MedConnect

Test Plan

MedConnect

Riferimento	C05_TP
Versione	2.0
Data	15/12/2024
Destinatario	Ministero della Salute
Presentato da	Cusati Daniel (Mat. 05121 12422) Medica Vincenzo (Mat. 05121 16808) Varone Giuseppe (Mat. 05121 17831)

Revision History

Data	Versione	Cambiamenti	Autori
15/12/2024	0.1	Prima Stesura	Cusati, Medica, Varone
16/12/2024	0.2	Parziale stesura Test Case	Cusati, Medica, Varone
21/12/2024	0.3	Completamento Test Case	Cusati, Medica, Varone
23/12/2024	0.4	Revisione e correzione degli errori	Cusati, Medica, Varone
27/12/2024	0.5	Revisione generale	Cusati, Medica, Varone
30/12/2024	1.0	Revisione	Cusati, Medica, Varone
06/01/2024	2.0	Aggiunto logo e ultima revisione	Cusati, Medica, Varone

Sommario

1 Introduzione

MedConnect nasce con l'obiettivo di rivoluzionare la gestione delle prenotazioni mediche, semplificando l'interazione tra cittadini e sistema sanitario.

Questo documento di Test Plan si propone di descrivere e analizzare le attività di testing per garantire che ogni funzionalità della piattaforma funzioni in modo corretto, soddisfacendo i requisiti del progetto e le aspettative degli utenti finali.

Sono state pianificate attività di testing per le seguenti gestioni:

- Gestione Utente
- Gestione Prenotazione
- Gestione Recensione

2 Relazione con gli altri documenti

Per la corretta individuazione dei test case si fa riferimento ad altri documenti che descrivono il nostro sistema.

Relazione con il documento di raccolta ed analisi dei requisiti (RAD)
La relazione tra questo documento e il RAD riguarda la fase di raccolta dei requisiti funzionali e non funzionali.

Relazione con il System Design Document (SDD)
I test case pianificati nel Test Plan devono rispettare la suddivisione in sottosistemi presentata nell'SDD.

3 Panoramica del sistema

Il sistema proposto si basa su un'architettura three-tier, dal lato frontend usiamo tecnologie quali HTML5, CSS e JavaScript mentre dal lato backend usiamo Java.

4 Funzionalità da testare/da non testare

Di seguito la lista delle funzionalità di cui si effettuerà il testing suddivise nei rispettivi sottosistemi:

- Gestione Utente
  - Inserimento informazioni di base
- Gestione Prenotazione
  - Effettuare prenotazione
- Gestione Recensione
  - Inserimento recensione

Le funzionalità di cui non si andrà ad effettuare le attività di testing riguardano i requisiti funzionali per i quali non sono stati creati gli use case.

5 Pass/Fail criteria

Dopo aver identificato tutti i dati di input del sistema, questi saranno raggruppati in base alle loro caratteristiche comuni. Questa tecnica consentirà di ridurre il numero di test da eseguire, ottimizzando il processo di verifica.

Considereremo il testing riuscito se il sistema evidenzia una failure, ovvero quando l'output prodotto per un determinato input non corrisponde a quello previsto dall'oracolo. In tal caso, la failure sarà analizzata in dettaglio per individuare la causa, si procederà alla sua correzione e verranno eseguiti nuovamente tutti i test necessari per valutare l'impatto della modifica sull'intero sistema.

Al contrario, considereremo il testing fallito se l'output generato dal sistema coincide con quello previsto dall'oracolo, poiché in tal caso non sarà possibile rilevare errori presenti nel sistema.

6 Approccio

La fase di testing del sistema si articola in tre fasi principali: testing di unità, testing di integrazione e testing di sistema. Queste fasi verranno eseguite seguendo l'ordine indicato, assicurando una progressiva verifica della funzionalità e dell'affidabilità del sistema.
Successivamente, saranno definiti specifici test case, nei quali verranno esplicitate le condizioni necessarie per determinare il successo o il fallimento di ogni test. Questo approccio garantirà una valutazione strutturata e oggettiva delle funzionalità implementate.

6.1 Testing di Unità

Il testing di unità si concentra sulla verifica di ogni funzionalità al momento della sua implementazione. I casi di test saranno progettati utilizzando un approccio black-box e documentati direttamente nel codice attraverso l'utilizzo del framework JUnit, specifico per il testing delle classi Java.

Per ogni classe sviluppata verrà definita una corrispondente classe di test, progettata per semplificare ed efficientare il processo di verifica delle singole funzionalità implementate.

