Dokument wymagań projektowych

Nazwa projektu: DevMatch

Autorzy: Czaplicka Anna, Nowak Anna, Wilczyńska Magdalena, Winiarski Bartłomiej

Data: 24.10.2020

0. Wersje dokumentu

W przypadku zmian w dokumencie, należy określić datę i zakres wprowadzonych zmian.

- wersja 1.0.0 - 24.10.2020 - Pierwsza wersja dokumentu
- wersja 2.0.0 - 15.11.2020 - Druga wersja dokumentu
- wersja 2.0.1 - 19.01.2021 - Trzecia wersja dokumentu (korekta numeracji listy w punkcie 3)

1. Elementy składowe projektu (produkty projektu)

1. **I faza, I semestr (10.2019 - 07.2020)**
    a. elementy programistyczne:
        i. aplikacja webowa
        ii. aplikacja serwerowa - REST API
        iii. Azure Function służąca do pobierania informacji o repozytoriach z serwisu GitHub
        iv. Azure Key Vault służący do przechowywania kluczy
        v. Relacyjna baza danych
    b. elementy nieprogramistyczne:
        i. Prototyp aplikacji
        ii. Domena aplikacji
        iii. Interaktywna dokumentacja Swagger
        iv. Tablica projektowa - Jira
        v. Analiza API serwisu GitHub
        vi. Dokument wizji projektu
        vii. Dokument zakresu projektu
2. **II faza, II semestr (10.2020 - 02.2021)**
    a. elementy programistyczne:
        i. kolejne wydania elementów programistycznych z semestru I

b. elementy nieprogramistyczne:
    i. Raport zawierający wyniki testów skuteczności i badań porównawczych funkcji klasyfikującej
    ii. Dokument wymagań projektowych
    iii. Instrukcja obsługi użytkownika

2. Granice projektu

1. Funkcjonalności zawarte w projekcie: zgodnie z punktem 7.
2. Funkcjonalności nie zawarte w projekcie
    a. **Drzewiasta struktura grup**
    Implementacja drzewiastej struktury tworzy problemy organizacyjne oraz implementacyjne, dotyczące limitu zagnieżdżenia, nadawania uprawnień, wyszukiwania projektów oraz związane z dopasowywaniem użytkowników.
    b. **Automatyczne sugerowanie grup użytkownikowi**
    Jedną z głównych potrzeb realizowaną przez aplikację jest dopasowanie do projektów, a sugestię grup z punktu widzenia użytkownika mogą wydawać być zbyt ofensywne. Istnieje większe ryzyko niż w przypadku projektów, że będą to propozycje potencjalnie niepożądane przez użytkownika.
    c. **Analiza kodu z dowolnych serwisów internetowych bądź nadesłanych plików w celu klasyfikacji zdolności technicznych użytkownika**
    Każda nowa platforma wymaga wdrożenia innego rozwiązania i integracji z nowym API, które musi spełniać pewne warunki, aby móc przetwarzać dane w sposób pozwalający na zbudowanie profilu kompetencji. Serwis GitHub należy do jednego z najpopularniejszych narzędzi informatycznych, który pozwala na uzyskanie takiego profilu, ponieważ dostarcza darmowe i rozbudowane API. Obok takich serwisów jak GitLab, Bitbucket, GitHub jest najpopularniejszy w środowisku deweloperów.
    d. **Lista rezerwowa osób proponowanych do projektu**
    Implementacja takiego mechanizmu pociąga za sobą zaprojektowanie logiki biznesowej, która nie jest niezbędna do realizacji głównego celu systemu oraz wiąże się z koniecznością przeznaczenia roboczogodzin przewyższającą liczbę przeznaczoną na realizację projektu inżynierskiego.
    e. **Tworzenie konwersacji wieloosobowych, czatów wewnątrz grup oraz czatów wewnątrz projektów**
    Funkcja wymieniania prywatnych wiadomości między użytkownikami jest funkcją dodatkową, wspierającą działanie systemu. Aplikacja nie jest przewidziana jako komunikator.
    f. **Eksportowanie rozmowy do plików**
    Jest to funkcja, z której potencjalnie skorzysta niewielu użytkowników, a wymaga dużego nakładu pracy po stronie serwerowej i może generować pliki o nieprzewidywalnych rozmiarach.
    g. **Automatyczne uzupełnianie brakujących w projekcie roli**
    Nie ma możliwości jednoznacznego określenia jakie role są brakujące w projekcie ze względu na indywidualne i niekiedy bardzo specyficzne potrzeby każdego Project Ownera.

