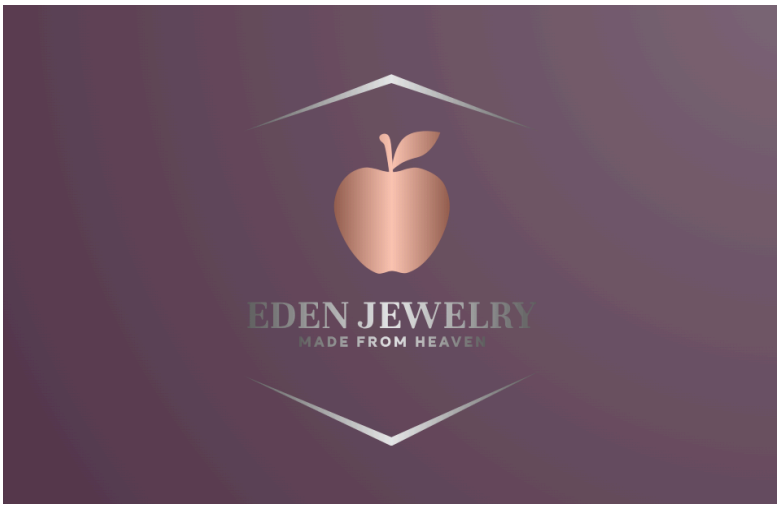
Università degli Studi di Salerno
Corso di Ingegneria del Software

EdenJewelry
TestPlanEdenJewelry
Versione 2.0



Data: 18/01/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Gaetano D'Alessio	0512110836
Luigi Montuori	0512117799
Miriam Eva De Santis	0512117121

Scritto da:	Luigi Montuori

Revision History

Data	Versione	Descrizione	Autore
16/01/2025	1.0	Prima stesura del file di Test Plan	Luigi Montuori
16/01/2025	1.1	Compilazione del punto 2 e 3 del documento	Gaetano D'Alessio
16/01/2025	1.3	Compilazione del punto 5 del documento	Miriam Eva De Santis
16/01/2025	1.4	Compilazione del punto 4 e 6 del documento	Gaetano D'Alessio
16/01/2025	1.5	Compilazione del punto 9 del documento	Luigi Montuori
16/01/2025	1.6	Compilazione del punto 7 del documento	Miriam Eva De Santis
16/01/2025	1.7	Compilazione del punto 10 del documento	Luigi Montuori
16/01/2025	1.8	Compilazione del punto 8 del documento	Miriam Eva De Santis
16/01/2025	1.9	Compilazione dell'indice	Miriam Eva De Santis
18/01/2025	2.0	Ultime modifiche	Gaetano D'Alessio

Indice

1. INTRODUZIONE 4
2 Riferimenti ad altri documenti 4
3 System overview 4
4 Feature to be tested/not be tested 5
5 Pass/Fail criteria 6
6 Approach 6
7 Suspension and resumption 6
8 Testing materials (hardware/software requirements) 7
9 Test Cases 7
10 Testing schedule 7

1. Introduzione

In questo documento affronteremo la fase di test a cui il prodotto deve essere sottoposto. Lo scopo del testing è quello di valutare il corretto funzionamento del sistema, tramite alcune tecniche di test specifiche ed affrontate a lezione, e di scoprire valori di confine o operazioni che portano alla generazione di un comportamento anomalo o lontano dalle nostre aspettative. Tramite queste valutazioni, potremmo creare ed aggiungere delle soluzioni che contrastino questi errori.

2. Riferimenti ad altri documenti

Illustriamo ora i riferimenti agli altri documenti sviluppati, durante le fasi di analisi e progettazione:

- Requirements Analysis Document (**RAD**): da questo documento ricaveremo i requisiti funzionali da testare;
- System Design Document (**SDD**): la visione di questo documento risulta essenziale per la suddivisione in sottosistemi in esso contenuta;
- Object Design Document (**ODD**): in questo documento riprendiamo le dipendenze tra gli oggetti individuati;
- Test Case Specification: è da intendersi come una prosecuzione del Test Plan.

