Università degli Studi di Salerno

Corso di Ingegneria del Software

SpeedScale
Test Plan
Versione 1.0

Data: 08/01/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Sepe Gennaro	0512116971
La Marca Antonio	0512117826

Scritto da:	Sepe Gennaro

Revision History

Data	Versione	Descrizione	Autore
15/12/2024	0.1	Creazione e scrittura del documento	Sepe Gennaro
08/01/2025	1.0	Aggiustamenti + Rilascio finale	Sepe Gennaro

Indice

1. Introduzione

Questo documento rappresenta il piano di test per il sistema SpeedScale, un'applicazione progettata per fornire una gestione efficiente di utenti, prodotti, ordini e carrelli. Il piano di test è stato concepito per garantire che il sistema soddisfi i requisiti funzionali e non funzionali definiti nei documenti di specifica. L'obiettivo primario di questo piano è fornire una guida chiara e strutturata per la pianificazione e l'esecuzione dei test. Attraverso un insieme di strategie, criteri e casi di test ben definiti, si intende identificare e correggere eventuali difetti, assicurando che il sistema funzioni in modo affidabile e conforme alle aspettative. I test seguiranno un approccio strutturato che includerà verifiche a livello unitario, di integrazione e di sistema. Inoltre, saranno definiti criteri per l'accettazione e il rifiuto dei test, specificando in che modo i risultati influiranno sul processo di sviluppo complessivo. La qualità del sistema sarà garantita attraverso una copertura completa dei test, che verificheranno ogni aspetto critico dell'applicazione.

2. Relazione con altri documenti

La progettazione di SpeedScale si è avvalsa di risorse teoriche e pratiche che hanno guidato la definizione dei requisiti, la progettazione del sistema e la stesura dei relativi documenti. Come riferimento principale, è stato adottato il manuale "Object Oriented Software Engineering Using UML, Patterns, and Java™" di Bernd Bruegge e Allen H. Dutoit. Questo testo ha fornito le linee guida per l'utilizzo di UML, la scrittura di documenti tecnici e l'applicazione di pattern di progettazione, risultando fondamentale per garantire una progettazione chiara e coerente. Il processo di sviluppo è iniziato con la creazione di un Problem Statement, che ha consentito di delineare con precisione gli obiettivi del sistema e le esigenze del mercato di riferimento. Successivamente, è stato realizzato un Requirements Analysis Document (RAD), che ha permesso di ottenere una visione iniziale e strutturata delle funzionalità e delle specifiche del sistema, servendo da base per le fasi successive. La progettazione ha tratto ulteriore valore dagli insegnamenti del corso universitario "Ingegneria del Software", tenuto dal professore Andrea De Lucia, che ha fornito una solida metodologia per la gestione strutturata del ciclo di vita dello sviluppo software. Questi insegnamenti hanno contribuito a definire approcci di qualità sia nella progettazione del sistema che nel suo testing, garantendo una piattaforma robusta ed affidabile. Un altro documento chiave per il test è l'Object Design Document (ODD). Questo documento contiene una descrizione dettagliata delle classi, delle loro responsabilità e delle loro relazioni, offrendo una guida essenziale per individuare i casi di test e per stabilire la copertura dei test rispetto ai requisiti. L'ODD rappresenta il punto di riferimento per il collegamento tra i test unitari e le funzionalità implementate. Infine, tutti i test descritti in questo piano sono strettamente correlati ai requisiti funzionali e non funzionali delineati nei documenti di specifica. Per mantenere una tracciabilità rigorosa, è stato adottato uno schema di denominazione standard che permette di stabilire con chiarezza il legame tra i requisiti e i relativi test. Questo garantisce una verifica completa e sistematica del sistema SpeedScale, assicurando che ogni requisito sia coperto da un caso di test appropriato.

