Aplikacja do zarządzania nieruchomościami
Plan testów

Version 1.0

Revision History

Date	Version	Description	Author
25.06.22	1.0	Opracowano punkty 1 - 10	Jakub Woźniak

Table of Contents

Plan testów

1. Wstęp

1.1 Cel

Celem dokumentu jest przedstawienie ogólnych założeń testowych oprogramowania tworzonego dla aplikacji zarządzania nieruchomościami Home4You.

Dokument ten zawiera fazy i techniki testowania oraz harmonogram przeprowadzanych testów.

Celem testowania jest zapewnienie niezawodności oprogramowania i sprostanie wymaganiom klienta i standardom konstrukcji oprogramowania.

Główne cele:

- Uzyskanie poprawnie działającego kodu
- Uzyskanie poprawnej specyfikacji wymagań
- Oszacowanie niezawodności systemu

1.2 Zakres

Zakres testów obejmuje takie element systemu jak:

- Analiza danych
- GUI
- Wprowadzanie danych
- Tworzenie raportów i gromadzenie danych
- Bezpieczeństwo
- Niezawodność i jakość pracy serwera
- Podsumowanie

1.3 Odbiorcy

Dokument ten stworzony jest w założeniu, że odbiorcą są menadżer projektu, zespół projektowy i zespół testerów. Niektóre części dokumentu mogą być udostępnione klientowi/użytkownikowi, jeżeli wyrazi taką wolę.

1.4 Słownik

GUI – interfejs graficzny użytkownika
Wprowadzanie danych – wydajność dla klienta
Bezpieczeństwo – zagrożenia, bezpieczeństwo informacji poufnych i wrażliwych
Bug – usterka programu komputerowego powodująca jego nieprawidłowe działanie
Patching – poprawka lub uaktualnienie do programu (rzadziej do danych), przeznaczona do usunięcia pewnych problemów, błędów

2. Ocena i motywacja

Główną motywacją do przeprowadzenia testów jest znalezienie błędów i naprawa ich w jak najszybszym czasie, tak by mieć pewność ze gotowy produkt będzie w stanie sprostać oczekiwaniom klienta.

2.1 Tło

Aplikacja przechowuje wrażliwe i prywatne dane użytkowników takie jak adres, hasła, adres email, numery telefonów, dlatego musi być jak najlepiej zabezpieczona i nie może posiadać żadnych błędów pozwalających na znalezienie wrażliwości systemu i wykorzystanie ich w celach zaszkodzenia użytkownikom.

2.2 Cele

Cele misji mogą być podzielone na 2 fazy:

1. Cel bezpośredni:

    - znajdywanie błędów – główny cel to znalezienie podatności i błędów we wczesnej fazie testowania. Im więcej błędów zostanie znalezionych tym większe prawdopodobieństwo otrzymania aplikacji bez błędów.
    - zapobieganie błędom – następuje po znalezieniu błędu, ma na celu zapobieganiu powtórzeniom tego samego i podobnych błędów

2. Cele długofalowe:

    - zapewnienie jakości – zapewnienie jakości następuje poprzez przeprowadzanie testów mających na celu zbadanie poprawności, integralności, wydajności i niezawodności aplikacji.
    - Zapewnienie satysfakcji klienta
    - Zarządzanie ryzykiem – aby zapewnić wyżej wymienione cechy należy wziąć pod uwagę ryzyka związane z tworzeniem systemu i za wczasów im przeciwdziałać, aby zapewnić wysoką jakość produktu

2.3 Motywatory

Głównymi motywatorami do przeprowadzenia testów są:

- Chęć ograniczenia kosztów
- Chęć ograniczenia ryzyka
- Zdobycie zaufania i satysfakcji klienta
- Zbadanie adaptacyjności oprogramowania
- Optymalizacja
- Zapewnienie bezpieczeństwa systemu
- Poprawienie jakości produktu

3. Lista produktów do testowania

1) Wydajność systemu w różnym środowisku (Windows 7, 8, 10, 11, MacOS, Linux)
2) Zużycie zasobów (procesor, pamięć operacyjna, miejsce na dysku)
3) Jakość systemu (jakość i czytelność komunikatów, informacji, czytelność poszczególnych okien,

ergonomia)

