Università degli Studi di Salerno
Corso di Ingegneria del Software

SwaGGed
Test Plan
Versione 1.0

Data: 13/12/2024

Coordinatore del progetto

Nome	Matricola

Partecipanti:

Nome	Matricola
Choaib Goumri	0512118390
Mattia Gallucci	0512116893

Scritto da:	Choaib Goumri
	Mattia Gallucci

Revision History

Data	Versione	Descrizione	Autore
13/12/2024	1.0	Definizione del test plan	Choaib Goumri, Gallucci Mattia

Indice

1. INTRODUZIONE

L'obiettivo del presente documento è delineare le strategie adottate per il testing del sistema SwaGGed. Di seguito verranno illustrate, in modo sintetico, le metodologie previste per l'esecuzione del testing e gli strumenti che saranno impiegati a tal fine.

2. RELAZIONE CON ALTRI DOCUMENTI

- **Relazioni con il Requirements Analysis Document (RAD)**
  I test case definiti nel Test Plan sono sviluppati in base ai requisiti funzionali e non funzionali descritti nel RAD.

- **Relazioni con il System Design Document (SDD)**
  I test case previsti nel Test Plan devono rispettare la suddivisione del sistema in sottosistemi così come illustrato nell'SDD.

- **Relazioni con l'Object Design Document (ODD)**
  Per quanto riguarda il test di unità e di integrazione, strettamente connessi all'ODD e alla suddivisione in package del sistema, tali test saranno documentati esclusivamente all'interno del codice dell'applicativo. Di conseguenza, nel presente documento non saranno inclusi riferimenti al loro design.

.

3. PANORAMICA DEL SISTEMA

Il test di unità sarà eseguito per ciascuna classe contenente uno o più metodi, ad eccezione dei metodi getter, setter e costruttori. Le classi candidate al test di unità sono le seguenti:

- Entità principali: Post, Commento, Community, User
- Data Access Object (DAO):
    - ApprezzaCommentoDAO, ApprezzaPostDAO,
    - CommentoDAO, CommunityDAO, PostDAO
    - SalvaPostDAO, SegnalaCommentoDAO, SegnalaCommunityDAO, SegnalaPostDAO, SegnalaUtenteDAO
    - SegueUtenteDAO, UtenteDAO

Successivamente si procederà con il test di integrazione e, infine, con il test funzionale, concentrandosi sulle funzionalità specificate di seguito.

4. FUNZIONALITÀ DA TESTARE/NON TESTARE

Saranno definiti i test case per tutte le funzionalità descritte nel RAD.

Tuttavia, per ragioni di costi e tempi, il comportamento del sistema è stato testato esclusivamente in base ai casi di test definiti per le seguenti funzionalità:

4.1 Funzionalità da testare:

- Funzionalità Account
  - Autenticazione Utente
  - Registrazione Utente
  - Seguito Utente
- Funzionalità Post
  - Aggiunta Post
  - Eliminazione Post
  - Apprezzamento Post
  - Segnalazione Post
  - Salvataggio Post
  - Ricerca Post
- Funzionalità Commento
  - Aggiunta Commento
  - Elimina Commento
  - Apprezza Commento
  - Segnala Commento
  - Ricerca Commento
- Funzionalità Community
  - Iscrizione alla Community
  - Eliminazione Community
  - Aggiungi Community
  - Ricerca Community

4.2 Funzionalità da non testare:

- Funzionalità Account
  - Modifica Credenziali
- Funzionalità Commento
  - Aggiunta Commento
  - Elimina Commento
  - Apprezza Commento
  - Segnala Commento
  - Ricerca Commento
- Funzionalità Community
  - Modifica nome community.

5 PASS/FAIL CRITERIA

Il testing ha come scopo principale verificare che il sistema SwaGGed adempia completamente ai requisiti funzionali e non funzionali. Il processo si focalizza sull'assicurare che ogni funzionalità implementata lavori correttamente, generando risultati accurati e coerenti con i dati di input.

L'approccio al testing prevede il confronto tra i risultati ottenuti e quelli previsti:

- **Esito positivo (PASS):** il test viene superato se il comportamento del sistema corrisponde a quanto previsto.
- **Esito negativo (FAIL):** il test fallisce qualora emergano incongruenze rispetto ai risultati attesi.

Gli errori vengono individuati attraverso un "oracolo", ovvero un riferimento che definisce il comportamento ideale del sistema. In caso di non conformità, sarà necessario correggere il codice sorgente, ripetere i test e accertarsi che i requisiti siano rispettati in modo completo.

6 APPROCCIO

L'approccio al testing di SwaGGed è strutturato in tre fasi principali, ognuna pensata per garantire che il sistema rispetti i requisiti previsti e funzioni correttamente in tutti gli aspetti.

1. **Test a livello di singole unità**
   La prima fase è dedicata al controllo delle singole classi e dei metodi del sistema. Utilizzando una tecnica di testing black-box, ci si concentra esclusivamente sugli input e sugli output delle componenti, senza esaminare la loro struttura interna. Per affrontare la complessità degli input possibili, si applica una metodologia che suddivide i dati in categorie, trattando ogni categoria come un caso di test distinto. Questo approccio permette di testare ogni classe in modo completo ed efficiente, riducendo il rischio di test ridondanti. Eventuali anomalie individuate durante questa fase vengono corrette e ritestate, per garantire che ogni componente rispetti le specifiche richieste.

