**TESTOWANIE OPROGRAMOWANIA**

**PROJEKT ZALICZENIOWY - RAPORT**

|  |  |
| --- | --- |
| Data rozpoczęcia prac | 7.05.16 |
| Data oddania raportu | 14.06.16 |

1. **Zespół**

|  |  |
| --- | --- |
| Nr grupy |  |
| Nazwisko i imię prowadzącego grupę | Goik Damian |

Członkowie zespołu:

|  |  |  |
| --- | --- | --- |
| Lp. | Nazwisko | Imię |
| 1 | Domański | Damian |
| 2 | Majchrzak | Aleksandra |
| 3 |  |  |

1. **Testowana aplikacja (5p)**

*opisać nazwę i wersję aplikacji, scharakteryzować aplikację (typ, przeznaczenie aplikacji, architektura aplikacji, komponenty, wykorzystana technologia), przedstawić zalety i wady aplikacji z punktu widzenia testowania (np. testowalności, kontrolowalności, możliwości automatyzacji itp.); ocenie podlega dokładność, kompletność i czytelność opisu aplikacji oraz merytoryka opisu testowalności*

Kalendarz Google (wersja zgodna z przeglądarką- 50.0)– aplikacja webowa umożliwiająca m.in. tworzenie wielu instancji kalendarza i planowanie spotkań/ wydarzeń/ ustawianie przypomnień, z przypisaniem ich do konkretnych terminów w kalendarzu użytkownika. Powiązana jest z kontem gmail danego użytkownika, ale może on również korzystać z kalendarzy udostępnionych przez innych użytkowników oraz sam udostępniać swoje kalendarze czy też zapraszać inne osoby do swoich wydarzeń. Aplikacja umożliwia personalizację widoku kalendarza oraz import danych z plików .vcs (pliki Microsoft Outlook), .isc (iCalendar) i .csv.

Główne komponenty aplikacji to: ekran personalizacji (umożliwia edycję ustawień dla całego konta takich jak język, strefa czasowa, domyślny widok kalendarza), manager kalendarzy (dodawania, usuwanie, edycja, udostęnianie, personalizacja), manager wydarzeń (o podobnych funkcjonalnościach jak poprzedni). Aplikacja posiada tez moduł Laboratorium zawierający funkcjonalności w stanie beta testów, jednak tymczasowo nie będziemy się w niego zagłębiać.

Wszystkie dane przechowywane są w chmurze (jest również możliwość korzystania offline, ale służy do tego oddzielna aplikacja), więc dane dostępne są na dowolnym urządzeniu z dostępem do internetu.

Od strony back-endowej aplikacja napisana jest w Javie, natomiast front-end stworzony jest w Javascripcie i korzysta z technologii AJAX (zmiany są automatycznie wyświetlane na stronie bez konieczności jej przeładowywania).

Możliwe tylko testy czarnoskrzynkowe- brak dostępu do kodu źródłowego. Zważywszy na bogatą dokumentację, która opisuje dokładnie popularne scenariusze użycia, testowalność jest dość wysoka. Jako, że dostęp mamy tylko do warstwy interfejsu, kontrolowalność zapewniona jest w stopniu podstawowym (ciężej jest zasymulować awarie powodowane przez czynniki zewnętrzne). Aplikacja jest obserwowalna – wszystkie akcje użytkownika powodują zmiany/komunikaty prezentowane w GUI, nieprawidłowości można szybko wychwycić. Jako, że zmiany w kodzie są kontrolowane przez producenta, aplikacja jest raczej stabilna.

Testy automatyczne sprawdzą się raczej pod kątem badania wydajności (dodawanie dużej ilości danych w tym samym czasie), współbieżności, czy wartości brzegowych we wprowadzanych danych.

