TEST PLAN V-1.0

1. **Introduzione**

Lo scopo di questo documento è di pianificare l’attività del testing del sistema SineCharta al fine di verificare se esistono differenze tra il comportamento atteso e il comportamento osservato. In questa attività andremo a rilevare gli eventuali errori prodotti all’interno del codice, per evitare che essi si presentino nel momento in cui il sistema verrà utilizzato dall’utente finale.

Le attività del test sono state pianificate per le seguenti gestioni:

1. Gestione Autenticazione;
2. Gestione Registrazione;
3. Gestione Storia;
4. Gestione Sessione;

Si noti, tuttavia, che verranno testate esclusivamente le funzionalità implementate e specificate nell’ODD, oltre alla gestione dei test delle funzionalità, vengono anche pianificate le responsabilità del team e lo scheduling del test. La fase di testing è strettamente legata alle fasi ad essa precedenti; ogni document, risultato delle differenti fasi di sviluppo, sarà un punto di partenza indispensabile per poter effettuare un testing corretto e adeguato.

1. **Documenti Correlati**

Il test plan ha ovviamente una stratta relazione con il resto dei documenti che sono stati prodotti finora, poiché prima di passare alla fase di testing, oltre ad aver implementato il sistema nella gran parte, esso era stato pianificato nelle precedenti documentazioni. Questo permette di rilevare le eventuali differenze tra ciò che si desiderava e ciò che invece fa il sistema.  
Di seguito verranno riportate le relazioni tra il test plan e la documentazione precedente.

* 1. **Relazione con il documento di analisi dei requisiti (RAD)**

La relazione tra test plan e RAD (Requirement Analysis Document) riguarda in particolare i requisiti funzionali e non funzionali del sistema poiché i test che verranno eseguiti su quelle funzionalità tenendo conto delle specifiche espresse nel documento sopracitato.  
In particolare il RAD contiene lo scopo del sistema, l’ambito del sistema e gli obiettivi, mostrando una panoramica di requisiti funzionali, requisiti non funzionali, scenari, casi d’uso, diagrammi e mockup del sistema.

2.2 **Relazione con il System Design Document (SDD)**

L’SDD (System Design Document) rappresenta l’architettura del sistema suddiviso in tre layer: Presentation Layer, Application Logic Layer e Storage Layer. Il test deve tener conto di queste suddivisioni. In particolare, l’SDD contiene l’architettura del software proposto, la divisione in sottosistemi e i loro servizi.

2.3 **Relazione con l’Object Design Document (ODD)**

Il test si baserà sulle class Interfaces nell’ODD (Object Design Document). In particolare, l’ODD contiene i package del sistema e i class interface del sistema.

1. **Panoramica del Sistema**

Come abbiamo analizzato e deciso nel System Design Document, la struttura del nostro Sistema è divisa secondo un’architettura “Three Tiers” cioè a tre livelli: Presentation Layer, Application Logic Layer e Storage Layer. Il livello più alto interagisce con il livello applicativo che a sua volta si occuperà di eseguire le operazioni nel database di SineCharta, cercando di garantire il più possibile basso accoppiamento e alta coesione tra le varie classi. Il sistema inoltre è suddiviso in sottosistemi più piccoli, in particolare è diviso per gestioni. Abbiamo individuato i seguenti sottoinsiemi:

* Gestione Utenti
* Gestione Storia
* Gestione Sessione

Tutte le precedenti gestioni prevedono principalmente operazioni di inserimento, modifica, visualizzazione e ricerca che saranno testate nel corso della fase di testing del Sistema.

1. **Funzionalità del sistema da testare (da decidere)**

Di seguito saranno elencate le funzionalità del sistema, per ogni gestione, che saranno testate.

1. Gestione Utenti

* LogIn
* LogOut
* Registrazione

1. Gestione Storia

* Editor Storia
* Invitare Giocatori alla Storia
* Accettare un invito
* Creare un Pg

1. Gestione Sessione

* Editor Sessione
* Gestione Mazzo
* Gestione NPC
* Gestione PG

1. **Criteri di Pass/Failed**

I dati di input del test saranno raggruppati in insiemi dalle caratteristiche comuni in modo da effettuare un test su di un unico elemento rappresentativo.  
Il testing ha successo se l’output osservato è diverso dall’output atteso: ciò significa che la fase di testing avrà successo se individuerà una failure. In tal caso verrà analizzata e, se legata ad un fault, si procederà alla sua correzione. Sarà infine iterata la fase di testing per verificare che la modifica non abbia impattato su altri componenti del sistema.  
La failure è quindi uno stato di condizione nel quale non si trova l’output desiderato, di può dire che è il contrario di un successo.

1. **Approccio**

L’approccio alla fase di testing si compone in tre fasi: la prima servirà a testare le componenti una ad una, la seconda servirà a testare le funzionalità delle integrazioni dei vari sottosistemi, infine si testerà l’intero sistema assemblato per verificare soprattutto che esso soddisfi le richieste del cliente.

**6.1 Testing di unità**

Durante questa fase, verranno ricercate le condizioni di fallimento isolando i componenti ed usando   
 test driver e stub, cioè implementazioni parziali di componenti che dipendono o da cui dipendono le componenti da testare. La strategia utilizzata per il testing si baserà esclusivamente sulla tecnica Black-Box, che si focalizza sul comportamento Input/Output, ignorando la struttura interna del componente. Al fine di minimizzare il numero di test cases, i possibili input verranno partizionati in classi di equivalenza e per ogni classe verrà selezionato un test case. Gli stati erronei scoperti in questa, come in una qualsiasi altra fase di testing, che comporteranno un fallimento del sistema dovranno essere tempestivamente comunicati agli sviluppatori al fine di correggerli e ripristinare il testing al più presto possibile, per comunicare i vari fallimenti identificati durante la fase di testing si utilizzerà un test incident report.

**6.2 Testing di integrazione**

In questa fase si procederà all’integrazione delle componenti di una funzionalità che verranno testate nel complesso attraverso una strategia Bottom-Up. Si passerà, poi, alla funzionalità successiva fino ad esaurire le funzionalità implementate. Questo approccio mira principalmente a a ridurre le dipendenze tra funzionalità differenti e a facilitare la ricerca di errori nelle interfacce di comunicazione tra sottosistemi.

**6.3 Testing di sistema**

Lo scopo di questa fase di testing è quello di dimostrare che il sistema soddisfi effettivamente i requisiti richiesti e sia quindi pronto all’uso. Come per il testing di unità, si cercherà di testare le funzionalità più importanti per l’utente e quelle che hanno una maggiore probabilità di fallimento. Si noti che, come per il testing di unità, si procederà attraverso la tecnica del Blcack-Box.

1. **Sospensione e ripresa**

**7.1 Criteri di sospensione**

La fase di testing del sistema verrà sospesa quando si otterranno i risultati attesi in accordo con i tempi di sviluppo previsti, tenendo sempre conto dei costi dei costi dell’attività del testing. Questo processo verrà quindi portato avanti quanto più avanti possibile nel tempo senza però rischiare di ritardare la consegna finale del progetto.

**7.2 Criteri di ripresa**

La fase di testing potrà riprendere in seguito a modifiche o correzioni che generano errori o fallimenti, i test case verranno quindi sottoposti nuovamente al sistema assicurandosi di aver risolto effettivamente il problema.

1. **Materiale per il testing**

Gli strumenti necessari per l’attività di testing sono un computer, su cui è installato un browser, con una connessione ad Internet poiché il database del sistema è stato caricato su un dominio online.

1. **Test Cases**