Test Plan

Università degli studi di Salerno

Corso di Ingegneria del Software 2017-2018

TEAM

Nome Cognome	Matricola
Carmine Picariello	0512103604
Prisco Luigi	0512103602
Luca D'Avino	0512103496

Indice

1.Introduzione

Il testing consiste nel trovare le differenze tra il comportamento atteso e quello specificato attraverso il modello del sistema e il comportamento osservato dal sistema implementato.
Lo scopo di tale attività è provare il sistema e rilevare i problemi. Si massimizzano gli errori in modo tale da correggerli successivamente. Il testing è un'attività contraria a quelle svolte precedentemente , infatti , a differenza delle altre cerca di distruggere il sistema. Lo scopo del documento è quello di definire i test case su cui verranno testate le funzionalità. Per ogni funzionalità saranno forniti un numero sufficiente di input che prevedano almeno un test case appartenente a classi valide e almeno un test case per classi non valide e almeno un test case che non soddisfi le condizioni definite nel test plan.

2.Riferimenti e Relazioni

In questo capitolo verranno definite le relazioni con gli altri documenti prodotti durante la fase di sviluppo della web app MaiGim.Tali relazioni sono necessarie per produrre una fase di testing coerente con le specifiche dedotte.Documenti :

[1] - “Documento Analisi dei Requisiti“ - RAD.pdf
[2] - “Documento di System Design“ - SDD.pdf
[3] - “Documento di Object Design“ - ODD.pdf

2.1 Relazioni con il documento di Analisi dei Requisiti (RAD)

Tali relazioni riguardano i requisiti funzionali e non funzionali. I test terranno conto delle specifiche espresse nel RAD , bisognerà tener conto principalmente degli attori descritti nel documento di analisi :

- ➢ Utente
- ➢ Istruttore

Entrambi gli attori saranno soggetti a testing , per ciascuno di essi dovranno esser testati la validità dei campi associati e dovranno essere considerate le funzioni ad essi associate.

2.2 Relazioni con il documento di System Design (SDD)

La pianificazione dei test delle componenti rispecchia la decomposizione in sottosistemi specificata nel documento di System Design.

2.3 Relazioni con il documento di Object Design(ODD)

La fase di testing deve tener conto del documento di Object Design in quanto esso fornisce la base per la realizzare l’implementazione.

3. Panoramica del sistema

Il sistema MaiGim segue il modello MVC , dove la M sta per Model , la V per Viewl e la C per Controller. Il Model rappresenta i dati dell'applicazione , la View rappresenta la visualizzazione degli oggetti e il Controller che rappresenta l'insieme di regole che permettono la trasformazione degli input in modifiche del modello. Ogni livello , come descritto nel documento SDD , è stato diviso in sottosistemi. Il model , invece , si estranea da tale suddivisione in quanto contenente l'insieme delle entità che forniscono una astrazione dalla natura della base di dati sottostante e l'insieme delle componenti che permettono la lettura/scrittura.

Gli obiettivi da raggiungere per questo Test Plan sono :

-Dettagliare le attività richieste per preparare e condurre il testing.

-Definire le fonti usate per preparare la pianificazione

3.1 Individuare quando deve partire il processo di testing e quale è il criterio di terminazione da utilizzare.

Identificare a priori lo stato di inizio e di fine all'interno di un processo è fondamentale per evitare che il processo cominci quando ancora non sono presenti tutte le informazioni necessarie o che si prolunghi eccessivamente.

3.2 Definire per ogni test i risultati attesi

La descrizione dei risultati ottenuti dall'esecuzione di un test case è determinante poiché, senza una precisa connotazione degli output attesi, si corre il rischio di interpretare come corretti dei risultati che viceversa non lo sono.

3.3 Analizzare a fondo i risultati di ciascun test , non pianificare il testing assumendo tacitamente che nessun errore sarà individuato

Questo è un comportamento che deriva direttamente da un'errata interpretazione del documento di testing, laddove si crede che il testing

sia la verifica delle funzionalità del sistema, mentre il vero processo di testing è la ricerca del maggior numero di errori.

3.4 Il miglior test case è quello che ha la più alta probabilità di scoprire errori non ancora rilevati.

Questo principio è poco pragmatico, in quanto non è possibile a priori valutare la probabilità di scoprire errori da parte di un test case, ma possiamo affermare che si deve sempre tendere ad una produzione limitata il più possibile di test case di qualità, al fine di massimizzare il rapporto tra gli errori scoperti ed i test utilizzati.

Ovviamente non sempre è possibile riuscire a rispettare tutti i principi guida elencati, per cui si cerca, all'interno delle metodologie, di seguire il più possibile le linee guida che essi tracciano.

