1. Wprowadzenie

1.1 Cel

Dokument planu testów aplikacji *<nazwa projektu>* pomaga w:

- *identyfikacji istniejących informacji o projekcie i komponentów oprogramowania, które powinny zostać sprawdzone,*
- *wylicza rekomendowane wymagania dla testu (na wysokim poziomie),*
- *poleca i opisuje strategie testowania, które należy użyć,*
- *identyfikuje potrzebne narzędzia i określa czas testów,*
- *wylicza dokumenty powstałe w wyniku testu.*

1.2 Podstawowe informacje

[Należy krótko opisać to, co będzie przedmiotem testu (komponent(y), aplikacja, system, itd.) i jego cele. Podpunkt powinien zawierać informacje o głównych funkcjach/możliwościach, architekturze i krótką historię projektu. Powinien być wielkości 3 – 5 akapitów.]

1.3 Zakres

[Opisuje etapy testowanie, np. Modułu, Integracji lub całego Systemu oraz typy testowania, które będę umieszczone w tym planie, takie jak Funkcjonalne lub Wydajności]

[Dostarcza krótkiej listy właściwości przedmiotu testu, funkcji, które będą testowane oraz które nie będą testowane.]

[Tworzy listę założeń stworzonych podczas tworzenia tego dokumentu, które mogą mieć wpływ na projektowanie, rozwój i implementację testowania.]

[Tworzy listę zagrożeń i ewentualnych zdarzeń, które mogą mieć wpływ na projektowanie, rozwój i implementację testowania.]

[Tworzy listę ograniczeń, które mogą mieć wpływ na projektowanie, rozwój i implementację testowania.]

1.4 Rozpoznanie projektu

Tabele poniżej przedstawia dokumentację (i jej dostępność), która jest użyteczna przy tworzeniu planu testu.

[UWAGA: należy usunąć lub dodać punkty, które uważa się za stosowne.]

Dokument (oraz wersja /data)	***Stworzony lub dostępny***	***Dostarczony lub przejrzany***	***Autor lub źródło***	***Uwagi***
Specyfikacja wymagań	°Tak °Nie	°Tak °Nie		
Raport przypadków użycia	°Tak °Nie	°Tak °Nie		
Specyfikacja projektowania	°Tak °Nie	°Tak °Nie		
Prototyp	°Tak °Nie	°Tak °Nie		
Instrukcja obsługi	°Tak °Nie	°Tak °Nie		
Plan projektu	°Tak °Nie	°Tak °Nie		

© Politechnika Białostocka, 2004

2. Wymagania stawiane dla testu

Poniżej zostały wypunktowane te pozycje (przypadki użycia, wymagania funkcjonalne i niefunkcjonalne), które zidentyfikowane zostały jako cele testu. Lista przedstawia, co będzie testowane.

[Należy wpisać wymagania stawiane dla testu na wysokim poziomie.]

3. Strategie testowania

[Strategie testowania prezentują rekomendowane przybliżenia testowania danego czynnika. Poprzedni rozdział, Wymagania stawiane dla testu, opisywał co będzie testowane, ten opisuje jak testować.

Dla każdego rodzaju dostarcza opisu testu i wyjaśnia dlaczego jest implementowany i wykonywany.

Jeżeli dany test nie zostanie zaimplementowany i wykonany, należy to zaznaczyć zdaniem, że nie test nie będzie zaimplementowany/wykonany lub że ten test nie jest właściwy.

Rozważane są głównie używane techniki oraz kryteria kompletności testu.

Dodatkowo należy pamiętać, że testy powinny być przeprowadzone używając znanych, sprawdzonych baz danych w bezpiecznym środowisku.]

3.1.1. Testowanie funkcjonalności

[Testowanie funkcjonalności powinno skupić się na każdym wymaganiu stawianym dla testu, które ma swoje odzwierciedlenie w przypadkach użycia (lub zadanych funkcjach programu) i regułach działania. Celem tego testu jest sprawdzenie właściwej tolerancji danych, przetwarzania, odszukiwania oraz właściwej implementacji zarządzania nimi. Testy te bazują na technice czarnej skrzynki, to jest sprawdzenia aplikacji (i jej wewnętrznych procesów) poprzez interakcję z aplikacją używając GUI i analizując wyjście (wyniki).

Poniżej znajduje się zarys zalecanego każdej aplikacji.]

