Test Plan

Gestione Prenotazioni

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 1.1 |
| Data | 08/06/2021 |
| 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, Emilio Schiavo |
| 08/06/2021 | 1.1 | Aggiornamenti e inserimento dei nuovi test case | Aldo Claudini, Emilio Schiavo |

**Indice**

[1. Introduzione 4](#_Toc74233705)

[2. Documenti correlati 4](#_Toc74233706)

[2.1 Master Document (MD) 4](#_Toc74233707)

[2.2 Impact Analysis (IA) 4](#_Toc74233708)

[2.3 Test Case Specification (TCS) 4](#_Toc74233709)

[3. Funzionalità da testare 5](#_Toc74233710)

[4. Criteri Pass/Fail 5](#_Toc74233711)

[5. Approccio 5](#_Toc74233712)

[5.1 Testing di unità 6](#_Toc74233713)

[5.2 Testing di integrazione 6](#_Toc74233714)

[5.3 Testing di sistema 6](#_Toc74233715)

[6. Test Cases 6](#_Toc74233716)

[6.1 Assegna/rimuovi aula tutor 6](#_Toc74233717)

[6.1.1 Assegna aula tutor 6](#_Toc74233718)

[6.1.2 Rimuovi aula tutor 7](#_Toc74233719)

[6.2.1 Assegna Tutor 8](#_Toc74233720)

[6.2.2 Rimuovi Tutor 8](#_Toc74233721)

[6.3 Visualizza Tutor 8](#_Toc74233722)

# 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 requisiti:

* Assegnazione/rimozione Aula per i tutor;
* Assegnazione /rimozione Tutor;
* Visualizzazione Tutor.

# Documenti correlati

Il test plan è strettamente correlato ai documenti prodotti fino ad ora e alla documentazione esistente.  
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 sviluppate ex-novo, dovranno essere testate utilizzando attività di testing funzionale.

## Test Case Specification (TCS)

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

# Funzionalità da testare

Per garantire consistenza con gli artefatti del sistema esistenti, gli identificativi dei test case seguono la numerazione adottata nel documento Test Plan, coerente con quanto già stabilito nel RAD.

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, verrà svolto test di regressione per verificare se sono state aggiunte failure nell’erogazione delle funzionalità del sistema.

# **Criteri Pass/Fail**

I dati di input del test saranno suddivisi secondo la tecnica dell’Equivalence class partitioning, la quale prevede il raggruppamento dei possibili input in insiemi equivalenti con caratteristiche comuni, in modo tale da poter testare solo uno degli elementi della classe che avrà comportamento uguale a tutti gli input appartenenti alla categoria. Il test verrà considerato “Passed” per un determinato input se l’output risultante coinciderà con quanto atteso. L’oracolo verrà identificato considerando 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 singolo componente aggiunto al sistema in questa release, in modo da testare nello specifico la correttezza di ciascuna unità;

1. **Testing di integrazione:**

test per verificare il corretto funzionamento delle interfacce e la interoperabilità dei vari sottosistemi;

1. **Testing di sistema:**

dove verrà esaminato il funzionamento dell’intero sistema assemblato e pronto alla nuova release.

## 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 sui dati di 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 individuato un test case.

Eventuali anomalie rilevate andranno comunicate agli sviluppatori in modo da permettere una pronta correzione e un rapido ripristino della fase di testing.

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

## 5.2 Testing di integrazione

Questa fase ha come oggetto di testing tutte le componenti di una funzionalità già integrate, 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.

# Test Cases

Di seguito verranno mostrati i test case individuati per garantire correttezza di erogazione delle nuove funzionalità.

## 6.1 Assegna/rimuovi aula tutor

### 6.1.1 Assegna aula tutor

Category Partition:

|  |  |
| --- | --- |
| Parametro: aula | |
| Esiste Aula tutor [AA] | Non esiste l’aula [error]  2. Esiste l’aula [AA\_OK] |
| Parametro: edificio | |
| Esiste Edificio tutor [EA] | Non esiste l’edificio [error]  2. Esiste l’edificio [EA\_OK] |
| Parametro: tipoaula | |
| setTipoAula [TA] | 1. =2 [property TA\_OK]  2. =3 [property TA\_OK]  3. <0 or =>4 [Error] |

Test cases:

|  |  |  |
| --- | --- | --- |
| Codice | Combinazione | Esito |
| TC\_AsAutut\_01 | AA, EA, TA | Aggiunta prenotabilità aula tutor |
| TC\_AsAutut\_02 | AA, EA | Error |
| TC\_AsAutut\_03 | AA, TA | Error |
| TC\_AsAutut\_04 | EA, TA | error |
| Valore di ritorno “prenotabilità aula cambiata”. | | |

### 6.1.2 Rimuovi aula tutor

Category Partition:

|  |  |
| --- | --- |
| Parametro: aula | |
| Esiste Aula tutor [RA] | Non esiste l’aula [error]  2. Esiste l’aula [RA\_OK] |
| Parametro: edificio | |
| Esiste Edificio tutor [RA] | Non esiste l’edificio [error]  2. Esiste l’edificio [EA\_OK] |
| Parametro: tipoaula | |
| SetTipoAula [RA] | 1. =0 [property TA\_OK]  2. =1 [property TA\_OK]  3. <0 or =>4 [Error] |

Test cases:

|  |  |  |
| --- | --- | --- |
| Codice | Combinazione | Esito |
| TC\_RimAutut\_01 | AA, EA, TA | Rimuovi prenotabilità aula tutor |
| TC\_RimAutut\_02 | AA, EA | Error |
| TC\_RimAutut\_03 | AA, TA | Error |
| TC\_RimAutut\_04 | EA, TA | error |
| Valore di ritorno “prenotabilità aula cambiata”. | | |

### 6.2.1 Assegna Tutor

Category Partition:

|  |  |
| --- | --- |
| Parametro: e-mail | |
| EmailTutorExist [ETC] | Non esiste l’email [error]  2. Esiste l’email [ETC\_OK] |
| Parametro: corso | |
| CorsoExist [CE] | Non esiste il corso [error]  2. Esiste il corso [CE\_OK] |

Test cases:

|  |  |  |
| --- | --- | --- |
| Codice | Combinazione | Esito |
| TC\_AssegnaTutor\_01 | ETC | Error |
| TC\_AssegnaTutor\_02 | CE | Error |
| TC\_ AssegnaTutor\_03 | ETC, CE | Corretto |
| Valore di ritorno “Riassegnamento Avvenuto”. | | |

### 6.2.2 Rimuovi Tutor

Category Partition:

|  |  |
| --- | --- |
| Parametro: email | |
| EmailTutorExist [ETC] | Non esiste l’email [error]  2. Esiste l’email [ETC\_OK] |

Test cases:

|  |  |  |
| --- | --- | --- |
| Codice | Combinazione | Esito |
| TC\_RimuoviTutor\_01 | ETC | Error |
| TC\_ RimuoviTutor\_02 | ETC | Corretto |
| Valore di ritorno “Modifica avvenuta con successo”. | | |

### 6.3 Visualizza Tutor

Non è stato previsto il testing di questa funzionalità in quanto non ci sono campi di input da testare.