Università degli Studi di Salerno
Corso di Ingegneria del Software

Pop!x
Test Plan
Versione 2.1

Data: 16/12/2024

Coordinatore del progetto:

Nome	Matricola
Scaparra Daniele Pio	0512116260

Partecipanti:

Nome	Matricola
Scaparra Daniele Pio	0512116260
Bonagura Grazia	0512116167
Nappi Antonio	0512117391
Nardiello Raffaele	0512118666

Scritto da:	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele

Revision History

Data	Versione	Descrizione	Autore
16/12/2024	1.0	Prima versione del Test Plan	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele
24/12/2024	1.1	Modificato sezione 3, 4, 5, 6, 8 del documento	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele
24/12/2024	1.2	Modificata sezione 7	Nardiello Raffaele
05/01/2025	2.0	Versione di rilascio test plan	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele
07/01/2025	2.1	Ritocchi finali	Scaparra Daniele Pio

Outline

1. Introduzione

Nel documento di Test Plan, vengono descritte le strategie di testing adottate e come esse si collegano alle documentazioni precedenti, come il RAD, SDD e ODD.
L'obiettivo principale è fornire un quadro chiaro che permetta di testare correttamente tutte le funzionalità del sistema.
Viene definito anche l'approccio del piano di test, che stabilisce come verranno suddivisi i diversi tipi di test, con particolare attenzione alla copertura completa di ogni componente e alle eventuali dipendenze tra i moduli.
I test includeranno sia scenari standard che situazioni eccezionali, affrontando errori comuni e comportamenti anomali per garantire un sistema stabile e sicuro.

2. Relazione con gli altri documenti

Come già anticipato nell'introduzione, il documento è in relazione con i documenti di SDD, RAD e ODD.

- **Relazione con il Requirements Analysis Document**: Il Test Plan utilizza i requisiti specificati nel RAD per pianificare e progettare i test, assicurandosi che tutti i requisiti siano coperti. Gli scenari di test si basano sui requisiti per verificare la loro validità.
- **Relazione con il System Design Document**: Il Test Plan integra informazioni dal SDD per creare test che riflettano la struttura del sistema e le sue interazioni. I test verificano l'aderenza alla progettazione del sistema.
- **Relazione con l'Object Design Document**: Il Test Plan utilizza le specifiche dell'ODD per creare test che valutano le interfacce tra le classi e i moduli, verificando che le relazioni e le dipendenze siano correttamente gestite.

3. Panoramica del sistema

Il sistema oggetto di test è un'applicazione e-commerce progettata per essere eseguita su un server Apache Tomcat 9 in ambiente Windows 11. L'architettura è basata su Java e include componenti di back-end implementati con servlet e logica applicativa, e un front-end sviluppato utilizzando JSP, HTML, CSS e Bootstrap.

Caratteristiche principali del sistema:

- **Gestione degli utenti:** Funzionalità di registrazione, autenticazione e gestione dei profili utente.
- **Catalogo prodotti:** Visualizzazione e gestione di un catalogo di prodotti con opzioni di filtraggio e ricerca.
- **Carrello e checkout:** Possibilità di aggiungere, rimuovere e modificare articoli nel carrello, completando gli acquisti tramite il modulo di checkout.
- **Dashboard amministrativa:** Accesso riservato agli amministratori per la gestione di prodotti.

L'obiettivo principale del sistema è fornire un'esperienza utente funzionale e fluida, con particolare attenzione alla stabilità delle funzionalità principali come carrello, autenticazione e checkout.

-

4. Caratteristiche da testare/da non testare

Saranno testate, per motivi di ottimizzazione di costi e risorse, solamente alcune funzionalità del sito, qui riportate di seguito.

- **Utente Guest**:
  - *Registrazione*
- **Utente registrato**:
  - *Autenticazione*
- **Catalogo**:
  - *Aggiunta di un nuovo prodotto*
  - *Modifica di un prodotto già esistente*
  - *Rimozione di un prodotto esistente*
- **Carrello**:
  - *Aggiunta di un articolo al carrello.*
  - *Rimozione di un articolo alla volta dal carrello*
  - *Modifica della quantità selezionata di un prodotto presente nel carrello*
- **Ordine**:
  - *Checkout*
- **Gestore ordine**:
  - *Cambio stato ordine*

5. Criteri di successo e fallimento

Nel contesto della verifica e validazione del sistema, i **criteri di fallimento e successo** definiscono le condizioni necessarie per determinare se un test, o l'intero sistema sottoposto a verifica, può essere considerato superato (**pass**) o fallito (**fail**). Questi criteri sono fondamentali per garantire che il sistema soddisfi i

requisiti funzionali, di esperienza utente e di prestazioni, rispettando le aspettative progettuali e gli standard di qualità.

Successo

Un test è considerato riuscito se il sistema sotto test (SUT) produce i risultati attesi, soddisfacendo i requisiti specificati.

Fallimento

Un test è considerato fallito se il SUT non produce i risultati attesi, indicando la presenza di un difetto o di un comportamento imprevisto.