Cel testu:	Zapewnia właściwą funkcjonalność przedmiotu testu, włączając nawigację, wejście danych, przetwarzanie oraz wyszukiwanie.
Technika:	Wykonywanie każdego przypadek użycia, ciągu przypadków lub funkcji używając właściwych lub niewłaściwych danych, aby zweryfikować: • oczekiwany rezultat pojawia się, gdy podano właściwe dane, • stosowna informacja pojawia się, gdy podano dane niewłaściwe, • każda reguła działania jest właściwie zastosowana.
Kryteria kompletności:	• Wszystkie zaplanowane testy zostały wykonane • Zostały odnalezione przyczyny wszystkich zidentyfikowanych defektów.
Uwagi specjalne:	*[Identyfikuje / opisuje te punkty i kwestie (wewnętrzne lub zewnętrzne), które mają wpływ na implementację oraz wykonanie testów funkcjonalności.]*

3.1.2. Testowanie interfejsu użytkownika

[Testowanie interfejsu użytkownika sprawdza interakcję użytkownika z oprogramowaniem. Celem jest zapewnienie tego by interfejs użytkownika dostarczał użytkownikowi właściwy dostęp oraz nawigację po funkcjach programu. Dodatkowo zapewniają by obiekty interfejsu użytkownika działały tak jak oczekiwano oraz są dostosowane do standardów korporacji lub przedsiębiorstwa.]

Cel testu:	Sprawdza czy: • nawigacja po przedmiocie testu właściwie odzwierciedla zadane funkcje, wymagania, włączając obecność okienek, pól oraz metod dostępu, • obiekty okna oraz jego właściwości takie jak menu, wielkość, pozycja, stan oraz czytelność odpowiadają standardom.
Technika:	Tworzenie/modyfikacja testów dla każdego okna w celu sprawdzenia właściwej nawigacji oraz stanów obiektu dla każdego okna aplikacji oraz obiektów.
Kryteria kompletności:	Każde okno zostało z powodzeniem sprawdzone czy jest zgodne z wersją odniesienia lub czy spełnia przyjęte standardy.
Uwagi specjalne:	Niektóre opcje niestandardowych obiektów mogą być niedostępne.

3.1.3. Testowanie spójności danych i baz danych

[Bazy danych i procesy bazodanowe powinny być testowane jako oddzielny podsystem wewnątrz <nazwa projekt>. Ten podsystem powinien zostać przetestowany bez udziału interfejsu użytkownika (jako interfejsu do danych). Dodatkowo rozpoznanie w DBMS (systemie zarządzania bazami danych) powinno zostać wykonane w celu identyfikacji narzędzi/technik, które istnieją i mogą wesprzeć testowanie opisane niżej.]

Cel testu:	Zapewnia bazodanowym metodom dostępu i procesom funkcjonować właściwe i bez przekłamań.
Technika:	Wywoływanie każdej bazodanowej metody i procesu, wprowadzając dane właściwe i niewłaściwe (lub zapytania o dane).
Kryteria kompletności:	Wszystkie bazodanowe metody i procesy są zaprojektowane i wykonały się bez strat danych.
Uwagi specjalne:	Testowanie może wymagać środowiska DBMS lub sterowników umożliwiających wprowadzenie lub zmodyfikowanie danych w bazie. Procesy powinny być wywoływane ręcznie. Małe lub ograniczone bazy danych (z ograniczoną liczbą krotek) powinny być użyte wyłącznie do wykrywania operacji nie spełniających kryteriów.

3.1.4. Profil wydajności

[Profil wydajności jest testem wydajności, w którym czas reakcji, częstotliwość transakcji oraz pozostałe wymagania czasowe są mierzone i liczone. Celem jest sprawdzenie czy wymagana wydajność jest spełniona. Profil wydajności jest implementowany i wykonywany, aby ukształtować i zapewnić odpowiedni poziom wyników wydajności jako funkcji takich czynników jak obciążenie i konfiguracja sprzętowa.

Uwaga: Transakcje poniżej odnoszą się do "logicznych zadanych (obowiązkowych)

transakcji" ("logical business transactions"). Transakcje tego typu są zdefiniowane jako specyficzne funkcje, o których zakłada się, że ostateczny użytkownik (end user) systemu będzie z nich korzystał używając przedmiotu testu, takie jak dodanie lub modyfikacja kontraktu.]

