MedQueue

TP

Test Plan

MedQueue

Afeltra Angelo
Amato Adriano
Fucile Andrea
Rapa Giovanni

Sommario

1. INTRODUZIONE

Nell'affrontare alcuni aspetti fondamentali del sistema MedQueue ci siamo posti delle domande:

Come ottenere un buon prodotto? Può rimanere soltanto un'idea?

È difficile dare una risposta a tale domanda ma sicuramente possiamo garantire all'utente una buona fruizione delle funzionalità offerte se mettiamo in campo un ottimo strumento in grado di migliorare la sua esperienza.
Nasce così la necessita di rilevare eventuali errori prodotti durante la fase d'implementazione per evitare che essi si presentino nel momento in cui il sistema verrà rilasciato all'utente finale.

Lo scopo di questo documento è quello di pianificare l'attività di testing del sistema MedQueue.
In questa fase occorre verificare il corretto funzionamento del sistema sotto determinate condizioni.
Abbiamo pensato ad opportuni casi e dati di input specifici in grado di mettere alla prova ogni singola funzionalità e caratteristica offerta dalla piattaforma.
I risultati dei test che verranno eseguiti saranno il punto cruciale nell'analisi delle failure e delle loro cause (fault) per individuare dove bisognerà intervenire per correggere gli errori o apportare modifiche per il miglioramento dei vari sottosistemi.

2. Documenti Correlati

2.1 Relazione con il documento di analisi

La progettazione dei casi di test avviene indipendentemente dalla struttura interna del prodotto ed operando solo sulle specifiche.
Per questo motivo facciamo riferimento al contenuto del documento di analisi dove vengono descritte dettagliatamente le funzionalità del sistema attraverso scenari e use case

2.2 Relazione con il System Design Document

Nel system design abbiamo definito la suddivisione in sottosistemi relativa al prodotto che intendiamo presentare.
Il sistema è suddiviso in tre livelli logici: presentazione, business e persistenza. Ogni livello è composto da vari sottosistemi.
In questa fase è importante focalizzare l'attenzione sul layer di business. Infatti pianificheremo le attività di testing relative alle funzionalità garantite nei sottosistemi specificati all'interno del System Design Document relativamente al livello di business.

2.3 Relazione con l'Object Design Document

Nel documento di Object Design sono state definite le classi che compongono il sistema e le loro mansioni.
Faremo riferimento ad esse nel corso del documento per associare i test al codice prodotto.

3. Panoramica del sistema

MedQueue è una piattaforma web che mira all'ottimizzazione delle code ospedaliere offrendo a qualsiasi persona la possibilità di poter effettuare una prenotazione presso una struttura sanitaria per un determinato giorno e una determinata ora ed all'impiegato della struttura ospedaliera la possibilità di accettare le prenotazioni.

Il sistema che proponiamo prevede due attori principali:
- **Utente**: ha la possibilità di effettuare una prenotazione presso una struttura ospedaliera
- **Impiegato Ospedaliero**: ha la responsabilità di servire le prenotazioni ricevute.

Nel System Design abbiamo definito l'architettura della piattaforma che si divide in tre layer: presentazione, business e persistenza.
I sottosistemi che compongono i vari livelli logici collaborano tra loro cercando però di garantire il più possibile basso accoppiamento ad alta coesione.

I sottosistemi individuati nel business layer sono:
- **Accesso**: Definisce l'utente generico del sistema ed offre tutti i servizi relativi all'applicazione
- **Prenotazioni**: Modella il lato d'inserimento, eliminazione, visualizzazione e validazione delle prenotazioni da parte degli utenti
- **Gestione**: Modella il lato di gestione delle prenotazioni da parte dell'impiegato
- **Visualizzazione Coda**: Modella le operazioni di visualizzazione coda

4. Funzionalita da testare

La fase di testing avrà come obiettivo quello di testare interamente la comunicazione tra webapp e applicazione desktop con il database sottostante, cosi come l'implementazione della logica di business e il layer di presentazione.
Il testing funzionale riguarderà nel dettaglio le funzionalità elencate:

- **Accesso**
  - Registrazione alla piattaforma
  - Autenticazione alla piattaforma

