Università degli Studi di Salerno
Corso di Ingegneria del Software

Rated
Master Test Plan Document
Versione 1.0

18/01/2026

Membri del gruppo

Nome	Matricola
Francesco Rao	NF22500211
Bruno Nesticò	NF22500213
Roberto Fiorenza	NF22500212

Revision History

Data	Versione	Descrizione	Autore
18/01/2026	1.0	Prima stesura completa	Francesco Rao, Bruno Nesticò, Roberto Fiorenza

Indice

1. Introduzione

Il presente documento definisce l'approcio e le specifiche delle attività pianificate per il processo di testing di Rated. Il piano copre i diversi livelli di test previsti, includendo test di unità, di integrazione e di sistema. Per il testing di unità e di integrazione vengono riutilizzati ed estesi i test già esistenti per il progetto originale, in piena continuità con le tecniche adottate in precedenza.

2. Pass/Fail Criteria e Coverage

L'obiettivo del testing è assicurare che Rated rispetti pienamente i requisiti funzionali e non funzionali del progetto originale, nel dettaglio:

- Un **test è considerato superato** se l'output prodotto dal sistema coincide con l'oracolo (comportamento atteso, definito nei casi di test).
- Un **test è considerato fallito** se si riscontrano discrepanze tra il risultato atteso e il risultato effettivo.

In caso di fallimento di un test, è necessario:

1. Identificare e correggere il problema (bug fix).
2. Rieseguire i test unitari e di integrazione interessati (test di regressione).
3. Verificare che le correzioni non abbiano introdotto nuove anomalie.

In termini di coverage dei test, l'obiettivo è raggiungere almeno l'80% di branch coverage. In dettaglio:

• **Unit Testing**: puntare a una copertura ≥80% per i core package.

• **Integration Testing**: valutare la percentuale di codice esercitata dai test integrati, con un obiettivo superiore al 70%.

• **System Testing**: assicurare la validazione completa dei flussi principali del sistema in condizioni operative reali.

3. Approccio

Il testing di Rated si svolge in tre fasi principali:

1. **Testing Unitario**
    - Vengono testate le singole classi (DAO, Service) e i metodi per verificarne la correttezza rispetto alle specifiche dell'ODD_Rated.

    - Si utilizza la metodologia **black-box**: si analizzano input e output senza indagare l'implementazione interna.
    - Si applica la tecnica di **Category Partition** per identificare classi di equivalenza e ridurre i casi di test.
2. **Testing di Integrazione**
    - Verifica le interazioni tra moduli, come ad esempio la cooperazione tra controller/servlet e DAO o tra i moduli di gestione film e gestione recensioni.
    - Si adotta un approccio **bottom-up**, iniziando a integrare e testare i componenti di basso livello (DAO, modelli) per poi passare ai servizi e infine alle servlet e controller.
3. **Testing di Sistema**
    - Si esegue una verifica complessiva in un ambiente il più vicino possibile a quello di produzione.
    - Vengono simulati gli scenari reali (es. registrazione, login, pubblicazione recensione, moderazione, ecc.) per rilevare eventuali problematiche non emerse nelle fasi precedenti.

4. Sospensione e Ripresa

La fase di testing può essere **sospesa** se si verificano condizioni bloccanti, come:

- **Bug critici o bloccanti** che impediscono l'esecuzione dei test pianificati (es. crash del sistema, DB non accessibile).
- **Modifiche non integrate** o non documentate, che rendono instabile l'ultima build del sistema.

Per la **ripresa** delle attività di test:

- Viene effettuata la **correzione dei bug** critici/bloccanti.
- Si eseguono test di regressione per verificare che le modifiche non abbiano introdotto nuovi errori.
- Si aggiornano eventualmente i casi di test e la documentazione, se necessario.

5. Strumenti utilizzati

Per l'esecuzione dei test su Rated, verranno utilizzati i seguenti strumenti:

- **JUnit**
    - Framework principale per i test di unità (sulle classi DAO, servlet, controller).
- **Mockito**
    - Utilizzato per creare **mock** di dipendenze nelle classi in test (es. simulazione accesso DB).