Laurea Magistrale in informatica-Università di Salerno
Corso di *Gestione dei Progetti Software*- Prof.ssa F.Ferrucci

GUARDIAN FLOW

FEELING SAFE

Test Plan

Guardian Flow

Riferimento	
Versione	0.9
Data	13/12/2023
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Intero team
Approvato da	Raffaele Mezza, Martina Mingione

Sommario

Revision History 3
1. Introduzione 4
2. Relazione con altri documenti 4
3. Panoramica del sistema 5
4. Funzionalità da testare/da non testare 5
4.1 Unit Test 5
4.2 Integration Test 6
4.3 System Test 6
5. Criteri Pass/Fail 7
6. Approccio 7
7. Sospensione e ripristino 9
8. Test Deliverables 10
9. Materiale per il testing 10
10. Test case di sistema 11
10.1 Gestione Accesso 11
10.1.1 Login utente 11
10.1.2 Modifica Password 11
10.1.3 Attivazione 2FA 12
10.2 Funzionalità utente 12
10.2.1 Modifica Piano 12
10.2.2 Gestione Utenti 13
10.3 Funzionalità del sistema 15
10.3.1 Notifica di un'anomalia 15

Revision History

Data	Versione	Descrizione	Autori
4/12/2023	0.1	Prima stesura	Raffaele Mezza Martina Mingione
7/12/2023	0.2	Stesura introduzione	Giuseppe Cerella
7/12/2023	0.3	Stesura panoramica del sistema	Edmondo De Simone
7/12/2023	0.4	Stesura funzionalità e relazioni con altri documenti	Danilo Gisolfi
7/12/2023	0.5	Stesura pass/fail criteria	Mattia Guariglia
7/12/2023	0.6	Stesura approccio, sospensione e ripristino	Vincenzo Maiellaro
7/12/2023	0.7	Funzionalità da testare e non	Vincenzo Maiellaro
7/12/2023	0.8	Individuazione Test Case di sistema	Tutto il team
7/12/2023	0.9	Stesura approccio e materiali per il testing	Tommaso Nardi
13/12/2023	0.9	Revisione del documento	Danilo Gisolfi
13/12/2023	0.9	Approvazione del documento	Martina Mingione Raffaele Mezza

Testing Plan

1. Introduzione

Guardian Flow è il risultato della necessità di creare una solida struttura digitale per analizzare il traffico di rete di un'azienda alla ricerca di eventuali anomalie. Tra gli obiettivi vi è quello di migliorare il processo esistente, il quale ha evidenziato problematiche legate a una rilevazione inefficace delle anomalie nel traffico di rete, spingendoci ad adottare un approccio più avanzato e scalabile. Guardian Flow costruisce una baseline personalizzata per ogni cliente e mette a disposizione una dashboard che consente ai clienti di monitorare in tempo reale il traffico di rete analizzato e le eventuali anomalie individuate. Inoltre, il sistema offre la possibilità agli utenti di modificare agevolmente il piano di abbonamento, garantendo loro la libertà di adattare il sistema e le risorse disponibili alle specifiche esigenze aziendali. L'obiettivo di questo documento è verificare il comportamento di ciò che sarà il sistema implementato, attraverso le fasi di test case specification, execution e report per i tre livelli di test: unità, in. Sarà fondamentale individuare errori e correggere le relative cause anche per fornire al cliente un prodotto il più affidabile possibile.

2. Relazione con altri documenti

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

Statement Of Work (SOW)

All'interno di questo documento, è stato definito un vincolo per quanto riguarda il branch coverage che deve essere uguale o superiore al 75%.

Requirements Analysis Document (RAD)

I test case pianificati nel Test Plan sono individuati a partire dalla specifica dei casi d'uso e dei requisiti individuati nella fase di analisi.

System Design Document (SDD)

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

3. Panoramica del sistema

Il sistema proposto basa la sua architettura sul sistema three-tier:

- Verranno usati Nuxt.js e tailwind.css per la parte di front-end;
- Per la logica applicativa e quindi il back-end sarà utilizzato Python;
- Per la gestione del database saranno usati:
    - NodeJs per il collegamento al database;
    - PostgreSQL per la gestione del database.

4. Funzionalità da testare/da non testare

Di seguito la lista delle funzionalità di cui si effettuerà il testing:

4.1 Unit Test

Con attenzione alle specifiche dell'architettura Three-Tier, l'approccio di testing a livello di unità sarà mirato a coprire i seguenti aspetti:

- La verifica di ciascun componente fondamentale per la logica di persistenza del sistema, inclusi quelli dedicati all'interazione con il database;
- La valutazione di ogni componente responsabile dell'implementazione della Business Logic del sistema, comprese le funzioni per l'elaborazione dei dati da salvare, modificare, aggregare o preparare per l'interfaccia utente, nonché le funzioni di supporto per compiti comuni come l'invio di notifiche e la gestione di anomalie.
- La valutazione delle unità di componenti di presentazione che si occuperanno di ricevere dati dai sottosistemi del Layer View e di rilasciare in output tramite meccanismi di risposta dati già elaborati dalle componenti di servizio.

