Plan testów aplikacji webowej ProjectMaster

1. Wstęp

1.1 Cel i zakres

Celem niniejszego dokumentu jest weryfikacja funkcjonalności oraz przeprowadzenie testów niefunkcjonalnych aplikacji webowej **ProjectMaster**.

1.2 Terminologia

aplikacja webowa-interaktywna platforma, która pozwala użytkownikom wykonywać określone zadania i funkcje

testy funkcjonalne – testy, które sprawdzają, czy funkcje aplikacji działają zgodnie z wymaganiami

testy niefunkcjonalne – testy sprawdzające aspekty takie jak wydajność, bezpieczeństwo, użyteczność

testy regresyjne – testy sprawdzające, czy nowe zmiany w aplikacji nie wprowadziły nowych błędów w istniejących funkcjonalnościach

testy integracyjne – testy weryfikujące, jak komponenty systemu współpracują ze sobą

testy akceptacyjne – testy mające na celu ocenę, czy aplikacja spełnia oczekiwania końcowego użytkownika

1.3 Odwołania

-specyfikacja wymagań aplikacji **ProjectMaster**
-dokumentacja techniczna aplikacji
-plany i harmonogramy rozwoju projektu

2. Opis projektu

2.1 Opis systemu/testowanego produktu

ProjectMaster to platforma internetowa zaprojektowana z myślą o małych i średnich firmach, która wspiera zarządzanie projektami. Dzięki niej użytkownicy mogą tworzyć projekty, organizować zadania oraz efektywnie nadzorować ich realizację. Aplikacja pozwala na przypisywanie zadań członkom zespołu, monitorowanie postępów, a także ułatwia komunikację wewnętrzną oraz generowanie raportów podsumowujących prace nad projektami.

2.2 Cele testowania

Celem testowania jest:

-zapewnienie, że wszystkie funkcjonalności aplikacji działają zgodnie z wymaganiami

-sprawdzenie wydajności i bezpieczeństwa aplikacji

-ocena użyteczności aplikacji w różnych środowiskach

-wykrycie błędów i nieprawidłowości, które mogą wpłynąć na użytkowanie aplikacji

2.3 Zakres testów

Zakres testów obejmuje:

-testy funkcjonalne: testowanie rejestracji i logowania, zarządzania projektami, zadań i użytkownikami, generowania raportów, powiadomień

-testy niefunkcjonalne: testy wydajnościowe (testy obciążeniowe, czas odpowiedzi), testy bezpieczeństwa (szczególnie przed atakami typu SQL Injection, XSS, CSRF)

-testy użyteczności: testowanie interfejsu, dostępności na różnych urządzeniach i przeglądarkach

-testy integracyjne: testowanie współpracy modułów aplikacji, szczególnie związanych z bazą danych, logowaniem, powiadomieniami

-testy akceptacyjne: testy wykonane przez użytkowników w celu weryfikacji, czy aplikacja spełnia ich oczekiwania

Zakres testów nie obejmuje:

-testów backendu, konfiguracji serwerów czy zarządzania bazą danych (poza testami integracyjnymi)

-testów związanych z wdrożeniem i migracją danych

2.4 Założenia i ograniczenia

-testy będą przeprowadzone w środowisku testowym, które odzwierciedla produkcyjne środowisko aplikacji

-zespół testerów będzie miał dostęp do pełnej dokumentacji aplikacji oraz środowisk testowych

-testy bezpieczeństwa będą realizowane tylko w zakresie aplikacji webowej, nie obejmując konfiguracji serwerów

3. Strategia testowania

3.1 Strategia testów funkcjonalnych

Testy funkcjonalne będą obejmowały:

-testowanie rejestracji, logowania, resetowania haseł

-testowanie tworzenia, edytowania i usuwania projektów i zadań

-testowanie przypisywania zadań, oznaczania ich jako zakończonych

-testowanie generowania raportów

-testowanie komunikacji wewnętrznej (komentarze, wiadomości, grupy dyskusyjne)

-testowanie powiadomień w aplikacji i e-mailowych

3.2 Strategia testów niefunkcjonalnych

Testy niefunkcjonalne obejmą:

-testy wydajnościowe: sprawdzenie czasu odpowiedzi aplikacji przy jednoczesnym logowaniu co najmniej 100 użytkowników, testy obciążeniowe

-testy bezpieczeństwa: testy ochrony danych użytkowników, weryfikacja mechanizmów szyfrowania (bcrypt dla haseł), testy odporności na ataki XSS, SQL Injection, CSRF

-testy użyteczności: sprawdzenie intuicyjności interfejsu, responsywności aplikacji na różnych urządzeniach, ocena dostępności (WCAG)

3.3 Strategia testów regresyjnych

Testy regresyjne będą wykonywane po każdej iteracji, aby upewnić się, że nowe zmiany nie wprowadziły błędów w istniejących funkcjonalnościach.