Cel testu:	Sprawdza wydajność systemu dla wyznaczonych transakcji lub zadanych funkcji w warunkach: • normalnego (przewidywanego) obciążenia, • obciążenia większego niż przewidywane.
Technika:	Wykonywanie testów wykorzystywanych w testach funkcjonalności. Modyfikacja plików z danymi w celu zwiększenia liczby transakcji lub modyfikacja skryptów w celu zwiększenia liczby iteracji każdej pojawiającej się transakcji. Skrypty powinny zostać wykonane na jednej maszynie (najlepszy sposób na przetestowanie i porównanie wyników w sytuacji, gdy mamy jednego użytkownika i pojedynczą transakcję) i powtarzane w sytuacji, gdy mamy wielu użytkowników (wirtualnych lub rzeczywistych, patrz uwagi specjalne).
Kryteria kompletności:	Pojedyncza transakcja / pojedynczy użytkownik: Skończone z powodzeniem skryptów testowych, pozbawione błędów i wykonane w określonym i przewidywanym czasie. Wiele transakcji / wielu użytkowników: Skończone z powodzeniem skryptów testowych, pozbawione błędów i w akceptowalnym przedziale czasowym.
Uwagi specjalne:	Niezależne testowanie wydajności jest przeprowadzane z obciążeniem na serwerze. Istnieje kilka metod do wykonania tego, wliczając: • "Kierowanie transakcji" bezpośrednio do serwera, zazwyczaj w formie zapytań SQL-owych. • Stworzenie "wirtualnego" obciążenia użytkownika w celu symulacji wielu (zazwyczaj kilkuset) klientów. Zdalne Narzędzia Emulacji są używane w realizacji tego. Ta technika może być także użyta do przeładowywania sieci. • Używacie wielu fizycznych klientów, każdy odpalający skrypty, aby wywołać procedury ładowania w systemie. Testy powinny zostać przeprowadzone na maszynie, która jest polecana jako spełniająca minimalne wymagania do sprawnego działania programu, czyli na docelowej platformie. To zezwoli na pełną kontrolę i właściwe pomiary. Bazy danych używane w testowaniu wydajności powinny być albo rzeczywistych rozmiarów, albo równo wyskalowane.

3.1.5. Testowanie ładowania danych

[Testowanie ładownia danych jest testem wydajnościowym, który skupia się na obiekcie testowanym przy różnym obciążeniu oraz mierzy i wylicza wydajność i zdolność do poprawnego kontynuowania funkcji przy tym obciążeniu. Celem jest doprowadzenie oprogramowania do stanu, kiedy funkcje systemu działają poprawnie przy zakładanym maksymalnym obciążeniu. Dodatkowo testowanie ładowania danych wylicza charakterystyki wydajności (czas reakcji, częstotliwość transakcji oraz pozostałe wymagania czasowe).]

[Uwaga: Transakcje poniżej odnoszą się do "logicznych zadanych (obowiązkowych) transakcji" ("logical business transactions"). Transakcje tego typu są zdefiniowane jako specyficzne funkcje, o których zakłada się, że ostateczny użytkownik (end user) systemu będzie z nich korzystał używając przedmiotu testu, takie jak dodanie lub modyfikacja kontraktu.]

Cel testu:	Sprawdza wydajność czasową dla wyznaczonych transakcji i przypadków użycia w różnych warunkach obciążeniowych.
Technika:	Używanie testów wyznaczonych dla cyklu testowania funkcjonalności lub sprawdzanie zakresu funkcjonalności (Business Testing). Modyfikacja plików z danymi (w celu zwiększenia liczby transakcji) lub testów w celu zwiększenia liczby wystąpień każdej transakcji.
Kryteria kompletności:	Wiele transakcji / wielu użytkowników: Zakończenie testów bez żadnych błędów i w akceptowalnym przedziale czasowym.
Uwagi specjalne:	Testowanie ładownia danych powinno zostać przeprowadzone na docelowej platformie. To zezwoli na pełną kontrolę i właściwe pomiary. Bazy danych używane w testowaniu wydajności powinny być albo rzeczywistych rozmiarów, albo równo wyskalowane.

3.1.6. Testowanie obciążenia

[Testowanie obciążenia jest testem wydajnościowym zaimplementowanym i wykonanym w celu znalezienia błędów spowodowanych niskimi zasobami lub rywalizacją o nie. Mało dostępnej pamięci lub przestrzeni na dysku może ujawniać defekty niewidoczne przy pracy w normalnych warunkach. Pozostałe defekty mogę być spowodowane "walką" o zasób współdzielony taki jak dostęp do bazy danych lub pasmo sieci komputerowej. Testowanie obciążenia może być także użyte do zbadania maksymalnego obciążenia, z którym obiekt testu da sobie radę.]