1. **Analiza podstawy testów (6p)**

*opisać istniejące dokumenty pozwalające na analizę aplikacji, jej funkcjonalności, sposobu działania, powiązań z innymi, zewnętrznymi aplikacjami/systemami; a także: repozytoria, istniejące testy jednostkowe, kwestie istnienia wyroczni testowej; opisać przydatność tych dokumentów oraz w jakim celu każdy z nich będzie wykorzystany; ocenie podlega szczegółowość opisu, umiejętność i trafność analizy podstawy testów*

Pod adresem <https://support.google.com/calendar> dostępna jest ogólna dokumentacja funkcjonalna aplikacji. Na jej podstawie określone zostały wymagania środowiskowe (współpraca z konkretnymi systemami operacyjnymi, przeglądarkami internetowymi czy innymi programami komunikacyjnymi), a także zdefiniowane konkretne scenariusze testowe oparte o funkcjonalności opisane w niej - głównie określenie odpowiedniej konfiguracji, podziału na moduły, podstawowych przypadków użycia.

Zgodnie z tym dokumentem, aplikacja powinna zostać przetestowana na trzech najnowszych wersjach przeglądarek: Google Chrome, Internet Explorer (oprócz Windows XP), Firefox i Safari na systemach Windows, Linux, Mac. Działanie aplikacji zależne jest też od aktywowania JavaScriptu i ciasteczek w przeglądarce.

W dokumencie zostało też opisane importowanie danych z innych aplikacji takich jak iCalendar czy MS Outlook oraz plików .cdv. Informacje te wykorzystane zostaną wykorzystane w testach inicjujących dane kalendarza oraz sprawdzających zdolność synchronizacji ze wspomnianymi aplikacjami.

Dokumentacja zawiera również scenariusze opisujące, zmiany w ustawieniach aplikacji, modyfikacji kalendarzy/wydarzeń/powiadomień oraz zasad współdzielenia elementów z innymi użytkownikami.

Aplikacja nie jest open source’owa, zatem nie są też dostępne dla niej testy jednostkowe.

Pod adresem <https://developers.google.com/google-apps/calendar/> dostępne jest jednak API aplikacji, jako, że jest ona otwarta na rozszerzenia i integracje z innymi projektami.

API może okazać się przydatne przy testach automatycznych sprawdzających głównie dodawanie/ edycję wydarzeń/ powiadomień do kalendarza

Jako, że aplikacja jest dość rozpowszechniona, a jej działanie intuicyjne i nie wymagające specjalistycznej wiedzy, wyrocznią testową może być sam tester.

1. **Plan testów (12p)**  
   *ocenie podlegają kwestie merytoryczne (np. zasadność, umiejętność uzasadnienia zaplanowanych podejść, czynności, stosowanych technik i narzędzi), a także stopień zdolności analitycznych zespołu, dokładność, wnikliwość, umiejętność sensownego planowania itp.*
   1. wybór podejścia do testowania (strategii testowej) wraz z uzasadnieniem

Z uwagi na fakt, iż przedmiotem naszego zainteresowania jest oprogramowanie już wdrożone na środowisku produkcyjnym strategie testów będą miały charakter reaktywny. Wśród zastosowanych metod znajdą się:

* + metody dynamiczne i heurystyczne – wykonane zostaną testy eksploracyjne, podczas których zostaną poddane w ocenę elementy tj. przyjazność interfejsu użytkownika oraz intuicyjność serwisu,
  + metody bazujące na modelach – na podstawie statystyk użycia zostaną określone, kluczowe funkcjonalności systemu, które w miarę możliwości będą pokryte testami automatycznymi,
  + metody systematyczne - biorąc pod uwagę listę znanych błędów w oprogramowaniu podobnego typu/przeznaczenia jak i znanych błędów w wykorzystywanych zależnościach opracowane zostaną testy kontrolne,
  + metody analityczne – ponieważ testowany produkt może przechowywać dane wrażliwe wykonane zostanę bardzo podstawowe (z racji niskiego poziomu wiedzy jak i nieznanych polityk producenta oprogramowania) testy bezpieczeństwa.
  1. zaplanowane poziomy testowania wraz z uzasadnieniem

Zaplanowane poziomy testowania:

