

**Università degli Studi di Salerno**

**Anno Accademico 2016/2017**

**Corso di Ingegneria del Software**

**System Test Plan**

**V 0.4.1**

****

Introduzione 4

Scopo 4

Obiettivi 4

Strategia globale e approccio 4

Strategia di testing 4

Tipi di test 4

Test di usabilità 4

Test funzionale 4

Criteri di sospensione dei test e requisiti di ripresa dei test 4

Criteri di sospensione 5

Requisiti di ripresa 5

Test data 5

Piano di esecuzione 5

Test database 5

**Top Manager:**

*Prof. De Lucia Andrea*

**Team di sviluppo:**

|  |  |
| --- | --- |
| Nome e Cognome | Matricola |
| *Stefano Foresta* | 0512100504 |
| *Gennaro Franzese* | 0512100270 |

**Revision History:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Autore** | **Data** | **Descrizione** | **Versione** |
| *Stefano Foresta* | *13/12/16* | *Struttura documento* | *v 0.1* |
| *Gennaro Franzese* | *13/12/16* | *Stesura capitolo uno* | *v 0.3* |
| *Gennaro Franzese* | *14/12/16* | *Stesura capitolo due* | *v 0.4* |
| *Stefano Foresta* | *14/12/16* | *Revisione capitoli uno e due* | *v 0.4.1* |
| *Stefano Foresta* | *14/12/16* | *Stesura capitolo tre* | *v 0.6* |

# Introduzione

## Scopo

Questo documento specifica un piano di test della web application “**I-Tech**”. Esso descrive la strategia di sperimentazione e l’approccio al test della stessa che il team utilizzerà per verificare che la domanda soddisfa i requisiti stabiliti dal business plan prima del rilascio. In questa attività andremo a rilevare gli eventuali errori prodotti all’interno del codice, per evitare che essi si presentino nel momento in cui il sistema verrà utilizzato dall’utente finale.

Verranno testate esclusivamente le funzionalità implementate e specificate nell’ODD. La fase di testing è quindi strettamente legata alle fasi ad essa precedenti; ogni documento, risultato delle differenti fasi di sviluppo, sarà un punto di partenza indispensabile e centrale per poter effettuare un testing corretto e adeguato al sistema software sviluppato.

## Obiettivi

* Soddisfare le specifiche, le esigenze e le regole di progetto;
* Supportare gli standard aziendali e software specificati;
* Soddisfare i criteri di input dell’utente che testerà il programma.

## Scenari di testing

Le attività di test che si andranno ad eseguire sono state pianificate per le seguenti operazioni:

* Gestione registrazione
* Gestione autenticazione
* Gestione riparazioni
* Gestione acquisti
* Gestione magazzino
* Gestione utenti

# Documentazione allegata al System Test Plan

Il test plan ha, come sottoscritto nell’introduzione, una relazione stretta con il resto dei documenti che sono stati prodotti durante l’elaborazione del prodotto software in questione, poiché in essi è stata pianificata la modalità di sviluppo e di futuro utilizzo dello stesso. Questo quindi permette di rilevare le eventuali differenze tra il comportamento atteso e quello osservato del sistema (cosa fondamentale). Di seguito verranno riportate le relazioni tra il test plan e la documentazione precedente.

## Relazione con il documento di analisi dei requisiti (RAD)

Riguarda particolarmente i requisiti funzionali e non funzionali che sono stati specificati in precedenza e descritti nel RAD in maniera accurata.

## Relazione con il documento di system design (SDD)

In tale documento vengono specificati i sottosistemi e la divisione in livelli è quindi, obbiettivo fondamentale di tale test, è di rimanere quanto più fedele a tale suddivisione.

## Relazione con il documento di object design (ODD)

Troviamo, in questo documento, la specifica sulle interfacce che saranno poi oggetto di test.

# Panoramica generale del sistema oggetto di testing

Come stabilito nel System Design Document la struttura del nostro sistema è divisa secondo una architettura “Two Layers” cioè a due livelli: Interface and Application Layer e Storage Layer. In questo caso il livello più alto è legato indissolubilmente con il livello applicativo che a sua volta eseguirà le operazioni nel database, cercando di garantire il più possibile basso accoppiamento e alta coesione tra le varie classi. Il sistema inoltre è stato suddiviso in sottosistemi più piccoli che cooperano per il funzionamento ottimale dell’applicazione sviluppata.

Quasi ognuna di queste gestioni prevede principalmente operazioni di inserimento, modifica, cancellazione e visualizzazione: saranno proprio queste funzionalità ad essere testate nel corso della fase di testing del sistema.

# Funzionalità da testare

Di seguito è riportato l’elenco delle funzionalità che verranno testate:

1. Gestione registrazione
   1. Registrazione
2. Gestione autenticazione
   1. Login
3. Gestione riparazioni
   1. Richiesta preventivo
   2. Invio preventivo
   3. Accettazione/rifiuto preventivo
   4. Inizio riparazione
   5. Comunicazione riparazione effettuata
4. Gestione acquisti
   1. Carrello
   2. Finalizzazione acquisto
5. Gestione magazzino
   1. Inserimento prodotti
   2. Carico/Scarico prodotti
   3. Modifica prezzo prodotti
6. Gestione utenti
   1. Modifica dati

# Criteri Pass/Failed

I dati di input del test saranno suddivisi in classi di equivalenza, ovvero verranno raggruppati in insiemi dalle caratteristiche comuni, per i quali sarà sufficiente testare un solo elemento rappresentativo. Un input,quindi, avrà superato un test se l’output risultante sarà quello atteso, cioè quello che è stato specificato dal team rispetto al test case.

# Strategia globale e approccio

Le tecniche di testing adottate riguarderanno inizialmente il testing di unità dei singoli componenti, come detto in precedenza, in modo da testare nello specifico la correttezza di ciascuna unità. Seguirà il testing di integrazione, che focalizzerà l’attenzione principalmente sul test delle interfacce delle suddette unità. Infine verrà eseguito il testing di sistema, che vedrà come oggetto di testing l’intero sistema assemblato nei suoi componenti. Quest’ultimo servirà soprattutto a verificare che il sistema soddisfi le richieste del cliente.

## Testing di unità

Durante questa fase, verranno ricercate le condizioni di fallimento, isolando i componenti ed usando test driver e stub, cioè implementazioni parziali di componenti che dipendono o da cui dipendono le componenti da testare. La strategia utilizzata per il testing si baserà esclusivamente sulla tecnica Black-Box, che si focalizza sul comportamento Input/Output, ignorando la struttura interna della componente. Al fine di minimizzare il numero di test cases, i possibili input verranno partizionati in classi di equivalenza e per ogni classe verrà selezionato un test case. Gli stati erronei scovati in questa, come in qualsiasi altra fase di testing, che comporteranno un fallimento del sistema dovranno essere tempestivamente comunicati agli sviluppatori al fine di correggerli e ripristinare il testing al più presto, per comunicare i vari fallimenti identificati durante la fase di testing si utilizzerà un test incident report.