POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Numer zlecenia i nazwa i akronim projektu: *EduPlaner*	**Zleceniodawca:** *Polsko-Japońska Akademia Technik Komputerowych*	**Zleceniobiorca:** *PJATK*
Zespół projektowy: *Czerwiński Cyprian* *Koniak Kamil* *Małolepszy Michał* *Bastek Aleksander*	**Kierownik projektu:** *Cyprian Czerwiński*	**Opiekun projektu:** *dr hab. inż. Marta Łabuda*
Nazwa dokumentu: *Wstępny Plan Projektu*	**Odpowiedzialny za dokument:** *{nazwisko, imię}*	

Historia dokumentu

Wersja	Opis modyfikacji	Rozdział / strona	Autor modyfikacji	Data
1.0	*Wstępna wersja*	*całość*	*Cały zespół projektowy*	*06.11.2024 r.*
{wersja}	*{np. poprawka wstępnego opisu}*	*{np. punkty 3.3 i 4}*	*{nazwisko, imię}*	*{data zmiany}*

Opis etapu: Celem etapu jest określenie założeń projektu (cele, zakres, ograniczenia, priorytety), przedstawienie wizji docelowego rozwiązania – kształtu systemu i strategii procesu wytwórczego oraz opisanie poszczególnych etapów, przedstawienie harmonogramu prac projektowych i konstrukcyjnych.

Oczekiwane produkty: Wstępny Plan Projektu w postaci dokumentu o nazwie GRX(XYZ)-WPP-WER1.doc o strukturze według poniższego szablonu. W czasie jego tworzenia należy wykorzystać dostępne w laboratoriach lub freeware narzędzia CASE.

1 Założenia projektu

{Klient i problemy biznesowe; Krótki opis projektu, jego uwarunkowania i priorytety}

Klient i problemy biznesowe:

Klientami są szkoły i uczelnie wyższe, które zmagają się z problemem czasochłonnego i mało efektywnego planowania grafików zajęć. Obecnie harmonogramy tworzone są głównie ręcznie, przy użyciu tablic lub arkuszy kalkulacyjnych, co powoduje, że proces ten jest podatny na błędy i trudności w uwzględnieniu preferencji nauczycieli oraz studentów. Z uwagi na ogromną liczbę możliwych kombinacji – mogącą sięgać 10^12k – ręczne tworzenie harmonogramów staje się niepraktyczne i nie pozwala na optymalizację przy uwzględnieniu preferencji zainteresowanych stron.

Krótki opis projektu:

Projekt zakłada opracowanie systemu opartego na technologii PlanningAI, który automatycznie generuje optymalne grafiki zajęć dla instytucji edukacyjnych. System uwzględnia zarówno dostępność nauczycieli, jak i preferencje uczniów/studentów, minimalizując konflikty i zapewniając efektywne wykorzystanie zasobów. Dzięki zaawansowanym algorytmom sztucznej inteligencji, system będzie w stanie przetworzyć olbrzymią liczbę możliwych harmonogramów i wygenerować optymalne rozwiązanie w krótkim czasie.

Uwarunkowania:

Realizacja projektu wymaga współpracy z przedstawicielami instytucji edukacyjnych, aby zrozumieć szczegółowe wymagania związane z planowaniem zajęć oraz ograniczenia czasowe i przestrzenne. System musi być skalowalny i łatwy do wdrożenia w różnorodnych instytucjach o różnych wymaganiach. Ważne jest również zapewnienie, że system będzie mógł być użytkowany bez zaawansowanego przeszkolenia technicznego.

Priorytety:

Zautomatyzowanie procesu planowania zajęć w szkołach i uczelniach.
Uwzględnienie preferencji nauczycieli i studentów, aby zwiększyć satysfakcję użytkowników.
Zoptymalizowanie czasu pracy i zasobów instytucji edukacyjnych poprzez minimalizację konfliktów i efektywne przydzielanie sal oraz zasobów.

1.1.Cele i zakres projektu

{Zakres projektu – odróżnić od zakresu systemu}

Celem projektu jest opracowanie i wdrożenie inteligentnego systemu planowania grafików zajęć dla instytucji edukacyjnych, który:

1. Usprawni proces planowania harmonogramów, zmniejszając czas potrzebny na ich przygotowanie.
2. Zapewni lepsze dopasowanie grafików do preferencji nauczycieli i studentów.
3. Zoptymalizuje wykorzystanie sal lekcyjnych oraz innych zasobów uczelni.
4. Zredukuje konflikty harmonogramowe.
5. Zwiększy satysfakcję użytkowników systemu.