* testy akceptacyjne – grupą docelową rozwiązania jest szerokie grono odbiorców,
* testy modułowe (API) - jednym z założeń produktu jest łatwość integracji z aplikacjami zewnętrznymi, dodatkowo testy modułowe łatwo zautomatyzować,
* testy systemowe – oprogramowanie wspiera kilka przeglądarek na kilku systemach operacyjnych oraz kilka rodzajów urządzeń - konieczna jest walidacja choćby sposobu wyświetlania na różnych urządzeniach.
  1. zaplanowane typy testów wraz z uzasadnieniem

Planowane typy testów:

a) manualne:

* eksploracyjne – pomocne w nauce systemu przez zespół testerski, ponadto intuicje testerów mogą pomóc w wykryciu błędów niewykrywalnych innymi metodami,
* bezpieczeństwa - system przechowuje dane wrażliwe,
* użyteczności - testy wykonywane ze względu na dużą liczbę użytkowników, bardzo ciężkie do zautomatyzowania,

b) automatyczne:

* funkcjonalne - dostępność głównych funkcjonalności powinna być testowana możliwie często, przydatne również w wykrywaniu regresji,
* modułowe - łatwe do zautomatyzowania, w przypadku produktu już wdrożonego stosunkowo mała zmienność interfejsów, pomagają również wykryć regresję.
  1. metryki wykorzystane w procesie testowym

Metryki wykorzystywane w procesie zarządzania testami:

* pwt - postęp wykonania testów - uproszczona forma w postaci ilorazu ilości wykonanych przypadków testowych do ogólnej liczby przygotowanych przypadków testowych,
* ppt - postęp przygotowania testów - ze względu na ograniczone zasoby metryka przyjmuje postać ilorazu czasu poświęconego na przygotowanie danego typu testów przez zaplanowaną ilość czasu na przygotowanie danego typu testów w harmonogramie,
  1. obszary aplikacji podlegające testom (przedmiot testów, elementy testowe)

W ramach testowanej aplikacji wyróżniamy następujące obszary testowania funkcjonalności:

* kalendarze,
* wydarzenia,
* powiadomienia,
* widoki (zgodność layout-ów dla różnych środowisk).
  1. obszary aplikacji nie testowane (uzasadnić dlaczego)

Obszary nietestowane:

* + wydajność - brak znajomości zasad nieautoryzowanych testów wydajności + polityki producenta odnośnie bot-ów i fałszywie generowanego ruchu,
  + funkcjonalności z działu laboratorium – ograniczone zasoby ludzkie,
  + wersja mobilna aplikacji – ograniczone zasoby ludzkie,
  + wersja offline – ograniczone zasoby ludzkie,
  + bezpieczeństwo - brak specjalistycznej wiedzy z zakresu bezpieczeństwa aplikacji web-owych uniemożliwił przeprowadzenie testów na poziomie wyższym niż elementarny.
  1. opis środowiska testowego, niezbędnych uprzęży testowych, osprzętu itd.

Głównymi cechami charakterystycznymi środowisk testowych są (w kolejności od najważniejszych):

* przeglądarka internetowa (nazwa + wersja),
* system operacyjny (nazwa + wersja),
* rozmiar wyświetlacza (szerokość x wysokość),
* urządzenia wejścia (mysz, klawiatura, touchpad, touchscreen, inne?).

Podczas wykonywania testów konieczne jest podanie możliwie najdokładniejszego opisu środowiska (min. cechy wyżej wymienione, możliwe podanie dodatkowych informacji uważanych za istotne wg testera).

* 1. analiza ryzyka, priorytetyzacja testów

Zidentyfikowane ryzyka:

* brak dostępności kluczowych funkcjonalności tj. wyświetlanie nieodległych wydarzeń, brak możliwości dodania wydarzenia – utrata głównej funkcjonalności produktu, duże straty wizerunkowe, możliwe zniechęcenie użytkowników do oprogramowania w przypadku zbyt częstych problemów,
* wyciek danych użytkowników - duże straty wizerunkowe, możliwe wykorzystanie ujawnionych danych na szkodę użytkownika,
* trudność zachowania kompatybilności z wieloma urządzeniami/programami (problemy inne niż niedostępność usług, np.: złe wyświetlanie treści, formatek) - mało istotne, w dłuższej perspektywie mogą spowodować utratę użytkowników.