4. Funzionalità testate e non

Il sistema permette agli utenti di loggare nel proprio account e controllare la propria scheda con gli esercizi , modificare la propria password , fare il logout e visualizzare il proprio profilo. L'istruttore potrà accedere al proprio account , aggiungere , rimuovere e modificare utenti e schede , fare il logout e modificare la propria password. Tutte queste attività richiedono meccanismi dinamici che permettono le diverse operazioni in base a dati persistenti.

4.1 Funzionalità da testare

Le funzionalità riguardano i tre livelli dell'architettura descritta.

4.1.1 View

Le funzionalità che devono essere testate per il livello di visualizzazione sono :

➢ Visualizzazione Scheda

➢ Visualizzazione Sezione

➢ Visualizzazione Profilo Utente/Istruttore

4.1.2 Controller

Tutte le funzionalità del livello di visualizzazione implicano il test parallelo delle componenti dipendenti nel livello di controllo , che forniscono l'elaborazione necessaria affinchè il livello di visualizzazione possa fornire all'utente le informazioni richieste. Le funzionalità da testare sono :

➢ Gestione Utente

➢ Gestione Scheda

4.1.3 Model

Tutte le funzionalità del livello di controllo implicano il test parallelo delle componenti dipendenti nel livello dati , che forniscono l'iterazione con il database. Le funzionalità da testare sono :

➢Connessione

➢Accesso ai dati

5. Pass/Fail Criteria

La fase di testing necessita di criteri formali per la determinazione del successo o dell'insuccesso di un determinato test : Pass Criteria e Fail Criteria.

5.1 Pass Criteria

I pass criteria determinano l'insuccesso del test e quindi la correttezza della componente testata. I pass criteria per il sistema MaiGim sono raggruppati in due categorie :

➢Comportamento atteso

➢Nessun errore rilevato dalla componente

5.2 Fail Criteria

I Fail Criteria determinano il successo del test e quindi il malfunzionamento della componente testata. I fail criteria per il sistema MaiGim sono divisi in :

- ➢Errore rilevato dalla componente
- ➢Comportamento non atteso

6. Approcci

Per il sistema sviluppato sono previste le seguenti fasi di test:

6.1. Test delle unità

Si verifica singolarmente ogni modulo del programma ed è condotta contestualmente alla fase di codifica

Tecnica di test:

White Box Testing

Il White Box Testing permette di ricavare i casi di test in base all'implementazione ed alla struttura del programma. Si focalizza sulla struttura interna della componente. Indipendentemente dall'input, ogni stato

nel modello dinamico dell'oggetto e ogni integrazione tra gli oggetti viene testata.
Vi sono 4 tipi di white-box testing:

➢**Statement Testing** (Algebraic Testing) : Si testano i singoli statement

➢**Loop Testing**:
Provoca l'esecuzione del loop che deve essere saltato completamente.
Loop da eseguire esattamente una volta
Loop da eseguire più di una volta

➢**Path testing**: Assicurarsi che tutti i path nel programma siano eseguiti

➢**Branch Testing** (Conditional Testing) : Assicurarsi che ogni possibile uscita da una condizione sia testata almeno una volta.

Il criterio di copertura che ci indirizza alla scelta dei test case è il Path Testing.

➢garanzia che tutti i cammini indipendenti entro un modulo, siano eseguiti almeno una volta;
➢esecuzione dei rami vero e falso per ogni decisione logica;
➢esecuzione di tutti i cicli nei casi limite e entro i confini operativi;
➢esame della validità di tutte le strutture dati interne.

6.2. Test d'integrazione

Si verifica la corretta integrazione tra un sottoinsieme dei moduli (ricerca di errori nelle interfacce tra i moduli)

Black Box Testing :

L'obiettivo è quello di determinare se il programma fa quello che si suppone debba fare in base ai requisiti funzionali. Per la costruzione dei test ci si avvale solo delle specifiche dei requisiti, ignorando completamente come sia stato realizzato il sistema al suo interno. Se ben condotto riesce a determinare il manifestarsi di molti malfunzionamenti funzionali e può evidenziare il mancato rispetto di qualche requisito funzionale.

7. Sospensione e Ripristino

La fase di testing di MaiGim può essere interrotta e ripresa più volte , se il fine ultimo è di rendere il sistema corretto ed ogni funzionalità.Di seguito sono riportati i criteri secondo cui è necessario sospendere la fase di testing e le modalità secondo cui deve poi essere ripresa.

7.1 Criteri di Sospensione

La sospensione della fase di testing deve avvenire qualora un test abbia esito positivo, ovvero si è riscontrato un errore all'interno di una componente.

7.2 Criteri di ripristino

La ripresa della fase di testing avviene a partire dal test case che ne ha causato la sospensione, soltanto dopo che gli sviluppatori hanno corretto l'errore riscontrato.

8. Risorse

Per i testing delle unità e dell'integrazione abbiamo usato Karma e Jasmine.

