Faza konstrukcji

System TrashOut

Michał Mikołajczyk, Wojciech Adamiec

12 stycznia 2021

1. Testy funkcjonalne

1.1. Administrator dodaje nowego kierowcę/pojemnik/pojazd

Scenariusz:

1. Użytkownik loguje się na konto z uprawnieniami administratora.
2. Użytkownik przechodzi do widoku kierowców/pojemników/pojazdów.
3. Użytkownik klika przycisk dodania nowej pozycji.
4. Na ekranie rozwija się formularz. Użytkownik wypełnia go.
5. Użytkownik naciska na przycisk przesłania formularza.
6. Wyświetlane są informacje zwrotne o ewentualnych błędach lub o sukcesie.
7. Formularz zostaje zamknięty, lista kierowców/pojemników/pojazdów odświeża się.
8. Użytkownik powraca do głównego widoku aplikacji.

Testy:

1. Testy uprawnień użytkownika.
2. Przetestowanie widoków aplikacji pod kątem problemów z wyświetlaniem interfejsu.
3. Sprawdzenie weryfikacji wprowadzanych danych.
4. Przetestowanie czy system wykrywa duplikaty.
5. Sprawdzenie czy listy kierowców/pojemników/pojazdów aktualizują się poprawnie po dodaniu nowych pozycji.

1.2. Kierownik modyfikuje trasę

Scenariusz:

1. Użytkownik przegląda mapę, na której wyświetlone są aktualne trasy, stan pojemników oraz pozycje pojazdów.
2. Użytkownik naciska na listę zaplanowanych tras.
3. Użytkownik przegląda listę tras. Zauważa potrzebę zmiany jednej z nich.
4. Użytkownik naciska na trasę wymagającą korekty.
5. Otwiera się widok trasy z listą pojemników na śmieci oraz przypisaną do niej ekipą śmieciarki.
6. Użytkownik dodaje/usuwa pojemniki na trasie.
7. Użytkownik zmienia przypisaną ekipę do trasy.
8. Użytkownik dodaje komentarz/notatkę do trasy.
9. Użytkownik potwierdza zmiany, otrzymuje informacje zwrotne od systemu.
10. Użytkownik powraca do głównego widoku aplikacji.

Testy:

1. Testy uprawnień użytkownika.
2. Przetestowanie widoków aplikacji pod kątem problemów z wyświetlaniem interfejsu.
3. Sprawdzenie weryfikacji wprowadzanych danych.
4. Testy integracji interfejsu z *Google Maps*.
5. Weryfikacja tras renderowanych na mapie.
6. Weryfikowanie pozycji kierowców/pojemników/pojazdów z tymi wyświetlanymi na mapie.
7. Testy zabezpieczeń przed modyfikacją tej samej pozycji przez dwóch użytkowników w tym samym czasie (blokada).

1.3. Kierowca wyświetla trasę, otwiera nawigację

Scenariusz:

1. Użytkownik naciska na swój profil.
2. Użytkownik naciska na listę aktualnych tras.
3. Użytkownik naciska na wybraną trasę.
4. Wybrana trasa zostaje zaznaczona na mapie. Wyświetlają się jej statystyki, lista pojemników oraz komentarze/notatki.
5. Użytkownik zapoznaje się z trasą.
6. Użytkownik naciska na przycisk, który przeniesie go do nawigacji.
7. Użytkownik zostaje przekierowany do aplikacji/serwisu *Google Maps* gdzie korzysta z nawigacji.
8. Użytkownik wyrusza w drogę. Po zakończonej pracy, powraca do widoku trasy.
9. Użytkownik zatwierdza wykonanie trasy, dodaje ewentualny komentarz/notatkę do pojemników lub do trasy.
10. Użytkownik powraca do listy tras.

Testy:

1. Testy uprawnień użytkownika.
2. Przetestowanie widoków aplikacji pod kątem problemów z wyświetlaniem interfejsu.
3. Sprawdzenie poprawności wyświetlanych statystyk.
4. Przetestowanie wyświetlania odpowiednich komentarzy i notatek.
5. Weryfikacja poprawności nanoszenia trasy na *Google Maps*
6. Przetestowanie odpowiedniego zachowania aplikacji po zakończeniu aktualnej trasy.
7. Sprawdzenie poprawności działania podsystemu zamieszczania komentarza do trasy lub pojemnika.