- **Prenotazioni**
  - Richiesta prenotazione
  - Convalida prenotazione
  - Visualizzazione prenotazioni
  - Elimina prenotazione
- **Visualizza Coda**
  - Visualizzazione coda prenotazioni
- **Gestione**
  - Accettazione prenotazione

Non saranno testate invece:
- Sicurezza
- Performance

TP – Test Plan

5. Criteri Pass/Failed

Abbiamo determinato un insieme d'input possibili che possono aiutarci a scovare errori nel sistema.
Pertanto il test ha successo se il comportamento osservato è diverso dal comportamento specificato nei requisiti funzionali.
Ciò comporta l'individuazione di un fault nel sistema.
In tal caso analizzeremo i sottosistemi convolti nell'errore, e procederemo con la correzione dell'errore
Il testing fallirà se non saranno scovati errori nelle componenti

6. Approccio

La fase di testing si compone di tre attività: una prima fase si occuperà di trovare errori in una singola componente; la seconda fase invece, avrà come compito quello di testare le funzionalità nate dall'integrazione dei vari sottosistemi e per ultimo andremo a testare l'intero sistema assemblato al fine di verificare che esso soddisfi i desideri del cliente.

Di seguito verranno descritte brevemente le strategie individuate per effettuare il test di unità, d'integrazione e di sistema.

6.1 Testing di unità

Durante tale fase di testing verranno testate singolarmente le componenti al fine di evidenziare gli errori.
Per il testing di unità si utilizza la tecnica di "Black-Box testing" che si focalizza sul comportamento di I/O, senza preoccuparsi della struttura interna della componente.
Se per ogni dato input, siamo in grado di prevedere l'output, allora l'unità supera il test.
Siccome è quasi impossibile generare tutti i possibili input, riduciamo il numero di casi di test effettuando un partizionamento: si dividono le condizioni di input tramite il category partition e si generano i test case.
Nel momento in cui utilizzando solo la tecnica di "Black-Box testing" non otteniamo che il 75% dei branch sviluppati sia testato, integreremo anche la tecnica di "White-Box-testing".
Il "White-Box-testing" si focalizza sulla completezza, dove ogni statemente nella componente è eseguita almeno una volta.

6.2 Testing di integrazione

Una volta che sono stati rilevati i bug per una singola componente e riparati, le componenti sono pronte per essere integrate in sottosistemi più grandi.
Pertanto dopo aver testato singolarmente le componenti del sistema, possiamo procedere a testarne le integrazioni.
Per il testing d'integrazione si utilizza la strategia: "bottom-up".
I sottosistemi al livello più basso della gerarchia sono testati individualmente.
I successivi sottosistemi ad essere testati sono quelli che chiamano i sottosistemi testati in precedenza. Si ripete quest'ultimo passo finché tutti i sottosistemi non sono stati testati

6.3 Testing di Sistema

L'ultimo testing prima della messa in uso del sistema prevede il controllo delle funzionalità del sistema, secondo i requisiti specificati.
Si utilizzerà il tool "Selenium" per simulare l'iterazione da parte dell'utente con la web app, e katalon per simulare l'iterazione da parte dell'utente con l'app desktop.

7. Sospensione e ripresa

Tenuto conto delle risorse necessarie impiegate durante la fase di testing, abbiamo stabilito dei criteri in base ai quali le attività di test saranno sospese o riprese.

7.1 Criteri di sospensione

Il test e sospeso se almeno il 10% dei casi di test riportano errori: in queste condizioni, il team deve provvedere a correggere i fault prima di procedere con l'implementazione o il testing di nuove funzionalità

7.2 Criteri di ripresa

Abbiamo previsto delle modifiche future al sistema dopo il rilascio.
Pertanto sarà necessario, dopo aver introdotto i cambiamenti, testare le nuove componenti: se esse introducono dei fault che impattano sulle componenti già esistenti, allora verranno testate nuovamente anche quest'ultime

7.3 Criterio di terminazione

Il test si considera terminato, quando la totalità dei casi di test somministrati al sistema riporta esito negativo.
Come da indicazione del top management, la suddetta condizione sussiste solo se il 75% dei branch sviluppati viene ricoperto in questa fase

8. Materiale per il testing

L'esecuzione dei test necessita di un server correttamente configurato su cui siano installati java e MySQL.

La configurazione deve avvenire come da manuale d'installazione.

