Kiedy chcesz zbudować oprogramowanie, najczęściej patrzysz na efekt końcowy, czyli działający produkt. Jednak otrzymując zestawienie kosztów, wielu właścicieli firm kwestionuje część rozwoju produktu zwaną testowaniem. Dlaczego? Zazwyczaj ze względu na cenę. Może to być (nie zawsze!) jeden z najdroższych procesów, który wielu chętnie wyłączyłoby z cyklu życia produktu. Jest to jednak absolutnie niezbędna i integralna część. W dzisiejszym artykule dowiesz się wszystkiego o testowaniu i dlaczego nigdy nie radzimy pomijać tej konkretnej części.

>

Dlaczego testujemy oprogramowanie?

Najprostszą odpowiedzią na to pytanie będzie: aby upewnić się, że oprogramowanie działa. Jeśli w kodzie występują jakieś błędy, testowanie pomaga je zidentyfikować przed wydaniem oprogramowania. Powody testowania są jednak niezliczone. Jeśli mielibyśmy podać najważniejsze z nich, zaczęlibyśmy od poniższych.

Efektywność kosztowa

.

Możesz pomyśleć, że to absurdalne stwierdzenie, ponieważ próbujesz uniknąć testowania z tego samego powodu. Niestety, ryzykujesz większe straty finansowe niż oszczędności, które przyniesie brak testów. Po pierwsze, wychwycenie nieprawidłowości na wczesnych etapach projektu sprawia, że są one stosunkowo łatwe i tanie do naprawienia. Jeśli błędy zostaną wyłapane po premierze, wiąże się to z takimi kosztami jak:

- utrata przychodów z powodu niedostępności oprogramowania dla klientów;
- utrata przychodów spowodowana przejściem klientów do konkurencji;
- zwiększone koszty związane z odnalezieniem i naprawą błędu;
- potencjalne reperkusje finansowe związane z postępowaniami prawnymi wynikającymi z awarii systemu.

Zwiększone bezpieczeństwo

Szczególnie w XXI wieku, gdzie cyberprzestępczość jest na tak niezwykle wysokim poziomie, bezpieczeństwo odgrywa jedną z najważniejszych ról w każdym procesie tworzenia oprogramowania. Wielu przedsiębiorców wciąż nie zdaje sobie sprawy, że

wadliwe aplikacje i oprogramowanie z błędami są po prostu zaproszeniem do wycieku informacji i danych użytkownika. Jednak coraz więcej użytkowników zwraca uwagę na niezawodność oprogramowania i wystarczające testy. Bardzo często błędy są głównym powodem, dla którego użytkownicy przestają korzystać z oprogramowania. Co więcej, powoduje to nie tylko niższe dochody, ale także niszczy reputację firmy. Pamiętaj, że niezadowolony użytkownik to bardzo niebezpieczna broń, strzelająca niepochlebnymi opiniami na lewo i prawo. Możesz być pewien, że zagrożenie bezpieczeństwa będzie wystarczającym powodem do złych recenzji.

Wysoka jakość oprogramowania

Oprogramowanie z błędami i bugami jest nie tylko drogie i niebezpieczne. Jest to również po prostu nieprzyjemne doświadczenie. Nawet jeśli błędy zostaną w końcu naprawione, pozostawiają kwaśny posmak w ustach. Nawet jeśli twoje oprogramowanie może być genialne, błędy i usterki znacznie obniżają jego jakość.

>

Większa satysfakcja użytkowników

Jeden mądry człowiek powiedział kiedyś: „Twój produkt jest tak dobry, jak zadowolenie klienta, które przynosi”. Kto chciałby używać oprogramowania z błędami? Powoduje to tylko problemy, frustrację i okropne doświadczenie klienta. Jakość produktu jest głównym czynnikiem wpływającym na zadowolenie klienta. A ponieważ ustaliliśmy już, że błędny produkt jest niskiej jakości, klienci będą niezadowoleni, jeśli otrzymają taki produkt.

