WYDZIAŁ INŻYNIERII ELEKTRYCZNEJ I KOMPUTEROWEJ

POLITECHNIKA KRAKOWSKA

PRACA INŻYNIERSKA

Platforma do efektywnego zarządzania zajęciami pozaszkolnymi

Platform for effective management of extracurricular activities

Autor:

Bartosz SZARŁOWICZ

Promotor: **dr inż. Damian GRELA**

Kraków, 2025

Spis treści

Rozdział 1

Wprowadzanie

Niniejszy rozdział stanowi przedstawienie głównych założeń oraz uzasadnienie wyboru podjętej tematyki. Omówione zostaną wszelkie aspekty rozważanego problemu, stanowiące fundament dla dalszej części opracowania.

1.1 Streszczenie

Obecnie edukacja na różnych poziomach kształcenia coraz częściej wykracza poza standardowe zajęcia dydaktyczne przeprowadzane w czasie szkolnym. W odpowiedzi na rosnące wymagania rynku pracy, podnoszące się progi punktowe rekrutacji na uczelnie wyższe oraz potrzeby rozwoju intelektualnego, uczniowie decydują się na aktywne poszerzanie swojej wiedzy i umiejętności poza standardowym programem nauczania. W tym celu angażują swój wolny czas w dodatkowe zajęcia edukacyjne, takie jak korepetycje, warsztaty czy kursy tematyczne, które są prowadzone przez nauczycieli poza godzinami lekcyjnymi. Dla nauczycieli, tego rodzaju zajęcia stanowią sposób realizacji pasji zawodowych oraz formę dodatkowego zarobku. Dzięki takim inicjatywom realizowane jest indywidualne podejście do nauki oraz ucznia, co w rezultacie przekłada się na lepsze przygotowanie do przyszłych wyzwań edukacyjnych i zawodowych.

Niniejsza praca dyplomowa koncentruje się na opracowaniu systemu, który zoptymalizuje proces zarządzania dodatkowymi zajęciami dydaktycznymi. Aplikacja, będąca głównym efektem realizacji pracy, ma na celu ułatwienie wszystkich aspektów związanych z prowadzeniem nauczania pozaszkolnego. Kluczowymi funkcjonalnościami systemu są m.in. zarządzanie harmonogramem zajęć, gromadzenie indywidualnych danych związanych z procesem nauczania oraz usprawnienie komunikacji pomiędzy grupami docelowymi. Platforma zapewnia łatwy dostęp do wszystkich niezbędnych informacji a rozwiązania zastosowane podczas rozwoju systemu minimalizują czas poświęcany na kwestie organizacyjne i techniczne niwelującym przy tym zagrożenie utraty danych.

Dzięki zintegrowanym funkcjom, platforma ma potencjał znacząco poprawić organizację i efektywność zajęć pozaszkolnych oraz w rezultacie przekształcić się w kompleksowe narzędzie wspierające cały proces edukacyjny.

1.2 Motywacje

Głównym powodem wyboru tematyki pracy jest chęć stworzenia rozwiązania, które ma realny wpływ na optymalizacje i usprawnienie pracy grup docelowych poprzez wprowadzenie automatyzacji części procesów oraz zgromadzenie funkcjonalności w jednym systemie. Powody, które były determinantami do stworzenia narzędzia mającego na celu rozwiązanie zaobserwowanych problemów:

Rosnąca popularność zajęć pozaszkolnych

Ciągły wzrost zainteresowania różnorodnymi aktywnościami edukacyjnymi poza programem nauczania wymaga skutecznego narzędzia do ich organizowania i monitorowania. Uczniowie coraz częściej poszukują możliwości poszerzenia swojej wiedzy poza tradycyjnym programem nauczania. Powodem takich działań jest stale rosnący poziom konkurencji na rynku pracy oraz uczelniach wyższych.

Brak kompleksowych narzędzi na rynku

Dotychczasowe rozwiązania nie oferują wszystkich niezbędnych funkcji dostosowanych do potrzeb grup docelowych w jednym systemie. Istnieje potrzeba wykonania dedykowanego systemu w celu zapewniania kompleksowej obsługi dodatkowych zajęć pozaszkolnych.

Realna potrzeba optymalizacji pracy pozaszkolnej

Wiele aktywności związanych z pracą nauczyciela wymaga nie tylko przygotowania merytorycznego, ale także skrupulatnego zarządzania czasem, organizacją harmonogramu oraz gromadzeniem danych przebiegu kształcenia każdego prowadzonego ucznia. Zamknięcie wszystkich aspektów pozaszkolnej pracy nauczyciela w jednej aplikacji umożliwi sprawne zarządzanie dużą ilością zajęć.

Stały kontakt z grupą docelową

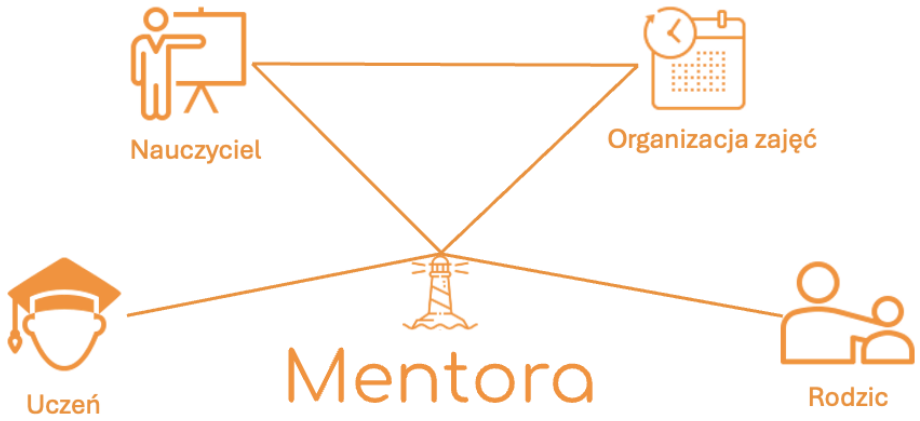
Możliwość wprowadzania najpotrzebniejszych funkcjonalności na podstawie feedbacku[1] od docelowej grupy początkowych użytkowników testowych (ang. early adopters) jeszcze w fazie deweloperskiej aplikacji. W wyniku przyjęcia takiej strategii system zawiera wszystkie kluczowe funkcje oraz jest odpowiedzią na priorytetowe potrzeby użytkowników.

[1] Feedback (pol. Informacja zwrotna)

1.3 Cel i zakres pracy

Celem pracy jest stworzenie kompleksowego narzędzia, które zoptymalizuje procesy związane z prowadzeniem dodatkowych form nauczania. Rozwijany system ma stanowić odpowiedzieć na oczekiwania grup docelowych za pomocą intuicyjnego interfejsu oraz wprowadzonych automatyzacji.

Wpływ na organizacje i przebieg dodatkowych zajęć pozaszkolnych mają trzy grupy: nauczyciele, uczniowie oraz rodzice. Kluczowym aspektem tworzonego systemu jest połączenie wszystkich przedstawicieli tych grup w jednym systemie, tak by mogli wspólnie korzystać ze wszystkich funkcjonalności oferowanych przez aplikacje, która nosi nazwę **Mentora**.



Rysunek 1.1: Połączenie najważniejszych aspektów systemu
źródło: opracowanie własne

Organizacja zajęć jest fundamentalną czynnością oraz główną funkcjonalnością systemu. Nauczyciel pełni rolę głównego organizatora i koordynatora zajęć pozaszkolnych. Właściwe zaplanowanie harmonogramu, dostosowanie godzin oraz rodzaju zajęć do dostępnych zasobów i potrzeb uczniów jest niezbędne dla zapewnienia płynności działań edukacyjnych. Zintegrowane funkcje systemu umożliwiają zautomatyzowaną organizacje zajęć, przez co cały proces zostaje znacznie uproszczony, dzięki czemu nauczyciel może skupić się na merytorycznym aspekcie nauczania.

Rozdział 2

Specyfikacja wymagań

Rozdział ten stanowi szczegółowy opis procesu tworzenia aplikacji, koncentrując się na specyfikacji wymagań funkcjonalnych i niefunkcjonalnych, wyborze odpowiednich technologii i narzędzi oraz przedstawieniu projektu bazy danych i architektury systemu. Przedstawione zostały diagramy i scenariusze przypadków użycia, a także istotne aspekty implementacji.

2.1 Wymagania funkcjonalne

Wymagania funkcjonalne opisują szczegółowo usługi jakie ma oferować system, sposób w jaki ma on reagować na określone dane wejściowe lub zdarzenia itp. […] [1]

Z metodycznego punktu widzenia, warto zauważyć, iż wymagania funkcjonalne definiowane na poziomie modelu architektury biznesowej powinny być opisywane w sposób oddający intencję (potrzebę) biznesu, abstrahując jednocześnie od sposobu ich realizacji. […] [1]

Wymagania funkcjonalne określają system informatyczny pod względem oferowanych możliwości, definiują jego funkcjonalności oraz obrazują interakcję użytkowników i innych systemów z tworzonym oprogramowaniem.

Wymagania funkcjonalne dla poszczególnych aktorów w systemie:

Nauczyciel

1. Zarządzanie harmonogramem zajęć pozaszkolnych
   1.1. CRUD[2] planów zajęć pozaszkolnych
   1.2. Wyświetlanie kalendarza zajęć w widokach dziennym, tygodniowym, miesięcznym
2. Zarządzanie kontami prowadzonych uczniów
   2.1. Możliwość tworzenia konta ucznia
   2.2. Możliwość uzupełniania raportu z danymi ucznia
3. Zarządzanie zasobami zajęć
   3.1. Możliwość załączania materiałów dydaktycznych
   3.2. Planowanie tematów zajęć

[2] CRUD – skrót opisujący podstawowe operacje wykonywane na danych w systemach informatycznych. (ang. Create, Read, Update, Delete)

4. Usprawnienie kontaktu i przepływu informacji
   4.1. Informowanie rodziców o planowanych zajęciach
   4.2. Możliwość prowadzenia konwersacji z uczniem

Uczeń

1. Dostęp do harmonogramu zajęć
   1.1. Dostęp do kalendarza zajęć
   1.2. Otrzymywanie powiadomień o zmianach w harmonogramie zajęć
2. Dostęp do materiałów
   2.1. Możliwość pobrania materiałów dydaktycznych przesłanych przez nauczyciela
   2.2. Możliwość załączenia własnych materiałów jako odpowiedź
3. Dostęp do konwersacji z nauczycielem

Nauczyciel, Uczeń (wspólne**):**

1. Rejestracja, logowanie, odzyskiwanie hasła
2. Edycja profilu użytkownika
3. Możliwość tworzenia prywatnych notatek

Rodzic

1. Mailowe powiadomienia dotyczące planowanych zajęć

2.2 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne opisują właściwości i ograniczenia systemu (np. niezawodność, czas odpowiedzi, zasady bezpieczeństwa itp.). W złożonych systemach często dochodzi do konfliktów pomiędzy wymaganiami funkcjonalnymi i niefunkcjonalnymi […] [1]

Wymagania niefunkcjonalne określają sposób działania systemu informatycznego, ściśle związane z bezpieczeństwem, wydajnością, użytecznością.

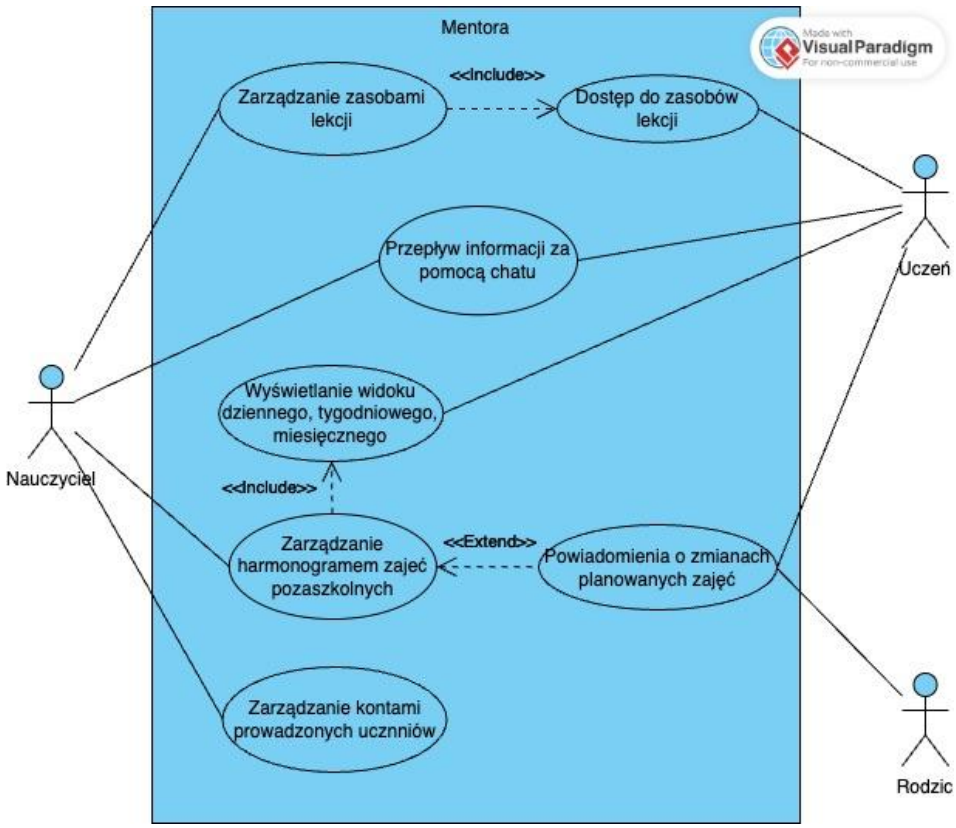
Wymagania niefunkcjonalne zdefiniowane dla projektu:

1. Bezpieczeństwo
   1.1. Przechowywanie zaszyfrowanych haseł
   1.2. Mailowa weryfikacja konta po rejestracji
   1.3. Automatycznie generowane tymczasowe hasło po założeniu konta ucznia przez nauczyciela
2. Użyteczność
   2.1. Intuicyjny interfejs umożliwiający szybkie wdrożenie do systemu
   2.2. Automatyczne zakładanie kont uczniów

2.3. Dostęp administratora do wszystkich niezaszyfrowanych danych i funkcjonalności
2.4. Widoki mobilne
3. Wydajność
3.1. Szybkie odpowiedzi na zapytania bazy, nieprzekraczające 0.5 sekundy
3.2. Obsługa asynchronicznych konwersacji użytkowników bez obciążania bazy danych
4. Niezawodność
4.1. Backup[3] bazy danych wykonywany raz dziennie
4.2. Aktualizacje serwera przeprowadzane w godzinach nocnych
5. Przenośność
5.1. System działa tak samo w różnych przeglądarkach (Google Chrome, Firefox, Safari itd.)

2.3 Diagram przypadków użycia

Przypadek użycia (ang. use case) to zbiór scenariuszy powiązanych ze sobą wspólnym celem użytkownika. Jest graficzną reprezentacją wymagań funkcjonalnych. Definiuje zachowanie systemu bez informowania o jego wewnętrznej strukturze i narzucania sposobu implementacji. […] [2]



Rysunek 2.1: Diagram przypadków użycia
źródło: https://online.visual-paradigm.com/pl/

[3] Backup (pol. kopia zapasowa) – proces tworzenia kopi bazy danych

2.4 Scenariusze przypadków użycia

Scenariusze przypadków użycia tworzone są w celu opisania przebiegu czynności wykonywanych przez aktorów systemu. Technika ta pozwala lepiej zrozumieć procesy zachodzące w systemie oraz zachowanie poszczególnych użytkowników.

Zarządzanie zasobami lekcji

Aktorzy	Nauczyciel
Zakres	System zarządzania zajęciami pozaszkolnymi
Udziałowcy i ich cele	1. Nauczyciel: Jego celem jest wprowadzanie lub modyfikowanie zasobów lekcji dla poszczególnych uczniów.
Zdarzenie wyzwalające (trigger)	Aktor (nauczyciel) wprowadza/modyfikuje zasoby lekcji
Warunki wstępne	Aktor musi być zalogowany do systemu
Warunki końcowe dla sukcesu	1. Wprowadzone/zmodyfikowane dane zostają poprawnie zapisane w systemie 2. Zaktualizowane dane są widoczne w systemie
Warunki końcowe dla niepowodzenia	1. Błąd walidacji wprowadzonych zasobów 2. Brak odpowiednich uprawnień 3. Komunikat o błędzie

Tabela 3.1: Scenariusze przypadków użycia – zarządzanie zasobami lekcji

Scenariusz główny

1. Nauczyciel loguje się do systemu
2. Nauczyciel wybiera zajęcia i przechodzi do panelu zarządzania lekcją
3. System wyświetla dane przypisane do wybranych zajęć
4. Nauczyciel dodaje/modyfikuje zasoby zajęć
5. Wprowadzone zasoby zostają poprawnie zweryfikowane
6. Zasoby zostają zapisane do systemu
7. System wyświetla komunikat o sukcesie procesu

Scenariusz alternatywny 1: Błąd zapisu danych

5a. Wprowadzone zasoby generują błąd walidacji systemu

5b. Dane nie zostają zapisane do systemu

5c. System wyświetla komunikat o błędzie procesu

Scenariusz alternatywny 2: Brak odpowiednich uprawnień

2a. Użytkownik bez odpowiednich uprawnień próbuje dostać się do panelu zarządzania

2b. System blokuje dostęp na podstawie braku uprawnień

Dostęp do zasobów lekcji

Aktorzy	Nauczyciel, uczeń
Zakres	System zarządzania zajęciami pozaszkolnymi
Udziałowcy i ich cele	1. Nauczyciel: Pełny dostęp do wszystkich zasobów lekcji 2. Uczeń: Pełny dostęp do wszystkich zasobów lekcji oraz możliwe załączanie plików w formie odpowiedzi
Zdarzenie wyzwalające (trigger)	Aktorzy (nauczyciel, uczeń) przechodzą do panelu zajęć
Warunki wstępne	Aktorzy muszą być zalogowani do systemu
Warunki końcowe dla sukcesu	1. Nauczyciel oraz uczeń mają swobodny do panelu zajęć 2. Uczeń ma możliwość załączania plików z odpowiedzią
Warunki końcowe dla niepowodzenia	1. Brak odpowiednich uprawnień dostępowych 2. Wprowadzone pliki przez ucznia generują błąd walidacji

Tabela 4.2: Scenariusze przypadków użycia – dostęp do zasobów lekcji

Scenariusz główny

1. Nauczyciel oraz uczeń logują się do systemu
2. Nauczyciel oraz uczeń mają pełny dostęp do zasobów zajęć
3. Uczeń załącza plik z odpowiedzią
4. Plik ucznia zostaje zapisany w systemie
5. Załączony plik jest dostępny w panelu zajęć dla ucznia i nauczyciela

Scenariusz alternatywny 1: Brak odpowiednich uprawnień

2a. Użytkownik bez odpowiednich uprawnień próbuje dostać się do panelu zajęć
2b. System blokuje dostęp na podstawie braku uprawnień

Scenariusz alternatywny 2: Wprowadzone pliki odpowiedzi ucznia generują błąd walidacji

4a. System generuje błąd walidacji plików wprowadzonych przez ucznia
4b. Komunikat o błędzie
4c. Ponowne wyświetlenie formularza