4) Zabezpieczenie systemu związane z Cyber Security:
    - Zapewnienie nienaruszalności prywatności, integralności, dostępności i tajności danych
    - Sprawdzenie możliwości wejścia do systemu przez niepowołane osoby
    - Sprawdzenie możliwości dostępu do wrażliwych plików użytkowników przez niepowołane osoby
    - Sprawdzenie jakości zabezpieczeń haseł użytkowników
    - Sprawdzenie czy system zareaguje natychmiastowym odcięciem dostępu do danych przy wyryciu próby włamania
5) Niezawodność systemu
6) Integralność i bezpieczeństwo przed utratą danych. Sprawdzenie ewentualnych skutków braku prądu czy nagłym brakiem dostępu do Internetu czy bazy danych. Ocena ewentualnych strat.
7) Elastyczność systemu, jego zdolność do modyfikacji i wprowadzania zmian.
8) Kompletność założeń funkcjonalnych
9) Ocena narzędzi ułatwiających obsługę systemu, poprawiających jego jakość, czytelność i łatwość obsługi

4. Zarys testów

1. Wewnętrzne testy aplikacji prototypowej – po wykonaniu odpowiedniej ilości testów jednostkowych możliwe jest wydanie wersji prototypowej i przetestowanie jej w systemie wewnętrznym. Na tym etapie kod wciąż jest w trakcie konstrukcji i możliwe są zmiany w implementacji.
2. Testy Alpha – po skończeniu pisania kodu i implementacji wymagań funkcjonalnych możliwe jest przeprowadzenie testów Alpha. Na tym etapie następuje patchowanie bugów.
3. Testy Beta – następują po eliminacji wszystkich poważnych bugów z fazy testów Alpha. W tej fazie następuje udoskonalenie aplikacji i dołożenie wszelkich starań by końcowy produkt sprostał oczekiwaniom klienta. Eliminowane są wszystkie mniej istotne bugi.

4.1 Zarys planowanych testów

- Testowanie GUI – sprawdzenie czy interfejs graficzny jest napisany zgodnie ze specyfikacją (poprawność wyświetlania etykiet, poprawność wyrównania przycisków, nakładanie się pól, zmiana rozmiaru strony i jej wpływ na jakość wyświetlanych treści)
- Testy integracyjne – sprawdzenie interfejsów pomiędzy modułami, systemami i interakcje z innymi częściami systemu (np. system operacyjny, sprzęt)
- Testowanie integralności danych – sprawdzenie czy zaktualizowane wartości pojawiają się we wszystkich formularzach/ekranach. Upewnienie się że mapowanie między różnymi formularzami jest dokładne i zgodne z kodem (mapowanie tabel, kolumn i typów danych)
- Testy funkcjonalne – sprawdzenie czy wszystkie funkcjonalności są zaimplementowane zgodnie ze specyfikacją
- Testy wydajnościowe – testowanie limitów i granic systemu. Wykorzystania zasobów (np. przydzielenie ograniczonej pamięci RAM)
- Testy instalacji – sprawdzenie zachowania instalacji modułu w sytuacji normalnej i ekstremalnej
- Testy konfiguracyjne – sprawdzenie jak aplikacja działa w różnych środowiskach o różnej konfiguracji (np. z innym oprogramowaniem czy w systemie z innym procesorem)
- Testy przywracania systemu – sprawdzenie jak zachowuje się system po niezaplanowanym odłączeniu od

bazy danych/zasilania. Sprawdzenie integralności danych po takim zajściu i sprawdzenie czy jakieś dane zostały utracone.
- Analiza podatności – sprawdzenie jakie zasoby są najbardziej istotne dla prawidłowego funkcjonowania systemu, np. miejsce na dysku, jakość połączenia internetowego, czy wydajność procesora
- Testy warunków skrajnych – przetestowanie zachowania systemu w sytuacji przeciążenia systemu.
- Testy objętościowe – w jaki sposób system zareaguje w środowisku z dużą ilością danych (poddanemu ogromnemu napływowi danych) np. zwiększając objętości danych w bazie danych.
- Testy skalowalności – określenie czy aplikacja będzie w stanie poradzić sobie ze stopniowym wzrostem obciążenia.
- Testy bezpieczeństwa – testowanie dostępności usługi przez osoby niepowołane. Sprawdzenie czy osoba nie posiadająca dostępu do danych wrażliwych może się do nich dostać.
- Testy post-release – zbieranie informacji od użytkowników na temat jakości oprogramowania, jego błędów i sugestii poprawek. Monitorowanie aplikacji i jej działania.

4.2 Zarys testów do przeprowadzenia w drugiej kolejności

- Error handling – testowanie możliwości systemu w radzeniu sobie z błędami w trakcie życia programu. Na celu ma zapewnienie radzenia sobie z błędami w przyszłości przez aplikację.
- Scenariusze użycia – sprawdzenie jak aplikacja będzie sobie radzić w trakcie codziennego użycia przez normalnego użytkownika.
- Akceptacja przez klienta – klient ma możliwość wglądu do końcowej wersji aplikacji.