h. **System oceniania użytkowników**
   Nie ma możliwości zapewnienia sprawiedliwej oceny użytkownika, która zapewniłaby mu stałe uczestnictwo w projektach. Na ocenę użytkownika wpływ mogłyby mieć takie czynniki, jak zawyżone oczekiwania wobec kompetencji użytkownika lub zbiorowe fałszowanie oceny konkretnej osobie
i. **Aplikacja mobilna**
   Głównym celem projektu jest stworzenie platformy webowej, która w późniejszych etapach może zostać przystosowana do urządzeń mobilnych dzięki technologii PWA, co eliminuje potrzebę tworzenia osobnej aplikacji mobilnej.

3. Lista wymagań funkcjonalnych

Poniżej sformułowano listę wymagań funkcjonalnych projektu w formie user stories, w której zdefiniowano aktorów w następujący sposób:

- **programista** - student kierunków informatycznych, osoba poszukująca pracy w branży informatycznej, osoba zainteresowana informatyką
- **osoba kompletująca zespół** - osoba poszukująca innych członków zespołu informatycznego, pracodawca
- **organizator wydarzeń** - organizator wydarzeń o charakterze informatycznym (np. hackathon)
- **użytkownik aplikacji** - osoba należąca do jakiejkolwiek zdefiniowanej powyżej z grup

Wymagania funkcjonalne:

1. Jako ***programista*** chcę mieć automatycznie tworzony profil kompetencji zawodowych, aby nie musieć wprowadzać wszystkich danych ręcznie.
2. Jako ***programista*** chcę mieć możliwość zarządzania i aktualizowania informacji o wykorzystywanych przez mnie technologiach i posiadanych umiejętnościach.
3. Jako ***programista*** chcę mieć możliwość ustawienia preferencji w zakresie ról, lokalizacji, godzin pracy, branży i formy pracy, by móc określić rodzaj projektów, które mnie interesują i które będą dla mnie wyszukiwane.
4. Jako ***programista*** chcę móc określić do jakiej grupy należę (student, freelancer, praca), aby móc przeglądać projekty z interesujących mnie środowisk.
5. Jako ***programista*** chcę mieć możliwość znalezienia innych programistów o określonych kompetencjach, aby móc realizować określone projekty.
6. Jako ***programista*** chcę mieć możliwość dołączenia do projektu, który mnie interesuje i służy realizacji moich celów.
7. Jako ***programista*** chcę otrzymywać propozycje projektów, aby móc dołączyć do dopasowanych do moich preferencji zespołów.
8. Jako ***osoba kompletująca zespół*** lub ***organizator wydarzeń*** chcę mieć możliwość stworzenia grupy i określenia jej przeznaczenia, by móc gromadzić osoby z tego samego środowiska.
9. Jako ***osoba kompletująca zespół*** chcę mieć możliwość stworzenia projektu i zarządzania nim, aby móc określać rodzaj poszukiwanych osób do projektu, liczbę członków zespołu, wymagane technologie i wynagrodzenie.
10. Jako ***osoba kompletująca zespół*** chcę mieć możliwość automatycznego dopasowania i sugerowania użytkowników o odpowiednich kompetencjach do zdefiniowanych przeze mnie ról, tak aby nie musieć przeszukiwać ich samodzielnie.

