Test Plan

Pr. D. - Programmazione Didattica

Riferimento	
Versione	1.1
Data	19/02/2020
Destinatario	Top Management
Presentato da	Carpentiero Edoardo, Della Greca Attilio, Prisco Antonio
Approvato da	

Sommario

1. Introduzione

Il **testing** è una tecnica di "fault detection" il cui scopo è appunto quello di rilevare errori in maniera pianificata all'interno del codice prodotto. L'obiettivo è, quindi, di evitare che questi errori si presentino in fase di utilizzo del sistema da parte dell'utente finale.
L'attività di testing è una componente fondamentale nello sviluppo di un prodotto software efficiente e funzionale perché permette di rilevare possibili errori all'interno del sistema, che potrebbero verificarsi in seguito alla scrittura del codice sorgente.
In questo documento vengono pianificate e descritte in modo dettagliato le attività di testing che il team si prepara ad effettuare riguardo alle attività di gestione del carico didattico.

2. Relazione con altri documenti

I test cases sono basati sulle funzionalità individuate nel documento *Master Document* **Prd.D._MD_Vers1.3.** La combinazione e il numero dei test case è riportato nel documento **Prd.D._CP_Vers1.0.** I test case individuati per ogni funzionalità sono riportati nel documento Test Case Specification - **Prd.D._TCS_Vers1.1**. I log di esecuzione dei test case sono riportati nel documento *Test Execution Report* - **Prd.D._TER_v1.3 -** e gli eventuali errori riscontrati durante il testing sono riportati nel documento *Test Incident Report* - **Prd.D._TIR_v1.0**

3. Panoramica del sistema

Il sistema risulta essere suddiviso in tre sottosistemi in cui ognuno di essi è composto da moduli che si occupano di svolgere determinate attività. Tra questi, i sottosistemi presi in considerazione nella change request sono:

- **Sottosistema Presidente:**
  - Gestione autenticazione
  - Gestione docenti
  - Gestione insegnamento
  - Gestione ordinamento
  - Gestione regolamento
  - Gestione programmazione didattica
  - **Gestione carico didattico (new)**
  - Gestione account Docenti

- **Sottosistema Docente:**
  - ➢ Visualizza docenti
  - ➢ Visualizza insegnamento
  - ➢ Visualizza ordinamento
  - ➢ Visualizza regolamento
  - ➢ Visualizza programmazione didattica
  - ➢ Visualizza carico didattico personale
  - ➢ **Gestione carico didattico (new)**
  - ➢ Gestione account personale

4. Funzionalità da testare

Le funzionalità considerate nel testing riguardano la gestione del carico didattico presenti nel sottosistema Presidente e Docente. Le funzionalità esterne da testare sono le seguenti:

FUNZIONALITÀ'	DESCRIZIONE
Visualizza insegnamenti proposti	Consente all'utente di visualizzare gli insegnamenti proposti e i relativi stati
Visualizza insegnamenti disponibili	Consente all'utente di visualizzare gli insegnamenti disponibili
Proponi Insegnamento disponibile	Consente all'utente di proporre un insegnamento disponibile
Valuta insegnamento proposto	Consente all'utente di cambiare lo stato dell'insegnamento proposto
Visualizza status carico didattico dei docenti	Consente all'utente di visualizzare lo status del carico didattico dei docenti
Visualizza informazioni insegnamento proposto o disponibile	Consente all'utente di visualizzare le informazioni dell'insegnamento proposto o disponibile

Per ognuna di queste funzionalità esterne, sono state individuate le principali funzionalità interne da testare, riportate nella tabella seguente, specificate nell'Object Design della classe GestioneCaricoDidattico descritta nel documento **Prd.D._MD_Vers1.3.**

Funzionalità esterna	Funzionalità interna/e principali
Visualizza insegnamenti proposti	getInsegnamentiAssociatiAlDocente()
Visualizza insegnamenti disponibili	getInsegnamentiDisponibili()
Proponi Insegnamento disponibile	proponiInsegnamento()
Valuta insegnamento proposto	cambiaStatoAssociazione()
Visualizza status carico didattico dei docenti	getDocentiPrD()
Visualizza informazioni insegnamento proposto o disponibile	getInfoDocentiInsegnamento()

5. Criteri Pass/Failed

Lo scopo del testing è quello di dimostrare la presenza di faults (errori), ovvero comportamenti non ottimali da parte del sistema.
La fase di test avrà successo se individuerà una failure, cioè se l'output osservato sarà diverso da quello atteso. Ogni qual volta verrà individuata una failure, questa verrà analizzata e, se legata ad un fault, si procederà alla sua correzione. Una volta completata la correzione, si procederà, in modo iterativo, ad una nuova fase di test per verificare che la modifica non ha impattato su altri componenti del sistema. Al contrario, il testing fallirà se l'output osservato sarà uguale all'oracolo.

