przy dublowaniu żądania.	Średni	Zadanie dla URL jest w stanie ‘analyzing’.	URL: "https://youtube.com/watch?v=test3	Kod 200/202, ID istniejącego zadania.	1. Dodaj aktywne za- danie do DB. 2. Wyślij POST /api/analysis. 3. Porównaj zwrócone ID z ID w bazie.
TC_04	Sprawdzenie statusu	SC_STATUS_01	Weryfikacja pobierania statusu istniejącego zadania.	Wysoki	Zadanie istnieje w bazie.	ID zadania.	Kod 200, poprawny status (np. ‘pending’).	1. Wyślij GET /api/status/{id}. 2. Sprawdź JSON odpowiedzi.
TC_05	Sprawdzenie statusu (brak zadania)	SC_STATUS_02	Weryfikacja obsługi błędu dla nieistniejącego ID zadania.	Niski	Brak zadania o danym ID.	ID: 9999	Kod 404 Not Found.	1. Wyślij GET /api/status/9999. 2. Sprawdź kod odpowiedzi.

TC_06	Pobranie wyniku	SC_RESULT_01	Weryfikacja pobierania szczegółów analizy (sentymentu).	Wysoki	Film i analiza istnieją w bazie.	ID filmu.	Kod 200, obiekt filmu z danymi ‘analysis’.	1. Wyślij GET /api/result/{id}. 2. Sprawdź poprawność pól (battery, screen itp.).
TC_07	Pobieranie audio (sukces)	SC_SERV_01	Sprawdzenie logiki serwisu pobierania przy poprawnej odpowiedzi z yt-dlp.	Średni	Dostęp do biblioteki yt-dlp (mock).	URL wideo.	Zwrócony słownik ze ścieżką pliku.	1. Wywołaj download_audio(). 2. Sprawdź czy wynik nie jest None.
TC_08	Transkrypcja (sukces)	SC_SERV_02	Sprawdzenie logiki serwisu transkrypcji przy poprawnej odpowiedzi OpenAI.	Średni	Plik audio istnieje (mock), dostęp do API (mock).	Ścieżka pliku audio.	Zwrócony tekst transkrypcji.	1. Wywołaj transcribe(). 2. Sprawdź czy zwrócony tekst jest poprawny.
TC_09	Logika NLP (agregacja).	SC_NLP_01	Sprawdzenie mapowania słów kluczowych na oceny liczbowe.	Wysoki	Model NLP załadowany (mock).	Tekst: ,,Battery is great. Screen is bad."	Battery=1.0, Screen=-1.0	1. Wywołaj analyze_text_sync(). 2. Sprawdź wartości w słowniku wynikowym.
TC_10	Logika NLP (uśrednianie)	SC_NLP_02	Weryfikacja uśredniania sentymentu dla wielokrotnych wzmianek o aspekcie.	Wysoki	Model NLP załadowany (mock).	Tekst: ,,Battery ok. Battery great.”	Battery=0.5.	1. Wywołaj analyze_text_sync(). 2. Sprawdź wartość uśrednioną.
TC_11	NLP Integracja (recenzja iPhone)	SC_NLP_INT_01	Sprawdzenie sentymentu dla prawdziwej recenzji produkt	Wysoki	Prawdziwe modele NLP załadowane.	Tekst: Recenzja iPhone.	Poprawne wykrycie sentymentu (np. Screen=0.33, Memory=0.0).	1. Wywołaj analyze_text_sync(). 2. Sprawdź wartości dla kluczowych aspektów.

TC_12	NLP Integracja (negatywna recenzja)	SC_NLP_INT_02	Weryfikacja sentymentu dla ogólnie negatywnego tekstu.	Wysoki	Prawdziwe modele NLP załadowane.	Negatywna recenzja telefonu.	Niskie sentymenty dla Battery, Screen, Camera, Price (<- 0.5).	1. Wywołaj analyze_text_sync(). 2. Sprawdź, czy wartości są negatywne.
TC_13	NLP Integracja (pozytywna recenzja)	SC_NLP_INT_03	Weryfikacja sentymentu dla ogólnie pozytywnego tekstu.	Wysoki	Prawdziwe modele NLP załadowane.	Pozytywna recenzja telefonu.	Wysokie sentymenty dla Battery, Screen, Camera, Price (>0.0).	1. Wywołaj analyze_text_sync(). 2. Sprawdź, czy wartości są pozytywne.
TC_14	Pobieranie audio (błąd)	SC_SERV_03	Sprawdzenie obsługi błędu biblioteki yt-dlp.	Średni	Mock rzuca wyjątek.	Błędny URL.	Zwrócony None (obsłużony błąd).	1. Wywołaj download_audio() z błędnym URL. 2. Sprawdź wynik.
TC_15	Transkrypcja (brak pliku)	SC_SERV_04	Walidacja istnienia pliku przed wysłaniem do API.	Niski	Plik nie istnieje (mock).	Ścieżka do nieistniejącego pliku.	Rzucony wyjątek FileNotFoundError.	1. Wywołaj transcribe(). 2. Sprawdź czy rzucono wyjątek.
TC_16	Transkrypcja (błąd API)	SC_SERV_05	Obsługa błędu zewnętrznego API (OpenAI).	Średni	Mock API rzuca wyjątek.	Poprawny plik.	Wyjątek przekazany wyżej (do obsłużenia przez pipeline).	1. Wywołaj transcribe(). 2. Sprawdź treść wyjątku.
TC_17	Pobranie wyniku (brak)	SC_RESULT_02	Obsługa błędu 404 dla nieistniejącego wyniku.	Niski	Brak filmu o danym ID w bazie.	ID: 9999	Kod 404 Not Found.	1. Wyślij GET /api/result/9999. 2. Sprawdź kod odpowiedzi.