**Siłka++ Faza konstrukcji**

Marcin Sarnecki, Marcin Wróbel

**1. Testy funkcjonalności**

I - Trener ogłasza się na portalu

| Krok | Opis | Oczekiwania |
| --- | --- | --- |
| 1 | Trener podaje swój  - **email** składający się z trzech części: lokalny adres, znaku @, domena,  przykładowo “jan.kowalski@gmail.com”  - **hasło** (składające się z co najmniej 8 znaków, w tym 1 znaku specjalnego, 1 cyfry, 1 wielkiej litery 1 małej litery), oraz klika przycisk **“Zaloguj”** | Aplikacja weryfikuje poprawność wprowadzonych danych i loguje użytkownika |
| 2 | Użytkownik klika odpowiedni przycisk **‘plus’** umożliwiający dodanie ogłoszenia | Aplikacja pokazuje formularz dodawania ogłoszenia |
| 3 | Użytkownik wypełnia wszystke wymagane pola wartościami  - **nazwa**: np. ‘Zajęcia indywidualne na siłowni’ będące dowolnym ciągiem znaków,  - **kategorie**: np. ‘trening siłowy, trening cardio’, będące elementami wybranymi z listy  - **lokalizacja**: np. ‘Wrocław’, będąca ciągiem znaków  - **dane kontaktowe**: np.  [**‘email’**: jan.kowalski@gmail.com,  **‘telefon’**: ‘+48 123456789’],  składają się z adresu email opisanego powyżej oraz numeru telefonu. Numer telefonu to numer kierunkowy poprzedzony znakiem plus, a następnie 9, albo 10 cyfr  - **dokładny opis:** np. ‘Zapraszam na indywidualny trening’, będący dowolnym ciągiem znaków  - **doświadczenie:** np. ‘2 lata’, będące okresem czasu  - **cena:** np. ‘60 zł/h’ składający się z liczby i jednostki | System wyświetla formularz dodawania ogłoszenia zawierający pole nazwy, listę kategorii, lokalizacji, danych kontaktowych, dokładnego opisu, dotychczasowego doświadczenia w zawodzie, cenę konsultacji |
| 4 | Użytkownik klika przycisk “**utwórz ogłoszenie”**. | Aplikacja dodaje ogłoszenie do bazy danych i wyświetla je |

II - Użytkownik tworzy swój plan treningowy

| Krok | Opis | Oczekiwania |
| --- | --- | --- |
| 1 | Użytkownik wchodzi w zakładkę **‘Treningi’** | System wyświetla dostępne opcje: przeglądaj zapisane plany treningowe; utwórz nowy plan treningowy; śledź postępy znajomych. |
| 2 | Użytkownik wybiera opcję **‘utwórz plan treningowy’** | System wyświetla początkowo pustą listę ćwiczeń w nowym planie treningowym, oraz pole wyszukiwania ćwiczeń, pole do wpisania nazwy. |
| 3 | Użytkownik dodaje nazwę planu składającą będącą **ciągiem znaków** | System sprawdza, czy nazwa planu jest niepusta, oraz to czy nie została już użyta |
| 4 | Użytkownik wpisuje w pole wyszukiwania interesującą go **nazwę ćwiczenia** będącą ciągiem znaków | System proponuje tylko ćwiczenia odpowiadające danym wpisanym w pole wyszukiwania |
| 5 | Użytkownik dodaje po kolei interesujące go ćwiczenia do swojego planu | System wyświetla odpowiednio coraz dłuższą listę ćwiczeń |
| 6 | Dla każdego ćwiczenia, użytkownik podaje odpowiadające mu parametry:  - **liczba powtórzeń** jako dodatnia liczba całkowita  - **czas wykonywania** jako dodatnia liczba całkowita w sekundach  - **ciężar** jako dodatnia liczba kg | System weryfikuje czy parametry zostały wprowadzone i są poprawne i wyświetla odpowiednią informację zwrotną |
| 7 | Użytkownik zatwierdza utworzenie całego planu poprzez kliknięcie przycisku **“utwórz”**. | System zapisuje nowy plan treningowy |

