Università degli Studi di Salerno
Corso di Ingegneria del Software

Test Plan
Versione 2.0

Data 23/01/19

Partecipanti:

Nome	Matricola
Michela Giovanna Scarpone	0512104490
Raffaela Romano	0512103774

Scritto da:	Scarpone/Romano

Revision History

Data	Versione	Descrizione	Autore
15/01/19	1.0	Stesura test plan	Scarpone/Romano
23/01/19	2.0	Ultima revisione test plan	Scarpone/Romano

Indice

1. Introduzione

Lo scopo di questo documento è di pianificare l'attività di test del sistema LaSaporita al fine di verificare se esistono differenze tra il comportamento atteso e il comportamento osservato.

In questa attività andremo a rilevare gli eventuali errori prodotti all'interno del codice, per evitare che essi si presentino nel momento in cui il sistema verrà utilizzato dall'utente finale.

Le attività di test sono state pianificate per le seguenti gestioni:

- Gestione Autenticazione
- Gestione Registrazione
- Gestione Prodotti

I risultati di questi test verranno utilizzati per capire dove intervenire, e quindi correggere eventuali errori o apportare modifiche per il miglioramento dei vari sottosistemi.

2. Documenti correlati

Il test plan ha ovviamente una stretta relazione con il resto dei documenti che sono stati prodotti finora, poiché prima di passare alla fase di testing, oltre ad aver implementato il sistema nella gran parte, esso era stato pianificato nelle precedenti documentazioni.

Questo quindi permette di rilevare le eventuali differenze tra ciò che si desiderava e ciò che invece il sistema fa.

Di seguito verranno riportate le relazioni tra il test plan e la documentazione precedente.

2.1 Relazioni con il documento di analisi dei requisiti (RAD)

La relazione tra test plan e RAD riguarda in particolare i requisiti funzionali e non funzionali del sistema poiché i test che saranno eseguiti su ogni funzionalità terranno conto delle specifiche espresse nel RAD.

2.2 Relazioni con il System Design Document (SDD)

Nel System Design Document abbiamo suddiviso il nostro sistema in sottosistemi e l'architettura in tre livelli: Presentation Layer, Application Layer e Storage Layer.

Il test dei vari componenti deve rimanere fedele a queste suddivisioni il più possibile.

2.3 Relazioni con l'Object Design Document (ODD)

Il test d'integrazione farà quanto più riferimento possibile alle class interfaces definite nell'ODD.

3. Panoramica del sistema

Come stabilito nel System Design Document la struttura del nostro sistema è divisa secondo un'architettura "Three Tiers" cioè a tre livelli: Presentation Layer, Application Layer, Storage Layer.

In questo caso il livello più alto interagisce con il livello applicativo che a sua volta si occuperà di eseguire le operazioni nel database de LaSaporita, cercando di garantire il più possibile basso accoppiamento e alta coesione tra le varie classi.

Il sistema inoltre è stato suddiviso in sottosistemi più piccoli, in particolare è stato diviso per gestioni.

Abbiamo infatti i seguenti sottosistemi:

- Gestione Autenticazione
- Gestione Registrazione
- Gestione Prodotti
- Gestione Utenti
- Gestione Ordini

Quasi ognuna delle precedenti gestioni prevede principalmente operazioni di inserimento, modifica, cancellazione, visualizzazione e ricerca e saranno proprio queste funzionalità ad essere testate nel corso della fase di testing del sistema.

4. Funzionalità <u>da testare</u> e da non testare

Di seguito saranno elencate per ogni gestione quali sono le funzionalità che saranno testate.

1. Gestione Autenticazione;
    - <u>Login</u>
    - Logout
2. Gestione Registrazione;
    - <u>Registrazione</u>
3. Gestione Prodotti
    - <u>Aggiunta Manuale</u>
    - Elimina Prodotto
    - Visualizza Prodotto

4.Gestione Utenti

- Visualizza Utenti
- Elimina Utenti

5.Gestione Ordini

- Visualizza Ordini
- Elimina Ordini

5. Criteri Pass/Failed

I dati di input del test saranno suddivisi in classi di equivalenza, ovvero verranno raggruppati in insiemi dalle caratteristiche comuni, per i quali sarà sufficiente testare un solo elemento rappresentativo.