2. Pomiary spełniania wymagań niefunkcjonalnych

- Czytelny, intuicyjny w użyciu interfejs na każdym poziomie.
  - **Pomiar:** $X = A/T$
  - A - ilość problemów z obsługą interfejsu napotkanych przez użytkownika
  - T - czas użytkowania w minutach
  - **Cel:** $X <= 0.05$
- Aplikacja dostępna w językach: polskim i angielskim.
  - **Pomiar:** $X = A/N$
  - A - liczba elementów przetłumaczonych na język polski/angielski
  - N - liczba elementów interfejsu do przetłumaczenia
  - **Cel:** $X = 1$
- Modułowa konstrukcja systemu. Możliwość uruchomienia podsystemów niezależnie od siebie.
  - **Pomiar:** $X = A/N$
  - A - Ilość podsystemów działających niezależnie
  - N - Ilość podsystemów w systemie
  - **Cel:** $X = 1$
- Niezawodność działania aplikacji.
  - **Pomiar:** $X = A/T$
  - A - ilość błędów napotkanych podczas użytkowania aplikacji
  - T - czas użytkowania w minutach
  - **Cel:** $X <= 0.005$
- Bezpieczeństwo systemu.
  - **Pomiar:** N
  - N - ilość luk w zabezpieczeniach
  - **Cel:** $N = 0$
- Zapewnienie prostej i szybkiej migracji.
  - **Pomiar:** $X = T(A + 1)/\frac{N}{10}$
  - A - Ilość napotkanych problemów podczas migracji
  - T - Czas wykonania migracji w godzinach
  - N - Rozmiar bazy danych w GB
  - **Cel:** $X <= 5$

3. Plan beta testowania

3.1. System logowania

Szereg testów pozwalających stwierdzić poprawność systemu logowania, w tym wszystkie klasyczne przypadki brzegowe.

3.2. Bezpieczeństwo bazy danych

Dogłębne testy mające dowieść, że odpowiednie dane z bazy są dostępne tylko właściwym użytkownikom po zalogowaniu. Próby wyciągnięcia danych z bazy w sposób niepożądany.

3.3. Moduł nawigacyjny

Sprawdzenie poprawności działania nawigacji *Google Maps*, będącej elementem oprogramowania pod kątem poprawności, spójności i integralności z resztą aplikacji. Skontrolowanie generatora tras pod kątem poprawności i optymalności.

3.4. Obsługa kierowców, pojazdów i pojemników na śmieci

Szereg testów mający na celu wykazać poprawność działania wszystkich funkcjonalności związanych z dodawaniem, usuwaniem i edycją odpowiednio kieroców, pojazdów oraz pojemników na śmieci.

4. Plan zarządzania jakością

1. Regularne testowanie aplikacji przez cały czas życia oprogramowania przez różne podmioty.
2. Zbieranie i archiwizowanie informacji o ewentualnych błędach lub niedoskonałościach, w celu ich późniejszej naprawy.
3. Opracowanie statystyk i wyników testów dostępnych dla programistów.
4. Ścisłe kontrolowanie wszystkich członków zespołu przez menadżera w celu weryfikacji efektów pracy w kontekście zakładanych celów.
5. Audyt oprogramowania przeprowadzony przez firmę zewnętrzną po ukończeniu wersji Alpha aplikacji.

5. Plan wykonania produktu

Nr	Nazwa Zadania	Czas	Początek	Koniec
1.	Projektowanie	14 dni	14-09-2020	02-10-2020
2.	Wywiad środowiskowy	7 dni	02-10-2020	13-10-2020
3.	Stworzenie dokumentu wymagań	3 dni	13-10-2020	16-10-2020
4.	Opracowanie architektury systemu	7 dni	16-10-2020	27-10-2020
5.	Wstępny projekt interfejsu	5 dni	27-10-2020	03-11-2020
6.	Omówienie projektu z drużyną, podział pracy	2 dni	03-11-2020	06-11-2020
7.	Implementacja	60 dni	06-11-2020	04-02-2021
8.	Inicjalizacja repozytorium	2 dni	06-11-2020	10-11-2020
9.	Tworzenie bazy danych	4 dni	10-11-2020	17-11-2020
10.	Implementacja backendu	40 dni	17-11-2020	15-01-2021
11.	Implementacja frontendu	40 dni	17-11-2020	15-01-2021
12.	Tworzenie testów jednostkowych	14 dni	15-01-2021	04-02-2021
13.	Testowanie funkcjonalności	30 dni	04-02-2021	18-03-2021
14.	Testy jakościowe, testy klientów	14 dni	18-03-2021	08-04-2021
15.	Testy bezpieczeństwa	14 dni	08-04-2021	28-04-2021
16.	Wdrażanie	30 dni	28-04-2021	11-06-2021
17.	Poprawki wykrytych błędów	14 dni	11-06-2021	01-07-2021

6. Ocena zgodności

Udało się zachować pełnię zgodności ze wcześniej sporządzonymi dokumentami: Koncepcją wykonania systemu oraz specyfikacją wymagań.