Zarządzanie harmonogramem zajęć pozaszkolnych

Aktorzy	Nauczyciel
Zakres	System zarządzania zajęciami pozaszkolnymi
Udziałowcy i ich cele	1. Nauczyciel: Jego celem jest planowanie lub modyfikowanie harmonogramu zajęć pozaszkolnych
Zdarzenie wyzwalające (trigger)	Aktor (nauczyciel) planuje/modyfikuje harmonogram zajęć
Warunki wstępne	Aktor musi być zalogowany do systemu
Warunki końcowe dla sukcesu	1. Zaplanowanie/zmodyfikowanie harmonogramu zajęć 2. Zaplanowane zajęcia zostają zapisane do systemu 3. Dostęp do panelu zajęć uzyskują przypisani do nich nauczyciel oraz uczeń
Warunki końcowe dla niepowodzenia	1. Błąd walidacji wprowadzonych zajęć 2. Komunikat o błędzie

Tabela 5.3: Scenariusze przypadków użycia – zarządzanie harmonogramem zajęć pozaszkolnych

Scenariusz główny

1. Nauczyciel loguje się do systemu
2. Nauczyciel przechodzi do widoku kalendarza i wybiera opcję modyfikuj/dodaj lekcję
3. Nauczyciel wypełnia formularz zajęć danymi m.in. dni tygodnia, czas trwania, przypisanie ucznia, rok szkolny, kolor kafelka
4. Formularz zostaje zaakceptowany
5. Zajęcia zostają dodane do systemu
6. Nauczyciel i uczeń uzyskują dostęp do panelu lekcji

Scenariusz alternatywny 1: Błąd zapisu danych

4a. Wprowadzone dane generują błąd walidacji systemu

4b. Zajęcia nie zostają zapisane do systemu

4c. System wyświetla komunikat o błędzie procesu

Wyświetlanie widoku dziennego, tygodniowego, miesięcznego harmonogramu

Aktorzy	Nauczyciel, uczeń
Zakres	System zarządzania zajęciami pozaszkolnymi
Udziałowcy i ich cele	1. Nauczyciel: Możliwość wyświetlania harmonogramu zajęć w widoku dziennym, tygodniowym, miesięcznym 2. Uczeń: Możliwość wyświetlania harmonogramu zajęć w widoku dziennym, tygodniowym, miesięcznym
Zdarzenie wyzwalające (trigger)	Aktorzy (nauczyciel, uczeń) wybierają rodzaj widoku harmonogramu
Warunki wstępne	Aktorzy muszą być zalogowani do systemu
Warunki końcowe dla sukcesu	1. Zalogowani aktorzy (uczeń, nauczyciel) przegląda harmonogram w wybranym formacie
Warunki końcowe dla niepowodzenia	1. Brak odpowiednich uprawnień dostępowych

Tabela 6.4: Scenariusze przypadków użycia – wyświetlanie widoku dziennego, tygodniowego, miesięcznego harmonogramu

Scenariusz główny

1. Uczeń/nauczyciel logują się do systemu
2. Przejście do zakładki kalendarz
3. Wybór rodzaju widoku harmonogramu

Scenariusz alternatywny 1: Brak odpowiednich uprawnień

2a. Użytkownik bez odpowiednich uprawnień próbuje dostać się do widoku harmonogramu
2b. System blokuje dostęp na podstawie braku uprawnień

Powiadomienia o zmianach planowanych zajęć

Aktorzy	Nauczyciel, uczeń, rodzic
Zakres	System zarządzania zajęciami pozaszkolnymi
Udziałowcy i ich cele	1. Nauczyciel: po utworzeniu zajęć przez nauczyciela system generuje powiadomienia 2. Uczeń: otrzymanie powiadomienia mailowego o utworzonych zajęciach 3. Rodzic: otrzymanie powiadomienia mailowego o utworzonych zajęciach
Zdarzenie wyzwalające (trigger)	Nauczyciel tworzy/modyfikuje zajęcia
Warunki wstępne	Adresy mailowe ucznia i rodzice muszą być zaakceptowane przez walidacje systemu
Warunki końcowe dla sukcesu	1. Uczeń i rodzic otrzymują powiadomienia maile na skrzynkę odbiorczą
Warunki końcowe dla niepowodzenia	1. Adresy mailowe ucznia/rodzica nie przeszły walidacji systemu

Tabela 7.5: Scenariusze przypadków użycia – powiadomienia o zmianach planowanych zajęć

Scenariusz główny

1. Nauczyciel z powodzeniem tworzy/modyfikuje zajęcia
2. System wysyła powiadomienie mailowe do przypisanego ucznia i rodzica

Scenariusz alternatywny 1: Adresy mailowe nie przeszły walidacji systemu

2a. System blokuje wysłanie powiadomień mailowych z powodu błędu walidacji adresów mailowych odbiorców

2b. System wyświetla komunikat o błędzie w panelu nauczyciela

Dodawanie kont prowadzonych uczniów

Aktorzy	Nauczyciel
Zakres	System zarządzania zajęciami pozaszkolnymi
Udziałowcy i ich cele	1. Nauczyciel: ma możliwość stworzenia konta ucznia
Zdarzenie wyzwalające (trigger)	Nauczyciel tworzy nowe konta ucznia
Warunki wstępne	Aktor musi być zalogowany do systemu
Warunki końcowe dla sukcesu	1. Konto ucznia zostaje dodane do systemu 2. Na wprowadzony adres mailowy ucznia zostaje wysłany link do potwierdzenia konta oraz tymczasowe hasło
Warunki końcowe dla niepowodzenia	1. Wprowadzony dane generują błąd walidacji systemu

Tabela 8.6: Scenariusze przypadków użycia –dodawanie kont prowadzonych uczniów

Scenariusz główny

1. Nauczyciel loguje się do systemu
2. Nauczyciel przechodzi do zakładki „Moi uczniowie”
3. Nauczyciel wybiera opcję dodaj ucznia
4. Wypełnienie formularza danymi ucznia
5. System przetwarza podany adres email
6. System nie znajduje adresu email w bazie danych, więc tworzy nowe konto
7. System wysyła mail na podany adres zawierający link aktywacyjny i tymczasowe hasło
8. System jest gotowy na logowanie ucznia, po zatwierdzeniu linku aktywacyjnego

Scenariusz alternatywny 1: Adres mailowy ucznia znajduje się w bazie danych

6a. System znajduje podany adres email, więc nie tworzy nowego konta ucznia

6b. System łączy konta nauczyciela i konto istniejącego ucznia

Scenariusz alternatywny 2: Wprowadzone dane generują błąd walidacji

4a. Wprowadzone dane przez nauczyciela generują błąd walidacji

4b. Komunikat o błędnych danych

4c. Ponowne wyświetlenie formularza

Przepływ informacji za pomocą chatu

Aktorzy	Nauczyciel, uczeń
Zakres	System zarządzania zajęciami pozaszkolnymi
Udziałowcy i ich cele	1. Nauczyciel: ma możliwość wysłania/odbierania wiadomości tekstowych do wybranego ucznia 2. Uczeń: ma możliwość wysłania/odbierania wiadomości tekstowych do wybranego nauczyciela
Zdarzenie wyzwalające (trigger)	Nauczyciel oraz uczeń wymieniają się wiadomościami tekstowymi za pomocą wbudowanego chatu
Warunki wstępne	Aktorzy muszą być zalogowani do systemu
Warunki końcowe dla sukcesu	1. Nauczyciel wysyła/odbiera wiadomości tekstowe 2. Uczeń wysyła/odbiera wiadomości tekstowe
Warunki końcowe dla niepowodzenia	1. Brak odpowiednich uprawnień dostępowych

Tabela 9.7: Scenariusze przypadków użycia – przepływ informacji za pomocą chatu

Scenariusz główny

1. Nauczyciel/uczeń logują się do systemu
2. Nauczyciel/uczeń przechodzą do panel zajęć
3. Po przejściu do panelu zajęć zostaje wyświetlony chat
4. Użytkownicy mają możliwość przeglądania historycznych wiadomości konwersacji
5. Po napisaniu wiadomości, zostaje ona wysłana do odbiorcy w czasie rzeczywistym
6. Odbiorca dostaje powiadomienie o nowej wiadomości

Scenariusz alternatywny 1: Brak odpowiednich uprawnień dostępowych

2a. Użytkownik bez odpowiednich uprawnień próbuje dostać się do widoku zajęć

2b. System blokuje dostęp na podstawie braku uprawnień

Rozdział 3

Analiza implementacji systemu

Rozdział ten poświęcony jest opisowi technologii wykorzystywanych podczas tworzenia platformy. Wybór narzędzi programistycznych to kluczowy aspekt budowy systemów informatycznych, który przekłada się na wszystkie etapy pracy oraz ostatecznie na jakoś, wydajność i sprawność produktu. Ta sekcja stanowi techniczną podstawę projektu, ilustrując w jaki sposób wykorzystane technologie wspierają realizację zamierzonych celów aplikacji.

3.1 Wybór technologii

Wybór odpowiedniej technologii stanowi istotną rolę, która wpływa na przebieg całego procesu tworzenia oprogramowania. Uwzględnienie wszystkich pozytywów i negatywów pozwoli na selekcje odpowiednich narzędzi, które są fundamentem tworzonego systemu, zapewniając jednocześnie jego stabilność i elastyczność w obliczu zmieniających się potrzeb. Właściwy dobór technologii może zatem decydować o sukcesie projektu, dlatego powinien być dokładnie przemyślany i oparty na solidnej analizie.

3.1.1 Frontend

Ruby On Rails – platforma programistyczna typu open source[4] tworząca w pełni wyposażone środowisko używane do tworzenia aplikacji internetowych. Jej celem jest ułatwienie procesu programowania aplikacji webowych poprzez przyjęcie założenia, że programista potrzebuje tylko podstawowych narzędzi do rozpoczęcia pracy. Rails pozwala na minimalizację ilości pisanego kodu przy jednoczesnym osiąganiu wysokiej funkcjonalności, co czyni go bardziej efektywnym w porównaniu z wieloma innymi językami programowania. Doświadczeni programiści stosujący Rails podkreślają, że platforma ta znacząco upraszcza proces tworzenia aplikacji internetowych, czyniąc go jednocześnie bardziej satysfakcjonującym. [4]

[4] Open source (pol. otwarte oprogramowanie) - *rodzaj oprogramowania komputerowego, w którym kod źródłowy jest wydawany na podstawie licencji, na mocy której właściciel praw autorskich przyznaje użytkownikom prawa do badania, zmiany i rozpowszechniania oprogramowania w ramach licencji wolnego oprogramowania […]* [3]

Filozofia Rails opiera się na dwóch kluczowych zasadach przewodnich:

1. Nie Powtarzaj Się (ang. Don't Repete Yourself): DRY to zasada inżynierii oprogramowania, która postuluje, że „każdy fragment wiedzy powinien mieć jedno, jednoznaczne i autorytatywne odwzorowanie w systemie informatycznym". Stosowanie tej zasady pozwala na eliminację redundancji w kodzie, co skutkuje poprawą jego czytelności, łatwiejszym utrzymaniem oraz rozbudową, a także zmniejsza ryzyko wprowadzania błędów. [4]
2. Konwencja Zamiast Konfiguracji: Rails zakłada, że istnieją ustalone, optymalne metody realizacji wielu zadań w kontekście aplikacji internetowych. W związku z tym, platforma preferuje stosowanie domyślnych konwencji, zamiast zmuszać programistów do definiowania szczegółowych ustawień za pomocą licznych plików konfiguracyjnych, co zwiększa spójność oraz redukuje złożoność konfiguracji systemu. [4]

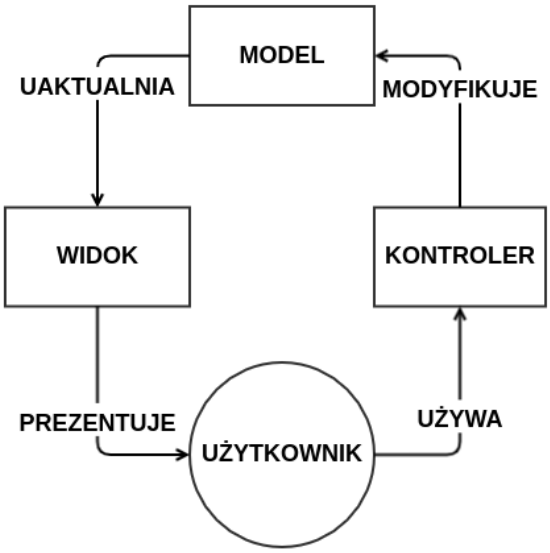
Framework[5] Rails został zbudowany w oparciu o architekturę MVC (ang. Model-View-Controller)

Model-View-Controller (pol. Model-Widok-Kontroler) – wzorzec architektoniczny służący do organizowania struktury aplikacji. Jego głównym celem jest oddzielenie logiki biznesowej, interfejsu użytkownika i kontrolowania przepływu danych, co zapewnia większą modularność i łatwiejsze zarządzanie kodem.

Podział aplikacji na trzy główne komponenty:

1. Model - odpowiada za przechowywanie i zarządzanie danymi aplikacji. Zawiera logikę biznesową oraz reguły, które operują na danych. Model reprezentuje stan aplikacji oraz odpowiada za interakcje z bazą danych.
2. Widok - odpowiada za prezentację danych użytkownikowi. Zawiera elementy interfejsu użytkownika, które umożliwiają interakcję systemu z użytkownikiem.
3. Kontroler – jest to pośrednik pomiędzy modelem a widokiem. Na podstawie otrzymanych danych wejściowych od użytkownika z warstwy widoku definiuje jakie operacje mają zostać wykonane w warstwie modelu. Może odpowiadać również za aktualizacje danych na widoku.

[5] Framework (pol. platforma programistyczna) - *szkielet do budowy aplikacji. Definiuje on strukturę aplikacji oraz ogólny mechanizm jej działania, a także dostarcza zestaw komponentów i bibliotek ogólnego przeznaczenia do wykonywania określonych zadań. […]* [5]



Rysunek 3.1: Graficzne przedstawienie wzorca architektonicznego MVC
źródło: https://pl.wikipedia.org/wiki/Model-View-Controller

Uzasadnienie wyboru

Koncepcja „konwencja zamiast konfiguracji" jest kluczowym powodem wyboru tej technologii. Podejście to minimalizuje ilość ręcznych ustawień i konfiguracji potrzebnych do rozpoczęcia pracy nad aplikacją. Zamiast tego bazuje na predefiniowanych konwencjach, które przyspieszają proces tworzenia oprogramowania i redukują możliwość błędów. Kolejnym ważnym aspektem jest zastosowanie architektury MVC, która zapewnia przejrzystą separację logiki biznesowej, interfejsu użytkownika oraz przepływu danych, co znacząco ułatwia rozwój i utrzymanie aplikacji. Wybrana technologia korzysta z Hotwire[6], czyli innowacyjnego podejścia do budowania dynamicznych i interaktywnych aplikacji. Dodatkowo korzystanie z Ruby On Rails umożliwia prostą konfiguracje połączenia z bazą danych oraz wykonywanie operacji na danych za pomocą ORM (ang. Object-Relational Mapping) o nazwie Active Record.

[6] Hotwire (ang. HTML Over The Wire) - pozwala na minimalizację kodu JavaScript poprzez przesyłanie gotowego HTML z serwera w odpowiedzi na akcje użytkownika. Dzięki komponentom takim jak Turbo, aplikacja może obsługiwać szybkie nawigacje, aktualizować fragmenty strony bez jej przeładowywania i dynamicznie reagować na zmiany w czasie rzeczywistym. [6]

3.1.2 Backend

Ruby - *interpretowany, w pełni obiektowy i dynamicznie typowany język programowania stworzony w 1995 roku przez Yukihiro Matsumoto. […]* [7]

Język ten wyróżnia się czytelną i zwięzłą składnią, która zwiększa produktywność programistów i upraszcza proces pisania kodu. Ruby wspiera programowanie zorientowane na testy i oferuje elastyczność dzięki możliwości dynamicznego rozszerzania klas.

Uzasadnienie wyboru

Ruby jako dynamiczny język programowania pozwala na elastyczność podczas procesu tworzenia aplikacji webowej. Połączenie tego języka z frameworkiem Ruby On Rails przyśpiesza i upraszcza proces budowy aplikacji oraz realizację założonych celów.

3.1.3 Baza danych

MySQL - popularny open sourcowy system zarządzania relacyjnymi bazami danych oparty na języku SQL (Structured Query Language – strukturalny język zapytań). SQL to język programowania, który jest wykorzystywany do tworzenia zapytań dotyczących danych, a także do ich przetwarzania, definiowania. Dzięki bazom danych możliwe jest poprawne działanie witryny. [8]

Relacyjna baza danych – sposób przechowywania danych w postaci tabel, zorganizowanych w kolumny i wiersze, gdzie każda tabela reprezentuje jeden typ informacji. Przechowywane zasoby tworzą powiązane ze sobą zbiory danych za pomocą kluczy głównych oraz kluczy obcych, które pozwalają na utrzymanie integralności danych.

Uzasadnienie wyboru

Istotnymi argumentami wyboru relacyjnej bazy danych MySQL jest jej wydajność, niezawodność oraz duża skalowalność, która ma kluczowe znaczenie w kontekście rozwoju tworzonego oprogramowania. Dodatkowo, jest to system, który łatwo integruje się z fremworkiem Ruby On Rails ułatwiając zarządzanie bazą danych.

Schemat bazy danych

Rysunek 3.2: Schemat bazy danych
źródło: https://dbdiagram.zio

Opis tabel

active_storage_blobs	Przechowuje metadane dotyczące plików przesyłanych do aplikacji, takie jak klucz, nazwa pliku, typ zawartości, rozmiar oraz suma kontrolna.
active_storage_variant_records	Zawiera dane o wariantach (np. zmniejszone wersje) plików zapisanych w **active_storage_blobs**.
active_storage_attachments	Przechowuje dane związane z przypisanymi plikami do modeli m.in. nazwa, typ, wielkość oraz powiązanie z plikami z tabeli **active_storage_blobs.**
users	Zawiera wszystkie dane związane z kontami użytkowników, takie jak e-mail, hasło, numer telefonu, a także dodatkowe informacje, takie jak szkoła i status konta. Pola związane z rejestracją/logowaniem zostały wygenerowane za pomocą biblioteki Devise, która odpowiada za te procesy. Tabela zawiera również pole **type**, które jest używane w ramach **Single Table Inheritance (STI)**, umożliwiając różnicowanie kont użytkowników pod względem ich ról, np. na uczniów i nauczycieli.
student_teachers	Tabela umożliwiająca łączenie modeli uczniów i nauczycieli za pomocą relacji wiele do wielu, co pozwala na identyfikacje ich powiązań
lessons	Przechowuje informacje dotyczące lekcji, takie jak godzina zajęć, czas trwania, rok szkolny, dni tygodnia, kolor kafelka na kalendarzu oraz powiązanie z uczniem i nauczycielem
reports	Stanowią rozszerzenie informacji dotyczących ucznia, zwierają takie dane jak prowadzony przedmiot, poziom nauczania, klasa, rodzaj szkoły, email do rodzica oraz powiązanie z uczniem i nauczycielem
topics	Reprezentuje tematy zajęć, przechowując tytuły, daty oraz powiązanie z konkretnymi lekcjami.
conversations	Zawiera informacje o konwersacji pomiędzy uczniem a nauczycielem
messages	Przechowuje wiadomości występujące w systemu oraz powiązanie z konwersacją
notes	Przechowuje notatki użytkowników, składające się z tytułu, treści i odniesienia do właściciela.

