Studencka Pracownia Inżynierii Oprogramowania

Instytut Informatyki Uniwersytetu Wrocławskiego

Krystian Jasionek, Jakub Skalski

Faza konstrukcji

Wrocław, 10 stycznia 2021

Wersja 0.6

Historia zmian dokonanych w dokumencie

Data	Numer wersji	Opis	Autor
2020-12-22	0.1	Utworzenie dokumentu	Krystian Jasionek
2020-12-23	0.2	Korekta dokumentu	Jakub Skalski
2020-12-06	0.3	Korekta dokumentu	Jakub Skalski
2021-01-07	0.4	Korekta dokumentu	Krystian Jasionek
2021-01-08	0.5	Korekta dokumentu	Jakub Skalski
2021-01-10	0.6	Korekta dokumentu	Krystian Jasionek
2021-01-14	0.7	Korekta dokumentu	Krystian Jasionek

Spis treści

1. Scenariusze przypadków użycia

1.1. Utworzenie własnego algorytmu

Scenariusz główny:

1. Użytkownik przechodzi do panelu algorytm.
2. Użytkownik wybiera opcję „Dodaj nowy algorytm".
3. Użytkownik wybiera opcję „Napisz nowy algorytm".
4. Otwiera się edytor kodu wewnątrz aplikacji.
5. Użytkownik zapisuje kod algorytmu.
6. Użytkownik wybiera opcję „Zapisz jako".
7. Użytkownik wprowadza nazwę algorytmu.
8. Algorytm zostaje dodany do biblioteki algorytmów użytkownika.

Testy dla kolejnych kroków scenariusza:

1. Panel algorytmu wyświetla się poprawnie.
2. Pojawia się okno wyboru między napisaniem własnego algorytmu a użyciem gotowego z biblioteki.
3. System otwiera okno edytora kodu
4. Edytor kodu zawiera wszystkie typowe funkcje edytora kodu (kopiowanie, wklejanie, zaznaczanie, wyszukiwanie frazy, powielanie), kolorowanie syntaktyczne tekstu, sprawdzanie składni oraz działający kompilator.
5. System pomyślnie przechowuje skompilowany kod.
6. System prezentuje pole wprowadzania nazwy dla przeznaczonego do zapisu kodu.
7. System sprawdza, czy nazwa zawiera jedynie znaki alfanumeryczne, spacje, podłogi i myślniki.
8. Biblioteka zostaje rozszerzona o algorytm o wybranej nazwie.

1.2. Zakup algorytmu w sklepie

Scenariusz główny:

1. Użytkownik przechodzi do panelu sklepu.
2. Użytkownik wybiera kategorię „Algorytmy”.
3. System wyświetla listę dostępnych algorytmów.
4. Użytkownik wybiera algorytm.
5. System otwiera podstronę opisującą algorytm.
6. Użytkownik wybiera opcję „Dodaj do koszyka”.
7. Użytkownik przechodzi do panelu „Mój koszyk”.
8. Użytkownik wybiera opcję „Przejdź do płatności”.
9. System wyświetla podsumowanie zamówienia i pyta o potwierdzenie kontynuacji.
10. Użytkownik potwierdza kontynuację.
11. System wyświetla formularz zakupu.
12. Użytkownik uzupełnia:
    - Imię (maksymalnie 100 znaków alfanumerycznych)
    - Nazwisko (maksymalnie 100 znaków alfanumerycznych)
    - Numer karty (dokładnie 16 cyfr)
    - Data ważności (4 cyfry w formacie mm-rr)
    - CVV (dokładnie 3 cyfry)
13. Użytkownik wybiera opcję „Zapłać”.
14. System wyświetla komunikat o poprawnym zakończeniu transakcji.
15. Algorytm zostaje dodany do biblioteki algorytmów użytkownika.

Testy dla kolejnych kroków scenariusza:

1. Panel sklepu wyświetla się poprawnie.
2. Wybranie kategorii wyświetla listę dostępnych algorytmów.
3. Lista zawiera wszystkie dostępne w sklepie algorytmy.
4. Wybranie opcji przenosi użytkownika do podstrony algorytmu.
5. Strona wyświetla opis algorytmu, przykładowy efekt działania, cenę oraz opcje dodania do koszyka.

