Plan projektu

Sklep z grami komputerowymi

Wersja 2024-04-16

|  |  |  |
| --- | --- | --- |
|  | **Imię nazwisko** | **Data** |
| Autor: | Adam Bień | 14.06.2024r. |
| Autor: | Patryk Słomian | 14.06.2024r. |

Spis treści

[**Sklep z grami komputerowymi** 7](#_Toc174891346)

[1. Wstęp 7](#_Toc174891347)

[2. Opis projektu 7](#_Toc174891349)

[3. Struktury organizacyjne projektu 7](#_Toc174891351)

[4. Role i odpowiedzialność 9](#_Toc174891352)

[5. Harmonogram projektu 12](#_Toc174891437)

[6. Kosztorys projektu 12](#_Toc174891438)

[7. Zasoby projektu 12](#_Toc174891451)

[8. Standardy i narzędzia w projekcie 13](#_Toc174891474)

[9. Procesy zarządzania 13](#_Toc174891483)

[9.1. Plan zarządzania konfiguracją 13](#_Toc174891484)

[9.2. Plan zarządzania ryzykiem 14](#_Toc174891485)

[9.3. Plan zarządzania testami 14](#_Toc174891486)

[9.4. Plan zapewnienia jakości 14](#_Toc174891487)

[9.5. Plan zarządzania zmianami 14](#_Toc174891488)

**Sklep z grami komputerowymi**

1. Wstęp

Celem projektu jest stworzenie zaawansowanego sklepu z grami komputerowymi, który umożliwi użytkownikom przeglądanie, wyszukiwanie oraz zakup gier komputerowych. Program zostanie napisany w języku C# i będzie zawierał zarówno interfejs użytkownika, jak i backend do zarządzania danymi gier oraz transakcjami.

1. Opis projektu

Projekt "Sklep z grami komputerowymi" ma na celu stworzenie nowoczesnej i intuicyjnej platformy zakupowej dla miłośników gier komputerowych. Platforma będzie umożliwiać użytkownikom przeglądanie oferty gier, wyszukiwanie według różnych kryteriów, dodawanie gier do koszyka oraz finalizowanie zakupu w sposób szybki i bezpieczny. Projekt będzie zrealizowany z wykorzystaniem języka programowania C# oraz technologii ASP.NET Core. Projekt ten dostarczy użytkownikom wygodne narzędzie do zakupu gier, a administratorom umożliwi efektywne zarządzanie asortymentem. Dzięki zastosowaniu nowoczesnych technologii i najlepszych praktyk programistycznych, system będzie wydajny, bezpieczny i łatwy w utrzymaniu, co zapewni jego długoterminowy rozwój i możliwość rozbudowy o nowe funkcjonalności w przyszłości.

1. Struktury organizacyjne projektu

**Zespoły w projekcie**

1. Zespół Zarządzający

* Project Manager (PM): Odpowiada za planowanie, organizację i nadzór nad całością projektu. Koordynuje pracę wszystkich zespołów, ustala priorytety i harmonogramy, zarządza ryzykami oraz budżetem projektu.
* Product Owner (PO): Odpowiada za wizję produktu, zbieranie wymagań od interesariuszy oraz priorytetyzację zadań w backlogu. Zapewnia, że zespół pracuje nad najbardziej wartościowymi funkcjonalnościami.

1. Zespół Analityczny

* Analityk Biznesowy: Zbiera i analizuje wymagania funkcjonalne oraz niefunkcjonalne, tworzy dokumentację wymagań oraz specyfikacje systemu. Współpracuje z Product Ownerem w celu zrozumienia potrzeb użytkowników końcowych.

1. Zespół Programistyczny

* Lead Developer: Kieruje zespołem programistów, podejmuje kluczowe decyzje technologiczne i architektoniczne, zapewnia zgodność kodu z najlepszymi praktykami i standardami.
* Backend Developers: Odpowiadają za implementację logiki biznesowej, integrację z bazą danych, implementację API oraz zarządzanie bezpieczeństwem aplikacji.
* Frontend Developers: Zajmują się tworzeniem interfejsu użytkownika, implementacją komponentów interfejsu, zapewnieniem responsywności i optymalizacji wydajności.

1. Zespół Testowy

* Test Manager: Koordynuje prace zespołu testowego, opracowuje strategię testowania, harmonogramy i zarządza raportowaniem wyników testów.
* Testerzy: Przeprowadzają testy funkcjonalne, niefunkcjonalne, regresyjne, integracyjne oraz akceptacyjne. Tworzą scenariusze testowe i raportują błędy.

1. Zespół DevOps

* DevOps Engineer: Odpowiada za automatyzację procesów wdrożeniowych, zarządzanie środowiskami developerskimi, testowymi i produkcyjnymi oraz monitorowanie wydajności i dostępności aplikacji.

1. Zespół Wsparcia Technicznego

* Support Specialist: Udziela wsparcia użytkownikom końcowym, rozwiązuje zgłaszane problemy techniczne oraz prowadzi dokumentację zgłoszeń.

**Przepływ danych między zespołami**

1. Zespół Zarządzający → Zespół Analityczny

* Przekazywanie wizji produktu, głównych celów biznesowych oraz kluczowych wymagań.

1. Zespół Analityczny → Zespół Programistyczny

* Dokumentacja wymagań, specyfikacje techniczne, przypadki użycia oraz diagramy procesów.

1. Zespół Programistyczny → Zespół Testowy

* Implementacja kodu źródłowego, specyfikacja techniczna oraz informacje o nowych funkcjonalnościach do testowania.