Tabela 3.1: Opis tabel bazy danych

3.1.4 Biblioteki i narzędzia programistyczne

Bootstrap – zintegrowany framework CSS i JavaScript wykorzystywany do tworzenia responsywnych interfejsów użytkownika za pomocą oferowanych komponentów. [9]

Redis – pamięć podręczna oparta na bazie danych typu NoSQL. [10]

Devise – kompleksowe narzędzie wykorzystywane do uwierzytelniania użytkowników w aplikacji, oferujące funkcje m.in. rejestracje, logowanie, przywracanie hasła. [11]

CanCanCan – kluczowe narzędzie wspomagające zarządzanie autoryzacją w aplikacjach Rails, pozwalające na wygodne zarządzanie rolami użytkowników oraz dostępem do treści i akcji. [12]

Ransack – biblioteka zapewniająca dostęp do zaawansowanego wyszukiwania i filtrowania danych z bazy. [13]

TOASTUI Calendar – zaawansowana biblioteka JavaScript dostarczająca interaktywny kalendarz oferujący dynamiczne generowanie wydarzeń oraz wyświetlanie harmonogramów w widokach dziennym, tygodniowym lub miesięcznym. [14]

ice_cube – biblioteka wspomagająca prace z powtarzającymi się zdarzeniami i harmonogramami. [15]

Slim – umożliwia korzystanie z dostarczanych szablonów Slim, które upraszczają proces tworzenia aplikacji Rails. [16]

Simple form – popularna biblioteka wykorzystywana w celu uproszczenia generowania formularzy w systemie. [17]

Kaminari – zapewnia proste stronicowanie danych, umożliwiając dzielenie dużych zasobów danych ma mniejsze sekcje. [18]

jquery-rails – zapewnia integrację biblioteki jQuery z aplikacją Rails, umożliwiając wykorzystywanie funkcji JavaScript na stronach aplikacji. [19]

Chosen– zapewnia stylowanie i konfiguracje list rozwijanych oraz list wielokrotnego wyboru, dzięki czemu stają się one bardziej intuicyjne dla użytkownika. [20]

Font Awesome – dostarcza dostęp do darmowych ikon. [21]

MiniMagick – przetwarzanie obrazów, umożliwia manipulowanie obrazami w aplikacji Rails. [22]

Phonelib – walidacja i formatowanie numerów telefonów z uwzględnieniem formatów międzynarodowych [23]

3.2 Spis wersji technologii

1. Ruby **3.3.2**
2. Rails **7.1.3**
3. MySQL **0.5.4**
4. Bootstrap **5.1.3**
5. Redis **7.2.5**
6. devise **4.9.3**
7. cancancan **3.5**
8. ransack **4.0**
9. tui-calendar **1.15.3**
10. ice_cube **0.17.0**
11. slim-rails **5.2.1**
12. simple-form **5.3.1**
13. kaminari **1.2.2**
14. jquery-rails **4.6.0**
15. chosen-rails **1.10.0**
16. font-awesome-rails **4.7.0.8**
17. mini_magick **4.13.2**
18. phonelib **0.9.2**

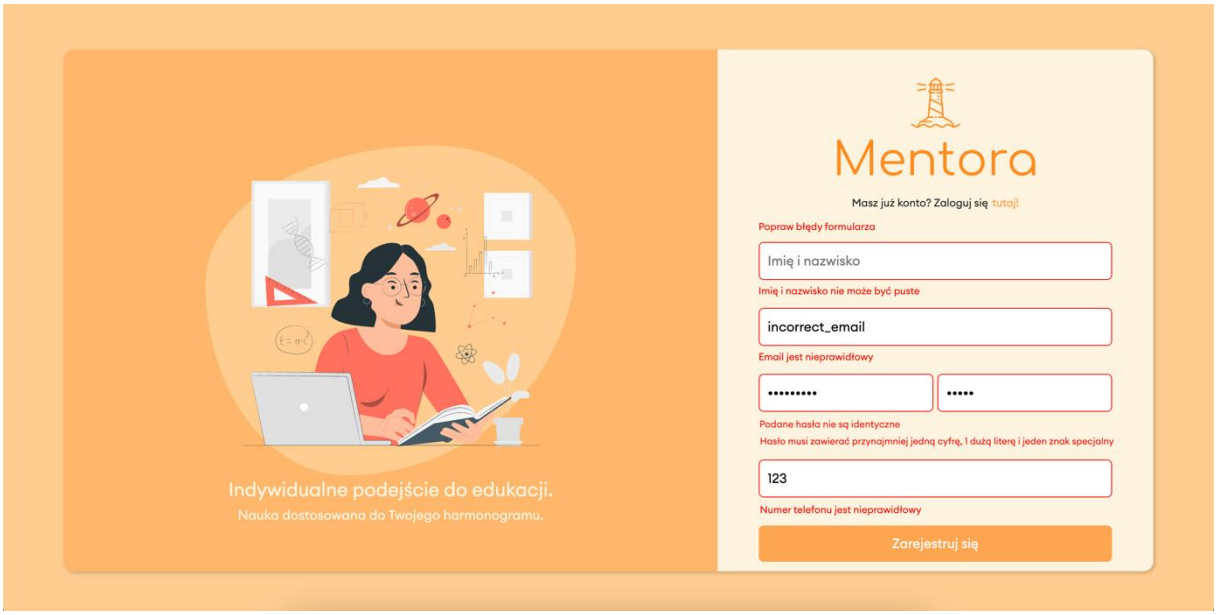
3.3 Testowanie

Testowanie aplikacji jest istotnym proces tworzenia oprogramowania, podczas którego system poddawany jest symulacją mającym na celu sprawdzenie jego niezawodności, wydajności oraz odporności na błędne dane lub znacznie zwiększony ruch. Dzięki przeprowadzonym testom możliwe jest wykrycie potencjalnych błędów lub słabości projektowanego systemu oraz podjęcie odpowiednich kroków by końcowa wersja aplikacji była produktem wysokiej jakości.

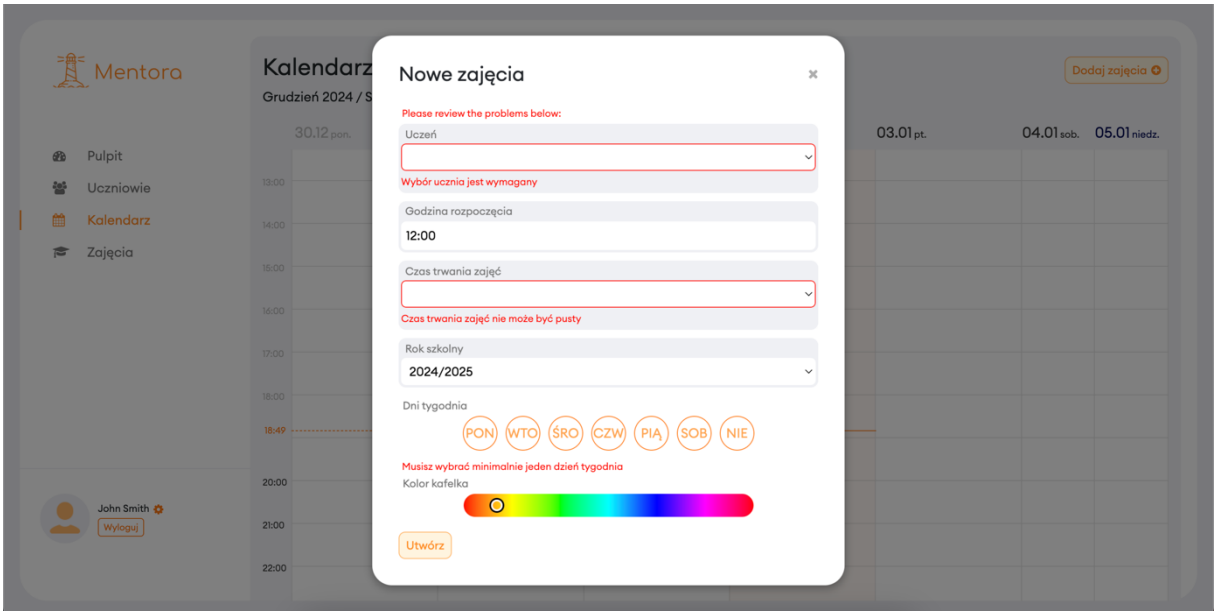
W niniejszym rozdziale zostaną zaprezentowane oraz opisane testy przeprowadzone na stworzonym systemie wraz z ich rezultatem.

3.3.1 Testy manualne

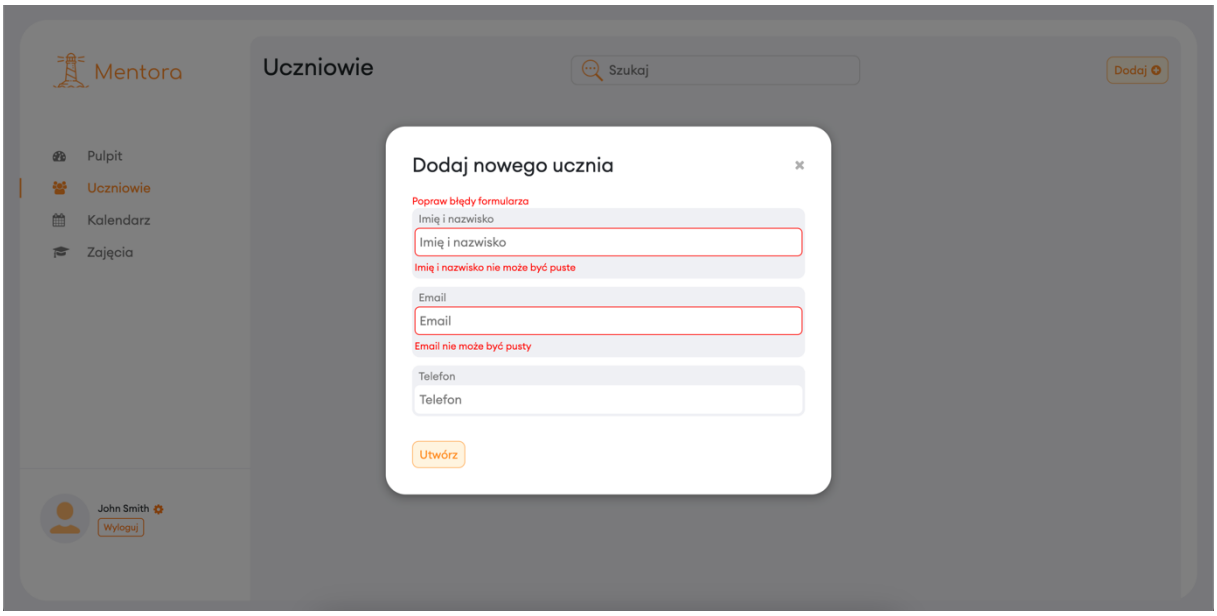
Pierwszym etapem prowadzonych testów będą próby wprowadzania nieprawidłowych lub niekompletnych danych do formularzy dostępnych w interfejsie użytkownika.



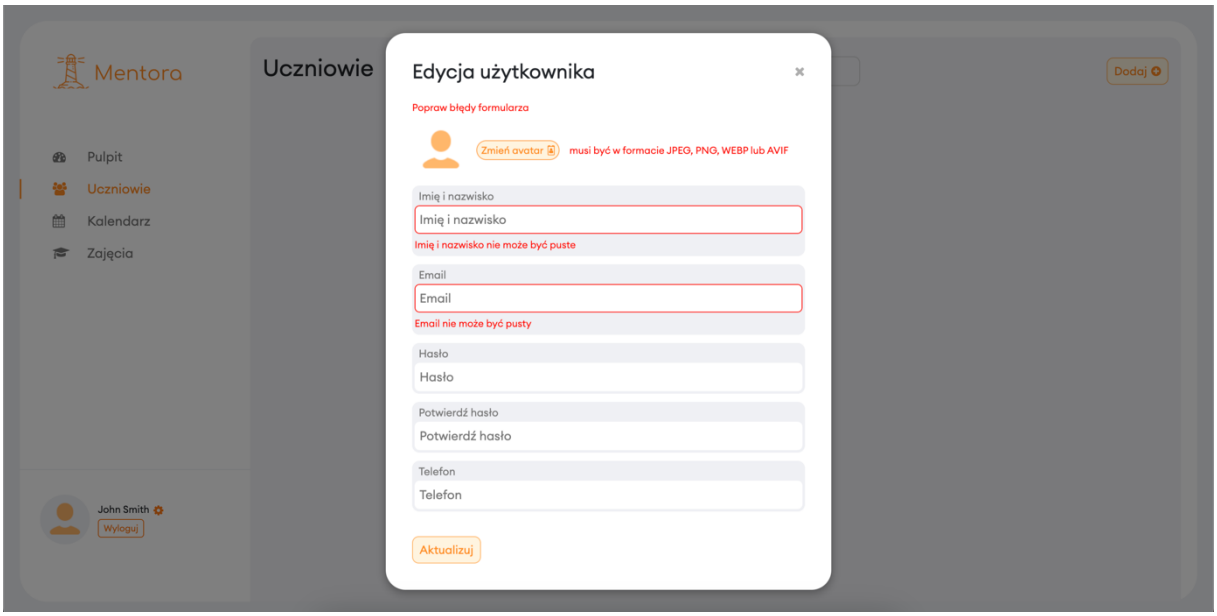
Rysunek 3.3: Próba wprowadzenia błędnych danych w formularzu rejestracji
źródło: opracowanie własne



Rysunek 3.4: Próba wprowadzenia błędnych danych w formularzu dodawania nowych zajęć
źródło: opracowanie własne



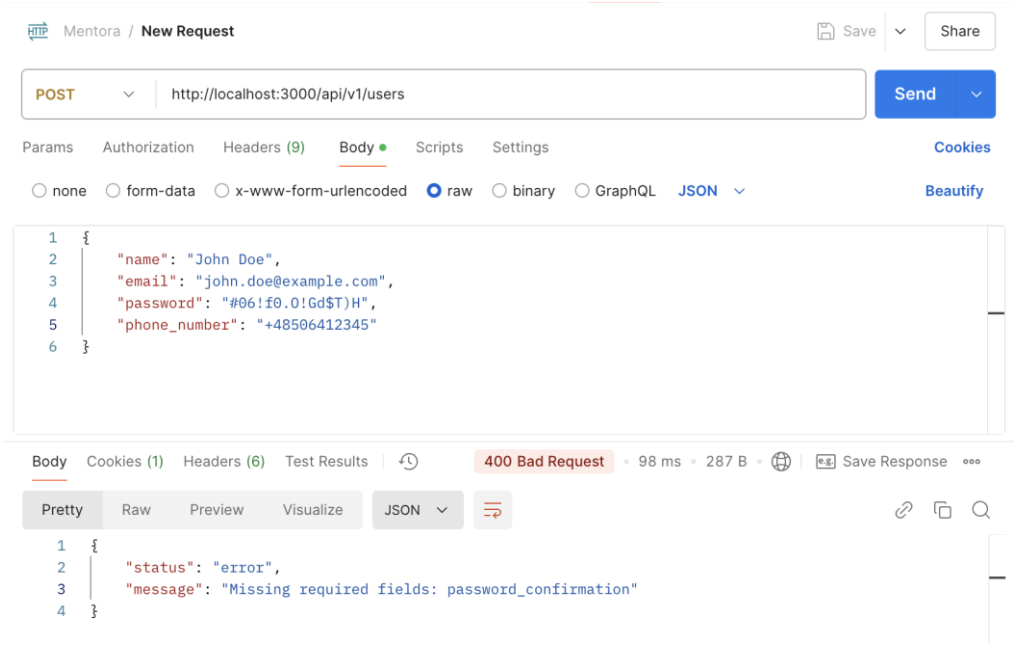
Rysunek 3.5: Próba wprowadzenia błędnych danych w formularzu dodawania nowego ucznia
źródło: opracowanie własne



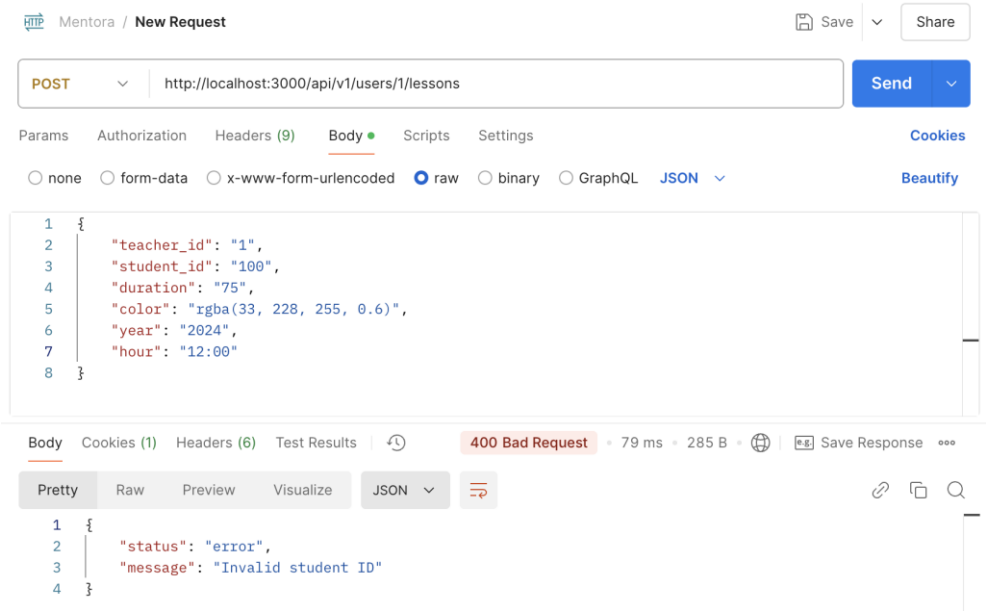
Rysunek 3.6: Próba wprowadzenia błędnych danych w formularzu edycji profilu użytkownika
źródło: opracowanie własne

Przeprowadzone testy przedstawiają działanie walidacji w systemie, które uniemożliwiają użytkownikowi wprowadzenie błędnych danych w formularzach aplikacji. Dzięki tym mechanizmom system jest odporny na błędy związane z nieprawidłowym formatem danych, brakiem wymaganych pól czy też próbami wprowadzenia wartości niespełniających określonych kryteriów.

Następnym krokiem jest wykonanie testów po stronie serwerowe za pomocą narzędzia **Postman**, które umożliwia przesłanie danych testowych na endpointy[7] backendu. Przesłane dane będą zawierać celowe braki lub błędy, w celu sprawdzenia reakcji serwera.



