Test Plan

MediCare

Riferimento	C14_TP_ver2.0.pdf
Versione	2.0
Data	05/12/2023
Destinatario	F. Ferrucci, F. Palomba
Presentato da	Primo Vinicio Calabrese Giovanni Casaburi Matteo Avella Gianluca Palumbo Salvatore Basilicata Domenico Alessandro Urciuoli
Approvato da	Luca Contrasto, Matteo Cicalese

Team Members

Ruolo	Nome	Acronimo	Contatto
Project Manager	Matteo Cicalese	MC	m.cicalese18@studenti.unisa.it
Project Manager	Luca Contrasto	LC	l.contrasto@studenti.unisa.it
Team Member	Primo Vinicio Calabrese	PVC	p.calabrese17@studenti.unisa.it
Team Member	Giovanni Casaburi	GC	g.casaburi16@studenti.unisa.it
Team Member	Matteo Avella	MA	m.avella17@studenti.unisa.it
Team Member	Gianluca Palumbo	GP	g.palumbo40@studenti.unisa.it
Team Member	Salvatore Basilicata	SB	s.basilicata@studenti.unisa.it
Team Member	Domenico Alessandro Urciuoli	DAU	d.urciuoli2@studenti.unisa.it

Revision History

Data	Versione	Descrizione	Autori
05/12/2023	0.1	Prima stesura	Luca Contrasto, Matteo Cicalese
06/12/2023	0.2	Caricamento Test Case	Matteo Avella
06/12/2023	0.3	Caricamento Test Case	Giovanni Casaburi
07/12/2023	0.4	Caricamento Test Case	Primo Vinicio Calabrese
07/12/2023	0.5	Caricamento Test Case	Domenico Alessandro Urciuoli
08/12/2023	0.6	Caricamento Test Case	Salvatore Basilicata
09/12/2023	0.7	Revisione Test Case	Matteo Avella
09/12/2023	0.8	Revisione Test Case	Tutti i team members

10/12/2023	0.9	Caricamento Test Case	Gianluca Palumbo
12/12/2023	1.0	Modifica e Revisione	Luca Contrasto, Matteo Cicalese
22/01/2024	2.0	Preparazione alla consegna	Gianluca Palumbo, Giovanni Casaburi, Matteo Avella, Salvatore Basilicata, Primo Vinicio Calabrese

Sommario

1. Introduzione 6
2. Relazione con altri documenti 6
3. Panoramica del sistema 7
4. Features da testare / da non testare 7
5. Pass/ Fail criteria 8
6. Approccio 8
7. Sospensione e ripristino 10
8. Materiale di testing 10
9. Test cases 10
   1.1 Gestione Utente 11
   1.2 Gestione Autenticazione 11
   1.3 Gestione Prenotazione 12
   1.4 Gestione Farmaci 13
   1.5 Gestione Ente 14
2. Testing schedule 15

1. Introduzione

MediCare è una piattaforma web che intende offrire agli utenti una soluzione completa per il controllo delle attività sanitarie degli utenti, in modo da sopperire le mancanze di varie piattaforme, che sebbene forniscano servizi analoghi, risultano essere lente, inaccessibili e malfunzionanti, comprese quelle statali.

Il documento di Test Plan ha l'obiettivo di descrivere ed analizzare le attività di Testing per la piattaforma MediCare con lo scopo di garantire che ogni aspetto funzioni in modo corretto.

All'interno del documento sono riportate le strategie di testing adottate, quali funzionalità saranno testate e gli strumenti scelti per la rilevazione degli errori, con lo scopo di presentare al cliente finale una piattaforma priva di malfunzionamenti. Sono state pianificate attività di testing per le seguenti gestioni:

1.1 Gestione Utente

1.2. Gestione Prenotazioni

1.3. Gestione Farmaci

1.4. Gestione Autenticazione

1.5. Gestione Ente

2. Relazione con altri documenti

Per la corretta individuazione dei test case, si fa riferimento ad altri documenti prodotti.

Relazioni con il Requirements Analysis Document (RAD)

I test case pianificati nel Test Plan sono elaborati in relazione ai requisiti funzionali e non funzionali presentati nel RAD.

Relazioni con il System Design Document (SDD)

I test case pianificati nel Test Plan devono rispettare la suddivisione in sottosistemi presentata nell'SDD.

Relazioni con il Object Design Document (ODD)

Per quanto riguarda il test di unità e di integrazione, maggiormente legato all'ODD e alla divisione in package del sistema, essi saranno scritti e documentati unicamente all'interno del codice dell'applicativo. Per tale motivo, nel presente documento, non vi saranno riferimenti al loro design.