11. Jako ***osoba kompletująca zespół*** chcę mieć możliwość dodawania użytkowników do swojego projektu, aby móc nawiązać współpracę z wybranymi osobami.
12. Jako ***osoba kompletująca zespół,*** chcę mieć możliwość wysłania do użytkownika zaproszenia do projektu, aby dodawać do projektu wybrane przeze mnie osoby.
13. Jako ***osoba kompletująca zespół*** chciałbym móc stworzyć projekt widoczny tylko dla zamkniętej grupy docelowej (np. uczelnianej), aby niepowołane osoby nie miały dostępu do chronionych danych.
14. Jako ***organizator wydarzeń*** chciałbym zapraszać uczestników wydarzeń do grupy, aby umożliwić im organizacje w zespoły w ramach wydarzenia.
15. Jako ***użytkownik aplikacji,*** chcę mieć możliwość przeszukiwania użytkowników według ich kompetencji, umiejętności, dostępności, lokalizacji i preferowanych ról, aby móc szybko odnaleźć potrzebnych współpracowników o pożądanych cechach.
16. Jako ***użytkownik aplikacji,*** chcę mieć możliwość przeszukiwania projektów według technologii, ról, dostępności, branży, wynagrodzenia i możliwości pracy zdalnej, aby móc szybko odnaleźć interesujące mnie projekty o pożądanych cechach.
17. Jako ***użytkownik aplikacji,*** chcę mieć możliwość przeszukiwania grup według technologii oraz tego, czy grupa jest predefiniowana, otwarta lub zamknięta, aby móc szybko odnaleźć interesujące mnie grupy o pożądanych cechach.
18. Jako ***użytkownik aplikacji*** chciałbym móc wysyłać prośby o dołączenie do grupy prywatnej, aby mieć dostęp do projektów w obrębie danego środowiska.
19. Jako ***użytkownik aplikacji*** chcę mieć możliwość przeglądania otrzymanych zaproszeń do projektów, by móc wybrać projekty, które mnie interesują.
20. Jako ***użytkownik aplikacji*** chciałbym móc skontaktować się z innymi użytkownikami poprzez wiadomość tekstową w celu przedyskutowania szczegółów projektu oraz warunków współpracy.
21. Jako ***użytkownik aplikacji*** chcę móc dodać swoje zdjęcie profilowe, aby być rozpoznawalnym wśród pozostałych użytkowników.

4. Lista wymagań niefunkcjonalnych

1. **Aplikacja webowa**
    a. Zastosowane technologie:
        i. Vue.js
        ii. LESS
    b. Zintegrowana z REST API, które pozwala na pobieranie i aktualizowanie danych na temat użytkowników, grup i projektów, przeprowadzenie autoryzacji i uwierzytelnienia użytkowników.
    c. Zastosowanie technologii Progressive Web Application
2. **Aplikacja serwerowa - REST API**
    a. Zastosowane technologie:
        i. ASP.NET Core
    b. Udostępnia serwisy REST
3. **Dokumentacja REST API**
    a. Interaktywna dokumentacja Swagger

4. **Baza danych**:
    a. Przechowuje informacje na temat użytkowników, ich preferencji, profilu, kompetencji zawodowych, członkostwa w grupach i zespołach, a także informacje na temat grup i projektów stworzonych w systemie.
    b. Azure Storage - usługa, pozwalająca na magazynowanie plików w chmurze.
    c. Azure Tables storage - Magazyn NoSQL przechowujący dane.
    d. Wykorzystanie Azure Key Vault jako narzędzia do bezpiecznego przechowywania kluczy i haseł środowiskowych.
    e. SQL
5. **Prototyp aplikacji:**
    a. Niefunkcjonalny prototyp aplikacji obejmujący całość systemu, pozwalający na prezentację wizji aplikacji oraz dyskusję nad funkcjonalnościami podczas konsultacji i negocjacji z klientami.
    b. Zastosowane narzędzia:
        i. AdobeXD
6. **Serwisy**
    a. Azure App Service - platforma oparta na chmurze pozwalająca na hostowanie witryn internetowych
    b. Azure Function - usługa obliczeniowa służąca do predykcji umiejętności na podstawie przetworzonego konta użytkownika z zewnętrznych serwisów
7. **Tablica projektowa - Jira**
    a. Zawiera historię zadań funkcjonalnych (user stories)
    b. Zawiera historię poszczególnych zadań zidentyfikowanych w obrębie poszczególnych zadań funkcjonalnych
    c. Stworzona zgodnie z metodyką Kanban
8. **Repozytoria kodu:**
    a. Repozytorium kodu zawierające kod aplikacji webowej w serwisie GitHub
    b. Repozytorium kodu zawierające kod aplikacji serwerowej w serwisie GitHub
    c. Zastosowanie systemu kontroli wersji Git
9. **Raport zawierający wyniki testów skuteczności** i badań porównawczych funkcji klasyfikującej.
    a. Dokument zawierający opis metodologii przeprowadzenia testów skuteczności oraz wydajności funkcji klasyfikującej oraz zestawienia wskaźników, świadczących o jej trafności w kolejnych jej wersjach