Zakres projektu:

Projekt obejmuje etapy od analizy wymagań przez projektowanie i implementację aż po testowanie oraz wdrożenie systemu w instytucjach edukacyjnych. Obejmuje także szkolenie personelu z obsługi systemu oraz zapewnienie wsparcia technicznego na etapie wdrożenia i początkowego użytkowania. Kluczowe etapy projektu to:

Analiza wymagań:

Zebranie wymagań od różnych grup użytkowników (nauczyciele, administracja, studenci), aby system spełniał potrzeby każdej z grup.

Projektowanie systemu:

Stworzenie architektury systemu opartego na PlanningAI, który umożliwi przetwarzanie dużej liczby kombinacji harmonogramowych.

Implementacja i integracja:

Opracowanie głównych funkcji systemu, takich jak generowanie grafików, uwzględnianie preferencji użytkowników, optymalizacja zasobów i integracja z innymi narzędziami (np. systemami zarządzania uczelnią).

Testowanie i walidacja:

Przeprowadzenie testów w celu sprawdzenia, czy system generuje optymalne harmonogramy zgodnie z założeniami oraz czy jest łatwy w obsłudze.

1.2.Spodziewane produkty

{Produkty projektu}

System planowania grafików oparty na PlanningAI:

Gotowy do wdrożenia system, który automatycznie generuje harmonogramy zajęć dla szkół i uczelni wyższych, uwzględniając preferencje nauczycieli i studentów oraz optymalizując wykorzystanie zasobów.

Dokumentacja techniczna, kompletny zestaw dokumentów zawierający:

- Opis architektury systemu.
  - CD - diagram klas (class diagram)
  - CMPD -diagram komponentów (component diagram)
  - PD- diagram pakietów (package diagram)

    - DEPD- diagram rozmieszczenia (wdrożenia) (deployment diagram)
    - DBD lub ERD- diagram bazy danych
- Szczegóły dotyczące algorytmów optymalizacji harmonogramów.
- Instrukcje instalacji i konfiguracji.

Dokumentacja użytkownika:

Przewodnik dla użytkowników końcowych systemu (administracja, nauczyciele), zawierający instrukcje dotyczące obsługi systemu oraz wyjaśnienia kluczowych funkcji, takich jak wprowadzanie preferencji, generowanie grafików i zarządzanie zmianami.

1.3.Interesariusze

{Adresaci, użytkownicy, inni interesariusze}

Adresaci:

- **Polsko-Japońska Akademia Technik Komputerowych (PJATK):**

Instytucja edukacyjna zlecająca projekt, której przedstawiciele, w tym rektor i wicerektorzy, nadzorują realizację projektu i akceptację planów.

Użytkownicy systemu:

- **Administratorzy:** Osoby odpowiedzialne za tworzenie i edycję planów zajęć, mogą to być członkowie administracji uczelni lub wyznaczeni planiści. Maksymalna liczba administratorów wynosi 10.
- **Nauczyciele:** Prowadzą zajęcia, mają dostęp do odczytu planu lekcji oraz swojego grafiku, mogą również komunikować się ze studentami. Maksymalna liczba nauczycieli to 100.
- **Studenci:** Uczestnicy procesu kształcenia, posiadają dostęp do przeglądu swojego planu lekcji oraz mogą otrzymywać powiadomienia od nauczycieli. Maksymalna liczba studentów to 500.

Inni interesariusze:

- **Opiekun projektu:** Marta Łabuda, odpowiada za nadzór nad projektem.
- **Zespół projektowy (EduPlaner):** W skład wchodzi Cyprian Czerwiński, Kamil Koniak, Michał Małolepszy i Aleksander Bastek, odpowiedzialnych za realizację projektu i wdrożenie systemu.
- **Planiści:** Zewnętrzni wykonawcy mogący wspierać tworzenie planów zajęć.
- **Personel techniczny PJATK:** Odpowiada za wsparcie techniczne, zarówno podczas wdrażania, jak i utrzymania systemu.
- **Administracja uczelni:** Współpracuje z zespołem projektowym w organizacji pracy uczelni i wdrażaniu harmonogramów.
- **Regulacje prawne (Prawo o szkolnictwie wyższym i nauce):** Wytyczne i przepisy regulujące projekt, m.in. maksymalne godziny zajęć w tygodniu.

