**Università degli Studi di Salerno**

**Corso di Ingegneria del Software**

# CheckUp –
# TEST PLAN
# V1.1


Data 15/02/2021

Partecipanti:

| *Cognome e Nome* | *Matricola* |
|---|---|
| **Tamburello Martina** | 0512105452 |

Revision History:

| *Data* | *Versione* | *Cambiamenti* | *Autori* |
|---|---|---|---|
| 15/01/2021 | 1.0 | Stesura documento | Tamburello Martina |
| 17/01/2021 | 1.1 | Aggiunta descrizione casi di test | Tamburello Martina |

Sommario

# 1. Introduzione

## 1.1 Scopo del sistema

In un momento così particolare della nostra storia, in cui, a causa dell'eccessiva pressione sul sistema sanitario, l'accesso elle strutture sanitarie è notevolmente ridotto, il sistema si propone di agevolare l'interazione a distanza tra medico e paziente.
Un paziente, in cura presso una specialista per una qualsiasi patologia, è costretto periodicamente ad effettuare particolari esami.
Nella situazione attuale il paziente è impossibilitato nel recarsi presso il proprio medico per il controllo di routine degli esami, quindi, paziente e medico sono costretti a comunicare via e-mail.
Il sistema si propone, quindi, di fornire una piattaforma in cui il paziente può caricare i risultati dei propri esami e il medico curante può accedere e visionare tali referti.

## 1.2 Ambito del sistema

Il progetto nasce dall'esigenza di ridurre, quando possibile, i contatti ravvicinati tra medico e paziente, agevolando i contatti a distanza.
Le principali operazioni e funzionalità offerte dal sistema sono le seguenti:
Caricamento referto da parte di un paziente con specifica del medico destinatario;
Prenotazione visita specialistica con eventuale disdetta;

Il sistema non supporta:
Lo spostamento delle prenotazioni effettuate in un giorno in cui il relativo medico ha avvisato di disdire le prenotazioni.

## 1.3 Obiettivi e criteri di successo del progetto

Gli obiettivi principali del progetto "CheckUp" sono:
Facilitare l'interazione a distanza tra paziente e medico;
Velocizzare l'effettuazione delle prenotazioni;
Rendere visibili al paziente tutti i suoi referti e le diagnosi;
Rendere disponibile attraverso poche operazioni la cartella clinica di un paziente.

## 1.4 Definizioni, acronimi e abbreviazioni

Definizioni

- Branch Coverage: tecnica adoperata durante la fase di testing, che prevede l'esecuzione di tutti i rami del programma almeno una volta durante la fase di testing.
- Failure: mancata o scorretta azione di un determinato servizio atteso.
- Fault: causa che ha generato una failure.
- Model View Control: è un metodo architetturale che prevede la divisione dell'applicazione in tre parti. Tale divisione viene effettuata per separate la rappresentazione delle informazioni interne del sistema dal meccanismo in cui le informazioni sono presentate all'utente

Acronimi
RAD = Requirements Analysis Document (Documento di Raccolta dei Requisiti),
SDD = System Design Document
ODD=Object Design Document
TP = Test Plan
MVC=Model View Controller.

DB=Database.
GUI=Graphical User Interface.

## 1.5 Riferimenti

Libro:
-- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition

Autori:
-- Bernd Bruegge & Allen H. Dutoit

Documenti:
- CheckUp_RAD_V1.5.docx – Requirements Analysis Document
- CheckUp_SDD_V1.3.docx – System Design Document
- CheckUp_ODD_V1.2.docx – Object Design Document

## 1.6 Panoramica

Il seguente documento è diviso in sezioni ed ha la seguente composizione:
Sezione di INTRODUZIONE:
Breve descrizione delle esigenze da cui parte l'idea del progetto, viene quindi fornito e descritto il contesto di utilizzo del sistema per poi passare definire gli obiettivi del sistema e i punti di forza/criteri di successo dell'intero progetto. Successivamente vengono citati definizioni, acronimi e abbreviazioni usati per facilitare il lettore a ricordare le parole più usate (con acronimi o abbreviazione) e/o nel capire parole del gergo tecnico (con le corrispettive definizioni). In seguito, i riferimenti utilizzati come linee guida per lo sviluppo dell'intero progetto dal punto di vista ingegneristico.

Sezione RELAZIONE CON ALTRI DOCUMENTI:
Sezione dedicata alla spiegazione del collegamento tra questo e gli altri documenti.

Sezione PANORAMICA:
Rapida ricapitolazione della suddivisione in sottosistemi.

Sezione dedicata alle FUNZIONALITA' DA TESTARE/NON TESTARE:
Specifica quali funzionalità tra quelle implementate saranno testate e quali no.