1. Zespół Programistyczny → Zespół DevOps

* Kod aplikacji, skrypty wdrożeniowe, konfiguracje środowisk.

1. Zespół Testowy → Zespół Programistyczny

* Raporty z testów, zgłoszenia błędów, sugestie poprawy kodu.

1. Zespół DevOps → Zespół Programistyczny/Zespół Testowy

* Feedback dotyczący wydajności i stabilności aplikacji, problemy z wdrożeniem.

1. Zespół Wsparcia Technicznego → Zespół Programistyczny

* Zgłoszenia problemów od użytkowników, propozycje poprawek i nowych funkcjonalności.

**Zasady organizacji spotkań**

1. Codzienne spotkania zespołów (Daily Stand-up)
   * Czas: 15 minut
   * Uczestnicy: Wszystkie zespoły
   * Cel: Omówienie postępów, planów na dany dzień oraz ewentualnych blokad.
2. Tygodniowe spotkania zespołów (Sprint Planning)
   * Czas: 1-2 godziny
   * Uczestnicy: Zespół Zarządzający, Zespół Programistyczny, Zespół Testowy
   * Cel: Planowanie zadań na nadchodzący sprint, priorytetyzacja backlogu.
3. Dwutygodniowe spotkania przeglądowe (Sprint Review)
   * Czas: 1-2 godziny
   * Uczestnicy: Wszystkie zespoły, interesariusze
   * Cel: Przegląd ukończonych prac, demonstracja nowych funkcjonalności, zebranie feedbacku.
4. Retrospektywa sprintu
   * Czas: 1 godzina
   * Uczestnicy: Zespół Programistyczny, Zespół Testowy, Project Manager
   * Cel: Omówienie co poszło dobrze, co można poprawić, planowanie usprawnień.
5. Spotkania ad-hoc
   * Uczestnicy: W zależności od potrzeby
   * Cel: Rozwiązywanie nagłych problemów, omawianie kluczowych decyzji technicznych.
6. Miesięczne spotkania strategiczne
   * Czas: 2 godziny
   * Uczestnicy: Zespół Zarządzający, Lead Developer, Test Manager
   * Cel: Przegląd postępów projektu, analiza ryzyk, planowanie długoterminowych działań.
7. Role i odpowiedzialność

**Role w projekcie**

1. Project Manager (PM)

* Zakres obowiązków:
  + Planowanie i organizacja projektu
  + Nadzór nad realizacją zadań
  + Zarządzanie harmonogramem i budżetem
  + Koordynacja prac zespołów
  + Zarządzanie ryzykiem
* Kompetencje:
* Doświadczenie w zarządzaniu projektami IT
* Umiejętności organizacyjne i przywódcze
* Znajomość metodologii zarządzania projektami
* Doskonałe umiejętności komunikacyjne

1. Product Owner (PO)

* Zakres obowiązków:
  + Definiowanie wizji produktu
  + Zbieranie i priorytetyzacja wymagań
  + Współpraca z interesariuszami
* Kompetencje:
  + Znajomość rynku gier komputerowych
  + Umiejętność analizy potrzeb biznesowych
  + Silne umiejętności komunikacyjne i negocjacyjne
  + Doświadczenie w pracy z zespołami deweloperskimi

1. Analityk Biznesowy

* Zakres obowiązków:
  + Zbieranie i dokumentowanie wymagań
  + Tworzenie specyfikacji funkcjonalnych
  + Współpraca z Product Ownerem i zespołem programistycznym
* Kompetencje:
  + Umiejętność analizy i rozwiązywania problemów
  + Doskonałe umiejętności dokumentacyjne
  + Doświadczenie w analizie systemów IT

1. Lead Developer

* Zakres obowiązków:
  + Kierowanie zespołem programistycznym
  + Podejmowanie decyzji technologicznych i architektonicznych
  + Przegląd i kontrola jakości kodu
* Kompetencje:
  + Zaawansowana znajomość C# i ASP.NET Core
  + Doświadczenie w projektowaniu systemów informatycznych
  + Znajomość wzorców projektowych

1. Backend Developer

* Zakres obowiązków:
  + Implementacja logiki biznesowej
  + Integracja z bazą danych
  + Optymalizacja wydajności backendu
  + Tworzenie i utrzymanie dokumentacji technicznej
* Kompetencje:
  + Bardzo dobra znajomość C# i ASP.NET Core
  + Doświadczenie w pracy z bazami danych
  + Umiejętność rozwiązywania problemów technicznych
  + Znajomość narzędzi do zarządzania wersjami

1. Frontend Developer

* Zakres obowiązków:
  + Tworzenie interfejsu użytkownika
  + Implementacja komponentów frontendowych
  + Zapewnienie responsywności i użyteczności aplikacji
  + Współpraca z zespołem UX/UI
* Kompetencje:
  + Znajomość technologii frontendowych (WPF, WinForms)
  + Doświadczenie w tworzeniu aplikacji webowych
  + Umiejętność pracy z narzędziami do kontroli wersji

1. Test Manager

* Zakres obowiązków:
  + Opracowanie strategii testowania
  + Koordynacja prac zespołu testowego
  + Zarządzanie harmonogramem testów
  + Raportowanie wyników testów
* Kompetencje:
  + Doświadczenie w zarządzaniu testami w projektach IT
  + Znajomość narzędzi i technik testowania
  + Umiejętność analizy i raportowania wyników testów
  + Doskonałe umiejętności komunikacyjne