Powyższe zidentyfikowane ryzyka przyczyniły się do ustalenia poniższych priorytetów.

Testy wg priorytetu:

* testy funkcjonalne,
* testy bezpieczeństwa,
* testy kompatybilności ze wspieranymi przeglądarkami, OS-ami, urządzeniami,
* pozostałe.
  1. harmonogram prac

1. Testowanie eksploracyjne – 2 roboczodni.
2. Przegląd dokumentacji – 2 roboczodni.
3. Implementacja testów modułowych (API) - 3 roboczodni.
4. Przygotowanie przypadków testowych do testów funkcjonalnych – 2 roboczodni.
5. Implementacja automatycznych testów funkcjonalnych - 2 roboczodni.
6. Wykonanie pozostałych testów funkcjonalnych oraz testów systemowych + przygotowanie środowisk testowych – 2 roboczodni.
7. Wykonanie checklist-y znanych błędów - 1 roboczodzień.
8. Wykonanie testów do checklist-y – 1 roboczodzień.
9. Wykonanie podstawowych testów bezpieczeństwa - 1 roboczodzień.

Łącznie: 16 roboczodni.

* 1. inne uwagi

1. **Projektowanie testów (12p)**

*opisać poziom ogólności przypadków testowych dla poszczególnych obszarów testowania; jakie techniki projektowania testów zostały zastosowane i wobec jakich fragmentów aplikacji; jeśli zastosowano testowanie oparte na modelu, opisać modele i sposób generacji przypadków testowych z tych modeli; wypisać przypadki/scenariusze testowe, historyjki użytkownika itp.; ocenie podlega: kreatywność, umiejętność posługiwania się różnymi technikami projektowania testów, zasadność stosowania określonych technik, umiejętność argumentacji na rzecz stosowania lub też nie stosowania określonych metod i podejść, umiejętność czytelnego przedstawienia projektów testów/przypadków testowych, umiejętność wykorzystania różnego rodzaju modeli do przedstawienia testów, identyfikacji warunków testowych, elementów pokrycia itp., umiejętne i zasadne przedstawienie organizacji testów w suity testowe, różnorodność zaprojektowanych testów i ich zdolność do wykrywania awarii, brak "redundantnych" testów itp.*

# **Przypadki testowe:**

1. **Kalendarze**

**A1. Dodanie kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Wejście w *Ustawienia*.
2. Przejście do zakładki *Kalendarze*.
3. Kliknięcie *Utwórz nowy kalendarz*.
4. Wypełnienie formatki nowego kalendarza.
5. Kliknięcie *Utwórz kalendarz*.

**Oczekiwany efekt:**

* dodanie nowego kalendarza.

**A2. Anulowanie subskrypcji kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Wejście w *Ustawienia*.
2. Przejście do zakładki *Kalendarze*.
3. Kliknięcie *Anuluj subskrypcje*  przy odpowiednim kalendarzu.
4. Potwierdzenie anulowania.

**Oczekiwany efekt:**

* anulowanie subskrypcji wybranego kalendarza.

**A3. Edytowanie danych kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Wybranie *Ustawienia kalendarza* z rozwijanej listy przy nazwie konkretnego kalendarza.
2. Zmiana nazwy kalendarza i opisu.
3. Kliknięcie *Zapisz*.

**Oczekiwany efekt:**

* zmieniona nazwa i opis wybranego kalendarza.

**A4. Próba dodania kalendarza bez nazwy**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Wejście w *Ustawienia*.
2. Przejście do zakładki *Kalendarze*.
3. Kliknięcie *Utwórz nowy kalendarz*.
4. Kliknięcie *Utwórz kalendarz*.

**Oczekiwany efekt:**

* komunikat o braku nazwy.