3. System overview

Il sistema EdenJewelry è progettato per:

- Registrarsi ed effettuare l'accesso;
- Fornire un vasto catalogo di gioielli;
- Visualizzare i dettagli riguardo i propri gioielli preferiti;
- Acquistare gioielli;
- Visualizzare il proprio storico degli ordini;
- Permettere ai venditori di inserire e/o rimuovere gioielli;
- Assicurare la disponibilità dei prodotti.

Il sistema utilizza un'architettura a tre layer dove la trasparenza permette all'*upper* layer di accedere al *bottom* layer direttamente, per motivi di efficienza. Ogni strato raggruppa i moduli concettualmente simili, senza ricorrere alla classica suddivisione "Presentation-Business-Data".

I moduli da testare durante lo unit testing sono i seguenti:

- ➔ Gestione account
  - ◆ Autenticazione
  - ◆ AreaUtente
  - ◆ AreaVenditore
  - ◆ Registrazione
  - ◆ Utente
- ➔ Gestione prodotti
  - ◆ Homepage
  - ◆ Catalogo
  - ◆ Carrello
  - ◆ Prodotto

- ➔ Gestione ordini
  - ◆ Ordini
  - ◆ Acquisto

4. Feature to be tested/not be tested

Le attività di testing interessano tutte le funzionalità che gli sviluppatori hanno reputato ad alta priorità.

Le funzionalità da testare sono:

- ➔ Gestione account
  - ◆ Autenticazione
    - Login (FR1)
    - Logout
  - ◆ Registrazione
    - Register (FR1)
  - ◆ AreaUtente
    - Visualizzazione area riservata all'utente
  - ◆ AreaVenditore
    - Visualizzazione area riservata al venditore

- ➔ Gestione prodotti
  - ◆ Homepage
    - Ricerca del prodotto
  - ◆ Catalogo
    - Visualizzazione del catalogo (FR2, FR3)
    - Visualizzazione stock dei prodotti (FR9)
    - Aggiunta e rimozione dei prodotti (FR10, FR11)
  - ◆ Carrello
    - Aggiunta e rimozione prodotti dal carrello
    - Modifica quantità di un prodotto del carrello
  - ◆ Prodotto
    - Selezione dei prodotti (FR4)

- ➔ Gestione ordini
  - ◆ Ordini
    - Visualizzazione dello storico (FR5)
    - Visualizzazione dello storico completo (FR12)
  - ◆ Acquisto
    - Acquisto dei prodotti (FR6)

Abbiamo escluso da questa lista:

- ➔ Il modulo Wishlist, in quanto avente priorità media è possibile svilupparlo e testarlo in un momento successivo;
- ➔ Le classi e i metodi di check, non definiti nelle fasi di analisi (sono comunque ausiliari e trascurabili);

5. Pass/Fail criteria

Allo scopo di stabilire se una certa componente testata ha superato o fallito il test, diremo:

- ➔ Ad una componente viene assegnato valore “fail” se il test non ha portato a comportamenti fuori dalle nostre aspettative. Dunque, il risultato ottenuto è anche in linea con quello atteso;
- ➔ Ad una componente viene assegnato valore “pass” se il test è riuscito a compromettere il suo comportamento. Quindi, il risultato ottenuto è diverso da quello atteso;

6. Approach

L’approccio scelto per il processo di testing comprende:

1. **Test Unitari**
   Il test di unità tratta il testing delle singole componenti (classi, metodi) che compongono il sistema.
   I partecipanti di questo progetto hanno deciso di utilizzare una metodologia di testing “**black-box**”, che si basa sul testare l’output atteso rispetto ad un determinato input senza badare alla struttura interna del metodo. Per fare ciò si divide l’input in classi di equivalenza, scegliendo i test case per ogni classe di equivalenza.
   Il tool scelto per svolgere quest'attività è **JUnit**, per ogni classe sviluppata sarà quindi prevista una classe di test contenente i casi previsti.

