**Test Plan Document**

1. **Introduzione**

Lo scopo di questo documento è definire quelli che sono gli aspetti gestionali del test tenendo traccia di quelle che sono le risorse e il programma dell’attività di test. L’obiettivo sarà quindi quello di cercare le differenze tra il comportamento previsto specificato all’interno dei modelli di sistema e il comportamento osservato dal sistema implementato. Sulla base dei risultati ottenuti verranno quindi individuati eventuali fault, che verranno corretti in un secondo momento. Verranno testate quindi le funzionalità descritte in precedenza all’interno dei seguenti documenti:

* RAD
* ODD

**2. Relazione con gli altri documenti**

Questo documento è chiaramente in stretta relazione con gli altri documenti: Nel RAD sono specificati i requisiti funzionali e non funzionali che ci aiuteranno nell’esecuzione dei test; Nell’SDD il sistema in uso è stato decomposto seguendo il modello MVC, quindi ottenendo 3 livelli, View, Control e Model. Cercheremo dunque di mantenere il testing quanto più fedele a quest’architettura. Infine, Il test d’integrazione farà riferimento alle interfacce delle classi definite nell’ODD.

**3. Panoramica del sistema**

Il nostro sistema è suddiviso in Management, ognuna dei Management ha operazione CRUD (creazione, lettura, aggiornamento e cancellazione). Verrà testato principalmente ogni Management con una particolare enfasi sulle operazioni di inserimento e aggiornamento. Verranno effettuali inizialmente testi sui singoli sottosistemi, in maniera tale da ridurre la complessità dell’attività di test, semplificare l’individuazione e la correzione dei guasti e garante lo svolgimento in parallelo delle diverse attività di test, fornendo i vari input dalle classi di equivalenza individuate non trascurando ovviamente i boundary element. Si passerà successivamente all’ Integration testing in modo da individuare eventuali errori che durante la prima fase di test descritta in precedenza non sono stati individuati mediante le opportune strategie. Il test verrà quindi svolto nella maniera descritta concentrandosi sulle operazione descritte nella prima parte del punto 3.

**4. Funzionalità da testare e da non testare**

Elenco dei Management con le funzioni da testare:

* ArticoloManagement:
  + Pubblicazione Articolo
  + Modifica articolo
  + Inserimento Commento
  + Inserimento Rating
* UserManagement:
  + Login
  + Registrazione
  + Modifica Dati Personali
* EventoManagement:
  + Inserimento evento
  + Modifica evento
* AllegatoManagement:
  + Inserimento allegato
* RicercaManagement:
  + Ricerca Articolo
  + Ricerca Autore
* NotificaManagement:
  + Invia Notifica
* AutoreMangement
  + Contatta Autore

Elenco dei Management con le funzioni che non vengono testate:

* ArticoloManagement:
  + Visualizza articolo
  + Visualizza Commento
  + Visualizza Rating
  + Moderazione articolo
  + Rifiuto articolo
* UserManagement:
  + Visualizza dati personali
  + Logout
* EventoManagement:
  + Visualizza evento
  + Cancella evento
* AllegatoManagement:
  + Scarica allegato
  + Cancella allegato
* RicercaManagement:
  + Le funzionalità fornire dal sottosistema vengono testate tutte.
* NotificaManagement:
  + Visualizza Notifica
* AutoreMangement
  + Visualizza Pagina Autore
  + Segui autore

Le seguenti funzioni non vengono testate per il seguente motivo: si tratta di funzionalità che si occupano di visualizzare semplicemente i dati memorizzati all’interno del sistema di memorizzazione e che quindi non ricevono alcun input. Di conseguenza la presenza all’interno di una di questa funzionalità è dovuto ad un eventuale fault all’interno delle funzioni fornite dai sottosistemi che devono essere testate.

**5. Criteri di Successo/Fallimento**

I dati di input verranno raggruppati in insiemi con caratteristiche comuni, per i quale testeremo solo un elemento rappresentativo e non tutti i possibili input. Un input supererà un test se l’output è tra quelli attesi, cioè quello che è stato specificato in precedenza dal membro che si occuperà del testing su quel determinato test case.