Un input avrà superato un test se l'output risultante sarà quello atteso, cioè quello che è stato specificato dal membro del team che si occuperà del testing su tale test case, il responsabile del testing conosce quale dovrebbe essere l'output corretto.

6. Approccio

Le tecniche di testing adottate riguarderanno inizialmente il testing di unità dei singoli componenti, in modo da testare nello specifico la correttezza di ciascuna unità.

Seguirà il testing d'integrazione, che focalizzerà l'attenzione principalmente sul test delle interfacce delle suddette unità.

Infine verrà eseguito il testing di sistema, che vedrà come oggetto di testing l'intero sistema assemblato nei suoi componenti. Quest'ultimo servirà soprattutto a verificare che il sistema soddisfi le richieste del committente

6.1 Testing di Unità

Durante questa fase, verranno ricercate le condizioni di fallimento isolando i componenti ed usando test driver e stub, cioè implementazioni parziali di componenti che dipendono o da cui dipendono le componenti da testare.

La strategia utilizzata per il testing si baserà esclusivamente sulla tecnica Black-Box, che si focalizza sul comportamento Input/Output, ignorando la struttura interna della componente.

Al fine di minimizzare il numero di test cases, i possibili input verranno partizionati in classi di equivalenza e per ogni classe verrà selezionato un test case.

Gli stati erronei scovati in questa, come in qualsiasi altra fase di testing, che comporteranno un fallimento del sistema dovranno essere tempestivamente comunicati agli sviluppatori al fine di correggerli e ripristinare il testing al più presto, per comunicare i vari fallimenti identificati durante la fase di testing si utilizzerà un test incident report.

6.2 Testing d'integrazione

In questa fase si procederà all'integrazione delle componenti di una funzionalità che verranno testate nel complesso attraverso una strategia Bottom-Up. Si passerà, poi, alla funzionalità successiva fino ad esaurire le funzionalità implementate.

Quest'approccio mira principalmente a ridurre le dipendenze tra funzionalità differenti e a facilitare la ricerca di errori nelle interfacce di comunicazione tra sottosistemi.

6.3 Testing di sistema

Lo scopo di questa fase di testing è quello di dimostrare che il sistema soddisfi effettivamente i requisiti richiesti e sia, quindi, pronto all'uso.

Come per il testing di unità, si cercherà di testare le funzionalità più importanti per l'utente e quelle che hanno una maggiore probabilità di fallimento.

Si noti che, come per il testing di unità, si procederà attraverso tecnica Black-Box.

7. Sospensione e ripresa

7.1 Criteri di sospensione

La fase di testing del sistema verrà sospesa quando si raggiungerà un compromesso tra qualità del prodotto e costi dell'attività di testing.

Il testing verrà quindi portato avanti quanto più possibile nel tempo senza però rischiare di ritardare la consegna finale del progetto.

7.2 Criteri di ripresa

In seguito ad ogni modifica o correzione delle componenti che genereranno errori o fallimenti, i test case verranno sottoposti nuovamente al sistema assicurandosi così di aver risolto effettivamente il problema.

8. Materiale per il testing

L'hardware necessario per l'attività di test è un pc con connessione ad internet, dato che il database del sistema è stato caricato online per renderlo disponibili a tutti.

9. Test Cases

Gestione Autenticazione

Login

Parametro: Username **Formato**: [A-Z a-z 0-9]	
Lunghezza[LU]	· NULL[error] · <=20 [proprety LunghezzaLU OK]

Parametro: Password **Formato**: [A-Z a-z 0-9]	
Lunghezza[LP]	· NULL [error] · <=20[proprety lunghezzaLP OK]

Codice	Combinazione	Esito
TC_0.1.1	LU	Errore
TC_0.1.2	LP	Errore
TC_0.1.3	LU,LP	Errore
TC_0.1.4	LU,LP	Successo

Gestione Registrazione

Parametro: Username **Formato**: [A-Z a-z 0-9]	
Lunghezza[LU]	· NULL[error] · <=20 [proprety LunghezzaLU OK]

