**.**

**UNIVERSITÀ DEGLI STUDI DI SALERNO**

**ANNO ACCADEMICO 2019/2020**

**REQUIREMENTS ANALYSIS DOCUMENT**

**Version 2.0**

**TOP MANAGER**

**Prof. Andrea De Lucia**

**PROJECT MANAGER**

**Giuseppe De Michele**

**Partecipanti**

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Mario Balbi | 0512102944 |
| Giuseppe De Michele | 0512102642 |
| Singh Karanbir | 0512104924 |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autore** |
| 12/11/2019 | 1.0 | Introduzione:  Scopo del sistema;  Ambito del sistema;  Obiettivi e criteri di successo del progetto; Panoramica; | De Michele Giuseppe |
| 14/11/2019 | 1.1 | Sistema corrente  Sistema proposto:  Panoramica;  Identificazione attori; | De Michele Giuseppe |
| 19/11/2019 | 1.2 | Requisiti non funzionali | De Michele Giuseppe  Balbi Mario |
| 19/11/2019 | 1.3 | Requisiti funzionali | Singh Karanbir |
| 19/11/2019 | 1.4 | Scenari GA | Balbi Mario |
| 26/11/2019 | 1.5 | Scenari e revisione | Singh Karanbir |
| 10/12/2019 | 1.6 | Tabelle casi d’uso | De Michele Giuseppe |

**Indice**

1. Introduzione
   1. Descrizione del Problema
   2. Design Goals
      1. DG\_0 Criteri di Performance
      2. DG\_1 Criteri di Affidabilità
      3. DG\_2 Criteri di Costo
      4. DG\_2 Criteri di Mantenimento
      5. DG\_2 Criteri di Utente
   3. Definizioni, Acronimi e Abbreviazioni
   4. Riferimenti
   5. Panoramica

**1. Introduzione**

L’obiettivo del seguente documento consiste nel pianificare l’attività di testing del sistema NashiraCustomGuitars in modo da osservare se vi sono differenze fra il comportamento osservato e il comportamento aspettato. Lo scopo di questa fase è quello di prevenire la presenza di errori nel sistema durante l’utilizzo dell’utente finale.

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

* Gestione Account
* Gestione Chitarra
* Gestione Personalizzazione
* Gestione Carrello
* Gestione Vendite
* Gestione Assistenza
* Gestione Esperti

Va comunque specificato, che verranno testate solamente le funzionalità implementante e

descritte nell’ODD. Per il testing, verranno utilizzati i test case, ovvero un insieme di input e di

risultati attesi che servono appunto a testare una componente per scoprirne gli errori e i fallimenti.

**2. Documenti correlati**

Il test plan è strettamente correlato alla documentazione precedentemente prodotta, in quanto buona parte del sistema era stata precedentemente progettata e anche implementata, e in questa fase risulta quindi più che mai necessario utilizzare il risultato delle fasi precedenti e dell’implementazione effettuata, in modo da rilevare correttamente le possibili differenze fra ciò che si era progettato e ciò che il sistema realizza davvero.

**2.1 Relazioni con il documento di analisi dei requisiti (RAD)**

La relazione del test plan con il documento di analisi dei requisiti (RAD) è molto stretta in quanto in quest’ultimo vengono descritti requisiti funzionali e non funzionali, e si ottengono le use case sulle quali dovranno poi basarsi i test case.

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

La relazione del test plan con il documento di analisi dei requisiti (RAD) è molto stretta in quanto in quest’ultimo vengono descritti requisiti funzionali e non funzionali, e si ottengono le use case sulle quali dovranno poi basarsi i test case.

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

Il test d’integrazione farà riferimento alle classi delineate nell’ODD in modo da attenersi il più possibile a quest’ultime.

**3. Panoramica del sistema**

Come descritto nel System Design Document (SDD) il nostro sistema è costruito da una architettura “Three Layers” ovvero a tre livelli, al fine di garantire basso accoppiamento e alta coesione fra le classi. I tre livelli della architettura sono **Presentation, Application e Storage Layer**. I layer contengono rispettivamente la parte visuale della applicazione, la parte applicativa e il database. Il sistema NashiraCustomGuitars è stato diviso ulteriormente un sottosistema più piccoli, in particolare è stato diviso per gestioni, le gestioni sono state definite nel capitolo 1. Le gestioni che abbiamo individuato prevedono principalmente metodi che effettuano operazioni CRUD(Create, Read, Update, Delete) e saranno queste le funzionalità che andranno testate.