**6. Approccio**

Abbiamo deciso di dividere l’attività di testing in tre categoria: test di unità, test d’integrazione e test di sistema

**6.1 Testing d’unità**

Durante questa fase ci concentriamo sul testing degli elementi che costituiscono il sistema software utilizzando test driver e stub ove necessario per verificare la componente. La strategia utilizzata per il testing si baserà esclusivamente sulla tecnica Black-box, che si focalizza sul comportamento Input/Output, ignorando la struttura interna della componente. Utilizzando il test d’unità possiamo semplificare l’individuazione e la correzione di fault consentendoci di concentrarci su piccole parti del sistema, ridurre la complessità dell’attività di testing e infine parallelizzare l’attività di test delle componenti. Utilizzeremo le seguenti tecniche:

* Equivalence testing: in questo modo possiamo ridurre il numero di casi di test raggruppando i possibili input in classi d’equivalenza e selezionare un caso di test per ogni classe.
* Boundary testing: per cercare di ottenere un testing completo ci concentriamo anche sulle condizioni al limite delle classi di equivalenza selezionando gli elemento che sono ai “bordi” della classe d’equivalenza

Gli errori che verranno individuati durante questa fase iniziale di testing verranno ovviamente corretti nel più breve tempo possibile e verranno riportati i vari fallimenti nell’apposito documento “Test incident report”.

**6.2 Testing d’integrazione**

In questa fase passiamo all’integrazione dei componenti precedentemente testati con l’obiettivo di identificare quelli che sono fault non identificati fino a questa fase. Siccome l’ordine in cui i componenti vengono integrati può influenzare lo sforzo totale richiesto dal test d’integrazione si è scelto di focalizzare la parte principale sull’integrazione dei seguenti sottosistemi: ArticoloManagement,AllegatoManagement e NotificaManagement. Si è scelto di utilizzare una stategia di test di integrazione orizzontale in particolare una strategia Bottom-up quindi testare prima i componenti del layer inferiore (fatto già nella fase precedente di testing) e poi integrarli con quelli del layer superiore. Si è scelto di utilizzare questa strategia in modo da poter individuare eventuali errori di interfaccia

**6.3 Testing di sistema**

Mediante questo test cerchiamo di garantire che l’intero sistema sia conforme ai requisisti funzionali e non funzionali. Verranno eseguite le diverse funzionalità del sistema passando di volta in volta alla funzionalità successiva fino al termine di esse.

**7. Sospensione e riassunzione**

La fase di testing verrà sospesa verrà sospesa quando si raggiungerà un compromesso tra la 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 e rispettando i budget del progetto. Inoltre, in seguito alle modifiche o correzioni delle componenti che introdurranno errori o fallimenti, i test case verranno sottoposti nuovamente al sistema facendo così in modo di risolvere definitivamente il problema.

**8. Materiale di testing**

Il testing verrà effettuato mediante l’utilizzo di un pc ,dotato degli opportuni software, e non sarà neanche necessaria una connessione ad internet visto che il pc sarà dotato di una copia locale e fornito di un’apposita API.

**9. Casi di Testing**

I core sono i seguenti:

TC\_1.2 Registrazione/Modifica dati Personali

|  |  |
| --- | --- |
| Parametro: nome | Formato: ^[A-Z][a-z][^#!@&<>\[\]\"~;$^%{}?{0-9}]{0,30}$ |
| Formato [FN] | 1. Rispecchia il formato [FNOK] 2. Non rispecchia il formato [error] |

|  |  |
| --- | --- |
| Parametro: cognome | Formato: ^[A-Z][a-z][^#!@&<>\[\]\"~;$^%{}?{0-9}]{0,30}$ |
| Formato [FC] | 1. Rispecchia il formato [FCOK] 2. Non rispecchia il formato [error] |