4.3 Zarys testów które nie zostaną przeprowadzone

Testy które nie zostaną przeprowadzone:

- Testy jednostkowe – przeprowadzane są automatycznie przez programistów w trakcie pisania kodu
- Przenośność – aplikacja nie jest stworzona po to by zapewniać możliwość przenoszenia napisanego programu na inną platformę.
- Jakość designu / spójność w designie – odpowiedzialni są za to osoby projektujące UX
- Zdolność oprogramowania do poprawnej pracy przy maksymalnym obciążeniu – brak możliwości przeprowadzenia testów z racji niedostatecznych zasobów

5. Podejście do testowania

5.1 Initial Test-Idea Catalogs and Other Reference Sources

5.2 Testing Techniques and Types

5.2.1 Data and Database Integrity Testing

Technique Objective:	Proces weryfikacji czy baza danych funkcjonuje zgodnie z wymaganiami. Sprawdza także czy dane nie są modyfikowane lub uszkadzane podczas dostępu do bazy danych.

Technique:	1. Sprawdzenie czy możliwa jest modyfikacja danych w tabelkach (dodawanie, usuwanie itp.) 2. Sprawdzenie czy możliwe jest wyszukanie pustej wartości w bazie danych 3. Sprawdzenie czy dane zostają poprawnie zapisane w bazie danych i czy nie występują błędy 4. Sprawdzenie kompatybilności innej wersji systemu operacyjnego lub interfejsu 5. Sprawdzenie czy domyślna wartość jest przypisywana automatycznie gdy użytkownik nie wprowadzi danych
Oracles:	• Spróbować wpisać złe dane do tabeli i obserwować czy wystąpi jakiś błąd • Spróbować co się stanie jeśli będzie chciało się wprowadzić dziecko przed wprowadzeniem rodzica (nieprawidłowe użycie Primary Key i Foreign Keys) • Spróbować usunąć zapis który ma odniesienie w innej tabeli i zobaczyć czy wystąpi błąd
Required Tools:	• ACCELQ - Test Script Automation Tool • R-STUDIO - backup and recovery tools • PRIMO -installation-monitoring tools (registry, hard disk, CPU, memory, and so forth) • AWS SQL - database SQL utilities and tools • DbUnit
Success Criteria:	• Wszystkie testy przebiegły pomyślnie • Wszytskie tabelki zostały poprawnie wyświetlone • Wszytskie modyfikacje tabel przebiegly pomyślnie (UPDATE, INSERT, DROP itp.) • Relacja rodzic – dziecko sprawdzona
Special Considerations:	• Procesy powinny zostać wywołane manualnie • Użycie ograniczonej liczby rekordów mające na celu uwidocznienie występujących błędów • Testowanie może wymagać środowiska DBMS

5.2.2 Function Testing

Technique Objective:	Celem jest sprawdzenie funkcjonalności software. Koncentruje się na sprawdzeniu czy nawigacja jest łatwa i swobodna, na dostępności do usługi, sprawdzeniu czy powiadomienie o błędzie się wyświetla.
Technique:	• Identyfikacja danych wejsciowych testowych • Kalkulacja danych wyjściowych na podstawie wybranych danych wejściowych (obliczanie spodziewanego wyniku) • Egzekucja przypadków testowych • Porównanie rzeczywistego i spodziewanego wyniku testu • Przeprowadzenie smoke testu w celu weryfikacji czy GUI reaguje w taki sposób w jaki powinno. Sprawdzenie stabilności kompilacji. • Sprawdzenie czy prawidłowy błąd wyświetla się, gdy użyje się nieprawidłowych danych • Test regresji w celu sprawdzenia czy dodanie ulepszeń, nowego kodu, naprawa błędów nie przerywa istniejącej funkcjonalności i nie powoduje niestabilności •
Oracles:	• Wprowadzenie nieprawidłowych danych uwierzytelniających powinno poinformować o tym użytkownika i ponownie załadować stronę logowania • Sprawdzenie jak zachowa się system, jeżeli administrator usunie konto użytkownika, gdy ten jest zalogowany i wykonuje jakąś operację
Required Tools:	• Narzędzie do automatyzacji testów • Narzedzie do monitorowania instalacji • Narzedzie do generowania danych • Sahi • SoapUI • Narzędzia do tworzenia backupów
Success Criteria:	• Wszystkie fuknjonalne przypadki testowe zostały zakończone • Żadne krytyczne błędy nie są otwarte
Special Considerations:	• Jeśli wymaganie nie jest kompletne, jest skomplikowane lub niejasne, wykonanie testu staje się trudne i może być czasochłonne • Należy uważać żeby nie przeoczyć błędów logicznych