Rysunek 3.7: Przesłanie błędnych danych rejestracyjnych do backendu
źródło: program Postman



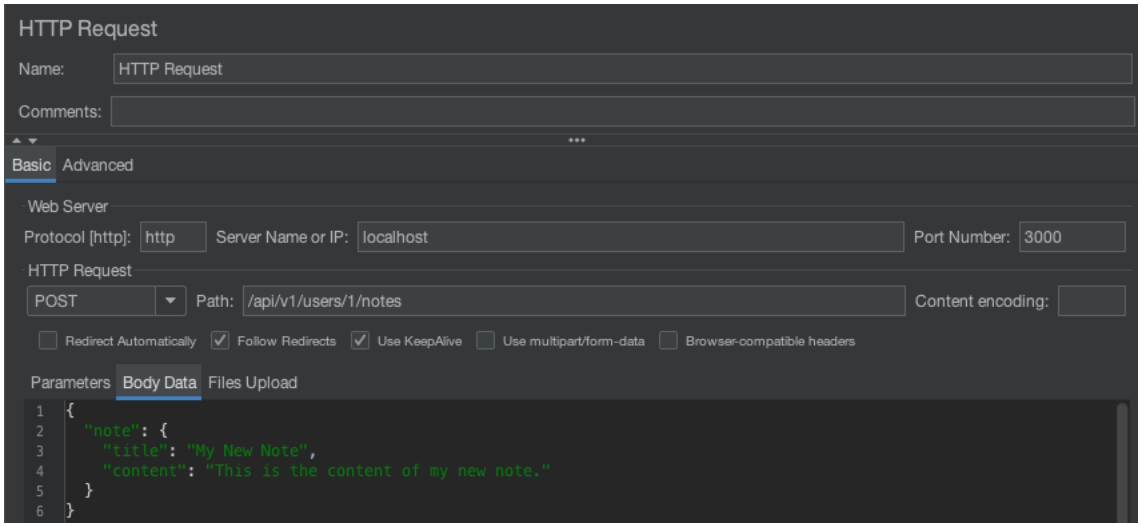
Rysunek 3.8: Przesłanie błędnych danych nowych zajęć do backendu
źródło: program Postman

[7]Endpoint - Punkt końcowy API to adres URL, który pełni rolę punktu kontaktowego między klientem API a serwerem API. Klienci API wysyłają żądania do punktów końcowych API, aby uzyskać dostęp do funkcjonalności i danych udostępnianych przez API. [24]

W przypadku przesłania błędnych lub niekompletnych danych serwer odpowiada statusem 400 (Bad Request), informującym o błędzie w żądaniu. W ten sposób dane przesyłane przez użytkownika są sprawdzane zarówno po stronie interfejsu jak i serwera.

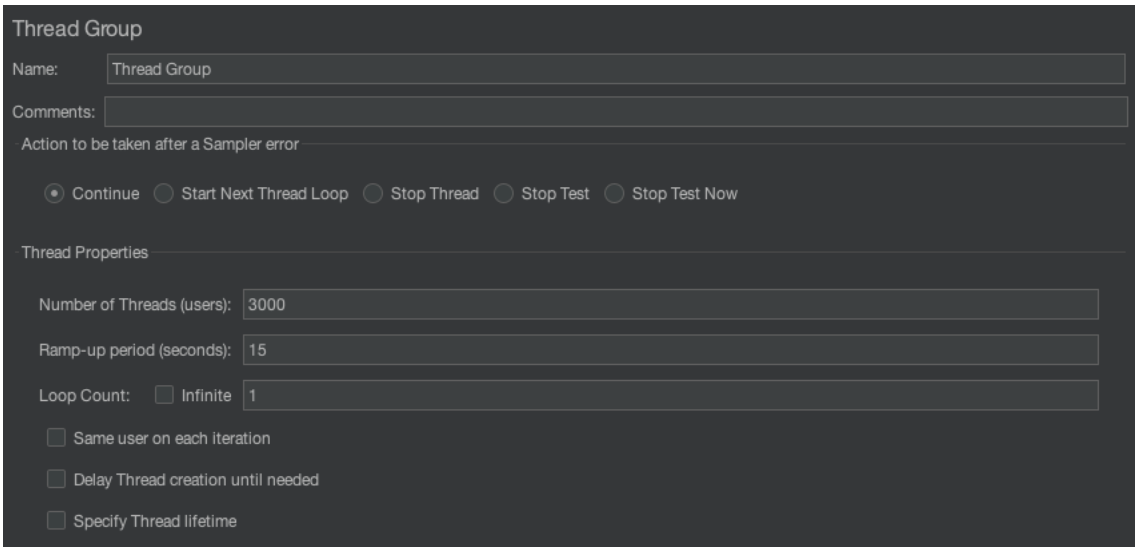
3.3.2 Testy obciążeniowe

W celu zweryfikowania wydajności i stabilności aplikacji, wykonane zostaną test obciążeniowe za pomocą narzędzia **Apache JMeter**, które umożliwia symulacje rzeczywistego ruchu użytkowników oraz określenie w jaki sposób system zareaguje na znacznie zwiększone obciążenie serwera.



Rysunek 3.9: Konfiguracja wysyłanych żądań do serwera aplikacji
źródło: program JMeter

Pierwszy krokiem jest odpowiednia konfiguracja żądań wysyłanych do serwera aplikacji. Testowe żądanie dotyczy tworzenia nowej notatki dla użytkownika posiadające numer ID równy 1. Przekazujemy w ciele zapytania tytuł oraz zawartość notatki.



Rysunek 3.10: Ustawienie odpowiednich parametrów symulacji
źródło: program JMeter

Następnie ustawiane są kluczowe wartości będące parametrami symulacji m.in. liczba wątków (użytkowników) wysyłających zapytania do serwera oraz czas trwania symulacji. Podczas tego testu zakładamy, że 3 tysięcy użytkowników będzie wykonywać zapytania w przeciągu 15 sekund.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Through...	Receive...	Sent KB...
HTTP R...	3000	1177	1041	1382	1670	6714	0	19517	6.57%	110.5/sec	74.65	29.75
TOTAL	3000	1177	1041	1382	1670	6714	0	19517	6.57%	110.5/sec	74.65	29.75

Rysunek 3.11: Szczegółowy raport przeprowadzonej symulacji
źródło: program JMeter

Po zakończeniu symulacji zostaje wygenerowany szczegółowy raport zawierający informacje:

Label – nazwa elementu testowanego (żądanie HTTP).

Samples – ilość próbek, liczba wykonanych żądań w czasie testu (3000 żądań).

Average – średni czas odpowiedzi serwera podany w milisekundach (1177 ms).

Median – mediana czasu odpowiedzi serwera podana w milisekundach (wartość środkowa wszystkich czasów odpowiedzi, połowa wszystkich żądań miało czas odpowiedzi mniejszy niż 1041 ms).

90% Line – 90 procent żądań miało czas odpowiedzi krótszy niż 1382 ms.

95% Line – 95 procent żądań miało czas odpowiedzi krótszy niż 1670 ms.

99% Line – 99 procent żądań miało czas odpowiedzi krótszy niż 6714 ms.

Min – najkrótszy czas odpowiedzi (0 ms. najprawdopodobniej odpowiedzi zawierające informacje o błędzie zwracane natychmiastowo)

Maximum – najdłuższy czas odpowiedzi (19 517 ms.)

Error % – procent żądań zakończonych błędem (6,57 %)

Throughput – Liczba żądań przetwarzanych na sekundę (110,5/sekundę)

Received KB/sec – Ilość otrzymanych danych od serwera na sekundę (74,65 KB/sekundę)

Sent KB/sec – Ilość danych wysłanych do serwera na sekundę (29,75 KB/sekundę)

Raporty przeprowadzone za pomocą narzędzia JMeter dostarczają cennych informacji dotyczących wydajności i stabliności systemu. Wartość średniego czasu odpowiedzi serwera w porównaniu z dobrym wynikiem mediany sugerują, że większość żądań jest obsługiwana w stosunkowo krótkim czasie, co potwierdza poprawność działania aplikacji w warunkach obciążenia. Wysoka wartość czasu odpowiedzi dla 1% żądań, procentowa ilość błędów wynosząca 6,57 oraz maksymalna wartość czasu odpowiedzi wskazują na potencjalne problemy pojawiające się w szczególnych przypadkach.

Wyniki przeprowadzonej symulacji wskazują na dobrą wydajność aplikacji w standardowych warunkach obciążenia, ale również są podstawą do przeprowadzenia analizy systemu w zakresie eliminacji długich czasów odpowiedzi oraz redukcji liczby błędów. W przypadku dalszego zwiększania obciążenia systemu jego wydajność może znacznie zmaleć, więc optymalizacja strony serwerowej jest kluczowym etapem realizowanym podczas dalszego rozwoju systemu.

Rozdział 4

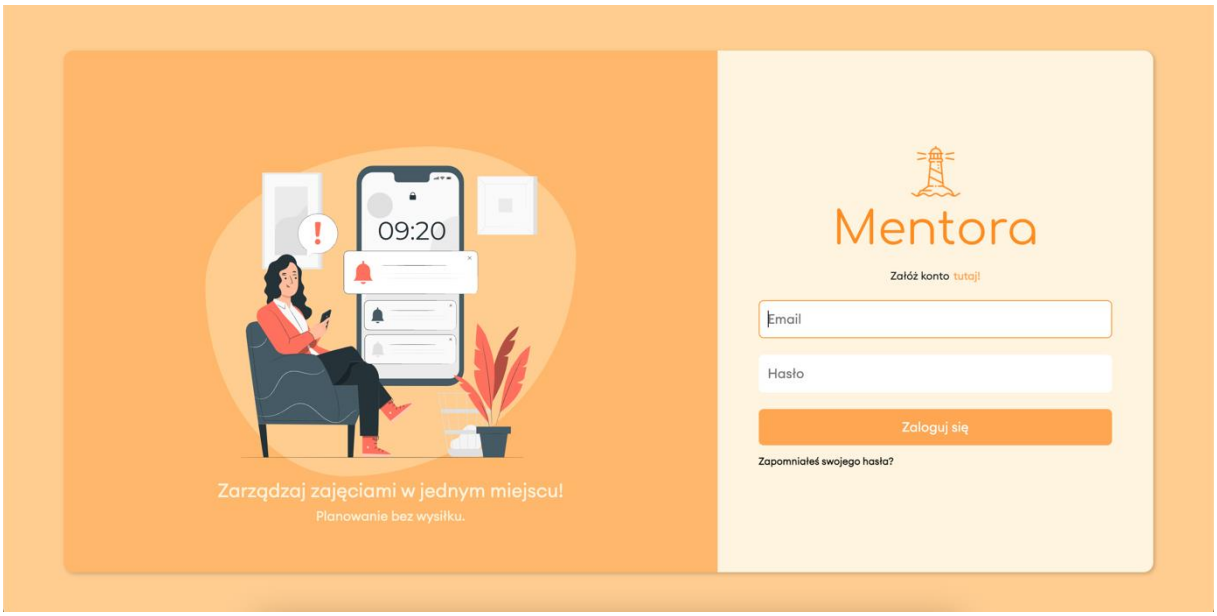
Prezentacja aplikacji

W niniejszym rozdziale przedstawiona zostanie opracowana aplikacja. Omówione zostaną kluczowe funkcjonalności systemu, jego struktura, moduły oraz rozwiązania technologiczne zastosowane w projekcie. Ta część pracy ma na celu ukazanie zarówno praktycznego zastosowania aplikacji, jak i korzyści jakie niesie, docelowym grupom użytkowników – uczniom, nauczycielom oraz rodzicom.

4.1 Moduł rejestracji i logowania

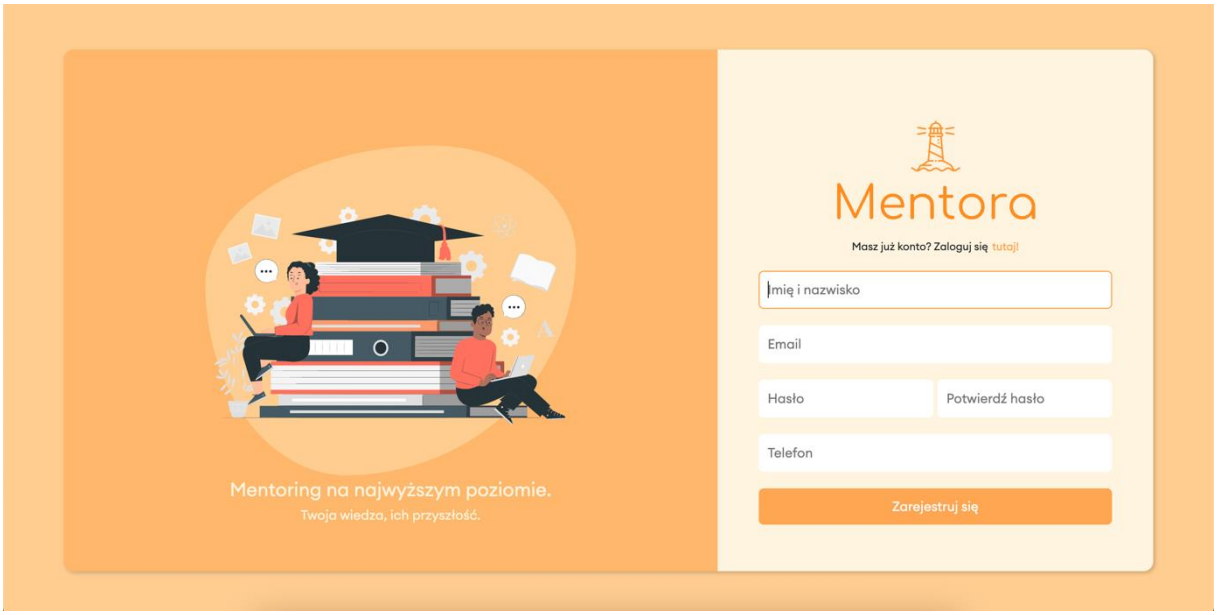
Funkcjonalność rejestracji, logowania oraz przywracania hasła, została utworzona za pomocą biblioteki Devise, która oferuje kompleksowe rozwiązania dotyczące zakładania kont użytkowników w systemie. Dostarczane są wbudowane mechanizmy uwzględniające obsługę wielu aspektów bezpieczeństwa m.in. walidacja danych wprowadzanych przez użytkownika, ochrona przed nieautoryzowanym dostępem oraz szyfrowanie haseł.

W przypadku wprowadzenia niepoprawnych danych w poniższych formularzach, system wyświetli odpowiednie powiadomienia dla użytkownika.



Rysunek 4.1: Ekran logowania użytkownika
źródło: opracowanie własne

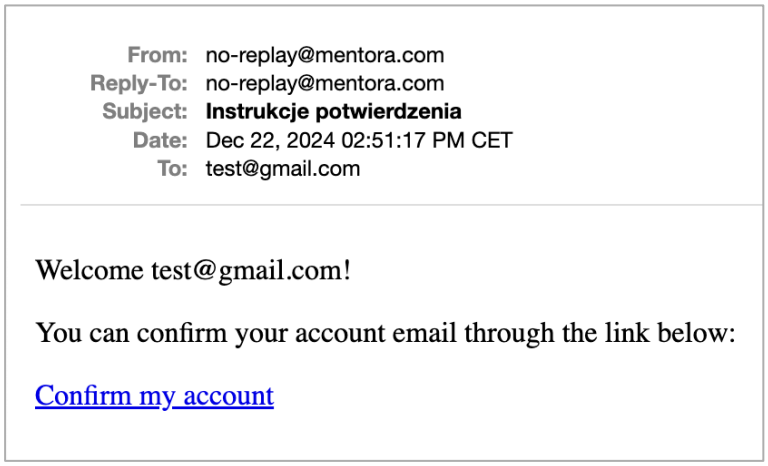
Użytkownicy na stronie logowania, mogą uzyskać dostęp do swojego konta po podaniu poprawnych danych.



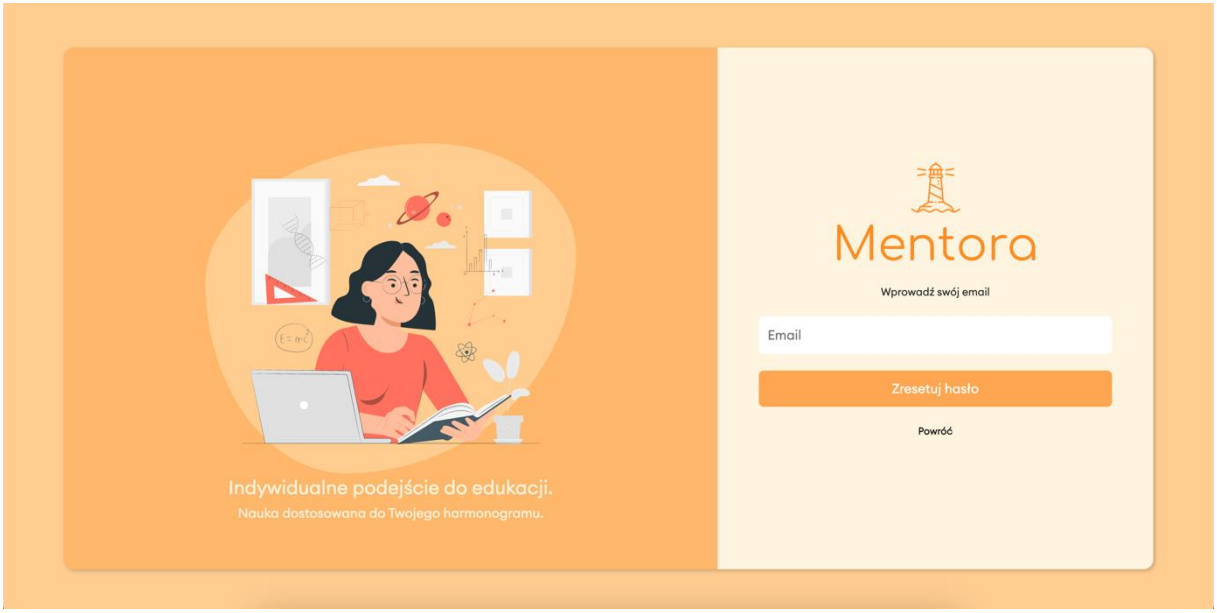
Rysunek 4.2: Ekran rejestracji użytkownika
źródło: opracowanie własne

Rejestracja nowego użytkownika dotyczy tworzenia tylko kont nauczyciela. Konta uczniowskie są automatycznie tworzone, w momencie dodania ucznia do systemu przez nauczyciela. Proces ten zostanie szczegółowo omówiony w kolejnych podrozdziałach niniejszego rozdziału.

Jeżeli użytkownik wypełni formularz poprawnymi danymi i proces rejestracji zostanie zakończony sukcesem, na podany adres email zostanie wysłana wiadomość mailowa z linkiem aktywacyjnym konta.