6. Wybranie opcji dodaje algorytm do koszyka użytkownika.
7. Strona poprawnie wyświetla listę wszystkich produktów w koszyku wraz z ich nazwą i ceną.
8. System przenosi użytkownika do strony z podsumowaniem zamówienia.
9. Okno podsumowania zamówienia zawiera łączną cenę produktów, opcje płatności i przycisk potwierdzający kontynuację.
10. System przenosi użytkownika do formularza zakupu.
11. Strona wyświetla formularz zakupu zawierający pola na imię, nazwisko, numer karty, jej data ważności i kod CVV.
12. Pola na imię i nazwisko przyjmują ciąg od 1 do 100 znaków, na numer karty dokładnie 16 cyfr, na datę ważności 4 cyfry w formacie mm-rr oraz na CVV dokładnie 3 cyfry.
13. Strona wyświetla kręciołka (ang. *throbber*).
14. System powiadamia o przebiegu transakcji – w przypadku niepowodzenia wyświetla kod błędu, w przypadku sukcesu informuje o pomyślnym przebiegu transakcji.
15. Biblioteka algorytmów użytkownika została powiększona o zakupione algorytmy.

1.3. Eksport gotowych wzorów

Scenariusz główny

1. Użytkownik przechodzi do panelu kreatora wzorów.
2. Użytkownik przygotowuje wzór.
3. Użytkownik wybiera opcję „Wyeksportuj wzór”.
4. System wyświetla listę dostępnych formatów graficznych.
5. Użytkownik wybiera format pliku.
6. System wyświetla przeglądarkę plików na komputerze użytkownika.
7. Użytkownik wybiera docelową nazwę pliku.
8. System eksportuje wzór do wskazanego formatu i przesyła go na komputer użytkownika.
9. System powiadamia użytkownika o zakończonej pomyślnie operacji.

Testy dla kolejnych kroków scenariusza:

1. Kreator wzorów wyświetla się poprawnie.
2. Wzór powstaje zgodnie z wprowadzonymi przez użytkownika parametrami. Jego wygląd odpowiada podglądowi gotowego wzoru.
3. Wybranie opcji otwiera listę dostępnych formatów graficznych.
4. Wyświetlana lista zawiera wszystkie dostępne formaty.
5. System pobiera informację o wybranym rozszerzeniu.
6. System otwiera przeglądarkę plików na komputerze użytkownika.
7. Udaje się wprowadzić nazwę pliku.
8. System wyświetla pasek postępu eksportu pliku. Plik o wybranym wcześniej formacie i nazwie zostaje utworzony na komputerze użytkownika i zawiera wygenerowany wzór.
9. System powiadamia o przebiegu operacji – w przypadku niepowodzenia wyświetla kod błędu, w przypadku sukcesu wyświetla nazwę, datę utworzenia, folder zawierający i rozmiar pliku.

2. Pomiary dla wymagań niefunkcjonalnych

2.1. Klarowność interfejsu

Przejrzystość i intuicyjność interfejsu powinna być jak największa, użytkownik nie może spędzać wiele czasu na domyślaniu się działania jego elementów.

Wyznacznik: X - średni czas nawigacji po interfejsie, $[X] = \text{s}$.

- n_p - liczba przypadków, w których wystąpiły problemy przy interakcji użytkownika z interfejsem.
- t_{total} - łączny czas rozwiązywania problemów
- $X = \frac{t_{total}}{n_a}$

Chcemy, aby $X \leqslant 5$ s.

2.2. Płynność działania

Średni czas realizacji algorytmu oraz generowania podglądu powinien być możliwie jak najkrótszy.

Wyznacznik: X - średni czas wykonania algorytmu, $[X] = \text{s}$.

- n_a - liczba wykonanych algorytmów.
- t_{total} - łączny czas wykonania algorytmów.

- $X = \frac{t_{total}}{n_a}$

Chcemy, aby $X \leqslant 1$ s.

2.3. Bezpieczeństwo danych

Wszelkie informacje należy przechowywać w postaci zaszyfrowane przy pomocy funkcji z rodziny SHA-2.

Wyznacznik: X - przewidywany czas potrzebny do rozszyfrowania danych, $[X] = \text{s}$.

- p - liczba możliwych szyfrowań wygenerowanych przez wybraną funkcję skrótu.
- c - liczba szyfrowań przetwarzanych przez komputer w czasie jednej sekundy.
- $X = \frac{p}{c}$

Chcemy, aby $X \geqslant 2^{64}$ s.

2.4. Niezawodność

System powinien być wolny od błędów i usterek.

Wyznacznik: X - średnia liczba problemów na sekundę, $[X] = \frac{1}{\text{s}}$.

- n_p - liczba problemów, które wystąpiły w czasie korzystania z aplikacji.
- t_{total} - łączny czas korzystania z aplikacji.
- $X = \frac{n_p}{t_{total}}$

Chcemy, aby $X \leqslant 10^{-6}\ \frac{1}{\text{s}}$.

2.5. Przystępna dokumentacja

Powinna stanowić samouczek dla nowych użytkowników, zawierać szczegółowy opis wszelkich elementów interfejsu, narzędzi, funkcji oprogramowania.

System powinien być wolny od błędów i usterek.

Wyznacznik: X - średnia liczba problemów na sekundę, $[X] = \frac{1}{\text{s}}$.