User Interface Testing

Technique Objective:	Sprawdzenie czy funkcjonalności software działają tak, jak są określone w specyfikacji, sprawdzając ekrany i ich elementy takie jak menu, przyciski, ikony itp.
Technique:	• Sprawdzenie wszystkich elementów GUI pod względem wielkości, pozycji, szerokości, długości. • Sprawdzenie czy jest możliwość wpisania treści tam, gdzie znajduje się na to pole • Sprawdzenie czy wiadomości o błędzie wyświetlają się poprawnie • Sprawdzenie czy font użyty w programie jest czytelny • Sprawdzenie wszystkich sekcji ekranu • Sprawdzenie jakości wyświetlanych treści w różnej rezolucji
Oracles:	• Obrazki i ikony powinny być widoczne w ten sam sposób i w tej samej rezolucji w każdej przeglądarce • Przyciski powinny działać tak jak jest to przewidziane • Jeżeli użytkownik zmniejszy lub zwiększy wielkość ekranu zdjęcia i treści prezentowane na stronie nie powinny zmienić swojej jakości i nie powinny zostać ucięte lub nie powinny na siebie zachodzić • Zmiana rezolucji z 640x480 na 600 x800 itp. i sprawdzenie jaki ma to wpływ na zmianę jakości treści
Required Tools:	Narzędzie do automatyzacji testów np. eggplant
Success Criteria:	• Przetestowanie wszystkich głównych funkcji z których użytkownicy będą korzystać najczęściej • Interfejs użytkownika jest czytelny, spójny i przyjemny w odbiorze
Special Considerations:	Nie jesteśmy w stanie sprawdzić wszystkich customowych ustawień użytkownika

5.2.3 Performance Profiling

Technique Objective:	Test ten pozwala na analize jakości aplikacji i poprawy fragementów złej jakości kodu. Pozawala zbadać jak długo zajmuje wykonanie danej czynności.
Technique:	Należy tak zmodyfikować pliki z danymi aby zwiększyć liczbę wychodzących zapytań do systemu Skrypty należy wykonać na pojedynczej maszynie i powinny być powtórzone przez wielu klientów
Oracles:	Używając narzędzia do monitorowania serwera może pomóc w ustaleniu czy problem z jakością i szybkością jest ze strony klienta, serwera webowego czy serwera aplikacji i znacznie przyspieszyć proces testowania.
Required Tools:	Narzędzie do automatyzacji skryptów Narzędzie do monitorowania instalacji Narzedzie do ograniczania zasobow Narzędzie do badania wydajności aplikacji
Success Criteria:	Technika pozwala na przetestowanie jednego lub wielu użytkowników i wykonywanych przez nich skryptów bez problemów związanych z implementacją programu
Special Considerations:	Możliwe jest użycie aplikacji pozwalającej na wytworzenie sztucznego ruchu mającego na celu symulacje wielu użytkowników na raz Możliwe użycie wielu jednostek, wykonujących jakiś skrypt testowy w celu zwiększenia obciązenia systemu Test powinien zostać wykonany na specjalnie przeznaczonej do tego maszynie

5.2.4 Load Testing

Technique Objective:	Zbadanie wydajności aplikacji pod specyficzną ilością obciążenia. Pozwala stwierdzić, jak aplikacja zachowuje się np. podczas próby dostępu do niej wielu użytkowników w jednym czasie. Testy mają na celu zapewnienie stabilności aplikacji.

Technique:	• Przygotowanie danych dla kazdej z transakcji • Liczba użytkowników mających dostęp do systemu musi być przewidziana • Szybkość łącza musi zostać określona. • Użytkownicy używają różnych przeglądarek internetowych i systemów operacyjnych • Konfiguracja serwerów internetowych i serwerów baz danych • Analiza rezultatów
Oracles:	• Stworzenie scenariusza testowego na podstawie analizy najczęstszych zachowań przeciętnych użytkowników w danej aplikacji • Należy wybrać odpowiednie narzędzie do testowania obciążenia • Należy ustalić realistyczny wskaźnik referencyjny
Required Tools:	• Narzędzie do automatyzacji skryptów • Narzędzie do overloadingu • Narzędzie do monitorowania instalacji • Narzędzie do generowania danych
Success Criteria:	Technika ma na celu przetestowanie obciążenie zadaniami i zapewnienie bezawaryjności systemu związanych z implementacją
Special Considerations:	• Bazy danych użyte do testowania powinny być specjalnie ograniczone wielkościowo • Testy obciążeniowe powinny zostać przeprowadzone na dedykowanych maszynach w celu zapewnienia dokładnych wyników