5. Mierzalne wskaźniki wdrożeniowe

1. I Semestr
    b. Na koniec semestru klienci otrzymują do dyspozycji system w wersji alfa
    c. System jest udostępniony w publicznej domenie internetowej
    d. Istnieje funkcjonująca baza danych w serwisie Azure
    e. System pozostaje stabilny w momencie przetwarzania 10 użytkowników naraz przez Azure Function odpowiadającą za automatyczne profilowanie kompetencji użytkowników

2. II Semestr
    a. Na koniec semestru klienci otrzymują do dyspozycji system w wersji beta
    b. Aplikacja posiada co najmniej 20 zarejestrowanych użytkowników
    c. System zostanie poddany testom obciążeniowym gwarantującym, że w przypadku, gdy z systemu będzie korzystać nie więcej niż 20 użytkowników jednocześnie, to maksymalny czas odpowiedzi serwera nie przekroczy 2 sekund
    d. System zostanie sprawdzony przez benchmark Lighthouse i otrzyma notę w wysokości minimum 70 punktów

6. Kryteria akceptacji projektu dla I semestru prac

1. **Wymagane**
    e. Akceptacja funkcjonalna
        i. Ukończenie funkcjonalności o numerze (1,4,8) zdefiniowanych w punkcie 3
        ii. Wdrożenie systemu na publiczną stronę WWW
    f. Akceptacja jakościowa
        i. Akceptacja produktu przez klientów zdefiniowanych w projekcie
    g. Inne
        i. Stworzenie dokumentacji produktu
        ii. Pierwsza wersja produktu zostanie opracowana zgodnie z terminem oddania pracy inżynierskiej
2. **Oczekiwane**
    a. Akceptacja funkcjonalna
        i. Dostosowanie funkcjonalności do wymagań klientów
        ii. Ukończenie funkcjonalności o numerach (3, 5) zdefiniowanych w punkcie 3
    b. Akceptacja jakościowa
        i. Akceptacja skuteczności wykrywania technologii w obrębie testowej grupy programistów
    c. Inne
3. **Planowane**
    a. Akceptacja funkcjonalna
        i. Ukończenie funkcjonalności o numerze (6) zdefiniowanych w punkcie 3
    b. Akceptacja jakościowa
        i. Implementacja i pomyślne przejście testów jednostkowych
    c. Inne
        i. Udostępnienie aplikacji klientom

7. Kryteria akceptacji projektu dla II semestru prac

1. **Wymagane**
    a. Akceptacja funkcjonalna
        i. Ukończenie funkcjonalności o numerach (9, 11, 12, 13 , 14, 16, 17) zdefiniowanych w punkcie 3
        ii. Dostosowanie funkcjonalności do wymagań klientów

b. Akceptacja jakościowa
    i. Akceptacja produktu przez klientów zdefiniowanych w projekcie
    ii. Implementacja i pomyślne przejście testów jednostkowych
    iii. Akceptacja skuteczności wykrywania technologii w obrębie testowej grupy programistów
    iv. Wdrożenie systemu zgodnie z wymaganiami określonymi w punkcie 5.
c. Inne
    i. Druga wersja produktu zostanie oddana zgodnie z terminem oddania pracy inżynierskiej

2. Oczekiwane

a. Akceptacja funkcjonalna
    i. Ukończenie funkcjonalności o numerach (2, 7, 10, 14, 15, 18, 19) zdefiniowanych w punkcie 3
    ii. Wdrożenie systemu na publiczną stronę WWW
b. Akceptacja jakościowa
    i. Implementacja i pomyślne przejście testów jednostkowych
c. Inne
    i. Stworzenie dokumentacji produktu