1.4.Uwarunkowania, ograniczenia, założenia strategii

{Czasowe, środowiska, standardy; Inne (dodatkowe zobowiązania)}

Czasowe:

- Projekt musi zostać zakończony do dnia 24.01.2025.
- Realizacja etapów projektu powinna przebiegać zgodnie z harmonogramem, obejmując kluczowe etapy takie jak analiza wymagań, projektowanie, implementacja, testowanie i wdrożenie.
- Prezentacja gotowego produktu odbędzie się 25.01.2025

Środowiskowe:

- System musi działać na różnych środowiskach IT wykorzystywanych przez szkoły i uczelnie, zapewniając dostępność zarówno lokalnie, jak i zdalnie.
- Środowisko produkcyjne powinno być przygotowane do pracy w kontenerach (Docker) w celu ułatwienia skalowania i wdrażania kolejnych wersji. Powinna być zapewniona konteneryzacja:
    - **Backend:** Dostarczenie serwisów backendowych jako osobnych kontenerów pozwala na łatwiejsze zarządzanie i możliwość niezależnej aktualizacji logiki biznesowej.
    - **Frontend:** Konteneryzacja interfejsu użytkownika pozwala na niezależny rozwój frontendu, co skraca czas implementacji poprawek i nowych funkcji.
    - **Bazy danych:** Konteneryzacja bazy danych umożliwia łatwe przenoszenie danych między środowiskami, co przyspiesza testowanie i wdrażanie oraz umożliwia tworzenie kopii zapasowych.

Standardy i regulacje:

- System musi być zgodny z przepisami prawa o szkolnictwie wyższym i nauce, w tym regulacjami dotyczącymi maksymalnej liczby godzin zajęć w tygodniu.

Dodatkowe zobowiązania i wymagania techniczne:

- **Test Driven Development (TDD):** Proces tworzenia systemu powinien opierać się na metodyce TDD, co oznacza, że przed implementacją każdej funkcji najpierw tworzone są testy. TDD pomaga zapewnić wysoką jakość kodu, wczesne wykrywanie błędów oraz ułatwia utrzymanie i rozwój systemu.

1.5.Priorytety

{Wśród celów projektu, priorytet projektu}

- **Nauka procesu inżynierii oprogramowania:** Głównym celem projektu jest umożliwienie zespołowi projektowemu zdobycia wiedzy na temat pełnego cyklu inżynierii oprogramowania, w tym planowania, projektowania, implementacji, testowania i wdrażania systemów informatycznych.
- **Zdobycie praktycznego doświadczenia przez zespół:** Projekt stanowi okazję dla zespołu do pracy nad rzeczywistym problemem, co pozwala na rozwój umiejętności technicznych, zarządzania projektem oraz komunikacji z interesariuszami.
- **Opracowanie funkcjonalnego systemu wspierającego automatyzację planowania grafików:** Chociaż priorytetem jest edukacja zespołu, równie ważne jest dostarczenie działającego systemu, który spełni wymagania PJATK, poprawi efektywność planowania i zmniejszy potrzebę ręcznego zarządzania harmonogramami.
- **Zgodność z założeniami czasowymi i standardami technicznymi:** Wysoka jakość techniczna i realizacja zgodnie z założonym harmonogramem są istotne, aby projekt mógł być wdrożony zgodnie z ustalonym terminem oraz spełniał oczekiwania wszystkich interesariuszy.

1.6. Projekty powiązane i partnerzy zewnętrzni

{Jeśli realizowany projekt jest fragmentem jakieś większego projektu (jego zamkniętą funkcjonalnością, modułem) lub kiedy do jego prawidłowego / pełnego działania współpraca z innymi projektami jest konieczna / wymagana / zalecana}

Projekt aplikacji mobilnej w ramach przedmiotu Interakcja Człowiek-Komputer (ICK):

Równolegle do głównego projektu tworzenia narzędzia do planowania grafików realizowany jest projekt aplikacji mobilnej na potrzeby przedmiotu ICK. Aplikacja ta będzie zintegrowana z systemem planowania i umożliwi użytkownikom, w tym studentom, grupom studentów oraz nauczycielom, przeglądanie swoich harmonogramów. Projekt aplikacji mobilnej zapewnia więc dodatkowy kanał dostępu do informacji o planach zajęć.