6.2 Testing di Integrazione

Per il test di integrazione sarà adottato un approccio bottom-up, considerato particolarmente adatto per un software basato sul paradigma Object-Oriented. Verrà utilizzato Mockito per la creazione dei mock, mentre l'automatizzazione dell'esecuzione dei test sarà gestita tramite Maven. Per la misurazione della copertura del codice e la generazione dei report, si utilizzerà il tool JaCoCo.

Il processo di test seguirà lo stesso schema per tutte le componenti da verificare. In particolare, si inizierà con il test delle classi Service, per poi passare alle classi Controller. Durante i test dei controller, le chiamate verranno simulate utilizzando Mockito per garantire un ambiente controllato e isolato.

6.3 Testing di Sistema

Per il testing di sistema verrà utilizzato il tool Selenium IDE, che consente di registrare e automatizzare le azioni eseguibili da un utente sul browser. Questo strumento sarà impiegato per implementare ed eseguire i test case di sistema. Durante la fase di testing, il server sarà deployato in locale su localhost. Il testing di sistema è suddiviso in diverse attività:

1. **Functional testing**
2. **Performance Testing**: a causa del basso budget a disposizione, non si assicura l'esecuzione del performance testing;
3. **Pilot Testing**: a causa del basso budget a disposizione, non si assicura l'esecuzione del pilot testing;
4. **Acceptance Testing**: esso verrà effettuato solo sul functional testing, ed un componente del nostro Team simulerà la figura del cliente;
5. **Installation Testing**: a causa del basso budget a disposizione, non si assicura l'esecuzione dell'installation testing.

7 Sospensione e ripristino

In questa sezione sono definiti i criteri di sospensione dei test e le attività da ripetere per la ripresa del testing.

7.1 Criteri di sospensione

Il processo di testing proseguirà fino al completamento, anche in caso di rilevazione di una failure. Tuttavia, il testing potrà essere temporaneamente sospeso nel caso in cui, durante l'esecuzione, si verifichi un errore nella definizione di uno o più test, rendendoli non eseguibili.

7.2 Criteri di ripristino

Il testing riprenderà una volta risolti i fault identificati, garantendo che i test siano correttamente definiti ed eseguibili.

8 Materiale per effettuare il testing

L'hardware richiesto per l'attività di test è un computer standard, non necessariamente connesso a Internet, poiché il sistema verrà eseguito localmente su **localhost** poiché il sistema non è ancora stato rilasciato.

9 Test Cases

L'approccio scelto per definire i test frame sarà il category partition. Gli input verranno suddivisi in classi di equivalenza per ridurre al minimo il numero di test case. L'output atteso sarà determinato da un oracolo umano, data la mancanza di specifiche formali o semi-formali.

9.1 Gestione Prenotazione

9.1.1 Effettuare prenotazione

Parametro : id_disponibilità	
FORMATO: ^[0-9]+$	
Nome Categoria	**Scelta per categoria**
Formato [FS]	1. Rispetta il formato = false [error] 2. Rispetta il formato = true [PROPERTY FS_OK]
Parametro : Nota	
Nome Categoria	**Scelta per categoria**
Lunghezza [LN]	1. Lunghezza > 255 = false [error] 2. Lunghezza <= 255 = true [PROPERTY LN_OK]

Test Case ID	Test Frame	Esito
TC_1.1	**FS1**	**Errato** : id_disponibilità corrotto.
TC_1.2	**FS2, LN1**	**Errato** : lunghezza nota della recensione non corretta.
TC_1.3	**FS2, LN2**	**Corretto.**

9.2 Gestione Utente

9.2.1 Inserimento informazioni di base

Parametro: Biografia	
Nome Categoria	**Scelte per la categoria**
Lunghezza [LB]	1. Lunghezza > 255 = false [error] 2. Lunghezza <= 255 = true [PROPERTY LB_OK]
Parametro: Data di nascita	
FORMATO: ^(?:(?:31(\/|-|\.)(?:0?[13578]|1[02]))\1|(?:(?:29|30)(\/|-|\.)(?:0?[13-9]|1[0-2])\2))(?:(?:19|2[0-9])\d{2})$\|^(?:29(\/\|-\|\.)0?2\3(?:(?:(?:19\|2[0-9])(?:0[48]\|[2468][048]\|[13579][26])\|(?:(?:16\|[2468][048]\|[3579][26])00))))$|^(?:0?[1-9]|1\d|2[0-8])(\/|-|\.)(?:(?:0?[1-9])|(?:1[0-2]))\4(?:(?:19|2[0-9])\d{2})$ **NOTA BENE:** questo formato valida una data nel formato gg/mm/aaaa (o varianti con i separatori "/", "-", "."), con anni che partono dal 1900.	
Nome Categoria	**Scelte per la categoria**
Formato [FDN]	1. Rispetta il formato = false [error] 2. Rispetta il formato = true [PROPERTY FDN_OK]

