1. Wprowadzenie

[We wprowadzeniu należy zawrzeć informacje, które mogą być potrzebne osobie czytającej niniejszy dokument we właściwym zrozumieniu go.]

1.1 Cel

[Cel powstania tego dokumentu. Celem tworzenia dokumentu „Specyfikacja wymagań funkcjonalnych i niefunkcjonalnych" jest szczegółowe przedstawienie funkcjonalności projektowanej aplikacji oraz wszystkich ograniczeń projektowych i funkcjonalnych. Wymagania funkcjonalne zostaną pokazane za pomocą modelu przypadków użycia i w sposób opisowy.]

<cel>

1.2 Zakres

[Krótki opis, jakiego projektu (projektów) dotyczy dokument.]

<zakres>

1.3 Definicje, akronimy i skróty

[Definicje wszystkich terminów, akronimów i skrótów niezbędnych do zrozumienia dokumentu. Oprócz tego lub zamiast tego można w tym miejscu umieścić odesłanie do dokumentu słownika utworzonego dla projektu, którego dotyczy także ten dokument.]

<definicje i/lub odwołanie do dokumentu słownika>

1.4 Referencje

[Kompletna lista dokumentów, na które powołuje się niniejszy dokument, lub dokumentów, które są potrzebne do właściwego zrozumienia niniejszego dokumentu. Dla każdej pozycji należy określić autora, tytuł, datę powstania (wydania), wydawnictwo (o ile jest to możliwe). Należy także określić źródło, gdzie można uzyskać dostęp do poszczególnych pozycji.]

<referencje>

1.5 Streszczenie

[Krótki opis, co zawiera pozostała część dokumentu. Wyjaśnienie, w jaki sposób dokument jest zorganizowany.]

<opis>

2. Opis ogólny

[Ten rozdział dokumentu opisuje główne czynniki wpływające na tworzony system i wymagania wobec nich stawiane. Nie specyfikuje on jednak dokładnie wymagań. Dostarcza jedynie podstawy, definiuje środowisko systemu, dla którego wymagania zostaną zdefiniowane w rozdziale 3. Określenie podstaw pozwoli na łatwiejsze zrozumienie właściwych wymagań systemu. W rozdziale tym mogą wystąpić poniższe podpunkty. W niektórych przypadkach pewne podpunkty mogą zostać pominięte, w innych należy umieścić dodatkowe podpunkty.]

2.1 Perspektywa produktu

2.1.1 Interfejsy systemowe

[Podział systemu na elementy składowe (komponenty) i przedstawienie które elementy i w jaki sposób powinny ze sobą współpracować]

<opis>

2.1.1 Interfejsy użytkownika

[W jakiej postaci poszczególne komponenty składowe dostarczają interfejs użytkownikowi (przeglądarka internetowa, command-line, xml, windows GUI, brak interfejsu użytkownika, ...)]

<opis>

2.1.1 Interfejsy sprzętowe

[Z jakimi urządzeniami system i jego poszczególne elementy będzie współpracował (komputer PC, specjalny serwer, ...)]

<opis>

2.1.1 Interfejsy programowe

[W jakiej postaci, technologii mają być zrealizowane interfejsy systemu (java applet, jsp, web server, jdbc, mfc, ...)]

<opis>

2.1.1 Interfejsy komunikacyjne

[Jak poszczególne elementy systemu mają się ze sobą komunikować (protokół TCP/IP, SOAP, muszą znajdować się na tej samej fizycznej maszynie, ...)]

<opis>

2.1.1 Ograniczenia pamięciowe

[Ile pamięci aplikacja będzie potrzebowała, aby bezbłędnie pracować]

<opis>

2.2 Funkcje produktu

<opis>

2.3 Charakterystyka użytkowników

<opis>

2.4 Ograniczenia

<opis>

3. Charakterystyka wymagań

[Ten rozdział zawiera wszystkie wymagania programowe na poziomie szczegółowości wystarczający projektantom systemu do stworzenie modelu spełniającego wyspecyfikowane tu wymagania, a testerom do przeprowadzenia testów sprawdzających, czy system spełnia te wymagania. Określane słownie wymagania są odwzorowane przez przypadki użycia i odpowiadające im specyfikacje dodatkowe.]

