Test plan Maciej Andrzejczak

1. Wprowadzenie

1.1 Cel

Stworzenie procesu testowego dla aplikacji “Zarządzanie koszykiem”.

1.2 Opis projektu

Aplikacja ecommerce umożliwiająca użytkownikowi:

- dodawanie produktów do koszyka,
- edytowanie zawartości koszyka pod względem ilości każdego z produktów,
- usuwanie produktów z koszyka,
- przechodzenie do etapu finalizacji zamówienia.

Aplikacja zostanie dostosowana do urządzeń mobilnych oraz desktopowych w najpopularniejszych rozdzielczościach oraz do najpopularniejszych wersji systemów operacyjnych iOS, Android i Windows.

1.3 Odbiorcy

Product Owner oraz zarząd firmy zlecającej projekt.

2. Strategia testów

2.1 Cel testowania

- testy dynamiczne
- znalezienie oraz naprawa defektów,
- retesty,
- zapewnienie najwyższej jakości aplikacji,
- testy pielęgnacyjne,
- testy funkcjonalne oraz niefunkcjonalne,

2.2 Założenia testów

- dostępne jest środowisko testowe (pre-produkcyjne),
- aplikacja zarządzania koszykiem została ukończona i jest gotowa do testów,
- zespół testerski posiada dokumentację dotyczącą aplikacji oraz wiedzę na temat jej działania,
- zespół QA oraz dev posiada szeroki dostęp do urządzeń testowych oraz licencję Browserstack,

2.3 Poziomy i typy testów

2.3.1 Testy jednostkowe

Cel: przetestowanie wszystkich modułów aplikacji “Zarządzanie koszykiem”. Zapobieganie przedostawania się defektów na wyższy poziom.

Zakres: dodawanie produktów do koszyka, usuwanie ich, zmiana ilości produktów w koszyku, przechodzenie do etapu finalizacji zamówienia.

Testerzy: zespół dev oraz tester automatyzujący.

Metoda: testy automatyczne.

Czas: przez cały czas.

2.3.2 Testy integracyjne i systemowe

Cel: Sprawdzenie zgodności zachowań funkcjonalnych i niefunkcjonalnych oraz kompletności i prawidłowości działania całej aplikacji. Wykrywanie defektów oraz zapobieganie przedostawania się ich na wyższy poziom.

Zakres: interfejs, komunikacja z serwerem (API), wydajność infrastruktury.

Testerzy: zespół QA manual + automation,

Metoda:

- testy manualne oraz automatyczne,
- testy z użyciem narzędzi sprawdzających wydajność i bezpieczeństwo.

Czas: przez cały czas

2.3.3 Testy akceptacyjne

Cel: zgodność aplikacji z założeniami i wymaganiami.

Zakres: cała aplikacja “Zarządzanie koszykiem”.

Testerzy: Product Owner oraz zarząd klienta.

Metoda: testy manualne.

Czas: przed wdrożeniem na środowisko produkcyjne

2.3.4 Testy eksploracyjne

Cel: zapoznanie się z aplikacją przez testera nieznającego produktu oraz dokumentacji.

Zakres: gotowa, w pełni działająca aplikacja.

Testerzy: zespół QA

Metoda: testy manualne.

Czas: gdy aplikacja jest gotowa lub gdy do zespołu dołącza nowa osoba.

2.3.5 Testy niefunkcjonalne

Cel: przetestowanie wydajności, bezpieczeństwa i użyteczności.

Zakres: przetestowanie czasu odpowiedzi serwera na zmiany w koszyku oraz bezpieczeństwa i prywatności użytkownika aplikacji.

Testerzy: zespół QA

Metoda: testy manualne i automatyczne

Czas: gdy aplikacja jest gotowa.

2.3.6 Automatyczne testy regresji

Cel: wsparcie dla testerów manualnych.

Zakres: przyspieszenie testów regresji poprzez ich automatyzację. Testowanie wszystkich funkcji aplikacji.

Testerzy: tester automatyzujący.

Metoda: testy automatyczne poprzez skrypty weryfikujące poprawność działania funkcji aplikacji.