2 Analiza ryzyka

{Wskazanie głównych zagrożeń poprzez wskazanie:

- *kontekstu zagrożenia tj. miejsca, w którym zagrożenie może wystąpić np. zespół (w tym umiejętności, doświadczenie, obciążanie innymi zadaniami itp.), proces wytwórczy (w tym budżet, harmonogram, przydzielanie zasobów itp.),*
- *czynników ryzyka,*
- *prawdopodobieństwa wystąpienia,*
- *możliwych skutków zdarzenia oraz ich wpływu na projekt (dotkliwości skutków zdarzenia).*

Należy uwzględnić również zdarzenia, które mają pozytywny wpływ na projekt – szanse, które planujemy wykorzystać. (opisać podobnie jak zagrożenie).
Zalecane przedstawienie w formie tabel(i) lub punktów}

2.1. Ryzyko związane z zespołem (kompetencje, doświadczenie, obciążenie)

Zagrożenie	Kontekst zagrożenia	Czynniki ryzyka	Prawdopodobieństwo	Skutki	Wpływ na projekt
Brak doświadczenia w zespole w zakresie programowania i zarządzania projektem	Zespół developerski, kierownik projektu	Niedostateczne umiejętności, brak doświadczenia w zarządzaniu projektem	Średnie	Opóźnienia w realizacji zadań, błędy w kodzie	Wydłużenie czasu realizacji, możliwe przekroczenie budżetu
Obciążenie zespołu innymi zadaniami	Zespół developerski, menedżerowie	Praca nad innymi projektami, niewystarczająca liczba osób w zespole	Wysokie	Opóźnienia, spadek jakości pracy, stres w zespole	Zmniejszenie tempa realizacji projektu, możliwe opóźnienia
Niska motywacja zespołu	Zespół developerski	Brak jasno określonych celów, niewystarczająca komunikacja, niejasność w zadaniach	Średnie	Niska jakość pracy, nieporozumienia, błędy w implementacji	Spowolnienie pracy nad projektem, niska jakość dostarczonych wyników

2.2. Ryzyko związane z procesem wytwórczym (budżet, harmonogram, przydzielanie zasobów)

Zagrożenie	Kontekst zagrożenia	Czynniki ryzyka	Prawdopodobieństwo	Skutki	Wpływ na projekt
Przekroczenie budżetu	Budżet projektu	Niewłaściwe oszacowanie kosztów, nieprzewidziane wydatki, zmiany w wymaganiach	Średnie	Zwiększenie kosztów, konieczność obniżenia jakości, konieczność rezygnacji z niektórych funkcji	Zmniejszenie zakresu projektu, trudności z finansowaniem
Opóźnienia w realizacji projektu	Harmonogram projektu	Zbyt ambitne terminy, nieprzewidziane trudności w realizacji, zmiany w wymaganiach	Wysokie	Zwiększenie kosztów, frustracja zespołu, obniżenie jakości	Konieczność zmiany terminu realizacji projektu.

Nieodpowiednie przydzielenie zasobów (np. nauczycieli, sal)	Proces przydzielania zasobów do harmonogramu	Niewłaściwe oszacowanie dostępnych zasobów, błędy w planowaniu	Średnie	Konflikty w planowaniu zajęć, nadmiar zadań dla zasobów	Spadek wydajności, niezadowolenie interesariuszy

2.3. Ryzyko technologiczne

Zagrożenie	Kontekst zagrożenia	Czynniki ryzyka	Prawdopodobieństwo	Skutki	Wpływ na projekt
Problemy z obsługą dużej liczby użytkowników	Technologia i wydajność	Obciążenie systemu przy dużej liczbie użytkowników, nieoptymalne zasoby infrastruktury	Średnie	Spadek wydajności, przeciążenia	Konieczność skalowania infrastruktury, dodatkowe koszty
Problemy z wydajnością systemu	Technologie programistyczne, infrastruktura	Niewystarczająca moc obliczeniowa, błędy w kodzie, nieoptymalna struktura bazy danych	Średnie	Wolne działanie systemu, błędy w działaniu funkcji	Zwiększenie wydatków na optymalizacje działania systemu.
Problemy z bezpieczeństwem danych użytkowników	Bezpieczeństwo i dane użytkowników	Niewłaściwe zabezpieczenia danych, błędy w kodzie	Niskie	Utrata danych, naruszenie prywatności. Utrata zaufania użytkowników.	Konieczność wdrożenia nowych zabezpieczeń

