Strategia testowa dla portalu: https://autodemo.testoneo.com/pl/

Spis treści

1. Zakres opracowania strategii testowej oraz obszary wykluczone.

Poniższy dokument opisuje strategię testową dla projektu: https://autodemo.testoneo.com/pl/ (lost hat), obejmuje ona zagadnienie stworzenia testów automatycznych wraz z dalszym ich rozwojem. Ze względu na charakterystykę inicjatywy, pominięto kwestie testów manualnych, nie mniej jednak, odnośniki do tego obszaru testów w dokumencie są nieuniknione.

1.1 Aktualna sytuacja.

W projekcie nie istnieją testy automatyczne, wszystkie testy wykonywane są manualnie, co przekłada się na dług czas trwania testów regresyjnych, a co za tym idzie – spowolnionej reakcji na zmiany oraz błędy na środowisku produkcyjnym.

2. Zastosowane narzędzia i wymagane środowiska.

2.1 Zastosowane narzędzia.

Język programowania	Python 3.8 (lub wyższe)
Biblioteki	Behave Selenium
Narzędzie do pracy z kodem	PyCharm (najnowszy lub przynajmniej ostatni wspierany)
Pluginy	Gherkin (ewentualnie inne, wybrane przez zespół testerski)
Integracja kodu	Jenkins (uruchamianie testów), GitHub
Podejście	BDD

2.2 Wymagane środowiska.

Nazwa	**Uwagi**	**Systematyczne uruchamianie testów**
Lokalne	Każdy z członków zespołu powinien mieć możliwość pobrania i wybudowania swojej wersji projektu testów automatycznych, oraz projektu samej aplikacji.	Nie ma takiej potrzeby – tester decyduje kiedy odpala testy.
Rozwojowe (integracyjne)	Z założenia może to być to samo środowisko co testowe (do testów manualnych). Jednak należy pamiętać, że wprowadzi to poważne obniżenie stabilności wykonywanych testów manualnych. Stąd zalecenie by było to osobne środowisko.	Regularne (najlepiej w godzinach nocnych).
Preprodukcyjne	Celem weryfikacji gotowej paczki – tzw. release candidate.	Regularne (najlepiej w godzinach nocnych).

3. Testy automatyczne w procesie wytwarzania oprogramowania.

Testy automatyczne są integralną częścią procesu testowania i zapewnienia jakości oprogramowania, zważywszy na to, sugeruje się przyjęcie następujących zasad:

- Nowa wersja kodu nie może być wydana bez wykonania testów automatycznych, przynajmniej na środowisku preprodukcyjnym.

- Do tworzenia testów automatycznych zostaje powołany osobny zespół, który będzie dbał o rozwój, utrzymanie i pielęgnację testów.

- W przypadku braku osobnego zespołu, testerzy wyznaczeni do pracy z testami automatycznymi będą mieli uwzględnione obowiązki w estymacji czasu pracy.

- Wraz z rozwojem testów, po konsultacji z zespołem programistów zostanie ustalona data, od której nie będzie możliwości tzw. merga kodu bez pozytywnego wyniku testów na środowisku integracyjnym.

4. Ogólny plan wraz z obszarami dalszego rozwoju projektu testów automatycznych.

4.1 Plan

Ramy czasowe mają charakter poglądowy, zostały stworzone przy założeniu, że zespół będzie składał się z 2 testerów automatyzujących oraz lidera, który z założenia poświęci ok. 30% na zadania automatyzacji testów. W planie testów zostaną podane konkretne ramy czasowe, na tym etapie, jest to niemożliwe.