Il testing è condotto utilizzando alcuni dei framework più famosi ed efficaci in ambienti Java: JUnit, Mockito e Selenium.

L'utilizzo di JUnit riguarda sia il testing di unità che quello di integrazione, mentre Mockito e utilizzato solo nel testing d'unita per mascherare le dipendenze.

Selenium sarà utilizzato per il testing di sistema.

9. Test Cases

Per ogni sottosistema mostriamo le funzionalità che andremo a testare. Inoltre, associamo ad ogni funzionalità una tabella in cui per ogni parametro, vengono definite le relative categorie, insieme alle possibili scelte.

9.1 Accesso

9.1.1 Autenticazione Impiegato

Per il sottosistema Accesso abbiamo previsto di testare la funzionalità di autenticazione alla piattaforma.

Tale funzionalità prevede la possibilità da parte di un impiegato di autenticarsi alla piattaforma mediante:

- Codice Fiscale: Stringa del formato “MNDCMN97R22A509S”
- Password: Stringa con un numero di caratteri compreso tra 6 e 50

Parametro	Codice Fiscale
Formato [FCF]	1. [a-zA-Z]{6}[0-9]{2}[a-zA-Z][0-9]{2}[a-zA-Z][0-9]{3}[a-zA-Z] **[propertyformatoFCFok]** 2. Non rispetta il formato **[error]**
Riscontro [R]	1. Trovato **[if formatoFCFok] [propertyriscontroRok]** 2. Non trovato **[if fomratoFCFok] [error]**

Parametro	Password
Formato [FP]	1. ^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{6,}$ **[propertyformatoFPok]** 2. Non rispetta il formato **[error]**
Corrispondenza [C]	1. Corrisponde **[if formatoFPok] [propertyCorrispondenzaCok]** 2. Non trovato **[if formatoFPok] [error]**

Codice	Combinazione	Esito
TC_GI_1:1	FCF2	Negativo
TC_GI_1:2	FCF1.R2	Negativo
TC_GI_1:3	FCF1.R1.FP2	Negativo
TC_GI_1:4	FCF1.R1.FP1.C2	Negativo
TC_GI_1:5	FCF1.R1.FP1.C1	Positive

9.1.2 Autenticazione Utente

Per il sottosistema Accesso abbiamo previsto di testare la funzionalità di autenticazione alla piattaforma.

Tale funzionalità prevede la possibilità da parte di un utente registrato di autenticarsi alla piattaforma mediante:

- Codice Fiscale: Stringa del formato “MNDCMN97R22A509S”
- Password: Stringa con un numero di caratteri compreso tra 6 e 50

Parametro	Codice Fiscale
Formato [FCF]	1. [a-zA-Z]{6}[0-9]{2}[a-zA-Z][0-9]{2}[a-zA-Z][0-9]{3}[a-zA-Z] **[propertyformatoFCFok]** 2. Non rispetta il formato **[error]**
Riscontro [R]	1. Trovato **[if formatoFCFok] [propertyriscontroRok]** 2. Non trovato **[if formatoFCF ok] [error]**

Parametro	Password
Formato [FP]	1. ^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{6,}$ **[propertyfromatoFPok]** 2. Non rispetta il formato **[error]**
Corrispondenza [C]	1. Corrisponde **[if propertyformatoFPok] [propertycorrispondenzaCok]** 2. Non trovato **[if propertyformatoFPok] [error]**

Codice	Combinazione	Esito
TC_GU_1:1	FCF2	Negativo
TC_GU_1:2	FCF1.R2	Negativo
TC_GU_1:3	FCF1.R1.FP2	Negativo
TC_GU_1:4	FCF1.R1.FP1.C2	Negativo
TC_GU_1:5	FCF1.R1.FP1.C1	Positive

9.1.3 Registrazione Account

Per il sottosistema Accesso abbiamo previsto di testare la funzionalità di registrazione alla piattaforma.