**4. Funzionalità da testare/non testare**

Di seguito vengono elencate le funzionalità da testare per ogni gestione:

* Gestione Account
  + Registrazione
  + Login
  + Modifica profilo
* Gestione Chitarra
  + Aggiungi chitarra
  + Rimuovi chitarra
* Gestione Carrello
  + Aggiungi prodotto nel carrello
  + Modifica prodotto nel carrello
  + Rimuovi prodotto nel carrello
* Gestione Vendita
  + Acquista chitarra

Di seguito vengono elencate le funzionalità da non testare per ogni gestione:

* Gestione Account
  + Visualizza profilo
  + Logout
  + Visualizza utenti
* Gestione Chitarra
  + Visualizza chitarra
* Gestione Carrello
  + Visualizza prodotto nel carrello
* Gestione Vendite
  + Visualizza ordine
* Gestione Assistenza
  + Effettua Richiesta
  + Visualizza richiesta
  + Rispondi richiesta
* Gestione Esperto
  + Visualizza esperto
  + Aggiungi esperto
  + Rimuovi esperto

Le funzionalità che non saranno testate sono divise in più categorie:

-funzionalità appartenenti a componenti prettamente grafiche che non offrono funzionalità di spicco per il corretto funzionamento del sistema.

-funzionalità appartenenti a componenti molto semplici, il cui testing è implicito nella loro stessa struttura.

Inoltre, il caricamento delle foto non verrà testato tramite IDE per problemi di convalida della correttezza di un file.

**5. Criteri Pass/Field**

I dati di input del test saranno suddivisi in classi di equivalenza, ovvero saranno raggruppati in insiemi con caratteristiche comuni, in modo tale da poter testare solo uno degli elementi della classe. L’input avrà superato il test se l’output risultante è quello atteso, il risultato atteso sarà quello specificato dal membro del team che si occuperà del testing su tale test case.

**6. Approccio**

La fase di testing sarà suddivisa in tre fasi:

**1. Testing di unità:** Verrà testata nello specifico il funzionamento di ogni singola unità del sistema;

**2. Testing di integrazione:** Dove verranno testate le interfacce delle suddette unità;

**3. Testing di sistema:** Dove verranno testato l’intero sistema assemblato.

**6.1 Testing di Unità**

Durante questa fase verranno ricercate le condizioni di fallimento, isolando i componenti. La strategia usata per il testing è la tecnica Black-Box, che si concentra sul comportamento Input/Output ignorando la struttura interna della componente. Per minimizzare il numero di test cases i possibili input verranno partizionati in classi di equivalenza e per ogni classe verrà usato un test case. Gli errori scovati in questa fase, e anche in tutte le altre, devono essere comunicati agli sviluppatori in modo tale che questi possano correggere l’errore e ripristinare la fase di testing al più presto. I fallimenti identificati durante la fase di testing verranno specificati nel test incidenti report.

**6.2 Testing di integrazione**

Questa fase ha lo scopo di integrare tutte le componenti di una funzionalità al fine di testarle nel complesso utilizzando una strategia Big Bang. L’approccio Big Bang mira a testare prima le componenti individualmente e poi tutte insieme, come un unico sistema.

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

SCEGLIERE UNO DEI 2 METODI

**6.3 Testing di sistema**

In questa ultima fase vogliamo dimostrare che il sistema soddisfi tutti i requisiti richiesti. Si cercherà di testare le funzionalità più utili per l’utente e quelle che hanno una maggiore probabilità di fallimento.

**7. Sospensione e ripresa**

### 

### **7.1 Criteri di sospensione**

La fase di testing del sistema verrà sospesa quando si raggiungerà un compromesso tra qualità del prodotto e costi dell’attività di testing. Il testing verrà quindi portato avanti quanto più possibile nel tempo senza però rischiare di ritardare la consegna finale del progetto.

### **7.2 Criteri di ripresa**

In seguito ad ogni modifica o correzione delle componenti che genereranno errori o fallimenti, i test case verranno sottoposti nuovamente al sistema assicurandosi così di aver risolto effettivamente il problema.

**8. Materiale per il Testing**

Le risorse software necessarie per il testing sono le seguenti:

-NetBeans: come ambiente di sviluppo ed esecuzione di applicazioni JAVA.

-Microsoft Word: come applicazione per la scrittura di test case e plan.

**9.Test Cases**

* 1. **Gestione Account**
     1. Registrazione

**Category partition**

|  |  |
| --- | --- |
| **Parametro:** Nome  **Formato:** ^[a-zA-Z]{1,20}$ | |
| Formato: **[FNome]** | 1. Rispecchia il formato  2. Lunghezza < 1 oppure > 20 [error]  3. Contiene caratteri speciali [error] |

|  |  |
| --- | --- |
| **Parametro:** Cognome  **Formato:** ^[a-zA-Z]{1,20}$ | |
| Formato: **[FCognome]** | 1. Rispecchia il formato  2. Lunghezza < 1 oppure > 20 [error]  3. Contiene caratteri speciali [error] |

