|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| USE CASE MODEL  Progetto  STORYTELLING   |  |  | | --- | --- | | Riferimento |  | | Versione | 0.2 | | Data | 24/10/2021 | | Destinatario | Prof. Gravino | | Presentato da | Alessandro Marigliano, Antonio Scotellaro, Emmanuele Virginio Coppola, Muriel Rossi. | | Approvato da |  | |

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 03/12/2021 | 0.1 | Prima stesura | Alessandro Marigliano, Emmanuele Virginio Coppola |
| 04/12/2021 | 0.2 | Aggiunta di Test Cases | Alessandro Marigliano, Emmanuele Virginio Coppola |
| 07/12/2021 | 0.3 | Correzione errori | Antonio Scotellaro |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Sommario

[1 Introduzione 4](#__RefHeading___Toc6771_3511940610)

[2 Relazione con altri documenti 4](#__RefHeading___Toc6773_3511940610)

[3 Panoramica Di Sistema 4](#__RefHeading___Toc361_4100221823)

[4 Feature da Testare 5](#__RefHeading___Toc6781_3511940610)

[5 Criteri di Fallimento/Successo 5](#__RefHeading___Toc6783_3511940610)

[6 Approccio 6](#__RefHeading___Toc6785_3511940610)

[7 Sospensione e Ripristino 7](#__RefHeading___Toc6787_3511940610)

[8 Materiale di Testing 8](#__RefHeading___Toc6789_3511940610)

[9 Test Cases 8](#__RefHeading___Toc6791_3511940610)

[10 Testing schedule 13](#__RefHeading___Toc6793_3511940610)

# 1 Introduzione

StoryTellling è un software che si impegna nel rendere l’espressione di idee tra i sugli utenti in maniera semplice ed efficace.

Il documento di Test Plan ha lo scopo di mettere in chiaro l’organizzazione l’attività di testing per assicurarsi che le funzionalità di sistema agiscano nella maniera corretta.

All’interno del documento sono documentate le strategie e gli strumenti impiegati per il testing per assicurarsi che il sistema risulti privo di errori.

Le Attività di testing descritte in questo documento sono dirette verso le seguenti gestioni:

* Gestione Utente
* Gestione Bacheca

# 2 Relazione con altri documenti

Per la corretta individuazione dei Test Case si faccia riferimento agli altri documenti prodotti:

**Relazioni con il Requraments Analisis Document(RAD)**

I Test Case pianificati nel Test Plan sono pianificati in base ai requisiti funzionali e non funzionali descritti nel RAD.

**Relazioni con il System Design Document(SDD)**

I Test Case Pianificati devono Rispettare la Suddivisione dei sottosistemi descritti all’interno del SDD.

**Relazioni con il Object Design Document(ODD)**

Il documento di object design document non è stato ancora formalizzato al tempo di scrittura di questo documento

# 3 Panoramica Di Sistema

Il Sistema proposto basa la sua architettura sul modello Model View Control.

Verranno usati HTML5, CSS3, AJAX e JavaScript per la parte front-end e la generazione delle view.

Per la logica applicativa e quindi il back-end sarà utilizzato Java nella sua versione OpenJdk ??.

Per la gestione del database verranno usati:

* Java Database Connector per il collegamento al database.
* MariaDB come database in fase di produzione e deployment.

# 4 Feature da Testare

Il testing verrà effettuato per le varie gestioni:

* + Gestione Account
    - Registrazione Account
    - Login
  + Gestione Bacheca
    - Inserimento Commento
    - Pubblicazione Storia

Le funzionalità di cui non si andrà a fare il testing sono funzionalità che non prevedono l’input manuale di dati da parte dell’utente come la visualizzazione di dati e l’Inserimento Reazione poiché quest’ultimo richiede solo un click su un’icona.

# 5 Criteri di Fallimento/Successo

L’esito di un test case è valutato mediante un oracolo, inteso come risultato della sua esecuzione in base ai requisiti.

Un test ha successo se, dato un input al sistema, l’output ottenuto è diverso dall’output previsto dall’oracolo.

Un test ha fallisce se, dato un input al sistema, l’output ottenuto è uguale all’output atteso dall’oracolo.