5.2.5 Stress Testing

Technique Objective:	Sprawdzenie jak system będzie zachowywać się pod wpływem sytuacji stresowych i w jakiej sytuacji przestanie funkcjonować poprawnie wykorzystując np. środowisko, w którym pamięć RAM do wykorzystania jest bardzo mała, ilość klientów połączonych z systemem jest bardzo duża, wiele użytkowników wykorzystuje ta sama funkcję, jak zachowuje się w trakcie przeciążenia

Technique:	• Należy przeprowadzić test na jednej maszynie i zmniejszyć na niej ilość dostępnej pamięci RAM • Należy przeprowadzić test, w którym wielu użytkowników będzie wykonywać tą samą czynność w tym samym czasie • Jak system zareaguje, gdy zalogowanych będzie bardzo dużo użytkowników w tym samym czasie • Jak system zareaguje gdy wprowadzone zostanie bardzo dużo danych do bazy danych w tym samym czasie • Sprawdzenie jak zareaguje system gdy baza danych odmowi dostępu gdy będzie o niego wnioskował
Oracles:	• Użycie narzędzia umożliwiającego symulacje ogromnej liczby użytkowników próbujących dostać się do platformy w tym samym czasie • Zaprogramowanie wirtualnych użytkowników w taki sposób żeby wykonywali typowe czynności które wykonywać będzie normalny użytkownik w trakcie używania produktu • Liczba użytkowników będzie wciąż powiększana do póki strona nie będzie już w stanie przyjmować większej liczby i nie będzie już dostępna do użycia (nastapi crash)
Required Tools:	• Narzędzie automatyzujące testy • Narzędzie do monitorowania instalacji • Narzędzie umożliwiające tworzenia dużej ilości wirtualnych użytkowników • Narzędzie do generowania danych
Success Criteria:	Wygenerowanie raportu na podstawie obserwacji przed i po udanej symulacji sytuacji stresowej dla systemu.

Special Considerations:	• Aby zasymulować sytuacje stresowa dla systemu możliwa jest potrzeba polaczenia z Internetem • Potrzebne jest silne narzędzie umożliwiające generowanie dużej ilości obciążenia • Należy tymczasowo zmniejszyć dostępną pamięć systemu na czas przeprowadzania testów

5.2.6 Volume Testing

Technique Objective:	• Maksymalna ilość użytkowników połączonych z systemem, dokonujących tych samych działań • Sprawdzenie jak system radzi sobie, gdy maksymalny rozmiar bazy danych został osiągnięty i następuje ciągłe dodawanie do niego nowych danych • Sprawdzenie w jakim momencie system będzie na skraju stabilności
Technique:	• Maksymalna wielkość bazy danych została osiągnięta i następują zapytania od wielu użytkowników jednocześnie • Powinna zostać użyta duża ilość użytkowników w celu dokonywania tych samych czynności, aby zasymulować duży ruch • Sprawdzić czas, w którym użytkownik uzyska odpowiedz na zapytanie • Sprawdzić czy jakieś dane zostają utracone • Sprawdzić czy w związku z dużą ilością zapytań wszystkie informacje na stronie wyświetlane są poprawnie • Sprawdzić czy wyświetlane są komunikaty o błędzie w trakcie występowania dużej ilości zapytań • Sprawdzić czy czas odpowiedzi na zapytanie do bazy danych jest akceptowalny
Oracles:	• Dodanie dużej ilości danych do bazy w jednym czasie i zobaczenie reakcji • Zalogowanie w jednym czasie ponad 10000 użytkowników
Required Tools:	Narzędzie automatyzacji skryptow - HammerDB Narzedzie monitorowania instalacji Narzędzie ograniczające dostęp do zasobow Narzedzie generujące dane
Success Criteria:	Wygenerowanie raportu na podstawie obserwacji przed i po udanej symulacji sytuacji dużej ilości danych i overloadingu dla systemu.
Special Considerations:	• Należy użyć narzedzia pozawalającego na generacje duzej ilosci zapytań • Możliwa jest konieczność dostepu do internetu

