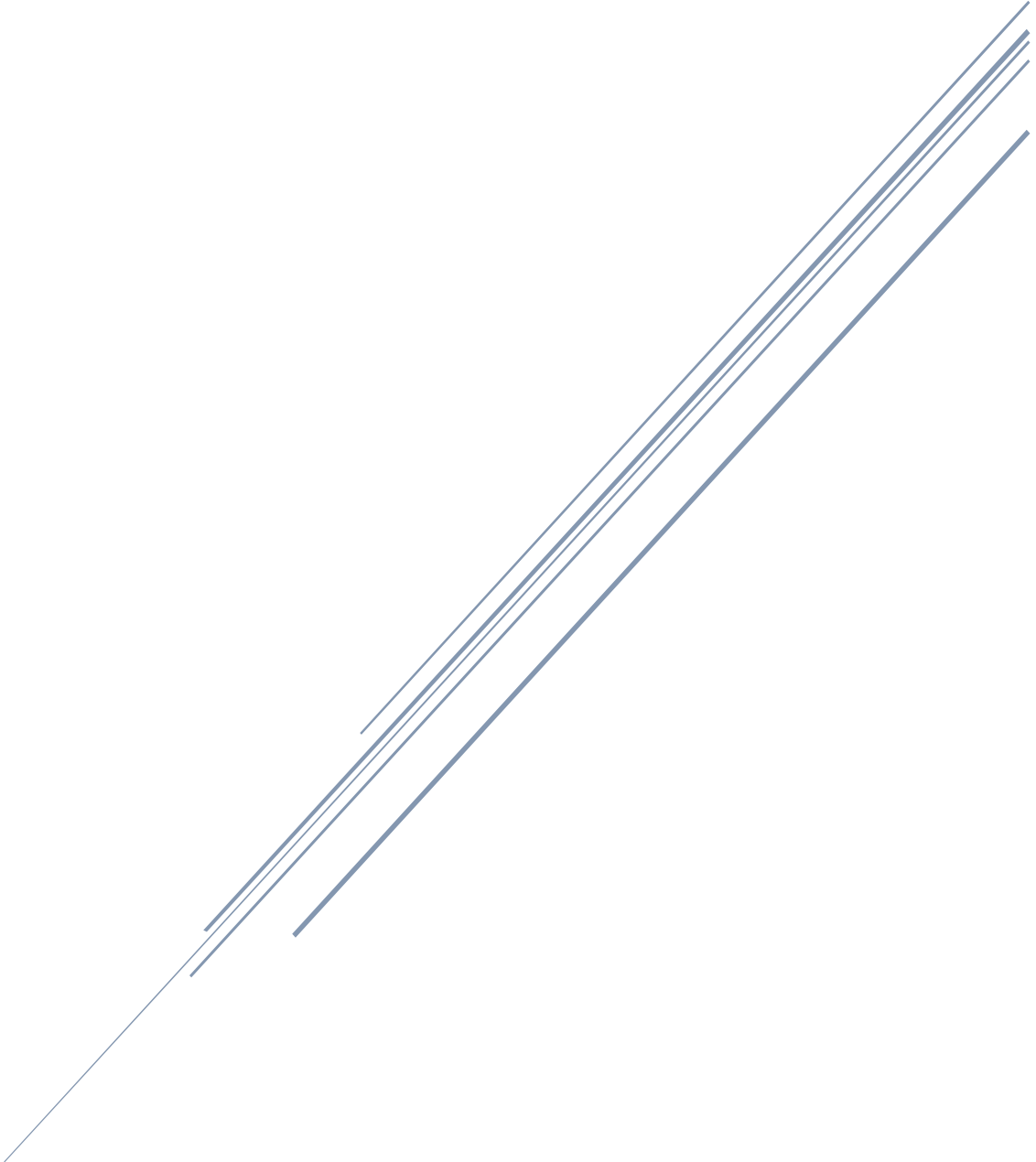


DOKUMENTACJA TESTOWA

Coin Hunter



Uniwersytet Jagielloński, Wydział Matematyki i Informatyki
Inżyniera Oprogramowania