4.2 Integration Test

Con l'avanzare dell'implementazione delle singole componenti del sistema, sarà necessario estendere il processo di testing all'integrazione tra le componenti critiche, le quali sono già state verificate a livello di unità. Si prevede di strutturare il testing di integrazione per garantire la corretta interazione tra:

- Il database e le componenti responsabili della gestione della logica di persistenza;
- Le componenti componenti sovrastanti di Business Logic.
- Le componenti del livello di presentazione.

4.3 System Test

Dato il carattere funzionale del testing di sistema, e la sua diretta correlazione con la GUI Utente, si ritiene opportuno strutturare il testing di sistema partendo dalle specifiche funzionali definite dal team di progetto. Sarà di particolare importanza concentrarsi sul testing funzionale a livello di sistema, soprattutto per i servizi che implicano un'interazione più intensa con l'utente.

Più nello specifico, si prevede di testare a livello di sistema i servizi che richiedono maggiori interazioni ed inserimento dati da parte degli utenti, in particolare:

- Per il sottosistema di Utente Management, si prevede critico il testing dei servizi di:
  - Login;
  - Modifica Password;
  - Attiva 2FA.
- Per il sottosistema di Gestione Utente Management sarà necessario testare a livello di sistema il servizio di:
  - Gestione Utenti.
- Per il sottosistema di Piano Management si testerà il servizio di:
  - Modifica Piano.
- Per il sottosistema di Dashboard Management si testerà il servizio di:
  - Notifica Anomalia.

Nella selezione dei servizi da testare sono stati coinvolti i Project Manager.

5. Criteri Pass/Fail

Le attività di testing hanno l'obiettivo di identificare la presenza di errori all'interno del sistema, per effettuare un eventuale intervento di correzione. L'esito di un test case è valutato con l'uso di un oracolo ovvero tramite il risultato atteso della sua esecuzione, basandosi sui requisiti definiti.

Il test è considerato superato (pass) se, dato un input al sistema, l'output ottenuto è uguale all'output previsto dall'oracolo.

Il test è considerato fallito (fail) se, dato un input al sistema, l'output ottenuto è diverso dall'output previsto dall'oracolo.

6. Approccio

Il testing dell'intero sistema si compone di tre fasi: testing di sistema, testing di integrazione e testing di unità.

6.1 Testing di unità

Per il testing di unità e i vincoli previsti, si prevede che il team sia avvantaggiato nel testing e nel rilevamento di problemi nelle singole componenti in quanto si ha a disposizione l'intero codice implementato ed è noto dall'inizio della stesura fino al deploy. Per questo i PM, ritengono opportuno preferire una metodologia di testing di tipo White Box, ed in particolare, al fine di monitorare per tutta l'attività di testing, il vincolo di 75% di Branch Coverage.

Si prevede che i casi di test di unità vengano estratti direttamente dal codice applicando correttamente il criterio di Analisi dei Branch, individuando specifici test case almeno per i Branch principali di ogni componente sotto test. In termini di documentazione, qui sarà necessario associare ad ogni singolo test case di unità almeno un commento nel codice che indichi una descrizione del comportamento atteso.

6.2 Testing di integrazione

Per il testing di integrazione, che può essere condotto utilizzando gli stessi strumenti definiti per il testing di unità, si pianifica di procedere mediante l'integrazione delle componenti seguendo la metodologia Bottom-Up. Questo significa integrare le componenti in base alle dipendenze già stabilite dall'architettura Three-Tier, specializzata attraverso opportune scelte di Object Design.

In dettaglio, si prevede di testare inizialmente l'integrazione delle componenti relative alla logica di persistenza con il database sottostante. Successivamente, si procederà all'integrazione delle componenti di servizio, le quali dipendono strettamente dalle componenti di persistenza già integrate con il database. È importante notare che la metodologia Bottom-Up potrebbe richiedere la simulazione delle componenti di livello superiore che dipendono dalle componenti attualmente in fase di test.

In previsione di questa eventualità, i PM raccomandano l'uso di Test Driver per simulare le componenti di livello superiore che non sono ancora state integrate.

6.3 Testing di sistema

Per la definizione dei casi di test di sistema, mirati a verificare l'integrità del comportamento del sistema implementato rispetto alle aspettative dell'utente, si tiene conto che la disponibilità del codice sorgente non è di importanza primaria. In questa fase, si prevede l'adozione di una metodologia black-box. In particolare, è cruciale bilanciare la completezza delle suite di test con il budget a disposizione del team. Pertanto, per il testing di sistema, è stata scelta la metodologia specifica Category Partition. Nonostante non sia la più completa in termini assoluti, questa metodologia offre comunque la possibilità di ottenere un buon livello di dettaglio considerando il tempo preventivato.