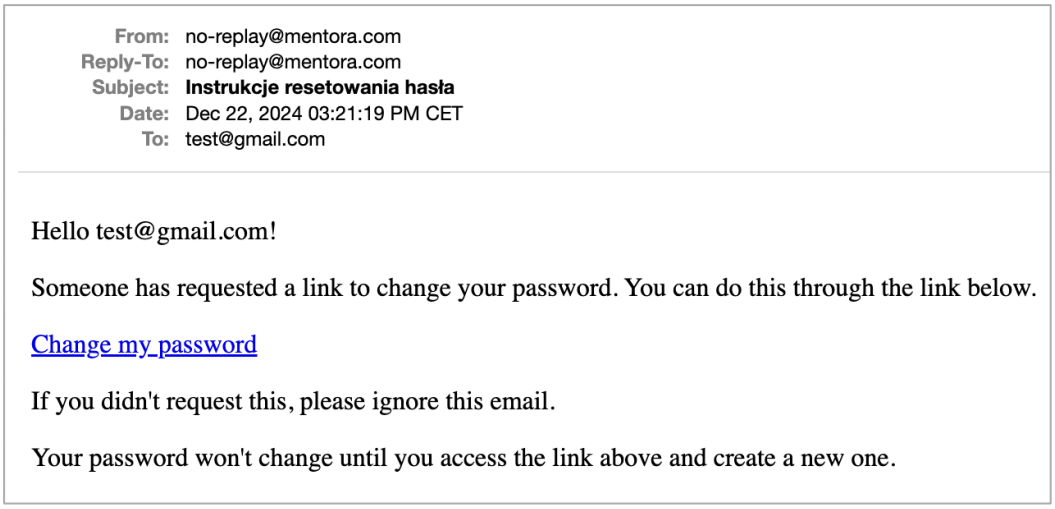


Rysunek 4.3: Wiadomość mailowa zawierająca link aktywacyjny
źródło: opracowanie własne



Rysunek 4.4: Ekran przywracania hasła użytkownika
źródło: opracowanie własne

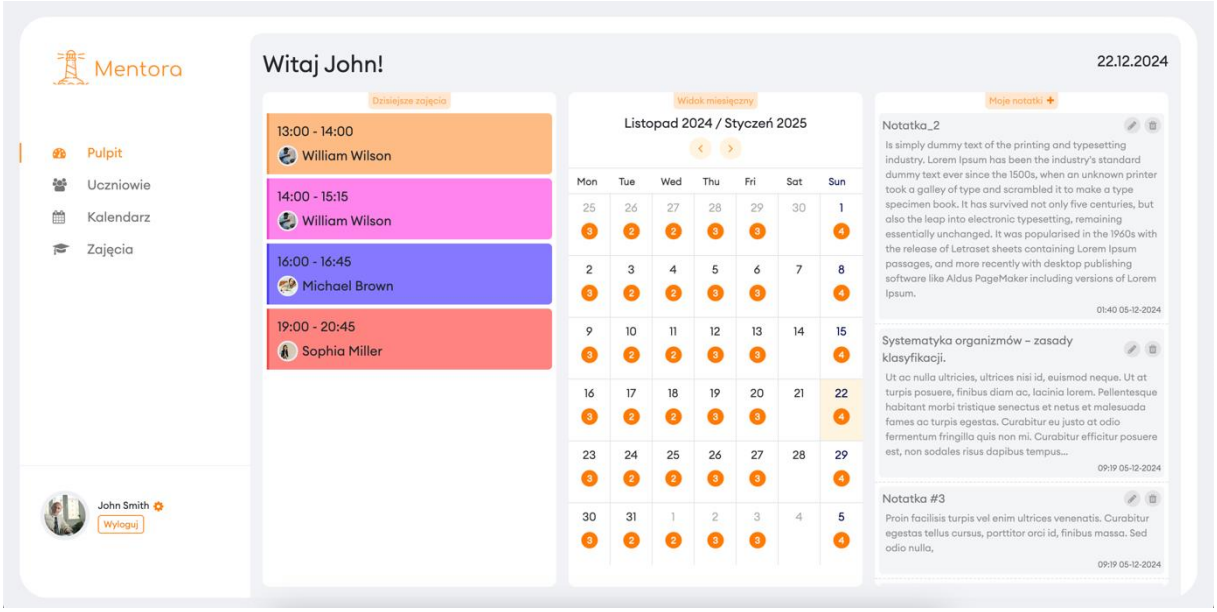
Użytkownik ma możliwość skorzystania z funkcjonalności przywrócenia hasła. Po podaniu odpowiedniego adresu email na skrzynkę pocztową zostaje wysłana wiadomość zawierająca instrukcje dotyczące przywracania hasła.



Rysunek 4.5: Wiadomość mailowa zawierająca instrukcje dotyczące przywrócenia hasła
źródło: opracowanie własne

4.2 Konto nauczyciela

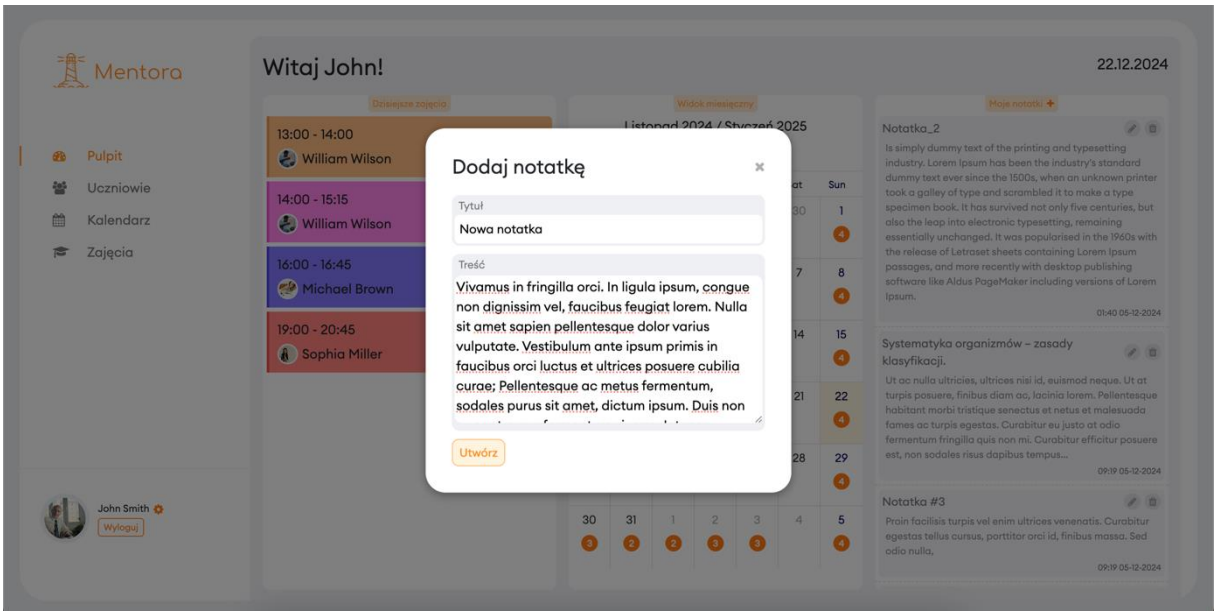
Zakończony sukcesem proces rejestracji tworzy w systemie konto nauczyciela. Nauczyciel ma dostęp do funkcji oraz dedykowanych narzędzi, które umożliwiają zarządzanie dodatkowymi zajęciami pozaszkolnymi.



Rysunek 4.6: Pulpit konta nauczyciela
źródło: opracowanie własne

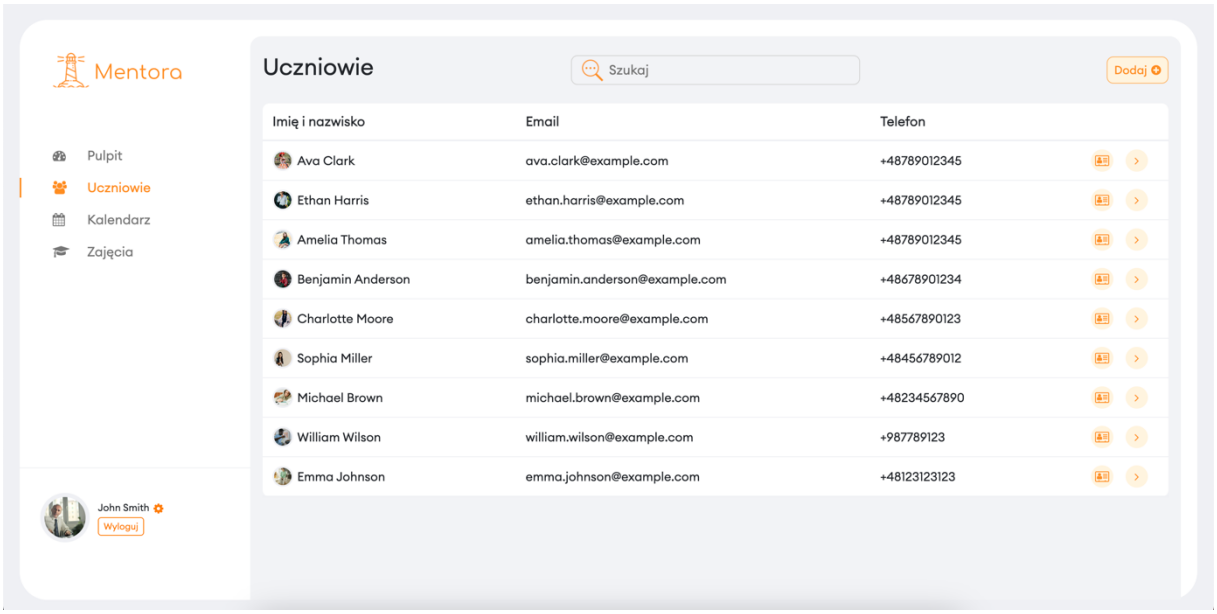
Po zalogowaniu się, nauczyciel zostaje przeniesiony na pulpit. Zawartość strony aplikacji zawsze podzielona jest na dwa segmenty – pionowy wąski panel nawigacyjny po lewej stronie oraz treść właściwa w dużym prostokącie po prawej stronie.

W zakładce „Pulpit" użytkownik ma szybki dostęp do trzech kluczowych sekcji. Pierwsza sekcja to **lista dzisiejszych zajęć**, które przedstawia wszystkie planowane zajęcia w danym dniu, pozwalając użytkownikowi na szybie sprawdzenie, jakie aktywności czekają go w ciągu dnia. Kafelek reprezentujący spotkanie zawiera informacje o godzinie spotkania oraz imię i nazwisko ucznia biorącego udział w zajęciach. Klikając w kafelek nauczyciel przechodzi do panelu zajęć. Drugą sekcją jest **miesięczny widok harmonogramu**, który umożliwia przeglądanie całego miesiąca i pozwala na łatwe śledzenie nadchodzących wydarzeń, zarówno tych zaplanowanych, jak i przyszłych. Ostatnia sekcja to **prywatne notatki**, w której nauczyciele mogą przechowywać swoje notatki związane z zajęciami, uczniami lub innymi istotnymi sprawami. Ta sekcja zapewnia szybki dostęp do zapisanych uwag i jest pomocna w organizowaniu i zarządzaniu codziennymi obowiązkami.



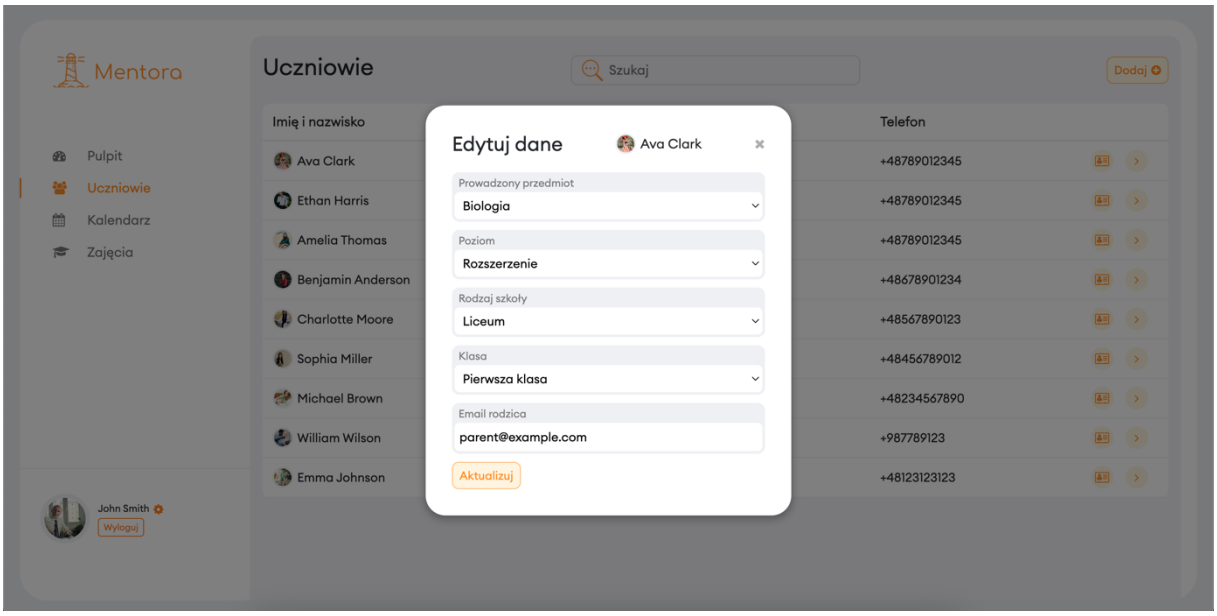
Rysunek 4.7: Formularz wyświetlany podczas dodawania nowej notatki
źródło: opracowanie własne

4.2.1 Dodawanie nowych uczniów



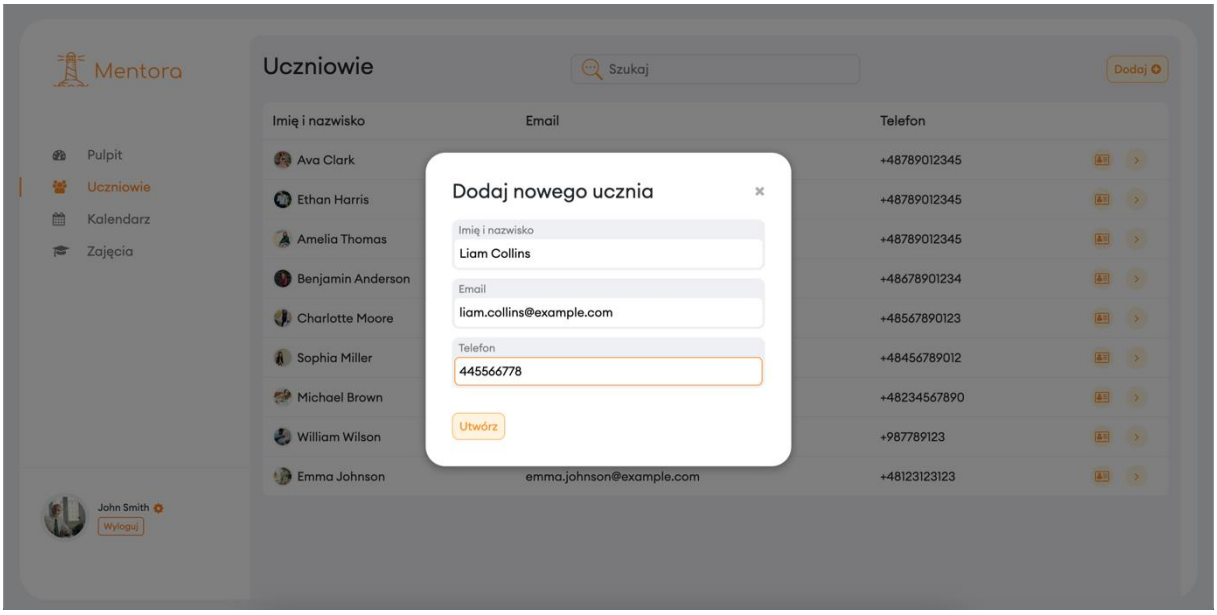
Rysunek 4.8: Lista uczniów przypisanych do nauczyciela
źródło: opracowanie własne

Po przejściu do zakładki „Uczniowie” zostaje wyświetlona lista uczniów przypisanych do konta nauczyciela. Z poziomu tego widoku nauczyciela ma możliwość uzupełnić dodatkowe informacje o uczniu takie jak prowadzony przedmiot, poziom nauczania, rodzaj szkoły, klasa, email rodzica lub przejść do panelu zajęć.



Rysunek 4.9: Formularz wyświetlany podczas uzupełniania dodatkowych informacji o uczniu
źródło: opracowanie własne

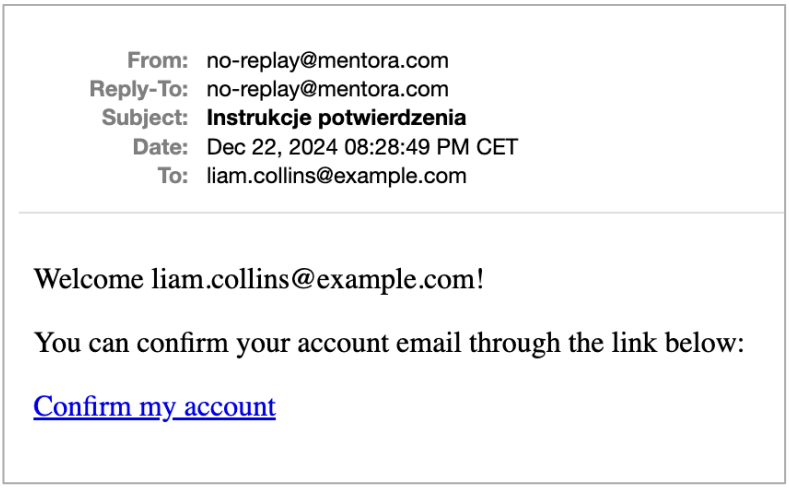
Najważniejszą funkcją w prezentowanej sekcji jest możliwość przypisania nowego ucznia do konta nauczyciela. Proces ten automatycznie tworzy nowe konto ucznia, jeśli podany email nie znajduje się jeszcze w systemie. W przypadku gdy uczeń posiada już konto powiązane z podanym adresem email, system dokonuje tylko połączenia konta nauczyciela i istniejącego konta ucznia.



Rysunek 4.10: Formularz wyświetlany podczas dodawania nowego ucznia
źródło: opracowanie własne

System ogranicza ingerencję ucznia w procedurę tworzenia konta, wykonując większość czynności związanych z tym procesem. Po wprowadzeniu przez nauczyciela wymaganych danych, system samodzielnie dokonuje założenia konta. Dzięki temu uczeń unika standardowych procedur rejestracyjnych, co pozwala na szybszy i bardziej efektywny dostęp do platformy, jednocześnie minimalizując ryzyko błędów związanych z ręcznym wprowadzaniem danych.

Zakończony sukcesem proces dodawania nowego ucznia skutkuje wysłaniem dwóch wiadomości email na podany adres mailowy. Pierwsza wiadomość zawiera standardowy link aktywacyjny, natomiast druga wiadomość zawiera tymczasowe hasło wygenerowane automatycznie.