3. Panoramica del sistema

Il sistema SpeedScale è stato progettato seguendo un'architettura multilivello, strutturata in modo da garantire modularità, separazione delle responsabilità e facilità di manutenzione. L'architettura si articola in tre livelli principali: il livello di presentazione, il livello applicativo e il livello dei dati. Il livello di presentazione rappresenta l'interfaccia utente del sistema, attraverso cui gli utenti interagiscono direttamente con la piattaforma. Questo livello è implementato utilizzando le pagine JSF, basate su Java EE 7. Le funzionalità accessibili a questo livello comprendono tutte le operazioni principali, dalla gestione del profilo utente fino all'esplorazione del catalogo e alla gestione del carrello. Il livello applicativo costituisce il nucleo logico del sistema ed è responsabile dell'elaborazione delle richieste provenienti dall'interfaccia utente. Questo livello è composto da servizi modulari progettati per assolvere a specifiche funzionalità. Ad esempio, il servizio di autenticazione gestisce l'accesso degli utenti, le loro credenziali e le sessioni; il servizio di profilo si occupa delle informazioni personali, degli indirizzi di spedizione, dei metodi di pagamento e dello storico degli ordini; il servizio catalogo permette la gestione dei prodotti, mentre il servizio carrello supporta tutte le operazioni relative agli acquisti, come l'aggiunta di prodotti e la loro conversione in ordine. Ogni servizio è stato sviluppato come una classe indipendente, progettata per garantire un'interfaccia chiara e un'interazione efficace con gli altri componenti del sistema. Il livello dei dati è invece dedicato alla gestione della persistenza, affidata a un database relazionale, come MySQL, utilizzando JPA (Java Persistence API) per garantire un accesso efficiente e sicuro alle informazioni. Le entità del dominio, tra cui Utente, Prodotto e Ordine, sono mappate direttamente alle tabelle del database, assicurando una chiara corrispondenza tra i dati logici e quelli fisici. Per garantire l'affidabilità del sistema, i test unitari si concentrano sui moduli fondamentali che compongono i principali sottosistemi, come l'autenticazione, la gestione del profilo, il catalogo e il carrello. Ogni modulo verrà analizzato in modo dettagliato per assicurarsi che le operazioni offerte siano conformi ai requisiti. Successivamente, i test di integrazione verificheranno il corretto funzionamento del sistema nel suo complesso, esaminando il flusso di interazione tra i livelli. La granularità dei componenti e le loro dipendenze saranno analizzate utilizzando un diagramma UML delle classi, che faciliterà la pianificazione delle attività di verifica. Questa panoramica fornisce il contesto strutturale del sistema, delineando gli aspetti fondamentali della sua progettazione e le basi per la definizione di una strategia di test efficace.

4. Caratteristiche da testare/non testare

Il piano di test del sistema **SpeedScale** si concentra sulle funzionalità principali e sulle caratteristiche critiche per garantire il raggiungimento degli obiettivi progettuali. Le aree di verifica includono:

1. **Autenticazione**: test su login, gestione credenziali e mantenimento della sessione utente.
2. **Gestione profilo**: verifica delle modifiche ai dati personali, indirizzi di spedizione, metodi di pagamento e storico ordini.
3. **Catalogo prodotti**: controllo sulla visualizzazione, modifica e gestione dei prodotti.
4. **Carrello**: test sulla gestione degli articoli e sulla conversione in ordine, per assicurare un'esperienza di acquisto fluida.

Verranno analizzati sia la logica applicativa che il comportamento nell'interfaccia utente.

Esclusioni:

- Funzionalità non core, sperimentali o pianificate per versioni future.
- Componenti di terze parti già certificate (es. server di database, librerie, framework Java EE), data la loro affidabilità consolidata.

Questa selezione ottimizza il processo di verifica, concentrando le risorse sulle funzionalità chiave che influenzano direttamente l'esperienza utente e la qualità del sistema.

5. Criteri di accettazione/rifiuto

I criteri di accettazione e rifiuto garantiscono la qualità del sistema **SpeedScale**, definendo condizioni chiare per valutare l'esito dei test.

Criteri di accettazione:

- Ogni test deve essere completato con successo, senza errori o criticità evidenti.
- Le funzionalità principali devono soddisfare pienamente i requisiti funzionali e non funzionali descritti nei documenti **RAD** (Requirements Analysis Document) e **SDD** (System Design Document).
- I requisiti critici devono essere coperti al 100%, mentre quelli secondari devono avere una copertura di almeno il 90%.
- Devono essere rispettati gli standard di performance, sicurezza e usabilità per garantire un'esperienza utente ottimale.