**A5. Dodania kalendarza ze znakami specjalnymi w nazwie**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Wejście w *Ustawienia*.
2. Przejście do zakładki *Kalendarze*.
3. Kliknięcie *Utwórz nowy kalendarz*.
4. Wpisanie !@#$%^&\*()\_+`~~ w nazwie kalendarza.
5. Kliknięcie *Utwórz kalendarz*.

**Oczekiwany efekt:**

* dodanie kalendarza o podanej nazwie.

**A6. Usunięcie kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.
* użytkownik posiada co najmniej jeden kalendarz.

**Kroki:**

1. Wejście w *Ustawienia* danego kalendarza.
2. Przejście do zakładki Usuwanie *kalendarza*.
3. Kliknięcie *Usuń wszystkie wydarzenia z tego kalendarza*.
4. Zaznaczenie opcji *Tak, z całą pewnością chcę usunąć wszystkie wydarzenia ze swojego kalendarza.*
5. Kliknięcie *Usuń wszystkie wydarzenia.*

**Oczekiwany efekt:**

* dany kalendarz został usunięty

**A7. Eksport kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.
* użytkownik posiada co najmniej jeden kalendarz.

**Kroki:**

1. Wejście w *Ustawienia* danego kalendarza.
2. Przejście do zakładki *Eksportowanie kalendarza*.
3. Kliknięcie *Eksportuj ten kalendarz*.

**Oczekiwany efekt:**

* do domyślnej lokalizacji eksportowany jest plik .ICS (być może skompresowany) z danymi kalendarza

**A8. Import kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.
* użytkownik posiada co najmniej jeden kalendarz.
* użytkownik w wybranej lokalizacji posiada plik .ICS z danymi kalendarza.

**Kroki:**

1. Wejście w *Ustawienia*.
2. Przejście do zakładki *Kalendarze*.
3. Kliknięcie *Importuj kalendarz*.
4. Kliknięcie *Przeglądaj*.
5. Wybór odpowiedniego pliku .ICS.
6. Kliknięcie *Otwórz*.
7. Wybór z rozwijanej listy kalendarza, do którego wydarzenia mają zostać zaimportowane.

**Oczekiwany efekt:**

* wyświetlony został komunikat o ilości przetworzonych wydarzeń
* dane wydarzenia zostały dodane do wybranego kalendarza

1. **Powiadomienia**

**B1. Dodanie powiadomienia do kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Wybranie Edytuj powiadomienia z rozwijanej listy przy nazwie konkretnego kalendarza.
2. Kliknięcie *Dodaj powiadomienie.*
3. Kliknięcie *Zapisz*.

**Oczekiwany efekt:**

* dodanie powiadomienia.

**B2. Usuwanie powiadomienia z kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Wybranie Edytuj powiadomienia z rozwijanej listy przy nazwie konkretnego kalendarza.
2. Kliknięcie *usuń* po prawej stronie od usuwanego powiadomienia*.*
3. Kliknięcie *Zapisz*.

**Oczekiwany efekt:**

* usunięcie powiadomienia.

1. **Wydarzenia**

**C1. Dodanie wydarzenia do kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Kliknięcie *Utwórz.*
2. Wypełnienie formatki wydarzenia.
3. Kliknięcie *Zapisz*.

**Oczekiwany efekt:**

* dodanie wydarzenia.

**C2. Usuwanie wydarzenia z kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany,
* kalendarz z co najmniej jednym wydarzeniem.

**Kroki:**

1. Kliknięcie na dolną krawędź wydarzenia.
2. Kliknięcie *Usuń na zawsze.*

**Oczekiwany efekt:**

* usunięcie wybranego wydarzenia z kalendarza.

**C3. Edytowanie wydarzenia ze swojego kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany,
* co najmniej jedno wydarzenie w kalendarzu użytkownika.

**Kroki:**

1. Kliknięcie w dolną krawędź wydarzenia.
2. Kliknięcie *Edytuj wydarzenie.*
3. Zmiana nazwy wydarzenia oraz czasu.
4. Kliknięcie *Zapisz*.

**Oczekiwany efekt:**

* zmiana wybranego wydarzenia.