6. Approccio

La tecnica di testing utilizzata si basa sull'approccio di tipo BlackBox.
Sarà praticato uno **unit testing**, per verificare il corretto funzionamento di tutte le singole unità di codice implementate in determinate condizioni. Sarà quindi testata ogni nuova classe introdotta ed ogni metodo appartenente ad essa.
Dopo la fase di unit testing seguirà l'**integration testing**, con l'obiettivo di collaudare l'interazione tra le nuove componenti e quelle pre-esistenti. Dopodiché verrà effettuato il **system testing** per dimostrare che il sistema soddisfa i requisiti definiti dalla change request, e infine il **regression testing** per verificare se le funzionalità preesistenti funzionano correttamente e che i nuovi cambiamenti non abbiano introdotto nuovi bug.
Al fine di minimizzare il numero dei test cases, è stato utilizzato il *category partition*.

6.1 Testing di unità

Durante questa fase l'attenzione viene focalizzata sulle nuove componenti implementate relative alla gestione del carico didattico. In particolare, vengono ricercate condizioni di fallimento isolando le componenti tramite utilizzo di **moduli Stub** (implementazione completa o parziale di una componente dalla quale dipende la componente sotto testing) e **moduli Driver** (implementazione completa o parziale di una componente che fa uso della componente sotto testing). Tali test vengono applicati dopo aver isolato ogni singola funzionalità della gestione del carico didattico da testare in modo da verificare una completa validità delle componenti e tener traccia dei possibili errori che vengono riscontrati. Se si verifica un errore con dei risultati inattesi si interviene in maniera tempestiva sulla componente in modo da renderla correttamente funzionante e procedere con le fasi di testing successive.

6.2 Testing di integrazione

La modalità utilizzata nel testing di integrazione si basa sul testing incrementale, dove le componenti che precedentemente erano state isolate e testate singolarmente vengono integrate man mano. La tecnica di approccio a questo tipo di operazione prevede l'uso combinato di strategie Bottom-Up e Top-Down, in particolare una strategia Sandwich Testing. Questo approccio mira principalmente a focalizzare la ricerca di errori nelle interfacce di comunicazione tra sottosistemi.

6.3 Testing di sistema

Tale testing permette di verificare la corretta adesione dei requisiti dettati dalla change request con il resto del sistema. L'approccio utilizzato sarà di tipo Blackbox. Sarà verificata la qualità e l'affidabilità del prodotto software, in modo che possa essere utilizzato dall'utente finale senza margini di errori.

6.4 Testing di regressione

Lo scopo è quello verificare se le funzionalità preesistenti del sistema abbiano preservato lo stesso comportamento anche dopo che la change request è stata implementata. Al fine di verificare ciò verifica bisogna rieseguire i test case su tali funzionalità in modo da verificare se quest'ultime presentano o meno malfunzionamenti prima della modifica.

7. Testing pre-modifica

Al fine di garantire il corretto funzionamento della piattaforma, verranno effettuati dei test di sistema per verificare il funzionamento pre-implementazione della change request. Verranno realizzati attraverso *Selenium* test automatizzati che verranno eseguiti sulle componenti considerate nell'impact analysis, cioè quelle presenti nel modulo **Gestione Programmazione Didattica**, il quale è risultato essere il principale componente ad essere impattato dalla change request. A conclusione di ciò, i requisiti funzionali preesistenti che verranno verificati saranno i seguenti:

- **RF_5 Gestione Programmazione Didattica**
  - RF_5.1 Creare Programmazione Didattica
  - RF_5.2 Modificare Programmazione Didattica
  - RF_5.3 Visualizzare Programmazione Didattica
  - RF_5.5 Visualizzare Monte Ore Docente
  - RF_5.6 Cambiamento Stato Programmazione Didattica
  - RF_5.8 Visualizza Carico Didattico Personale

8. Sospensione e ripristino

Il testing sarà **sospeso** quando tutti i test case riportati in **Prd.D._TCS_Vers1.1** saranno stati eseguiti con successo e la coverage del nuovo modulo avrà raggiunto una percentuale maggiore o uguale a **76**.
La **ripresa** del testing avverrà in seguito a delle modifiche che potrebbero introdurre nuovi errori all'interno del sistema.

8. Strumenti per il testing

La dotazione software richiesta per l'attività di testing è la seguente:

- ***PHP Unit*** utilizzato per lo sviluppo in locale del software. Tale strumento è impiegato sia nel testing di unità che di integrazione.
- ***Selenium,*** utilizzato per poter effettuare il testing di sistema.
- Un qualunque browser Web tra i vari *Firefox, Google Chrome, Internet Explorer, Safari, Opera, ecc*.
- **Intellij IDEA Coverage Function**: Questa funzionalità integrata all'interno dell'IDE permette di controllare la percentuale di metodi, classi, linee di codice coperte dai casi test creati.