Parametro: Password **Formato**: [A-Z a-z 0-9]	
Lunghezza[LP]	· >10 [error] · <=20[proprety lunghezzaLP OK]

Parametro: Nome **Formato**: [A-Z a-z 0-9]	
Lunghezza[LN]	· NULL[error] · <=20 [proprety LunghezzaLU OK]
Parametro: Cognome **Formato**: [A-Z a-z 0-9]	
Lunghezza[LC]	· NULL[error] · <=20 [proprety LunghezzaLU OK]

Codice	**Combinazione**	**Esito**
TC_1.1.1	LU	Errore
TC_1.1.2	LU,LP	Errore
TC_1.1.3	LU,LP,LN	Errore
TC_1.1.4	LU,LP,LN,LC	Errore
TC_1.1.5	LU,LP,LN,LC	Successo

Gestione Prodotti

Aggiunta prodotto

Parametro: Codice **Formato**: [0-9]	
Formato[FC]	· NULL[error] · [Formato FC Intero OK]

Parametro: Nome **Formato**: [A-Z a-z 0-9]	
Lunghezza[LNP]	· NULL[error] · <=20 [LunghezzaLNP OK]

Parametro: Prezzo **Formato**: [0-10 €]	
Formato[FZ]	· Formato not intero[error] · <=10[Formato FZ OK]

Parametro: Ingredienti **Formato**: [A-Z a-z 0-9]	
Lunghezza[LI]	· NULL[error] · <=100 [LunghezzaLI OK]

Parametro: Tipo **Formato**: [A-Z a-z 0-9]	
Tipologia[TT]	· NULL[error] · <=4 [Tipologia OK]

Codice	Combinazione	Esito
TC_2.2.1	FC	Errore
TC_2.2.2	FC,LNP	Errore
TC_2.2.3	FC,LNP,TT	Errore
TC_2.2.4	FC,LNP,TT,FZ	Errore
TC_2.2.5	FC,LNP,TT,FZ,LI	Errore
TC_2.2.6	FC,LNP,TT,FZ,LI	Successo

10. Pianificazione del Testing

Il team per il testing deve essere composto da persone che hanno una completa e approfondita conoscenza del sistema e delle tecniche di testing con i documenti associati, quali Test plan e Test case specification.

Tali tecniche devono essere applicate nei tempi, nel budget e nei vincoli di qualità stabiliti.

Solitamente i componenti del team di testing non sono coinvolti nella fase di sviluppo, in quanto non deve esistere una reticenza nel decomporre il sistema. Il team dedicato al controllo della qualità è responsabile dell'attività di testing e quindi della ricerca di fault. La documentazione dei fault trovati è inviata agli sviluppatori per consentire la correzione del sistema.

Il sistema revisionato è poi testato nuovamente non solo per verificare se gli errori trovati in precedenza sono stati eliminati ma soprattutto per verificare che non ne siano stati introdotti dei nuovi.

L'attività di testing è fondamentale nello sviluppo di un sistema software in quanto la mancanza di tale attività o una cattiva interpretazione di essa può portare al completo fallimento del sistema. Data l'importanza del testing ne risulta fondamentale la schedulazione.

10.1 Determinazione dei rischi

I rischi di un completo fallimento verranno minimizzati effettuando una pianificazione verticale delle attività di testing funzionale. Questo permetterà in caso di ritardi, dovuti ad una grande quantità di failure trovati, di rilasciare meno funzionalità del previsto, ma completamente testate. Inoltre tale pianificazione ridurrà notevolmente la produzione di driver e stub, evitando l'introduzione di nuovi errori, dovuti all'implementazione di tali componenti.

10.2 Organizzazione delle attività di testing

Le attività di testing verranno organizzate secondo uno schema che effettuerà una divisione funzionale di tipo verticale. In questo modo al termine di ogni attività si avrà una funzionalità completamente testata nei suoi livelli gerarchici.

I vantaggi principali sono che in caso di ritardi dovuti al ritrovamento di numerosi failure il sistema verrà rilasciato con meno componenti, ma interamente testate e funzionanti.