UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

CORSO DI
INGEGNERIA, GESTIONE ED EVOLUZIONE DEL SOFTWARE

Test Plan

HeartCare

DOCENTE

Prof. Andrea De Lucia

Università degli Studi di Salerno

AUTORI

Annamaria Basile

Matricola: 0522501844

Paolo Carmine Valletta

Matricola: 0522501828

Alessandro Zoccola

Matricola: 0522501804

Anno Accademico 2023-2024

Indice

CAPITOLO 1

Introduzione

1.1 Panoramica del Sistema HeartCare

HeartCare è una piattaforma che fornisce uno strumento di telemonitoraggio ai pazienti affetti da malattie cardiache, permettendo loro di tenere sotto osservazione i propri valori e di ricevere notifiche preventive in caso di possibili problemi grazie all'applicazione dell'Intelligenza Artificiale. Il paziente può, tramite le misurazioni, aggiornare il proprio Fascicolo Sanitario Elettronico (FSE), che è accessibile al medico.

HeartCare è stata sviluppata per offrire un supporto online alle problematiche delle persone cardiopatiche. La piattaforma consente al paziente di accedere al proprio FSE, di aggiungere note destinate al medico, di registrare un nuovo dispositivo di monitoraggio e di eseguire misurazioni. Per il medico, HeartCare offre l'accesso ai fascicoli di ogni suo paziente, la possibilità di inviare note destinate al paziente e di stabilire il giorno della prossima visita. Inoltre, un amministratore gestisce la presenza di medici e pazienti sulla piattaforma e le relazioni tra di loro.

Ogni paziente può assegnarsi autonomamente un caregiver inserendo la sua email, nome e cognome. Quest'ultimo è una persona esterna al sistema che riceverà comunicazioni tramite mail qualora una misurazione effettuata dal paziente avrà prodotto dei valori pericolosi.

CAPITOLO 2

Funzionalità da testare

Nel processo di implementazione delle 3 Change Request (CR), sono state identificate una serie di funzionalità chiave che costituiscono il nucleo delle nuove modifiche apportate al sistema. Tuttavia, a causa della natura specifica di ciascuna CR, è necessario adottare approcci diversificati per il testing.

In particolare, per la CR1, che riguarda la migrazione dei dati, la fase di test riguarderà la verifica del corretto funzionamento del livello di archiviazione dei dati.

Invece, per quanto riguarda la CR2, verranno realizzati i test di unità dei vari moduli e successivamente andremo a verificare la corretta interazione tra i moduli mediante il testing di integrazione, che andrà anche a completare il processo di testing della CR1.

Infine, per la CR3, che riguarda l'interfaccia del sistema, ci concentreremo esclusivamente sui test di sistema, in quanto l'aspetto principale da valutare è l'interazione tra gli utenti e il sistema stesso.

Infine, per quanto riguarda la CR2, abbiamo sviluppato una serie di funzionalità specifiche da sottoporre a test, che includono:

- **Registrazione Caregiver**: Verificare che il caregiver possa registrarsi correttamente al sistema utilizzando il link ricevuto via email.

- **Login Caregiver**: Verificare che il caregiver possa effettuare correttamente il login tramite la pagina di login dedicata.
- **Assegnamento Caregiver**: Verificare che il caregiver venga assegnato correttamente a un paziente.
- **Invio nota da parte del Caregiver**: Verificare che il caregiver possa inviare correttamente una nota al medico del paziente.

Durante il processo di testing, ci si concentrerà sulla verifica della corretta implementazione di queste funzionalità.

CAPITOLO 3

Criteri di adeguatezza

Un test viene considerato **passato** quando il comportamento ottenuto è uguale al comportamento atteso dall'oracolo. Nel caso in cui il comportamento ottenuto sia diverso dall'oracolo allora il test viene considerato **fallito**, ciò significa che è presente una **failure** nel sistema. Saranno testati tutti i requisiti indicati nel Capitolo 2.

CAPITOLO 4

Strategie

Al fine di soddisfare i criteri di adeguatezza del testing, in questo capitolo definiamo delle opportune strategie di testing ed i tool che verranno utilizzati. Verranno effettuati diversi livelli di testing, in particolare testing a livello di unità, di integrazione, di sistema, testing di regressione e pratiche di ispezione del codice per individuare possibili errori introdotti.

4.1 Testing di Unità

Per il testing di **unità**, considerato che **Java** è il linguaggio di programmazione scelto, per questa fase sarà usato il framework ***JUnit***. La metodologia scelta per la progettazione dei casi di test è **black-box** (o testing funzionale) con l'utilizzo di category partition per ridurre l'elevato numero di combinazioni possibili che avremmo con una suddivisione in classi di equivalenza semplice.

4.2 Testing di Integrazione

Il testing di integrazione ha l'obiettivo di verificare l'interazione tra i vari moduli del sistema, assicurando che lavorino correttamente quando combinati. La metodologia di testing

usata per questa fase è il **bottom-up**. Il testing **bottom-up** è un approccio che inizia con il test delle unità più basse della gerarchia del software, solitamente i moduli o i componenti di base. Questi moduli, dopo essere stati testati individualmente dal testing di unità per verificarne la correttezza, vengono combinati per formare unità più grandi, che a loro volta vengono testate. Questo processo continua fino a quando l'intero sistema non è stato integrato e testato.

4.3 Testing di Sistema

Per la fase di testing di sistema, l'obiettivo è garantire l'integrità del comportamento del sistema implementato rispetto alle aspettative dell'utente. In questa fase sarà utilizzata una metodologia black-box. Per garantire la completezza della test suite, utilizzeremo la strategia Category Partition. Sarà utilizzato il tool ***Selenium*** che permette di registrare una sequenza di azioni effettuate sul browser client, in modo da rendere i casi di test ripetibili. Il testing di sistema verrà implementato su quelle che sono state le funzionalità elencate nel Capitolo 2.

4.4 Testing di Regressione

L'approccio di testing di regressione sarà basato sui test di unità e di integrazione che saranno sviluppati sul sistema originale. In questo modo, si garantirà che le modifiche apportate al sistema o alle sue componenti non introdurranno nuovi bug o errori. Di conseguenza, ad ogni push nel branch main tramite una GitHub Action, verranno eseguiti i test di unità e di integrazione per assicurare che il sistema continui a funzionare correttamente dopo ogni modifica.