**C4. Dodanie wydarzenia z datą końca przed datą początku**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Kliknięcie *Utwórz*.
2. Zmiana godziny zakończenia na godzinę przed początkiem.
3. Zmiana daty zakończenia na datę wcześniejszą lub równą dacie początku.
4. Kliknięcie *Zapisz*.
5. Zmiana godziny zakończenia na godzinę po początku.
6. Zmiana godziny zakończenia na godzinę przed początkiem.
7. Zmiana daty zakończenia na datę wcześniejszą lub równą dacie początku.
8. Kliknięcie *Zapisz*.

**Oczekiwany efekt:**

* komunikat o nieprawidłowym czasie zakończenia względem czasu początku wydarzenia.

**C5. Dodanie wydarzenia bez daty końca do kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Kliknięcie *Utwórz*.
2. Skasowanie daty końca wydarzenia.
3. Kliknięcie *Zapisz.*

**Oczekiwany efekt:**

* komunikat o braku daty końca wydarzenia lub dodanie wydarzenia o nieograniczonym czasie.

**C6. Dodanie wydarzenia cyklicznego do kalendarza**

**Warunki początkowe:**

* użytkownik zalogowany.

**Kroki:**

1. Kliknięcie *Utwórz*.
2. Zaznaczenie *Powtarzaj...* .
3. Ustawienie powtarzania wydarzenia co 2 dni przez miesiąc.
4. Kliknięcie *Gotowe.*
5. Kliknięcie *Zapisz*.

**Oczekiwany efekt:**

* dodanie wydarzenia co 2 dni przez cały miesiąc.

1. **Implementacja testów (7p)**

*w przypadku testów automatycznych opisać podejście do automatyzacji; załączyć do raportu wszystkie testalia związane z implementacją: procedury testowe, skrypty, kody źródłowe, generatory danych, implementację komparatorów i wyroczni, zaślepki, sterowniki, konfigurację sprzętu, środowiska i narzędzi, instalacje i konfiguracje niezbędnych baz danych, raporty (w tym wygenerowane z użytych narzędzi) itp.; opisać organizację testów w suity testowe (jeśli to konieczne); ocenie podlega profesjonalizm tworzenia kodu, zasadność jego tworzenia, czytelność*

Testy automatyczne przeprowadzone zostały na przeglądarkach:

* Google Chrome (wersja 51.0)
* Mozilla Firefox (wersja 44.0)
* Internet Explorer (wersja 11)

Implementacja przeprowadzona została w języku Java (JDK 1.8) na IDE IntelliJ IDEA w wersji 1016.1.3 z wykorzystaniem biblioteki Selenium ( <http://www.seleniumhq.org/download/> ) w wersji 2.53, biblioteki TestNG ( <http://testng.org/doc/download.html> ) w wersji 6.8 oraz przy wykorzystaniu frameworku Maven.

Selenium łączy się z przeglądarkami za pomocą mechanizmu WebDriver, więc do jej działania potrzebne są odpowiednio:

* do współpracy z Google Chrome – ChromeDriver ( <https://sites.google.com/a/chromium.org/chromedriver/> )
* do współpracy z Mozilla Firefox – FirefoxDriver ( wyarczy pobrać plugin <https://addons.mozilla.org/pl/firefox/addon/selenium-ide/> )
* do współpracy z Internet Explorer – IEDriver/server ( [https://github.com/SeleniumHQ/selenium/wiki/InternetExplorerDriver#required-configuration](https://github.com/SeleniumHQ/selenium/wiki/InternetExplorerDriver" \l "required-configuration) – wymagana dodatkowa konfiguracja opisana na stronie)

Selenium może mieć problemy ze współpracą z najnowszą wersją przeglądarki Firefox, więc najbezpieczniej jest korzystać ze starszej wersji oraz dezaktywować opcję automatycznej aktualizacji.

W kwestii początkowej konfiguracji przeglądarek, aby ułatwić testowanie eksportowania kalendarzy w przeglądarce Google Chrome I Mozilla Firefox należy ustawić opcję automatycznego zapisywania plików w danej lokalizacji. W przeglądarce Internet Explorer nie ma takiej możliwości, więc testy eksportowania kalendarza nie będą na niej działać.