Tale funzionalità prevede la possibilità di registrarsi alla piattaforma compilando un modulo in cui risulta necessario l'inserimento dei dati personali quali:

- Codice Fiscale: Stringa del formato "MNDCMN97R22A509S"
- Nome: Stringa con un numero di caratteri compreso tra 2 e 50
- Cognome: Stringa con un numero di caratteri compreso tra 2 e 50
- Password: Stringa con un numero di caratteri compreso tra 6 e 50
- Data di nascita
    - Anno di nascita: intero compreso tra l'anno corrente -17 e l'anno corrente -130
    - Mese di nascita: intero compreso tra 1 e 12
    - Giorno di nascita: intero compreso tra 1 e 31
- Indirizzo Email: Stringa con un numero di caratteri compreso tra 8 e 255
- Numero di telefono: Stringa con un numero di caratteri uguale a 14

Parametro	Codice Fiscale
Formato [FCF]	1. [a-zA-Z]{6}[0-9]{2}[a-zA-Z][0-9]{2}[a-zA-Z][0-9]{3}[a-zA-Z] **[propertyformatoFCFok]** 2. Non rispetta il formato **[error]**
Riscontro [R]	1. Trovato **[if propertyformatoFCFok] [propertyriscontroRok]** 2. Non trovato **[if propertyformatoFCFok] [error]**

TP – Test Plan

Parametro	Nome
Lunghezza [LN]	1. <2 or >50 **[error]** 2. ≥2 and ≤50 **[propertylunghezzaLNok]**

Parametro	Cognome
Lunghezza [LC]	1. <2 or >50 **[error]** 2. ≥2 and ≤50 **[propertylunghezzaLCok]**

Parametro	Password
Formato [FP]	1. ^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{6,}$ **[propertyformatoFPok]** 2. Non rispetta il formato **[error]**

Parametro	Data di nascita
Valore [DDN]	1. Valida and Dista piu di 17 anni dalla data corrente or dista meno di 130 anni dalla data corrente. **[propertyvaloreDDNok]** 2. Non valida or Dista meno di 17 anni dalla data corrente and dista più di 130 anni dalla data corrente **[error]**

Parametro	Indirizzo Email
Formato [FIE]	1. ^[a-zA-Z0-9][.!#$%&'*+=?^_`~a-zA-Z0-9-]{1,99}@[a-zA-Z0-9]{1,46}.[a-zA-Z]{2,5}$ **[propertyformatoFIEok]** 2. Non rispetta il formato **[error]**

Parametro	Numero di Telefono
Formato [FNF]	**1.** [0-9]{10}*$ **[propertyformatoFNFok]** 2. Non rispetta il formato **[error]**

Codice	Combinazione	Esito
TC_GU_2:1	FCF2	Negativo
TC_GU_2:2	FCF1.R1	Negativo
TC_GU_2:3	FCF1.R2.LN1	Negativo
TC_GU_2:4	FCF1.R2.LN2.LC1	Negativo
TC_GU_2:5	FCF1.R2.LN2.LC2.FP2	Negativo
TC_GU_2:6	FCF1.R2.LN2.LC2.FP1.DDN2	Negativo
TC_GU_2:7	FCF1.R2.LN2.LC2.FP1.DDN1.FIE2	Negativo
TC_GU_2:8	FCF1.R2.LN2.LC2.FP1.DDN1.FIE1.FNF2	Negativo
TC_GU_2:9	FCF1.R2.LN2.LC2.FP1.DDN1.FIE1.FNF1	Positivo

TP – Test Plan

9.2 Prenotazione

9.2.1 Richiesta Prenotazione

Per il sottosistema prenotazione abbiamo previsto di testare la funzionalità che permette di richiedere una prenotazione presso un ufficio ospedaliero convenzionato.

Tale funzionalità prevede la possibilità per un utente loggato di richiedere una prenotazione compilando un form in cui seleziona la struttura ospedaliera, una data, un'ora ed un tipo di operazione.

Parametro	Struttura Ospedaliera
Selezione [SO]	1. Selezionata **[propertyselezioneSOok]** 2. Non Selezionata **[error]**

Parametro	Data
Selezione [SD]	1. Selezionata **[propertyselezioneSDok]** 2. Non Selezionata **[error]**

Parametro	Ora
Selezione [ST]	1. Selezionata **[propertyselezioneSTok]** 2. Non Selezionata **[error]**

Parametro	Operazione
Selezione [SOP]	1. Selezionata **[propertyselezioneSOPok]** 2. Non Selezionata **[error]**

Codice	Combinazione	Esito
TC_GP_1:1	SO2	Negativo
TC_GP_1:2	SO1.SD2	Negativo
TC_GP_1:3	SO1.SD1.ST2	Negativo
TC_GP_1:4	SO1.SD1.ST1.SOP2	Negativo
TC_GP_1:5	SO1.SD1.ST1.SOP1	Positivo

9.2.2 Eliminazione Prenotazione

Per il sottosistema prenotazione abbiamo previsto di testare la funzionalità che permette di poter eliminare una prenotazione effettuata dall'utente.