Criteri di rifiuto:

- Presenza di errori bloccanti che impediscono l'esecuzione delle operazioni principali.
- Deviazioni significative dai requisiti definiti.
- Malfunzionamenti ricorrenti non risolti durante i test di regressione.

Gli errori rilevati vengono documentati in rapporti dettagliati, con classificazione per gravità, per facilitare la correzione e i test successivi. Questi criteri assicurano la qualità del sistema e forniscono obiettivi chiari per il completamento dei test.

6. Approccio

Il processo di test per il sistema **SpeedScale** adotta un approccio strutturato e metodico, che combina tecniche manuali e automatiche per garantire una copertura completa delle funzionalità e delle interazioni tra i componenti. L'obiettivo è assicurare che il sistema soddisfi i requisiti in termini di correttezza, efficienza e robustezza, verificando il funzionamento sia dei singoli componenti che del sistema nel suo insieme.

6.1 Testing unitario

Il testing unitario rappresenta la prima fase e si concentra sulla verifica dei singoli moduli. Ogni componente viene testato isolatamente per individuare e correggere eventuali errori a livello di unità. Questo approccio minimizza il rischio di propagazione di difetti e permette di affrontare tempestivamente i problemi legati al codice sorgente. Per questa fase vengono utilizzati strumenti come **JUnit**, che garantiscono un'ottimale automazione e una copertura elevata del codice.

6.2 Testing di integrazione

La seconda fase, il testing di integrazione, mira a verificare le interazioni tra i diversi moduli, garantendo che funzionino correttamente quando combinati. L'approccio seguito è principalmente di tipo bottom-up, partendo dai moduli di livello inferiore e integrandoli progressivamente con quelli di livello superiore. Questa strategia consente di individuare incongruenze o errori nei punti di interfaccia tra i componenti. Le dipendenze tra i moduli, rappresentate in fase di progettazione tramite diagrammi UML, facilitano l'individuazione delle aree critiche da testare.

6.3 Testing di sistema

Il testing di sistema costituisce la fase finale e si focalizza sulla verifica del comportamento complessivo del sistema. In questa fase vengono simulati scenari realistici di utilizzo per testare funzionalità, sicurezza, performance e usabilità. Particolare attenzione è riservata all'uso di **Selenium**, che consente un'automazione efficace e una copertura completa delle interazioni tra le componenti e l'interfaccia utente. Grazie alla sua capacità di simulare scenari complessi, Selenium risulta particolarmente utile per individuare errori difficilmente rilevabili nelle fasi precedenti.

Questo approccio graduale, che spazia dal testing unitario a quello di sistema, garantisce che ogni livello del progetto sia attentamente verificato. L'attenzione posta sull'utilizzo di strumenti automatizzati, in particolare Selenium, assicura che il sistema funzioni correttamente non solo a livello tecnico ma anche in termini di esperienza utente.

7. Sospensione e ripresa

Nel piano di test del sistema **SpeedScale**, la sospensione delle attività di testing può rendersi necessaria in presenza di ostacoli significativi. Tra le cause principali figurano la rilevazione di difetti critici che compromettono funzionalità essenziali, come la gestione di utenti, prodotti o ordini, oppure problemi tecnici come malfunzionamenti hardware o software che impediscono l'esecuzione dei test. Anche la necessità di rivedere le specifiche o risolvere incertezze sui requisiti può giustificare un'interruzione. In caso di sospensione, è essenziale definire con precisione le condizioni per la ripresa delle attività. I test riprenderanno solo dopo che i problemi identificati saranno stati risolti. Inoltre, ogni test interrotto dovrà essere rieseguito integralmente, per verificare che le correzioni apportate non abbiano introdotto nuovi errori o compromesso altre aree del sistema. Prima di riprendere, sarà necessario controllare la disponibilità e il corretto funzionamento delle risorse e degli strumenti, aggiornando al contempo la documentazione per mantenere una traccia chiara delle operazioni svolte. Il processo di sospensione e ripresa deve essere gestito con attenzione per preservare l'integrità e la qualità del sistema. Tutte le fasi del testing, comprese le interruzioni e le riprese, devono essere documentate in dettaglio. Questo include la registrazione dei problemi riscontrati, le azioni correttive adottate e i risultati ottenuti dopo la verifica. Una gestione accurata assicura che i test possano proseguire in modo efficiente, senza compromettere l'affidabilità e la qualità complessiva del sistema.

