**Specyfikacja systemu zarządzania projektami studenckimi v1.1**

**1. Cel i opis systemu**

**System ma na celu stworzenie centralnej platformy umożliwiającej studentom publikowanie i zarządzanie swoimi projektami, a nauczycielom ich weryfikację i akceptację. Projekty będą uporządkowane według kategorii tematycznych, a każdy użytkownik z odpowiednimi uprawnieniami będzie miał dostęp do ich edycji lub oceny.**

**System zapewni intuicyjny interfejs użytkownika, który umożliwi łatwe wyszukiwanie i przeglądanie projektów bez konieczności ich pobierania na komputer. Logowanie do systemu będzie realizowane wyłącznie przez MS Teams i Active Directory uczelni, co zapewni bezpieczeństwo i kontrolę nad dostępem. Publikacja projektów wymaga zatwierdzenia przez nauczycieli, co zapewni wysoką jakość udostępnianych materiałów.**

**Uzasadnienie: Centralizacja zarządzania projektami studenckimi w jednym systemie znacząco usprawni proces nauczania i oceniania prac studentów. Integracja z istniejącymi systemami uczelnianymi (MS Teams, Active Directory) eliminuje potrzebę tworzenia nowych kont i haseł, a proces weryfikacji przez nauczycieli zapewnia kontrolę jakości materiałów.**

**2. Wymagania techniczne**

**2.1. Technologie**

**Frontend**

* **Framework: Vue.js 3 (Composition API)**
* **CSS Framework: Tailwind CSS**
* **Zarządzanie stanem: Pinia/Vuex**
* **Routing: Vue Router**

**Uzasadnienie: Vue.js zapewnia lekkość, elastyczność oraz łatwą krzywą uczenia się. Composition API wprowadzone w Vue 3 umożliwia tworzenie bardziej modułowego i czytelnego kodu. Tailwind CSS pozwala na szybkie tworzenie spójnego i responsywnego interfejsu poprzez komponowanie klas CSS.**

**Backend**

* **Framework: Node.js + NestJS**
* **ORM: Prisma**
* **API: RESTful**

**Uzasadnienie: NestJS dostarcza solidną strukturę do budowy skalowalnych aplikacji serwerowych w TypeScript. Wspiera modułową architekturę i dependency injection, co ułatwia organizację kodu. Prisma ORM zapewnia type-safe zapytania i ułatwia migracje bazy danych.**

**Baza danych**

* **System: PostgreSQL**

**Uzasadnienie: PostgreSQL to solidna, relacyjna baza danych o otwartym kodzie, która doskonale sprawdza się w aplikacjach z kompleksowym modelem danych, oferując zaawansowane możliwości zapytań i integralność danych.**

**Autoryzacja**

* **Logowanie wyłącznie przez MS Teams i Active Directory uczelni**
* **Wielopoziomowy system dostępu (Administrator, Nauczyciel, Student, Gość)**

**Uzasadnienie: Integracja z MS Teams i Active Directory uczelni eliminuje potrzebę tworzenia osobnego systemu rejestracji i logowania. Użytkownicy mogą korzystać z tych samych danych uwierzytelniających, których używają na co dzień, co upraszcza dostęp do aplikacji i zwiększa bezpieczeństwo.**

**Infrastruktura**

* **Hostowanie: Maszyna wirtualna w akademii bialskiej**
* **Konteneryzacja: Docker**
* **Zabezpieczenia: HTTPS, certyfikat SSL**
* **Dostępność: Przez adres WWW**

**Uzasadnienie: Konteneryzacja z Docker zapewnia jednolite środowisko dla rozwoju, testowania i produkcji. HTTPS i certyfikat SSL zapewniają bezpieczną komunikację i zwiększają zaufanie użytkowników do systemu.**

**3. Architektura systemu**

**3.1. Schemat architektury**

**System jest zbudowany w architekturze trójwarstwowej:**

1. **Warstwa prezentacji (frontend) - aplikacja SPA w Vue.js**
2. **Warstwa logiki biznesowej (backend) - API NestJS**
3. **Warstwa danych (baza danych) - PostgreSQL**

**Uzasadnienie: Podział na trzy warstwy pozwala na niezależny rozwój każdej z nich, zwiększa modularność i ułatwia testowanie. Jest to sprawdzony standard w nowoczesnych aplikacjach webowych, który umożliwia skalowanie poszczególnych komponentów w zależności od potrzeb.**