3.1 Funkcjonalność

[Ten podrozdział opisuje wymagania funkcjonalne systemu, które wyrażone w języku naturalnym. Organizacja, uporządkowanie wymagań może być dowolna. Można to zrobić dzieląc wymagania na grupy ze względu na użytkowników lub podsystemy.]

3.1.1 <Wymaganie funkcjonalne 1>

<opis wymagania>

3.1.2 <Wymaganie funkcjonalne 2>

<opis wymagania>

3.2 Specyfikacja przypadków użycia

[W modelowaniu za pomocą przypadków użycia przypadki użycia często określają główne wymagania funkcjonalne wraz z niektórymi wymaganiami niefunkcjonalnymi.]

3.2.1 Diagramy przypadków użycia

[Diagramy przypadków użycia należy umieścić w tym miejscu. Dla lepszej czytelności może to być kilka diagramów, każdy obejmujący wydzielony pakiet funkcjonalny.]

<diagram lub diagramy przypadków użycia>

3.2.2 Opis aktorów

[Lista aktorów występujących w systemie wraz z opisem każdego z nich (rola odgrywana).]

<lista aktorów z opisem>

3.2.3 Opis przypadków użycia

3.2.3.1 <Nazwa przypadku użycia 1>

- **opis**

  [Opis prezentujący krótko rolę i cel przypadku użycia.]

- **podstawowy ciąg zdarzeń**

  [Przypadek użycia rozpoczyna się w momencie, gdy aktor coś zrobi. Zawsze aktor inicjuje przypadek użycia. Przypadek użycia opisuje co aktor robi i co system robi w odpowiedzi. Opis ciągu zdarzeń przedstawia dialog pomiędzy aktorem i systemem.
  Przypadek użycia opisuje co dzieje się wewnątrz systemu. Nie określa jednak jak ani dlaczego tak się dzieje. Jeżeli zachodzi wymiana informacji, należy określić jakie są to informacje. Na przykład, powiedzenie, że aktor wprowadza dane klienta niewiele wyjaśnia. Lepiej jest powiedzieć, że aktor wprowadza nazwisko i adres klienta. Użyteczne jest zdefiniowanie np. w dokumencie Słownika terminów takich jak "dane klienta". Pozwoli to na uniknięcie konieczności umieszczania w przypadku użycia tych szczegółów.
  Proste alternatywy mogą być zaprezentowane wewnątrz opisu podstawowego ciągu zdarzeń, jeżeli ich opis składa się tylko z kilku zdań. Jeżeli alternatywny ciąg zdarzeń jest bardziej skomplikowany należy opisać go w oddzielnym podpunkcie.
  Rysunek czasem zastępuje tysiąc słów. Jeżeli bardziej zrozumiałe będzie przedstawienie ciągu zdarzeń zachodzących w przypadku użycia na jakimś wykresie lub rysunku można z tego skorzystać. Należy jednak pamiętać, żeby symbole użyte na diagramie były zrozumiałe dla osób, które będą w przyszłości korzystały z tego dokumentu (można użyć np. diagramu czynności UML).]

- **alternatywne ciągi zdarzeń**

  [Bardziej skomplikowane alternatywne ciągi zdarzeń należy opisać w tym podpunkcie. Każdy alternatywny ciąg zdarzeń opisuje zachowanie odbiegające od podstawowego ciągu zdarzeń.]

- **warunki wstępne**

  [Stan systemu jaki musi być spełniony, aby przypadek użycia mógł się wykonać.]

- **warunki końcowe**

  [Lista możliwych stanów systemu bezpośrednio po zakończeniu przypadku użycia.]