>

Jakie ryzyko wiąże się z brakiem testowania oprogramowania?

Czasami korzyści nie są wystarczające, aby podkreślić znaczenie pewnych działań lub zachowań. Są różni ludzie i różne metody dotarcia do nich. Bądźmy bardziej realistyczni i wspomnijmy o kilku rzeczywistych sytuacjach, w których testowanie było ‘zbyt drogie’ pozornie ‘niepotrzebne’ lub po prostu niewystarczające, aby zagwarantować sukces produktu.

W 2016 r. lider branży motoryzacyjnej, Nissan, doznał ogromnej porażki. Firma musiała wycofać ponad 3 miliony samochodów z rynku amerykańskiego z powodu awarii oprogramowania w czujnikach poduszek powietrznych. Jednak było to już trzecie wycofanie z powodu tego samego błędu oprogramowania! Pierwsza z nich (2013) objęła ponad 80 tysięcy samochodów, a druga (2014) prawie milion aut.

>

W 1985 roku w Kanadzie kilku pacjentów przedawkowało promieniowanie w ramach terapii otrzymanej z wadliwie działającego urządzenia Therac-25. Śmiertelne dawki promieniowania zabiły 3 pacjentów i pozostawiły kolejne 3 osoby w stanie krytycznym.

Błędy w oprogramowaniu były również przyczyną wielu innych wypadków, takich jak katastrofa China Airlines Airbus A300 w 1994 roku, w której zginęły 264 osoby, czy nieudane wystrzelenie satelity wojskowego. Ten ostatni został ogłoszony najdroższym wypadkiem w historii (1,2 miliarda dolarów).

Możesz nie myśleć, że oprogramowanie, które chcesz wydać, może prowadzić do tak kosztownych lub poważnych wypadków. Czy jednak którykolwiek z przedsiębiorców odpowiedzialnych za powyższe projekty pomyślał, że takie przypadki mogą się zdarzyć? Założę się, że nie. Dlatego w Sailing Byte cenimy testowanie i nigdy nie pominiemy go, aby zaoszczędzić pieniądze lub czas.

Jak testujemy oprogramowanie?

Jak zapewne możesz sobie wyobrazić, wystarczające testowanie wymaga dokładnego procesu i sekwencji, aby nie pominąć żadnej ważnej części. Przeanalizujmy podstawowe kroki w procesie testowania oprogramowania.

Podstawowe testowanie funkcjonalności

Jak sama nazwa wskazuje, takie testowanie jest odpowiedzialne za sprawdzenie, czy wszystkie funkcje oprogramowania działają dobrze. Obejmuje to klikanie wszystkich przycisków, wprowadzanie tekstu w każdym polu, sprawdzanie podstawowej funkcjonalności API (z funkcjami zaprojektowanymi tak, aby były dostępne za pośrednictwem API) itp.

Przegląd kodu

Dobrą praktyką jest poproszenie o recenzję. Inna para oczu i świeże spojrzenie mogą odkryć wiele przeoczonych kwestii. Błędy funkcjonalności muszą zostać naprawione przed wykonaniem przeglądu kodu.

Statyczna analiza kodu

Narzędzia do statycznej analizy kodu mogą identyfikować słabe punkty, luki w zabezpieczeniach i problemy ze współbieżnością w kodzie źródłowym lub bajtowym bez jego wykonywania. Dzięki tym narzędziom programiści mogą egzekwować standardy kodowania i uruchamiać je automatycznie w ramach kompilacji.

Testowanie jednostkowe

Testy jednostkowe testują prawidłową funkcjonalność jednostki/komponentu/metody/klasy w zakresie prawidłowych i nieprawidłowych danych wejściowych. Jeśli programista pracuje w środowisku ciągłej integracji, takie testy powinny być zaplanowane za każdym razem po wprowadzeniu zmiany w repozytorium kodu źródłowego. Powinny być one również uruchamiane na maszynie deweloperskiej. Programiści używają również obiektów pozorowanych i zwirtualizowanych usług, aby zapewnić niezależne testowanie jednostek.

