**Plan testów aplikacji Project Master**

**Spis treści:**

1. Wstęp
   1. Cel i zakres
   2. Terminologia
   3. Odwołania
2. Opis projektu

2.1 Opis systemu/ testowanego produktu

2.2 Cele testowania

2.3 Zakres testów

2.4 Założenia i ograniczenia

3. Planowanie testów

3.1 Harmonogram testów

3.2 Zasoby ludzkie i materiałowe

4. Specyfikacja przypadków testowych

4.1 Identyfikacja przypadków testowych

4.2 Opis przypadków testowych

5. Wykonie testów

5.1 Przygotowanie środowiska testowego

5.2 Dokumentacja wyników

6. Zarządzanie incydentami

6.1 Raportowanie incydentów

7. Zakończenie

7.1 Podsumowanie wyników testów i raporty z testów

7.2 Rekomendacja i wnioski

7.3 Uwagi końcowe

1. **Wstęp**
   1. **Cel i zakres**

Celem tworzonego planu testów jest szczegółowa weryfikacja czy aplikacja ProjectMaster spełnia zarówno wymagania funkcjonalne, jak i niefunkcjonalne.

Testy mają na celu zidentyfikowanie jak największej liczby błędów, defektów, które w konsekwencji mogą prowadzić do awarii aplikacji. Wszystkie napotkane defekty zostaną poprawione i przekazane testerom do testów potwierdzających. Poziomy testów objęte w planie testów to:

* Testowanie modułowe
* Testowanie systemowe
* Testowanie integracyjne
* Testowanie akceptacyjne

**1.2 Terminologia**

* UI (User Interface) - interfejs użytkownika
* API (Application Programming Interface) - zestaw reguł i protokołów określających, jak oprogramowania lub ich komponenty powinny się komunikować ze sobą.
* UAT (User Acceptance Testing) - testy akceptacji użytkownika pozwalają na sprawdzenie, czy rozwijane rozwiązanie spełni oczekiwania końcowych użytkowników.
* Użytkownik - osoba korzystająca z aplikacji ProjectMaster, może pełnić różne role: administratora, menadżera projektu, członka zespołu.
* Administrator - użytkownik z najwyższymi uprawnieniami. Jest odpowiedzialny za zarządzanie użytkownikami, dostępami, konfigurację aplikacji.
* Menadżer projektu - użytkownik odpowiedzialny za projekt. Ma dostęp do funkcji tworzenia i śledzenia projektów, zarządzania zadaniami, przypisywaniem ich do konkretnych użytkowników oraz generowania raportów.
* Członek zespołu - użytkownik przypisany do konkretnego projektu/ projektów. Ma dostęp do zadań w ramach projektu, do którego jest przypisany, realizuje zadania, aktualizuje statusy, ma dostęp do komunikatora członków zespołu.
* Projekt - to przedsięwzięcie, które ma na celu stworzenie: usługi, produktu lub określonego celu biznesowego. Projekt obejmuje zbiór zadań koniecznych do realizacji. Może zawierać wiele etapów.
* Zadanie - część pracy do wykonania, zawierająca opis, status, priorytet, termin wykonania oraz użytkownika, który ma to zadanie wykonać.
* Status zadania - aktualny status zadania: „Do zrobienia”, „W toku”, „Zakończony”, „Zablokowany”.
* Priorytet - określa jak dane zadanie jest ważne/ istotne oraz pilność jego wykonania. Określany jako: „Wysoki”, „Średni”, „Niski”.
  1. **Odwołania**

Zgodnie z dokumentem „Test Policy”

1. **Opis projektu**
   1. **Opis systemu/testowanego produktu**

Project Master to kompleksowe rozwiązanie do zarządzania projektami dla małych i średnich firm. Aplikacja ma na celu usprawnić procesy projektowe, zwiększyć efektywność, ułatwić pracę i komunikację zespołów. Główne funkcje aplikacji obejmują tworzenie projektów, zarządzanie zadaniami, przypisywanie zadań do użytkowników, śledzenie postępów, komunikację między członkami zespołu oraz generowanie raportów.

