**Warrior’s Adventure - Testing Plan Document**

**1. Introduction**

Lo scopo del documento è quello di specificare gli aspetti manageriali del testing. La seguente fase andrà ad effettuare dei singoli test in modo tale da poter verificare il corretto funzionamento dei vari sottosistemi del software. In caso di rilevazioni di eventuali errori la fase che segue, sé finalizzata alla riparazione dei bug riscontrati. In questo modo si cerca di soddisfare i requisiti funzionali e non funzionali del prodotto. Il seguente documento specifica i requisiti e le componenti che devono essere testati.

**2. Relationship to other documents**

* **2.1 Relationship with RAD:** Il documento di analisi dei requisiti verrà utilizzato per specificare quali requisiti funzionali andranno testati, si farà riferimento al numero del requisito funzionale.
* **2.2 Relationship with SDD:** Il documento di design del sistema verrà utilizzato per specificare come il sistema è stato suddiviso, di conseguenza verranno menzionati i vari sottosistemi.

**3.System Overview**

Come specificato nel documento di design (nel paragrafo 3 System Architecture) il sistema è stato suddiviso in 3 sottosistemi: Presentation layer, Business layer e Persistance layer. Ogni sottosistema si occuperà di una logica diversa. Il layer di presentazione ha lo scopo di costruire l’interfaccia utente. Il layer di Business si occuperà di implementare tutti i requisiti funzionali specificati nel RAD al paragrafo 4. Infine, il layer della Persistenza si occuperà di salvaguardare i dati dei giocatori e di mantenerli in memoria al sicuro.

**4. Features to be tested/not to be tested**

Il layer di business a sua volta si divide in diversi sottosistemi, ognuno che implementa una propria logica che fa riferimento ai requisiti funzionali. Il seguente paragrafo divide i sottosistemi e specifica quali requisiti funzionali verranno testati.

|  |  |  |
| --- | --- | --- |
| **Presentation Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF3** | Si | Iscrizione di un nuovo giocatore. |
| **RF10** | Si | Autenticazione e accesso al gioco. |

|  |  |  |
| --- | --- | --- |
| **Combat Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF1** | No | Requisiti di livello per scontro PvP. |
| **RF7** | No | Scontro PvE. |

|  |  |  |
| --- | --- | --- |
| **Guild Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF2** | No | Gestione e creazione di una gilda. |

|  |  |  |
| --- | --- | --- |
| **Market Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF8** | Si | Gestione del mercato per acquisto armi e armature. |

|  |  |  |
| --- | --- | --- |
| **Administration Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF12** | Si | Comandi dedicati all’amministratore |

|  |  |  |
| --- | --- | --- |
| **Moderation Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF11** | No | Comandi dedicati al moderatore per gestire la qualità della vita |

|  |  |  |
| --- | --- | --- |
| **Character Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF4** | Si | Creazione di un personaggio [Limite di tre] |
| **RF5** | No | Eliminazione personaggi |
| **RF15** | Si | Salvataggio dei progressi |

|  |  |  |
| --- | --- | --- |
| **Tournament Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF6** | No | Gestione e creazione tornei |
| **RF9** | No | Ogni giocatore può partecipare ad un torneo |

|  |  |  |
| --- | --- | --- |
| **UI Logic** |  | |
| **Requisito funzionale** | **Testing** | **Descrizione** |
| **RF13** | Si | Gestione delle notifiche |
| **RF14** | Si | Visualizzazione statistiche personaggio |

**5. Pass/Fail criteria**

Gli input verranno divisi in base alle caratteristiche comuni in modo da andare a testare solo un elemento della componente. Se la componente da testare restituisce i risultati attesi il test su quella componente è superato.

**6. Approach**

L’approccio scelto per la fase di testing si è basata sulla suddivisione del testing in due fasi: testing di unità, e testing di sistema. In questo modo è possibile controllare in maniera efficiente ogni sottosistema implementata e di trovare e corregge eventuali bug prima dell’inserimento all’interno del sistema.

* **6.1 Testing di unità:** In questa fase il testing viene effettuato in maniera dinamica andando ad utilizzare la tecnica di Black-Box. Se la fase di testing è superata si passa alla fase di integrazione.
* **6.2 Testing di sistema:** In questa fase l’intero sistema è pronto per essere testato. La strategia scelta è di tipo Testing funzionale utilizzando l’approccio Black-Box.

**7. Suspension and Resumption**

* **7.1 Suspension:** Il testing verrà sospeso nel caso in cui alcuni sottosistemi dipendenti non sono ancora stati implementati oppure sono stati trovati dei bug da correggere. In caso non siano stati rilevati degli errori è possibile passare alla fase di testing successiva.
* **7.2 Resumption:** In caso di interruzione del testing, in riferimento al paragrafo precedente, una volta che il bug è stato risolto il testing riprende andando a testare l’unità appena modificata, se il nuovo test viene superato è possibile continuare con la fase di testing di altre unità, in caso contrario il testing si ferma e si cerca di risolvere il bug.

**8. Testing materials**

Come supporto alla fase di testing si utilizzerà un browser Internet, la scelta è ricaduta su Google Chrome. Come strumenti di supporto verrà utilizzato il framwork Selenium aggiungendo a Google Chrome Selenium IDE.

**9. Testing materials**

I test case fanno riferimento al documento: “Testing Case Specification”.

**10. Testing Schedule**

La fase di testing sarà strutturata in modo che ogni membro testerà e compilerà un report di una componente da lui non implementata. Il report verrà consegnato al membro del team che ha implementato quella specifica componente per poter correggere eventuali bug riscontrati.