7. Sospensione e ripristino

Seguono 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 verrà sospeso se almeno il 10% dei risultati appare come successo con la rilevazione di almeno un fault. In caso di sospensione, il team si occuperà di risolvere tutti i fault prima di implementare nuove funzionalità.

Criteri di ripristino

Il testing riprenderà una volta risolti tutti i fault individuati e dopo aver controllato che non ce ne siano altri. Bisogna anche assicurarsi che le modifiche non abbiano loro stesse dai fault all'interno, i quali avranno impatto sulle funzionalità implementate; in questi casi si testeranno anche quest'ultime funzionalità tramite regressione.

Criteri di terminazione

L'attività di testing è terminata quando tutti i test cases avranno esito negativo:
Secondo il documento 'Statement of Work', è obbligatorio raggiungere una branch coverage pari al 75%, quindi almeno il 75% dei branch del sistema non devono avere fault.

8. Test Deliverables

I documenti prodotti durante la fase di testing saranno:

- Test Plan;
- Test Case System Specification;
- Test Case Integration Specification;
- Unit Test Report;
- Test Execution Report;
- Test Incident Report;
- Test Summary Report;

9. Materiale per il testing

Come detto, il testing richiederà il supporto dei documenti RAD, SDD e ODD in modo da individuare le componenti da testare. L'esecuzione dei test avverrà, ovviamente, sul sistema presente (tramite deploy) in un server sul quale è configurato PostgreSQL. Ovviamente il primo deploy dovrà essere eseguito seguendo le istruzioni fornite dal manuale di installazione in quanto è il deploy di configurazione.

Il testing di unità e il testing di integrazione dovranno essere svolti tramite i framework Vitest e unittest. Il testing di sistema, invece, dovrà essere svolto tramite Selenium, il quale automatizzerà le interazioni utente con le interfacce del sistema.

10. Test case di sistema

10.1 Gestione Accesso

10.1.1 Login utente

Parametro: Email	
Nome Categoria	**Scelte per la categoria**
Match [MA]	1. Rispetta il match = false [PROPERTY: ERROR] 2. Rispetta il match = true [PROPERTY MA_OK]
Parametro: Password	
Nome Categoria	**Scelte per la categoria**
Match [MP]	1. Rispetta il match = false [PROPERTY: ERROR] 2. Rispetta il match = true [PROPERTY MP_OK]

Test Case ID	Test Frame	Esito
TC_1.1_1	MA1	Errato: email non corretta
TC_1.1_2	MA2, MP1	Errato: email corretta ma password no
TC_1.1_3	MA2, MP2	Corretto

10.1.2 Modifica Password

Parametro: Password	
FORMATO: ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$	
Nome Categoria	**Scelte per la categoria**
Formato [FP]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FP_OK]
Parametro: Conferma Password	
Nome Categoria	**Scelte per la categoria**
Match [MP]	1. Rispetta il match = false [PROPERTY: ERROR] 2. Rispetta il match = true [PROPERTY MP_OK]

Test Case ID	Test Frame	Esito
TC_1.2_1	FP1	Errato: password non corretta
TC_1.2_2	FP2, MP1	Errato: conferma password errata
TC_1.2_3	FP2, MP2	Corretto

10.1.3 Attivazione 2FA

Parametro: Codice	
FORMATO: ^[0-9]{6}$	
Nome Categoria	**Scelte per la categoria**
Lunghezza [LC]	1. Lunghezza > 6 = false [error] 2. Lunghezza <= 6 = true [PROPERTY LC_OK]

Test Case ID	**Test Frame**	**Esito**
TC_1.3_1	LC1	Errato: codice non corretto
TC_1.3_2	LC2	Corretto

10.2 Funzionalità utente

10.2.1 Modifica Piano

Parametro: Numero Carta	
Nome Categoria	**Scelte per la categoria**
Lunghezza [LC]	1. Lunghezza > 16 = false [error] 2. Lunghezza <= 16 = true [PROPERTY LC_OK]
Parametro: Intestatario	
FORMATO: ^[A-Za-z]+(?:\s[A-Za-z]+)?$	
Nome Categoria	**Scelte per la categoria**
Formato [FI]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FNC_OK]
Parametro: Scadenza	
Nome Categoria	**Scelte per la categoria**
Correttezza [CF]	1. Scadenza <= DataOraCorrente = false [errore] 2. Scadenza > DataOraCorrente = true [PROPERTY CF_OK]
Parametro: CVV	
FORMATO: ^[0-9]{3}$	
Nome Categoria	**Scelte per la categoria**
Lunghezza [LCS]	1. Lunghezza > 3 = false [error] 2. Lunghezza <= 3 = true [PROPERTY LCS_OK]

