Opis wersji aplikacji  
  
Aplikacja E-Dziennik  
Fabian Szkudlarski, Michał Tiedemann

wersja 1.00

Spis treści

[1. Wstęp 2](#_gjdgxs)

[2. Opis projektu 2](#_30j0zll)

[3. Struktura organizacyjna projektu 2](#_1fob9te)

[4. Role i odpowiedzialność 3](#_3znysh7)

[5. Harmonogram projektu 3](#_2et92p0)

[6. Kosztorys projektu 4](#_tyjcwt)

[7. Zasoby projektu 4](#_32hioqz)

[8. Standardy i narzędzia w projekcie 4](#_1hmsyys)

[9. Procesy zarządzania](#_41mghml) 5

[9.1 Plan zarządzania konfiguracją](#_2grqrue) 5

[9.2 Plan zarządzania ryzykiem](#_vx1227) 5

[9.3 Plan zarządzania testami](#_3fwokq0) 5

[9.4 Plan zapewnienia jakości 6](#_1v1yuxt)

[10. Zasady dokumentowania projektu wraz z listą szablonów 7](#_4f1mdlm)

1. Wstęp

**E-dziennik** to nowoczesna aplikacja edukacyjna, która umożliwia użytkownikom:

a) **Nauczycielom**: Wystawianie ocen uczniom, zarządzanie frekwencją oraz śledzenie postępów uczniów.  
b) **Uczniom**: Sprawdzanie swoich ocen w czasie rzeczywistym oraz komunikowanie się z nauczycielami.  
d) **Administratorom**: Zarządzanie kontami użytkowników (uczniów, nauczycieli), aktualizowanie informacji szkolnych, a także monitorowanie statystyk oraz systemu oceniania w szkole.

2. Opis projektu   
Aplikacja oferuje zintegrowaną platformę do zarządzania wszystkimi kluczowymi aspektami edukacyjnymi w jednym miejscu, co znacząco ułatwia współpracę między uczniami i nauczycielami.

3. Struktura organizacyjna projektu

Zespół Backend: Fabian Szkudlarski, Michał Tiedemann

Zespół FrontEnd: Michał Tiedemann

#### Przepływ danych między zespołami

1. **Zespół Backend**:
   * Zajmuje się projektowaniem oraz implementacją logiki biznesowej aplikacji, tworzeniem modeli danych oraz połączeń z bazą danych.
   * Współpracuje z zespołem Frontend w celu integracji API z interfejsem użytkownika.
   * Dostarcza dokumentację API dla zespołu Frontend.
2. **Zespół Frontend**:
   * Odpowiada za tworzenie i utrzymanie interfejsu użytkownika.
   * Współpracuje z zespołem Backend w celu integracji interfejsu z API.
   * Zgłasza wymagania dotyczące API do zespołu Backend.

#### Zasady organizacji spotkań

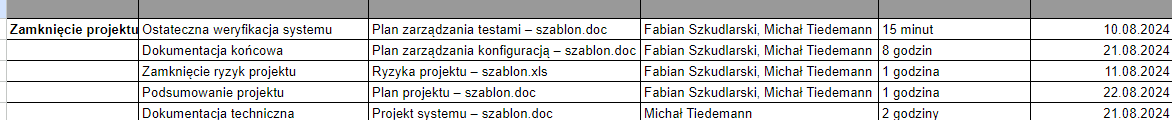
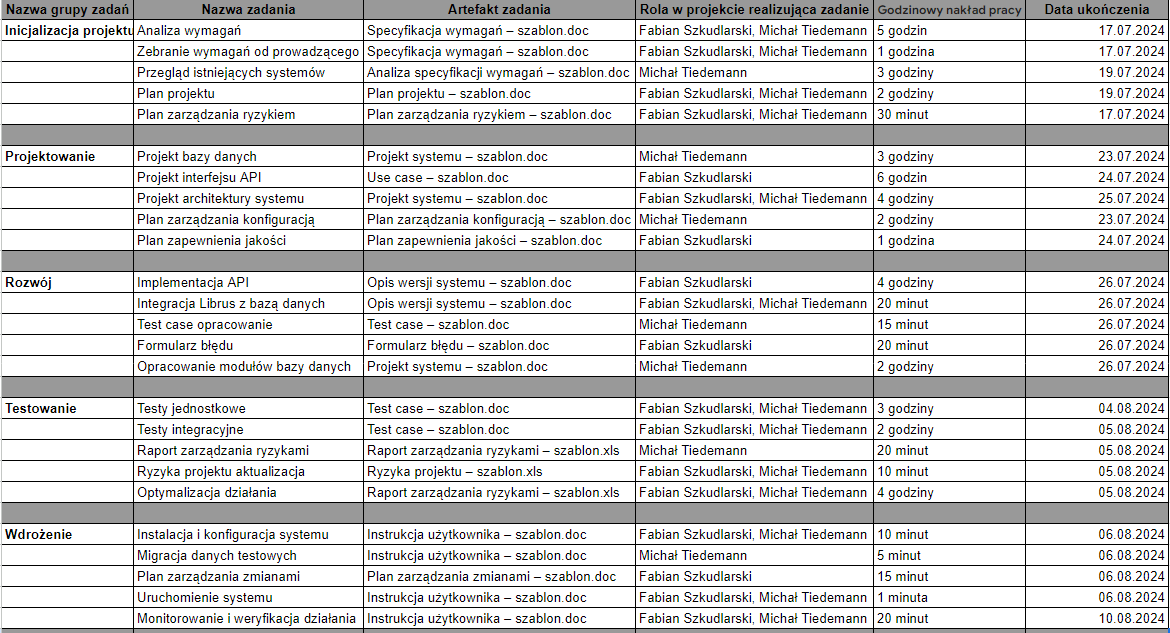
* Cotygodniowe spotkania statusowe z udziałem obu zespołów.
* Miesięczne retrospektywy i planowanie.