2.4. Ryzyko związane z interesariuszami (nauczyciele, dyrektorzy, uczniowie)

Zagrożenie	Kontekst zagrożenia	Czynniki ryzyka	Prawdopodobieństwo	Skutki	Wpływ na projekt
Nieakceptacja systemu przez użytkowników	Użytkownicy systemu (nauczyciele, dyrektorzy, uczniowie)	Brak zaangażowania użytkowników w proces projektowania, niewłaściwe szkolenia	Średnie	Niska adopcja systemu, frustracja użytkowników	Wydłużenie procesu implementacji, konieczność dostosowania systemu
Zmiana wymagań w trakcie realizacji projektu	Interesariusze (dyrektorzy, nauczyciele, uczniowie)	Zmiany w wymaganiach, nowe pomysły lub potrzeby w trakcie realizacji	Wysokie	Opóźnienia w realizacji, konieczność modyfikacji funkcji systemu	Zwiększenie kosztów, zmiana harmonogramu
Brak zaangażowania kluczowych interesariuszy	Nauczyciele, dyrektorzy	Niewłaściwa komunikacja, brak włączenia	Średnie	Niedostosowanie systemu do potrzeb użytkowników,	Zmniejszenie efektywności systemu.

		interesariuszy w proces projektowania		błędne założenia. Zmniejszenie satysfakcji użytkowników.	

2.5. Szanse (pozytywne zdarzenia)

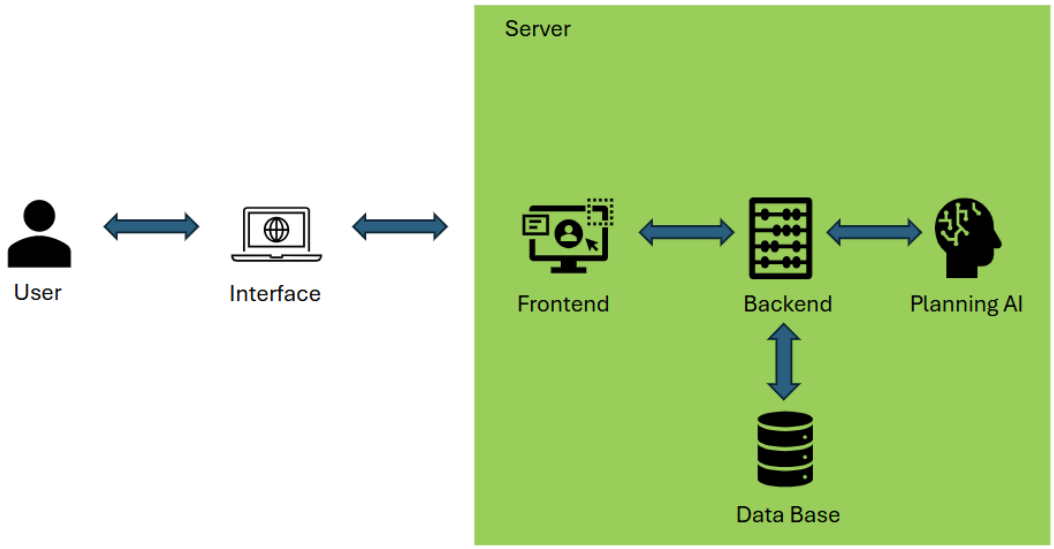
Szansa	**Kontekst zdarzenia**	**Czynniki ryzyka**	**Prawdopodobieństwo**	**Korzyści**	**Wpływ na projekt**
Zwiększenie zaangażowania interesariuszy	Współpraca z nauczycielami i dyrektorami	Regularne konsultacje z interesariuszami, uwzględnienie ich potrzeb	Średnie	Lepsze dopasowanie systemu do potrzeb użytkowników, wyższa jakość produktu	Zmniejszenie ilości poprawek po realizacji projektu.
Dzięki dobrze wdrożonemu systemowi, automatyzacja procesów planowania zajęć	Użycie systemu w wielu szkołach	Pozytywna reakcja na wdrożenie, adaptacja systemu przez szkoły	Wysokie	Zwiększenie efektywności w wielu szkołach, oszczędność czasu	Większe możliwości rozwoju projektu.

Podsumowanie:

W ramach projektu EduPlaner, istnieje szereg zagrożeń związanych z zespołem, procesem wytwórczym, technologią oraz interesariuszami. Warto wziąć pod uwagę zarówno ryzyka negatywne, jak i pozytywne szanse, które mogą wpłynąć na sukces projektu. Kluczowe będzie zaplanowanie działań zapobiegawczych oraz reakcji na nieoczekiwane sytuacje, aby minimalizować ryzyko i wykorzystać szanse.