Sezione CRITERI PASS/FAIL:
Chiarisce i concetti di "test passato" e "test fallito".

Sezione APPROCCIO:
Illustra l'approccio utilizzato durante la fase di testing.

Sezione CRITERI SOSPENSIONE/RIPRESA:
Specifica a quale del punto dell'attività di testing ci si fermerà e quando essa, eventualmente, riprenderà.

Sezione MATERIALI:
Descrive tool e materiali necessari alla fase di testing.

Sezione CASI DI TEST:
Breve descrizione dei casi di test che verranno effettuati.

Sezione PROGRAMMA DEI TEST:
*Tale sezione c*opre le responsabilità, il personale e le esigenze di formazione.

# 2. Relazione con altri documenti

Questo documento è correlato a tutti i documenti prodotti fino al rilascio del sistema.
I test case sono basati sulle funzionalità del sistema, individuate e raccolte nel RAD.

## 2.1 Relazione con il RAD

La relazione tra questo documento e il RAD riguarda i requisiti funzionali in esso definiti.
A partire da essi e, tramite l'ausilio di casi d'uso, scenari e sequence si otterranno i casi di test.

## 2.2 Relazione con l'SDD

La relazione tra questo documento e l'SDD riguarda i servizi offerti dai sottosistemi. Infatti, i test case sono predisposti in base alla suddivisione del sistema in sottosistemi in esso definita.

## 2.3 Relazione con l'ODD

La relazione tra questo documento e l'ODD riguarda il fatto che è in quest'ultimo che sono definiti i package e classi di sistema.

# 3. Panoramica del sistema

Come definito nel System Design Document, il sistema segue il pattern MVC: il controller si occupa principalmente di dirigere il flusso di controllo; il model contiene le entità che hanno un collegamento persistente nel DB e le classi e le interfacce per gestire tali entità; infine la view comprende le interfacce per le diverse tipologie di utente.
Il livello view è stato decomposto in:

- **GUI Medico**
- **GUI Paziente**
- **GUI Segretario**

Il livello control è stato decomposto in:

- **Profile:**
  Contiene le servlet per effettuare le operazioni di login, logout, modifica password;
- **Booking:**
  Contiene le servlet per gestire le richieste riguardanti le prenotazioni;
- **Util:**
  Contiene classi che fungono d'ausilio alle precedenti.

Il livello model è stato decomposto in:

- **Bean**
  Contiene le classi che modellano gli oggetti del dominio applicativo. Ciascuna classe definisce dei metodi get e set.
- **Dao**
  Contiene le interfacce dei Data Access Object (DAO), ovvero oggetti che si relazionano con il gestore della persistenza per effettuare operazioni di tipo CRUD (Create, Read, Update, Delete)
- **Database**
  Contiene le implementazioni delle interfacce definite nel package Dao e contiene una classe DbConnection attraverso la quale avviene la connessione al database.

# 4. Funzionalità da testare/non testare

Le attività di testing previste per il sistema CheckUp prevedono di testare il corretto funzionamento di tutte le funzionalità implementate, dunque di tutti e soli i requisiti ad ALTA priorità.
Saranno quindi testate le seguenti funzionalità:

- Gestione Profilo
  -Login paziente
  -Login dipendente
  -Logout
  -Modifica password
- Gestione Prenotazioni
  -Effettua prenotazione
  -Disdetta prenotazione
  -Visualizzazione lista prenotazioni (per il paziente)

Funzionalità da non testare:

Le funzionalità che non verranno testate sono quelle relative a tutti i requisiti funzionali a bassa e media priorità.

# 5. Criteri di Pass/Fail

Dopo aver individuato tutti i dati di input del sistema, essi verranno raggruppati insieme in base alle caratteristiche in comune. Questa tecnica servirà per poter diminuire il numero di test da dover effettuare. La fase di test avrà successo se verrà individuata effettivamente una failure all'interno del sistema, cioè l'output atteso per quel determinato input non è lo stesso previsto dall'oracolo. Successivamente la failure sarà analizzata e si passerà eventualmente alla sua correzione e verranno eseguiti nuovamente tutti i test necessari per verificare l'impatto che la modifica ha avuto sull'intero sistema. Diremo invece che il testing fallirà se l'output mostrato dal sistema coincide con quello previsto dall'oracolo.

# 6. Approccio

## 6.1 Testing di unità

Lo scopo del testing di unità è quello di testare le funzioni del sistema in isolamento. Ogni funzione rappresenta quella che viene definita una componente.

### 6.1.1 Approccio scelto

L'approccio scelto è di tipo white-box. Con tale tipo di approccio si testano le componenti conoscendone il funzionamento interno. Per questa fase verrà utilizzato il tool *JUnit*.
Il criterio di accettazione scelto per considerare il testing "superato con successo" è di 80% branch coverage.