**3.2. Komunikacja między warstwami**

* **Frontend komunikuje się z backendem poprzez RESTful API**
* **Backend komunikuje się z bazą danych poprzez Prisma ORM**
* **Autoryzacja wykorzystuje MS Teams i Active Directory**

**Uzasadnienie: RESTful API zapewnia standaryzowany sposób komunikacji między frontendem a backendem, niezależny od technologii. Prisma ORM znacząco upraszcza operacje na bazie danych i zapewnia typebezpieczne zapytania.**

**4. Moduły systemu**

**4.1. Moduł Użytkowników**

**Role użytkowników**

* **Administrator: Pełny dostęp do systemu, zarządzanie użytkownikami, generowanie raportów**
* **Nauczyciel: Weryfikacja i zatwierdzanie/odrzucanie projektów, zapewnienie kontroli jakości**
* **Student: Tworzenie i edycja własnych projektów przed zatwierdzeniem**
* **Gość: Tylko przeglądanie zatwierdzonych projektów**

**Uzasadnienie: System ról i uprawnień jest kluczowy dla kontroli dostępu do różnych funkcjonalności aplikacji. Podział na administratorów, nauczycieli, studentów i gości odpowiada rzeczywistym rolom w środowisku akademickim i zapewnia odpowiedni poziom kontroli nad treściami.**

**Funkcjonalność**

* **Logowanie wyłącznie przez MS Teams / Active Directory uczelni (bez rejestracji)**
* **Zarządzanie profilami użytkowników**
* **Kontrola dostępu oparta na rolach (RBAC)**
* **Bezpieczeństwo i kontrola nad dostępem do systemu**

**Uzasadnienie: Integracja z istniejącymi systemami uczelnianymi eliminuje potrzebę pamiętania dodatkowych danych logowania. Role bazujące na RBAC (Role-Based Access Control) zapewniają precyzyjną kontrolę nad tym, kto może wykonywać poszczególne operacje w systemie.**

**4.2. Moduł Projektów**

**Funkcjonalność**

* **Tworzenie nowych projektów (student)**
* **Edycja projektów przed zatwierdzeniem (student)**
* **Weryfikacja i akceptacja/odrzucenie projektów (nauczyciel)**
* **Przeglądanie projektów z możliwością filtrowania (wszyscy)**
* **Szczegółowy widok projektu z pełnym opisem i metadanymi**
* **Przeglądanie projektów bez konieczności ich pobierania**
* **Automatyczne kategoryzowanie projektów według tematyki**

**Uzasadnienie: Centralna funkcjonalność systemu, umożliwiająca zarządzanie całym cyklem życia projektu - od utworzenia, przez recenzję, aż po publikację. Możliwość przeglądania projektów bez pobierania oszczędza miejsce na dysku użytkowników i zwiększa dostępność materiałów.**

**Struktura projektu**

* **Tytuł**
* **Opis**
* **Kategoria**
* **Data utworzenia**
* **Autor**
* **Status (nowy, w weryfikacji, zatwierdzony, odrzucony)**
* **Pliki/załączniki**
* **Komentarze od nauczyciela**
* **Tagi**

**Uzasadnienie: Struktura projektu została zaprojektowana tak, aby zawierać wszystkie niezbędne informacje do skutecznego zarządzania i wyszukiwania. Status umożliwia śledzenie etapu projektu, a metadane ułatwiają kategoryzację i wyszukiwanie.**

**4.3. Moduł Administracyjny**

**Funkcjonalność**

* **Zarządzanie użytkownikami (dodawanie, edycja, usuwanie)**
* **Zarządzanie kategoriami projektów**
* **Moderacja treści**
* **Generowanie raportów (popularność projektów, aktywność użytkowników)**

**Uzasadnienie: Moduł administracyjny zapewnia narzędzia do zarządzania całym systemem, pozwalając administratorom na kontrolę użytkowników, kategorii i treści. Generowanie raportów umożliwia analizę wykorzystania systemu i podejmowanie decyzji na podstawie danych, np. identyfikację najbardziej aktywnych studentów lub najpopularniejszych kategorii projektów.**

