TEST PLAN

Sommario

1. Introduzione

UniQuestions nasce con l'obiettivo di ottimizzare il servizio di formazione offerto dall'Università degli Studi di Salerno, sviluppando un progetto con lo scopo di fornire uno strumento di interazione con gli studenti, utile per lo scambio di domande e valutazioni relative ai vari argomenti spiegati a lezione in modo tale da migliorare sempre di più la didattica all'interno dei vari insegnamenti.

Scopo principale della fase di testing è quello di controllare che i requisiti funzionali definiti in fase di analisi siano effettivamente rispondenti alle aspettative. L'obiettivo di questa fase è quindi quello di trovare quanti più fault possibili, in maniera da poter migliorare il sistema prima di rilasciarlo completamente al Dipartimento d'Informatica dell'Università degli Studi di Salerno.

2. Riferimenti

- RAD_UniQuestions
- SDD_UniQuestions

3. System Overview

Come specificato nel documento di design (nel paragrafo 3 System Architecture) il sistema è stato suddiviso in 3 sottosistemi: Interface layer, Application layer e Data Storage layer. Ogni sottosistema si occuperà di una logica diversa. Il layer di Interfaccia ha lo scopo di costruire l'interfaccia utente. Il layer di Application si occuperà di implementare tutti i requisiti funzionali specificati nel RAD. Infine, il layer del Data Storage si occuperà di salvaguardare i dati dei Progetti e degli utenti e di mantenerli in memoria al sicuro.

4. Componenti da testare

Il software che s'intende testare come anticipato nei paragrafi precedenti è UniQuestions che presenta come la maggior parte di tutti i prodotti software delle aree di estrema criticità, quelle identificate nel nostro sistema sono le componenti sulle quali si effettuerà il testing:

- Gestione Utente
- Gestione Corsi Insegnamento
- Gestione Lezioni
- Gestione Domande

Di seguito sono elencate le varie unità delle componenti dell'Application Logic Layer.

4.1 Gestione Utente

Componente	Ruoli applicabili	Livello di rischio
Registrazione utente	Utente	Alto
Login	Utente	Alto
Logout	Utente	Basso
Reset password	Utente	Alto
Visualizza profilo	Utente	Basso

4.2 Gestione Corsi d'insegnamento

Componente	Ruoli applicabili	Livello di rischio

Inserisci corso d'insegnamento	Amministratore	Alto
Elimina corso d'insegnamento	Amministratore	Medio
Visualizza corsi d'insegnamento	Utente	Basso
Iscrizione corso d'insegnamento	Studente	Basso
Disiscrizione corso d'insegnamento	Studente	Basso

4.3 Gestione Lezioni

Componente	Ruoli applicabili	Livello di rischio
Aggiungi lezione	Docente	Alto
Elimina lezione	Docente	Medio
Visualizza lezioni	Utente	Basso
Inserisci valutazione lezione	Studente	Medio
Visualizza media valutazioni	Docente	Basso

4.3 Gestione Domande

Componente	Ruoli applicabili	Livello di rischio
Invio domanda	Studente	Alto
Visualizza risposte	Docente	Basso
Visualizza domande	Studente	Basso
Invia risposta	Docente	Alto
Ricerca AQ	Studente	Alto

5. Componenti da non testare

Le componenti di UniQuestions che non sono state testate sono le seguenti:

- Logout
- Visualizza Profilo
- Elimina corso d'insegnamento
- Visualizza corsi d'insegnamento
- Iscrizione corso d'insegnamento
- Disiscrizione corso d'insegnamento
- Elimina lezione
- Visualizza lezioni
- Inserisci valutazione lezione

- Visualizza media valutazioni
- Visualizza risposte
- Visualizza domande

Per tali componenti non si è ritenuto effettuare testing perché è stato associato un basso o medio livello di rischio.

6. Approccio

L'approccio scelto per la fase di testing si è basata sulla suddivisione del testing in due fasi: testing di unità, e testing di sistema. In questo modo è possibile controllare in maniera efficiente ogni sottosistema implementata e di trovare e corregge eventuali bug prima dell'inserimento all'interno del sistema.

6.1 Test di unità

In questa fase il testing viene effettuato testando ogni singola funzione implementata nel Layer Data Storage. Tramite il framework Junit su Eclipse verranno quindi testati i metodi nei Manger dei 4 Sottosistemi: Gestione Utente, Gestione CorsoInsegnamento, Gestione Lezione e Gestione Domande.

6.2 Testing di sistema

In questa fase il testing viene effettuato in maniera dinamica andando ad utilizzare la tecnica di Black-Box.

6.3 Testing di integrazione

In questa fase il testing viene effettuato sull'Application layer e quindi su tutte le Servlet del nostro sistema. Siccome l'Application Layer comunica con il layer di DataStorage e quello di Interface, in ogni test case, abbiamo fatto interagire i servizi offerti dai livelli in questione.

7. Criteri di successo/errore

Il testing ha successo se l'output osservato è diverso dall'output atteso: ciò significa che parliamo di successo se il test individuerà una failure. In tal caso questa verrà analizzata e, se legata ad un fault, si procederà alla sua correzione. Sarà infine iterata la fase di testing per verificare che la modifica non abbia impattato su altri componenti del sistema. Viceversa parliamo di fallimento se il test non riesce ad individuare un errore.

8. Testing Materials

Come supporto alla fase di testing di Sistema si utilizzerà un browser Internet, la scelta è ricaduta su Google Chrome. Come strumenti di supporto verrà utilizzato: il framwork Selenium aggiungendo a Google Chrome Selenium IDE.

Per la fase di testing di integrazione abbiamo utilizzato il framework JUnit e spring.

Per la fase di testing di unità abbiamo utilizzato il framework JUnit.

9. Testing Materials

I test case fanno riferimento al documento: "Testing Case Specification"