- **punkty rozszerzające**

  [Lista przypadków użycia rozszerzających (stereotyp <<extend>>) lub zawierających się (stereotyp <<include>>) w tym przypadku. Krótki opis kiedy nastąpi skorzystanie z wymienionego przypadku (punkt rozszerzający).]

3.2.3.2 <Nazwa przypadku użycia 2>

3.2.3.3 <Nazwa przypadku użycia 3>

3.2.3.4

3.3 Dodatkowe wymagania

[Dodatkowe wymagania określają wymagania, których nie sprecyzowano w przypadkach użycia i wymaganiach

funkcjonalnych. Na wymagania te składają się wymagania prawne i regulacyjne, wymagania określające standardy aplikacji, parametry jakościowe budowanego systemu, użytkowalność, niezawodność, wydajność, wspieralność, inne wymagania, takie jak system operacyjny i środowisko, kompatybilność, ograniczenia projektowe.]

3.3.1 Założenia i zależności

[Opis wszystkich kluczowych technicznych założeń, m.in. dostępne podsystemy lub komponenty lub inne projekty, na których bazuje oprogramowanie opisywane w niniejszym dokumencie.]

3.3.2 Użytkowalność

[określenie wymaganego czasu treningu, w celu osiągnięcia przez użytkownika (zwykłego, zaawansowanego) umiejętności posługiwania się poszczególnymi funkcjami oferowanymi przez system,określenie czasu wykonania typowego zadania przez system]

3.3.3 Niezawodność

[Sugerowane wymagania dotyczące niezawodności są następujące:
- dostępność – określenie procentowo czasu dostępności systemu (xx.xx%), godzin użytkowania, bezawaryjnego dostępu, itp.,
- średni czas pomiędzy awariami (ang. Mean Time Between Failures (MTBF)) – zwykle wyrażony w godzinach, ewentualnie dniach, miesiącach lub latach,
- średni czas naprawy (ang. Mean Time To Repair (MTTR)) – jak długo trwa przywracanie systemu do poprawnej pracy po wystąpieniu awarii,
- dokładność – określa precyzję i dokładność (według standardów), która jest wymagana na wyjściu systemu,
- maksymalna liczba błędów – zwykle wyrażona jako liczba_błędów/KLOC (ang. thousands of lines of code),
- błędy – podzielone na drobne, znaczne i krytyczne, określenie skutków wystąpienia defektów krytycznych, np. całkowita utrata danych, niemożność korzystania z części funkcji systemu.]

3.3.4 Wydajność

[Do wymagań tego rodzaju można zaliczyć:
- czasy odpowiedzi systemu na polecone transakcje (średnie, maksymalne),
- przepustowość (np. liczba transakcji na sekundę),
- wydajność właściwa (np. liczba klientów lub transakcji, które system jest w stanie jednocześnie obsłużyć),
- używane zasoby: pamięć, miejsce na dysku, komunikacja, itp.]

3.3.5 Wspieralność

[Tu należy wymienić wszystkie wymagania pokazujące oparcie systemu o obowiązujące standardy lub wspierające budowę systemu. Na wymagania te składają się standardy kodowania (pisania kodu programu), konwencje nazewnictwa, biblioteki klas, narzędzia wspierające.]

3.3.6 Bezpieczeństwo

[Zidentyfikowanie danych, które muszą być chronione i określenie sposobów ich ochrony i udostępniania (fizyczne zabezpieczenia, ludzie, którzy mogą stanowić zagrożenie, specjalne wymagania odnośnie zabezpieczeń uwzględniające: dostęp do systemu, szyfrowanie danych, autoryzowanie). Na koniec lista obiektów, które wymagają ochrony poprzez logiczne lub fizyczne zabezpieczenia.]

4. Klasyfikacja wymagań funkcjonalnych

[Lista w formie tabeli, wszystkich wymagań funkcjonalnych uporządkowana według rodzajów wymagań (Istotne, Pożądane, Opcjonalne) lub kolejności występującej wcześniej w niniejszym dokumencie.]

Funkcjonalność	Rodzaj
...	

...	