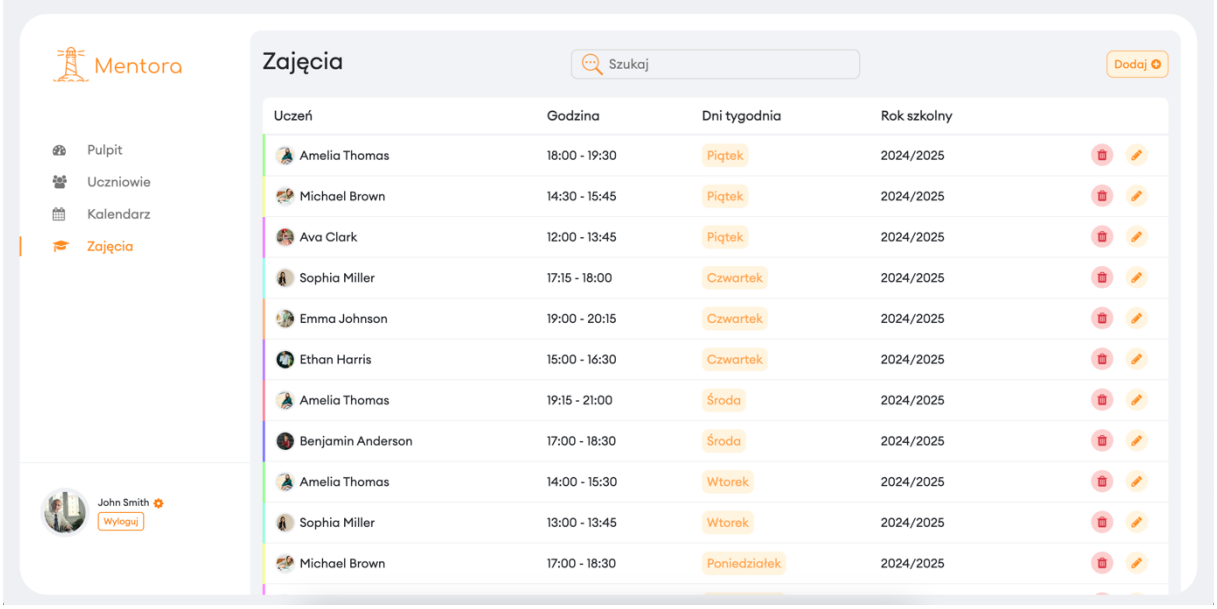


Rysunek 4.11: Wiadomość mailowa zawierająca link aktywacyjny
źródło: opracowanie własne

Rysunek 4.12: Wiadomość mailowa zawierająca automatycznie wygenerowane hasło
źródło: opracowanie własne

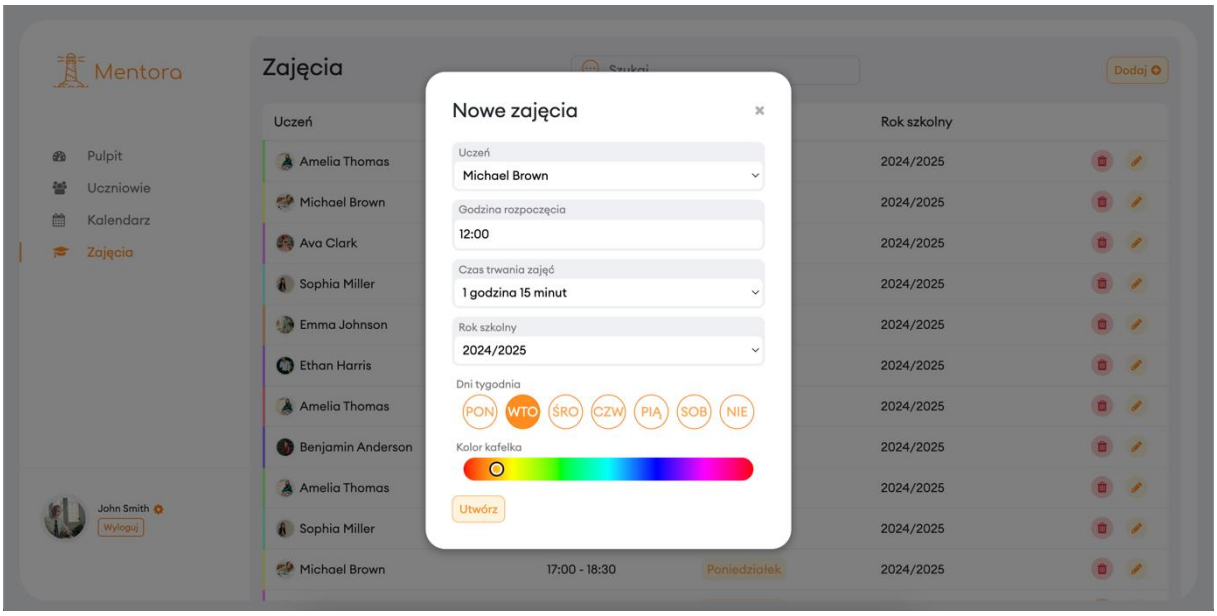
4.2.2 Planowanie zajęć

Zakładka „Zajęcia” wyświetla listę wszystkich zajęć użytkownika. Dla nauczyciela zakładka ta oferuje dodatkowe funkcjonalności, takie jak możliwość edytowania szczegółów poszczególnych zajęć, w tym ich daty, godziny, roku szkolnego czy przypisanych uczniów. Ponadto, nauczyciel ma opcję usunięcia wybranych zajęć, co pozwala na bieżące dostosowywanie harmonogramu do zmieniających się potrzeb oraz filtrowania listy po imieniu i nazwisku ucznia.



Rysunek 4.13: Lista wszystkich zajęć użytkownika
źródło: opracowanie własne

Funkcjonalność planowania nowych zajęć jest kluczowym elementem systemu, umożliwiającym nauczycielom precyzyjne organizowanie swojej pracy. Za pomocą dedykowanego formularza, nauczyciel może łatwo dodać nowe zajęcia, określając ich szczegóły, takie jak godzina rozpoczęcia zajęć, czas trwania zajęć, rok szkolny, dni tygodnia, kolor kafelka na kalendarzu oraz wybór ucznia.



Rysunek 4.14: Formularz wyświetlany podczas dodawania nowych zajęć
źródło: opracowanie własne

Po utworzeniu zajęć nauczyciel oraz przypisany uczeń uzyskują dostęp do panelu lekcji. Gdy proces zostanie poprawnie zakończony, system wyśle powiadomienie w postaci wiadomości mailowej na adres rodzica przypisany do konta ucznia.

From: no-reply@example.com
Subject: **Schedule update!**
Date: Dec 27, 2024 12:06:35 PM CET
To: hubert.brown@example.com

Welcome hubert.brown@example.com

We would like to inform you that new classes have been scheduled in the system. Below are the details:

- **Day of the week:** Monday
- **School year:** 2023/2024
- **Start time:** 2:30 PM
- **Duration:** 1 hour

[Information about the user Emily Brown, emily.brown@example.com]

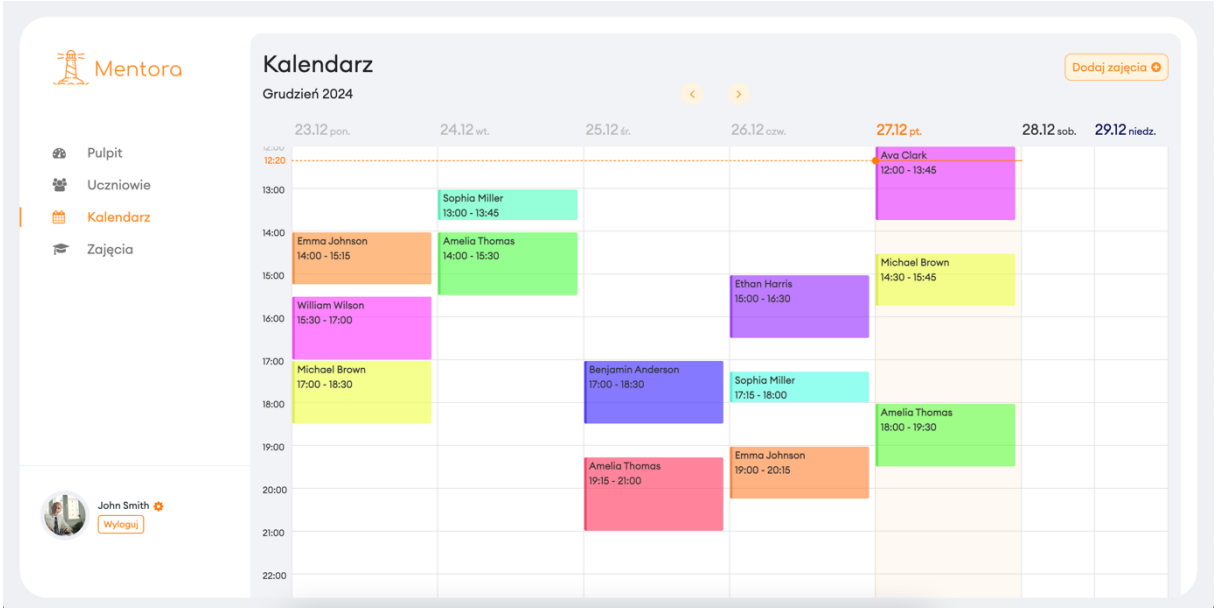
Rysunek 4.15: Wiadomość mailowa wysyłana do rodzica po utworzeniu nowych zajęć w harmonogramie
źródło: opracowanie własne

Rodzic zostaje również poinformowany wiadomością mailową w przypadku wprowadzenia zmian harmonogramu przez nauczyciela. System zapewnia automatyczne i stałe informowanie rodzica o wszystkich najważniejszych zmianach bez konieczności tworzenia konta dla opiekuna.

4.2.3 Harmonogram zajęć

Po utworzeniu zajęć zostają one automatycznie dodane do harmonogramu i są widoczne w widoku kalendarza użytkownika. Nauczyciel ma możliwość swobodnego przeglądania różnych tygodni za pomocą nawigacji kalendarza, co pozwala na szybkie sprawdzenie harmonogramu zajęć w różnych okresach. Ta funkcjonalność zapewnia stały dostęp do planu zajęć oraz umożliwia lepsze zarządzanie czasem oraz elastyczne dostosowanie harmonogramu do zmieniających się okoliczności.

Na tym widoku nauczyciel posiada również dostęp do szybkiego planowania nowych zajęć za pomocą formularza przedstawionego w poprzednim podrozdziale.



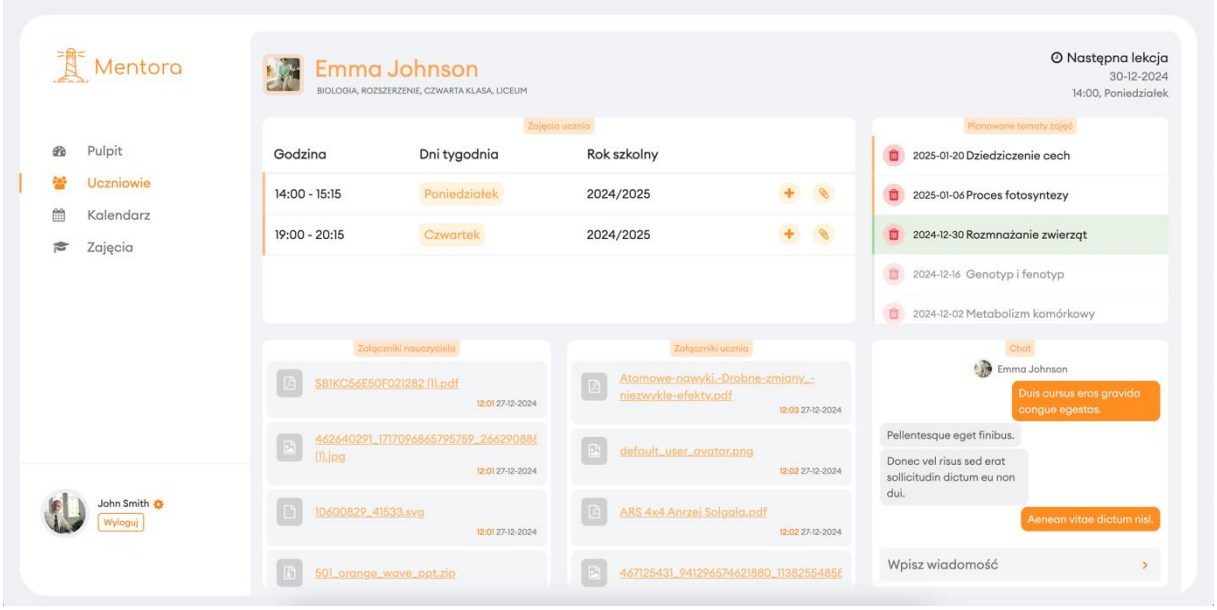
Rysunek 4.16: Kalendarz nauczyciela
źródło: opracowanie własne

Kalendarz wyposażony jest w dynamiczną pionową linię, która na bieżąco wskazuje aktualny moment w ciągu dnia. Linia ta, wraz z wyświetlaną godziną, pozwala użytkownikowi na łatwe zorientowanie się, która godzina jest w danym momencie, co znacząco ułatwia śledzenie postępu dnia.

Kolory kafelków, które nauczyciel wybiera na etapie tworzenia zajęć, pełnią ważną rolę w organizacji i wizualnej strukturze harmonogramu. Użytkownik może w prosty sposób zidentyfikować, które zajęcia należą do określonych kategorii lub grup, co minimalizuje ryzyko pomyłek i ułatwia planowanie. Kolorystyka wspomaga optyczne rozplanowanie harmonogramu, dzięki czemu cały plan dnia staje się bardziej czytelny.

4.2.4 Panel zajęć

Nauczyciel oraz przypisany do wydarzenia uczeń uzyskują dostęp do panelu zajęć. Ta sekcja systemu stanowi kluczową podstawę komunikacji pomiędzy uczniem i nauczycielem oraz jest zbiorem wszystkich danych i informacji powstałych w procesie uczenia.



Rysunek 4.17: Panel zajęć, perspektywa nauczyciela
źródło: opracowanie własne

Nagłówek panelu zawiera najważniejsze dane ucznia oraz informację o dacie i godzinie najbliższych zajęć.

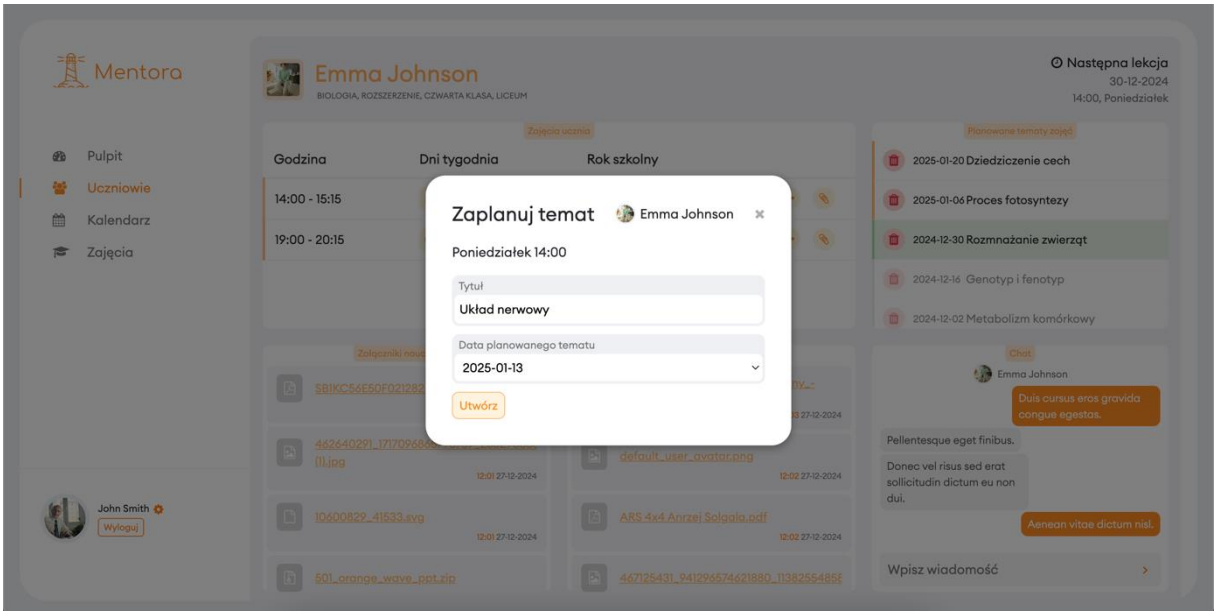
Panel składa się z pięciu odrębnych modułów biorących udział w gromadzeniu i wyświetlaniu informacji: lista zajęć, lista planowanych tematów zajęć, lista załączników nauczyciela, lista załączników ucznia oraz chat.

Lista zajęć

Pierwszy moduł przedstawia listę wszystkich prowadzonych zajęć wybranego ucznia wraz z informacją o godzinie, dniu tygodnia oraz roku szkolnym w którym te zajęcia się odbywają. Nauczyciel ma dostęp do dwóch przycisków funkcyjnych przy każdych zajęciach. Pierwszy przycisk umożliwia zaplanowanie tematu zajęć, drugi natomiast służy do dodawania załączników nauczyciela dostępnych w panelu.

Lista planowanych tematów

Nauczyciel może wprowadzać tematy, które będą realizowane podczas nadchodzących zajęć, co umożliwia lepszą organizację pracy zarówno nauczyciela, jak i uczniów. System pozwala na łatwe dodawanie lub usuwanie tematów, co daje elastyczność w dostosowywaniu planu do potrzeb edukacyjnych. Wprowadzane tematy zajęć tworzą w rezultacie historię nauczania, obrazują ścieżkę procesu edukacyjnego oraz umożliwiają śledzenie postępów ucznia.



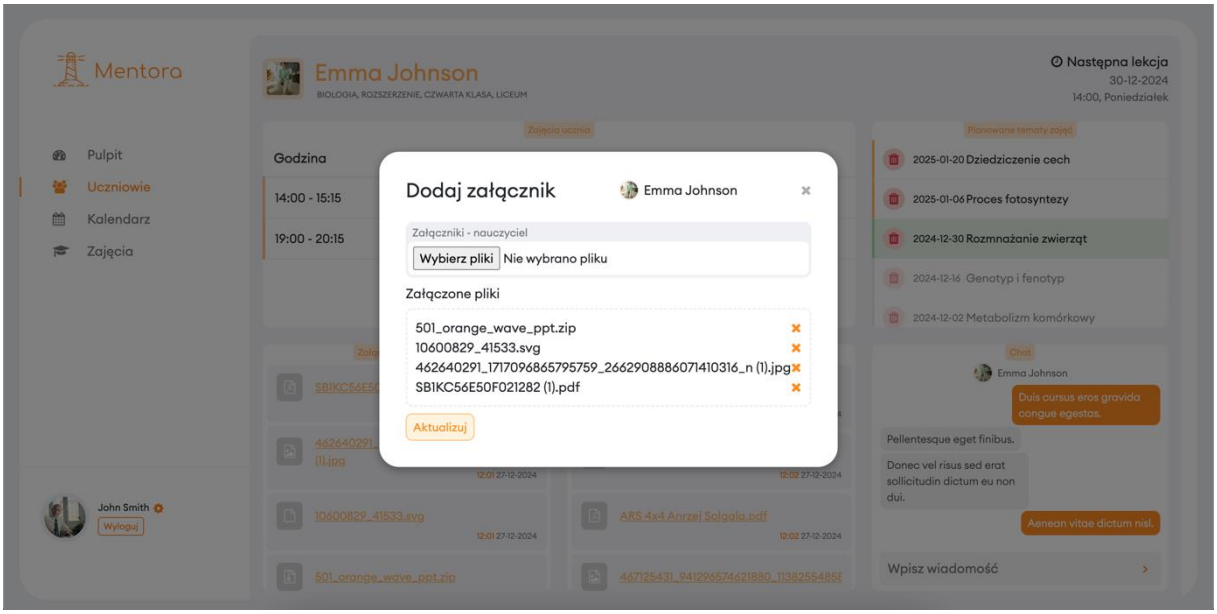
Rysunek 4.18: Formularz wyświetlany podczas planowania tematu zajęć
źródło: opracowanie własne

Dodane tematy sortowane są na podstawie daty zajęć. Tematy zajęć, które już się odbyły oznaczane są na liście kolorem szarym, temat najbliższych zajęć oznaczony jest kolorem zielonym tak by przyciągał uwagę nauczyciela oraz ucznia, tematy zaplanowane w przyszłości są oznaczane kolorem pomarańczowym.

Rysunek 4.19: Lista planowanych tematów zajęć
źródło: opracowanie własne

Lista załączników nauczyciela

Nauczyciel może załączyć pliki zawierające materiały dydaktyczne do każdych zajęć. Udostępnione załączniki mogą zostać pobrane zarówna przez nauczyciela jak i ucznia. Rezultatem tej funkcjonalności jest uporządkowana biblioteka materiałów dostępnych dla użytkowników bez ograniczeń.