Tutto il testing sarà considerato valido se tutti i seguenti vincoli saranno rispettati:

* + Testare tutti i requisiti funzionali ad alta priorità;
  + Effettuare test di regressione ogni volta che si introducono nuove caratteristiche al sistema o vengono modificate quelle presenti;
  + Raggiungere un branch coverage non inferiore al 75%

# 6 Approccio

Il testing dell’intero sistema si compone di 3 fasi: testing di sistema, testing di integrazione e testing di unità. Verranno progettati nell’ordine appena definito e testati nel medesimo ordine.

Prima della fase di implementazione di sistema, avverrà la progettazione dei casi di test di sistema, perfezionati in seguito nella loro fase di esecuzione; durante la fase implementativa sarà presente la progettazione dei casi di test di unità.

Durante lo sviluppo saranno eseguite periodiche attività di revisione sul codice del prodotto.

Poiché la progettazione è organizzata seguendo un modello simile al modello a V, il testing di sistema è stato pianificato in seguito alla stesura del documento Requirament Analysis Document, mentre la pianificazione del testing di integrazione avverrà dopo la stesura del System Design Document.

**Testing di Sistema**

Per il testing di sistema sarà utilizzato il tool Selenium IDE, che permette di registrare le azioni che un utente può intraprendere sul browser, in modo da poter implementare ed eseguire i test case di sistema. Il server per la fase di testing verrà effettuato il deploy su macchina locale identificata con localhost.

**Performance Testing**

A causa del basso budget a disposizione, non si assicura l’esecuzione del performance testing.

**Pilot Testing**

A causa del basso budget a disposizione, non si assicura l’esecuzione del pilot testing.

**Acceptance Testing**

L’acceptance testing verrà effettuato solo sul functional testing, e il Professore Gravino simulerà la figura del cliente.

**Installation Testing**

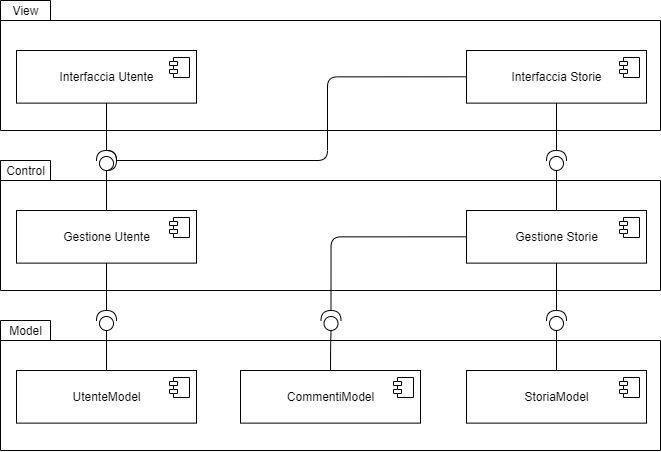
A causa del basso budget a disposizione, non si assicura l’esecuzione dell’installation testing.

**Testing di Integrazione**

Verrà utilizzato un approccio bottom-up, metodo ritenuto più adatto per un software basato sul paradigma Object Oriented

Il test di integrazione sarà il medesimo per tutte le componenti da testare. Nello specifico, si procederà prima con il test delle classi appartenenti al package Model, poi dei delle classi appartenenti al package Control.

Per quanto riguarda il test ecco qui presente uno schema che rappresenta le dipendenze tra i sistemi:



**Testing di Unità**

Per il testing di unità la strategia prevista consiste nel testare ogni metodo delle classi del sistema. Da esse, sono escluse le interfacce e le classi entity, poiché quest’ultime presentano solo metodi get e set. I casi do test saranno definiti attraverso un approccio black-box e saranno documentati direttamente nel codice--------.

-

-

-

# 7 Sospensione e Ripristino

In questa sezione verranno specificati i criteri di sospensione del test e le attività di test che dovranno essere ripetute quando si riprende il test.

**Criteri di sospensione**

Il testing non verrà sospeso fino alla sua terminazione, anche in caso di rivelazione di una failure. Il testing potrà essere momentaneamente sospeso nel caso venga restituito, al momento di esecuzione, un errore nella definizione di uno dei test stessi.

**Criteri di ripristino**

Il testing verrà ripreso in caso di risoluzione di fault precedentemente individuati.

# 8 Materiale di Testing

L’hardware necessario per l’attività di test è un computer per cui non è necessario il collegamento a internet, in quanto il sistema non è stato ancora rilasciato al pubblico.