[UWAGA: Transakcje poniżej odnoszą się do "logicznych zadanych (obowiązkowych) transakcji" ("logical business transactions")].

Cel testu:	Sprawdza czy obiekt testu działa poprawnie i bez błędów przy następujących warunkach: • mało lub wcale niedostępna pamięć na serwerze (RAM i DASD) • maksymalna (aktualnie lub fizycznie dopuszczalna) liczba klientów podłączonych (lub symulowanych) • wielu użytkowników wykonuje te same transakcje na tych samych danych kontach • najgorszy przypadek obciążenia/połączenia transakcji (patrz wyżej testowanie wydajności). UWAGI: Cel testowanie obciążeniowe może być także określone jako zidentyfikowanie i udokumentowanie warunków, przy których system nie działa prawidłowo. Testowanie obciążeniowe po stronie klienta zostało omówione w podrozdziale 3.10, testowanie konfiguracyjne.
Technika:	Używanie testów wyznaczonych dla profili wydajność i testów ładowania danych. W celu przetestowania w warunkach ograniczonych zasobów, testy powinny być przeprowadzone na pojedynczej maszynie, RAM lub DASD na serwerze powinien być zredukowany (lub ograniczony). Pozostałe testy obciążeniowe powinno się przeprowadzać dla wielu klientów, z których każdy powinien wykonywać ten sam test lub testy uzupełniające w celu stworzenia najgorszego przypadku poziomu/połączenia transakcji.
Kryteria kompletności:	Wszystkie zaplanowane testy są wdrożone, a wyznaczone ograniczenia systemowe zostały osiągnięte/przekroczone bez błędów z oprogramowaniem (lub warunki, w których wystąpił błąd nie są zawarte w specyfikacji)
Uwagi specjalne:	Obciążenie sieci może wymagać narzędzi do wysyłania pakietów i wiadomości. DASD użyty w systemie powinien czasowo zostać zredukowany w celu ograniczenia przestrzeni używanej do rozrastania się bazy danych. Symulowani klienci powinni uzyskiwać dostęp do tych samych zasobów.

3.1.7. Testowanie obciążenia (*volume testing*)

[Testowanie obciążenia poddaje przedmiot testu załadowaniu dużej ilości danych w celu ustalenia czy zostały osiągnięte wartości graniczne, dla których oprogramowanie zaczyna błędnie funkcjonować. Testy obciążenia rozpoznają maksymalne stałe obciążenie, przy którym program działa poprawnie przez dany okres czasu. Na przykład, jeżeli przedmiot testu przetwarza zbiór rekordów bazodanowych w celu wygenerowania raportu, w testowaniu obciążenia użyłby się dużej testowej bazy danych i sprawdziło czy oprogramowanie zachowuje się normalnie i produkuje poprawny raport].

Cel testu:	Sprawdza czy obiekt testu działa poprawnie przy następujących scenariuszach wysokiego obciążenia: • Maksymalna (aktualna lub fizycznie możliwa) liczba klientów podłączonych (lub symulowanych) wykonujących wszyscy ten sam, najgorszy (wydajnościowo) przypadek działania przez dłuższy okres czasu. • Maksymalny rozmiar bazy danych został osiągnięty (aktualnych rozmiarów lub równo zeskalowane) oraz wielorakie transakcje zapytań/raportu zostały wykonane równocześnie.
Technika:	Używanie testów wyznaczonych dla profili wydajność i testów ładowania danych. Wielodostęp powinien powinien zostać użyty albo wykonując te same testy, albo uzupełniając testy w celu wytworzenia najgorszego przypadku obciążenia/połączenia transakcji (patrz test obciążeniowy wyżej) przez dłuższy okres czas. Maksymalny rozmiar bazy danych jest tworzony (aktualnych rozmiarów, równo zeskalowany lub wypełniony wykorzystywanymi danymi) oraz używany jest wielodostęp wykonujący transakcje zapytań / raportu stale przez dłuższy okres czasu.
Kryteria kompletności:	Wszystkie zaplanowane testy są wykonane, a wyznaczone ograniczenia systemowe zostały osiągnięte/przekroczone bez błędów z oprogramowaniem (lub warunki, w których wystąpił błąd nie są zawarte w specyfikacji)
Uwagi specjalne:	Jak długi okres czasu należy uważać za akceptowalny przy warunkach dużego obciążenia (jakie wyżej opisano)?