Test Case ID	Test Frame	Esito
TC_2.1_1	LC1	Errato: numero della carta non corretto
TC_2.1_2	LC2, FI1	Errato: intestatario non corretto
TC_2.1_3	LC2, FI2, CF1	Errato: scadenza della carta non corretta
TC_2.1_4	LC2, FI2, CF2, LCS1	Errato: codice di sicurezza non corretto
TC_2.1_5	LC2, FI2, CS2, LCS2	Corretto

10.2.2 Gestione Utenti

Parametro: Email	
FORMATO: ^[A-z0-9._%+-]+@[A-z0-9.-]+\.[A-z]{2,10}$	
Nome Categoria	**Scelte per la categoria**
Formato [FE]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FE_OK]
Parametro: Password	
FORMATO: ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$	
Nome Categoria	**Scelte per la categoria**
Formato [FP]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FP_OK]
Parametro: Nome	
FORMATO: ^[A-Z][-a-zA-Z]+$	
Nome Categoria	**Scelte per la categoria**
Formato [FN]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FN_OK]
Parametro: Cognome	
FORMATO: ^[A-Z][-a-zA-Z]+$	
Nome Categoria	**Scelte per la categoria**
Formato [FC]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FC_OK]
Parametro: Permessi	
FORMATO: ^(Amministratore | Moderatore)$	
Nome Categoria	**Scelte per la categoria**
Formato [FPEA	1. Rispetta il formato = false [PROPERTY FCA_OK] 1. Rispetta il formato = true [PROPERTY FCA_OK]

Test Case ID	Test Frame	Esito
TC_2.2_1	FA1	Errato: email non corretta
TC_2.2_2	FA2, FP1	Errato: nome non corretto
TC_2.2_3	FA2, FP2, FN1	Errato: cognome non corretto
TC_2.2_4	FA2, FP2, FN2, FC1	Errato: password non corretta
TC_2.2_5	FA2, FP2, FN2, FC2, FPE1	Errato: permesso non corretto
TC_2.2_6	FA2, FP2, FN2, FC2, FPE2	Corretto

10.3 Funzionalità del sistema

10.3.1 Notifica di un'anomalia

Parametro: Protocollo	
FORMATO: ^(TCP | UDP)$	
Nome Categoria	**Scelte per la categoria**
Formato [FPT]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FTP_OK]
Parametro: Stato	
FORMATO: ^(aperto | chiuso)$	
Nome Categoria	**Scelte per la categoria**
Formato [FS]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FS_OK]
Parametro: Data	
FORMATO: ^\d{4}-\d{2}-\d{2}$	
Nome Categoria	**Scelte per la categoria**
Formato [FD]	1. Rispetta il formato = false [PROPERTY FD_OK] 2. Rispetta il formato = true [PROPERTY FD_OK]
Parametro: Porta	
Nome Categoria	**Scelte per la categoria**
Lunghezza [LP]	1. Lunghezza > 5 = false [error] 2. Lunghezza <= 5 = true [PROPERTY LP_OK]
Parametro: IP_Sorgente	
FORMATO: ^((25[0-5] | 2[0-4][0-9] | [01]?[0-9][0-9]?)\.){3}(25[0-5] | 2[0-4][0-9] | [01]?[0-9][0-9]?)$	
Nome Categoria	**Scelte per la categoria**
Formato [FIS]	1. Rispetta il match = false [PROPERTY: ERROR] 2. Rispetta il match = true [PROPERTY FIS_OK]
Parametro: IP_Destinatario	
FORMATO: ^((25[0-5] | 2[0-4][0-9] | [01]?[0-9][0-9]?)\.){3}(25[0-5] | 2[0-4][0-9] | [01]?[0-9][0-9]?)$	
Nome Categoria	**Scelte per la categoria**
Formato [FID]	1. Rispetta il formato = false [PROPERTY: ERROR] 2. Rispetta il formato = true [PROPERTY FID_OK]

Test Case ID	Test Frame	Esito
TC_3.1_1	FPT1	Errato: protocollo non corretto
TC_3.1_2	FPT2, FS1	Errato: stato non corretto
TC_3.1_3	FPT2, FS2, FD1	Errato: data non corretta
TC_3.1_4	FPT2, FS2, FD2, LP1	Errato: porta non corretta
TC_3.1_5	FPT2, FS2, FD2, LP2, FIS1	Errato: indirizzo sorgente non corretto
TC_3.1_6	FPT2, FS2, FD2, LP2, FIS2, FID1	Errato: indirizzo destinatario non corretto
TC_3.1_7	FPT2, FS2, FD2, LP2, FIS2, FID2	Corretto