5.2.7 Security and Access Control Testing

Technique Objective:	Bezpieczeństwo na poziomie aplikacji – aktor może posiadac dostep tylko do danych do których jest upoważniony Bezpieczeństwo na poziomie systemu – tylko aktorzy posiadający dostep do systemu maja możliwość jego używania
Technique:	• Stworzyć testy dla kazdego typu użytkownika i sprawdzic ich dostęp do każdej z czynności dostępnych dla danej grupy użytkowników • Zmodyfikować typ użytkownika i przeprowadzić testy ponownie • Sprawdzenie czy system zaakceptuje niezidentyfikowanego użytkownika • Sprawdzić czy wszystkie pola do wprowadzania tekstu mają zdefiniowane miary tak, aby zapobiec atakom SQL Injection
Oracles:	• Automatyczne skanowanie systemu w poszukiwaniu znanych już typowych podatności podobnych systemów • Symulacja ataku hackerskiego i zobaczenie w jaki sposób zareaguje system • Sprawdzić czy możliwe jest zapytanie do bazy danych o hasła użytkowników i czy jeżeli tak, to czy je zwraca • Sprawdzić czy możliwy jest atak brute force, po wprowadzeniu hasła 3 razy błędnego powinien zostać zablokowany dostęp do ponownego wprowadzenia hasła • Wprowadzić tekst ze skryptem w pole umożliwiające wprowadzenie tekstu i zobaczyć reakcje systemu
Required Tools:	• Narzędzia automatyzacji skryptow • Narzędzia hakerskie np. maszyna RedHat • OWASP
Success Criteria:	Przetestowanie wszystkich aktorów na możliwość nieupoważnionego dostępu i sprawdzenie wszystkich funkcji i danych mogących ulec modyfikacji na skutek złego zabezpieczenia
Special Considerations:	Moze byc niemozliwe przeporwadzenie testow sieciowych zwiazane z brakiem akceptacji administratora sieci

5.2.8 Failover and Recovery Testing

Technique Objective:	Symulacja awarii i procesu odzyskiwania bazy danych, aplikacji i systemu do poprzedniego, zdrowego stanu. • Awaria ze strony klienta • Awaria ze strony serwera • Awaria komunikacji ze strony serwerów internetowych • Niekompletne/przerwane cykle • Nieprawidłowe lub zniszczone dane w bazie danych
Technique:	• Wyłączenie jednostki klienta (np. komputer) • Symulacja wyłączenia serwera • Symulacja przerwania łączności internetowej • Przywołanie procedur odzyskiwania systemu •
Oracles:	• W trakcie otrzymywania danych przez siec internetową przez aplikację odłączyć kabel od sieci internetowej • Po jakimś czasie podłączyć go ponownie i przeanalizować możliwość kontynuacji pobierania danych od momentu przerwania połączenia z Internetem
Required Tools:	• Narzędzie monitorujące instalacje • Narzędzie do backupów
Success Criteria:	Technika pozwala na symulacje awarii i odzyskiwania systemu związanych z aplikacją, systemem i bazami danych
Special Considerations:	• Testy powinny zostać przeprowadzone na wyizolowanych maszynach • Odłączenie od internetu może być problematyczne, należy w takim przypadku przeprowadzić symulacje na maszynie wirtualnej gdzie możliwe jest odłączenie łączności z internetem

5.2.9 Configuration Testing

Technique Objective:	Sprawdzenie zachowania aplikacji w różnych konfiguracjach, mające na celu ustalenie w jakiej pracuje najlepiej zapewniając zastosowanie się do wymagań funkcjonalnych programu

Technique:	• Analiza wydajnosci systemu po zmniejszeniu lub zwiększeniu ilości pamięci, podlaczeniu roznych urządzeń itp. • Użycie skryptów testów funkcjonalnych • Otworzenie dużej ilości programów innych niż testowany, np. Word z duża ilością tekstu lub Excel z duża ilością danych
Oracles:	Użycie wirtualnych maszyn aby zmniejszyć czas potrzebny na zbadanie wszystkich konfiguracji systemowych. Wirtualna maszyna może pozwolić na symulacje specyficznej prawdziwej konfiguracji systemowej, zamiast instalacji i dezinstalacji wielu różnych konfiguracji systemowych.
Required Tools:	• Narzędzie do monitorowania instalacji • Narzędzie do tworzenia backupów • Narzędzie do przywracania konfiguracji • Wirtualne maszyny
Success Criteria:	Technika pozwala na przetestowanie jednej lub większej ilości kombinacji testów w zaplanowanych i skonfigurowanych środowiskach
Special Considerations:	• Potrzebne są takie narzędzia jak np. Word czy Excel z dostępną dużą ilością danych • Potrzebna jest wirtualna maszyna np. VirtualBox