2. **Test di integrazione**
   La seconda fase si concentra sul funzionamento del sistema nel suo complesso, verificando come le diverse componenti interagiscono tra loro. L'obiettivo è assicurarsi che tutti i moduli del sistema, come quelli legati alla gestione delle prenotazioni, degli utenti e dei film, funzionino correttamente quando operano insieme. Anche qui, vengono utilizzati test basati su combinazioni significative di input, ottimizzando così il numero di casi di test da eseguire. In questa fase vengono verificati i flussi di dati tra i moduli e il loro corretto interscambio, al fine di evitare errori nelle comunicazioni interne del sistema.

3. **Test complessivo del sistema**
   L'ultima fase prevede la valutazione dell'intero sistema, simulando l'uso reale da parte degli utenti. Oltre a testare le funzionalità principali, come la registrazione, la prenotazione e la gestione delle programmazioni, vengono esaminati anche gli aspetti non funzionali, come la performance e la sicurezza. Questo tipo di test ha lo scopo di verificare che il sistema rispetti pienamente tutti i requisiti definiti nei documenti di progettazione (RAD, SDD, ODD). Il sistema viene testato in un ambiente che replica quello di produzione, per identificare e risolvere eventuali problemi prima del rilascio finale.

7 SOSPENSIONE E RIPRESA

Il testing del sistema SwaGGed verrà interrotto nel caso si verifichino situazioni che impediscano di proseguire con le attività. Le principali cause di sospensione includono il rilevamento di errori critici o bloccanti, come malfunzionamenti che compromettono le funzionalità essenziali, ad esempio guasti del sistema o anomalie nel database che impediscono l'esecuzione dei test pianificati. Inoltre, la sospensione avverrà se ci sono modifiche al sistema che non sono state integrate correttamente, come aggiornamenti o modifiche al codice che non sono stati adeguatamente testati o documentati.

Per riprendere i test, sarà necessario risolvere gli errori bloccanti identificati. Una volta corretti i problemi, si procederà con test di regressione mirati per garantire che le modifiche non abbiano introdotto nuovi malfunzionamenti.

8 TESTING MATERIALS

Per garantire che il sistema CineNow soddisfi tutti i requisiti funzionali e non funzionali, saranno utilizzati strumenti software specializzati che ottimizzano il processo di testing. Di seguito vengono descritti gli strumenti principali che verranno adottati:

Selenium

Selenium sarà utilizzato per l'automazione dei test sull'interfaccia utente (UI). Questo strumento consentirà di simulare le azioni dell'utente finale, come il login, la registrazione, la prenotazione dei posti e la gestione delle programmazioni. Attraverso l'automazione, si potrà verificare l'affidabilità e la correttezza delle interazioni tra l'utente e il sistema, permettendo di eseguire test ripetibili su vari browser e dispositivi. Selenium è particolarmente utile per rilevare anomalie nell'interfaccia grafica, garantendo che l'esperienza utente rimanga coerente e priva di errori.

JUnit

JUnit sarà impiegato come framework principale per i test unitari e di integrazione. Con JUnit, verranno testate le singole classi, come i metodi delle classi DAO, i controller e la logica di business. I test unitari garantiranno che ciascun componente funzioni in modo corretto e isolato, mentre i test di integrazione verificheranno la corretta interazione tra le diverse parti del sistema. Grazie a JUnit, sarà possibile generare report dettagliati che consentiranno di monitorare i risultati dei test e identificare tempestivamente discrepanze o errori, assicurando che tutte le funzionalità siano implementate

9 TEST CASES

Questa sezione costituisce il cuore del piano di test, elencando tutti i casi di test impiegati nel processo di validazione. Ogni caso di test viene descritto in modo approfondito in un documento separato chiamato **Test Case Specification**. Inoltre, per ogni esecuzione dei test, verrà redatto un **Test Incident Report** che registrerà i risultati e segnalerà eventuali anomalie o problemi riscontrati. I dettagli relativi alla gestione di questi documenti saranno forniti in modo più esaustivo nelle sezioni successive.

10 TESTING SCHEDULE

Il piano di testing per il sistema SwaGGed è stato sviluppato per garantire un processo di verifica completo, ben strutturato e sincronizzato con il ciclo di sviluppo. Il testing si articolerà in tre fasi principali, ciascuna focalizzata su un aspetto specifico del sistema, e avrà una durata di una settimana per ogni fase:

- **Test Unitari**: Durante questa fase, che durerà una settimana, verranno testate le singole classi e i metodi tramite JUnit per verificarne il corretto funzionamento in conformità ai requisiti. I test saranno eseguiti parallelamente allo sviluppo del codice, in modo da ottimizzare i tempi e risolvere tempestivamente eventuali problemi a livello di singoli componenti.

- **Test di Integrazione**: Nella seconda settimana, sarà testato l'intero flusso di interazione tra i moduli del sistema, comprese le comunicazioni tra i DAO e il database tramite il DataSource, nonché le interazioni tra le classi di gestione e le JSP.

  L'obiettivo di questa fase è verificare che tutte le dipendenze tra i vari componenti siano correttamente gestite e che i dati vengano scambiati senza errori tra le diverse parti del sistema.

- **Test di Sistema**: Nell'ultima settimana, il sistema sarà testato nel suo complesso. Verranno eseguiti test sulle funzionalità generali e sulle performance del sistema.

  Utilizzando Selenium, saranno simulati scenari d'uso reali, come la registrazione degli utenti, la prenotazione dei posti e la gestione delle programmazioni, per garantire che tutte le funzionalità siano operative e performanti.

Alla fine dello sviluppo, tutti i test verranno rieseguiti per una verifica finale, assicurando che il sistema funzioni correttamente. Ogni fase di test sarà seguita da una revisione dettagliata dei risultati per correggere eventuali anomalie prima di passare alla fase successiva, in modo da garantire un rilascio privo di bug critici.