1. Tester

* Zakres obowiązków:
  + Przeprowadzanie testów funkcjonalnych, niefunkcjonalnych, regresyjnych i integracyjnych
  + Tworzenie scenariuszy testowych
  + Zgłaszanie i śledzenie błędów
  + Współpraca z zespołem programistycznym w celu poprawy jakości kodu
* Kompetencje:
  + Doświadczenie w testowaniu aplikacji webowych i desktopowych
  + Znajomość narzędzi do zarządzania testami
  + Umiejętność tworzenia dokumentacji testowej
  + Znajomość podstaw programowania

**Zasady dokumentowania projektu**

|  |  |
| --- | --- |
| Nazwa dokumentu - szablon | Osoba wypełniająca dokument |
| Plan Projektu | Project Manager |
| Harmonogram projektu | Project Manager |
| Specyfikacja wymagań | Analityk biznesowy, Klient – Product Owner |
| Analiza specyfikacja wymagań | Analityk biznesowy |
| Projekt aplikacji | Lead Developer |
| Struktura use case’a | Analityk biznesowy |
| Analiza ryzyka projektu | Project Manager |
| Opis wersji systemu | Klient – Product Owner |
| Formularz test case’u | Tester |
| Formularz błędu | Tester |

1. Harmonogram projektu

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa grupy zadań | Nazwa zadania | Artefakt zadania  (owoce pracy) | Rola w projekcie  realizująca zadanie |
| Zarządzanie projektem | Przygotowanie planu projektu | Plan projektu | Kierownik Projektu |
| Przygotowanie harmonogramu projektu | Harmonogram projektu | Kierownik Projektu |
| Przygotowanie kosztorysu projektu | Kosztorys projektu | Kierownik Projektu, Analityk |
| Napisanie planu testów | Plan testów | Tester |
| Przygotowanie planu zarządzanie ryzykiem | Plan zarządzania ryzykiem | Kierownik Projektu, Specjalista ds. Zarzadzania Ryzykiem |
| Przygotowanie planu zarządzania konfiguracją | Plan zarządzania konfiguracją | Kierownik Projektu, Specjalista ds. Konfiguracji |
| Akceptacja planów | Zatwierdzone plany projektu | Kierownik Projektu |
| Przygotowanie specyfikacji wymagań | Zbieranie wymagań | Lista wymagań | Analityk |
| Dokumentacja wymagań | Specyfikacja wymagań | Analityk |
| Przegląd wymagań z klientem | Zatwierdzona specyfikacja wymagań | Analityk, Klient |
| Analiza specyfikacji wymagań | Analiza techniczna wymagań | Raport z analizy technicznej | Architekt systemu, Analityk |
| Przegląd analizy technicznej | Zatwierdzony raport z analizy technicznej | Architekt systemu, Kierownik Projektu |
| Przygotowanie projektu aplikacji | Projektowanie architektury systemu | Dokumentacja architektury | Architekt systemu |
| Projektowanie interfejsu użytkownika | Specyfikacja UI | Projektant UI |
| Projektowanie bazy danych | Model bazy danych | Inżynier baz danych |
| Przegląd projektu aplikacji | Zatwierdzona dokumentacja projektu | Architekt systemu, Kierownik Projektu |
| Implementacja aplikacji | Implementacja modułów aplikacji | Kod źródłowy | Programista |
| Integracja modułów | Zintegrowania aplikacja | Programista |
| Kodowanie testów jednostkowych | Skrypty testów jednostkowych | Programista, Tester |
| Testy aplikacji | Przygotowanie środowiska testowego | Skonfigurowane środowisko testowe | Tester |
| Testy funkcjonalne | Raport z testów funkcjonalnych | Tester |
| Testy wydajnościowe | Raport z testów wydajnościowych | Tester |
| Testy akceptacyjne | Raport z testów akceptacyjnych, zatwierdzona aplikacja | Tester, Klient |

1. Kosztorys projektu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rola | Średnie godziny pracy dziennie | Liczba dni pracy | Łączny nakład godzinowy | Stawka godzinowa (PLN) | Łączny koszt (PLN) |
| Project Manager (PM) | 6 | 60 | 360 | 150 | 54 000 |
| Product Owner (PO) | 4 | 50 | 200 | 180 | 36 000 |
| Analityk Biznesowy | 7 | 45 | 315 | 130 | 40 950 |
| Lead Developer | 8 | 90 | 720 | 200 | 144 00 |
| Backend Developer | 8 | 85 | 680 | 160 | 108 800 |
| Frontend Developer | 7 | 80 | 560 | 160 | 89 600 |
| Test Manager | 6 | 50 | 300 | 140 | 42 000 |
| Tester | 7 | 60 | 420 | 120 | 50 400 |

1. Zasoby projektu
2. Zasoby sprzętowe

* Komputery dla zespołów programistycznych
* Specyfikacja minimalna:
  + Procesor: AMD Ryzen 5 3500U
  + RAM: 8 GB
  + Dysk SSD: 512 GB
  + Karta graficzna: Zintegrowana lub dedykowana (np. AMD Radeon Vega 8 Graphics)
  + System operacyjny: Windows 10/11

1. Inne zasoby sprzętowe

* Monitory: Dla każdego członka zespołu programistycznego i testowego minimum dwa monitory, rozdzielczość Full HD.
* Urządzenia peryferyjne: Klawiatury, myszy.