Spis treści

1. Wprowadzenie	2
1.1. Cele testów	2
1.2. Narzędzia	2
1.3. Plan testów	2
1.4. Definicje, skróty i akronimy	3
2. Testy jednostkowe	4
2.1. Test jednostkowy J1	4
2.2. Test jednostkowy J2	4
2.3. Test jednostkowy J3	4
2.4. Test jednostkowy J4	5
2.5. Test jednostkowy J5	5
2.6. Test jednostkowy J6	5
2.7. Test jednostkowy J7	5
2.8. Test jednostkowy J8	6
2.9. Test jednostkowy J9	6
2.10. Test jednostkowy J10	6
2.11. Test jednostkowy J11	6
2.12. Test jednostkowy J12	7
2.13. Test jednostkowy J13	7
3. Testy integracyjne	8
3.1. Test integracyjny I1	8
3.2. Test integracyjny I2	8
4. Testy systemowe	9
5. Testy wymagań niefunkcjonalnych	9
5.1. Test wymagań niefunkcjonalnych T-NFR1.1	9
5.2. Test wymagań niefunkcjonalnych T-NFR1.2	9
5.3. Test wymagań niefunkcjonalnych T-NFR1.3	10
6. Testy akceptujące	11
6.1. Rezultat	11
7. Kryteria zakończenia testów	11
8. Mowa końcowa	12

1. Wprowadzenie

Niniejszy dokument przedstawia dokumentację testową aplikacji „Coin Hunter” – aplikacji przeglądarkowej wraz z bazą danych i grą przeglądarkową.

1.1. Cele testów

Głównym celem testów jest wyeliminowanie jak największej ilości mankamentów z aplikacji. Dodatkowo proces testowy skupia się również na znalezieniu niedoskonałości wynikających z na przykład zbyt szybkiego wykonania animacji. Zatem testy docelowo mają sprawdzić warstwę wizualną, jak i funkcjonalną.

Testy będą się skupiać na następujących punktach:

1. Czy aplikacja spełnia wszystkie wymagania (funkcjonalne i niefunkcjonalne) określone w dokumentacji,
2. Poczucie estetyki przy realizacji animacji,
3. Błędy ludzkie,
4. Niestandardowe zachowania,
5. Przyzwyczajenia z innych tego typu aplikacji,
6. Bezpieczeństwo.

Strefy i funkcjonalności, które nie podlegały testowaniu:

1. Połączenie internetowe,
2. Zasoby bazy danych.

1.2. Narzędzia

Z uwagi na charakter aplikacji wszystkie testy sprowadziły się do testów manualnych. Wykonanie testów automatycznych nie ma zastosowania, gdyż aplikacja nie analizuje wielu alternatywnych scenariuszy i wariantów postępowania.

1.3. Plan testów

Testy wykonane będą zgodnie z następującą kolejnością poziomów:

- Jednostkowe,
- Integracyjne,
- Systemowe,
- Akceptacyjne.

Dodatkowo w testach widać pewną tendencję do większego nakładu testowego do części z rozgrywką. Jest to uzasadnione faktem, iż nie powstawała ona z gotowych elementów, które były wcześniej sprawdzone w innych aplikacjach.

1.4. Definicje, skróty i akronimy

Numer, nazwa	Definicja
[1], Awaria	Odchyłka modułu lub systemu od oczekiwanego zachowania lub rezultatu działania
[2], Błąd	Działanie człowieka powodujące powstanie nieprawidłowego rezultatu
[3], Defekt	Wada modułu lub systemu, która może spowodować, że moduł lub system nie wykona zakładanej czynności
[4], Incydent	Każde zdarzenie wymagające omówienia
[5], Byt	Element w grze, pojęcie to używane jest głównie jako określenie na elementy, z którymi postać wchodzi w interakcje.
[6], Moneta	Element, który należy kolekcjonować podczas rozgrywki, jeden z kryteriów formowania rankingu.
[7], Przeciwnik	Byt, z którym należy unikać kolizji podczas rozgrywki.
[8], Postać	Element, którym sterujemy podczas rozgrywki
[9], Dokumentacja	Termin ten odnosi się do całej powstałej dokumentacji projektowej. (a nie tylko do testowej)

2. Testy jednostkowe

Na tej grupie podczas testowania skupiliśmy się najbardziej, wychodząc z założenia, iż jeśli małe elementy będą poprawnie skonstruowane, to nie będzie problemu na kolejnych etapach. Dzięki szczegółowemu podejściu w tej fazie mogliśmy skupić się również na realizacji części wymagań funkcjonalnych i niefunkcjonalnych podanych w dokumentacji.

Poniżej przedstawione są testowane jednostki wraz z dokładnym opisem rezultatu.

2.1. Test jednostkowy J1

ID: J1

Nazwa: Test skoku

Rezultat: Defekt

Opis:

Zbyt długi czas oczekiwania na kolejny skok po wylądowaniu.

Wpływ:

Brak możliwości wykonania kolejnego skoku w krótkim interwale czasowym, by zareagować na pojawiające się byty.

Rozwiązanie:

Zbadanie metodą prób i błędów optymalnego czasu wykonania skoku oraz spasowanie czasu oczekiwania na koniec animacji z czasem trwania animacji.