3. Planowane

a. Akceptacja funkcjonalna
    i. Ukończenie funkcjonalności o numerach (20, 21) zdefiniowanych w punkcie 3
    ii. Umożliwienie łatwego korzystania z aplikacji na urządzeniach mobilnych z użyciem technologii PWA
b. Inne
    i. Udostępnienie aplikacji do użytku publicznego

8. Organizacja pracy zespołu

8.1. Role projektowe oraz zakres prac

Zespół składa się z czterech członków. Poniżej w tabeli przedstawiono role oraz zakres prac poszczególnych członków zespołów.

Tabela 1. Członkowie zespołu oraz ich role projektowe i deweloperskie w projekcie DevMatch.

Członek zespołu	Role projektowe i developerskie	Zakres prac
Czaplicka Anna	-Backend Developer	-Tworzenie API dla aplikacji -Opracowanie i implementacja metody klasyfikacji umiejętności technicznych użytkowników -Opracowanie i implementacja algorytmu dopasowywania użytkowników do projektów -Kontakt z klientem -Tworzenie dokumentacji projektowej -Testy automatyczne

Nowak Anna	-Backend Developer -Administrator	-Nadzór i obsługa środowiska na platformie Microsoft Azure -Tworzenie API dla aplikacji -Projekt i wdrożenie architektury systemu -Nadzór architektury po stronie back-end -Kontakt z klientem -Tworzenie dokumentacji projektowej -Testy jednostkowe
Wilczyńska Magdalena	-Frontend Developer -Product Owner -UX Designer	-Stworzenie prototypu aplikacji -Implementacja interfejsu użytkownika -Kontakt z klientem -Tworzenie dokumentacji projektowej
Winiarski Bartłomiej	-Frontend Developer -Software architect -Team leader	-Projekt i wdrożenie architektury systemu -Nadzór architektury po stronie front-end -Implementacja interfejsu użytkownika -Tworzenie dokumentacji projektowej

8.2. Zarządzanie komunikacją z klientami

Zespół pozyskał trzech klientów, którzy reprezentują trzy grupy docelowe. Grupa klientów składa się z: reprezentanta studentów, reprezentanta osób realizujących projekty na zlecenie (*ang*. freelancers) oraz reprezentanta pracowników firm informatycznych. Za zarządzanie komunikacją z klientem odpowiada powołany trzyosobowy podzespół. Każda z wyznaczonych trzech osób odpowiedzialna jest za kontakt z jednym z klientów oraz ustalanie szczegółów spotkań, w tym czasu i miejsca, poszczególnych spotkań w określonym przez podzespół okresie. W fazie przygotowawczej projektu inżynierskiego przedstawiono każdemu z klientów zakres oczekiwanego zaangażowania ze strony klienta, harmonogram projektu inżynierskiego oraz ustalono szacowaną średnią miesięczną liczbę godzin przeznaczoną na kontakt z zespołem, sposób komunikacji z klientem oraz oczekiwaną częstotliwość spotkań. Uzgodniono przeprowadzać spotkania poprzez wideorozmowy z wykorzystaniem jednej z dostępnym platform (takich jak Google Meet, Discord), co najmniej raz w miesiącu. Każda z trzech osób delegowanych do kontaktu z klientem sporządza po spotkaniu sprawozdanie, które zawiera uwagi, opinie oraz oczekiwania wystosowane przez klienta. Po przeprowadzeniu serii spotkań zespół dokonuje analizy i integracji otrzymanych informacji zwrotnych oraz dyskutuje nad sposobem zastosowania się do uwag i wdrożenia zgłaszanych zmian. W przypadku powstałych konfliktów i wymagań niemożliwych do pogodzenia zespół kontaktuje się z klientami.