1. Licencje oprogramowania

* Środowisko IDE
  + Visual Studio: Licencje dla wszystkich deweloperów (preferowana wersja Enterprise dla zaawansowanych funkcji).
* Systemy operacyjne
  + Windows 11 Home: Dla komputerów programistycznych i testowych.
* Bazy danych
  + Microsoft SQL Server: Licencje na odpowiednią edycję (np. Standard lub Enterprise) dla środowisk deweloperskich, testowych i produkcyjnych.
  + SQL Server Management Studio (SSMS): Narzędzie do zarządzania bazami danych.
* Narzędzia do kontroli wersji
  + Git: Oprogramowanie open-source do kontroli wersji.
  + GitHub/GitLab: Licencje na wersje korporacyjne, jeśli używane są prywatne repozytoria.

1. Standardy i narzędzia w projekcie

**Narzędzia**

1. Środowisko programistyczne (IDE)

* Visual Studio 2022: Główne środowisko IDE używane przez programistów do tworzenia aplikacji w C#.

1. System operacyjny

* Windows 11 Home: System operacyjny dla komputerów programistycznych.

1. Narzędzia do komunikacji

* Microsoft Teams: Platforma do komunikacji i współpracy.
* Discord: Alternatywne narzędzie do komunikacji zespołowej.

1. Procesy zarządzania
   1. Plan zarządzania konfiguracją
   2. Wstęp

Plan zarządzania konfiguracją (Configuration Management Plan) ma na celu zdefiniowanie i ustanowienie procedur zarządzania konfiguracją w projekcie "Sklep z grami komputerowymi". CM jest niezbędnym procesem, który zapewnia kontrolę nad rozwojem, zmianami i integralnością produktu na różnych etapach projektu. Główne cele CM w tym projekcie to:

* Zapewnienie, że wszystkie elementy konfiguracji są odpowiednio identyfikowane, kontrolowane i śledzone.
* Ustanowienie jasnych zasad zarządzania wersjami i zmianami.
* Minimalizowanie ryzyka związanego z utratą kontroli nad komponentami systemu.

Zakres stosowania CM obejmuje wszystkie fazy projektu, od analizy wymagań, przez implementację, aż po testowanie i wdrożenie.

* 1. Organizacja dokumentu

CM odgrywa kluczową rolę w zapewnieniu, że projekt jest realizowany zgodnie z planem, a wszystkie zmiany są odpowiednio dokumentowane i kontrolowane. Proces CM jest ściśle powiązany z innymi procesami, takimi jak:

* Zarządzanie projektem: CM wspiera zarządzanie projektem, dostarczając dane dotyczące statusu wersji oraz zmian, co umożliwia lepsze zarządzanie harmonogramem i zasobami.
* Zarządzanie jakością (QA): CM współpracuje z procesami QA, aby zapewnić, że tylko zweryfikowane i zatwierdzone zmiany są implementowane, co minimalizuje ryzyko wprowadzenia defektów.
* Procesy wytwarzania oprogramowania: CM integruje się z procesami programistycznymi, zapewniając, że wszystkie wersje kodu są odpowiednio zarządzane i kontrolowane.

Plan CM jest również ściśle powiązany z innymi planami projektu, takimi jak plan zarządzania ryzykiem, plan zarządzania testami i plan zapewnienia jakości.

* 1. Harmonogram projektu

Harmonogram projektu "Sklep z grami komputerowymi" jest podzielony na kilka kluczowych faz:

* Faza analizy: Identyfikacja wszystkich elementów konfiguracji (CI), które będą podlegać kontroli.
* Faza implementacji: Ustalanie wersji bazowych dla kodu i dokumentacji, oraz śledzenie wszelkich zmian.
* Faza testowania: Przeprowadzanie audytów konfiguracji, aby upewnić się, że wszystkie CI są zgodne z wymaganiami.
* Faza wdrożenia: Ostateczne ustalenie wersji produkcyjnej (release) i jej wprowadzenie na rynek.

Każda z tych faz będzie miała przypisane punkty kontrolne (milestones) związane z CM, które będą monitorowane i oceniane. Zbiory informacji 'baseline' będą definiowane na początku każdej fazy i aktualizowane w miarę postępu projektu.

* 1. Identyfikacja konfiguracji

Identyfikacja konfiguracji obejmuje określenie wszystkich elementów, które będą zarządzane w ramach CM. W projekcie "Sklep z grami komputerowymi" będą to:

* Kod źródłowy: Każdy moduł i komponent systemu będzie podlegać kontroli wersji.
* Dokumentacja: Wszystkie dokumenty projektowe, specyfikacje, plany testowe i instrukcje użytkownika.
* Środowiska testowe: Wszelkie skrypty i narzędzia używane do testowania oprogramowania.

Struktura identyfikatorów konfiguracji będzie oparta na unikalnych tagach, które będą nadawane każdej wersji komponentu. Zasady nazewnictwa będą obejmować numerację główną (major), podrzędną (minor) oraz numerację poprawek (patch).