Timing: gdy aplikacja jest gotowa oraz po zmianie w kodzie.

2.4 Dokumenty, które powstaną w procesie testowym

1. przypadki testowe
2. sumaryczny raporty defektów,
3. test plan,
4. skrypty automatyzujące testy,
5. scenariusze testowe,
6. historyjki użytkownika,
7. raport podsumowujący testy.

2.5 Oszacowanie nakładu pracy przy testach

Działania zespołu testerskiego	Potrzebny czas (MD)
Opracowanie test planu	4MD
Analiza wymagań	2MD
Rozmowa z deweloperami i klientem aby dowiedzieć się jak najwięcej o aplikacji	1MD
Projektowanie scenariuszy testowych	3MD
Opracowanie przypadków testowych	5MD
Przeglądanie i modyfikacja istniejących przypadków testowych	2MD
Wykonanie przypadków testowych	4MD
Raportowanie defektów	2MD
Retesty	5MD
Testy regresji	3MD
Napisanie skryptów testów automatyzujących	7MD
Raport podsumowujący	1MD
Analiza raportu podsumowującego oraz wyciąganie wniosków	1MD

3. Strategia realizacji

3.1 Kryteria wejścia i wyjścia

Kryteria wejścia:

- środowisko testowe jest dostępne i aktualne,
- dostęp do środowiska testowego,
- dokumentacja biznesowa jest dostępna i kompletna,
- aplikacja jest gotowa do testów,
- dostępne jest 10 kont testowych na środowisku produkcyjnym oraz testowym,

Kryteria wyjścia:

- wszystkie przypadki testowe zostały wykonane,
- wszystkie defekty zostały naprawione co zostało potwierdzone przez retesty,
- testy regresji zostały zakończone pomyślnie,
- raport podsumowujący został wygenerowany, opisany oraz przekazany do interesariuszy,
- testy akceptacyjne zostały zakończone pomyślnie,

3.2 Walidacja i zarządzanie defektami

- po wykryciu defektu przez dev/QA tworzy on zgłoszenie defektu poprzez Jira,
- dev podejmuje zgłoszenie,
- odtwarzanie defektu - debugowanie,
- naprawa defektu lub prośba doprecyzowania przez osobę zgłaszającą,
- po naprawie przekazanie do retestu,
- retest wykonywany przez testera,
- jeżeli błąd nadal występuje - powrót do debugowania przez developera. Jeżeli błąd nie pojawia się ponownie - zamknięcie zgłoszenia.

3.3 Metryka testów

Metryka	Formuła
Udana ilość testów	$Udana\ ilość\ testów = \left(\frac{Ilość\ poprawnych\ testów}{Liczba\ wszystkich\ wykonanych\ testów}\right) \cdot 100\%$ $\left(\frac{382}{395}\right) \cdot 100\% = 96\%$
Nieudana ilość testów	$Nieudana\ ilość\ testów = \left(\frac{Ilość\ nieudanych\ testów}{Liczba\ wszystkich\ wykonanych\ testów}\right) \cdot 100\%$ $\left(\frac{13}{395}\right) \cdot 100\% = 4\%$
Poprawione defekty	$Poprawione\ defekty = \left(\frac{Ilość\ poprawionych\ defektów}{Całkowita\ liczba\ zgłoszeń\ defektów}\right) \cdot 100\%$ $\left(\frac{551}{801}\right) \cdot 100\% = 68,7\%$
Procent wykonanych przypadków testowych	$Pokrycie\ wykonanych\ testów = \left(\frac{Ilość\ testów\ wykonanych}{Całkowita\ liczba\ testów\ do\ wykonania}\right) \cdot 100\%$ $\left(\frac{423}{451}\right) \cdot 100\% = 93\%$