3 Wizja rozwiązania

{zakładany kształt systemu i organizacja działania}



3.1. Architektura systemu

{opis architektury, wykorzystywane wzorce (np. MVC), frameworki, itp }

System zostanie oparty na architekturze klient-serwer, z wyraźnym podziałem na frontend, backend i bazę danych. Wzorzec projektowy Model-View-Controller (MVC) będzie zastosowany w architekturze frontendu, co umożliwi oddzielenie logiki aplikacji od interfejsu użytkownika.

Frameworki i narzędzia:
Backend: Java z wykorzystaniem Spring Boot, co umożliwi wystawianie endpointów RestAPI.
PlanningAI: wykorzysta framework Timefold, który obsłuży logikę planowania grafików.
Frontend: będzie rozwijany w HTML, CSS, JavaScript oraz prawdopodobnie Svelte dla tworzenia dynamicznego interfejsu użytkownika.

3.2. Rozwiązania techniczne

{kształt systemu, koncepcja, standardy, technologia, dekompozycja systemu na podsystemy, opis interakcji pomiędzy podsystemami }

Kształt systemu:

- System będzie zdekomponowany na trzy główne podsystemy: frontend, backend, i bazę danych.
- Wszystkie te komponenty będą uruchamiane w **kontenerach Docker**, co umożliwi łatwiejsze wdrożenie i zarządzanie systemem.

Dekompozycja i interakcje pomiędzy podsystemami:

- **Backend** (Spring Boot) będzie odpowiedzialny za obsługę logiki biznesowej i wystawianie REST API dla frontendu oraz integrację z modułem planowania PlanningAI (Timefold).
- **Frontend** będzie odpowiadał za prezentację danych użytkownikowi oraz interakcje z backendem poprzez API.
- **Baza danych** (prawdopodobnie MariaDB) przechowa wszystkie dane związane z harmonogramami, użytkownikami oraz preferencjami planowania.

Standardy:

- **CI/CD**: Procesy **Continuous Integration** i **Continuous Deployment** będą zautomatyzowane z użyciem GitHub Actions, aby przyspieszyć testowanie i wdrażanie.

3.3. Technologia i zamierzone środowisko

{docelowe, wytwórcze, języki }

- Kod będzie zarządzany w repozytorium GitHub z wykorzystaniem Git.
- Backend, frontend i baza danych zostaną uruchomione w kontenerach Docker, co zapewni przenośność i skalowalność systemu.
- **Środowisko produkcyjne:** Serwer będzie postawiony na prywatnym komputerze członka zespołu projektowego.
- **Baza danych:** MariaDB jako relacyjna baza danych.
- **Frontend:** HTML, CSS, JavaScript, prawdopodobnie wspierany przez Svelte.
- **Backend**: Java

4 Proces wytwarzania

{dobór strategii wytwarzania (patrz poniższe podpunkty)}

4.1. Stategia prowadzenia prac

{wskazanie wg jakiej strategii będzie prowadzony projekt; pokazanie 1-2 alternatywnych rozwiązań; uwzględnienie ryzyka jakie niesie ze sobą każde z rozwiązań; punkt powinien być zakończony uzasadnionym wyborem strategii prowadzenia prac}

4.2. Proces wytwórczy

{szczegółowe opisanie wybranej strategii procesu, etapy procesu, cykl życia, cele, zadania, powiązania, produkty, miary}

4.2.1. Rozpoznanie

{jeśli projekt realizowany w kilku etapach należy dla każdego etapu określić: cele, produkty etapu, główne zadania, miary oceny i kryteria akceptacji; wskazanie potencjalnych zagrożeń realizacji etapu (skutków, czynników ryzyka, prawdopodobieństwa wystąpienia, działań); zasady zapewnienia jakości i zarządzania.
Jeśli projekt realizowany bez podziału na kilka etapów powyższe musi być wyspecyfikowane tylko jeden raz dla całego projektu.}

Cele:

- Stworzenie wstępnych zadań
- Zapoznanie się z technologiami
- Odkrycie ograniczeń technologicznych
- Stworzenie wstępnego planu projektu

Produkt etapu: Wyszkolenie każdego z członków zespołu do używania części technologii, które potrzebujemy.

Zagrożenia: Różne technologie mają różny próg wejścia. Ktoś z członków zespołu może utknąć w procesie zapoznawania się.