**5. Model danych**

**5.1. Główne encje**

1. **User (Użytkownik)** 
   * **id: UUID**
   * **email: String (unikalny)**
   * **firstName: String**
   * **lastName: String**
   * **role: Enum (ADMIN, TEACHER, STUDENT, GUEST)**
   * **adUserId: String (ID w Active Directory)**
   * **msTeamsId: String (ID w MS Teams)**
   * **createdAt: DateTime**
   * **updatedAt: DateTime**
2. **Project (Projekt)** 
   * **id: UUID**
   * **title: String**
   * **description: Text**
   * **status: Enum (NEW, IN\_REVIEW, APPROVED, REJECTED)**
   * **createdAt: DateTime**
   * **updatedAt: DateTime**
   * **publishedAt: DateTime (opcjonalnie)**
   * **authorId: UUID (relacja do User)**
   * **approverId: UUID (relacja do User, opcjonalnie)**
   * **categoryId: UUID (relacja do Category)**
3. **Category (Kategoria)** 
   * **id: UUID**
   * **name: String (unikalna)**
   * **description: Text (opcjonalnie)**
   * **parentId: UUID (relacja do nadrzędnej kategorii, opcjonalnie)**
   * **createdAt: DateTime**
   * **updatedAt: DateTime**
4. **File (Plik)** 
   * **id: UUID**
   * **filename: String**
   * **originalName: String**
   * **mimeType: String**
   * **size: Integer**
   * **path: String**
   * **createdAt: DateTime**
   * **projectId: UUID (relacja do Project)**
5. **Comment (Komentarz)** 
   * **id: UUID**
   * **content: Text**
   * **createdAt: DateTime**
   * **updatedAt: DateTime**
   * **authorId: UUID (relacja do User)**
   * **projectId: UUID (relacja do Project)**
6. **Tag (Tag)** 
   * **id: UUID**
   * **name: String (unikalna)**
   * **createdAt: DateTime**
7. **ProjectTag (Relacja Projekt-Tag)** 
   * **projectId: UUID (relacja do Project)**
   * **tagId: UUID (relacja do Tag)**
8. **Report (Raport)** 
   * **id: UUID**
   * **name: String**
   * **type: String**
   * **parameters: JSON (opcjonalnie)**
   * **result: JSON (opcjonalnie)**
   * **createdAt: DateTime**

**Uzasadnienie: Rozbudowany model danych z wieloma powiązanymi encjami umożliwia przechowywanie wszystkich niezbędnych informacji o projektach, użytkownikach i ich interakcjach. Hierarchiczna struktura kategorii pozwala na logiczną organizację projektów, a model tagów zapewnia elastyczne grupowanie. Przechowywanie plików i komentarzy bezpośrednio w modelu danych upraszcza zarządzanie zasobami.**

**5.2. Relacje między encjami**

* **User 1 -> N Project (autor)**
* **User 1 -> N Project (zatwierdzający)**
* **User 1 -> N Comment**
* **Project N -> 1 Category**
* **Project 1 -> N File**
* **Project 1 -> N Comment**
* **Project N -> N Tag (poprzez ProjectTag)**

**Uzasadnienie: Relacje między encjami odzwierciedlają naturalne zależności w systemie. Użytkownik może być autorem wielu projektów, a nauczyciel może zatwierdzać wiele projektów. Projekty mogą należeć tylko do jednej kategorii, ale mogą mieć wiele plików, komentarzy i tagów.**

**6. Interfejs użytkownika**

**6.1. Ogólny układ**

* **Górny pasek nawigacyjny:** 
  + **Logo systemu (po lewej)**
  + **Linki: "Strona Główna", "Baza Projektów", "Kontakt", "O nas"**
  + **Wyszukiwarka projektów (po środku)**
  + **Panel logowania / menu użytkownika (po prawej)**

**Uzasadnienie: Spójny układ interfejsu z nawigacją na górze strony jest zgodny z oczekiwaniami użytkowników i standardami UX. Umieszczenie wyszukiwarki w centralnym miejscu podkreśla jej znaczenie w systemie zorientowanym na wyszukiwanie i przeglądanie projektów.**

