TEST PLAN

Indice

[1. Introduzione 3](#_Toc535419477)

[2. Documenti correlati 3](#_Toc535419478)

[2.1. Relazioni con il Requirement Analysis System (RAD) 3](#_Toc535419479)

[2.2. Relazioni con il System Design Document (SDD) 3](#_Toc535419480)

[2.3. Relazioni con l’Object Design Document (ODD) 3](#_Toc535419481)

[3. Panoramica del sistema 3](#_Toc535419482)

[4. Testing Scope 4](#_Toc535419483)

[4.1. Funzionalità da testare 4](#_Toc535419484)

# Introduzione

Lo scopo di questo documento è quello di pianificare l’attività di testing del sistema EsteticaMente identificando le componenti e le funzionalità da testare e le tipologie di testing che verranno adottate. Questo al fine di verificare se il comportamento atteso coincide con quello conseguente all’implementazione. Con questa attività si potranno così rilevare eventuali errori e quindi apportare correzioni in maniera tale da fornire all’utente un prodotto in linea con le sue aspettative.

In questa attività saranno testate le seguenti gestioni:

* Gestione Utente
* Gestione Carrello
* Gestione Ordine
* Gestione Prodotto

Tali gestioni sono quelle analizzate nel documento dell’ODD e verranno testate le rispettive funzionalità.

Le attività di test sono strettamente legate ai documenti fino ad ora prodotti che ne costituiscono un punto di partenza.

# Documenti correlati

## Relazioni con il Requirement Analysis System (RAD)

La relazione del Test Plan con il RAD riguarda i requisiti funzionali e non funzionali del sistema. I test delle funzionalità prenderanno in considerazione le specifiche espresse nel RAD.

## Relazioni con il System Design Document (SDD)

Il testing delle componenti del sistema rispecchierà la suddivisione in sottosistemi riportata nel SDD.

## Relazioni con l’Object Design Document (ODD)

Il test di integrazione e di unità faranno riferimento alle class interfaces dichiarate nell’ODD.

# Panoramica del sistema

Come stabilito nel SDD la struttura di EsteticaMente è divisa secondo un’architettura three-tier: Presentation Layer, Application Layer, Storage. Il livello più alto interagisce con il livello applicativo che a sua volta si occuperà di eseguire le operazioni nel database. Abbiamo quindi identificato i principali sottosistemi:

* Gestione Utente
* Gestione Autenticazione
* Gestione Registrazione
* Gestione Ordine
* Gestione Prodotto
* Gestione Pagamento

Di questi sottosistemi andremo a testare le principali funzionalità.

# Testing Scope

## Funzionalità da testare

Di seguito andremo ad elencare le funzionalità, per ogni gestione, che andranno testate:

* Gestione Utente
* Modifica Utente
* Eliminazione Utente
* Gestione Autenticazione
* Login
* Gestione Registrazione
* Registrazione
* Gestione Prodotti
* Modifica Prodotto
* Eliminazione Prodotto
* Inserimento Prodotto
* Ricerca Prodotto
* Gestione Pagamento
* Effettua Pagamento
* Gestione Ordine
* Conferma Ordine
* Modifica stato Ordine
* Elimina Ordine

## Funzionalità da non testare

Le funzionalità da non testare sono principalmente le interfacce lato utente, interfacce hardware e logica di database.

o

Tutte le funzionalità precedentemente non menzionate non saranno testate poiché considerate attività **low-risk** per il nostro sistema, utilizzate e considerate stabili.

o

Tutte le funzionalità precedentemente non menzionate non saranno testate poiché non sono considerate critiche per il nostro sistema.

# Pass/Fail criteria

Un test si reputerà superato se l’output risultante dallo specifico test case sarà quello atteso e precedentemente determinato dal responsabile dei test.

# Approccio

Il testing prevederà prima di tutto il test delle unità dei singoli componenti, così da assicurarsi della loro correttezza.  
In seguito sarà effettuato il test di integrazione che si concentrerà principalmente sul test delle interfacce delle unità di cui prima.  
Infine verrà svolto il test di sistema che verterà nel testare l’intero sistema assemblato con tutti i componenti.

## Test di Unità

Per il test di unità sarà utilizzato testing Black-Box, ovvero le componenti verranno testate senza conoscere la loro struttura interna, ma tenendo conto solo dell’input e dell’output.  
In particolare verrà utilizzato il metodo di Category Partition:

* nel sistema vengono individuate le funzioni che possono essere testate individualmente
* si identificano i parametri di ogni funzione
* per ogni parametro si definiscono delle categorie (proprietà o caratteristiche importanti)
* le categorie vengono divise in scelte, ovvero nei possibili valori che possono assumere
* vengono identificati i vincoli esistenti tra le varie scelte
* vengono costruiti i Test Frames che contengono tutte le possibili combinazioni di scelte nelle categorie, ovvero tutti i possibili test case.

L’esecuzione dei test case sarà poi riportata nel documento di Test Execution ed eventuali errori verranno segnalati nel documento di Test Accident Report. Un report dell’attività di testing completo verrà riportato nel Test Summary Report.

## Test d’Integrazione

In questa fase si provvederà all’integrazione delle componenti del sistema, testandole insieme con strategia [strategia: bottom-up, sandwich, top-down?]