III - Wykonaniu treningu przez użytkownika

| Krok | Opis | Oczekiwania |
| --- | --- | --- |
| 1 | Użytkownik wchodzi w zakładkę **‘Treningi’** | System wyświetla dostępne opcje: przeglądaj zapisane plany treningowe; utwórz nowy plan treningowy; śledź postępy znajomych. |
| 2 | Użytkownik wybiera przeglądaj zapisane plany treningowe. | System wyświetla zapisane plany treningowe |
| 3 | Użytkownik wpisuje dowolny ciąg znaków w pole wyszukaj | System wyświetla te plany treningowe, które zawierają w nazwie, lub opisie ciąg znaków odpowiadający temu, co wpisał użytkownik. Jeżeli pole wyszukaj jest puste system wyświetla wszystkie plany treningowe |
| 4 | Użytkownik wybiera kategorie zaznaczając je w **liście kategori** | System wyświetla tylko te plany treningowe, które mają przypisaną co najmniej jedną z wybranych kategorii. Jeżeli użytkownik nie wybrał żadnej kategorii to system wyświetla wszystkie plany treningowe, które zgadzają się z polem wyszukiwania. |
| 5 | Użytkownik znajduje wybrany plan treningowy oraz klika przycisk **‘trenuj’** | Aplikacja będzie wyświetlała ćwiczenia po kolei |
| 6 | Po każdym wykonanym ćwiczeniu użytkownik klika przycisk **‘następne ćwiczenie’** | Dla każdego ćwiczenia aplikacja wyświetla odpowiednie parametry:  - liczbę powtórzeń (jeśli dotyczy)  - czas jako liczbę sekund (jeśli dotyczy)  - ciężar w kilogramach (jeśli dotyczy)  - opis ćwiczenia jako ciąg znaków  - zdjęcie |
| 7 | Po zakończonym treningu aplikacja prosi o zważenie się i podanie wagi będącej liczbą całkowitą dodatnią, użytkownik jeśli chce wpisuje swoją wagę | Aplikacja wyświetla podsumowanie treningu |