3.4 Strategia testów integracyjnych

Testy integracyjne będą obejmowały:

-sprawdzenie, czy różne moduły systemu (np. rejestracja użytkowników, zarządzanie projektami, powiadomienia) poprawnie współpracują ze sobą

-testowanie przepływu danych między frontendem a backendem

3.5 Strategia testów wydajnościowych

Testy wydajnościowe obejmą:

-sprawdzanie czasu odpowiedzi aplikacji przy różnych obciążeniach (100, 500, 1000 użytkowników)

-testy obciążeniowe (symulowanie pracy wielu użytkowników w tym samym czasie)

3.6 Strategia testów bezpieczeństwa

Testy bezpieczeństwa obejmą:

-testy odporności aplikacji na ataki XSS, SQL Injection, CSRF

-weryfikacja mechanizmów szyfrowania haseł

-westowanie dostępności aplikacji przez HTTPS z certyfikatem SSL

3.7 Planowanie i harmonogram testów

Harmonogram testów będzie dostosowany do cyklu rozwoju projektu. Testy funkcjonalne będą wykonywane w trakcie każdej iteracji sprintu, natomiast testy niefunkcjonalne (wydajnościowe, bezpieczeństwa) oraz testy akceptacyjne po zakończeniu implementacji głównych funkcji.

3.8 Środowiska testowe i narzędzia

-środowisko testowe: testowe serwery aplikacyjne, bazodanowe

-narzędzia: Selenium (testy UI), JMeter (testy wydajnościowe), OWASP ZAP (testy bezpieczeństwa), Jira (zarządzanie incydentami), Postman (testy API)

4. Planowanie testów

4.1 Harmonogram testów

Testy będą realizowane równolegle z rozwojem projektu, z podziałem na sprinty. Harmonogram testów będzie dostosowany do iteracyjnego procesu deweloperskiego.

4.2 Zasoby ludzkie i materiałowe

-testerzy: Zespół 3 testerów funkcjonalnych, 1 tester wydajnościowy, 1 tester bezpieczeństwa

-środowiska testowe: Aplikacja uruchomiona na serwerach testowych

-urządzenia: Komputery, tablety, smartfony z różnymi systemami operacyjnymi (Windows, macOS, Android, iOS)

4.3 Zarządzanie ryzykiem testowym

Potencjalne ryzyka:

-problemy z dostępnością środowisk testowych

-zmiany w wymaganiach w trakcie testów

-ograniczenia czasowe w związku z iteracyjnym procesem rozwoju

4.4 Plan testów na poziomie modułów i komponentów

Testy na poziomie modułów obejmą:

-rejestrację, logowanie, zarządzanie użytkownikami

-zarządzanie projektami i zadaniami

generowanie raportów, powiadomienia

4.5 Plan testów na poziomie integracji

Testy integracyjne obejmą interakcje pomiędzy modułami: logowanie, przypisanie zadań, generowanie raportów.

4.6 Plan testów na poziomie systemu

Testy systemowe obejmują pełne testy aplikacji w środowisku testowym, w tym testy wydajnościowe, bezpieczeństwa oraz testy interakcji między komponentami.

4.7 Plan testów akceptacyjnych

Testy akceptacyjne będą wykonane na końcu cyklu testowego przez użytkowników końcowych, aby upewnić się, że aplikacja spełnia ich oczekiwania.

5. Specyfikacja przypadków testowych

5.1 Identyfikacja przypadków testowych

Przypadki testowe będą identyfikowane na podstawie wymagań funkcjonalnych i niefunkcjonalnych.

5.2 Opis przypadków testowych

Każdy przypadek testowy będzie zawierał opis działania, warunki początkowe, kroki testowe oraz oczekiwane wyniki.

5.3 Warunki początkowe i kroki testowe

Każdy przypadek testowy będzie miał jasno określone warunki początkowe oraz kroki do wykonania.

5.4 Oczekiwane wyniki

Oczekiwane wyniki będą szczegółowo opisane, uwzględniając poprawność działania funkcji oraz czas odpowiedzi.

6. Wykonanie testów

6.1 Przygotowanie środowiska testowego

Środowisko testowe zostanie skonfigurowane zgodnie z wymaganiami projektu, z dostępem do niezbędnych baz danych, serwerów aplikacyjnych i systemów monitorujących.

6.2 Wykonanie testów funkcjonalnych

Testerzy będą przeprowadzać testy funkcjonalne zgodnie z wcześniej opracowanymi przypadkami testowymi.

6.3 Wykonanie testów niefunkcjonalnych

Testy wydajnościowe i bezpieczeństwa będą realizowane równolegle z testami funkcjonalnymi.

6.4 Dokumentacja wyników testów

Wszystkie wyniki testów będą dokumentowane w systemie zarządzania testami, z odpowiednimi raportami błędów.

7. Zarządzanie incydentami

7.1 Raportowanie incydentów

Wszelkie błędy i problemy będą zgłaszane w Jira, z przypisanymi priorytetami.

7.2 Monitorowanie i kontrola incydentów

Testerzy będą monitorować status incydentów, aby upewnić się, że są one rozwiązywane na bieżąco.

7.3 Zarządzanie priorytetami

Incydenty będą klasyfikowane na podstawie ich wpływu na funkcjonalność aplikacji i czas naprawy.

8. Zakończenie

8.1 Podsumowanie wyników testów

Po zakończeniu testów przygotowany zostanie raport zawierający podsumowanie wyników i szczegóły napotkanych problemów.

8.2 Rekomendacje i wnioski

Na podstawie wyników testów zostaną przygotowane rekomendacje dotyczące dalszego rozwoju aplikacji.

8.3 Uwagi końcowe

Testy będą realizowane zgodnie z harmonogramem, a wszelkie zmiany w planie testów będą na bieżąco konsultowane z zespołem projektowym.