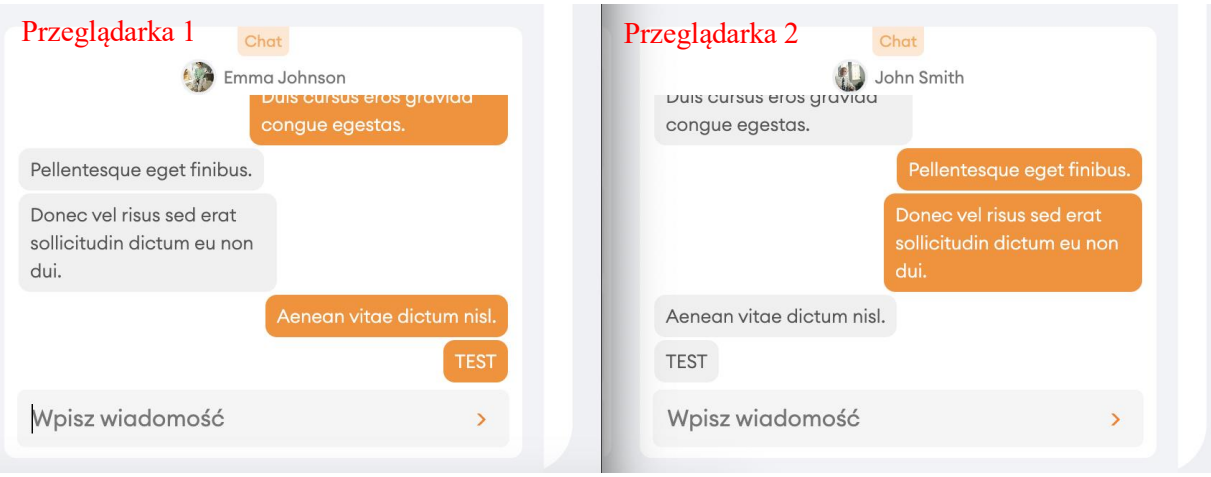
Rysunek 4.20: Formularz wyświetlany podczas modyfikowania załączników nauczyciela
źródło: opracowanie własne

Lista załączników ucznia

Nauczyciel posiada dostęp do załączników dodawanych przez ucznia z możliwością pobrania zasobów. Funkcjonalność ta zapewnia łatwy dostęp do materiałów przesłanych przez ucznia mogą to być prace domowe, projekty czy inne pliki, które są niezbędne do oceny postępów ucznia.

Chat

System zapewnia ułatwioną komunikację pomiędzy użytkownikami za pomocą chatu znajdującego się w panelu zajęć. Konwersacja zostaje utworzona automatycznie po przyznaniu dostępu do panelu a wiadomości pomiędzy użytkownikami są przesyłane w sposób dynamiczny, bez konieczności przeładowania zasobów systemu. Wszystkie wiadomości są przechowywane w bazie danych, przez co możliwe jest przeglądanie historycznych wiadomości w konwersacji.

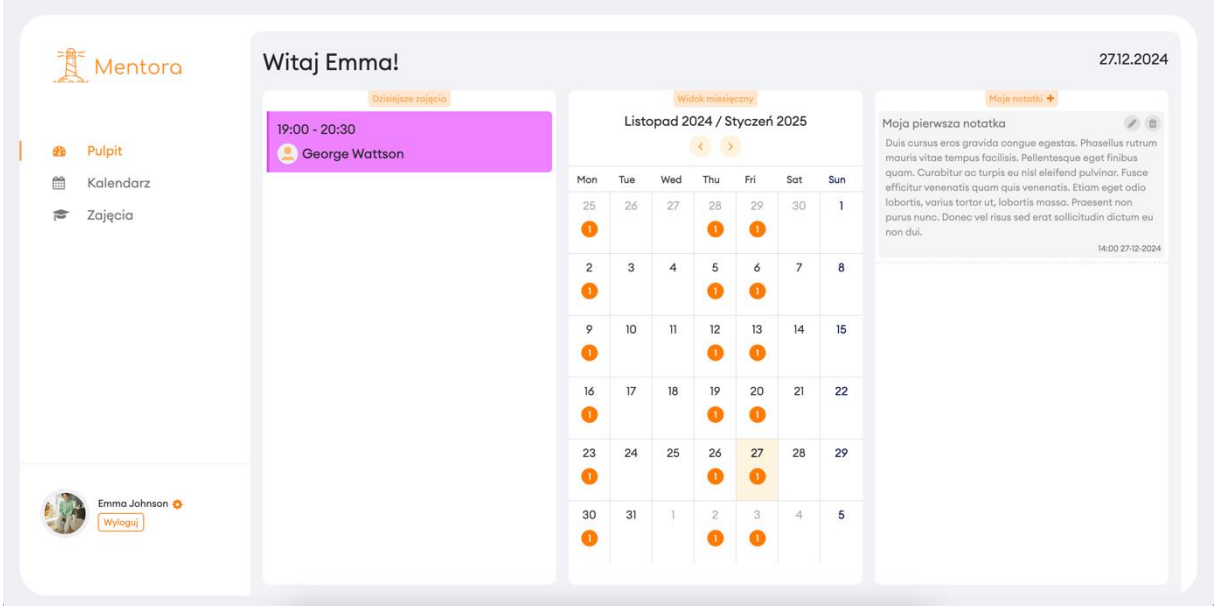


Rysunek 4.21: Konwersacja użytkowników systemu
źródło: opracowanie własne

W celu sprawdzenia dynamiczności konwersacji użytkowników, system został włączony w dwóch oddzielnych przeglądarkach. W przeglądarce pierwszej zalogowany został uczeń, natomiast w drugiej nauczyciel. Po wysłaniu wiadomości z konta ucznia zostaje ona wyświetlona w czasie rzeczywistym w oknie drugiej przeglądarki na koncie nauczyciela.

4.3 Konto ucznia

Konto ucznia zostaje utworzone w wyniku procesu wykonanego przez nauczyciela oraz aktywowane po potwierdzeniu przez użytkownika za pomocą systemowej wiadomości mailowej. Uczeń uzyskuje dostęp do funkcjonalności systemu oraz narzędzi umożliwiających nadzorowanie swojego harmonogramu komunikację z nauczycielami oraz dostęp do wszelkich zasobów edukacyjnych udostępnianych w aplikacji.

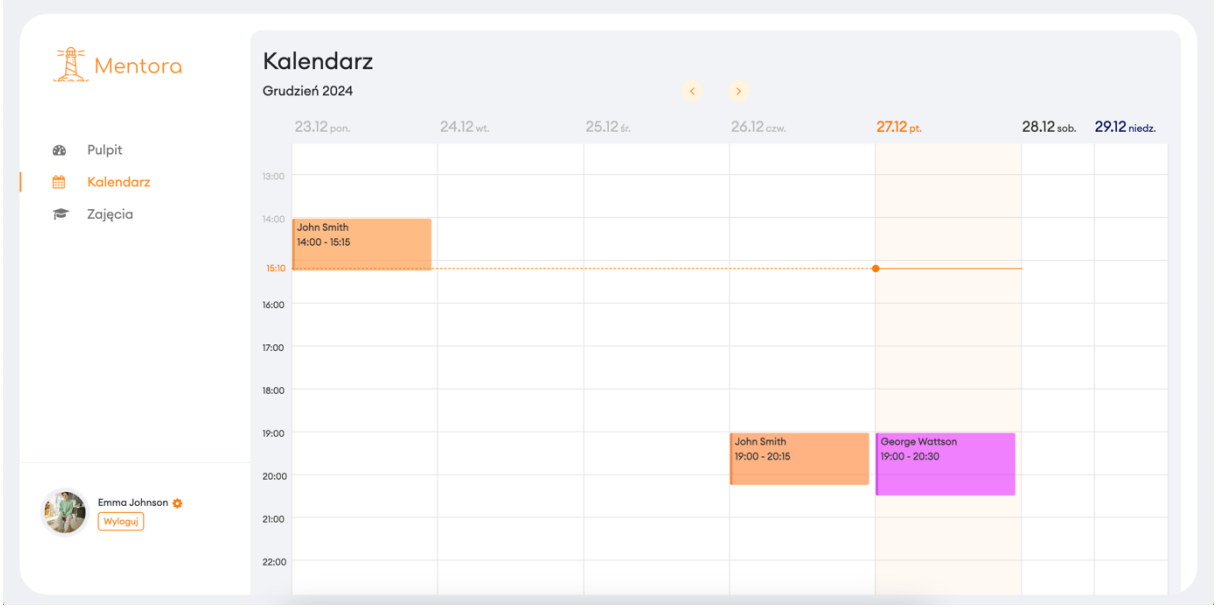


Rysunek 4.22: Pulpit konta ucznia
źródło: opracowanie własne

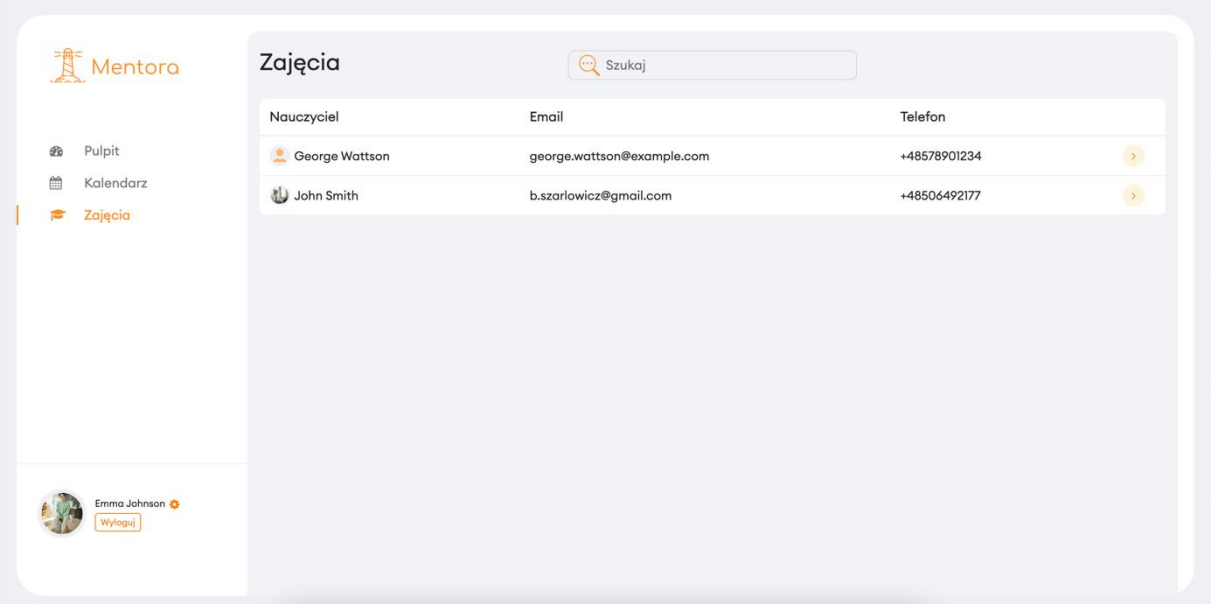
Po zalogowaniu się uczeń ma dostęp do pulpitu użytkownika. Ten widok jest analogicznym widokiem do pulpitu nauczyciela i składa się z takich samych sekcji. Uczeń ma szybki dostęp do trzech kluczowych sekcji: **lista dzisiejszych zajęć**, **miesięczny widok harmonogramu** oraz **prywatne notatki**.

4.3.1 Harmonogram zajęć

Uczeń ma również dostęp do omawianego wcześniej kalendarza zajęć, gdzie w prosty i przejrzysty sposób może nadzorować swój plan tygodnia. Na kalendarzu wyświetlane są wszystkie zajęcia, do których zalogowany użytkownik jest przypisany. Zajęcia te mogą być dodawane i modyfikowane wyłącznie z poziomu konta nadzorującego, czyli konta nauczyciela. Uczeń nie ma możliwości ingerencji w techniczne aspekty zajęć.



Rysunek 4.23: Kalendarz ucznia
źródło: opracowanie własne

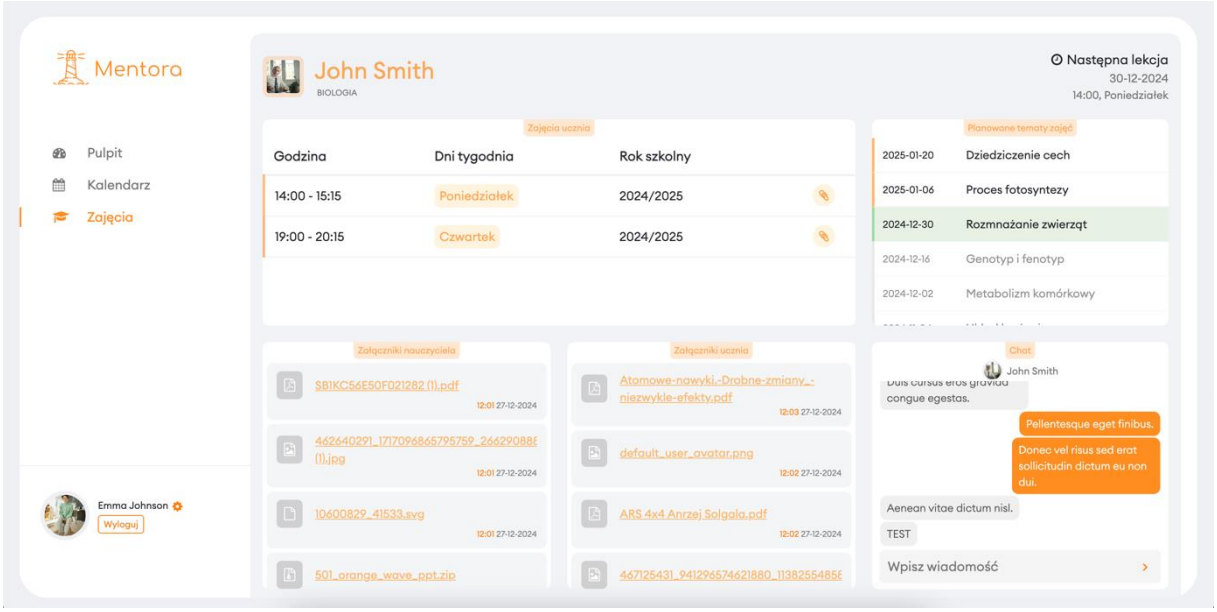


Rysunek 4.24: Lista wszystkich zajęć ucznia
źródło: opracowanie własne

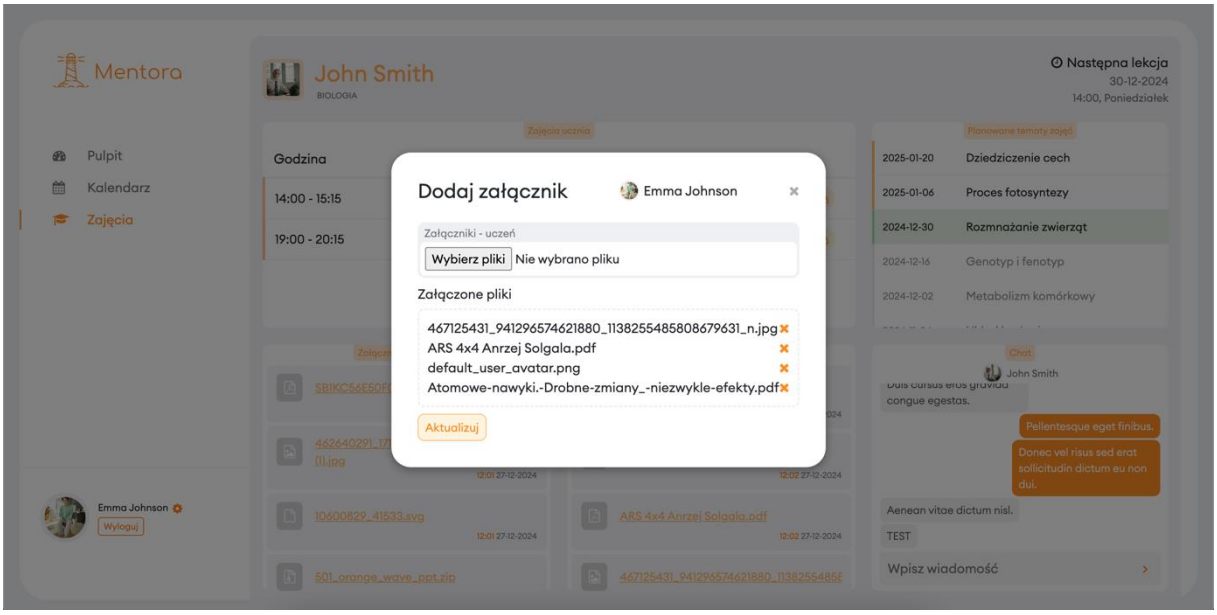
Prezentowane konto użytkownika jest połączone z dwoma nauczycielami. System został zaprojektowany w sposób umożliwiający uczniowi połączenie i uczęszczanie na zajęcia prowadzone przez różnych nauczycieli. Taka struktura wspiera elastyczność w edukacji, umożliwiając współpracę z kilkoma nauczycielami jednocześnie, a także ułatwia organizację i zarządzanie harmonogramem ucznia w systemie.

4.3.2 Panel zajęć

Uczeń uzyskuje dostęp do panelu zajęć utworzonych przez nauczyciela. W perspektywie ucznia panel wygląda analogicznie do panelu nauczyciela, ale jest pozbawiony niektórych przycisków funkcyjnych.



Rysunek 4.25: Panel zajęć, perspektywa ucznia
źródło: opracowanie własne

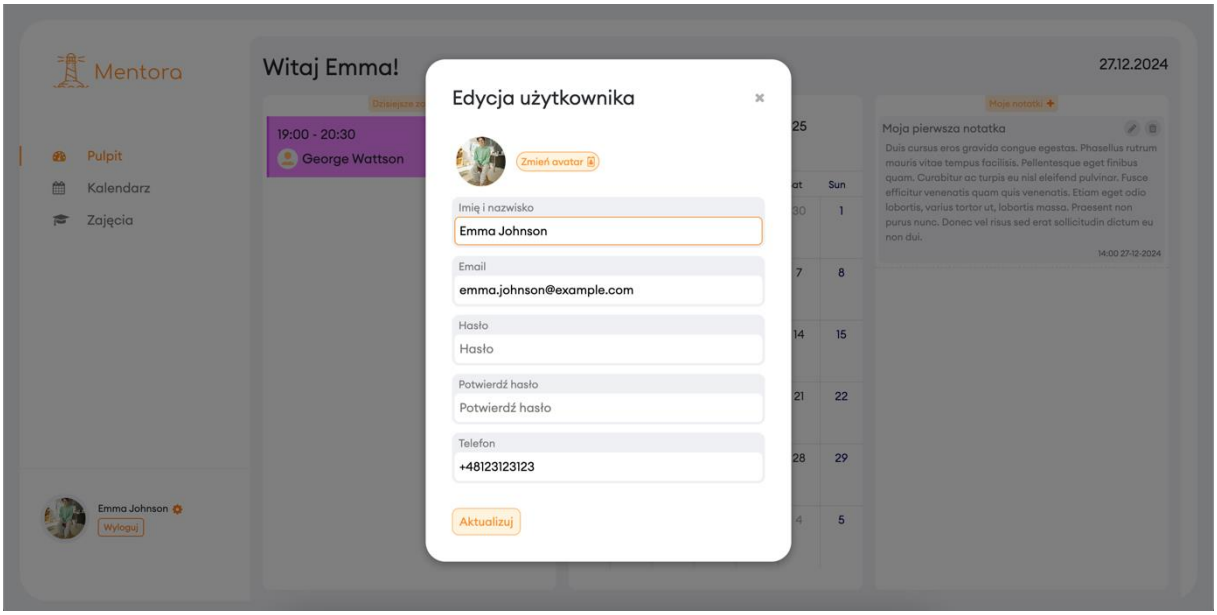


Rysunek 4.26: Formularz wyświetlany podczas modyfikowania załączników ucznia
źródło: opracowanie własne

Uczeń ma możliwość przeglądania listy zajęć oraz listy planowanych tematów, bez możliwości wykonywania akcji modyfikujących. Posiada dostęp do materiałów załączonych przez nauczyciela z możliwością ich pobrania oraz ma możliwość załączenia własnych plików, które będą dostępne dla nauczyciela. Dynamiczny chat umożliwia komunikacje użytkowników w czasie rzeczywistym.