Parametro: Luogo di nascita	
FORMATO: ^[a-zA-ZÀ-ÖØ-öø-ÿ\'\-\s]{1,50}$ **NOTA BENE:** Accetta stringhe fino a 50 caratteri, non accetta stringhe vuote e stringhe con simboli. Accetta lettere accentate di ogni tipo.	
Nome Categoria	**Scelte per la categoria**
Formato [FLN]	1. Rispetta il formato = false [error] 2. Rispetta il formato = true [PROPERTY FLN_OK]
Parametro: Codice Fiscale	
FORMATO: ^[A-Z]{3}[A-Z]{3}\d{2}[ABCDHLMPRST](?:0[1-9] | [1-2]\d | 3[0-1] | 4[1-9] | 5[0-9] | 6[0-9] | 7[0-1])[A-Z]\d{3}[A-Z]$ **NOTA BENE:** Questa regex verifica la correttezza formale di un codice fiscale italiano	
Nome Categoria	**Scelte per categoria**
Formato [FCF]	1. Rispetta il formato = false [error] 2. Rispetta il formato = true [PROPERTY FCF_OK]
Parametro : Genere	
FORMATO : ^(maschio | femmina | altro)$	
Nome Categoria	**Scelte per categoria**
Formato [FG]	1. Rispetta il formato = false [error] 2. Rispetta il formato = true [PROPERTY FG_OK]
Parametro : Numero di cellulare	
FORMATO: ^\d{10}$	

Nome Categoria	**Scelte per categoria**
Formato [FNC]	1. Rispetta il formato = false [error] 2. Rispetta il formato = true [PROPERTY FNC_OK]
Parametro : Firma	
FORMATO: ^(medico|paziente)$	
Nome Categoria	**Scelte per categoria**
Formato [FF]	1. Rispetta il formato = false [error] 2. Rispetta il formato = true [PROPERTY FF_OK]

Test Case ID	**Test Frame**	**Esito**
TC_2.1	**LB1**	**Errato:** lunghezza biografia non valida.
TC_2.2	**LB2, FDN1**	**Errato:** data di nascita non valida.
TC_2.3	**LB2, FDN2, FLN1**	**Errato**: luogo di nascita non valido.
TC_2.4	**LB2, FDN2, FLN2, FCF1**	**Errato**: codice fiscale non valido.
TC_2.5	**LB2, FDN2, FLN2, FCF2, FG1**	**Errato**: genere non valido.
TC_2.6	**LB2, FDN2, FLN2, FCF2, FG2, FNC1,**	**Errato**: numero di cellulare non valido.
TC_2.7	**LB2, FDN2, FLN2, FCF2, FG2, FNC2, FF1**	**Errato**: Firma non valida.
TC_2.8	**LB2, FDN2, FLN2, FCF2, FG2, FNC2, FF2**	**Corretto.**

9.3 Gestione Recensione

9.3.1 Inserimento recensione

Parametro : Stelle	
FORMATO: ^[1-5]$	
Nome Categoria	**Scelta per categoria**
Formato [FS]	1. Rispetta il formato = false [error] 2. Rispetta il formato = true [PROPERTY FS_OK]
Parametro : Nota	
Nome Categoria	**Scelta per categoria**
Lunghezza [LN]	1. Lunghezza > 255 = false [error] 2. Lunghezza <= 255 = true [PROPERTY LN_OK]

Test Case ID	**Test Frame**	**Esito**
TC_3.1	**FS1**	**Errato** : numero stelle errato.
TC_3.2	**FS2, LN1**	**Errato** : lunghezza nota della recensione non corretta.
TC_3.3	**FS2, LN2**	**Corretto.**

10 Testing schedule

I test saranno eseguiti sia durante l'implementazione del sistema, per identificare e risolvere tempestivamente eventuali problemi, sia al termine dello sviluppo. Una volta completata la fase di sviluppo, tutti i test verranno rieseguiti in modo sistematico per verificare il corretto funzionamento del sistema nel suo insieme e generare i report finali.