* 1. Zarządzanie interfejsami

Zarządzanie interfejsami obejmuje identyfikację i kontrolę wszystkich interfejsów, zarówno wewnętrznych, jak i zewnętrznych, które są istotne dla projektu. Procedury te obejmują:

* Identyfikacja wymagań interfejsowych: Dokumentowanie wszystkich wymaganych interfejsów między modułami systemu oraz zewnętrznymi systemami.
* Zarządzanie umowami interfejsowymi: Ustalanie formalnych porozumień dotyczących interfejsów z partnerami zewnętrznymi.
* Kontrola interfejsów: Regularne przeglądy i audyty interfejsów, aby upewnić się, że są zgodne z wymaganiami i nie wprowadzają ryzyka.
  1. Zasady rejestracji statusów konfiguracji

Rejestracja statusów konfiguracji będzie prowadzona przez cały czas trwania projektu. Zasady te obejmują:

* Częstotliwość rejestracji: Informacje o stanie konfiguracji będą rejestrowane przy każdym ważnym zdarzeniu, takim jak zakończenie fazy projektu lub wprowadzenie kluczowej zmiany.
* Zakres informacji: Rejestracja będzie obejmować informacje o stanie edycji, wersji, historii zmian oraz statusie zatwierdzenia.
* Udostępnianie informacji: Wszystkie informacje będą dostępne w centralnym repozytorium CM, które będzie dostępne dla wszystkich członków zespołu projektowego.

Kontrola konfiguracji będzie prowadzona zgodnie z ustalonymi procedurami, które obejmują:

* Audyt procesów CM: Regularne przeglądy i audyty, aby upewnić się, że procedury CM są zgodne z planem i spełniają wymagania projektu.
* Kontrola wersji: Ustalanie, która wersja komponentu jest wersją bazową, a która wersją roboczą. Kontrola dostępu do wersji roboczych i bazowych.
* Zarządzanie zmianami: Każda zmiana w projekcie będzie podlegała kontroli i zatwierdzeniu przed wdrożeniem.  
  1. Zarządzanie podwykonawcami

Podwykonawcy odgrywają istotną rolę w projekcie "Sklep z grami komputerowymi", dlatego konieczne jest ustanowienie jasnych zasad zarządzania nimi:

* Wymagania dotyczące CM: Podwykonawcy będą zobowiązani do stosowania tych samych procedur CM co główny zespół projektowy.
* Kontrola zgodności: Regularne przeglądy i audyty dokumentacji i procesów CM podwykonawców w celu zapewnienia zgodności z wymaganiami projektu.
* Raportowanie: Podwykonawcy będą zobowiązani do regularnego raportowania statusu konfiguracji oraz wszelkich zmian wprowadzanych w ramach swoich działań.
  1. Plan zarządzania ryzykiem

Plan zarządzania ryzykiem definiuje procesy identyfikacji, oceny, monitorowania i zarządzania ryzykami, które mogą mieć wpływ na projekt. Główne elementy tego procesu to:

* **Identyfikacja ryzyk:** Rozpoznanie i opisanie potencjalnych zagrożeń dla projektu.
* **Ocena ryzyk:** Analiza prawdopodobieństwa wystąpienia oraz wpływu ryzyka na projekt.
* **Planowanie reakcji na ryzyko:** Określenie strategii działania w przypadku wystąpienia ryzyka, np. unikanie, łagodzenie, przeniesienie lub akceptacja ryzyka.
* **Monitorowanie ryzyk:** Regularne przeglądy ryzyk i aktualizacja planów działania w odpowiedzi na zmieniające się warunki projektowe.
  1. Plan zarządzania testami

1. Opis terminologii i skrótów używanych w procesie testowania.

W niniejszym planie zarządzania testami zostaną wykorzystane różne terminy i skróty, które są istotne dla zrozumienia i realizacji procesu testowania w projekcie "Sklep z grami komputerowymi".

* TC (Test Case): Przypadek testowy, czyli specyfikacja warunków wejściowych, działań oraz oczekiwanych wyników.
* TDD (Test-Driven Development): Metoda tworzenia oprogramowania, w której najpierw pisane są testy, a potem kod, który te testy przechodzi.
* QA (Quality Assurance): Proces zapewniania jakości, który koncentruje się na poprawności i integralności systemu.
* Defect/Bug: Błąd lub usterka w oprogramowaniu, która powoduje jego niewłaściwe działanie.

2. Spis dokumentów do których odwołuje się plan testów.

Plan testów odwołuje się do następujących dokumentów, które dostarczają dodatkowych informacji i wytycznych dla procesu testowania:

* Specyfikacja wymagań: Dokument opisujący wymagania funkcjonalne i niefunkcjonalne systemu.
* Projekt systemu: Dokument zawierający szczegółowy opis architektury i modułów systemu.
* Plan zarządzania ryzykiem: Dokument identyfikujący potencjalne ryzyka związane z projektem i metody ich minimalizacji.
* Instrukcja użytkownika: Dokument opisujący sposób korzystania z systemu przez użytkowników końcowych.

3. Omówienie testowanego systemu.

Systemem podlegającym testom jest platforma "Sklep z grami komputerowymi", która umożliwia użytkownikom przeglądanie, kupowanie oraz pobieranie gier komputerowych. System składa się z następujących głównych modułów:

* Frontend: Interfejs użytkownika dostępny przez przeglądarkę internetową.
* Backend: Serwer obsługujący logikę biznesową, zarządzanie bazą danych, oraz integrację z zewnętrznymi systemami płatności.
* Baza danych: System zarządzający danymi dotyczącymi użytkowników, gier, transakcji oraz historii zakupów.
* API: Interfejsy programistyczne umożliwiające integrację z zewnętrznymi systemami i usługami.