3. Panoramica del sistema

Il sistema proposto basa la sua architettura sul sistema three-tier, in particolare usando un sistema tramite Python e i framework Flask e Jinja.

Inoltre, verranno usati HTML5, CSS3, Bootstrap, Taiwind CSS e JS per la parte di front-end e la generazione delle view.

Per la logica applicativa e quindi il back-end utilizzato Python. Per la gestione del database saranno usati:

3.1. SQLAlchemy per il collegamento al database.

3.2. MySQL come database in fase di produzione

4. Features da testare / da non testare

Di seguito la lista delle features di cui si effettuerà il testing per le varie gestioni:

- 4.1 Gestione Autenticazione
  - 4.1.1 Login
- 4.2 Gestione Prenotazioni
  - 4.2.1 Pagamento prestazione sanitaria
  - 4.2.2 Prenotazione Visita Medica
- 4.3 Gestione Utente
  - 4.3.1 Gestione ISEE
- 4.4 Gestione Ente
  - 4.4.1 Creazione account medico pubblico

Le funzionalità di cui non si andrà ad effettuare le attività di testing riguardano requisiti funzionali di bassa o media priorità; sono inoltre escluse le funzionalità che non prevedono input manuale da parte dell'utente - ad esempio attività riguardanti esclusivamente visualizzazioni di dati.

5. Pass/ Fail criteria

Le attività di testing sono mirate ad identificare la presenza di faults (errori) all'interno del sistema, per effettuarne un successivo intervento di eliminazione.

L'esito di un test case è valutato mediante un oracolo, inteso come il risultato atteso della sua esecuzione, basandosi sui requisiti.
Un test ha successo (pass) se, dato un input al sistema, l'output ottenuto è diverso dall'output atteso dall'oracolo.
Un test fallisce (fail) se, dato un input al sistema, l'output ottenuto è uguale all'output atteso dall'oracolo.

L'attività di testing darà considerata valida se tutti i seguenti vincoli saranno rispettati:

- effettuare test di regressione ogni volta che si introducono nuove caratteristiche al sistema o vengono modificate quelle presenti;
- ogni team member dovrà effettuare il testing di unità di esattamente un metodo di una classe sviluppata;
- ogni team member dovrà effettuare il testing di sistema di esattamente una funzionalità dell'applicazione.

6. Approccio

Il testing dell'intero sistema si compone di tre fasi: testing di sistema, testing di integrazione e testing di unità. Verranno progettati nell'ordine appena definito, ma verranno eseguiti in ordine inverso.

Prima della fase di implementazione del sistema, avverrà la progettazione dei casi di test di sistema, perfezionati in seguito nella loro fase di esecuzione; durante la fase implementativa avverrà la progettazione dei casi di test di unità.

Durante lo sviluppo saranno eseguite periodiche attività di revisione sul codice prodotto.

L'approccio utilizzato per eseguire il testing sarà di tipo Bottom-up (dal basso verso l'alto), dunque si effettuerà il testing e l'integrazione delle singole componenti partendo dal livello più basso, ovvero il database, passando per il layer di controllo, e concludendo con le interfacce degli utenti.

Per il testing, ci serviremo di PyUnit per effettuare il testing di unità e di sistema, Coverage.Py per l'analisi della code coverage e la valutazione dell'efficacia dei test, e Selenium per testare l'interazione dell'utente sulla piattaforma.

Testing di Unità

Per il testing di unità la strategia prevista consiste nel testare ogni metodo delle classi del sistema. Da esse, sono escluse le interfacce e le classi entity, poiché quest'ultime presentano solo metodi getters e setters. I casi di test saranno definiti attraverso un approccio black-box e saranno documentati direttamente nel codice, attraverso l'uso del framework per il testing di classi Java *JUnit*.

Per ogni Production Class sarà definita una Test Class che rispetterà il formato *NomeProductionClassTest*. Tali classi saranno scritte in parallelo alle Production class, per garantire una più facile copertura del codice. Le stesse classi saranno poi revisionate e modificate da sviluppatori differenti.

Altra tecnologia usate in tale fase sarà *Coverage.py*: per il calcolo di metriche, tra le quali la Branch Coverage.

Testing di Integrazione

Il test di integrazione viene eseguito per testare che, componenti che autonomamente funzionano in modo corretto, funzionano bene anche quando integrati con altri componenti, in modo da individuare eventuali problemi legati alle interfacce degli stessi.

Verrà utilizzato un approccio di tipo "bottom-up", che prevede l'utilizzo dei driver necessari a simulare le componenti più ad alto livello non ancora testate.