Ocena: Pozytywną miarą sukcesu, będzie wytworzenie podstawowych komponentów w danej technologii. (Np. Endpointu w Spring, lub prostego widoku w Svelte)

4.2.2. Opracowanie planu systemu

Cele i działania:

- Sporządzenie dokumentacji planu systemu
- Stworzenie modelu architektury rozwiązania
- Określenie zadań dotyczących wersji minimalnej

Produkt Etapu: Dokumentacja, w tym projekt systemu rozpisanie planów w Asanie.
Ocena: Dokumentacja i rys architektoniczny mają być na tyle jasne, by można było na ich podstawie określić zadania. Widzialnym efektem ma być rozpisanie zadań w Asanie dotyczących całego systemu, w wersji minimalnej.

Zagrożenia: Stworzenie dokumentacji na tak wysokim poziome abstrakcji, że zadania nie będą mogły zostać sformułowane. Jeżeli tworzenie zadań, będzie czasochłonne jest to znak, że sam plan został źle opracowany.

4.2.3. Stworzenie prototypu

Cele i działania:

- Stworzenie infrastruktury sieciowej – konteneryzacja i baza danych
- Postawienie backendu z timefold.ai
- Integracja widoku timefold.ai z Svelte

Produkt Etapu: Aplikacja działająca na serwerze. Oddzielne kontenery przechowujące backend i frontend. Produkt jest pozbawiony funkcjonalności opisanych w SWS.
Ocena: Działająca aplikacja na serwerze. Zintegrowane dwie części backend i frontend projektu oraz baza.
Zagrożenia: Część zespołu nie będzie miała zajęcia, póki nie powstaną zasadnicze funkcjonalności (np. Jeśli nie ma backendu, nie można pracować jeszcze nad frontendem.)

4.2.4. Stworzenie aplikacji MVP

Cele i działania:

- Zaimplementowanie wszystkich wymaganych funkcjonalności o priorytecie “Must”, w aplikacji.
- Oddanie działającego projektu zgodnego z wymaganiami.

Produkt Etapu: aplikacja z pełnią działających funkcjonalności jw.
Ocena: Przejście testów akceptacyjnych.

Infrastruktura

Technologia, narzędzia, środowisko i ich dostępność:

- **Backend**: System zostanie opracowany przy użyciu technologii Java i frameworka Spring, zapewniając wydajną obsługę mikrousług i optymalizację zarządzania zasobami .

- **Frontend**: Zastosowana zostanie technologia JavaScript najprawdopodobniej z biblioteką React, umożliwiając tworzenie dynamicznych i responsywnych interfejsów.
- **Baza danych**: System będzie oparty na bazie danych SQL, co zapewni stabilność i efektywność przechowywania informacji.
- **Konteneryzacja**: Do konteneryzacji aplikacji zostanie wykorzystany Docker, co pozwoli na odseparowane uruchamianie frontendowej i backendowej części systemu.
- **Infrastruktura serwerowa**: Zabezpieczenie serwerów i połączeń poprzez protokoły szyfrujące w celu ochrony danych użytkowników i integralności systemu.
- **Środowisko użytkowania**: Aplikacja będzie działać w środowisku webowym, z dostępem z różnych urządzeń (PC, tablety, smartfony) oraz przeglądarek internetowych.

4.3. Zakładane zasoby

Czym dysponujemy?:

- Zespół projektowy składający się z czterech programistów: Cyprian Czerwiński, Kamil Koniak, Michał Małolepszy, Aleksander Bastek.
- Opiekun projektu: dr hab. inż. Marta Łabuda, mgr. inż. Grzegorz Cysewski

Co jest nieodzowne?:

- **Serwery**: Serwery do hostowania aplikacji.
- **Narzędzia programistyczne**: Kompilatory, środowiska programistyczne (IDE), narzędzia do testowania.
- **Środki zabezpieczające**: Oprogramowanie zabezpieczające dane i komunikację. Base 64, Keytool.

Co musimy pozyskać?:

- **Budżet**: Dokumenty wskazują na budżet ok. 100 000 PLN, obejmujący koszty programistyczne, testowanie, wdrożenie oraz zarządzanie projektem.
- **Infrastruktura sprzętowa**: Możliwe dodatkowe serwery lub przestrzeń w chmurze.
- **Licencje oprogramowania**: Oprogramowanie do konteneryzacji i testowania może wymagać licencji.

4.4. Organizacja zespołu, odpowiedzialność

- **Product Owner :** Kamil Koniak
- **Kierownik projektu**: Cyprian Czerwiński
- **Członkowie zespołu projektowego**: Aleksander Bastek, Michał Małolepszy, Kamil Koniak, Cyprian Czerwiński
- **Opiekun projektu**: dr hab. inż. Marta Łabuda – odpowiedzialna za odbiór i aprobatę projektu.