|  |  |
| --- | --- |
| Parametro: email | Formato: ^[a-zA-Z0-9.!#$%&'\*+/=?^\_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)\*$ |
| Formato[FE] | 1. Rispecchia il formato [FEOK] 2. Non rispecchia il formato [error] |
| Esiste [EE] | 1. L’email non esiste nel DB [EEOK] [if FEOK] 2. L’email è già registrata nel DB [error][if FEOK] |

|  |  |
| --- | --- |
| Parametro: username | Formato: ^[A-Za-z0-9\_!#$%&'\*+/=?^\_`{|}~@-]{5,30}$ |
| Formato[FU] | 1. Rispecchia il formato [FUOK] 2. Non rispecchia il formato [error] |
| Esiste [EU] | 1. L’email non esiste nel DB [EUOK][if FUOK] 2. L’email è già registrata nel DB [error][if FUOK] |

|  |  |
| --- | --- |
| Parametro: password | Formato: ^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[#$^+=!\*()@%&]).{8,30}$ |
| Formato[FP] | 1. Rispecchia il formato [FP] 2. Non rispecchia il formato [error] |

Test case

|  |  |  |
| --- | --- | --- |
| Codice | Combinazione | Esito |
| TC\_1.2\_01 | FE2 | ERROR |
| TC\_1.2\_02 | FE1, EE2 | ERROR |
| TC\_1.2\_03 | FE1, EE1, FN2 | ERROR |
| TC\_1.2\_04 | FE1, EE1, FN1, FC2 | ERROR |
| TC\_1.2\_05 | FE1, EE1, FN1, FC1, FU2 | ERROR |
| TC\_1.2\_06 | FE1, EE1, FN1, FC1, FU1, EU2 | ERROR |
| TC\_1.2\_07 | FE1, EE1, FN1, FC1, FU1, EU1, FP2 | ERROR |
| TC\_1.2\_08 | FE1, EE1, FN1, FC1, FU1, EU1, FP1 | SUCCESS |

TC\_1.3 inserimento evento/modifica evento

|  |  |
| --- | --- |
| Parametro: data | Formato: ^([0-2][0-9]|(3)[0-1])(\/)(((0)[0-9])|((1)[0-2]))(\/)\d{4}$ |
| Formato[FD] | 1. Rispecchia il formato [FD] 2. Non rispecchia il formato [error] |
| Data Antecedente (DA) | 1. La data è precedente a quella odierna [error] [if FDOK] 2. La data è successiva a quella odierna (anche la stessa) [if DFOK][DAOK] |

|  |  |
| --- | --- |
| Parametro: nome | Formato: ^[A-Z][a-z][^#!@&<>\[\]\"~;$^%{}?{0-9}]{0,50}$ |
| Formato[FN] | 1. Rispecchia il formato [FN] 2. Non rispecchia il formato [error] |

|  |  |
| --- | --- |
| Parametro: via | Formato: [A-Z][a-zA-Z][^#&@<>\"~;$^%{}?{0-9}]{4,30}$ |
| Formato[FV] | 1. Rispecchia il formato [FL] 2. Non rispecchia il formato [error] |

|  |  |
| --- | --- |
| Parametro: città | Formato: [A-Z][a-zA-Z][^#&<>@\"~;$^%{}?{0-9}]{4,30}$ |
| Formato[FC] | 1. Rispecchia il formato [FP] 2. Non rispecchia il formato [error] |

|  |  |
| --- | --- |
| Parametro: documenti | Formato: ([a-z\_\-\s0-9\.]+)+\.(pdf|doc|docx) |
| Formato[FDOC] | 1. Rispecchia il formato [FP] 2. Non rispecchia il formato [error] |

|  |  |
| --- | --- |
| Parametro: argomento | Formato:[a-zA-Z0-9#&<>\"~;$^%{}?][^~^]{4,500}$ |
| Formato[FA] | 1. Rispecchia il formato [FA] 2. Non rispecchia il formato [error] |