In questa fase si prevede di testare dapprima le componenti inerenti alla logica di persistenza con il DB SQL, e successivamente andare ad integrare anche le componenti di servizio, che dipendono strettamente dalle componenti di persistenza già integrate con il DB.

Testing di Sistema

Mentre le fasi precedenti si focalizzano sulla ricerca di bug nelle componenti individuali e nelle interfacce tra le componenti, con il system testing assicura che il sistema completo si comporti in modo conforme ai requisiti funzionali e non funzionali, di conseguenza non è di primaria importanza avere a disposizione il codice sorgente del sistema.
A tal proposito, per questa fase si prevede l'utilizzo di un approccio black-box per la definizione dei test suites, in particolare adottando la metodologia Category Partition, che permette nel complesso di ottenere un buon livello di dettaglio considerando il tempo a disposizione.
Il system testing, dato che verrà eseguito nelle fasi finali del progetto, verrà effettuato con l'ausilio del framework Selenium per testare in modo completo l'interfaccia utente.

7. Sospensione e ripristino

In questa sezione verranno specificati i criteri di sospensione del test e le attività di test che dovranno essere ripetute quando si riprende il test.

Criteri di sospensione

Il testing non verrà sospeso fino alla sua terminazione, anche in caso di rilevazione di una failure. Il testing non verrà sospeso fino alla sua terminazione, anche in caso di rilevazione di una failure. Il testing potrà essere momentaneamente sospeso nel caso venga restituito, al momento dell'esecuzione, un errore nella definizione di uno dei test stessi.

Criteri di ripristino

Il testing verrà ripreso dopo aver risolto i fault individuati.

8. Materiale di testing

L'hardware necessario per l'attività di test è un semplice computer, non necessariamente connesso ad internet, in quanto il sistema non è stato ancora rilasciato.

9. Test cases

L'approccio per la definizione dei test frame sarà il category partition. Al fine di minimizzare il numero di test case, gli input saranno partizionati in classi di equivalenza. Per definire l'output atteso si userà un oracolo umano, per via dell'assenza di specifiche formali/semi-formali.

1.1 Gestione Utente

TC_GU.3: Inserimento e Gestione ISEE

Parametro: ISEE Ordinario	
Formato: ^[0-9]+ (\.\[0-9]{1,2})?$	
Nome categoria	**Scelte per la categoria**
Formato [FD]	1. Formato rispettato = false [errore] 2. Formato rispettato = true [PROPERTY_FE1_OK]

Test Case ID	Test Frame	Esito
TC_GU.3_1	FD1	Errore: formato ISEE non valido
TC_GU.3_2	FD2	Corretto: Esenzione Applicata

1.2 Gestione Autenticazione

TC_GA.1: Login

Parametro: E-mail	
Formato: ^[A-z0-9\.\+_-]+@[A-z0-9\._-]+\.[A-z]{2,6}$	
Nome categoria	**Scelte per la categoria**
Formato [FE]	1. Formato rispettato = false [error] 2. Formato rispettato = true [PROPERTY_FE_OK]
Match [ME]	1. E-mail Match in DB = true [error] 2. E-mail Match in DB = false [PROPERTY_ME_OK]
Parametro: Password	
Nome categoria	**Scelte per la categoria**
Lunghezza [LN]	1. Lunghezza > 255 AND Lunghezza < 8 [error] 2. Lunghezza <= 255 AND Lunghezza >= 8 [PROPERTY_LN_OK]
Match [MP]	1. Password Match in DB = false [error] 2. Password Match in DB = true [PROPERTY_MP_OK]

Test Case ID	Test Frame	Esito
TC_GA.1_1	FE1	Errore: formato e-mail non valido
TC_GA.1_2	FE2, LN1	Errore: lunghezza password non rispettata
TC_GA.1_3	FE2, LN2, ME1	Errore: e-mail non presente nel DB

TC_GA.1_4	FE2, LN2, ME2, MP1	Errore: password non presente nel DB
TC_GA.1_5	FE2, LN2, ME2, MP2	Corretto

1.3 Gestione Prenotazione

TC_GP.6: Pagamento prestazione sanitaria

Parametro: PAN	
Formato: ^(\d\s?){16}$	
Nome categoria	**Scelte per la categoria**
Formato [FP]	1. Formato rispettato = false [errore] 2. Formato rispettato = true [PROPERTY_FP_OK]
Match [MP]	1. PAN match in DB = true [errore] 2. PAN match in DB = false [PROPERTY_MP_OK]
Parametro: DATA SCADENZA	
Formato: ^(0[1-9]|1[0-2])\/(\d{4})$	
Nome categoria	**Scelte per la categoria**
Formato [FD]	1. Formato rispettato = false [errore] 2. Formato rispettato = true [PROPERTY_FD_OK]
Parametro: CVV	
Formato: ^\d{3}$	
Nome categoria	**Scelte per la categoria**
Formato [FC]	1. Formato rispettato = false [errore] 2. Formato rispettato = true [PROPERTY_FC_OK]

