Test Plan

Indice

1. Introduzione

Lo scopo di questo documento è quello di pianificare l'attività di test del sistema al fine di verificare se esistono differenze tra il comportamento atteso e il comportamento osservato.
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.
Le attività di test sono state pianificate per le seguenti gestioni:

- Gestione Pratica
- Gestione newsLetter
- Gestione Avvisi
- Visualizzazione

Lo scopo del documento è quello di definire i test case su cui verranno testate le funzionalità del sistema.
Per ogni funzionalità saranno forniti un numero sufficiente di istanze di input in modo tale da fornire almeno un test case composto da dati corretti, quindi appartenenti a classi valide per ogni campo di input che rispettano le condizioni de finite nel documento di test plan, almeno un test case per ogni input che prevede una classe di equivalenza non valida e almeno un test case che non soddisfa le condizioni definite nel test plan.
La fase di testing è strettamente legata alle fasi ad essa precedenti; ogni documento, risultato delle differenti fasi di sviluppo, sarà punto di partenza

indispensabile per poter effettuare un testing corretto e adeguato.

2. Documenti correlati

Il test plan ha ovviamente una stretta relazione con il resto dei documenti che sono stati prodotti finora, poiché prima di passare alla fase di testing,esso era stato pianificato nelle precedenti documentazioni. Questo quindi permette di rilevare le eventuali differenze tra ciò che si desiderava e ciò che invece il sistema fa.
Di seguito verranno riportate le relazioni tra il test plan e la documentazione precedente.

2.1: Relazioni con il documento di analisi dei requi siti (RAD)

La relazione tra test plan e RAD riguarda in particolare i requisiti funzionali e non funzionali del sistema poiché i test che saranno eseguiti su ogni funzionalità terranno conto delle specifiche e spresse nel RAD.

2.2: Relazioni con il System Design Document (SDD)

Nel System Design Document avevamo suddiviso il nostro sistema in sottosistemi e l’architettura in tre

livelli: Presentation Layer, Application Layer e Storage Layer. Il test dei vari componenti deve rimanere fedele a queste suddivisioni il più possibile.

3. Overview del Sistema

Come stabilito nel System Design Document la struttura del nostro sistema è divisa secondo un'architettura "Tree Tiers" cioè a tre livelli:presentation Layer, application Layer, storage Layer.
In questo caso il livello più alto interagisce con il livello applicativo che a sua volta si occuperà di eseguire le operazioni nel database, cercando di garantire il più possibile basso accoppiamento e alta coesione tra le varie classi.
Le caratteristiche da testare per il controllo del corretto funzionamento di ciascuna funzionalità saranno:
• robustezza: capacità del sistema di reagire fornendo input non valido per il dominio applicativo;
• usabilità: analisi di ogni forma di interazione corrisposta da messaggi di aiuto (in caso di errore) o di notifica (in caso di operazioni eseguite con successo) ;
• sicurezza: verrà testato che un utente col solo interfacciamento grafico non possa intaccare dati non inerenti alla specifica dell'operazione o accedere a dati non consentiti;

- correttezza: verrà testato che le operazioni vengano eseguite correttamente così come dalla specifica dei requisiti sono stati designati.

4. Funzionalità da testare

Di seguito saranno elencate le funzionalità che saranno testate.
Gestione Pratica,news,avvisi: Inserimento, seleziona (modifica, elimina) elenca , ricerca.
La precedente gestione prevede principalmente operazioni di inserimento, modifica, cancellazione, visualizzazione e ricerca e saranno proprio queste funzionalità ad essere testate nel corso della fase di testing del sistema.

5. 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 avrà superato un test se l'output risultante sarà quello atteso, cioè quello che è stato specificato dal membro di testing su tale test case, il membro di testing conosce quale dovrebbe essere l'output corretto.

6. Approccio

Le tecniche di testing adottate riguarderanno inizialmente il testing di unità dei singoli componenti, in modo da testare nello specifico la correttezza di ciascuna unità.
Seguirà il testing d'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 Committente.

6.1. Testing di Unità

Durante questa fase, verranno ricercate le condizioni di fallimento isolando i componenti ed usando test driver e test 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 BlackBox, che si focalizza sul comportamento Input/Output,ignorando la struttura interna della componente. Al fine di minimizzare il numero dei 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 implementatori al fine di correggerli e ripristinare il testing al più presto.

6.2. Testing d’integrazione

In questa fase si procederà all’integrazione delle componenti di una funzionalità che verranno testate nel complesso attraverso una strategia di Bottom-Up. Si passerà, poi, alla funzionalità successiva fino ad esaurire le funzionalità implementate. Quest’approccio mira principalmente a ridurre le dipendenze tra funzionalità differenti e a facilitare la ricerca di errori nelle interfacce di comunicazione tra sottosistemi.