➤Karma è uno strumento da riga di comando JavaScript che può essere utilizzato per generare un server Web che carica il codice sorgente dell'applicazione e esegue i test. Si può configurare Karma in modo che venga eseguito su un certo numero di browser, il che è utile per essere sicuro che l'applicazione funzioni su tutti i browser che devono essere supportati. Karma viene eseguito sulla riga di comando e mostrerà i risultati dei test sulla riga di comando una volta eseguiti nel browser , è un'applicazione NodeJS e dovrebbe essere installata tramite npm / yarn. Le istruzioni complete per l'installazione sono disponibili sul sito web di Karma.

➤Jasmine è un framework di test javascript che supporta una pratica di sviluppo del software chiamata Behavior Driven Development, o BDD in breve. Jasmine fornisce funzioni per aiutare a strutturare i test e anche a fare asserzioni. Man mano che i test crescono, mantenerli ben strutturati e documentati è vitale, e Jasmine aiuta a raggiungere questo obiettivo.

Per usare Jasmine con Karma, abbiamo usato il karma-jasmine test runner.

9. Test Case

I test case sono diretti a scoprire eventuali malfunzionamenti o comportamenti errati da parte del sistema. Per fare ciò è necessario testare il sistema su diversi istanze di input, ognuna diretta a testare comportamenti del sistema in determinate condizioni, un tipo o classe di input piuttosto che un'altra.
Il testing viene quindi strutturato in base alle funzionalità fornite dal sistema. Per ognuna di queste verranno fornite delle istanze di input che costituiscono i test case dei vari scenari. I test case potranno appartenere a tre diverse categorie di input in base al tipo di dati da cui sono costituiti relativamente allo use case in esame, che possono essere:

- ➢Grammaticalmente non validi: fanno parte di questa categoria gli input che contengono caratteri non validi, che sono a loro volta suddivisi in classi di equivalenza.

- ➢Logicamente non validi: fanno parte di questa categoria i dati che potrebbero essere grammaticalmente validi ma che nel contesto non lo sono, ad esempio in un form che richiede l'inserimento di due numeri per la definizione di un range di valori, due input logicamente non validi possono essere un lower bound (il valore inferiore del range)
  più alto del upper bound (il valore superiore del range).

- ➢Validi: fanno parte di questa categoria i dati validi, cioè che sono grammaticalmente validi e hanno senso nel contesto in cui sono utilizzati.

Quindi per ogni funzionalità verranno presentati un insieme di test. Ognuno di questi insiemi è diviso a sua volta in tre sottoinsiemi:

- ➢ Input grammaticalmente non validi, ma logicamente validi;
- ➢ Input logicamente non validi, ma grammaticalmente validi;
- ➢ Input grammaticalmente e logicamente validi;

9.1 Login

Le funzionalità da testare riguardano una componente del sistema dell'utente e dell'istruttore.

9.1.1 Classi di Equivalenza

Input	Email , Password	
Classi Valide	CE01	Stringa alfanumerica di lunghezza arbitraria
Classi Non Valide	CE02	Stringa non conforme ad email o vuota.

9.1.2 Considerazioni e supposizioni relative al sistema :

Nel database del sistema :

- ➤è presente un account con email : " luca@luca.it " e password "123456"
- ➤non è presente l'account con email : " gigginone@gmail.it "

9.2 Personal Trainer

Di seguito vengono testate tutte le funzionalità riguardanti il sottosistema Utente

9.2.1 Inserisci Utente

Input	Nome , Cognome	
Classi valide	CE03	Stringa di caratteri arbitraria
Classi non valide	CE04	Stringa vuota

Input	Email	
Classi valide	CE05	Stringa alfanumerica arbitraria
Classi non valide	CE06	Stringa non conforme ad email o vuota.

Input	Nickname	
Classi valide	CE07	Stringa alfanumerica arbitraria
Classi non valide	CE08	Stringa vuota

9.2.2 Modifica utente

Input	Nome , Cognome	
Classi valide	CE03	Stringa di caratteri arbitraria
Classi non valide	CE04	Stringa vuota

Input	Email	
Classi valide	CE05	Stringa alfanumerica arbitraria
Classi non valide	CE06	Stringa non conforme ad email o vuota.

9.2.3 Aggiungi Esercizio

Input	Nome	
Classi valide	CE07	Stringa di caratteri arbitraria
Classi non valide	CE08	Stringa vuota.

10. Testing Schedule

Di seguito sono elencate la gestione dei rischi che occorre seguire durante la fase di testing, l'organizzazione delle attività di testing e la schedulazione delle attività durante il periodo prestabilito per il testing.

10.1 Gestione dei rischi

I rischi generati dalle attività di testing sono stati minimizzati , 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 MaiGim sono previste per il seguente periodo :

- Inizio testing : 2 Gennaio 2018
- Fine testing : 10 Gennaio 2018