3.1.8. Testowanie bezpieczeństwa i kontroli dostępu

[Testowanie bezpieczeństwa i kontroli dostępu skupia się na dwóch kluczowych aspektach bezpieczeństwa:

1. *Bezpieczeństwo z poziomu aplikacji włączając dostęp do funkcji operujących na danych*
2. *Bezpieczeństwo z poziomu systemu, włączając logowanie do systemu, dostęp z zewnątrz*

Bezpieczeństwo z poziomu aplikacji zapewnia to, że bazując na żądanym bezpieczeństwie, aktorzy mogą używać tylko ściśle określonych funkcji / przypadków użycia lub mają ograniczoną ilość danych dostępnych dla nich. Na przykład, każdy ma pozwolenie na wprowadzanie danych i tworzenie nowych rekordów, ale tylko kierownik może je usuwać. Jeżeli jest mowa o bezpieczeństwie na poziomie danych, testowanie zapewnia, że użytkownik jednego typu może tylko oglądać informacje klientów, łącznie z danymi finansowymi, jednakże użytkownik drugiego typu widzi tylko dane demograficzne tego samego klienta.

Bezpieczeństwo z poziomu systemu zapewnia, to że tylko Ci aktorzy, którym się udzieliło dostępu do systemu, są zdolni uzyskać dostęp do aplikacji i tylko właściwymi drogami.]

Cel testu:	Bezpieczeństwo z poziomu aplikacji: Sprawdza czy aktor może uzyskać dostęp tylko do tych funkcji / danych, do których rodzaju użytkownika posiada zezwolenia. Bezpieczeństwo z poziomu systemu: Sprawdza czy tylko tym aktorom, którzy mają uprawnienia do dostępu do systemu i aplikacji, zezwalany jest do nich dostęp.
Technika:	Bezpieczeństwo z poziomu aplikacji: Identyfikowanie i wyliczanie każdego rodzaju aktora i wszystkich funkcji / danych, do których ma zezwolenia. Tworzenie testów dla każdego rodzaju aktora i sprawdzanie wszystkich pozwoleń poprzez tworzenie transakcji specyficznych dla każdego aktora użytkownika. Zmienianie rodzaju użytkownika i ponowne wykonanie wcześniejszych testów dla tego użytkownika. Sprawdzanie za każdym razem czy dodatkowe funkcje / dane są prawidłowo udostępniane lub odmawiany jest do nich dostęp. Bezpieczeństwo z poziomu systemu (patrz uwagi specjalne poniżej)
Kryteria kompletności:	Dla każdego znanego rodzaju aktora odpowiednie funkcje / dane są dostępne i wszystkie transakcje działają tak jak się spodziewano i działały w testach funkcjonalnych.
Uwagi specjalne:	Dostęp do systemu musi być przeanalizowany / przedyskutowany z właściwym administratorem sieci lub systemu. Testowanie te może nie być wymagane, w przypadku gdy należy to do zadania administratora sieci.

3.1.9. Testowanie typu Awaria / Regeneracja

[Testowanie typu Awaria / Regeneracja zapewnia to, że przedmiot testu może z powodzeniem ulec awarii, a następnie regeneracji spowodowanej błędnym działaniem sprzętu, oprogramowania lub niesprawnością sieci z nadmierną utratą danych lub spójnością danych.

Testowanie typu Awaria zapewnia to, że dla systemów, które muszą być ciągle wykorzystywane (włączone), w przypadku awarii alternatywny lub zapasowy system właściwie zastąpi niesprawny system bez straty danych lub transakcji.

Testowanie typu Regeneracja jest "wrogim" procesem testującym, w którym aplikacja lub system jest poddany ekstremalnym warunkom (lub symulowanym warunkom) w celu spowodowania awarii, takiej jak błędy urządzeń wejścia/wyjścia lub niewłaściwe uchwyty / klucze bazodanowe. Procesy regeneracji są wywoływane i aplikacja / system jest monitorowany i / lub badany czy prawidłowa regeneracja aplikacji / systemu / i danych została osiągnięta.]