2.2. Test jednostkowy J2

ID: J2

Nazwa: Zbieranie monet - animacja

Rezultat: Pozytywny

Opis:

Poprawność wykonania animacji

2.3. Test jednostkowy J3

ID: J3

Nazwa: Zbieranie monet - interakcja

Rezultat: Pozytywny

Opis:

Poprawne zachowanie podczas interakcji z postacią, brak opóźnienia.

2.4. Test jednostkowy J4

ID: J4

Nazwa: Licznik monet - animacja

Rezultat: Pozytywny

Opis:

Animacja wykonała się poprawnie, bez opóźnień i z odpowiednią płynnością.

2.5. Test jednostkowy J5

ID: J5

Nazwa: Licznik monet – funkcjonalność

Rezultat: Pozytywny

Opis:

Poprawna inkrementacja wartości monet.

2.6. Test jednostkowy J6

ID: J6

Nazwa: Starcie z przeciwnikiem – interakcja

Rezultat: Awaria: interakcja z przeciwnikiem

Opis:

Widoczny kontakt nie powoduje interakcji.

Wpływ:

Brak odpowiedniej reakcji na kontakt z przeciwnikiem.

Rozwiązanie:

Podział animacji na fazy czasowe, by umożliwić odpowiednią reakcję na kontakt z przeciwnikiem.

2.7. Test jednostkowy J7

ID: J7

Nazwa: Starcie z przeciwnikiem – animacja

Rezultat: Awaria: błysk po kontakcie z przeciwnikiem

Opis:

Po stracie jednego z żyć, pojawia się animacja błysku, jednak podczas jednego z testów okazała się ona słabo widoczna.

Wpływ:

Konfundujące wrażenie podczas rozgrywki, niedokładność animacji psuje estetykę warstwy wizualnej.

Rozwiązanie:

Zwiększenie czasu trwania animacji błysku.

2.8. Test jednostkowy J8

ID: J8

Nazwa: Dekrementacja serc – funkcjonalność

Rezultat: Defekt: usuwanie serc

Opis:

Opóźnienie funkcji, usuwającej sprawia, że zanim usunie dany parametr, zostaje on zmieniony i przez to wykonywana jest praca na innym (nieistniejącym) elemencie.

Wpływ:

Błąd w konsoli, praca na -1 elemencie tablicy.

Rozwiązanie:

Dodanie warunku by uniknąć pracy na elementach mniejszych niż 0.

2.9. Test jednostkowy J9

ID: J9

Nazwa: Naliczanie dystansu – animacja

Rezultat: Pozytywny

Opis:

Animacja w sposób poprawny ukazuje aktualny dystans, brak opóźnień.

2.10. Test jednostkowy J10

ID: J10

Nazwa: Naliczanie dystansu – funkcjonalność

Rezultat: Defekt: naliczanie dystansu jeszcze przed startem

Opis:

Naliczanie przebytego dystansu jeszcze podczas odliczania do startu rozgrywki.

Wpływ:

Fałszywe dane dotyczące dystansu, defekt wizualizacji.

Rozwiązanie:

Opóźnienie startu funkcji wyliczającej przebyte dystans.

2.11. Test jednostkowy J11

ID: J11

Nazwa: Procedura startowa – animacja

Rezultat: Pozytywny

Opis:

Procedura startowa wykonuje się poprawnie, przejścia są płynne i bez opóźnień.

2.12. Test jednostkowy J12

ID: J12

Nazwa: Stan końcowy – animacja

Rezultat: Pozytywny

Opis:

Odpowiednie napisy są podświetlone podczas najechania, wszystkie dane wyświetlone są poprawne.

2.13. Test jednostkowy J13

ID: J13

Nazwa: Stan końcowy – funkcjonalność

Rezultat: Pozytywny

Opis:

Przycisk rozpoczęcia gry od nowa odpowiednio resetuje stan gry, dane końcowe są zgodne ze stanem faktycznym.

3. Testy integracyjne

Aplikacja posiadała tylko dwa punkty krytyczne wymagające przetestowania integracji pomiędzy technologiami. Obrazują je dwa poniższe rozważone przypadki testowe.

3.1. Test integracyjny I1

ID: I1

Nazwa: Połączenie aplikacji z bazą danych

Rezultat: Pozytywny

Opis:

Aktualizacja danych przebiega poprawnie, nie ma zbędnego opóźnienia, dane nie są zmieniane podczas aktualizacji.