Odpowiedzialności:

- **Zespół projektowy**: Implementacja funkcjonalności, rozwój backendu, frontend, integracja systemu, testowanie i wdrożenie.
- **Kierownik projektu**: Koordynacja działań zespołu, komunikacja z opiekunem i klientem, kontrola nad realizacją celów projektowych.

4.5.Infrastruktura techniczna

- **Serwery i środowisko produkcyjne**: System oparty na architekturze klient-serwer z obsługą mikroserwisów integracja z bazami danych.
- **Zabezpieczenia**: Mechanizmy uwierzytelniania i autoryzacji, szyfrowanie danych

4.6. Infrastruktura komunikacyjna

- Komunikacja w zespole projektowym
  Codzienne rozmowy: Microsoft Teams – używane do bieżącej komunikacji i szybkiej wymiany informacji.
  Spotkania zespołowe: Microsoft Teams – narzędzie do wideokonferencji i planowania zespołowego.
- Zarządzanie zadaniami i dokumentacją
  Zarządzanie zadaniami: Asana – do organizacji i śledzenia postępu prac w projekcie.
  Dokumentacja: Confluence – centralne miejsce na specyfikacje, decyzje i szczegóły techniczne.
- Zarządzanie kodem i CI/CD

Zarządzanie kodem: GitHub – platforma do hostowania kodu źródłowego i współpracy.
Powiadomienia CI/CD: GitHub Actions – automatyzacja testów, budowania i wdrażania aplikacji.

- Komunikacja z użytkownikami
  Wsparcie użytkowników: e-mail – kanał do zgłaszania problemów i zapytań.
  Zbieranie opinii: Google Forms – narzędzie do ankiet i zbierania uwag od użytkowników.
- Promocja i komunikacja z klientami
  Promocja: poczta pantoflowa – nieformalny kanał komunikacji dla rozpowszechniania informacji o projekcie.

4.7.Infrastruktura dokumentacyjna

- **Historia wersji dokumentów**: Śledzenie zmian w dokumentach w formacie SWS i DZW .
- **Dokumentacja Elektorniczna:** Confluence
- **System raportowania**: Regularne raporty postępu projektu do opiekuna, prowadzenie szczegółowej dokumentacji wymagań.

{zasady raportowania i dokumentacji}

Ramowy harmonogram

TASK NAME	START DATE	END DATE	TEAM MEMBER
Rozpoznanie			
Stworzenie WPP	4.11.2024	6.11.2024	All
Zapoznanie się z biblioteką Timefold	7.11.2024	14.11.2024	Kamil, Aleksander
Zapoznanie się z biblioteką Svelte	7.11.2024	14.11.2024	Michał
Zapoznanie się z Github Action	7.11.2024	14.11.2024	Cyprian
Projekt systemu i wdrożenie w technologie			
Stworzenie projektu systemu	17.11.2024	29.11.2024	Kamil
Stworzenie aplikacji prototypowej			
Stworzenie infrastruktury sieciowej i konteneryzacja	13.11.2024	29.11.2024	Cyprian
Wdrożenie i pierwsze działania w timefold	23.11.2024	29.11.2024	Aleksander
Integracja Svelte z frontendem timefold	25.11.2024	29.11.2024	Michał
Zintegrowanie wszystkich komponentów aplikacji	2.12.2024	13.12.2024	All
Stworzenie aplikacji MVP			
Stworzenie infrastruktury bazodanowej	16.12.2024	22.12.2024	All
Stworzenie ról i systemu logowania	16.12.2024	22.12.2024	Olek, Kamil
Stworzenie widoku logowania	16.12.2024	22.12.2024	All

Stworzenie widoku planu lekcji dla nauczyciela i administratora	28.12.2024	4.01.2025	Olek, Kamil, Michał

4.8.Ograniczenia i uwarunkowania

4.9.Rejestr produktu

{Product backlog – np. tabelaryczne przedstawienie wymagania/user stories wraz z priorytetyzacją}

4.10. Rejestr sprintu

4.10.1. Rejestr sprintu 1

{Sprint backlog – tabelaryczne przedstawienie zadań realizowanych w ramach sprintu wraz z przypisaniem osoby realizującej, nakładu godzinowego, itp.
Podpunkt powtórzony tyle razy ile jest planowanych sprintów. }

5 Załączniki

{ }