Dokument „Specyfikacja aplikacji Project Master” zawiera szczegółowy opis aplikacji.

**2.2 Cele testowania**

Testy mają na celu zidentyfikowanie jak największej liczby błędów i defektów przed wdrożeniem aplikacji do produkcji. Pozwoli to zminimalizować liczbę potencjalnych awarii aplikacji **ProjectMaster**. Głównym celem jest zapewnienie wysokiej jakości produktu oraz maksymalnej satysfakcji użytkowników. Sprawdzenie czy aplikacja jest stabilna, bezpieczna oraz wydajna.

**2.3 Zakres testów**

Zakres testów obejmuje:

* Testy funkcjonalne - sprawdzenie poprawności działania poszczególnych funkcji aplikacji (tworzenie projektów, przypisywanie zadań, użytkowników, generowanie raportów)
* Testy niefunkcjonalne - obejmujące testy bezpieczeństwa, użyteczności czy wydajności
* Testy regresyjne - przeprowadzane po każdej aktualizacji, dodaniu nowej funkcji czy naprawie błędu
* Testy akceptacyjne - testy przeprowadzane w celu sprawdzenia czy aplikacja spełnia wymagania i jest gotowa na wdrożenie

Testy jakie nie zostaną przez nas wykonane to:

* Testy wydajnościowe - na urządzeniach mobilnych nie będą wykonywane na tym etapie projektu, ze względu na ograniczone zasoby i brak stabilnej wersji aplikacji mobilnej
* Testy automatyczne dotyczące UI zostaną pominięte na etapie początkowym projektu, ponieważ interfejs użytkownika jest w trakcie dynamicznych zmian

**2.4 Założenia i ograniczenia**

Założenia:

* stabilność środowiska testowego (zakłada się, że środowisko testowe będzie stabilne i w pełni skonfigurowane przed rozpoczęciem testów)
* kompletność dokumentacji wymagań (dokumentacja wymagań funkcjonalnych i technicznych jest aktualna i kompletna)
* współpraca zespołów (zespół developerów będzie dostępny do wsparcia i szybkiego naprawienia błędów zgłaszanych podczas testowania)

Ograniczenia:

* ograniczenia budżetowe (ograniczony budżet na testy uniemożliwia zatrudnienie dodatkowych testerów)

1. **Planowanie testów**
   1. **Harmonogram testów**

|  |  |  |  |
| --- | --- | --- | --- |
| Faza testów | Data rozpoczęcia | Data zakończenia | Odpowiedzialność |
| Przygotowanie środowiska | 12.11.2024 | 19.11.2024 | Zespół QA |
| Testy funkcjonalne | 20.11.2024 | 25.11.2024 | Zespół Testerów Manualnych |
| Testy niefunkcjonalne | 26.11.2024 | 02.12.2024 | Zespół Testerów Manualnych/ Automatycznych |
| Testy regresyjne | 03.12.2024 | 06.12.2024 | Zespół Testerów Automatycznych |
| Testy akceptacyjne (UAT) | 09.12.2024 | 20.12.2024 | Interesariusze |

* 1. **Zasoby ludzkie i materiałowe**
* Testerzy manualni – 8 osób
* System operacyjny - Windows, macOS, Linux
* Przeglądarki – Safari, Chrom, Firefox
* Narzędzia testowe – Cypress, Selenium

1. **Specyfikacja przypadków testowych**
   1. **Identyfikacja przypadków testowych**

Przypadki testowe dla aplikacji ProjectMaster będą tworzone na podstawie następujących źródeł:

* Dokumentacji wymagań funkcjonalnych
* Scenariuszu użytkownika (User Stories)
* Specyfikacji technicznej
* Analizy ryzyka

Przypadki testowe będą tworzone i wybierane na postawie następujących kryteriów:

* Pokrycie funkcjonalne (każdy przypadek testowy musi się odnosić do konkretnego wymagania funkcjonalnego)
* Priorytet biznesowy (przypadki testowe będą priorytetyzowane, od najistotniejszych z punktu widzenia procesu biznesowego)
* Ryzyko (większy nacisk będzie kładziony na obszary, które są bardziej narażone na błędy/defekty)
  1. **Opis przypadków testowych**

Przypadki testowe zawierają kluczowe elementy, które umożliwiają zrozumienie co będzie testowane, w jaki sposób oraz jaki jest oczekiwany rezultat.