4.4 Modyfikacja danych profilu

Wszystkie rodzaje kont w systemie posiadają możliwość modyfikacji danych konta np. danych dostępowych lub danych kontaktowych. Użytkownik systemu jest również uprawniony do wyboru zdjęcia profilowego, które będzie widoczne dla wszystkich powiązanych użytkowników.



Rysunek 4.27: Formularz wyświetlany podczas modyfikowania danych profilu użytkownika]
źródło: opracowanie własne

Zmiana hasła powoduje automatyczne wylogowanie użytkownika, zmuszając go do ponownego zalogowania za pomocą nowych danych dostępowych. Wiadomość mailowa zawierająca link aktywacyjny nie zostaje wysłana, dzięki czemu użytkownik nie musi ponownie przechodzi przez rejestracyjne procesy techniczne.

Rozdział 5

Podsumowanie

W niniejszym rozdziale dokonane zostanie podsumowanie pracy nad przedstawionym projektem oraz omówione zostaną potencjalne obszary dalszego rozwoju i doskonalenia systemu.

5.1 Stopień realizacji założeń

Kluczowym elementem realizacji projektu było zdefiniowanie początkowych założeń projektowych, które kierunkowały cały proces tworzenia oprogramowania i wspierały podejmowanie odpowiednich decyzji w trakcie prac. Głównymi założeniami projektu były opracowanie i implementacja systemu ułatwiającego wszystkie procesy i czynności związane z prowadzeniem i uczestnictwem w dodatkowych zajęciach edukacyjnych. Połączenie trzech grup docelowych: uczniów, nauczycieli oraz rodziców w jednej aplikacji, oferującej zestaw funkcjonalności ułatwiających pracę nauczyciela i ucznia stanowiło fundament całego projektu.

W tym kontekście wszystkie założenia funkcjonalne zostały zrealizowane w pełnym zakresie. System umożliwia rejestrację użytkowników, w tym tworzenie kont dla uczniów i nauczycieli, automatyczne informowanie mailowe rodzica oraz zapewnia intuicyjny interfejs wspierający zarządzanie harmonogramem zajęć, historią nauczania oraz materiałami dydaktycznymi a także usprawnia komunikację między użytkownikami systemu za pomocą dynamicznej konwersacji tekstowej.

Dodatkowo, na podstawie opinii i sugestii użytkowników testowych, którzy byli reprezentantami grupy docelowej nauczycieli, w projekcie zostało wdrożonych szereg funkcjonalności mających na celu ulepszenie i dostosowanie aplikacji do ich potrzeb. Opinie nauczycieli pozwoliły na precyzyjne określenie niezbędnych funkcji i nadały odpowiedni priorytet podejmowanym pracom.

Podsumowując stopień realizacji założeń można ocenić jako wysoki. Wszystkie początkowe założenia zostały zrealizowane a w trakcie prac po konsultacjach z docelową grupą użytkowników zostały wprowadzone dodatkowe funkcje mające bardzo duży wpływ na optymalizacje pracy nauczyciela i funkcjonalność systemu.

5.2 Napotkane problemy i ich rozwiązanie

Mechanizm dodawania nowego ucznia

Konto ucznia tworzone jest na żądanie nauczyciela, automatycznie przez system. Proces ten został tak skonstruowany w celu ograniczenia ingerencji ucznia w procedurę rejestracji konta. Dzięki temu uczeń unika standardowych procedur rejestracyjnych, co pozwala na szybszy i bardziej efektywny dostęp do platformy.

Podczas implementacji tego mechanizmu wystąpił problem związany z dodawaniem tego samego ucznia do systemu przez dwóch różnych nauczycieli. System początkowo nie przewidywał takiej sytuacji, co skutkowało błędami przy próbie dodania konta, ponieważ podejmowana była próba założenia drugiego konta na istniejący w systemie adres email.

Problem ten został rozwiązany poprzez wdrożenie procedury sprawdzającej, czy konto o podanym adresie email istnieje już w bazie danych. Procedura ta włączana jest, gdy nauczyciel prześle formularz z danymi nowego ucznia i akcja **create** w **StudentController** zostanie wywołana.

```
def create
  @search_url = students_path
  @title = Student.model_name.human(count: 2)

  @students = @teacher.students
  @search = @students.ransack(params[:q])

  @student = find_or_initialize_student(params[:student])

  respond_to do |format|
    if @student.persisted?
      @teacher.students << @student
      create_conversation(@teacher, @student)
      flash[:notice] = flash_message(:create, User)
      format.turbo_stream
      flash.discard
    else
      if @student.save
        @teacher.students << @student
        create_conversation(@teacher, @student)
        send_temporary_password_email(@student)
        flash[:notice] = flash_message(:create, User)
        format.turbo_stream
        flash.discard
      else
        format.html { render :new, status: :unprocessable_entity }
      end
    end
  end
end
```

Rysunek 5.1: Akcja create w StudentController zawierająca procedurę sprawdzającą
źródło: opracowanie własne

```ruby
def find_or_initialize_student(params)
  email = params[:email]
  search_for_existing_student(email) || build_new_student
end

def search_for_existing_student(email)
  Student.find_by(email: email)
end

def build_new_student
  student = Student.new(student_params)
  password = generate_random_password(12)
  student.password = password
  student.password_confirmation = password
  student
end

def send_temporary_password_email(student)
  UserMailer.temporary_password(student, student.password).deliver_now
end
```

Rysunek 5.2: Funkcje pomocnicze biorące udział w procedurze sprawdzającej
źródło: opracowanie własne

Pierwszym krokiem w procesie dodawania nowego ucznia jest wywołanie funkcji pomocniczej **find_or_initialize_student**, do której przekazujemy jako parametr hash[8] zawierający dane ucznia wprowadzone przez nauczyciela w formularzu. Funkcja ta wyodrębnia z hasha adres e-mail ucznia i zapisuje go do zmiennej.

Następnie wywoływana jest kolejna funkcja pomocnicza, **search_for_existing_student**, do której przekazujemy jako parametr adres e-mail ucznia. Jej zadaniem jest przeszukanie bazy danych w celu sprawdzenia, czy istnieje już konto ucznia o podanym adresie e-mail. Jeśli konto takie już istnieje, funkcja zwróci obiekt modelu Student, w przeciwnym razie zwróci wartość nil.

Jeżeli funkcja **search_for_existing_student** zwróciła obiekt modelu Student, funkcja **find_or_initialize_student** przekazuje go dalej i zostaje on zapisany w zmiennej **@student** w kontrolerze.

Jeżeli konto ucznia z takim adresem email nie istnieje, zostaje wywołana funkcja pomocnicza **build_new_student**, która tworzy nowy obiekt modelu Student, oraz tymczasowe hasło i przekazuje do zmiennej **@student**.

Następnie dokonujemy weryfikacji, czy obiekt przechowywany w zmiennej **@student** jest już zapisany w bazie danych. Jeśli obiekt jest zapisany, oznacza to, że konto ucznia istnieje w systemie, a zatem wystarczy je powiązać z kontem nauczyciela.

W przeciwnym przypadku, gdy konto ucznia nie istnieje, należy najpierw utworzyć nowe konto, a dopiero potem połączyć je z kontem nauczyciela i przesłać wiadomości email, zawierające link aktywacyjny oraz tymczasowe, automatycznie wygenerowane hasło.

[8] Hash - *to kolekcja przypominająca słownik, zawierająca unikalne klucze i odpowiadające im wartości. Często nazywana jest również tablicą asocjacyjną. Jest podobna do tablicy (Array), ale podczas gdy tablica używa liczb całkowitych jako indeksów, Hash pozwala na użycie dowolnego typu obiektu. Hash enumeruje swoje wartości w kolejności, w jakiej klucze zostały dodane.* […][25]

Różnicowanie typów użytkowników w systemie

Początkowo rodzaje kont w systemie definiowane były za pomocą pola **role** w modelu User, które pełniło rolę prostego identyfikatora określającego, rodzaj konta w systemie. Wartość tego pola była przypisywana podczas rejestracji użytkownika, a następnie wykorzystywana w logice aplikacji do kontrolowania dostępu do różnych funkcji i zasobów systemu. Jednak takie podejście miało swoje ograniczenia, szczególnie w kontekście bardziej skomplikowanych relacji i funkcjonalności specyficznych dla różnych ról.

Pierwszym rozwiązaniem tej kwestii było usunięcie modelu User i stworzenie oddzielnych modeli dla każdej roli w systemie: Teacher oraz Student. Takie podejście okazało się problematyczne, ponieważ wymagało rozdzielenia mechanizmu rejestracji i logowania, co znacznie komplikowało proces zakładania konta. W efekcie, dostęp do systemu stał się mniej intuicyjny, a zarządzanie kontami bardziej skomplikowane, zarówno pod względem technicznym, jak i użytkowym.

Ostatecznym rozwiązaniem jest zastosowanie wzorca projektowego Single Table Inheritance.

STI (ang. Single Table Inheritance) to wzorzec projektowy bazy danych, w którym wiele modeli jest przechowywanych w jednej tabeli bazy danych. Każdy model jest reprezentowany przez jeden wiersz w tabeli, a specjalna kolumna, zazwyczaj nazywana **type**, służy do określenia konkretnego typu modelu. Implementacja STI w aplikacji Ruby on Rails polega na stworzeniu klasy nadrzędnej, która zawiera wspólne atrybuty i zachowania dla wszystkich klas podrzędnych. Każda klasa podrzędna dziedziczy po tej klasie nadrzędnej i definiuje swoje unikalne atrybuty i metody. [26]

```
class User < ApplicationRecord
```

Rysunek 5.3: Definicja modelu User (klasa nadrzędna)
źródło: opracowanie własne

```
class Teacher < User
```

Rysunek 5.4: Definicja modelu Teacher (klasa podrzędna)
źródło: opracowanie własne

```
class Student < User
```

Rysunek 5.5: Definicja modelu Student (klasa podrzędna)
źródło: opracowanie własne

```ruby
create_table "users", charset: "utf8mb3", force: :cascade do |t|
  t.string "email", default: "", null: false
  t.string "encrypted_password", default: "", null: false
  t.string "reset_password_token"
  t.datetime "reset_password_sent_at"
  t.datetime "remember_created_at"
  t.integer "sign_in_count", default: 0, null: false
  t.datetime "current_sign_in_at"
  t.datetime "last_sign_in_at"
  t.string "current_sign_in_ip"
  t.string "last_sign_in_ip"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.string "confirmation_token"
  t.datetime "confirmed_at"
  t.datetime "confirmation_sent_at"
  t.string "unconfirmed_email"
  t.string "name"
  t.string "phone"
  t.string "role_mask"
  t.boolean "disabled", default: false
  t.string "type"
  t.string "school_name"
  t.index ["confirmation_token"], name: "index_users_on_confirmation_token", unique: true
  t.index ["email"], name: "index_users_on_email", unique: true
  t.index ["reset_password_token"], name: "index_users_on_reset_password_token", unique: true
  t.index ["role_mask"], name: "index_users_on_role_mask"
end
```

Rysunek 5.6: Schemat modelu User, zawierający pole type
źródło: opracowanie własne

Zastosowanie STI upraszcza schemat bazy danych, konsolidując podobne modele w jednej tabeli, co redukuje liczbę tabel i nadmiarowość. Umożliwia łatwiejsze utrzymanie aplikacji, ponieważ zmiany w wspólnych atrybutach i metodach można wprowadzać w jednym miejscu tzn. w klasie nadrzędnej. STI promuje ponowne użycie kodu, zwiększa wydajność zapytań i minimalizuje ryzyko błędów oraz sprawdza się najlepiej, gdy modele mają wspólne cechy.

5.3 Kierunki rozwoju systemu

Rozwój systemu jest kluczowym elementem zapewniającym jego długotrwałą użyteczność oraz zdolność do spełniania rosnących wymagań użytkowników. Udoskonalenia bezpieczeństwa oraz intuicyjności aplikacji to kluczowe aspekty rozbudowy platformy. Takie działania mają na celu podniesienie jakości doświadczeń użytkowników i wzmocnienie pozycji systemu jako niezawodnego narzędzia wspierającego codzienną pracę.

Jednym z kierunków rozwoju systemu jest wprowadzenie zaawansowanych funkcji networkingowych[9], które będą umożliwiać uczniom łatwiejsze wyszukiwanie nauczycieli w zależności od ich specjalizacji, lokalizacji lub preferowanego trybu nauczania. W tym celu planowane jest utworzenie szczegółowych, publicznych profili nauczycieli, zawierających informacje takie jak doświadczenie zawodowe, dostępne terminy, opinie innych użytkowników, prowadzone przedmioty, poziom nauczania. Mechanizm ten będzie centralnym punktem procesu nawiązywania kontaktu między uczniami a nauczycielami, ułatwiając wybór najlepszego kandydata do współpracy. Dodatkowo system umożliwiałby przeprowadzanie kilkuminutowej rozmowy wstępnej pomiędzy uczniem a nauczycielem, w ten sposób zwiększyłaby się satysfakcja użytkownika systemu, poprzez świadome podjęcie ostatecznej decyzji o wyborze nauczyciela.

W celu zwiększenia bezpieczeństwa danych użytkowników oraz spełniania wszystkich standardów ochrony prywatności, jednym z proponowanych kierunków rozwoju systemu jest wprowadzenie mechanizmu anonimizacja historycznych wiadomości tekstowych użytkowników. Taki proces polegałby na przekształceniu danych w sposób uniemożliwiający identyfikację konkretnej osoby, przy jednoczesnym zachowaniu ich struktury w celach analitycznych lub archiwizacyjnych. Takie rozwiązanie stanowczo ograniczyłoby zagrożenia wynikające z ewentualnego wycieku danych lub nieautoryzowanego dostępu do systemu.

Stały kontakt z docelową grupą użytkowników jest kluczowym aspektem rozwoju oraz umożliwia wprowadzanie kolejnych niezbędnych funkcjonalności do systemu. Udostępnianie wczesnych wersji systemu użytkownikom testowym pełni niezwykle istotną rolę w procesie zbierania opinii i sugestii. Opinie testerów umożliwiają identyfikacje obszarów wymagających dalszej optymalizacji oraz przyczyniają się do systematycznego udoskonalania systemu. Dzięki zaangażowaniu użytkowników w proces projektowania i testowania, możliwe jest budowanie systemu, który nie tylko spełnia wysokie standardy technologiczne, ale także odpowiada na specyficzne potrzeby grupy docelowej, zwiększając tym samym jego użyteczność i satysfakcję odbiorców.

[9] Networking – Wymiana informacji lub usług między osobami, grupami lub instytucjami, szczególnie: kultywowanie produktywnych relacji w celu zatrudnienia lub prowadzenia działalności gospodarczej. [27]

5.4 Wnioski końcowe

Niniejsza praca przedstawia problemy związane ze świadczeniem prywatnych usług edukacyjnych oraz analizuje wyzwania, które pojawiają się w kontekście organizacji, zarządzania i optymalizacji czynności związanych z powyższymi usługami. W wyniku prac oraz diagnostyki wszelkich aspektów zauważonych problemów powstało kompleksowe narzędzie łączące trzy grupy docelowe: uczniów, nauczycieli oraz rodziców. System ten umożliwia efektywne zarządzanie edukacyjnymi zajęciami pozaszkolnymi oraz zapewnia płynność komunikacji między grupami użytkowników. Stworzona aplikacja została opisana z uwzględnieniem technicznych rozwiązań wykorzystanych w procesie jej tworzenia oraz omówione zostały kluczowe aspekty implementacji użytych narzędzie i technologii, dzięki czemu prezentowana praca stanowi źródło wiedzy o charakterze edukacyjnym. Ponadto, przedstawiono potencjalne kierunki rozwoju systemu, zmierzające do jego udoskonalenia oraz dostosowania go do rosnących potrzeb użytkowników.

Bibliografia

[1] Lech Madeyski, Mirosław Ochodek, *Inżynieria oprogramowania, Badania i praktyka,* Nakom, 2014

[2] https://wolski.pro/diagramy-uml/diagram-przypadkw-uzycia/ [data dostępu: 2024-12-16]

[3] https://opensource.org/osd [data dostępu: 2024-12-30]

[4] https://guides.rubyonrails.org/v7.1/index.html [data dostępu: 2024-12-18]

[5] Rebecca Wirfs-Brock, Alan McKean: Object design: roles, responsibilities, and collaborations. Addison-Wesley, 2002

[6] https://hotwired.dev/ [data dostępu: 2024-12-21]

[7] Thomas D., Programming Ruby 1.9. The Pragmatic Programmers' Guide, 2009

[8] https://1stplace.pl/blog/czym-jest-baza-danych-mysql-pierwsze-kroki-w-zarzadzaniu-bazami-danych/ [data dostępu: 2024-12-20]

[9] https://getbootstrap.com/docs/5.3/getting-started/introduction/ [data dostępu: 2024-12-30]

[10] https://redis.io/docs/latest/ [data dostępu: 2024-12-30]

[11] https://github.com/heartcombo/devise [data dostępu: 2024-12-30]

[12] https://github.com/CanCanCommunity/cancancan [data dostępu: 2024-12-30]

[13] https://github.com/activerecord-hackery/ransack [data dostępu: 2024-12-30]

[14] https://github.com/nhn/tui.calendar [data dostępu: 2024-12-30]

[15] https://github.com/ice-cube-ruby/ice_cube [data dostępu: 2024-12-30]

[16] https://github.com/slim-template/slim [data dostępu: 2024-12-30]

[17] https://github.com/heartcombo/simple_form [data dostępu: 2024-12-30]

[18] https://github.com/kaminari/kaminari [data dostępu: 2024-12-30]

[19] https://github.com/rails/jquery-rails [data dostępu: 2024-12-30]

[20] https://github.com/tsechingho/chosen-rails [data dostępu: 2024-12-30]

[21] https://fontawesome.com/v4/icons/ [data dostępu: 2024-12-30]

[22] https://github.com/minimagick/minimagick [data dostępu: 2024-12-30]

[23] https://github.com/daddyz/phonelib [data dostępu: 2024-12-30]

[24] https://blog.postman.com/what-is-an-api-endpoint/ [data dostępu: 2024-12-30]

[25] https://ruby-doc.org/core-2.7.5/Hash.html [data dostępu: 2024-12-27]

[26] https://medium.com/nyc-ruby-on-rails/single-table-inheritance-sti-ruby-on-rails-basics-2a08c81e50af [data dostępu: 2024-12-27]

[27] https://www.merriam-webster.com/dictionary/networking [data dostępu: 2024-12-30]