Test case

|  |  |  |
| --- | --- | --- |
| Codice | Combinazione | Esito |
| TC\_1.3\_01 | FD2 | ERROR |
| TC\_1.3\_02 | FD1, DA1 | ERROR |
| TC\_1.3\_03 | FD1, DA2, FN2 | ERROR |
| TC\_1.3\_04 | FD1, DA2, FN1, FV2 | ERROR |
| TC\_1.3\_05 | FD1, DA2, FN1, FV1, FNC1, FC2 | ERROR |
| TC\_1.3\_06 | FD1, DA2, FN1, FV1, FNC1, FC1, FDOC1, FA2 | ERROR |
| TC\_1.2\_07 | FD1, DA2, FN1, FV1, FNC1, FC1, FDOC1, FA1 | SUCCESS |

TC\_1.8 richiesta pubblicazione articolo/ modifica articolo

|  |  |
| --- | --- |
| Parametro: titolo | Formato: [A-Z][a-zA-Z0-9][^#@&<>\"~;$^%{}?]{4,50}$ |
| Formato[FT] | 1. Rispecchia il formato [FTOK] 2. Non rispecchia il formato [error] |
| Esiste [ET] | 1. Il titolo è già presente nel database [if FTOK][error] 2. Il titolo non è presente nel database [if FTOK][ETOK] |

|  |  |
| --- | --- |
| Parametro: documenti | Formato: ([a-z\_\-\s0-9\.]+)+\.(pdf|doc|docx) |
| Formato[FDOC] | 1. Rispecchia il formato [FDOCK] 2. Non rispecchia il formato [error] |

|  |  |
| --- | --- |
| Parametro: contenuto | Formato: [a-zA-Z0-9 #.:(),!@&<>'’\è\é\ù\ò\"~;$^%{}?]{200,15000}$ |
| Formato[FC] | 1. Rispecchia il formato [FCOK] 2. Non rispecchia il formato [error] |
| Esiste [EC] | 1. Il contenuto non è registrato già nel database [ECOK][if FCOK] 2. Il contenuto è già registrato nel database [error] |

|  |  |
| --- | --- |
| Parametro: categoria |  |
| Selezione [FC] | 1. Un valore selezionato [FCOK] 2. Nessun valore selezionato [error] |

Test case

|  |  |  |
| --- | --- | --- |
| Codice | Combinazione | Esito |
| TC\_1.8\_01 | FT2 | ERROR |
| TC\_1.8\_02 | FT1, ET1 | ERROR |
| TC\_1.8\_03 | FT1, ET2, FDOC2 | ERROR |
| TC\_1.8\_04 | FT1, ET2, FDOC1, FC2 | ERROR |
| TC\_1.8\_05 | FT1, ET2, FDOC1, FC1, EC2 | ERROR |
| Tc\_1.8\_06 | FT1, ET2, FDOC1, FC1, EC1, FC2 | ERROR |
| TC\_1.8\_07 | FT1, ET2, FDOC1, FC1, EC1, FC1 | SUCCESS |

**10. Testing schedule**

L’attività di testing richiede agli sviluppatori di trovare fault nei componenti del sistema ed è meglio che il test venga eseguito da uno sviluppatore che non è stato coinvolto nello sviluppo della componente o da un utente esperto del sistema. Tuttavia in questo caso il test è stato svolto dagli stessi autori del sistema. Testando inizialmente le singole componenti abbiamo cercato di individuare quante più fault possibili nel sistema e quindi successivamente correggerli. Dopo aver corretto gli eventuali fault abbiamo nuovamente testato il sistema per verificare che la modifica non abbia introdotto nuovi fault. Infine siamo passati ai test d’integrazione ripetendo ovviamente lo stesso procedimento precedentemente descritto. L’attività di testing è fondamentale per lo sviluppo corretto di un software, in quanto una tale mancanza potrebbe causare il fallimento dell’intero sistema. Per tale motivo, è fondamentale schedulare il testing.

**10.1 Determinazione dei ruoli**

Le tecniche di test che abbiamo utilizzato sono state scelte principalmente per poter velocizzare l’attività di test in particolar modo la possibilità di eseguire testing in parallelo. Quindi ogni autore del sistema si è occupato, secondo una divisione equa del lavoro, di effettuare test sulle varie componenti.