Elementy opisu przypadków testowych:

* ID przypadku testowego – unikalny identyfikator
* Tytuł przypadku testowego – zrozumiały, krótki
* Warunki wstępne – lista wymagań, które muszą być spełnione w danym przypadku testowym,
* Kroki do reprodukcji – szczegółowy opis kroków, który należy wykonać
* Oczekiwany rezultat- rezultat jaki powinien być osiągnięty po wykonaniu kroków testowych
* Warunki końcowe – stan aplikacji jaki powinien być osiągnięty po wykonaniu przypadku testowego
* Priorytet – ważność przypadku testowego

1. **Wykonanie testów**
   1. **Przygotowanie środowiska testowego**

Przed przystąpieniem do testów konieczne jest przygotowanie (stworzenie, skonfigurowanie) środowiska testowego, które będzie jak najbardziej zbliżone do środowiska produkcyjnego.

* Aplikacji ProjectMaster zostanie wdrożona na serwerze staging
* Wersja aplikacji 1.0.0. Alfa
* Środowisko testowe korzysta z baz danych PostgreSQL 14.5, skonfigurowanej zgodnie z wymaganiami aplikacji projectMaster
* Baza danych zostanie zainicjowana kopią danych produkcyjnych (przy zachowaniu zasad RODO)

Narzędzia jakie będą wykorzystywane do testowania:

* Postman- do testów API
* Selenium WebDriver – do testów automatycznych UI

Zestaw danych jaki zostanie przygotowany, będzie zawierał rożne przypadki użytkownika (przykładowe projekty, zadania, użytkowników z różnymi rolami). Dane testowe będą aktualizowane i resetowane po każdej sesji testowej.

* 1. **Dokumentacja wyników**

Opis sposobu raportowania wyników testów:

* Codzienne raporty testowe - listę przetestowanych przypadków testowych z ich statusem, lista defektów
* Tygodniowe raporty- wyniki testów, analiza błędów (dokładna liczba w tym ich klasyfikacja), najczęściej występujące błędy- rekomendacje dla zespołu developerskiego.

1. **Zarzadzanie incydentami**
   1. **Raportowanie incydentów**

Wyniki incydentów jak i samych testów będą monitorowane na bieżąco za pomocą dashboardu w narzędziu JIRA. Menadżer będzie regularnie analizował wyniki i identyfikował obszary, które wymagają dodatkowej uwagi.

W przypadku wykrycia krytycznych błędów, zespół developerski zostanie o tym natychmiast poinformowany w celu ich naprawy.

1. **Zakończenie**

**7.1 Podsumowanie wyników testów i raport z testów**

Raport końcowy zostanie sporządzony po zakończeniu wszystkich testów. Dokument będzie zawierał: podsumowanie wykonanych testów, procentowo określony stopień pokrycia testów, liczba błędów ze statusem naprawy, lista przypadków jakie nie udało się wykonać- z wyjaśnieniem.

**7.2 Rekomendacje i wnioski**

Wnioski dotyczące gotowości aplikacji do wdrożenia na produkcję również będą dostarczone po zakończeniu testów.

**7.3 Uwagi końcowe**

Rekomendacje dotyczące dalszych działań, takich jak dodatkowe testy, poprawki lub optymalizacja po zakończeniu testów. Identyfikacją potencjalnego ryzyka związanego z wdrożeniem aplikacji.