4. Role i odpowiedzialność

1. **Project Manager**:
   * Zarządzanie projektem, koordynacja zespołów.
   * Monitorowanie postępów, raportowanie do interesariuszy.
   * Rozwiązywanie problemów i zarządzanie ryzykiem.
2. **Lead Backend Developer** (Michał Tiedemann):
   * Projektowanie i implementacja backendowego WebAPI.
   * Nadzór nad pracą zespołu Backend.
   * Współpraca z zespołem Frontend.
3. **Backend Developer** (Fabian Szkudlarski):
   * Wspieranie implementacji WebAPI.
   * Pisanie kodu, testowanie, debugowanie.
4. **Lead Frontend Developer** (Michał Tiedemann):
   * Projektowanie i implementacja interfejsu użytkownika.
   * Nadzór nad pracą zespołu Frontend.
   * Współpraca z zespołem Backend.
5. **QA Specialist**:
   * Planowanie i przeprowadzanie testów.
   * Zarządzanie błędami, raportowanie wyników testów.

5. Harmonogram projektu

Harmonogram projektu został szczegółowo opisany w pliku “harmonogram projektu.xlsx”



6. Kosztorys projektu

| **Rola** | **Liczba godzin** | **Stawka godzinowa[PLN]** | **Koszt całkowity**  **[PLN]** |
| --- | --- | --- | --- |
| Project Manager | 100 | 100 | 8,000 |
| Lead Backend Dev | 200 | 120 | 20,000 |
| Backend Dev | 150 | 100 | 15,000 |
| Lead Frontend Dev | 200 | 120 | 25,000 |
| QA Specialist | 100 | 90 | 4,000 |
| **Łącznie** | **750** | **-** | **72,000** |

7. Zasoby projektu

 **Sprzęt**: Komputery, serwery.

 **Oprogramowanie**:

* IDE: IntelliJ IDEA (Java)
* Baza danych: MySQL.
* Systemy kontroli wersji: Git, GitHub.
* Licencje: IntelliJ IDEA Ultimate, MySQL Enterprise.

8. Standardy

 **Standardy kodowania**:

* Java: Google Java Style Guide.

9. Procesy zarządzania

9.1 Plan zarządzania konfiguracją

* **Narzędzie zarządzania konfiguracją**: Git, GitHub.
* **Zasady tagowania dokumentów**:
  + Nazewnictwo commitów: [type] short description (type: feat, fix, docs, style, refactor, test, chore).
  + Tagowanie wersji: vX.Y.Z (e.g., v1.0.0).

9.2 Plan zarządzania ryzykiem

 **Cele zarządzania ryzykiem**:

* Identyfikacja potencjalnych ryzyk na wczesnym etapie.
* Ocena i priorytetyzacja ryzyk.
* Planowanie i wdrażanie strategii mitygacyjnych.

 **Definicje pojęć**:

* **Ryzyko projektu**: Wydarzenie, które może negatywnie wpłynąć na realizację celów projektu.
* **Waga ryzyka**: Ocena potencjalnego wpływu ryzyka na projekt.
* **Stopień ryzyka**: Prawdopodobieństwo wystąpienia ryzyka pomnożone przez jego wagę.

 **Struktura bazy ryzyk**:

* Identyfikator ryzyka.
* Opis ryzyka.
* Prawdopodobieństwo wystąpienia.
* Waga ryzyka.
* Stopień ryzyka.
* Strategia mitygacyjna.
* Status ryzyka.

9.3 Plan zarządzania testami

 **Cele zarządzania testami**:

* Zapewnienie wysokiej jakości produktu końcowego.
* Wykrycie i naprawa błędów na wczesnym etapie.
* Weryfikacja zgodności produktu z wymaganiami.

 **Zasady projektowania testów**:

* Testowanie jednostkowe (Unit Testing).
* Testowanie integracyjne (Integration Testing).
* Testowanie end-to-end (E2E Testing).
* Testowanie wydajnościowe (Performance Testing).

 **Proces zarządzania błędami**:

* Wykrycie błędu.
* Zgłoszenie błędu w systemie zarządzania błędami (JIRA).
* Analiza błędu.
* Naprawa błędu.
* Retest.
* Zamknięcie zgłoszenia błędu.