4. Opis strategii testowania.

Strategia testowania w projekcie obejmuje następujące kluczowe podejścia:

* Testowanie jednostkowe (Unit Testing): Każdy moduł systemu będzie testowany niezależnie, aby upewnić się, że działa zgodnie z oczekiwaniami.
* Testowanie integracyjne (Integration Testing): Testy skupiają się na interakcjach między różnymi modułami systemu, aby zapewnić, że są one ze sobą zgodne.
* Testowanie systemowe (System Testing): Kompletny system będzie testowany w celu weryfikacji, czy spełnia wymagania określone w specyfikacji.
* Testowanie akceptacyjne (Acceptance Testing): Testy przeprowadzane w celu potwierdzenia, że system spełnia potrzeby użytkowników końcowych.

5. Opis metodologii testowania.

Metodologia testowania obejmuje cykliczne i iteracyjne podejście do testowania, z naciskiem na ciągłe doskonalenie i adaptację do zmian w projekcie. Wykorzystane zostaną następujące metody:

* Testy ręczne: Wykonywane przez testerów w celu sprawdzenia funkcji, które wymagają interakcji z użytkownikiem.
* Testy automatyczne: Skrypty automatyzujące testy regresji, które będą wykonywane po każdej większej zmianie w kodzie.
* Testy regresji: Zapewnienie, że nowe zmiany nie wprowadzają błędów w już przetestowane i działające funkcjonalności.

6. Analiza zasobów potrzebnych do przeprowadzenia testów.

Zasoby potrzebne do przeprowadzenia testów obejmują:

* Zespół testerów: W skład którego wchodzą testerzy manualni oraz specjaliści od automatyzacji testów.
* Środowisko testowe: Serwery, bazy danych oraz infrastruktura sieciowa niezbędna do przeprowadzania testów.
* Narzędzia testowe: Oprogramowanie do zarządzania testami, takie jak JIRA dla zarządzania błędami, Selenium dla testów automatycznych, oraz narzędzia CI/CD, takie jak Jenkins.

7. Identyfikacja testów - przygotowanie listy testów (test case'ów).

Każda funkcjonalność systemu będzie miała przypisane Test Case'y, które będą obejmować:

* TC001: Rejestracja nowego użytkownika.
* TC002: Logowanie użytkownika.
* TC003: Przeglądanie katalogu gier.
* TC004: Dodanie gry do koszyka.
* TC005: Zrealizowanie zakupu gry.
* TC006: Pobranie zakupionej gry.

8. Określenie parametrów które będą monitorowane podczas testów.

Podczas testów będą monitorowane następujące parametry:

* Czas odpowiedzi systemu: Szybkość, z jaką system reaguje na zapytania użytkowników.
* Użycie zasobów: Monitorowanie użycia procesora, pamięci, oraz zasobów sieciowych podczas testów obciążeniowych.
* Liczba wykrytych błędów: Statystyki dotyczące liczby błędów znalezionych i naprawionych podczas testów.

9. Określenie kryteriów pozytywnego zakończenia każdego testu.

Każdy test będzie uznany za zakończony pozytywnie, jeśli:

* Spełnia wszystkie wymagania funkcjonalne: Testowana funkcja działa zgodnie z opisem w specyfikacji wymagań.
* Brak krytycznych błędów: Test nie wykrywa błędów, które uniemożliwiają normalne funkcjonowanie systemu.
* Czas odpowiedzi jest akceptowalny: System spełnia założenia wydajnościowe określone w specyfikacji.

10. Określenie kryteriów akceptujących system.

System zostanie zaakceptowany, jeśli:

* Wszystkie przypadki testowe są zakończone pozytywnie: Wszystkie zdefiniowane testy zostały wykonane i zakończone sukcesem.
* Brak otwartych błędów krytycznych: Wszystkie krytyczne błędy zostały rozwiązane lub zidentyfikowane jako nieistotne dla funkcjonowania systemu.
* System działa stabilnie: System przeszedł testy obciążeniowe i stabilnościowe bez awarii.

11. Opis metodologii zarządzania wykrytymi błędami.

Zarządzanie błędami będzie realizowane w następujący sposób:

* Rejestracja błędów: Wszystkie błędy będą rejestrowane w narzędziu do zarządzania błędami (np. JIRA) z pełnym opisem, krokiem do odtworzenia oraz priorytetem.
* Klasyfikacja błędów: Błędy będą klasyfikowane według priorytetu (krytyczne, wysokie, średnie, niskie) oraz przypisywane do odpowiednich deweloperów.
* Monitorowanie postępu: Postęp naprawy błędów będzie monitorowany na bieżąco, a statusy będą regularnie aktualizowane.
* Retesty: Po naprawieniu błędu, test zostanie powtórzony w celu weryfikacji poprawki.

12. Opis metod analizy i raportowania wyników.

Wyniki testów będą analizowane i raportowane w następujący sposób:

* Dzienniki testów: Wszystkie wyniki testów będą zapisywane w dziennikach testów, które będą zawierać szczegółowe informacje o każdym teście, w tym o wynikach, czasie trwania oraz ewentualnych problemach.
* Raporty z testów: Regularne raporty będą generowane, podsumowujące stan testów, wykryte błędy, oraz stopień zaawansowania procesu testowania.
* Analiza regresji: Analiza wyników testów regresji będzie prowadzona, aby upewnić się, że nowe funkcjonalności nie wprowadzają błędów do wcześniej działających części systemu.

13. Opis środowiska testowego (konfiguracja sprzętu, oprogramowanie).

Środowisko testowe będzie obejmować następujące elementy:

* Sprzęt: Serwery testowe z odpowiednią specyfikacją techniczną, zbliżoną do środowiska produkcyjnego.
* Oprogramowanie: Wersje systemu operacyjnego, baz danych, oraz innych zależnych komponentów oprogramowania, które będą zbliżone do wersji produkcyjnych.
* Konfiguracja: Konfiguracja sieci, firewalla, oraz innych komponentów infrastrukturalnych będzie zbliżona do środowiska produkcyjnego.