Przy testach importowania kalendarza z pliku na każdej z przeglądarek podnosi się okno wyboru lokalizacji. Jest ono obsługiwane z poziomu systemu operacyjnego, więc do wprowadzania lokalizacji została wykorzystana biblioteka Robot, która do swojego działania wymaga, by program miał focus na konkretnym polu tekstowym. Zatem testy importowania kalendarza nie mogą odbywać się w tle (pozostałe testy mogą).

Wszystkie parametry konfiguracyjne projektu testowego znajdują się w pliku testng.xml (konfiguracja wymagana dla biblioteki TestNG- zawiera podział na suity, testy, klasy testowe, grupy, definiuje potrzebne parametry)

Projekt testowy podzielony jest na 3 główne pakiety: dataProviders, pages oraz tests.

Klasy zawarte w pakiecie dataProviders dostarczają danych do testów automatycznych (dla każdego modułu testowania stworzona jest osobna klasa providera).

W pakiecie pages znajdują się klasy odpowiadające za widoki poszczególnych stron testowanej aplikacji (wzorzec projektowy Page Object Pattern).

W pakiecie tests znajdują się zaś klasy odpowiadające bezpośrednio za wywołanie testów danych modułów.

1. **Wykonanie testów, raport z przebiegu testów (8p, punktowane łącznie z 8)**

*opisać przebieg wykonania testów, z wykorzystaniem metryk zdefiniowanych w punkcie 4.4. (np. liczba zaprojektowanych, wykonanych, zawieszonych testów, liczba znalezionych defektów, pokrycie funkcjonalności, ryzyk, kodu, czas projektowania, implementacji, wykonania testów itp.); ocenie podlega umiejętność raportowania i analizy danych (np. metryk, raportów z wykorzystanych narzędzi)*

Utworzonych zostało w sumie 9 ogólnych testów realizujących 10 przypadków testowych.

Metryka pwt: 9/16.

Pokryte zostały przypadki użycia:

* dodawanie/ usuwanie nowego kalendarza,
* dodanie kalendarza ze specjalnymi znakami w nazwie,
* edycja danych istniejącego kalendarza,
* import/eksport kalendarza do pliku ICS,
* dodanie nowego wydarzenia (w sposób standardowy i szybki) z poprawnymi danymi oraz błędnymi,
* dodanie wydarzenia cyklicznego,
* usunięcie wydarzenia (w sposób szybki).

Testy funkcjonalności powiadomień musiały zostać zawieszone ze względu na niewystarczającą ilość czasu.

Czas poświęcony na implementację testów wyniósł 4 roboczodni (względem 2 dni zaplanowanych w harmonogramie).

Poszczególne testy nie sprawdziły się tak samo na wszystkich przeglądarkach.

Wszystkie testy przeszły jedynie na przeglądarce Google Chrome.

Na przeglądarce Mozilla Firefox nie sprawdziły się testy szybkiego dodawania/ usuwania wydarzeń, zaś na przeglądarce Internet Explorer nie zadziałały testy eksportu kalendarza (ze względu na trudności w automatyzacji wyboru formy przetworzenia pliku) oraz szybkiego dodawania/usuwania wydarzeń.

Błędy w testach spowodowane były najprawdopodobniej różnicami w automatycznie generowanym kodzie stron dla każdej z przeglądarek. Narzędzia do analizy kodu nie są na tym samym poziomie dla testowanych przeglądarek, więc w niektórych przypadkach niemal uniemożliwiało to odpowiednią interakcję testów automatycznych z przeglądarką.

Kontrola poprawności testów również była dość ograniczona ze względu na bazowanie tylko i wyłącznie na graficznych elementach aplikacji. A co za tym idzie: aplikacja nie uniemożliwia dodania dwóch kalendarzy/ wydarzeń o tej samej nazwie zatem aby móc kontrolować obecność danych obiektów poprzez obecność ich reprezentacji graficznej na stronie, w testach założyć trzeba było, że nie istnieją dwa kalendarze/wydarzenia o jednakowej nazwie.