# 9 Test Cases

TC\_1.0 Registrazione account

|  |  |
| --- | --- |
| **Parametro: Username**  **Formato: {1,20}** | |
| **Categorie** | **Scelte** |
| Lunghezza lu | * lunghezza < 1 [errore] * lunghezza >= 1 && lunghezza <= 20 [property lunghezzaLUok] * lunghezza > 20 [errore] |

|  |  |
| --- | --- |
| **Parametro: email**  ***Formato:* ^[\\w!#$%&'\*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'\*+/=?`{|}~^-]+){1,64}@(?:[a-zA-Z0-9-]+\\.){1,200}[a-zA-Z]{2,6}$** | |
| **Categorie** | **Scelte** |
| Formato fe | * Rispetta il formato [property formatoFEok] * Non rispetta il formato[errore] |

|  |  |
| --- | --- |
| **Parametro: Password**  **Formato: ^(?=.\*[0-9])(?=.\*[a-z]).{8,15}$** | |
| **Categorie** | **Scelte** |
| Formato fp | * Rispetta il formato [property formatoFPok] * Non rispetta il formato [errore] |

|  |  |
| --- | --- |
| **Parametro: Password**  **Formato: ^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z]).{8,15}$** | |
| **Categorie** | **Scelte** |
| Match mcp | * Match con password = false [error] * Match con password = true [property MatchMCPok] |

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_1.0\_01 | lu1 | Errore |
| TC\_1.0\_02 | lu3 | Errore |
| TC\_1.0\_03 | lu2.fe2 | Errore |
| TC\_1.0\_04 | lu2.fe1.fp2 | Errore |
| TC\_1.0\_05 | lu2.le1.fp1.mcp1 | Errore |
| TC\_1.0\_06 | lu2.le1.fp1.mcp2 | Inserimento |

TC\_2.0 Inserimento Commento

|  |  |
| --- | --- |
| **Parametro: Commento**  **Formato: {3,100}** | |
| **Categorie** | **Scelte** |
| Lunghezza lc | 1. lunghezza < 3 [errore] 2. lunghezza >= 3 && lunghezza <= 100 [] 3. lunghezza > 100 [errore] |

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_2.0\_01 | lc.1 | Errore |
| TC\_2.0\_02 | lc.3 | Errore |
| TC\_2.0\_03 | lc.2 | Inserimento |

TC\_3.0 Pubblicazione Storia

|  |  |
| --- | --- |
| **Parametro: Storia**  ***Formato: {1,500}*** | |
| **Categorie** | **Scelte** |
| Lunghezza ls | * lunghezza < 1 [errore] * lunghezza >= 1 && lunghezza <= 500 [property lunghezzaLSok] * lunghezza > 500 [errore] |

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_3.0\_01 | ls.1 | Errore |
| TC\_3.0\_02 | ls.3 | Errore |
| TC\_3.0\_03 | ls.2 | Inserimento |

TC\_4.0 Login

|  |  |
| --- | --- |
| **Parametro: Email**  ***Formato:* ^[\\w!#$%&'\*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'\*+/=?`{|}~^-]+){1,64}@(?:[a-zA-Z0-9-]+\\.){1,200}[a-zA-Z]{2,6}$** | |
| **Categorie** | **Scelte** |
| Pre-esistente prem | 1. Pre-esistente[property pre-esitentePREMok] 2. Non pre-esistente [errore] |

|  |  |
| --- | --- |
| **Parametro: Password**  ***Formato: ^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z]).{8,15}$*** | |
| **Categorie** | **Scelte** |
| 1. Match mp | 1. Password corretta [if pre-esistentePREMok] [property matchMPok] 2. Password non corretta [if pre-esistentePREMok] [errore] |

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_4.0\_01 | prem2 | Errore |
| TC\_4.0\_02 | prem1.mp2 | Errore |
| TC\_4.0\_03 | prem1.mp1 | Login |

# 10 Testing schedule

Le attività di pianificazione del testing avverranno come definito nei capitoli precedenti, cioè subito dopo la fase di design necessaria per la pianificazione.

La scrittura dei casi di test avverrà in contemporanea con lo sviluppo del codice.

L’esecuzione dei test avverrà sia durante che dopo l’implementazione del sistema. Una volta concluso lo sviluppo, tutti i test saranno rieseguiti per garantirne il corretto funzionamento e produrre i report finali.

Per altre informazioni si rimanda ai documenti di management sulle schedule.