4. Proces zarządzania testami

4.1 Narzędzia do zarządzania procesem testowym

- Jira - do zgłaszania defektów i zarzadzania nimi.
- Confluence - do tworzenia i weryfikacji dokumentacji.
- Testrail / wtyczka Xray do Jira - do tworzenia przypadków testowych.
- Trello/Evernote - do tworzenia małych notatek dla zespołu.
- Wycinek i szkic (wbudowane w windows) - do dokumentacji defektów.
- Screenpresso - do nagrywania ekranu przy dokumentacji defektów.
- Developer tools (chrome) - używane przy wykonywaniu przypadków testowych oraz do testów wydajnościowych.
- Fake mail oraz generatory losowych danych.
- Slack/Microsoft Teams - komunikacja wewnątrz oraz na zewnątrz zespołu.
- IntelliJ + biblioteki Java (Oracle) - do tworzenia testów automatycznych.
- GitHub - do archiwizacji napisanego kodu testów automatycznych.
- Postman - testowanie API.
- Gatling - testy wydajnościowe.
- Kali Linux - testy bezpieczeństwa.

4.2 Proces tworzenia testów

Tworzeniem przypadków testowych zajmować się będzie zespół testerki i odbywać się będzie ono przy pomocy wtyczki Xray do Jira.

4.3 Proces wykonywania testów

Testy manualne wykonywane będą na bazie stworzonych w Xray przypadków testowych. Defekty zgłaszane będą przy pomocy Jira. Raport z testów zostanie stworzony przy pomocy wtyczki Xray.

4.4 Ryzyka i sposoby ich minimalizacij

Ryzyko	**Prawdopodobieństwo wystąpienia**	**Wpływ**	**Plan złagodzenia skutków**
Zastosowanie innowacyjnej architektury	Wysokie	Wysoki	Dokładne przeanalizowanie dokumentacji dotyczącej nowej architektury.
Brak doświadczenia testerów	Średnie	Wysoki	Dodatkowe szkolenia dla testerów oraz konsultacja z mid/senior testami z innych zespołów.
Braki i błędy w dokumentacji	Średnie	Wysoki	Spotkania z developerami.
Błędy w komunikacji w zespole	Średnie	Średni	Częstsze spotkania zespołowe.
Brak urządzeń do testów	Niskie	Wysoki	Zakup odpowiednich urządzeń do testów.
Brak narzędzi do testów	Niskie	Średni	Weryfikacja i/lub zakup licencji na odpowiednie oprogramowanie.
Braki kadrowe	Niskie	Wysoki	“Wypożyczenie” testerów z innych zespołów.
Bariera językowa	Niskie	Niski	Lekcje językowe dla członków zespołu oraz zapraszanie na spotkania developerów z innych zespołów w roli supportu.
Inna strefa czasowa	Niskie	Niski	Praca w innych godzinach, dostosowanych do innych stref czasowych.

4.5 Odpowiedzialność za testy

4.5.1 Skład zespołu testerskiego

Zespół QA:

- 3 Manual tester - 2 junior i 1 mid,
- 1 Junior Automation Tester
- 1 Test Manager

4.5.2 Development team??

1. Środowisko testowe
    1. środowisko pre-produkcyjne,
    2. Najpopularniejsze urządzenia mobilne oparte na systemach Android od wersji 8 w górę oraz iOS od wersji 10 w górę, w różnych rozdzielczościach.
    3. Urządzenia desktopowe oparte na systemach Windows 7, 8 i 8.1, 10, 11 oraz iOS od 10 górę, do najnowszej wersji.
2. Narzędzia używane w całym procesie testowania

Obszar	Narzędzia
Testowanie API	Postman
Tworzenie przypadków testowych	Wtyczka Xray do Jira / Testrail
Tworzenie dokumentacji, tj: • Test plan, • raporty z testów, • raport defektów	Conflience oraz Jira
Dokumentacja defektów oraz rezultatów testów	• Wycinek i szkic (wbudowane w Windows) • Screenpresso - do nagrywania ekranu.
Wykonywanie testów	• Dev tools-y wbudowane w przeglądarkę • Gatling - testy wydajnościowe • Kali Linux - testy bezpieczeństwa • Fake mail oraz generatory losowych danych
Skrypty do testów automatycznych	• IntelliJ + biblioteki Java (Oracle) • GitHub - do archiwizacji kodu