 **Definicje i lista statusów błędu**:

* **New**: Nowy błąd, czeka na analizę.
* **In Progress**: Błąd jest naprawiany.
* **Fixed**: Błąd został naprawiony.
* **Retest**: Błąd czeka na ponowne testowanie.
* **Closed**: Błąd został rozwiązany i zamknięty.
* **Rejected**: Błąd został odrzucony jako nieważny.

 **Struktura bazy błędów**:

* Identyfikator błędu.
* Opis błędu.
* Status błędu.
* Priorytet błędu.
* Kto zgłosił.
* Kto naprawia.
* Data zgłoszenia.
* Data naprawy.

 **Cykle testowe**:

* Sprintowe testowanie jednostkowe i integracyjne.
* Testowanie regresji na koniec każdego sprintu.
* Testy końcowe przed wydaniem wersji produkcyjnej.

9.4 Plan zapewnienia jakości

 Role QA w procesie wytwarzania oprogramowania

* QA Manager: Odpowiedzialny za nadzór nad procesami QA, zarządzanie zespołem QA, planowanie i realizację zadań QA
* QA Engineer: Przeprowadza testy zgodnie z planem testów, raportuje błędy, współpracuje z developerami w celu rozwiązania wykrytych błędów.

 **Metodologie QA**:

* Testowanie ręczne
* Testy jednostokowe
* Testy systemowe
* Testy wydajnościowe
* Testy bezpieczeństwa

 **Elementy podlegające kontroli QA**:

* Kod źródłowy
* Wymagania
* Przypadki testowe.

 **Lista dokumentów Tworzonych przez QA**

* Plan testów
* Przypadki testowe
* Raporty testów
* Rejestr błędów

· **Zakres odpowiedzialności osób realizujących zadania QA**:

* QA Manager
  + Opracowanie i utrzymanie planu QA
  + Nadzór nad realizacją zadań QA
  + Raportowanie stanu jakości projektu
* QA Engineer
  + Tworzenie test case-ów
  + Wykonywanie testów ręcznych
  + Raportowanie i zarządzanie błędami

10. Zasady dokumentowania projektu wraz z listą szablonów

* Plan projektu
  + Wypełnia kierownik projektu (Project Manager).
  + Zawartość: Cele projektu, harmonogram, zasoby, budżet.
* Harmonogram projektu
  + Wypełnia kierownik projektu
  + Zawartość: Szczegółowy plan czasowy dla wszystkich zadań i kamieni milowych.
* Plan zarządzania ryzykiem
  + Wypełnia kierownik projektu wraz z zespołem ds. ryzyka.
  + Zawartość: Identyfikacja ryzyk, analiza ryzyka, strategie zarządzania ryzykiem.
* Plan ryzyka projektu
  + Wypełnia kierownik projektu wraz z zespołem ds. ryzyka.
  + Szczegółowa lista ryzyk, ocena ryzyk, plany łagodzenia ryzyka.
* Plan zapewnienia jakości
  + Wypełnia kierownik jakości (Quality manager).
  + Zawartość: Standardy jakości, procedury kontroli jakości.
* Plan zarządzania konfiguracją
  + Wypełnia kierownik konfiguracji (Configuration managment).
  + Zawartość: Procedury zarządzania konfiguracją, kontrola zmian, audyty konfiguracji.
* Plan zarządzania zmianami
  + Wypełnia kierownik projektu wraz z zespołem ds. zmian.
  + Zawartość: Procedury zarządzania zmianami, narzędzia do śledzenia zmian.
* Plan zarządzania testami
  + Wypełnia kierownik testów (Test manager).
  + Zawartość: Zakres testów, harmonogram testów, procedury testowe, zasoby testowe.
* Specyfikacja wymagań
  + Wypełnia analityk wymagań.
  + Zawartość: Wymagania funkcjonalne i niefunkcjonalne, diagramy usecase-ów.
* Analiza specyfikacji wymagań
  + Wypełnia analityk wymagań/zespół projektowy.
  + Zawartość: Przegląd wymagań, analiza wykonalności, priorytetyzacja wymagań.
* Use case
  + Wypełnia analityk wymagań wraz z zespołem projektowym.
  + Zawartość: Opis interakcji użytkowania z systemem.
* Projekt systemu
  + Wypełnia architekt systemu wraz z zespołem programistycznym.
  + Zawartość: architektura systemu, diagramy komponentów.
* Opis wersji systemu
  + Wypełnia kierownik konfiguracji wraz z zespołem programistycznym.
  + Zawartość: Opis wersji, zmiany wprowadzone w wersji.
* Test case
  + Wypełnia kierownik testów/testerzy.
  + Zawartość: Scenariusze testowe, kroki do wykonania, warunki początkowe, oczekiwane wyniki.
* Formularz błędu
  + Wypełnia testerzy wraz z zespołem programistycznym.
  + Zawartość: Opis błędu, piorytet błędu, status błędu.