**6.2. Główne widoki**

1. **Strona główna:** 
   * **Lista najnowszych projektów**
   * **Wyszukiwarka projektów**
   * **Kategorie projektów**
2. **Baza projektów:** 
   * **Zaawansowane filtrowanie (kategoria, data, autor, status)**
   * **Lista projektów z podstawowymi informacjami**
   * **Paginacja wyników**
3. **Szczegóły projektu:** 
   * **Pełny opis projektu**
   * **Metadane (autor, data, kategoria)**
   * **Załączniki**
   * **Komentarze**
   * **Przyciski akcji (zależne od roli)**
4. **Panel użytkownika:** 
   * **Lista własnych projektów (dla studentów)**
   * **Projekty do weryfikacji (dla nauczycieli)**
   * **Panel administracyjny (dla administratorów)**
5. **Formularz projektu:** 
   * **Pola do wprowadzania danych projektu**
   * **Możliwość dodawania załączników**
   * **Przyciski zapisz/anuluj**

**Uzasadnienie: Główne widoki zostały zaprojektowane tak, aby pokryć wszystkie kluczowe przypadki użycia systemu. Strona główna zapewnia szybki dostęp do najważniejszych funkcji, baza projektów umożliwia zaawansowane wyszukiwanie, a dedykowane widoki dla projektów i użytkowników dostarczają szczegółowych informacji i funkcji specyficznych dla roli.**

**6.3. Responsywność**

* **Interfejs w pełni responsywny (mobile-first)**
* **Dostosowany do różnych rozmiarów ekranów (smartfon, tablet, komputer)**

**Uzasadnienie: Responsywny interfejs zapewnia dostęp do systemu z dowolnego urządzenia, co jest kluczowe w środowisku akademickim, gdzie użytkownicy korzystają z różnorodnych urządzeń.**

**7. Bezpieczeństwo**

**7.1. Autoryzacja i uwierzytelnianie**

* **Integracja z MS Teams i Active Directory uczelni**
* **Tokeny JWT do autoryzacji API**
* **Role i uprawnienia oparte na RBAC**

**Uzasadnienie: Wykorzystanie istniejących mechanizmów uwierzytelniania uczelni zwiększa bezpieczeństwo i upraszcza proces logowania. Tokeny JWT zapewniają bezstatusową autoryzację API, a RBAC precyzyjnie kontroluje dostęp do funkcji systemu.**

**7.2. Zabezpieczenia na poziomie infrastruktury**

* **HTTPS z certyfikatem SSL**
* **Bezpieczne zarządzanie sekretami**
* **Konteneryzacja z Docker**

**Uzasadnienie: HTTPS i SSL zapewniają szyfrowanie komunikacji między klientem a serwerem. Konteneryzacja izoluje poszczególne komponenty systemu, a bezpieczne zarządzanie sekretami chroni wrażliwe dane konfiguracyjne.**

**7.3. Ochrona danych**

* **Walidacja danych wejściowych**
* **Ochrona przed atakami CSRF, XSS, SQL Injection**
* **Logowanie zdarzeń i monitorowanie**

**Uzasadnienie: Kompleksowa ochrona danych obejmuje walidację wszystkich danych wejściowych, zabezpieczenia przed popularnymi atakami webowymi oraz monitorowanie systemu w celu szybkiego wykrywania potencjalnych naruszeń bezpieczeństwa.**

**8. Etapy rozwoju**

**8.1. Etap 1: Konfiguracja podstawowa**

* **Inicjalizacja projektu Vue.js i NestJS**
* **Konfiguracja Tailwind CSS**
* **Podstawowa struktura bazy danych**
* **Konfiguracja kontenera Docker**

**Uzasadnienie: Pierwszy etap skupia się na przygotowaniu środowiska, co jest kluczowe dla sprawnego rozwoju projektu. Obejmuje konfigurację wszystkich technologii i narzędzi, które będą wykorzystywane w dalszych etapach.**

**8.2. Etap 2: Implementacja modułów podstawowych**

* **Moduł Użytkowników (logowanie, role)**
* **Moduł Projektów (CRUD)**
* **Podstawowy interfejs**

**Uzasadnienie: Drugi etap koncentruje się na implementacji podstawowej funkcjonalności, która stanowi fundament systemu. Obejmuje zarządzanie użytkownikami, podstawowe operacje na projektach oraz interfejs użytkownika.**

**8.3. Etap 3: Rozbudowa funkcjonalności**

* **Zaawansowane wyszukiwanie i filtrowanie**
* **Zatwierdzanie projektów**
* **Panel administracyjny**