8.3. Metodyka stosowana w projekcie

Do realizacji projektu inżynierskiego wybrano metodykę Kanban. W przygotowawczej fazie projektu dokonano analizy i zestawienia głównych najważniejszych założeń znanych metodyk zwinnych oraz prześledzono etapy, składające się na cykl życia zadania w projekcie. Metodykę Kanban wybrano, ponieważ pozwala na ograniczanie pracy w toku oraz w przeciwieństwie do metodyki Scrum nie nakazuje wykorzystywania iteracji ograniczonych czasowo. Kanban pozwala także na optymalizację procesu, aby czas dostarczenia nowych funkcjonalności był jak najkrótszy i jak najbardziej przewidywalny, co w kontekście realizacji projektu inżynierskiego ograniczonego czasowo

oraz podlegającego ścisłemu harmonogramowi było istotnym aspektem o dużej wadze. W przeciwieństwie do metodyki Scrum jest także mniej nakazowym sposobem zarządzania projektem i narzuca zespołowi mniejszą liczbę reguł, co w przypadku małego zespołu pozwala unikać nadmiernej formalizacji.

8.4. Narzędzia projektu i zarządzanie kodem źródłowym w projekcie

Zespół w ramach projektu wykorzystuje narzędzie Jira, które służy do zarządzania projektem oraz śledzenia postępów projektu. Do kontroli wersji Git w projekcie wybrano serwis GitHub. Projekt rozwijany jest w ramach dwóch repozytoriów. Jedno repozytorium przeznaczono do prac nad implementacją interfejsu użytkownika aplikacji webowej, natomiast drugie do rozwoju interfejsu web API, obsługi połączenia klient-serwer oraz komunikacji z bazą danych. Dla każdego zadania przeznaczonego do implementacji zgodnie z podejściem *branch per task* zostaje utworzony osobny *branch* z numerem powiązanego ticketu w systemie Jira. W celu zachowania wysokiej jakości kodu zespół stosuje *pull requesty* oraz przeprowadza wzajemne *code review,* które odbywa się pomiędzy członkami pracującymi w obrębie tego samego repozytorium. Każde z repozytoriów posiada główną gałąź *master*, na której umieszczany jest oficjalny i przetestowany kod zgodny z ostatnią, wydaną wersją oprogramowania. Prace nad kolejnym wydaniem aplikacji *mergowane* są do gałęzi *develop*. Wersjonowanie zadań oraz przeprowadzanie i śledzenie kolejnych wydań oprogramowania przeprowadzane jest w systemie Jira. Projekt posiada jedną instancją testową, a kolejne wydania wdrażane są w sposób półautomatyczny. Jako narzędzie CI/CD w projekcie używany jest *Azure App Service Deployment Center*. Publikowanie zmian w interfejsie API dokonywane jest automatycznie w momencie dodania nowych zmian do gałęzi *master*, pozwalając tym samym na ciągłe dostarczanie aktualnego oprogramowania. Dodatkowo publikację zmian można przeprowadzić z lokalnych środowisk programistów za pomocą utworzonych na platformie Azure profili publikacji (*Publish Profiles*). W przypadku odtwarzania infrastruktury projektu zespół posługuje deklaratywnymi szablonami, które pozwalają wdrożyć i odtworzyć usługi, potrzebne do funkcjonowania całego systemu.

8.4. Cykl życia zadań w projekcie

Przepływ pracy w projekcie ilustrowany jest za pomocą wirtualnej tablicy Kanban, utworzonej na platformie Jira i podzielonej na dwa tory - do realizacji produktów i wymagań związanych z częścią front-end oraz back-end. Tablica składa się z pięciu kolumn, które odpowiadają poszczególnym etapom w cyklu życia zadania w projekcie. W pierwszej fazie zadanie, które zostało zidentyfikowane jako konieczne do realizacji funkcjonalnych bądź niefunkcjonalnych wymagań projektu zostaje umieszczone w kolumnie „Backlog". Zadania o najwyższym priorytecie oraz te przeznaczone do realizacji danej funkcjonalności zostają umieszczone w kolumnie „Selected for Development". Kiedy zespół jest gotowy do pracy, każdy z członków wybiera zadanie do realizacji i przeciąg je do kolumny „In progress", która posiada ograniczenie do 8 zadań jednocześnie znajdujących się w fazie developmentu. Po ukończeniu pracy nad zadaniem, zostaje ono umieszczone w kolumnie „In review", podczas którego wyznaczony członek zespołu dokonuje *review* zadania. W przypadku uwag lub wykrycia przez recenzenta konieczności dokonania zmian, zadanie umieszczane jest z powrotem w kolumnie „In progress", skąd, po przeprowadzonych zmianach, ponownie może trafić do *review*. Po zatwierdzeniu zadania przez dokonującego *review*, zadanie zostaje umieszczone w kolumnie „Done". Zadania w projekcie podlegają wersjonowaniu. W drugim semestrze dodano także kolumnę "Suspended", w której znajdują się zadania tymczasowo zawieszone.

