Laurea Magistrale in informatica
Università degli Studi di Salerno

Master Test Plan

QuantuMoonLight

Team

Matteo Cicalese - 0522501516
Luca Contrasto – 0522500104

Repo

github.com/CicaMatt/QML-IGES
github.com/Robertales/QuantuMoonLight

Sommario

1. Approccio di test

L'obiettivo dei test sviluppati sarà di garantire la qualità del prodotto e verificare la corretta funzionalità dello stesso in rapporto alle change request realizzate. L'approccio utilizzato per il testing sarà di tipo Bottom-up, dunque si effettuerà il testing e l'integrazione delle singole componenti partendo dalle funzionalità atomiche della piattaforma.
Come per i precedenti test implementati, faremo uso di *PyUnit* per testing di unità e di sistema, insieme con *Coverage.py* per l'analisi della code coverage e la valutazione dell'efficacia dei test.
Per l'attività di testing si prevede un branch coverage pari almeno al 75%, per rispettare il requisito definito in fase di sviluppo iniziale della piattaforma.

2. Strategie di test

Di seguito sono descritte le considerazioni effettuate e le operazioni condotte per lo sviluppo delle varie tipologie di testing:

Testing di unità

Il testing di unità sarà condotto sulla base delle funzionalità sviluppate nel corso delle change request, dunque cercheremo, nel modo più esaustivo possibile, di testare tutte le funzionalità aggiunte e modificate in maniera **white-box** tramite valutazione della code coverage. Cercheremo di coprire i branch scoperti, selezionando input appropriati per raggiungerli.

Testing di integrazione

Per il testing di integrazione valuteremo le combinazioni di interazione tra le funzionalità e i moduli chiave della piattaforma, anch'essi testati in maniera **white-box**, per valutare il risultato della fruizione combinata degli stessi.

Testing di sistema

Per il testing di sistema, in modo analogo a come veniva fatto precedentemente alle modifiche, la strategia presa in considerazione è il **category partitioning**. Pertanto, suddividiamo i campi dei dati in ingresso in classi di casi di test, identifichiamo i parametri con le possibili scelte associate e poi eseguiamo i test per verificare la presenza di errori. Considerando l'approccio scelto per il testing di sistema, quest'ultimo avverrà in maniera black-box.

3. Target del testing

Le attività di test prevederanno le seguenti attività:

- Creazione dei casi di test
- Esecuzione dei test
- Report dei risultati
- Gestione dei fallimenti

4. Classi testate

Le classi sottoposte a testing saranno le seguenti:

- Il metodo *experimentDownlaod* in *UtenteControl*, tramite gli unit test *test_download* e *test_Download_invalidPath* in *test_UtenteControl*;
- I metodi *encrypt* e *decrypt* in *encryption*, rispettivamente tramite gli unit test *test_encrypt*, *test_decrypt*, *test_encrypt_fail* e *test_decrypt_fail*;
- I metodi *resetPW* e *SetNewPW* in *UtenteControl*, tramite gli unit test *test_SetNewPW*, *test_sendCode*, *test_sendCode_emailNotFoundError* ed il test *test_email_sending_failure* in *test_GestioneControl*.

La funzionalità del download sarà inoltre testata tramite la test suite di sistema contenuta in *test_download* e analogamente la funzionalità del reset della password sarà testata a livello di sistema tramite la test suite *test_password_reset*.

Per quanto riguarda la change request CR_1, relativa all'aggiornamento di alcune librerie e della versione di Python, essa verrà testata tramite il testing di sistema, per verificare eventuali errori e problemi introdotto, a causa di deprecazioni e nuove funzionalità.

5. Classi/Package esclusi dai Test

- Il package *blog,* in quanto riguarda una funzionalità di peso minore nel contesto del progetto
- Il package *utils,* in quanto contiene metodi banali di generale utilità, prettamente relativi ad operazioni su *DataFrame*, utilizzati in singole occorrenze
- Il package *model*, in quanto contiene il codice di DDL necessario a *SQLAlchemy* per popolare il database, le cui operazioni a livello pratico sono totalmente gestite dalla libreria

6. Branch Coverage

La coverage, successivamente all'aggiunta e modifica dei casi di test necessari per renderli coerenti con le modifiche apportate al progetto in passato, risulta essere del 78%.

Ci poniamo come obbiettivo che, successivamente al testing delle change request, la coverage rimanga invariata o superiore.