- n_q - łączna liczba pytań użytkowników zadanych podczas testów.
- n_a - liczba pytań, na które odpowiedź znajduje się w dokumentacji.
- $X = \frac{n_a}{n_q}$

Chcemy, aby $X \geqslant 0.95\ \frac{1}{\text{s}}$.

3. Plan beta testowania

Testowanie naszej aplikacji będzie przebiegać w dwóch fazach – wewnętrznej i zewnętrznej.

3.1. Faza wewnętrzna

Będzie przeprowadzona przez zespół programistyczny składający się z sześciu osób oraz dwóch testerów. Każdy programista będzie zobowiązany do przetestowania fragmentu aplikacji, który wykonał. Testerzy zadbają o sprawdzenie gotowej aplikacji przed wypuszczeniem jej na rynek. W tej fazie główny nacisk zostanie położony na:

- niezawodność i płynność wykonywania algorytmów,
- funkcjonalność interfejsu aplikacji, jego intuicyjność i atrakcyjność,
- pomyślne wczytywanie, przetwarzanie i wyświetlanie algorytmów sklepu internetowego oraz bezpieczeństwo przy ich zakupie
- mechanizmy społecznościowe, takie jak komentowanie i ocenianie algorytmów,
- bezpieczeństwo baz danych.

Przewidywany czas testowania po ukończeniu programowania aplikacji – 1 miesiąc.

3.2. Faza zewnętrzna

Niedługo po zakończeniu fazy wewnętrznej testów przed wypuszczeniem produktu na rynek rozpocznie się faza zewnętrzna. Otwarty zostanie program beta testów, do którego będą rekrutowani użytkownicy końcowi. Ta faza będzie miała miejsce przed kampanią marketingową aplikacji, zatem próbka testerów będzie relatywnie mała (domyślnie celujemy w nie mniej niż tysiąc osób). Testerzy będą opłacani na podstawie ilości znalezionych błędów.

4. Plan wykonania projektu

Harmonogram		
Numer	Nazwa etapu	Czas
1	Opracowanie algorytmów	30 dni
2	Przygotowanie odpowiedniego interfejsu programowania aplikacji (ang. *API*)	30 dni
3	Przygotowanie kompilatora kodu algorytmu	60 dni
4	Wstępny projekt interfejsu	10 dni
5	Opracowanie okna podglądu algorytmu	60 dni
6	Zbudowanie baz danych	20 dni
7	Integracja sklepu i profilów użytkownika z bazami danych	10 dni
8	Wykończenie interfejsu	20 dni
9	Dodanie funkcji eksportu plików	30 dni
10	Testy wewnętrzne	30 dni
11	Testy zewnętrzne	30 dni

5. Plan zarządzania jakością wytwarzania oprogramowania

Na podstawie cykl Deminga oraz standard ISO 9000, przedstawione na wykładzie, wyróżniliśmy następujące etapy planu ZJO:

1. **Planowanie** - wykonany zostanie harmonogram realizacji określonych zadań, powstanie standard jakości kodu, którego przestrzegać będą mieli programiści, poszczególne zadania zostaną rozdysponowane między członków zespołu.

2. **Wykonanie** - w utrzymaniu wysokiej jakości pomoże stałe testowanie ukończonych fragmentów aplikacji, przez pracujących nad nimi programistów. Rolą kierownika projektu będzie zadbanie o utrzymanie prawidłowego tempa pracy oraz zgodności napisanego kodu z przyjętym wcześniej standardem. Co tydzień organizowane będą spotkania, na których członkowie zespołu zaprezentują ukończone przez siebie zadania i grupowo ocenią je pod kątem poprawności, zgodności z projektem i jakości.

3. **Sprawdzenie** - analizując wyniki testów, opracowywane będą plany poprawy danych fragmentów aplikacji, np. jej kodu lub interfejsu. Zespół będzie miał na celu przedyskutowanie możliwych opcji realizacji usprawnień, wybrać najlepsze z nich i przygotować do wdrożenia.

4. **Poprawianie** - wcześniej zaplanowane usprawnienia zostają wprowadzone do systemu, a wykryte błędy usunięte. W dalszym ciągu przeprowadzane będą testy, których wyniki wskażą na fragmenty aplikacji wymagające dalszych usprawnień.

6. Ocena zgodności pracy

Koncepcja wykonania naszego systemu jest wierna opisowi zawartemu w tablicy koncepcyjnej oraz z fazą opracowania oprogramowania. Obecna analiza projektu nie wskazuje na konieczność wprowadzania zmian w dokumentacji.

7. Przypisy

- Standard zapisu jednostek fizycznych `http://www.foton.if.uj.edu.pl/documents/12579485/cb289cdc-65ba-4770-8dbf-9685be2f466f`