5.2.10 Installation Testing

Technique Objective:	Przeporwadzenie testow w zaleznosci od stanu maszyny: • nowa instalacja – maszyna nie była nigdy instalowana • update – maszyna była wcześniej zainstalowana, ta sama wersja • update – maszyna była wcześniej zainstalowana, starsza wersja

Technique:	• Sprawdzenie czy wyświetla się odpowiednia wiadomość podczas instalacji starszej lub najnowszej wersji produktu • Sprawdzenie jak zachowa się aplikacja przy próbie instalacji przy niewystarczającej ilości pamięci na dysku • Przyciski GUI odpowiadające za dezinstalacje lub naprawę powinny być widoczne podczas kolejnego uruchomienia aplikacji już po jej pierwszym zainstalowaniu • Sprawdzenie czy po dezinstalacji programu wszelkie pliki po nim zostaną usunięte z systemu • Sprawdzić, czy instalacja działa prawidłowo na wielu maszynach w tym samym czasie • Sprawdzić kompletność plików po instalacji •
Oracles:	• Mając zainstalowaną starszą wersję systemu, zainstalować nowszą wersję • Stworzyć automatyczne skrypty do procesów instalacji i dezinstalacji aplikacji • Zmniejszyć dostępną pamięć dysku i spróbować zainstalować aplikację • Użyć środowiska w którym przeprowadzone zostanie zainstalowanie aplikacji w tym samym czasie na wielu maszynach • Zatrzymać instalację przed końcem i kontynuować ją
Required Tools:	• Narzędzie monitorujące instalację • RSpec
Success Criteria:	Technika pozwala na sprawdzenie przebiegu instalacji produktu na więcej niż jeden sposób i na różnych jej fazach
Special Considerations:	Należy się upewnić, że instalacja przebiegła pomyślnie, to znaczy wszystkie pliki zostały pomyślnie zainstalowane i nie pominięto żadnego

6. Kryteria wejścia i wyjścia

6.1 Plan testów

6.1.1 Kryteria wejscia

- Kod jest ukończony lub jest prawie ukończony i możliwe jest jego przetestowanie
- Utworzone są przypadki testowe
- Wymagania są zdefiniowane i zaakceptowane
- Dostępna jest wystarczająca ilość materiału do testowania
- Środowisko zostało zaimplementowane
- Bugi znalezione podczas unit testów zostały naprawione
- Środowisko testowe jest stabilne

6.1.2 Kryteria wyjscia

- Wyczerpanie budżetu
- Koniec czasu przeznaczonego na projekt

- Przeprowadzono wystarczającą ilość testów funkcjonalności systemu
- Wszytskie poważne bugi zostały naprawione
- Testy objętościowe, wydajnościowe i stress testing przebiegły pomyślnie
- System jest kompatybilny z danym softwarem i hardwarem

6.1.3 Kryteria wstrzymania i kontynuacji testów

Kryteria wstrzymania testowania:

- Poważny defekt, który ma duży wpływ na dalsze postępy testów
- Niedostępność hardware lub software
- Brak dostępności odpowiednich zasobów do przeprowadzenia testów
- Harmonogram projektu (inne priorytety)
- Różnice w działaniach funkcjonalności aplikacji i założeniach funkcjonalnych

Kryteria kontynuacji testowania:

- Problem, przez który nastąpiło wstrzymanie testowania został rozwiązany
- Hardware i software są dostępne jak przewidziano w wymaganiach
- W trakcie dalszych badań nie znaleziono poważnych błędów uniemożliwiających dalsze testowanie

6.2 Cykle testów

6.2.1 Kryteria wejscia dla cykli testów

- Dokumentacja wymagań funkcjonalnych i niefunkcjonalnych jest dostępna
- Zdefiniowanie kryteriów akceptacji ryzyka
- Dostępna jest dokumentacja architektury aplikacji
- Plan testów jest utworzony
- Środowisko testowe jest utworzone
- Rezultaty testów są dostępne

6.2.2 Kryteria wyjścia dla cykli testów

- Zaakceptowany dokument strategii testów
- Zaakceptowane skrypty testów
- Środowisko testowe jest gotowe i działa, jak powinno
- Smoke test przebiegł pomyślnie
- Wszystkie zaplanowane testy zostały przeprowadzone

6.2.3 Nieplanowane zakończenie testów

- Środowisko testowe zostało przygotowane w sposób nieprawidłowy lub nie jest gotowe do testów
- Brak dokumentacji wymagań funkcjonalnych i niefunkcjonalnych
- Nieprawidłowości w planie testów
- Testy nie zostały przeprowadzone pomyślnie

