Test Summary Report

GreenBridge

Riferimento	
Versione	1.0
Data	02/02/2024
Destinatario	Filomena Ferrucci
Presentato da	Mauro Pasquariello Michele Martino Giuseppe Di Sarno Salvatore Mattiello Giovanni De Gregorio Davide Califano El Mehdi Zitouni
Approvato da	Daniele Donia

Revision History

Data	Versione	Descrizione	Autori
31/1/2024	0.1	Prima stesura	Davide Califano
02/02/2024	1.0	Revisione Finale	Davide Califano

Sommario

1. Introduzione

Il Test Summary Report è un documento contenente i risultati dell'esecuzione dei test case di unità delle varie componenti dell'applicazione GreenBridge. Le classi dei package model e application sono testate con l'utilizzo di JUnit e Mockito; successivamente, viene effettuato il testing di sistema con Selenium.

Lo scopo di questo documento è quello di fornire una presentazione dei casi di testing di unità per GreenBridge. Il team ha sviluppato e testato queste classi in modo da garantirne il corretto funzionamento.

2. Relazione con altri documenti

Di seguito vengono elencate le relazioni tra il presente documento e gli altri documenti di testing.

Relazione con il Test Plan

Il Test Summary Report fa riferimento alle attività di testing specificate nel Test Plan.

Relazione con il Test Case Specification

Il Test Summary Report contiene il sunto dell'esecuzione dei test di sistema specificati nel Test Case Specification.

Relazione con il Test Incident Report

I Test Summary Report contiene il sunto dei risultati sull'esecuzione specificati nel Test Incident Report. I risultati sono stati generati attraverso il tool Maven Surefire:

Summary

[Summary] [Package List] [Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
72	1	2	0	95.8%	254.8 s

Package List

[Summary] [Package List] [Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
com.greenbridge.system	37	1	1	0	94.6%	250.9 s
com.greenbridge.unit	35	0	1	0	97.1%	3.927 s

3. Testing Unitario

La fase di testing di unità prevede di testare in maniera isolata le diverse componenti del sistema realizzato. Per fare questo, si è deciso di testare i metodi delle diverse classi che compongono i vari sottosistemi.

Ogni team member, prima di fare un push, doveva assicurarsi che tutti i test scritti riguardanti le classi da lui modificate avessero successo.

Esecuzione	# Fallimenti	# Successi
31/1/2024	0	35

4. Testing di sistema

Per ciò che concerne il test di sistema si sono andate a definire varie test suites tramite il tool Selenium IDE, per Chrome. Nello specifico, si è generata una Test Suite basata sui test descritti nel Test Plan. Di seguito vengono riportati i risultati delle esecuzioni dei tests.

Esecuzione	# Fallimenti	# Successi
31/1/2024	1	36
31/1/2024	0	37

5. Coverage ottenuta

Nel corso e a fine progetto è stato usato il tool JaCoCo per la raccolta di metriche sulla coverage del test. Di seguito vengono riportati i risultati della coverage dei test.

Greenbridge

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.greenbridge.controllers		41%		36%	77	129	252	438	30	71	0	16
com.greenbridge.entities		55%		35%	97	212	124	267	69	182	0	11
com.greenbridge.services		40%		10%	28	50	52	95	23	45	0	9
com.greenbridge		37%		n/a	1	2	2	3	1	2	0	1
com.greenbridge.configurations		100%		n/a	0	2	0	2	0	2	0	1
Total	1,674 of 3,155	46%	122 of 186	34%	203	395	430	805	123	302	0	38

6. Glossario

JUnit, un framework per Java utilizzato al fine di automatizzare il testing;

Mockito, un framework per Java utilizzato per emulare delle componenti di una classe al fine di eseguire il testing;

Selenium, un tool che permette di registrare le azioni da effettuare al fine di automatizzare il testing di sistema;

JaCoCo, è una libreria per il code coverage in ambito Java