14. Harmonogram testów.

Harmonogram testów będzie obejmować następujące etapy:

* Planowanie testów: Przygotowanie planu testów oraz identyfikacja wszystkich przypadków testowych.
* Przygotowanie środowiska: Ustawienie i skonfigurowanie środowiska testowego.
* Przeprowadzenie testów jednostkowych: Testy jednostkowe każdego komponentu systemu.
* Testy integracyjne: Testowanie integracji pomiędzy komponentami.
* Testy systemowe: Przeprowadzenie testów systemowych w pełnym środowisku.
* Testy akceptacyjne: Testy końcowe, które potwierdzą, że system jest gotowy do wdrożenia.
* Raportowanie: Analiza wyników testów i przygotowanie końcowego raportu.
  1. Plan zapewnienia jakości

1. Cel dokumentu

Celem niniejszego dokumentu jest przedstawienie planu zapewnienia jakości (QA) dla projektu "Sklep z grami komputerowymi". Plan ten ma na celu zapewnienie, że wszystkie aspekty projektu są realizowane zgodnie z ustalonymi standardami jakości, a produkt końcowy spełnia wymagania funkcjonalne i niefunkcjonalne oraz oczekiwania klientów.

1. Zakres dokumentu

Plan QA obejmuje wszystkie procesy związane z zarządzaniem jakością w projekcie. Obejmuje to monitorowanie, kontrolowanie i audytowanie działań związanych z wytwarzaniem oprogramowania oraz zapewnienie, że produkt końcowy spełnia ustalone standardy i normy jakościowe.

1. Charakterystyka projektu i budowanego systemu

Projekt "Sklep z grami komputerowymi" ma na celu stworzenie platformy internetowej umożliwiającej użytkownikom przeglądanie, zakup i pobieranie gier komputerowych. System składa się z kilku modułów, w tym frontend, backend, baza danych oraz API integrujące się z zewnętrznymi systemami płatności. Główne cele projektu to zapewnienie użytkownikom intuicyjnego interfejsu, szybkości działania oraz bezpieczeństwa transakcji.

1. Identyfikacja obszaru QA

W procesie zapewnienia jakości będą podlegać następujące elementy systemu (Computer Software Configuration Items - CSCI):

* Frontend: Interfejs użytkownika, który będzie testowany pod kątem użyteczności, wydajności oraz zgodności z wymaganiami.
* Backend: Moduł zarządzający logiką biznesową, który będzie oceniany pod kątem stabilności, wydajności oraz zgodności z wymaganiami funkcjonalnymi.
* Baza danych: Struktura danych oraz mechanizmy zarządzania danymi, które będą monitorowane pod kątem integralności, wydajności oraz bezpieczeństwa.
* API: Interfejsy programistyczne, które będą testowane pod kątem zgodności z dokumentacją, wydajności oraz bezpieczeństwa.

1. Standardy i procedury QA

Proces zapewnienia jakości w projekcie "Sklep z grami komputerowymi" będzie oparty na następujących standardach i procedurach:

* ISO/IEC 25010: Standard jakości oprogramowania, który definiuje atrybuty jakości, takie jak funkcjonalność, wydajność, użyteczność, niezawodność, bezpieczeństwo i zgodność.
* ISTQB: Procedury związane z testowaniem oprogramowania, w tym definicje poziomów testów, metodologii testowania oraz technik projektowania testów.
* Agile Testing: Procedury testowania w środowisku Agile, gdzie QA jest integralną częścią procesu deweloperskiego i odbywa się w sposób iteracyjny.

1. Relacja do innych planow projektu

Plan zapewnienia jakości jest ściśle powiązany z innymi planami projektu, w tym:

* Planem zarządzania konfiguracją: QA będzie weryfikować, czy wszystkie zmiany wprowadzane do systemu są zgodne z ustalonymi procedurami zarządzania konfiguracją.
* Planem zarządzania testami: QA będzie monitorować i oceniać procesy testowania, aby zapewnić, że są one przeprowadzane zgodnie z wymaganiami jakościowymi.
* Planem zarządzania ryzykiem: QA będzie uczestniczyć w identyfikacji i minimalizacji ryzyk związanych z jakością oprogramowania.

1. Referencje

Do realizacji planu QA odnosić się będą następujące dokumenty:

* Specyfikacja wymagań funkcjonalnych i niefunkcjonalnych.
* Dokumentacja projektowa systemu.
* Plan zarządzania ryzykiem.
* Instrukcja użytkownika.

1. QA w strukturze organizacyjnej projektu

W strukturze organizacyjnej projektu, zespół QA pełni kluczową rolę w monitorowaniu i kontrolowaniu procesów związanych z jakością oprogramowania. Zespół QA będzie współpracował z innymi działami, w tym zespołem deweloperskim, menedżerami projektu oraz testerami, aby zapewnić, że wszystkie działania są zgodne z ustalonymi standardami jakości.

1. Zasoby QA

Do realizacji zadań QA w projekcie "Sklep z grami komputerowymi" będą wymagane następujące zasoby:

* Zespół QA: Składający się z inżynierów jakości, którzy będą odpowiedzialni za przeprowadzanie audytów jakościowych, testowanie oraz monitorowanie zgodności z procedurami.
* Narzędzia QA: Oprogramowanie do zarządzania jakością, takie jak JIRA dla zarządzania błędami, narzędzia do automatyzacji testów oraz systemy raportowania.