8. Materiali di prova (requisiti hardware/software)

Per l'esecuzione dei test sul sistema, è sufficiente l'uso di un comune computer fisso o portatile che soddisfi i requisiti hardware e software minimi. In particolare, l'hardware necessario comprende un dispositivo con almeno 8 GB di RAM e una CPU moderna (ad esempio, Intel i5 o equivalente) per garantire che i test vengano eseguiti in modo fluido, anche in presenza di carichi elevati. Per quanto riguarda i software, il sistema di test può essere eseguito su qualsiasi macchina che supporti i requisiti di esecuzione di Java EE 7, poiché il sistema SpeedScale è sviluppato in questo ambiente. Pertanto, sono necessari i seguenti software:

1. **Java Development Kit (JDK)**: Per eseguire i test sui componenti sviluppati in Java, è necessaria l'installazione di una versione compatibile con Java EE 7 (ad esempio JDK 8 o superiore);
2. **Database Relazionale (MySQL)**: SpeedScale utilizza un database relazionale, come MySQL, per gestire la persistenza dei dati. È necessario un ambiente di test che consenta l'esecuzione e la gestione di un database di test, configurato con le entità del dominio;
3. **Strumenti di automazione dei test**: Per i test automatizzati, è consigliato l'uso di strumenti come **JUnit** per i test unitari e **Selenium** per i test di sistema e di integrazione, particolarmente per l'interazione con l'interfaccia utente. Questi strumenti richiedono l'installazione di Java e di browser web compatibili;
4. **Sistema operativo**: Il sistema di test può essere eseguito su sistemi operativi come Windows, macOS o Linux, purché compatibili con le versioni di software sopra indicate.

L'ambiente di test non richiede configurazioni hardware particolarmente avanzate. Un semplice computer, fisso o portatile, che rispetti i requisiti sopra indicati è sufficiente per eseguire tutti i test, garantendo che le verifiche vengano svolte in modo efficiente e senza necessità di risorse hardware complesse.

9. Casi di test

I casi di test sono essenziali per assicurare che tutte le funzionalità del sistema siano correttamente implementate e conformi ai requisiti previsti, sia funzionali che non funzionali. Ogni caso è stato progettato considerando le specifiche del sistema e le esigenze di business, con l'obiettivo di coprire i principali flussi operativi. I test verificano operazioni chiave come il login, la gestione del carrello, l'elaborazione degli ordini e la ricerca dei prodotti. Per ciascun caso di test sono specificati i requisiti di ingresso, i risultati attesi, le dipendenze con altri test e le configurazioni ambientali necessarie per l'esecuzione. Ogni test è descritto in una tabella dedicata, che fornisce dettagli su identificativo, input, output e dipendenze, offrendo una visione chiara e strutturata. Questo formato semplifica la lettura, la gestione e l'esecuzione dei test. L'esecuzione accurata di questi casi garantirà che il sistema sia pronto per il rilascio e per l'uso da parte degli utenti finali, confermando la qualità e l'affidabilità del software.

Di seguito i Test Case Specification dei vari test che si affronteranno:

FILE	***FUNZIONALITA'***
TCS1_Speedscale	Login
TCS2_Speedscale	Registrazione
TCS3_SpeedScale	Finalizzazione ordine

10. Programma di test

I tester devono padroneggiare strumenti come JUnit e Selenium e avere una solida comprensione del sistema basato su Java EE. Il programma di test affronta rischi quali l'integrazione complessa tra moduli, performance sotto carichi elevati, problemi di compatibilità con componenti terze, e l'impatto di modifiche non previste.