**2. Pomiary dla wymagań niefunkcjonalnych**

**2.1. Najgorszy czas odpowiedzi serwera - Worst case response time ratio**

X = Tmax / Rmax

Tmax - maksymalny czas odpowiedzi

Rmax - wymagany maksymalny czas odpowiedzi

**Cel:** X mniejszy od 1, im bliżej tego celu tym lepiej

**2.2 Przepustowość wykonywania zapytań przez serwer - Throughput time**

X = A / B

A - liczba ukończonych zapytań

B - czas obserwacji

**Cel:** jak największy X

**2.3 Częstość występowania awarii - Failure density**

X = NFAI / SIZE

NFAU - ilość wykrytych awarii

SIZE - rozmiar projektu

**Cel:** X jak najmniejsze, zmniejszające się w fazie testowania.

**2.4 Średni czas pomiędzy awariami - Mean time between failures**

a) X = TOPT / NAFI

b) Y = TSIB / NAFI

TOPT - czas działania

TSIB - suma czasów pomiędzy awariami

NAFI - ilość wykrytych awarii

**Cel:** Jak największe X,Y, zwiększające się w fazie testowania

**2.5 Dostępność aplikacji - Physical accessibility**

X = A / B

A - liczba funkcji, z których może korzystać osoba z pewną niepełnosprawnością

B - liczba funkcji

Wiadomo, że 0 <= X <= 1

**Cel:** jak największy X (najlepiej równy 1)

**2.6 Przystosowanie do systemów operacyjnych**

X = A / B

A - liczba obsługiwanych wersji systemów operacyjnych

B - planowana liczba obsługiwanych wersji systemów operacyjnych

Wiadomo, że 0 <= X <= 1

**Cel:** jak największy X (najlepiej równy 1)

**2.7 Szyfrowanie danych - Data encryption**

X = A / B

A - ilość danych, która jest szyfrowana

B - ilość danych, która powinna być zabezpieczona, szyfrowana

Wiadomo, że 0 <= X <= 1

**Cel:** X równy 1 (im bliżej 1, tym lepiej)

**2.8 Zapisywanie logów - Activity recording**

X = A / B

A - ilość zapisywanych danych podczas operacji

B - planowana ilość zapisywanych danych podczas operacji

Wiadomo, że 0 <= X <= 1

**Cel:** X równy 1 (im bliżej 1, tym lepiej)

**3. Plan beta testowania**

Planujemy podzielić beta testy na kilka faz, podczas których damy aplikację do przetestowania testerom i potencjalnym użytkownikom.

I - pierwsza faza

Na samym starcie chcielibyśmy przetestować najbardziej podstawowe funkcje naszej aplikacji, takie jak logowanie czy możliwość stworzenia własnego planu treningowego. Uczestniczyć w tym będą testerzy.

II - druga faza

Następnie chcielibyśmy przetestować bardziej zaawansowane funkcjonalności, takie jak dodawanie publicznych ogłoszeń przez trenerów oraz przeglądanie planów znajomych.

III - trzecia faza

Na końcu chcielibyśmy przetestować aplikację na grupie kilkudziesięciu chętnych użytkowników.

**4. Plan zarządzania jakością oprogramowania**

1. Każdy programista będzie miał obowiązek tworzenia testów jednostkowych

2. Dedykowani testerzy oprogramowania będą mieli za zadanie stworzenie testów integracyjnych w celu automatycznego sprawdzania poprawności zaimplementowanych funkcji.

3. Będziemy regularnie weryfikować to, czy spełnione są wymagania niefunkcjonalne

4. Będziemy używać systemu kontroli wersji (Git) w celu lepszego zarządzania kodem.

5. Zespół programistów będzie raportował swoje postępy podczas cotygodniowych spotkań.

6. Cały projekt będzie używał jednego stylu kodowania, który będzie sprawdzany przez gwarantowany formatery.

7. Będziemy starannie rozważali wszystkie uwagi testerów.

**5. Dokładniejszy plan wykonania produktu**

| **Nr** | **Zadanie** | **Czas** |
| --- | --- | --- |
| **1** | Rozmowa z zleceniodawcą | 1 dzień |
| **2** | Stworzenie tablicy koncepcyjnej | 1 dzień |
| **3** | Rozmowa z potencjalnymi odbiorcami aplikacji | 2 tygodnie |
| **4** | Stworzenie specyfikacji wymagań | 3 dni |
| **5** | Wybranie pracowników, podział pracy | 2 dni |
| **6** | Wstępna implementacja projektu | 80 dni |
| **7** | Przetestowanie aplikacji na pierwszej grupie testerów | 7 dni |
| **8** | Wprowadzenie poprawek na podstawie problemów wskazanych przez testerów | 45 dni |
| **9** | Druga iteracja testerów | 7 dni |
| **10** | Wprowadzenie poprawek na podstawie problemów wskazanych przez testerów | 14 dni |
| **11** | Weryfikacja wymagań niefunkcjonalnych | 7 dni |
| **12** | Udostępnienie aplikacji dla ograniczonej liczby użytkowników | 21 dni |
| **13** | Poprawa wykrytych błędów | 7 dni |
| **14** | Odbiór aplikacji przez zleceniodawcę i opublikowanie jej | 1 dzień |
| **15** | Marketing i promowanie aplikacji | 60 dni |

**6. Ocena zgodności wykonanych prac z wizją systemu i specyfikacją wymagań**

Główną różnicą pomiędzy poprzednią wizją a aktualnym stanem jest czas produkcji, który niestety się wydłużył z 6 do 8 miesięcy (nie wliczamy fazy marketingu). Wynika to z tego, że po dokładniejszej analizie złożoności projektu okazało się, że potrzebujemy więcej czasu na przetestowanie oprogramowania i wprowadzenie poprawek wykrytych podczas testów. Natomiast udało się nam opracować pomiary dla wszystkich założonych wymagań niefunkcjonalnych