Problematyczna okazała się też konfiguracja sterowników. Dla Mozilla Firefox odpowiednie wersji przeglądarki i Selenium, dla Internet Explorer niedziałający sterownik w wersji 64- bitowej. Przy dalszym utrzymaniu testów powyższe problemy mogą okazać się dość poważne (stała kontrola wersji oprogramowania, budowanie odpowiednich przypadków testowych.

Zauważalne były również (oprócz bezpośrednich wyników testów) różnice w prędkości działania poszczególnych testów na różnych przeglądarkach.

1. **Raport o incydentach/awariach/defektach**

*opisać jakie incydenty/awarie/defekty zostały znalezione w formie raportów o defektach, które będą umożliwiać reprodukcję błędu; w razie konieczności załączyć również screenshoty, wykorzystane dane testowe itp.; ocenie podlega profesjonalizm w raportowaniu incydentów/awarii/defektów (kompletność informacji pomocna np. w reprodukcji defektu, niesprzeczność, dokładność, klarowność itp.)*

Podczas testów eksploracyjnych zostało stworzonych 13 przypadków testowych. Odpowiadają one głównym funkcjonalnościom testowanego systemu. Przy ich pomocy odnaleziono 3 defekty o niskim priorytecie:

1) Podczas wykonywania przypadku C4 po pierwszej próbie wprowadzenia i zaakceptowania nieprawidłowych danych pojawił się stosowny komunikat o braku możliwości utworzenia wydarzenia z czasem zakończenia przed czasem rozpoczęcia, ale po drugiej próbie w sposób opisany w przypadku udało się zapisać wydarzenie (natomiast zapisało się z ostatnim poprawnie wprowadzonym czasem zakończenia - innym niż wyświetlanym w chwili kliknięcia *Zapisz*). Wynik jest różny od oczekiwanego.

Pożądanym efektem jest powtórne wyświetlanie stosownego komunikatu do momentu próby zapisu poprawnych danych.

Uwaga: test wykonywany był ok. 18:00 w związku z czym ustawienie czasu zakończenia na godzinę wcześniej nie zmieniało daty zakończenia.

2) Test C5 nie przechodzi, ponieważ udało się zapisać wydarzenie po skasowaniu daty końca wydarzenia - zostało zapisane z ostatnią dobrze wprowadzoną datą.

Pożądanym wynikiem jest wyświetlenie komunikatu o braku dacie zakończenia wydarzenia lub dodanie wydarzenia bez czasu końca.

3) Podczas wykonywania przypadku testowego C6 zauważony został defekt polegający na tym, że użytkownik wybiera *Powtarzanie: Codziennie* co jest mylące biorąc pod uwagę, iż w następnym kroku określa *Powtarzaj co 2 dni*. Codziennie wg słownika języka polskiego znaczy to samo co każdego dnia.

Pożądanym wynikiem jest wyświetlenie użytkownikowi niesprzecznego interfejsu.

W ramach testów bezpieczeństwa zostały wykonane 2 typy testów w wersji bardzo podstawowej, które znajdują się w grupie OWASP TOP 10:

1) SQL Injection - próba doklejenia ' OR '1'='1 do nazwy tworzonego wydarzenia zakończyła się stworzeniem wydarzenia o takiej nazwie. Wniosek z tego taki, że w tym są dobrze escape-owane znaki składni SQL.

2) XSS - próba umieszczenia na stronie skyptu javascript-owego poprzez dodanie kalendarza o nazwie <script>alert(123)</script> również zakończyła się dodaniem kalendarza o takiej nazwie, więc to pole również jest dobrze escape-owane.

**Środowiska testowe**

1) Win10 + Edge 25 + touchpad + klawiatura + 1366x768

2) Win10 + Chrome 51 + touchpad + klawiatura + 1366x768

1. **Uwagi**

*uwagi i komentarze zespołu dotyczące aspektów nie występujących w punktach 1-8*

...