>

Rodzaje testów, które uruchamiamy

Oprócz procesu testowania, istnieje wiele rodzajów testów, które programiści przeprowadzają, aby upewnić się, że ich oprogramowanie działa zgodnie z przeznaczeniem. Poniżej wymieniamy te, których używamy w naszych codziennych procesach tworzenia oprogramowania:

- Testowanie czarnej skrzynki – testuje funkcjonalność bez wiedzy o jej implementacji i bez widoczności kodu źródłowego. Testerzy wiedzą tylko, co oprogramowanie powinno robić, ale nie jak;

  .

- Testowanie białoskrzynkowe – testuje wewnętrzne struktury programu, a nie

funkcjonalność udostępnianą użytkownikowi końcowemu. Na przykład PHPUnit to framework testowy dla PHP, który koncentruje się na programiście;
- Testowanie UI/UX – testowanie interfejsu użytkownika oprogramowania lub strony internetowej w celu zapewnienia intuicyjnego i łatwego użytkowania; z drugiej strony testowanie UX koncentruje się na tym, jak produkt lub strona internetowa wpływa na całe doświadczenie użytkownika;
- Hallway usability testing- testuje oprogramowanie z pomocą współpracowników i innych osób;
- BrowserStack – narzędzie dające dostęp do platformy w chmurze, która umożliwia programistom testowanie oprogramowania, stron internetowych i aplikacji mobilnych na ponad 3500 rzeczywistych urządzeniach mobilnych i przeglądarkach;
- Testy wydajności – przeprowadzane w celu sprawdzenia szybkości, stabilności i responsywności oprogramowania. W szczególności testujemy obciążenie pamięci, obciążenie procesora i obciążenie danymi;
- Testy penetracyjne – te testy bezpieczeństwa koncentrują się na odkrywaniu i wykorzystywaniu słabości oprogramowania. Wdrażamy te testy przy użyciu narzędzi, takich jak Kali Linux, SQLMap i Metasploit;
- Testowanie A/B – porównuje dwie wersje produktu ze sobą, aby określić, która z nich działa lepiej;
- Google Analytics/Tags – wysyłając dane z danej witryny do powiązanych miejsc docelowych produktów Google, możemy mierzyć jej skuteczność i skuteczność reklam.

Wiele testów można zautomatyzować, co znacznie pomaga w rozwoju oprogramowania poprzez:

- przyspieszenie procesu testowania w porównaniu do testowania ręcznego
- umożliwienie przeprowadzenia większej liczby testów przy mniejszej liczbie analityków
- gwarancja spójności dzięki ponownemu wykorzystaniu tych samych przypadków testowych
- zwiększenie pokrycia testami, ale nie czasu testowania.

Sailing Byte zapewnia najwyższą jakość dzięki

ciągłej walidacji

.

Chcesz poznać prawdę o tworzeniu oprogramowania? Idealne oprogramowanie nie istnieje. To mit, któremu ulega wielu przedsiębiorców. Każdy popełnia błędy. Jesteśmy ludźmi i jest to jedna z naszych głównych cech. Dlatego każdy kawałek oprogramowania wymaga testowania, aby wyłapać jak najwięcej błędów i potencjalnych problemów. Najwyższą jakość, zwłaszcza w tak złożonym i stale rozwijającym się obszarze jak tworzenie oprogramowania, można uzyskać jedynie poprzez ciągłą walidację. Taka ciągła walidacja ocenia, czy rozwiązanie jest użyteczne i służy planowanym potrzebom.

W Sailing Byte testujemy nasze oprogramowanie sami, a także z pomocą oddzielnych testerów (ręcznych lub zautomatyzowanych). Z nami możesz mieć pewność, że Twój produkt będzie tak bezbłędny, jak to tylko możliwe. Zarezerwuj telefon już dziś, aby omówić wszystkie etapy i metody testowania, którym poddamy Twoje oprogramowanie.