Cel testu:	Sprawdza czy procesy regeneracyjne (ręczne lub zautomatyzowane) właściwie przywróciły bazę danych, aplikację i system do żądanego, znanego stanu. Następujące warunki są muszą być wzięte pod uwagę: • przerwa w dostępie prądu do klienta • przerwa w dostępie prądu do serwera • przerwa w dostępie usług komunikacyjnych w sieci, do której należy serwer(y) • przerwa w dostępie usług komunikacyjnych lub spadki napięć dla DASD lun kontrolera DASD • niedokończone cykle (przerwanie procesu filtrowania danych, przerwanie procesu synchronizacji danych) • niewłaściwe uchwyty / klucze bazodanowe • niewłaściwy / uszkodzony element danych w bazie danych

Technika:	Testy wyznaczone dla cyklu testów funkcjonalnych lub sprawdzających zakres funkcjonalności (Business Testing) powinny zostać użyte w celu stworzenia serii transakcji. Kiedy żądany punkt początkowy testu zostanie osiągnięty, następujące czynności powinny zostać wykonane (lub zasymulowane) osobno: • przerwa w dostępie prądu do klienta: odłączanie prądu od komputera • przerwa w dostępie prądu do serwera: symulacja lub rozpoczęcie procedur odłączający prąd od serwera • przerwa w dostępie usług komunikacyjnych w sieci, do której należy serwer(y): symulacja lub zapoczątkowanie strat komunikacyjnych w sieci (fizyczne odłączenie kabli komunikacyjnych lub wyłączenie zasilania serwera sieciowego / routera) • przerwa w dostępie usług komunikacyjnych lub spadki napięć dla DASD lun kontrolera DASD: symulacja lub fizyczna likwidacja komunikacji z jednym lub więcej kontrolerami lub urządzeniami DASD Kiedy powyższe warunki / symulowane warunki zostają osiągnięte, dodatkowe transakcje powinny się wykonać i dopóki się nie osiągnie tego drugiego punktu stanu testowania, procedury regeneracji powinny zostać wstrzymane. Testowanie dla niedokończonych cykli wykorzystuje te same techniki co opisane wyżej z wyjątkiem procesów bazodanowych, które same powinny się przerwać lub przedwcześnie skończyć. Testowanie dla powyższych warunków wymaga, aby znane stan bazy danych został osiągnięty. Kilka bazodanowych pól, wskaźników i kluczy powinno być uszkodzone ręcznie wewnątrz bazy danych (używając narzędzi bazodanowych). Dodatkowe transakcje powinny być wykonane używając testów wykorzystywanych do testowania funkcji aplikacji i sprawdzających zakres funkcjonalności, wykonując pełne cykle.
Kryteria kompletności:	We wszystkich przypadków opisanych wyżej: aplikacji, bazy danych i systemu, kompletność testu oznacza powrót do znanego, żądanego stanu. Stan ten oznacza, że starty danych ograniczyły się do znanych uszkodzonych pól, wskaźników / kluczy i wskazano procesy lub transakcje, które nie zostały skończone z powodu przerwań.

Uwagi specjalne:	Testu typy regeneracja są wysoko niepożądane. Procedury odłączające okablowanie (symulujące brak prądu lub komunikacji) mogą nie być wskazane lub wykonalne. Alternatywne metody, takie jak diagnostyka narzędzi oprogramowania może być bardziej wymagana. Zasoby z systemowych (lub operacji komputerowych), bazy danych i sieciowych zestawów są wymagane. Testy powinny zostać wykonane po godzinach lub na odizolowanej maszynie(ach).

3.1.10. Testowanie konfiguracyjne

[Testowanie konfiguracyjne sprawdzają operacje przedmiotu testu wykonywane na różnych konfiguracjach oprogramowania i sprzętu. W większości środowisk produkcyjnych poszczególne specyfikacje sprzętowe na stacje robocze klienta, połączenia sieciowe i serwery bazodanowe różnią się. Stacje robocze klienta mogą mieć wgrane różne oprogramowanie (np. aplikacje, sterowniki, itd.) i w każdej chwili różne konfiguracje mogą być aktywne i używać różnych zasobów.]

Cel testu:	Sprawdza czy przedmiot testu działa właściwie na wymaganych konfiguracjach sprzętu / oprogramowania.
Technika:	Używanie skryptów testu funkcjonalnego. Otwieranie / zamykanie różnych niebędących przedmiotem testu powiązanego oprogramowania takiego jak aplikacje Microsoft, Excel i Word, zarówno jako część testu, jak i przed rozpoczęciem testu. Wykonanie wybranych transakcji w celu zasymulowania interakcji aktora z przedmiotem testu i oprogramowaniem niebędącym przedmiotem testu. Powtarzanie procesu powyżej minimalizując dostępną konwencjonalną pamięć po stronie klienta.
Kryteria kompletności:	Dla każdej kombinacji przedmiotu testu oraz oprogramowania niebędącego przedmiotem testu wszystkie transakcje wykonały się pomyślnie, bez błędu.
Uwagi specjalne:	Które oprogramowanie niebędące przedmiotem testu jest potrzebna, dostępne na pulpicie? Jakie aplikacje są najczęściej używane? Jakie dane przetwarzają aplikacje (np. duże arkusze danych otwarte w Excel'u, 100 stronicowy dokument w Word'ie)? Wewnętrzne systemu, NetWare, serwery sieciowe, bazy danych, itd. także powinny być opisane w dokumentacji jako część testu.