**Uzasadnienie: Trzeci etap rozbudowuje podstawową funkcjonalność o zaawansowane funkcje, które zwiększają użyteczność systemu. Obejmuje mechanizmy wyszukiwania, procesu zatwierdzania projektów oraz narzędzia administracyjne.**

**8.4. Etap 4: Integracja i wdrożenie**

* **Integracja z MS Teams i Active Directory**
* **Konfiguracja HTTPS i SSL**
* **Testy i optymalizacja**

**Uzasadnienie: Ostatni etap skupia się na integracji z systemami uczelnianymi, zabezpieczeniach oraz testach i optymalizacji przed wdrożeniem produkcyjnym. Jest to kluczowe dla zapewnienia bezpieczeństwa i wydajności systemu.**

**9. Podział zadań w zespole**

**9.1. Frontend Developer**

* **Tworzenie struktury aplikacji Vue.js**
* **Implementacja komponentów UI z Tailwind CSS**
* **Integracja z API backendu**
* **Zapewnienie responsywności**

**Uzasadnienie: Frontend Developer odpowiada za całość interfejsu użytkownika, który jest kluczowy dla akceptacji systemu przez użytkowników. Wymagana jest znajomość Vue.js, Tailwind CSS oraz integracji z API.**

**9.2. Backend Developer**

* **Projektowanie i implementacja API NestJS**
* **Integracja z bazą danych PostgreSQL przez Prisma**
* **Implementacja logiki biznesowej**
* **Integracja z MS Teams i Active Directory**

**Uzasadnienie: Backend Developer odpowiada za serwerową część aplikacji, która przetwarza dane i implementuje logikę biznesową. Wymagana jest znajomość NestJS, Prisma i integracji z zewnętrznymi systemami.**

**9.3. DevOps/Administrator**

* **Konfiguracja infrastruktury (maszyna wirtualna)**
* **Konfiguracja Docker, HTTPS, SSL**
* **Monitorowanie aplikacji**
* **Zarządzanie backupami i aktualizacjami**

**Uzasadnienie: DevOps/Administrator odpowiada za infrastrukturę i wdrożenie systemu, co jest kluczowe dla jego dostępności i bezpieczeństwa. Wymagana jest znajomość Docker, HTTPS/SSL oraz administracji systemami.**

**9.4. Product Owner/Tester**

* **Definiowanie wymagań**
* **Testowanie funkcjonalności**
* **Koordynacja rozwoju**
* **Dokumentacja**

**Uzasadnienie: Product Owner/Tester zarządza wymaganiami projektu i zapewnia jakość poprzez testowanie. Jest to kluczowe dla dostarczenia systemu zgodnego z oczekiwaniami użytkowników.**

**9.5. UI/UX Designer**

* **Projektowanie interfejsu w Figma**
* **Optymalizacja UX**
* **Współpraca z frontend developerem**

**Uzasadnienie: UI/UX Designer odpowiada za wizualną i funkcjonalną stronę interfejsu, co bezpośrednio wpływa na satysfakcję użytkowników. Wymagana jest znajomość zasad projektowania interfejsów i narzędzi takich jak Figma.**

**10. Harmonogram prac**

**10.1. Faza początkowa (2 tygodnie)**

* **Przygotowanie środowiska**
* **Prototypowanie UI/UX**
* **Projektowanie bazy danych**

**Uzasadnienie: Faza początkowa koncentruje się na planowaniu i przygotowaniu, co jest kluczowe dla powodzenia projektu. Obejmuje projektowanie architektury, interfejsu i bazy danych.**

**10.2. Faza rozwoju (8 tygodni)**

* **Implementacja podstawowej funkcjonalności**
* **Integracja komponentów**
* **Testy jednostkowe**

**Uzasadnienie: Faza rozwoju jest najdłuższa i obejmuje implementację wszystkich funkcjonalności systemu. Równoległe testy jednostkowe zapewniają jakość na wczesnym etapie.**

**10.3. Faza finalna (4 tygodnie)**

* **Integracja z MS Teams i Active Directory**
* **Testy końcowe**
* **Wdrożenie produkcyjne**
* **Dokumentacja**

**Uzasadnienie: Faza finalna koncentruje się na integracji, testach końcowych i wdrożeniu. Kompletna dokumentacja jest kluczowa dla przyszłego utrzymania i rozwoju systemu.**