6.3. Testing di Sistema

Lo scopo di questa fase del testing è quello di dimostrare che il sistema soddisfi effettivamente i requisiti richiesti e che sia, quindi, pronto all’uso.
Come per il testing di unità, si cercherà di testare le funzionalità più importanti per l’utente e quelle che hanno una maggiore probabilità di fallimento. Si noti che, come per il testing di unità, si procederà attraverso la tecnica BlackBox.

7. Sospensione e Ripresa

7.1 Criteri di sospensione

La fase di Testing del sistema verrà sospesa quando si raggiungerà un compromesso tra qualità del prodotto e costi dell'attività di Testing. Il Testing verrà quindi portato avanti quanto più possibile nel tempo senza però rischiare di ritardare la consegna finale del progetto.

7.2 Criteri di ripresa

In seguito ad ogni modifica o correzione delle componenti che genereranno errori o fallimenti, i test case verranno sottoposti nuovamente al sistema assicurandoci così di aver risolto effettivamente il problema.

8.Materiale per il testing (Requisiti Hardware/software)

L'hardware necessario per l'attività di test è un PC, o possibilmente uno per ogni oggetto incaricato del testing, su cui sia installato il DBMS MySQL 5.0 e Apache 2 + PHP 5.0

9. Test Cases

Le varie fasi di testing necessiteranno ognuna di test case utili ad individuare errori ed anomalie sia analizzando il codice che provando la sua esecuzione.
Si sono individuate varie classi di equivalenza per ogni input che possa essere immesso per l'utilizzo di una o più componenti. In tal modo, è possibile sviluppare test case con input delle tipologie identificate per testare una o più unità.

Di seguito sono elencate le classi di equivalenza che saranno prese in considerazione durante i successivi documenti di Testing per sviluppare i
test case.

9.1 Classi di equivalenza

-PRATICA
Inserimento,modifica

Input	**Classe**	**Valido**	**Classe**	**Non Valido**
codice	CE01	Codice≠ ‘ ’ Codice > 0	CE02	Codice= ‘ ’ Codice <= 0
NomeP	CE03	NomeP ≠ ‘ ’ NomeP=Formato Stringa	CE04	NomeP= ‘’ NomeP=Formato intero
data	CE05	Data ≠ ‘ ’ Data = formato stringa	CE06	Data = ‘ ’ Data = formato intero
fileP	CE07	fileP ≠ ‘ ’ fileP=file pratica	CE07	fileP= ‘’ fileP=formato intero fileP=formato Stringa

AVVISI
Inserimento,modifica

Input	**Classe**	**Valido**	**Classe**	**Non Valido**
Codice	CE01	Codice≠ ‘ ’ Codice > 0	CE02	Codice= ‘ ’ Codice <= 0
data	CE03	Data ≠ ‘ ’ Data=formato stringa	CE04	Data = ‘ ’ Data=formato intero
visibilità	CE05	Visibilità ≠ ‘’ Visibilità=tipo utente	CE06	Visibilita= ‘’ Visibilita=formatoStringa Visibilita=formato intero

NEWS
Inserimento,modifica

Input	**Classe**	**Valido**	**Classe**	**Non Valido**
Codice	CE01	Codice≠ ‘ ’ Codice > 0	CE02	Codice= ‘ ’ Codice <= 0
data	CE03	Data ≠ ‘ ’ Data=formato stringa	CE04	Data = ‘ ’ Data=formato intero

10. Pianificazione del Testing

L’attività di testing è fondamentale nello sviluppo di un sistema software in quanto la mancanza di tale attività o un cattiva interpretazione di essa può portare al completo fallimento del sistema.

10.1 Gestione dei rischi

I possibili rischi generati dalle attività di testing sono stati minimizzati diminuendo le componenti del sistema da implementare e, quindi, testare.
Inoltre, effettuando un testing di tipo funzionale viene limitato lo sviluppo di stub e driver per il testing delle singole componenti e quindi l’introduzione di nuovi errori nel nuovo codice di cui si sarebbero composti.
D’altro canto il testing di funzionalità rallenta l’individuazione di errori qualora un caso di test avesse esito positivo, poiché l’utilizzo di più componenti per il test di una singola funzionalità estranea dall’ipotesi di totale correttezza di ogni componente interessata.
Il risultato atteso è di riscontrare al più un errore parziale per funzionalità: le componenti interagiscono fra loro e la funzionalità viene eseguita ma non in modo completamente corretto.
Qualora la fase di testing evidenziasse un numero di errori maggiore rispetto alla media attesa, viene pianificato un impegno maggiore dei membri del team sulle attività di testing ed in casi estremi l’abbandono

delle altre attività finché errori gravi (funzionalità non corretta, risultati errati, modifiche apportate in modo errato) non vengano risolti.

10.2 Organizzazione delle attività

Le attività di testing devono svolgersi sulle singole funzionalità divise nei livelli di suddivisione del sistema, rispettando le direttive indicate dal documento di system design.

10.3 Schedulazione delle attività

Le attività di testing per il sistema sono previste per un periodo di tempo massimo di 5 giorni.