3.1.11. Testowanie instalowania

[Testowanie instalowania ma dwa cele. Pierwszym jest zabezpieczenie tego, że

oprogramowanie może być zainstalowane w różnych warunkach, takich jak nowa instalacja, zastąpienie nowszą wersją (upgrade) oraz kompletna lub z wybranymi opcjami ("custom") instalacja w normalnych lub nienormalnych warunkach. Nienormalne warunki to takie, kiedy nie ma wystarczającej przestrzeni na dysku lub brak przywilejów do tworzenia folderów, itd. Drugim celem jest zidentyfikowanie tego, czy po zainstalowaniu, oprogramowanie działa poprawnie. Najczęściej oznacza to wykonanie wielu testów, które zostały opracowane dla testowania funkcjonalności.]

Cel testu:	Sprawdza czy przedmiot testu właściwie instaluje się na każdym wymaganej konfiguracji sprzętowej w następujących warunkach (jako wymaganych): Nowa instalacja, nowa maszyna, na której nigdy wcześniej nie instalowano <Nazwa Projektu>. Uaktualnienie maszyny z wcześniej zainstalowanym <Nazwa Projektu>, ta sama wersja. Uaktualnienie maszyny z wcześniej zainstalowanym <Nazwa Projektu> starsza wersja.
Technika:	Ręcznie lub poprzez stworzenie zautomatyzowanych skryptów sprawdzanie stanu maszyny będącej przedmiotem testu (nowa - <Nazwa Projektu> nigdy nie instalowana, ta sama lub starsza wersja <Nazwa Projektu> jest zainstalowana. Manually or develop automated scripts to validate the condition of the target machine (new - <Project Name> never installed, <Project Name> same version or older version already installed). Włączenie / Wykonanie instalacji. Używając wcześniej zdefiniowanego podzbioru skryptów testujących funkcjonalność wykonywanie transakcji.
Kryteria kompletności:	Transakcje <Nazwa Projektu> wykonywane pomyślnie i bez błędu.
Uwagi specjalne:	Które transakcje <Nazwa Projektu> powinny zostać wybrane, aby mieć pewność, że aplikacja została pomyślnie zainstalowana i nie brakuje głównych modułów oprogramowania?

3.2. Narzędzia

Poniższe narzędzia będą miały zastosowanie w tym projekcie.

[UWAGA: należy usunąć lub dodać punkty, które uważa się za stosowne.]

	Narzędzie
Zarządzanie testem	
Wyszukiwanie błędów	
Testowanie funkcjonalności	
Testowanie wydajności	

	Narzędzie
Zarządzanie projektem	

4. Zasoby

[Rozdział ten przedstawia zalecane zasoby dla próby testu <Nazwa Projektu>, ich główne obowiązki, wiedzę i umiejętności.]

4.1. Pracownicy

Tabela poniżej przedstawia założenia o załodze projektu.

[UWAGA: należy usunąć lub dodać punkty, które uważa się za stosowne.]

Zasoby ludzkie		
Pracownik	**Minimalna rekomendowana liczba zasobów** **(liczba pracowników zajmujących się tym cały czas)**	**Przydzielone obowiązki / komentarze**
Kierownik testów		Zarządza niedopatrzeniami Odpowiedzialność: • prowadzi nadzór techniczny, • zarządza właściwymi zasobami, • zarządza raportami.
Projektant testu		Identyfikuje, ustala ważność oraz wdraża przypadki testowe. Odpowiedzialność: • wykonuje plan testów, • wykonuje model testów, • ocenia skuteczność testów.
Tester		Wykonuje testy. Odpowiedzialność: • wykonuje testy, • opisuje wyniki, • wychwytuje błędy, • dokumentuje żądania zmian.