9. Ryzyka projektowe

Poniżej zidentyfikowano ryzyka, związane z realizacją projektu oraz sposób postępowania, w wyniku spełnienia, któregoś z nich:

- **Ryzyka rozpoznane ze względu na zasoby:**
  - Niska użyteczność funkcji przewidującej kompetencje (klasyfikującej technologie) ze względu na jej niską trafność. Zwiększenie trafności funkcji ograniczone jest stopniem zaawansowania algorytmu klasyfikującego. W celu minimalizowania tego ryzyka przeprowadzone zostaną testy skuteczności funkcji z potencjalnymi użytkownikami. Jeśli skuteczność funkcji wykazana testem spadnie poniżej 50%, użytkownik będzie zobligowany do manualnego zweryfikowania i zatwierdzenia kompetencji oraz zostanie poinformowany poprzez komunikat w aplikacji o stopniu skuteczności tej funkcji.
  - Niska miarodajność testów skuteczności funkcji ze względu na zebranie niewystarczających danych lub zbyt małej liczby danych do zbiorów testowych. Uwarunkowane jest to stopniem zaangażowania osób trzecich, posiadających konta w serwisie GitHub. Jeśli wielkość zbioru testowego nie przekroczy lub nie wyniesie 25 osób, użytkownik będzie zobligowany do manualnego zweryfikowania i zatwierdzenia kompetencji oraz zostanie poinformowany poprzez komunikat w aplikacji o niskiej miarodajności funkcji.
  - Zmiana polityki serwisu GitHub w zakresie korzystania z dostarczanego API, np. wprowadzenie opłat lub zmniejszenie liczby możliwych do wykonania zapytań. W przypadku wprowadzenia opłat za korzystanie z API, zostanie przeprowadzona integracja z innym systemem kontroli wersji Git, np. Bitbucket. Integracja z innym serwisem będzie wymagała wdrożenia sposobu komunikacji z nowym API oraz restrukturyzacji funkcji do predykcji kompetencji, co wiązałoby się z wysokim nakładem roboczogodzin. Przewiduje się niskie prawdopodobieństwo wystąpienia tego ryzyka. W przypadku ograniczenia limitu requestów do poziomu poniżej 300 zapytań na użytkownika, użytkownik będzie zobligowany do manualnego zweryfikowania i zatwierdzenia kompetencji oraz zostanie poinformowany poprzez komunikat w aplikacji o stopniu skuteczności funkcji.
  - Brak możliwości przeanalizowania wszystkich repozytoriów użytkownika ze względu na niewystarczającą liczbę requestów dozwolonych do wykonanie przez GitHub. W przypadku osiągnięciu limitu requestów użytkownik otrzyma informację zwrotną z przedziałem czasowym, jakiego dotyczą predykcje umiejętności. Funkcja przewidująca kompetencje analizować będzie repozytoria od najnowszych do najstarszych, aby predykcje kompetencji dotyczyły technologii najbardziej aktualnych.
  - Brak zaangażowania ze strony któregokolwiek członka zespołu lub działanie na szkodę innych członków zespołu. W takiej sytuacji należy zaplanować z co najmniej 72 godzinnym wyprzedzeniem spotkanie, poinformować wszystkich członków zespołu o jego czasie, miejscu oraz celu, jakim jest wyjaśnienie zaistniałej sytuacji i sformułowanie planu mającego dokonać realnej zmiany w kierunku stawianym

zarzutom. W przypadku braku możliwości wypracowaniu kompromisu (absencja na spotkaniu, brak porozumienia w obrębie sformułowanych punktów mających rozwiązać problem), za zgodą wszystkich pozostałych członków zespół zgłasza do opiekuna projektu inżynierskiego wniosek o wykluczenie członka z zespołu.

