** Università degli studi di Salerno**

**Corso di Ingegneria del Software**

****

*Partecipanti:*

|  |  |
| --- | --- |
| **Nome Cognome** | **Matricola** |
| **David Grieco** | **0512105112** |
| **Ciro Perfetto** | **0512105520** |
| **Silvio Esposito** | **0512105100** |
| **Fabrizio Trapani** | **0512105298** |

**Test Plan**

**1 Introduzione**

Lo scopo di questo documento è quello di gestire lo sviluppo e le attività di test riguardanti il sito web GamEvaluate. Saranno identificati: gli elementi da testare, le funzionalità da testare, le strategie di testing usate e gli strumenti utilizzati, il personale responsabile dei test, le risorse e le attività richieste per completare i test ei rischi associati al piano. Lo scopo del testing è quello di rilevare errori in maniera pianificata all'interno del codice realizzato. Quindi, l’obiettivo consiste nell’evitare che gli errori si presentino durante l’utilizzo del sistema dell'utente finale. I risultati prodotti dai test saranno utilizzati per comprendere dove intervenire per correggere gli errori o apportare modifiche per il migliorare il sistema. In questo documento verranno analizzate, in particolar modo, le seguenti attività:

- Gestione Account;

- Gestione Generi;

- Gestione Giochi;

- Gestione Piattaforme;

- Gestione Recensioni;

- Gestione Ricerca;

- Gestione Utenti;

Per ogni gestione saranno testate solo le funzionalità contenente i requisiti con priorità **Medio/Alta**.

* **1.1 Relazioni con altri documenti**

Il Test Plan ha una stretta relazione con i documenti prodotti finora, dato che il sistema è stato pianificato nelle precedenti documentazioni. Per verificare il corretto funzionamento del sistema Libra saranno usati i test cases individuati e documentati precedentemente nel processo di sviluppo del sistema. I test cases sono basati sulle funzionalità individuate nel documento di raccolta ed analisi dei requisiti (RAD).

* + **1.1.1 Relazioni 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 dato che i test saranno eseguiti su quelle funzionalità tenendo conto delle specifiche espresse nel documento precedente. In particolare il RAD contiene lo scopo del sistema, l’ambito del sistema e gli obiettivi, evidenziando una panoramica di requisiti funzionali, requisiti non funzionali, scenari, casi d’uso, diagrammi e mockup del sistema.

I documenti a cui si fa riferimento sono:

- Requirement Analysis Document\_GamEvaluate.

* + **1.1.2 Relazioni con il System Design Document (SDD)**

L’SDD (System Design Document) rappresenta l’architettura del sistema MVC.In particolare l’SDD contiene l’architettura del software corrente e proposto e i servizi dei sottosistemi.

I documenti a cui si fa riferimento sono:

- System Design Document\_GamEvaluate.

* + **1.1.3 Relazioni con l’Object Design Document (ODD)**

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

I documenti a cui si fa riferimento sono:

- Object Design Document\_GamEvaluate.

**2 Panoramica del sistema**

Come analizzato e deciso nel System Design Document la struttura del nostro sistema segue il pattern architetturale MVC (Model – View - Control). Il componente centrale del MVC, il model, cattura il comportamento dell'applicazione in termini di dominio del problema, indipendentemente dall'interfaccia utente. Il model gestisce direttamente i dati, la logica e le regole dell'applicazione. La view può essere una qualsiasi rappresentazione in output di informazioni, come un grafico o un diagramma. Sono possibili view multiple delle stesse informazioni, come ad esempio un grafico a barre per la gestione e la vista tabellare per l'amministrazione. La terza parte, il controller, accetta l'input e lo converte in comandi per il modello e/o la vista. Il sistema inoltre è stato suddiviso in sottosistemi più piccoli, in particolare è stato diviso per gestioni. Abbiamo individuato i seguenti sottosistemi:

-Game Manager;

-Review Manager;

-Platform Manager;

-Genre Manager;

-Account Manager;

**3 Funzionalità da testare e da non testare**

Abbiamo deciso di non testare i metodi *get e set*, e i metodi con priorità bassa.

Di seguito saranno riportati le funzionalità introdotte nel sistema che saranno sottoposte a test e suddivise in sottosistemi:

- Gestione Account:

* + Log-in;
  + Sign-in;

- Gestione Utenti:

* Ban/Unban di un utente;
* Upgrade/Downgrade di un utente;

- Gestione Gioco:

* Aggiungere un gioco;
* Eliminare un gioco;
* Votare un gioco;

-Gestione Amministratore:

* Aggiungere un genere;
* Aggiungere una piattaforma;
* Eliminare una piattaforma;

- Gestione Recensioni:

* Aggiungere una recensione;
* Rimuovere una recensione scritta da un altro utente;
* Rimuovere una propria recensione;

- Gestione Ricerca:

* Ricerca utente;
* Ricerca videogioco;
* Visualizzare le informazioni di un videogioco;
* Visualizzare le informazioni di un utente;

**4 Criteri di successo/insuccesso**

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 questa 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 quindi è uno stato di condizione nel quale non si trova l’output desiderato, si può dire che è il contrario di un successo.

**5 Approccio**

L’approccio alla fase di testing si compone di 3 fasi, la prima servirà a testare le componenti una ad una, poi si passerà 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.

* **5.1 Testing di unità**

Per realizzare il testing di ogni singola componente verrà utilizzata la tecnica “Black-Box testing”. Così facendo andremo ad esaminare le funzionalità dell’applicazione ed il comportamento input/output delle singole componenti senza tener conto della loro struttura interna. Essendo quasi impossibile generare tutti i possibili input, verranno create classi d’equivalenza scegliendo per ognuna un test case per ridurre la ridondanza e rendere il test più efficiente. I risultati del testing verranno analizzati e usati per correggere gli errori che causano il fallimento del sistema.

* **5.2 Testing di integrazione**

Dopo aver sottoposto ogni componente al testing di unità, ed aver corretto gli eventuali errori scaturiti dal test, essi verranno integrati in sottosistemi più grandi per sottoporli ad un test di integrazione. Il testing verrà effettuato seguendo la strategia “Bottom-Up” che prevede l’integrazione dal basso verso l’alto, realizzando dei test stub.

* **5.3 Testing di sistema**

Prima di essere pronto all’uso, il sistema affronterà l’ultima fase di testing, quello di sistema, per dimostrare che siano soddisfatti tutti i requisiti richiesti. Lo scopo di questa fase è testare le funzionalità più importanti, usate maggiormente e con maggior probabilità di fallimento. Trattandosi di un sistema web-based verrà utilizzato, per effettuare il testing di sistema, il tool Selenium. Selenium si occupa di simulare l’interazione con il sistema dal punto di vista dell’utente.

**6 Sospensione e ripresa**

* **6.1 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 dell’attività di testing.

Questo processo verrà quindi portato avanti quanto più possibile nel tempo senza però rischiare di ritardare la consegna finale del progetto.

* **6.2 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 così di aver risolto effettivamente il problema.

**7 Materiale per il testing**

Gli strumenti necessari per l’attività di test sono una macchina, il software MySQL per il database, e un’ide per java EE.

**8 Test Cases**