Tale funzionalità prevede la possibilità per un utente loggato di selezionare una prenotazione dalla propria lista di prenotazioni ed eliminarla

Parametro	Prenotazione
Conferma [CP]	**1.** Confermata **[propertyconfermaCPok]** 2. Non Confermata **[error]**

Codice	Combinazione	Esito
TC_GP_2:1	SP2	Negativo
TC_GP_2:2	SP1	Positivo

9.2.3 Validazione prenotazione

Per il sottosistema prenotazione abbiamo previsto di testare la funzionalità che permette ad un utente recatosi nella struttura ospedaliera di convalidare la propria prenotazione

Tale funzionalità prevede la possibilità per un utente recatosi nella struttura ospedaliera di convalidare la propria prenotazione inserendo:

- Codice Fiscale: Stringa del formato “MNDCMN97R22A509S”

Parametro	Codice Fiscale
Formato [FCF]	1. [a-zA-Z]{6}[0-9]{2}[a-zA-Z][0-9]{2}[a-zA-Z][0-9]{3}[a-zA-Z] **[propertyformatoFCFok]** 2. Non rispetta il formato **[error]**
Riscontro [R]	1. Trovato **[if propertyformatoFCFok] [propertyriscontroRok]** 2. Non trovato **[if propertyformatoFCFok] [error]**

Codice	Combinazione	Esito
TC_GP_3:1	FCF2	Negativo
TC_GP_3:2	FCF1.R2	Negativo
TC_GP_3:3	FCF1.R1	Negativo

9.3 Gestione

9.3.1 Accettazione Prenotazione

Per il sottosistema gestione abbiamo previsto di testare la funzionalità che permette all’impiegato di accettare e servire le prenotazioni degli utenti

Tale funzionalità prevede la possibilità per un impiegato di accettare e servire le prenotazioni degli utenti per una determinata operazione.

Sara possibile accettare le prenotazioni presso la propria struttura selezionando la coda di prenotazioni che si vuole servire e cliccando sul bottone accetta prenotazione.

Parametro	Id Operazione
Selezione [SO]	1. Coda Selezionata **[propertyselezioneSOok]** 2. Coda non Selezionata **[error]**

Parametro	Id Struttura
Accettata [AS]	**1.** Prenotazione Accettate **[propertyaccettaASok]** 2. Prenotazione non accettate **[error]**

Codice	Combinazione	Esito
TC_GPI_1:1	SO2	Negativo
TC_GPI_1:2	SO1.AS2	Negativo
TC_GPI_1:3	SO1.AS1	Positivo

9.4 Visualizzazione Coda

9.4.1 Visualizzazione coda prenotazioni

Per il sottosistema visualizzazione coda abbiamo previsto di testare la funzionalità che permette all'utente di visualizzare la coda di prenotazioni presso una struttura ospedaliera

Sara possibile visualizzare la coda di prenotazioni selezionando una struttura ospedaliera e scegliendo la data

Parametro	Struttura Ospedaliera
Selezione [S]	**1.** Selezionata **[propertyselezioneSok]** 2. Non Selezionata **[error]**

Parametro	Data
Selezione [SD]	1. Data Selezionata **[propertyselezioneSDok]** 2. Data non selezionata **[error]**

Codice	Combinazione	Esito
TC_VC_1:1	S2	Negativo
TC_VC_1:2	S1.SD2	Negativo
TC_VC_1:3	S1.SD1	Positivo

10. Riferimenti ad altri documenti di test

Le combinazioni di input che verranno somministrate al sistema sono definite nel documento di Test Case Specification, mentre sarà nel Test Execution Report che indicheremo i risultati del testing funzionale.

Eventuali errori rilevati verranno riportati nel Test Incident Report, mentre il riassunto dei riscontri ottenuti in questa fase verrà proposto tramite il documento Test Summary Report.