Test Plan

Gestione Prenotazioni

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 1.0 |
| Data | 07/06/2021 |
| Destinatario | Prof.re De Lucia |
| Presentato da | Aldo Claudini e Emilio Schiavo |

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | autore |
| 07/06/2021 | 1.0 | Stesura del documento TestPlan con :  -introduzione  -funzionalità da testare  -approccio | Aldo Claudini |

**Indice**

[1. Introduzione 4](#_Toc73976595)

[2. Documenti correlati 4](#_Toc73976596)

[2.1 Master Document (MD) 4](#_Toc73976597)

[2.2 Impact Analysis (IA) 4](#_Toc73976598)

[2.3 TestCase (TC) 5](#_Toc73976599)

[3. Funzionalità da testare 5](#_Toc73976600)

[4. Criteri Pass/Field 6](#_Toc73976601)

[5. Approccio 6](#_Toc73976602)

[5.1 Testing di unità 6](#_Toc73976603)

[5.2 Testing di integrazione 6](#_Toc73976604)

[5.3 Testing di sistema 7](#_Toc73976605)

# Introduzione

In questo documento viene specificata la pianificazione della attività di testing del sistema Gestione Prenotazione al fine di verificare se esistono differenze fra il comportamento atteso e il comportamento reale del sistema. In questa attività andremo a rilevare gli eventuali errori prodotti dal codice scritto, per evitare che essi si presentino nel momento in cui il sistema viene messo in esercizio. Le attività di testing sono state pianificate per i seguenti sottosistemi:

* Assegnazione/rimozione Aula per i tutor
* Asegnazione /rimozione Tutor
* Visualizzazione Tutor

# Documenti correlati

Il test plan è strettamente collegato ai documenti prodotti fino ad ora e alla documentazione esistete, poiché prima di passare alla fase di testing abbiamo bisogno di avere una gran parte delle funzioni previste già implementate. Di seguito verranno descritte le relazioni che ci sono fra il test plan e gli altri documenti

## 2.1 Master Document (MD)

La relazione tra test plan e MD riguarda in particolare i requisiti funzionali e non funzionali del sistema poiché i test che saranno eseguiti solo sulle nuove funzionalità implementate terranno conto delle specifiche espresse nel MD.

## Impact Analysis (IA)

Nell’ impact analysis sono state individuate tutte le componenti del sistema che verranno sottoposte a modifiche. Queste componenti, insieme a quelle ex novo dovranno essere testate utilizzando attività di testing funzionale.

## TestCase (TC)

Al fine di valutare il reale impatto delle nuove modifiche, bisogna fare test di regressione, sono stati creati tutti i test case delle nuove funzionalità.

# Funzionalità da testare

Per garantire consistenza con gli artefatti del sistema esistenti gli idetificativi dei test case seguono la numerazione adottata nel documento TestPlan.

Di seguito vengono elencate le funzionalità da testare per il sottosistema designato:

|  |
| --- |
| TC\_GEAD 2.6: assegna/rimuovi tutor  TC\_GEAD 2.7: Assegna/rimuovi aula tutor  TC\_GEAD 2.8: Visualizza tutor |
| TC\_GEAD 2.6: : assegna/rimuovi tutor   * TC\_GEAD 2.6.1: Assegna Tutor * TC\_GEAD 2.6.2: Rimuovi Tutor   TC\_GEAD 2.7: Assegna/Rimuovi aula tutor   * TC\_GEAD 2.7.1: Assegna Aula tutor * TC\_GEAD 2.7.2: Rimuovi Aula tutor |
| TC\_GEAD 2.8: Visualizza tutor   * TC\_GEAD 2.8.1: Visualizza tutor |

Ogni qualvolta sarà testata una di queste nuove funzionalità elencate sopra, verrà svolto test di regressione per verificare se sono state aggiunte delle failure all’interno delle funzionalità del vecchio sistema.

# **Criteri Pass/Field**

I dati di input del test saranno suddivisi in classi di equivalenza, ovvero saranno raggruppati in insiemi con caratteristiche comuni, in modo tale da poter testare solo uno degli elementi della classe. L’input avrà superato il test se l’output risultante è quello atteso, il risultato atteso sarà quello ottenuto utilizzando come oracolo la versione precedente del software.

# Approccio

La fase di testing sarà suddivisa in tre fasi:

1. **Testing di unità:**

verrà testata nello specifico il funzionamento di ogni singola unità del sistema;

1. **Testing di integrazione:**

dove verranno testate le interfacce delle suddette unità;

1. **Testing di sistema:**

dove verranno testato l’intero sistema assemblato.

## 5.1 Testing di unità

Durante questa fase verranno ricercate le condizioni di fallimento, isolando i componenti ed usando test drive e stub. La strategia usata per il testing è la tecnica Black-Box, che si concentra sul comportamento Input/Output ignorando la struttura interna della componente. Per minimizzare il numero di test cases i possibili input verranno partizionati in classi di equivalenza e per ogni classe verrà usato un test case. Gli errori scovati in questa fase, e anche in tutte le altre, devono essere comunicati agli sviluppatori in modo tale che questi possano correggere l’errore e ripristinare la fase di testing al più presto.

I fallimenti identificati durante la fase di testing verranno specificati nel **test incidenti report.**

## 5.2 Testing di integrazione

Questa fase ha lo scopo di integrare tutte le componenti di una funzionalità al fine di testarle nel complesso utilizzando una strategia Bottom-Up. L’approccio Bottom-Up mira a ridurre le dipendenze fra funzionalità differenti e a facilitare la ricerca di errori nelle interfacce di comunicazione tra sottosistemi.

## 5.3 Testing di sistema

In questa ultima fase vogliamo dimostrare che il sistema soddisfi tutti i requisiti richiesti. Si cercherà di testare le funzionalità più utili per l’utente e quelle che hanno una maggiore probabilità di fallimento.