- Zmniejszenie liczebności zespołu projektowego, w wyniku odejścia któregoś z jego członków, może przełożyć się na trudności w ukończeniu poszczególnych funkcjonalności i utrzymywaniu infrastruktury za którą odpowiedzialnym był dany członek. W takiej sytuacji konieczne będzie podzielenie między pozostałych członków zespołu niewypracowanych roboczogodzin członka opuszczającego zespół (maksymalnie do 80 roboczogodzin) oraz re-negocjacje z klientem odnośnie wymagań funkcjonalnych, jeśli istnieje ryzyko ich niezrealizowania.
- Zablokowanie dostępu do platformy Microsoft Azure, uniemożliwiające wdrożenie aplikacji oraz korzystanie z API, a w konsekwencji także zintegrowanie backendowej i frontendowej części projektu. W przypadku wystąpienia takiej sytuacji, konieczne będzie podjęcie wznowienie subskrypcji i ponowne uruchomienie serwisów lub wdrożenie systemu na innej dostępnej platformie, co wiązać się będzie z dodatkowym nakładem roboczogodzin.

- **Ryzyka związane z klientami:**
  - Ryzyko niespełnienia wymagań klienta, wynikające ze sprzecznych, niemożliwych do pogodzenia w implementacji wymagań wobec projektu wynikające z odmiennych oczekiwań trzech różnych grup klientów. W przypadku konfliktu między wymaganiami klientów zostaną przeprowadzone negocjacje pozwalające na wypracowanie kompromisu. W przypadku braku kompromisu, pierwszeństwo realizacji wymagań określa się w kolejności: reprezentant grupy studentów, reprezentant pracowników firm, reprezentant freelancerów.
  - Ryzyko braku implementacji lub wdrożenia zmian / funkcjonalności, zgłoszonych przez klienta w okresie uniemożliwiającym ich dokonanie przed ukończeniem projektu. Niejasne, niespójne i zmienne oczekiwania klientów względem projektu. W przypadku takiej sytuacji należy przeprowadzić z klientem spotkanie mające na celu wyjaśnienie ograniczeń realizacji projektu inżynierskiego, jego czasu trwania, harmonogramu. Realizacja i wdrożenie funkcjonalności odbywać się będzie zgodnie z wcześniejszymi ustaleniami i harmonogramem.
  - Utrudniony kontakt z klientami wynikający z wprowadzanych obostrzeń i ograniczeń ze względu na pandemie COVID-19. W przypadku utrudnionego kontaktu z klientem, rozumianego jako: trudność z ustalaniem terminu spotkania, brak aktywności ze strony klientów, należy ponownie opracować i uzgodnić sposób komunikacji z klientem, ustalić oczekiwaną częstotliwość spotkań, zakres oczekiwanego zaangażowania ze strony klienta, szacowaną średnią miesięczną liczbę godzin przeznaczoną na kontakt z zespołem, przedstawić klientom harmonogramu prac nad projektem.

10. Kamienie milowe

3. **I faza, I semestr (10.2019 - 07.2020)**
    a. Sporządzenie wizji projektu, zdefiniowanie głównych funkcjonalności i stworzenie fundamentalnych user stories
    b. Pozyskanie klientów zainteresowanych aplikacją
    c. Przygotowanie prototypu aplikacji
    d. Rozpoczęcie iteracyjnego wytwarzania oprogramowania.
    e. Wdrożenie aplikacji
4. **II faza, II semestr (10.2020 - 02.2021)**
    a. Kontynuowanie prac nad iteracyjnym wytwarzaniem oprogramowania
    b. Ukończenie prac nad aplikacją
    c. Wdrożenie aplikacji u klientów
    d. Opracowanie końcowej dokumentacji aplikacji
    e. Przekazanie produktu końcowego klientom