DocuExchange

Test Case Plan Document

DocuExchange Unisa Informatica

Presentato da:

Emanuele Bruno
Giacomo Impronta
Michele de Rosa
Paolo Erra

Revision History

Data	Versione	Cambiamenti	Autori
30/12/19	1.0	Strutturazione documento	Impronta Giacomo

Sommario

1.Introduzione

Il testing è una componente fondamentale nello sviluppo di un software. Permette infatti di testare le componenti del sistema rilevando, in maniera controllata, eventuali errori in modo da poterli correggere ed offrire un prodotto senza errori rilevanti/critici. In questo documento andremo a mostrare una panoramica dei test da effettuare sul software "DocuExchange". Introdurremo le strategie di testing utilizzate; cosa sarà testato del software e perché. Verranno testati tutti i requisiti con priorità **alta** contenuti nelle varie gestioni del sistema. Le gestioni presenti nel sistema sono:

1. Gestione Utente
2. Gestione appunti per lo studente
3. Gestione Admin

2. Documenti correlati

2.1 Correlazione con lo Statement of Work(SOW)

Nello SOW, documento in cui vengono definiti gli accordi tra i componenti del team e da questi ultimi con i clienti, si è fissata una soglia molto alta di Branch Coverage (per garantire un testing ottimale del sistema) del 75%. Qualora la soglia minima prefissata non sarà raggiunta il testing dovrà considerarsi assolutamente fallimentare. Il documento in cui sono contenute le suddette informazioni è il seguente:

NC_7_SOW.

2.2 Correlazione con il Requirement Analysis Document (RAD)

Nel RAD vengono definite tutte le gestioni presenti nel sistema e i requisiti funzionali e non funzionali. I requisiti funzionali con una priorità **alta** presenti nel RAD saranno testati durante la fase di testing. Il documento in cui sono contenute le suddette informazioni è il seguente:

NC_7_RAD_V_0.12.

2.3 Correlazione con il System Design Document (SDD)

Nel SDD vengono definiti tutti gli obiettivi del sistema e del relativo design. Nella definizione dei suddetti obiettivi saranno presenti vincoli da rispettare anche durante la fase di testing. Il documento in cui sono contenute le suddette informazioni è il seguente:

NC_7_SDD_V_0.7.

3. Panoramica del sistema

Il nostro sistema (come specificato nel SDD) segue un modello MVC (Model-Control-View). Come accennato in precedenza il sistema può essere suddiviso in vari sottosistemi, ognuno con molteplici funzionalità:

1. **Gestione Utente**
    - **a.** Effettua Registrazione
    - **b.** Accesso Utente
    - **c.** Modifica Profilo
    - **d.** Cancellazione Utente
    - **e.** Recupero Password
    - **f.** Ricerca studente
    - **g.** Visualizza Profilo

2. **Gestione appunti per lo studente**
    - **a.** Invio Richiesta Pubblicazione Appunti
    - **b.** Ricerca Appunti
    - **c.** Scarica Appunti
    - **d.** Cancella Appunti
    - **e.** Valuta Appunti
3. **Gestione Admin**
    - **a.** Visualizza Studenti
    - **b.** Visualizza Richieste
    - **c.** Ricerca Richiesta
    - **d.** Verifica Richieste di Pubblicazione (Pubblica Appunti)

4. Funzionalità da testare e da non testare

Tra le funzionalità del sistema, andremo a testare quelle con priorità **alta.** Le funzionalità che saranno testate saranno quindi, per ogni sottosistema, le seguenti:

1. **Gestione Utente**
    - **a.** Effettua Registrazione
    - **b.** Accesso Utente
    - **c.** Ricerca studente
    - **d.** Visualizza Profilo

2. **Gestione appunti per lo studente**
    - **a.** Invio Richiesta Pubblicazione Appunti
    - **b.** Ricerca Appunti
    - **c.** Scarica Appunti
    - **d.** Valuta Appunti
3. **Gestione Admin**
    - **a.** Visualizza Studenti

b. Visualizza Richieste
c. Ricerca Richiesta
d. Verifica Richieste di Pubblicazione (Pubblica Appunti)

5. Criteri pass/failed

Il testing ha lo scopo di scovare eventuali fault e failure nel sistema. Si deduce quindi che il testing ha successo quando viene riscontrata una failure oppure un fault; se tutto va come dovrebbe il testing avrà fallito. Nel caso in cui il testing abbia successo si interviene per risolvere il problema e successivamente si ricontrolla la parte modificata ripetendo i test. Nel caso in cui i test ripetuti portino ad un fallimento allora l'errore è stato corretto con successo.

6. Approccio

Per ottenere un testing ottimale ed esaustivo suddivideremo il tutto in 3 diverse fasi:

- testing di unità;
- testing di integrazione;
- testing di sistema;

Nel testing di unità andremo a testare tutte le singole componenti del sistema, poi in quello di integrazione andremo a testare tutti i sottosistemi del sistema. Infine nel testing di sistema andremo a testare tutto il sistema nel complesso.

6.1 Testing di unità

Per testare tutte le singole componenti del sistema utilizzeremo le tecniche di whitebox e blackbox. Per quanto riguarda il blackbox testing andremo a testare tutte le possibili soluzione offerte dalla tecnica del category partition. Per il whitebox testing testeremo le strutture interne dei singoli componenti. Utilizzeremo la tecnica del Branch Coverage cercando di ottenere una copertura di testing pari almeno al 75% rispetto al totale.

6.2 Testing di integrazione

Per il testing di integrazione avremo una suddivisione delle varie componenti in sottosistemi del sistema. La tecnica adottata per testare i vari sottosistemi è la "Bottom-up".

6.3 Testing di sistema

Per il testing di sistema viene testato il sistema nel complesso generale, testando tutti i requisiti funzionali e non. Verrà testata l'efficienza del sistema e come riesce a soddisfare le richieste per il quale è stato progettato.

7. Sospensione e ripresa

7.1 Sospensione

Nel caso in cui si arrivasse ad un punto in cui il testing è esaustivo e rispetta i limiti minimi prefissati, il testing può essere sospeso

7.2 Ripresa

Nel caso in cui dovessero essere aggiunte ulteriori funzionalità verrà ripresa l'attività di testing per testare le nuove componenti e raggiungere nuovamente la soglia minima di branch coverage.

8. Materiale per il testing

Per effettuare correttamente il testing abbiamo bisogno di materiale specifico, come un DBMS per gestire il database; un server in locale per far funzionare il sistema; ed un client-web per effettuare richieste al server e testare le funzionalità. Per effettuare testing withebox utilizzeremo il tool JUNIT. Per effettuare il testing blackbox utilizzeremo il tool SELENIUM.

9. Glossario

DBMS: database managment system

JUNIT: tool per testing withebox

SELENIUM: tool per testing blackbox