7. Potrzeby środowiskowe

7.1 Hardware

System Resources		
Resource	**Quantity**	**Name and Type**
Database Server		
Network or Subnet		TBD
Server Name		TBD
Database Name		TBD
Client Test PCs		
Include special configuration requirements		TBD
Test Repository		
Network or Subnet		TBD
Server Name		TBD
Test Development PCs		TBD

7.2 Software

Software Element Name	Version	Type and Other Notes
Widnows 10		Operating System
Windows 11		Operating System
Google Chrome		Internet Browser
Mozilla Firefox		Internet Browser
MS Outlook		eMail Client software
Network Associates McAfee Virus Checker		Virus Detection and Recovery Software

8. Potrzeby kadrowe

Zaleca się, aby przynajmniej sześć testerów na pełnym wymiarze pracy było przydzielonych do prac nad systemem oraz testami akceptacyjnymi. Testerzy ci będą przydzieleni na pół etatu wstępnych etapach prac projektowych, aby udzielali się w ocenach jakości. Planowo, po czterech miesiącach testerzy zostaną przydzieleni do prac na pełny etat. Jeśli dany tester nie będzie dostępny, kierownik projektu bądź kierownik testów przejmie jego obowiązki. Reszta osób na stanowiskach wymienionych w podpunkcie 10.1 również w fazach wstępnych projektu planowana jest być zatrudniona w niepełnym wymiarze pracy, natomiast po czterech miesiącach zatrudniona na pełny etat.

Szacuje się, że płynność kadrowa będzie niska, dlatego nie planuje się dodatkowych zatrudnień. W razie odejścia któregoś z członków zespołu jego obowiązki będą rozdysponowane pozostałym członkom.

9. Kamienie milowe

Milestone	Planned Start Date	Actual Start Date	Planned End Date	Actual End Date
Iteration Plan agreed	01.10.22	01.10.22	01.10.22	01.10.22
Iteration starts	03.10.22	03.10.22	06.12.22	06.12.22
Requirements baselined	04.10.22	04.10.22	04.10.22	04.10.22
Architecture baselined	05.10.22	05.10.22	05.10.22	06.10.22
User Interface baselined	07.10.22	07.10.22	07.10.22	07.10.22
First Build delivered to test	15.10.22	15.10.22	15.10.22	15.10.22
First Build accepted into test	16.10.22	16.10.22	16.10.22	16.10.22
First Build test cycle finishes	18.10.22	18.10.22	22.10.22	22.10.22
[Build Two will not be tested]	23.10.22	23.10.22	03.11.22	03.11.22
Third Build delivered to test	12.11.22	12.11.22	12.11.22	12.11.22
Third Build accepted into test	13.11.22	13.11.22	13.11.22	13.11.22
Third Build test cycle finishes	16.11.22	16.11.22	18.11.22	18.11.22
Fourth Build delivered to test	27.11.22	27.11.22	27.11.22	27.11.22
Fourth Build accepted into test	28.11.22	28.11.22	28.11.22	28.11.22
Iteration Assessment review	30.11.22	30.11.22	03.12.22	04.12.22
Iteration ends	06.12.22	06.12.22	06.12.22	06.12.22

10. Proces i procedury zarządzania

10.1 Assessing the Deliverables of this Test Plan

Produkty dostarczane planu testów muszą zostać przejrzane i zaakceptowane przez kierownika projektu po skonsultowaniu się z klientem. Kryteriami do spełnienia są zgodność ze specyfikacją, budżetem i ograniczeniami czasowymi.

10.2 Problem Reporting, Escalation, and Issue Resolution

Wszelkie problemy będą zgłaszane przez dedykowane oprogramowanie wraz ze stopniem pilności i osobami odpowiedzialnymi za zażegnanie problemu. Następnie będą po kolei rozwiązywane biorąc pod uwagę stopień pilności. Problemy będą rozwiązywane metodami adekwatnymi do natury problemu.

10.3 Managing Test Cycles

- Sprawdzenie czy czynności mieszczą się w planowanym czasie
- Sprawdzenie, czy wszystkie potrzebne zasoby są dostępne
- Wprowadzenie poprawek w zależności od sytuacji projektu
- Zadbanie o wydajność zespołu zgodną z wymaganą wydajnością

10.4 Approval and Signoff

Plan testów zostanie przekazany Kierownikowi Projektu i sprawdzony pod kątem poprawności i szczegółowości a następnie przez niego zatwierdzony, jeśli plan testów spełnia wymagane kryteria.