6. Approccio

1. Introduzione

Questo approccio al testing descrive le metodologie, gli strumenti e le tecniche utilizzate per garantire che il sistema soddisfi i requisiti funzionali e non funzionali. Si adatta perfettamente al contesto in quanto consente di affrontare le principali sfide di validazione del software, come la verifica delle funzionalità critiche, la gestione dei dati e la simulazione di scenari realistici. Inoltre, utilizza tecniche consolidate e strumenti intuitivi, rendendo il processo di testing accessibile anche a un team che si approccia a queste pratiche per la prima volta. Questo garantisce una copertura completa dei requisiti e una maggiore affidabilità del prodotto finale.

2. Obiettivi del Test

Gli obiettivi principali del piano di test sono:

- Validare che il sistema funzioni in conformità ai requisiti definiti.
- Identificare e correggere difetti nelle funzionalità principali e nelle interazioni utente.
- Garantire che le prestazioni e la sicurezza del sistema soddisfino gli standard richiesti.
- Verificare che il sistema sia stabile e utilizzabile su tutte le piattaforme previste.

3. Strategia di Test

La strategia di test si basa su un approccio integrato che include test di unità, test di integrazione e test di sistema. Ogni fase copre aspetti specifici dello sviluppo del software, utilizzando strumenti e tecniche appropriate.

3.1. Test di Unità

- Obiettivo: Validare il funzionamento delle unità individuali, come funzioni di gestione del carrello o algoritmi di calcolo dei prezzi.
- Applicazione: Test delle operazioni fondamentali di basso livello, garantendo che ogni componente isolato sia conforme alle specifiche.

4. Criteri di Ingresso e di Uscita

4.1 Criteri di Ingresso:

- Requisiti documentati e approvati.
- Ambiente di test configurato.
- Dati di test pronti e accurati.

4.2. Criteri di Uscita

- Nessun difetto critico bloccante.
- Copertura test delle funzionalità testate.
- I bug critici identificati sono stati risolti o accettati come rischio residuo.

5. Ambito del Test

- **Inclusioni:**
  - Autenticazione.
  - Gestione del carrello e checkout.
  - Operazioni amministrative base.
- **Esclusioni:**
  - Test di performance su larga scala.
  - Test di usabilità approfonditi.

6 Risorse e Strumenti:

- **Strumenti:** JUnit e Mockhito per test di unità,
- **Ambiente:**
  - Sistema operativo: Windows 11.
  - Server: Apache Tomcat 9.
  - Browser: Chrome, Firefox, Edge (versioni recenti).

7. Gestione dei Difetti

I difetti saranno identificati, registrati e monitorati attraverso un sistema di gestione semplice e collaborativo. Ogni difetto sarà classificato in base a gravità e priorità:

- **Critici:** Bloccanti, devono essere risolti prima della consegna.
- **Alti:** Risolvibili, ma accettabili con rischio mitigato.
- **Medi/Bassi:** Correzione pianificata per operazioni future.

Un processo di verifica garantirà che ogni difetto corretto sia testato nuovamente prima della chiusura.

8. Metriche di Test

Per valutare l'efficacia del testing, saranno utilizzate le seguenti metriche:

- **Percentuale di Casi di Test Superati:** Indica il numero di casi di test che hanno avuto successo rispetto al totale eseguito.
- **Numero di Difetti Identificati:** Classificati per gravità (critico, alto, medio, basso).
- **Copertura dei Requisiti:** Percentuale dei requisiti coperti dai casi di test

9. Riassunto

L'approccio al testing integrato e strutturato garantisce una copertura efficace delle funzionalità critiche del sistema, con un focus particolare su stabilità e qualità complessiva. L'utilizzo di tecniche come Black-Box, Bottom-Up e Category Partition, supportato da strumenti come JUnit e Selenium, assicura che il processo di testing sia affidabile e gestibile nei limiti del progetto accademico. Questo approccio riduce significativamente i rischi legati al rilascio e migliora l'esperienza complessiva dell'utente.

7. Sospensione e ripresa

Modulo Autenticazione e Registrazione:

- *Criteri di Sospensione*: Errori bloccanti nel login o registrazione.
- *Criteri di Ripresa*: Risoluzione dei problemi verificata in staging.

Modulo Carrello:

- *Criteri di Sospensione*: Errori nella gestione degli articoli.
- *Criteri di Ripresa*: Correzione e test validati.

Modulo Checkout e Pagamenti:

- *Criteri di Sospensione*: Errori nei flussi di pagamento.
- *Criteri di Ripresa*: Funzionalità ripristinate e validate.

8. Materiali di testing (requisiti hardware/software)

Requisiti Hardware:

- PC con almeno 8 GB di RAM e processore multi-core.

Requisiti Software:

- Windows 11, Chrome/Firefox/Edge (versioni recenti).

Strumenti di Testing:

- **JUnit**: Per testare metodi e unità della logica di business.
- **Mockito**: Per simulare dipendenze e verificare moduli isolati.
- **MySQL**: Per testare query SQL e verificare la consistenza dei dati.

9. Casi di test

- Riferimento al documento di Test Case Specification