|  |  |
| --- | --- |
| **Parametro:** Indirizzo  **Formato:** ^[a-zA-Z]{1,20}$ | |
| Formato: **[FIndirizzo]** | 1. Rispecchia il formato  2. Lunghezza < 1 oppure > 100 [error]  3. Contiene caratteri speciali [error] |

|  |  |
| --- | --- |
| **Parametro:** Città  **Formato:** ^[a-zA-Z]$ | |
| Formato: **[FCittà]** | 1. Rispecchia il formato  2. Lunghezza < 1 oppure > 50 [error]  3. Contiene caratteri speciali [error] |

|  |  |
| --- | --- |
| **Parametro:** ZIP  **Formato:** ^{1,9}$ | |
| Formato: **[FZip]** | 1. Rispecchia il formato  2. Lunghezza < 1 oppure > 6 [error]  3. Contiene caratteri speciali [error] |

|  |  |
| --- | --- |
| **Parametro:** email  **Formato:** ^[a-zA-Z0-9.!#$%&'\*+/=?^\_`{|}~-]+ @ [A-z0-9\\.\_-]+\\.[A-z]{2,6}$ | |
| Formato: **[FEmail]** | 1. Rispecchia il formato  2. Contiene caratteri speciali [error]  3. L’email inserita è già presente nel sistema[error]  4. Non contiene il caratte “@”[error] |

|  |  |
| --- | --- |
| **Parametro:** password  **Formato:** ^[a-zA-Z0-9\\.-\_]{1,10}$ | |
| Formato: **[FPassword]** | 1. Rispecchia il formato  2. Lunghezza <1 oppure >10 [error] |

|  |  |
| --- | --- |
| **Parametro:** Confermapassword  **Formato:** ^[a-zA-Z0-9\\.-\_]{1,10}$ | |
| Formato: **[FCPassword]** | 1. Rispecchia il formato  2. Lunghezza <1 oppure >10 [error]  3. Non è uguale alla password [error] |

**Test Cases**

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_1.1\_1 | FNome2 | Errato |
| TC\_1.1\_2 | FNome3 | Errato |
| TC\_1.1\_3 | FNome1, FCognome2 | Errato |
| TC\_1.1\_4 | FNome1, FCognome3 | Errato |
| TC\_1.1\_5 | FNome1, FCognome1, FIndirizzo2 | Errato |
| TC\_1.1\_6 | FNome1, FCognome1, FIndirizzo3 | Errato |
| TC\_1.1\_7 | FNome1, FCognome1, FIndirizzo1, FCitta2 | Errato |
| TC\_1.1\_8 | FNome1, FCognome1, FIndirizzo1, FCitta3 | Errato |
| TC\_1.1\_9 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip2 | Errato |
| TC\_1.1\_10 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip3 | Errato |
| TC\_1.1\_11 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip1, FEmail2 | Errato |
| TC\_1.1\_12 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip1, FEmail3 | Errato |
| TC\_1.1\_13 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip1, FEmail4 | Errato |
| TC\_1.1\_14 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip1, FEmail1, Fpassword2 | Errato |
| TC\_1.1\_15 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip1, FEmail1, Fpassword1, FCPassword2 | Errato |
| TC\_1.1\_16 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip1, FEmail1, Fpassword1, FCPassword3 | Errato |
| TC\_1.1\_17 | FNome1, FCognome1, FIndirizzo1, FCitta1, FZip1, FEmail1, Fpassword1, FCPassword1 | Registrazione effettuata |

9.2.1 Login

Category partition

|  |  |
| --- | --- |
| **Parametro:** Email  **Formato:** ^[a-zA-Z0-9.!#$%&'\*+/=?^\_`{|}~-]+@[A-z0-9\\.\_-]+\\.[A-z]{2,6}$ | |
| Formato: **[FEmail]** | 1. Rispecchia il formato  2. Contiene caratteri speciali [error] |
| Riscontro nel  Database **[RDBE]** | 1.L’email è presente nel database  2.L’email non è presente nel database [error] |

|  |  |
| --- | --- |
| **Parametro:** Password  **Formato:** ^[a-zA-Z0-9\\.-\_]{1,10}$ | |
| Formato: **[FPassword]** | 1. Rispecchia il formato  2. Lunghezza <1 oppure >10 [error] |
| Riscontro nel  Database **[RDBP]** | 1. La password è associata all’email  2. La password non è associata all’email [error] |

**Test Cases**

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_2.1\_1 | FEmail2 | Errato |
| TC\_2.1\_2 | FEmail1, RDBE2 | Errato |
| TC\_2.1\_3 | FEmail1, RDBE1, FPassword2 | Errato |
| TC\_2.1\_4 | FEmail1, RDBE1, FPassword1, RDBP2 | Errato |
| TC\_2.1\_5 | FEmail1, RDBE1, FPassword1, RDBP1 | Login riuscito |