2. **Test D’Integrazione**
   Una volta effettuato il testing delle singole componenti si passa al test d’integrazione, che consiste nel testare le componenti insieme, al fine di trovare nuovi errori che non erano comparsi in precedenza.
   Si è scelto di utilizzare un approccio cosiddetto “**sandwich modificato**”. In questa strategia di testing si testa in parallelo il livello superiore, il livello in mezzo e il livello in basso. Inoltre, testa in parallelo il livello superiore che accede a quello nel mezzo e il livello in basso acceduto da quello nel mezzo.
   Anche in questo caso, si utilizzerà **JUnit** per i driver e **Mockito** per gli stub.

   (Ovviamente test di unità e d’integrazione prevedono inevitabilmente **test di regressione**).

3. **Test di Sistema**
   Infine verrà eseguito il test del sistema nella sua completezza non solo dal punto di vista funzionale, ma anche per quanto riguarda i requisiti non funzionali.
   Ovviamente con la dovuta premessa che il per **performance test** eseguito sara’ amatoriale, per le risorse e il tempo limitati di cui disponiamo.
   E’ da prevedersi anche una parte di testing eseguita nell’ambiente finale per cui il sistema e’ stato progettato (Apache Tomcat), con gli sviluppatori pronti a correggere eventuali bug.
   Il tool che utilizzeremo per quest’ultima fase è **Selenium**, il quale è specificatamente pensato per automatizzare il testing di siti web.

7. Suspension and resumption

- ➔ **Criteri di sospensione**

- Ogni test eseguito è fallito. Quindi la componente esegue le sue operazioni correttamente;
- I test eseguiti hanno avuto successo. Quindi le funzioni della componente devono essere riformulate e corrette, in modo che questa possa compensare le sue falle;

➔ **Criteri di ripresa**
- In seguito a delle modifiche effettuate su una componente, si rieffettuano dei test per valutare il suo nuovo comportamento;
- Dopo aver unificato tutte le componenti in un unico sistema integrato, si esegue un test complessivo per poter valutare la cooperazione tra esse;

8. Testing materials (hardware/software requirements)

Le apparecchiature che ci supportano in questa fase di test di componenti e sistema sono i nostri personal computer, attraverso l'utilizzo di software come JUnit, Mockito e Selenium (già precedentemente introdotti nel file System Design Document e citati nel punto 6). Il tutto, avviene con la cooperazione dei tre componenti di questo gruppo, che conducono le loro riunioni nelle aule studio del Dipartimento di Informatica dell'università degli Studi di Salerno.

9. Test cases

I test cases rappresentano la parte principale del piano di test. Sono introdotti e illustrati nel documento "TestCaseSpecification_EdenJewelry". Ogni test case appartenente alle test suite viene qui approfondito e redatto nella sua interezza.

10.Testing schedule

Il nostro testing schedule sarà strutturato in maniera tale da assicurare un processo di verifica completo.
Come già specificato nell'approccio, andremo a dividere la fase di testing in diverse parti.

➔ **Test unitari**:
In questa fase che durerà all'incirca una settimana, salvo problemi o ritardi, gli sviluppatori condurranno il testing parallelamente allo sviluppo del codice. Eventuali bug o errori saranno risolti tempestivamente dal team.

➔ **Test di integrazione**:
Sarà svolto approssimativamente in 4 giorni, durante i quali saranno testate le interazioni e le commistioni dei vari moduli del sistema.

➔ **Test di sistema**:
Infine, il sistema sara' sottoposto ad un testing di sistema dalla durata complessiva di 3 giorni. Bug o errori in questa fase potrebbero provocare un parziale fallimento o rinvio del progetto.