TP

Test Plan

AEMME

Riferimento	TP_AEMME
Versione	0.3
Data	28/12/2024
Destinatario	Prof. Carmine Gravino
Presentato da	NC13

Revision History

Data	Versione	Descrizione	Autori
20/12/2024	0.1	Prima stesura	GG, CS, AR, MDL
28/12/2024	0.2	Ultimata la stesura	AR
28/12/2024	0.3	Prima revisione	MDL

Team members

Nome	Ruolo nel progetto	Acronimo	Informazioni di contatto
Alessandra Raia	Team Member	AR	a.raia7@studenti.unisa.it
Gregorio Garofalo	Team Member	GG	g.garofalo31@studenti.unisa.it
Martina De Lucia	Team Member	MDL	m.delucia18@studenti.unisa.it
Carla Stefanile	Team Member	CS	c.stefanile1@studenti.unisa.it

Sommario

1. Introduzione

Il documento di Test Plan ha l'obiettivo di definire le strategie che verranno utilizzate per effettuare il testing del sistema AEMME.

Sono state pianificate attività di testing per i seguenti use case:

- UC_GU_1: Registrazione Cliente
- UC_GWL_1: Inserimento di un libro nella WishList
- UC_GC_1: Aggiunta di un nuovo prodotto nel catalogo
- UC_GO_4: Procedura di acquisto

2. Riferimenti ad altri documenti

- **RAD** (RAD_AEMME): riferimento ai requisiti funzionali estratti durante la fase di requirements elicitation e agli Use case.

3. Panoramica del sistema

Il sistema proposto basa la sua architettura sul sistema three-tier, che rispetta il framework MVC J2EE.

Verranno usati HTML5, Javascript e CSS3 per la parte di front-end e la generazione delle view.

Per la logica applicativa e quindi il back-end sarà utilizzato Java.

4. Features da testare/da non testare

Le funzionalità che saranno testate sono:

- Funzionalità Cliente Registrato:
  - Inserimento libro nella WishList
  - Procedura di acquisto
- Funzionalità Ospite:
  - Registrazione cliente
- Funzionalità Gestore Catalogo:
  - Aggiunta di un prodotto nel catalogo

Le funzionalità di cui non si andrà ad effettuare le attività di testing riguardano requisiti funzionali di bassa o media priorità, ed in generale i requisiti non riguardanti gli use case definiti nel RAD.

5. Criterio Passed/Failed

L'esito di un test case viene determinato confrontando il risultato ottenuto con quello atteso, definito come oracolo, in base ai requisiti.

Un test viene considerato "Passed" se, dato un input al sistema, l'output ottenuto è diverso dall'output atteso dall'oracolo.

Un test viene considerato "Failed" se, dato un input al sistema, l'output ottenuto è uguale all'output atteso dall'oracolo.

6. Approccio

Il testing dell'intero sistema si compone di: testing di sistema e testing di unità.

Prima della fase di implementazione del sistema, avverrà la progettazione dei casi di test di sistema (pianificato in seguito alla stesura del documento RAD); durante la fase implementativa avverrà il testing di unità.

Testing di Sistema

Per il testing di sistema sarà utilizzato il tool Selenium IDE, che permette di registrare le azioni che un utente può intraprendere sul browser, in modo da poter implementare ed eseguire i test case di sistema. Il server, per la fase di testing, verrà deployato in localhost.

> **Functional testing**
>
> Il functional testing ha il fine di validare i requisiti funzionali. Consiste nell'individuare i possibili faults generati dagli input degli utenti.

Testing di Unità

Per il testing di unità la strategia prevista consiste nel testare ogni metodo delle classi del sistema. Da esse, sono escluse le classi entity, poiché quest'ultime presentano solo metodi getters e setters. I casi di test saranno definiti attraverso un approccio black-box e saranno documentati direttamente nel codice, attraverso l'uso del framework per il testing di classi Java JUnit.

Le tecnologie usate in tale fase saranno:

- Mockito: per la costruzione degli stub e l'isolamento della componente testata.
- Maven: per la build e l'esecuzione automatica dei tests.
- JUnit è un framework open source che fornisce asserzioni per testare i risultati attesi, annotazioni per identificare i metodi di prova e test runner grafici e testuali.

7. Sospensione e ripristino

In questa sezione verranno specificati i criteri di sospensione del test e di ripristino.

Criteri di sospensione

Il testing non verrà sospeso fino alla sua terminazione, anche in caso di rilevazione di una failure. Il testing potrà essere momentaneamente sospeso nel caso venga restituito, al momento dell'esecuzione, un errore nella definizione di uno dei test stessi.

Criterio di ripristino

Il testing verrà ripreso dopo aver risolto i fault individuati.

8. Materiale di testing

L'hardware necessario per l'attività di test è un semplice computer, non necessariamente connesso ad internet, in quanto il sistema non è stato ancora rilasciato.

9. Test cases

L'approccio per la definizione dei test frame sarà il category partition. Al fine di minimizzare il numero di test case, gli input saranno partizionati in classi di equivalenza. Per definire l'output atteso si userà un oracolo umano, per via dell'assenza di specifiche formali/semi-formali.