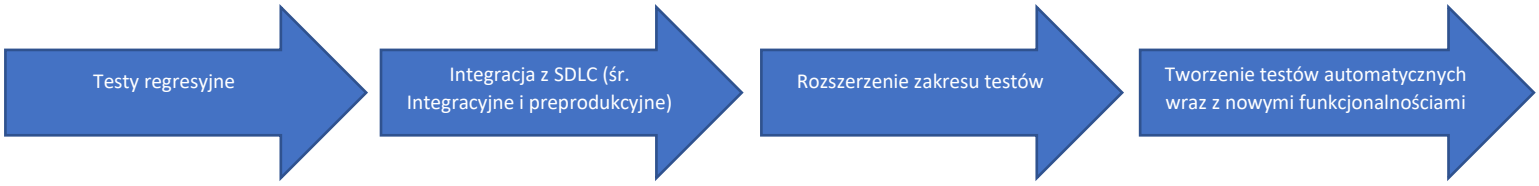
Ze względu na brak testów automatycznych na tym etapie, zaleca się rozpocząć prace od przygotowania testów regresyjnych. Taki zakres prac pozwoli uwidocznić pomyłki powstałe na etapie strategii i planu testów, dzięki czemu istnieje realna szansa stworzenia długoterminowego projektu automatyzacji, który będzie mógł być kontynuowany w przyszłości.

Czynności i zadania	**Czas trwania**	**Uwagi**
Przygotowanie środowisk lokalnych, dobór narzędzi, pluginów, bibliotek	ok. 3 dni	Wraz z rozwojem projektu może się okazać, że do projektu będą dodawane lub odejmowane produkty wybrane na tym etapie.
Przygotowanie scenariuszy BDD dla testów regresyjnych.	5 dni	Przy założeniu, że istnieje w projekcie dokumentacja wymagań. Jeśli nie, koniecznym będzie zaangażowanie zespołu testerów manualnych.
Przygotowanie testów regresyjnych.	25 dni	Przy założeniu, że konfiguracja środowisk integracyjnych zostanie uwzględniona w innej aktywności.
Przygotowanie środowisk integracyjnego i preprodukcyjnego.	5 dni	Przy wsparciu zespołu programistów oraz administratorów

4.2 Dalszy rozwój.

- Integracja testów do zapewnienia jakości w podejściu CI/CD (regularne uruchomienia na środowiskach integracyjnych i preprodukcyjnym).
- Pokrycie testami automatycznych kolejnych obszarów portalu.
- Zintegrowanie tworzenia testów automatycznych wraz z testowaniem manualnym nowych funkcjonalności.
- Wyodrębnienie testów automatycznych całej aplikacji od testów regresji. Przygotowanie podzbiorów testów dla szybkiej kontroli (tylko najważniejsze obszary i tzw. testy dymne) oraz pełnej weryfikacji portalu.

- Powiększenie zespołu testerów automatycznych celem zapewnienia zastępstw w okresie urlopów i świąt .



5. Uzasadnienie powyższej strategii.

W chwili obecnej nie istnieją testy automatyczne dla wskazanego portalu, co jasno wskazuje, że w przeszłości projekt musiał napotkać problemy w tym obszarze. Biorąc to pod uwagę, że nie można rozpocząć procesu rozwoju testów automatycznych na szeroką skalę, zespół testerski, programistów oraz pozostałe osoby w projekcie muszą stopniowo „przyzwyczaić się" do nowej aktywności w zapewnieniu jakości.

Za takim podejściem przemawia również możliwość wysondowania najlepszego rozwiązania dla tworzenia testów. Należy pamiętać, że narzędzia powinny zostać dobrane do potrzeb i umiejętności zespołu, a nie na odwrót.

Wykorzystanie języka Python ułatwi pozyskiwanie w przyszłości nowych członków zespołu, ze względu na popularność języka, oraz fakt, że jest dość prosty w nauce.

Zastosowanie języka Python może przyczynić się do zwiększenia atrakcyjności testów w obszarze rozwoju osobistego członków zespołu. Język ten w przyszłości może być wykorzystywany w obszarze Big Data i sztucznej inteligencji, co za tym idzie, członkowie zespołu mogą postrzegać automatyzację testów jako pierwszy krok w rozwoju w kierunku tych obszarów.