3.2. Test integracyjny I2

ID: I2

Nazwa: Przekazanie danych z gry do aplikacji przeglądarkowej

Rezultat: Pozytywny

Opis:

Dane przekazane są bez zbędnej zwłoki i bez fałszowania rezultatu monet i przebytego dystansu.

4. Testy systemowe

Testy systemowe są niejako zwieńczeniem połączenia wszystkich modułów w jeden. Zatem podczas tych procedur skupiliśmy się na początkowych założeniach podanych w dokumentacji. Na tym etapie mogliśmy w pełni sprawdzić wymagania zarówno funkcjonalne jak i niefunkcjonalne. Fuzją dwóch wcześniej wymienionych wymagań jest poprawna realizacja przypadków użycia co również zostało poddane weryfikacji.

5. Testy wymagań niefunkcjonalnych

5.1. Test wymagań niefunkcjonalnych T-NFR1.1

ID: T-NFR1.1

Nazwa: Obsługiwane przeglądarki

Opis:

Aplikacja powinna działać na trzech najpopularniejszych przeglądarkach w Polsce według rankingu umieszczonego na stronie podanej przez klienta. Według wspomnianego rankingu są to: „Google Chrome”, „Mozilla Firefox”, „Opera”.

Ad.:

Testy wykonane były na następujących przeglądarkach: Mozilla Firefox, Google Chrome, Opera.

Testom podlegały scenariusze dołączone do dokumentacji.

Podczas testów nie znaleziono incydentów wymagających poprawy, zachowanie aplikacji było zgodne z założonymi przypadkami użycia.

5.2. Test wymagań niefunkcjonalnych T-NFR1.2

ID: T-NFR1.2

Nazwa: Wymagania dotyczące wieku

Opis:

Strona powinna być dostosowana dla osób, które mają 7 lat.

Ad.:

Test polegał na obejrzeniu przez testera wszystkich podstron i sprawdzeniu czy nie znajdują się tam treści nieodpowiednie, które wymienione były w wymaganiach.

Podczas testu nie znaleziono treści wykraczających poza przyjęte w wymaganiach normy.

5.3. Test wymagań нефunkcjonalnych T-NFR1.3

ID: T-NFR1.3

Nazwa: Możliwość zgłaszania błędów

Opis:

Aplikacja powinna umożliwić użytkownikom zgłaszanie napotkanych przez nich błędów.

Ad.:

Test polegał na odnalezieniu miejsca, w którym jest możliwość zgłoszenia incydentu. Następnym krokiem było wysłanie wiadomości poprzez pole do tego przeznaczone

i sprawdzenie czy wiadomość dotarła na wyszczególniony adres mailowy.

Ostatnim krokiem testu było upewnienie się, że adres kopiuje się za pomocą przycisku.

Nie wykryto żadnych defektów, aplikacja zachowywała się zgodnie z założeniami. Mail wysłał się na poprawny adres bez opóźnień.

6. Testy akceptujące

Jedna z końcowych wersji produktu została poddana testom akceptującym przez grupę około 20 osób. Podczas sesji testowej nie zostały zgłoszone incydenty, które nie byłyby przewidziane w procedurach.

6.1. Rezultat

Na podstawie przeprowadzonej kontroli aplikacji nie zostały zgłoszone nieprzewidziane zachowania, czy też braki i niedociągnięcia. Zatem wszelkie wykonane dotychczas poczynania sprawiają, że testy akceptujące zakończyły się rezultatem pozytywnym, co implikuje gotowość produktu.

7. Kryteria zakończenia testów

1. Zgodność aplikacji z wymaganiami przedstawionymi w dokumentacji,
2. Wykonanie testów manualnych z zadowalającym rezultatem,
3. Pozytywna ocena warstwy wizualnej,
4. Stwierdzenie braku incydentów w kodzie (np. komunikat konsolowy).

8. Mowa końcowa

Wykonane testy znacząco przyczyniły się do poprawnego działania aplikacji, jak i jej dobrej prezentacji. Należy dostrzec wkład wszelkich wcześniej przygotowanych zasobów wspomagających testowanie oraz samo należyte zaplanowanie samego procesu. Dzięki temu weryfikacje przebiegły sprawnie, co oszczędziło czas i zasoby. Dodatkowo testy spełniły swoją rolę, minimalizując ilość możliwych incydentów. Praktycznie do zera zredukowały luki w bezpieczeństwie. Ponadto niebagatelnie uodporniły produkt na zachowania nietypowe i skrajne.

Dobrze zaplanowany i wykonany proces testowy oraz możliwe scenariusze wraz z dobrze sprecyzowanymi wymaganiami sprawiają, że finalny produkt spełnia swoje założenia i działa w należyty oraz przewidziany wcześniej sposób.