|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 0.1 |
| Data | 14/01/2020 |
| Destinatario | Prof.ssa F. Ferrucci |
| Presentato da | Marco Delle Cave, Francesco Pagano,  Manuel Pisciotta, Alessia Olivieri |
| Approvato da |  |



Test Summary Report

TutoratoSmart

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Cambiamenti** | **Autori** |
| 14/01/2020 | 0.1 | Prima stesura | Pisciotta Manuel |

Sommario

[1. Introduzione 3](#_Toc31454413)

[2. Riferimenti 3](#_Toc31454414)

[3. Approccio utilizzato 3](#_Toc31454415)

[3.1 Unit Testing 3](#_Toc31454416)

[Black-Box Testing: 3](#_Toc31454417)

[White-Box Testing: 3](#_Toc31454418)

[3.2 Integration Testing 3](#_Toc31454419)

[3.3 System Testing 4](#_Toc31454420)

[3.4 Continous Integration 4](#_Toc31454421)

# 1. Introduzione

Questo documento viene creato per spiegare le varie attività svolte per testare il sistema “TutoratoSmart”. Le attività di testing eseguite sono relative alle funzionalità di:

* Request\_Management
* Tutoring\_Activity\_Management
* Tutoring\_Request
* Tutoring\_Supervision
* User

Per ogni funzionalità è stato effettuato il testing di unità e di integrazione. Durante l’implementazione è stata adottata una strategia di Continuous Integration. Al termine dell’implementazione, è stato poi effettuato il testing di sistema.

# 2. Riferimenti

* TS\_RAD\_V\_1.1
* TS\_TER\_V\_0.1
* TS\_TP\_V\_0.2
* TS\_TCS\_V\_0.3
* TS\_UTR\_V\_0.1
* TS\_MTR\_V\_0.1

# 3. Approccio utilizzato

## 3.1 Unit Testing

* **Black-Box Testing:**

Come linea guida abbiamo seguito gli use case identificati in precedenza nel RAD. Abbiamo effettuato un test per ogni test-case individuato nel Test Plan e specificato nel Test Case Specification, utilizzando il plugin JUnit e Mockito.

* **White-Box Testing:**

Abbiamo utilizzato la tecnica del branch testing per assicurarci che il codice prodotto funzionasse correttamente e abbiamo scelto come percentuale di branch coverage delle classi di almeno il 75%.

## 3.2 Integration Testing

La strategia adottata per il testing di integrazione è quella di tipo “Bottom-up” la cui strategia prevede che i sottosistemi nel layer più in basso della gerarchia dopo essere stati testati individualmente, vengano testati congiuntamente ai sottosistemi nel layer di livello superiore, cioè il layer della logica e al passo successivo vengano testati i due strati sottostanti con il layer di presentazione.

Siamo quindi partiti col testare le classi del layer Model, che contiene sia i Bean, classi Java che rappresentano le entità memorizzate nel DB, che i DAO, classi che gestiscono l’accesso e modifica dei dati memorizzati nel DB.

Successivamente sono stati testati i sottosistemi del layer Control che utilizzano il layer Model e infine è stato testato l’intero sistema.

**3.3 System Testing**

Per il testing di sistema è stato utilizzato l’IDE Selenium, rif. <https://selenium.dev/selenium-ide/>,

che ha permesso il testing delle funzionalità di input/output principali e la verifica della corrispondenza tra i risultati attesi e quelli effettivi del sistema.

**3.4 Continous Integration**

Durante lo sviluppo della nostra applicazione è stata adottata la pratica della CI attraverso l’impiego del servizio Travis CI, riferimento <https://travis-ci.org/>, che ha permesso l’esecuzione automatica di tutti i test ad ogni commit effettuato tramite Git.

# 4. Risultati

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Numero di componenti testate** | **Numero di errori trovati** | **Numero di errori corretti** | **Numero di componenti non testate** |
| **Model** | 22 | 0 | 0 | 0 |
| **Control** | 24 | 0 | 0 | 2 |

**4.1 Model**

I dettagli sull’esecuzione del testing del layer Model e sulla coverage raggiunta sono disponibili nel documento di test di integrazione “TS\_MTR”.

**4.1.1 Bean**

Sono stati eseguiti 102 test per i bean, non riscontrando errori, e raggiungendo una coverage del 100% delle singole classi.

**4.1.2 DAO**

Sono stati eseguiti 50 test, non riscontrando errori, e raggiungendo una coverage del 93% del package Model e del 100% delle singole classi DAO.

**4.2 Control**

I dettagli sull’esecuzione del testing del layer Control e del testing di unità e sulla coverage raggiunta sono disponibili nel documento di test di unità “TS\_UTR”.

### 4.2.1 Black-box

Sono stati eseguiti 101 test, senza riscontrare errori.

**4.2.2 White-box**

Non sono stati riscontrati errori nel testing white-box.

* Request\_Management: sono stati eseguiti 11 test, raggiungendo una coverage dell’80,5%.
* Tutoring\_Activity\_Management: sono stati eseguiti 8 test, raggiungendo una coverage dell’82.2%.
* Tutoring\_Request: sono stati eseguiti 14 test, raggiungendo una coverage dell’82.1%.
* Tutoring\_Supervision: sono stati eseguiti 11 test, raggiungendo una coverage dell’81.8%.
* User: sono stati eseguiti 10 test, raggiungendo una coverage del 76%.

**4.3 System**

Infine, i dettagli sull’esecuzione del testing di sistema sono disponibili nel documento “TS\_TER”.