Adrian Borek

ETAP II

ZADANIE 1: Testy

ID	1.0
Tytuł	Definiowanie danych dostępu do bazy danych
Wynik	Nie wykryto błędów

ID	1.1
Tytuł	Definiowanie danych dostępu do bazy danych z niepoprawnym hasłem
Wynik	Nie wykryto błędów

ID	1.2
Tytuł	Definiowanie danych dostępu do bazy danych ze złym numerem portu
Wynik	Nie wykryto błędów

ID	1.3
Tytuł	Definiowanie danych dostępu z wprowadzeniem nieistniejącej nazwy bazy danych
Wynik	Nie wykryto błędów

ID	2.1
Tytuł	Logowanie do aplikacji przy użyciu prawidłowych danych
Wynik	Nie wykryto błędów

ID	2.2
Tytuł	Logowanie do aplikacji z wykorzystaniem prawidłowego loginu i błędnego hasła
Wynik	Wykryto błędy: *3-krotnie błędnie wpisane hasło nie blokuje konta użytkownika*

ID	2.3
Tytuł	Rezygnacja z procedury logowania
Wynik	Nie wykryto błędów

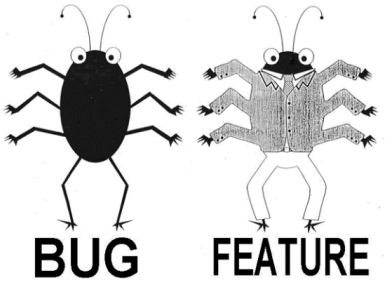
ZADANIE 2: Raportowanie błędów

ID błędu	ID przypadku	Nazwa błędu	Obszar występowania	Priorytet	Scenariusz reprodukcji	Stan obecny	Stan pożądany
1	2.2	3-krotnie błędnie wpisane hasło nie blokuje konta użytkownika	Formularz logowania	Wysoki	▪ uruchom aplikację; ▪ w formularzu logowania wpisz 3-krotnie błędne hasło; ▪ spróbuj zalogować się, podając prawidłowe dane.	Użytkownik zostaje zalogowany, mimo wprowadzenia 3-krotnie błędnego hasła	Po 3-krotnie nieudanej próbie logowania powinno nastąpić zablokowanie użytkownika

Odpowiedzi na pytania

1. **Co oznaczają pojęcia *Gray Box Testing* oraz *Glass Box Testing*?**
   Testy szaroskrzynkowe to połączenie testów białoskrzynkowych i testów czarnoskrzynkowych. Celem takiego testowania jest poszukiwanie ewentualnych błędów, które powstają na skutek nieprawidłowego użytkowania lub niewłaściwej struktury aplikacji.
   Testy szklanej skrzynki to metody testowania oprogramowania polegające na sprawdzaniu struktury programu oraz uzyskiwaniu testowych danych z logiki programu. Testy strukturalne są czasami określane jako clear-box testing, odkąd białe skrzynki zostały uznane za nieprzezroczyste ze względu na brak możliwości podglądu kodu.

2. **Różnica między błędem a funkcjonalnością.**
   Błędy obniżają wartość oprogramowania, sprawiając kłopoty w użytkowaniu, natomiast „ficzery" wzbogacają jego funkcjonalność.



3. **Czy można w 100% przetestować oprogramowanie?**
   Nie można w pełni przetestować oprogramowania, ponieważ człowiek nie jest w stanie wyłapać wszystkich błędów. Można jedynie minimalizować ryzyko ich wystąpienia.
   Bywa także tak, że w procesie testowania występują pewne znane ograniczenia takie jak wyznaczone ramy czasowe, czy założony na ten cel budżet.

4. **Kiedy można zakończyć testowanie?**
   Najlepszym wyjściem byłoby utrzymanie stanu nieustannego testowania oprogramowania, w celu wykluczenia jak największej ilości błędów. Niestety, taki wybór byłby nieracjonalny, ponieważ oprogramowanie można by testować w nieskończoność, a błędy nadal byłyby wykrywane, co pochłonęłoby ogromną ilość zasobów. Najlepszym wyjściem byłoby ustalenie odpowiedniego procentu przypadków testowych, których wykonanie nie spowodowało wykrycia błędu, a pokrycie kodu, funkcjonalności, wymagań zostałoby zrealizowane w założonym zakresie. Ponadto osoba odpowiedzialna uznałaby, że krzywa wykrywania błędów plasuje się poniżej założonego wcześniej progu i zdecydowałaby, że oprogramowanie jest już wystarczająco funkcjonalne i można je przekazać użytkownikowi końcowemu.