1. Zadania QA

Zespół QA będzie odpowiedzialny za następujące zadania:

* Monitorowanie procesów deweloperskich: Zapewnienie, że wszystkie etapy procesu tworzenia oprogramowania są zgodne z ustalonymi procedurami jakościowymi.
* Przeprowadzanie testów: QA będzie odpowiedzialne za organizowanie i przeprowadzanie testów jakościowych, w tym testów funkcjonalnych, wydajnościowych oraz bezpieczeństwa.
* Audytowanie: Regularne przeprowadzanie audytów, aby upewnić się, że wszystkie procesy projektowe spełniają ustalone standardy jakości.
* Raportowanie: Dokumentowanie wyników testów oraz przedstawianie raportów jakościowych menedżerom projektu.

1. Zakres odpowiedzialności za procesy QA

Zakres odpowiedzialności za QA w projekcie obejmuje:

* Zespół QA: Odpowiedzialny za przeprowadzanie testów i audytów oraz monitorowanie zgodności z procedurami.
* Menedżer projektu: Odpowiedzialny za nadzór nad realizacją planu QA oraz podejmowanie decyzji w oparciu o raporty jakościowe.
* Zespół deweloperski: Odpowiedzialny za dostosowanie się do zaleceń QA oraz implementację wymaganych poprawek.

1. Harmonogram

Harmonogram działań QA w projekcie obejmuje:

* Planowanie: Ustalenie standardów i procedur QA na początku projektu.
* Realizacja: Regularne przeprowadzanie testów i audytów jakościowych podczas trwania projektu.
* Końcowe audyty: Ostateczna weryfikacja zgodności systemu z wymaganiami jakościowymi przed jego wdrożeniem.

1. Dokumentacja

W ramach QA w projekcie "Sklep z grami komputerowymi" będą tworzone następujące dokumenty:

* Raporty z testów: Zawierające wyniki testów oraz zalecenia dotyczące poprawek.
* Raporty z audytów: Dokumentujące zgodność procesów z ustalonymi standardami jakościowymi.
* Dokumentacja błędów: Rejestrująca wszystkie znalezione błędy oraz status ich naprawy.

1. Standardy i procedury

Obowiązujące standardy w procesie QA to:

* Standardy kodowania: Określenie konwencji nazewnictwa, formatowania kodu oraz struktury projektu.
* Standardy testowania: Wykorzystanie standardów ISTQB do projektowania, przeprowadzania oraz raportowania testów.
* Standardy dokumentacji: Stosowanie jednolitych standardów dla całej dokumentacji projektowej.

Kontrola stosowania standardów będzie realizowana poprzez:

* Przeglądy kodu: Regularne przeglądy kodu przez zespół QA oraz zespół deweloperski.
* Automatyczne sprawdzanie zgodności: Użycie narzędzi do automatycznego sprawdzania zgodności kodu ze standardami.

1. Testy

Proces QA obejmuje następujące rodzaje testów:

* Testy funkcjonalne: Sprawdzające, czy system spełnia wszystkie zdefiniowane wymagania funkcjonalne.
* Testy wydajnościowe: Ocena, czy system działa efektywnie pod różnymi obciążeniami.
* Testy bezpieczeństwa: Weryfikacja, czy system jest odporny na ataki i czy spełnia wymagania dotyczące ochrony danych.
* Testy regresyjne: Zapewnienie, że nowe funkcje nie wprowadziły błędów do wcześniej działających elementów systemu.

1. Zasady rejestracji i raportowania o błędach

Błędy będą rejestrowane i raportowane w następujący sposób:

* Rejestracja błędów: Każdy znaleziony błąd będzie rejestrowany w systemie zarządzania błędami (np. JIRA), z dokładnym opisem oraz przypisanym priorytetem.
* Raportowanie: Regularne raporty będą generowane, aby informować zespół o stanie błędów oraz postępach w ich naprawianiu.

1. Zarządzanie ryzykiem

Zarządzanie ryzykiem w QA będzie obejmować:

* Identyfikacja ryzyk: Każdy etap procesu wytwarzania oprogramowania będzie analizowany pod kątem potencjalnych ryzyk związanych z jakością.
* Formularze kontrolne: Dla każdego etapu procesu będą tworzone formularze kontrolne, które będą monitorować realizację założonych standardów oraz identyfikować potencjalne ryzyka.
* Minimalizacja ryzyka: QA będzie współpracować z zespołem deweloperskim oraz menedżerem projektu, aby wdrażać działania minimalizujące zidentyfikowane ryzyka.
  1. Plan zarządzania zmianami

Plan zarządzania zmianami definiuje procedury, które są stosowane w celu kontrolowania i zatwierdzania zmian w projekcie. Procesy zarządzania zmianami obejmują:

* **Identyfikacja zmian:** Zgłaszanie i dokumentowanie proponowanych zmian w projekcie.
* **Ocena zmian:** Analiza wpływu zmiany na zakres, budżet, harmonogram i jakość projektu.
* **Zatwierdzanie zmian:** Przegląd i zatwierdzenie lub odrzucenie zmian przez komitet sterujący lub inne odpowiedzialne osoby.
* **Wdrażanie zmian:** Implementacja zatwierdzonych zmian oraz aktualizacja dokumentacji projektowej.
* **Monitorowanie zmian:** Śledzenie wdrożonych zmian i ocena ich wpływu na projekt.