## 6.2 Testing di integrazione

Lo scopo del testing di integrazione è quello di unire le componenti testate precedentemente tramite il test di unità per testarne l'interazione.

### 6.2.1 Approccio scelto

Per effettuare il testing di integrazione si è scelto di adoperare un approccio bottom-up. Il vantaggio fondamentale di questa tipologia di testing è quello della riusabilità del codice. Questo tipo di approccio prevede però la costruzione driver per simulare l'ambiente chiamante. È stato scelto quindi questo tipo di approccio perché sembra quello più intuitivo e semplice.

## 6.3 Testing di sistema

Lo scopo del testing di sistema è quello di verificare che i requisiti richiesti dal cliente siano stati

effettivamente compresi e rispettati. In questo tipo di testing si vanno a verificare le funzionalità utilizzate più spesso da parte del cliente e quelle che risultano più "critiche".

### 6.3.1 Approccio scelto

Per la specifica dei test case è stata adoperata la tecnica del Category Partion.

Il tool scelto per il testing di sistema è *Selenium* che si occuperà di simulare le interazioni con il sistema stesso come se le stesse svolgendo l'utente.

# 7. Criteri di sospensione/ripresa

*Criteri di sospensione*

La fase di testing verrà sospesa nel momento in cui saranno raggiunti i risultati previsti in accordo con quello che è il budget a disposizione.

*Criteri di ripresa*

Tutte le attività di testing riprenderanno nel momento in cui verranno effettuate modifiche all'interno del sistema.

# 8. Materiale per effettuare il testing

Per la definizione dei casi di test e lo svolgimento dell'attività di testing sono necessari tutti i documenti fino ad ora stilati:

- CheckUp_RAD_V1.5 - per il testing di sistema
- CheckUp_SDD_V1.3 – per il testing di integrazione
- CheckUp_ODD_V1.1 – per il testing di unità

Per il testing di unità e di integrazione verrà usato il tool *JUnit* e per quello di sistema il tool *Selenium*.

# 9. Casi di Test

Questa sezione elenca i casi di test utilizzati durante il test. Ciascun caso di test sarà descritto in dettaglio in un documento separato (TCD).

Ogni esecuzione di questi test che avrà successo (si veda il criterio di successo definito nella sezione 5 di questo documento), sarà documentata in un documento Report Incident Test.

Elenco dei casi di test:

*TC_2 Login paziente:*
Tale caso di test testa la funzionalità di autenticazione per il paziente.
Esso soddisfa il requisito funzionale numero 2 (si veda il documento CheckUp_RAD_V1.5).
Il sistema deve verificare che le credenziali inserite non siano nulle e, in seguito, che nel database sia contenuta la coppia (codice fiscale, password).
Non sono previsti altri controlli per questo caso di test.

*TC_2bis:*
Tale caso di test testa la funzionalità di autenticazione per il dipendente.
Esso soddisfa il requisito funzionale numero 2 (si veda il documento CheckUp_RAD_V1.5).
Il sistema deve verificare che le credenziali inserite non siano nulle e, in seguito, che nel database sia contenuta la coppia (email, password).

Non sono previsti altri controlli per questo caso di test.
*TC_3 Modifica password:*
Tale caso di test testa la funzionalità di modifica password per l'utente generico.
Esso soddisfa il requisito funzionale numero 3 (si veda il documento CheckUp_RAD_V1.5).
Il sistema deve verificare che:

- Il campo nuova password corrisponda al campo conferma password;
- Il campo nuova password rispetti il formato previsto e che il suo valore non corrisponda a quello dell'attuale password;
- Il campo password corrisponda all'attuale password utente.

*TC_9 Nuova prenotazione:*
Tale caso di test testa la funzionalità di modifica password per l'utente generico.
Esso soddisfa il requisito funzionale numero 9 (si veda il documento CheckUp_RAD_V1.5).
Il sistema deve verificare che:

- I campi specializzazione, medico, data e ora non siano vuoti;
- Esista, per il medico selezionato almeno una data e un orario disponibile.

# 10. Programma dei Test

## 10.1 Responsabilità

Il team comprende un unico membro che assumerà i ruoli di sviluppatore e tester e avrà tutte le responsabilità ad essi relative.

## 10.2 Esigenze di formazione

Il team non conosce attualmente *Selenium*. Sarà effettuata attività di training prima di passare all'attività di testing di sistema.

# 11. Glossario

- Form: termine utilizzato per riferirsi all'interfaccia di una applicazione che consente all'utente di inserire e inviare dati digitali.
- Testing: procedimento utilizzato per individuare le carenze di correttezza, completezza e affidabilità dei componenti software nel corso dello sviluppo.
- Tool: strumento software utilizzato per ottenere un determinato risultato.