Test Case ID	Test Frame	Esito
TC_GP.6_1	FP1	Errore: formato PAN non valido
TC_GP.6_2	FP2, FD1	Errore: formato DATA SCADENZA non valido
TC_GP.6_3	FP2, FD2, FC1	Errore: formato CVV non valido
TC_GP.6_4	FP2, FD2, FC2, MP1	Errore: Metodo di pagamento già in database

TC_GP.6_5	FP2, FD2, FC2, MP2	Corretto

TC_GP.4: Prenotazione Visita Medica

Parametro: DataVisita	
Nome categoria	**Scelte per la categoria**
Lunghezza[LD]	1.Lunghezza>10 and Lunghezza<10 [error] 2.Lunghezza==10 [Property_LD_OK]
Parametro: OraVisita	
Nome categoria	**Scelte per la categoria**
Lunghezza[LO]	1.Lunghezza>2 And Lunghezza<1 [error] 2.Lunghezza<=2 [Property_LO_OK]
Valore[VO]	1.Valore<9 and Valore>19 [error] 2.Valore>=9 and Valore<=19 [Property_VO_OK]
Parametro: Carta	
Formato: ^\d{16}	
Nome categoria	**Scelte per la categoria**
Formato[FC]	1.Formato rispettato=false[error] 2.Formato rispettato=True[Property_FC_OK]

Test Case ID	Test Frame	Esito
TC_GP.4_1	LD1	Errore: la data non è della lunghezza esatta
TC_GP.4_2	LD2, LO1	Errore: l'ora non è stata fornita
TC_GP.4_3	LD2, LO2, VO1	Errore: ora inserita non valida
TC_GP.4_4	LD2, LO2, VO2, FC1	Errore: formato carta non valido
TC_GP.4_5	LD2, LO2, VO2, FC2	Corretto

1.4 Gestione Farmaci

TC_GF.2: Filtro farmaci

Parametro: Farmaco	
Nome categoria	**Scelte per la categoria**
Match [MP]	1. Farmaco Match in DB = false [error] 2. Farmaco Match in DB = true [PROPERTY_MP_OK]
Match [MC]	1. Farmaco Match in DB = false [error]

	2. Farmaco Match in DB = true [PROPERTY_MP_OK]

Test Case ID	Test Frame	Esito
TC_GF.2_1	MC1	Errore: farmaco di quella categoria non trovato
TC_GF.2_2	MC2	Corretto
TC_GF.2_3	MP1	Errore: farmaco di quel prezzo non trovato
TC_GF.2_4	MP2	Corretto
TC_GF.2_5	MC1, MP1	Errore: farmaco di quella categoria e prezzo non trovato
TC_GF.2_6	MC2, MP2	Corretto

1.5 Gestione Ente

TC_GE.1: Creazione account medico pubblico

Parametro: E-mail	
Formato: ^[A-z0-9\.\+_-]+@[A-z0-9\._-]+\.[A-z]{2,6}$	
Nome categoria	**Scelte per la categoria**
Formato [FE]	1. Formato rispettato = false [errore] 2. Formato rispettato = true [PROPERTY_FE_OK]
Match [ME]	1. E-mail match in DB = true [errore] 2. E-mail match in DB = false [PROPERTY_ME_OK]
Parametro: Password	
Nome categoria	**Scelte per la categoria**
Lunghezza [LN]	1. Lunghezza > 20 AND Lunghezza < 8 [errore] 2. Lunghezza <= 20 AND Lunghezza >= 8 [PROPERTY_LN_OK]

Test Case ID	Test Frame	Esito
TC_GE.1_1	FE1	Errore: formato e-mail non valido
TC_GE.1_2	FE2, LN1	Errore: lunghezza password non rispettata
TC_GE.1_3	FE2, LN2, ME1	Errore: e-mail già presente nel DB
TC_GE.1_4	FE2, LN2, ME2	Corretto

2.Testing schedule

Le attività di pianificazione del testing avverranno come definito nei capitoli precedenti, cioè subito dopo la fase di design necessaria per la pianificazione.

La scrittura dei casi di test avverrà in contemporanea con lo sviluppo del codice.

L'esecuzione dei test avverrà sia durante che dopo l'implementazione del sistema. Una volta concluso lo sviluppo, tutti i test saranno rieseguiti per garantirne il corretto funzionamento e produrre i report finali. Per altre informazioni si rimanda ai documenti di management sulle schedule.