Zasoby ludzkie		
Administrator systemu testów		Zajmuje się zarządzaniem i utrzymaniem środowiska oraz wkładami finansowymi testu. Odpowiedzialność: • administruje systemem zarządzania testem, • instaluje / zarządza dostępem pracowników do systemów testujących.
Administrator Bazy Danych / Zarządca Bazy Danych		Zajmuje się zarządzaniem i utrzymaniem środowiska oraz wkładami finansowymi danych (bazy danych) testu. Odpowiedzialność: • administruje test danych (bazy danych).
Projektant		Identyfikuje i definiuje operacje, atrybuty i powiązania klas testów. Odpowiedzialność: • identyfikuje i definiuje klasy testów, • identyfikuje i definiuje pakiety testów.
Programista		Implementuje i agreguje klasy i pakiety testów. Odpowiedzialność: • tworzy klasy i pakiety testów zimplementowne w modelu testu.

4.2. System

Tabela poniżej ustala zasoby systemowe dla testowanego projektu.

Poszczególne elementy systemu testu nie są znane do końca w tej chwili. Zaleca się, aby system symulował środowisko produkcji, zmniejszając dostęp i rozmiar bazy danych jeżeli / gdzie jest to stosowne.

Zasoby systemowe	
Zasób	***Nazwa / Typ***
Serwer bazodanowy	
Sieć / Podsieć	B.D.
Nazwa Serwera	B.D.
Nazwa Bazy Danych	B.D.
Testowy Komputer Klienta (Client Test PC's)	
Włączając specjalne wymagania konfiguracyjne	B.D.
Repozytorium testu	
Sieć / Podsieć	B.D.

Zasoby systemowe	
Nazwa Serwera	B.D.
Komputer Wykonawcy Testu	B.D.

[UWAGA: należy usunąć lub dodać punkty, które uważa się za stosowne.]

5. Kamienie milowe

[Testowanie <Nazwa Projektu> powinno zawierać czynności testowe dla każdego punktu testowania wyodrębnionego w poprzednich rozdziałach. Poszczególne kamienie milowe powinny być wyznaczone w celu komunikacji pomiędzy stanem projektu a dokonaniami.]

Kamień milowy	*Poświęcony czas*	*Data rozpoczęcia*	*Data zakończenia*
Planowanie testów			
Projektowanie testów			
Implementacja testów			
Wykonanie testów			
Podsumowanie (ocenienie) testów			

6. Dokumenty powstałe w trakcji i po przeprowadzeniu testów

[W tym rozdziale wyliczane są różne dokumentu, narzędzia i raporty, które będą stworzone. Określane jest kto będzie je robił, kiedy i do kogo będą dostarczone.]

6.1. Model testów

[Ten rozdział identyfikuje raporty, które będą stworzone i dostarczone na podstawie modelu testu.]

6.2. Sprawozdanie testu

[Opisuje metody i narzędzia użyte w celu zarejestrowania i sporządzenia sprawozdania testu opisującego wyniki testu i jego status.]

6.3. Raport podsumowujący testy

[W tym rozdziale zidentyfikowane są metody i narzędzia użyte do zarejestrowania, znalezienia i sporządzenia sprawozdania opisującego incydenty podczas testowania i ich status.]

7. Dodatek A: Zadania projektu

Poniżej są zadania powiązane z testem:

- Planowanie testu
  - Wyznaczenie wymagań dla testu
  - Ocena ryzyka
  - Wyznaczenie strategii testu
  - Identyfikacja zasobów
  - Stworzenie harmonogramu

- Wygenerowanie planu testów

- Projektowanie testu
  - Analiza obciążeniowa
  - Identyfikacja i opisanie przypadków testowych
  - Identyfikacja i konstrukcja procedur testowania
  - Przejrzanie i zwiększenie pokrycia testu
- Implementacja testu
  - Zapisanie i zaprogramowanie skryptów testowych
  - Identyfikacja funkcjonalności specyficznej dla testu w modelu projektowym i implementacyjnym
  - Ustalenie zewnętrznych zbiorów danych
- Wykonanie testu
  - Wykonanie procedur testowych
  - Ocena przeprowadzonego testu
  - Regeneracja po zatrzymanym teście
  - Sprawdzenie wyników
  - Szukanie przyczyny wyników innych niż się spodziewaliśmy
  - Sporządzenie sprawozdania o błędach
- Ocenienie Testu
  - Ocena pokrycia przypadków testowych
  - Ocena pokrycia kodu
  - Analiza błędów
  - Ustalenie czy kryteria kompletności i powodzenia testu zostały osiągnięte