Università degli Studi di Salerno
Corso di Ingegneria del Software

7Movies
Test Plan (TP)
Versione 1.0


SEVEN
MOVIES

Data: 20/01/2020

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Luca Siviero	0512104862
Mario Castigliano	0512104718

Scritto da:	Luca Siviero

Revision History

Data	Versione	Descrizione	Autore
20/01/2020	1.0	Prima stesura	Luca Siviero
31/01/2020	1.1	Completamento	Luca Siviero, Mario Castigliano

Indice

1. INTRODUZIONE

Lo scopo di questo documento è quello di analizzare e gestire lo sviluppo e le attività di testing riguardanti la piattaforma 7Movies. Questa sessione di lavoro deve verificare il corretto funzionamento della piattaforma nella totalità dei casi, studiati appositamente per mettere alla prova ogni singola funzionalità e caratteristica del sistema. I risultati di questi test saranno utilizzati per capire dove bisognerà intervenire, e quindi correggere eventuali errori o apportare modifiche per il miglioramento dei vari sottosistemi. Il processo viene iterato fino a che non si raggiungeranno i risultati attesi in accordo con i tempi di sviluppo previsti.

2. RELAZIONE CON GLI ALTRI DOCUMENTI

Per verificare il corretto funzionamento della piattaforma sono stati predisposti dei test case basati sul alcune delle funzionalità individuate nella fase di raccolta dei requisiti e verranno trattati nel seguente documento. Inoltre, ad ogni test case sarà associato un test case specification (inclusi nel documento 7Movies_TCS), contenente la simulazione di input e output con il conseguente risultato.

3. PANORAMICA DEL SISTEMA

La piattaforma 7Movies nasce per soddisfare due scopi: permettere al pubblico la fruizione di opere cinematografiche da casa e raccogliere appassionati da tutto il mondo offrendo un luogo dove recensire i film.
Il primo obiettivo viene raggiunto grazie alla possibilità, che la piattaforma offre, di acquistare in forma digitale le pellicole a cui si è interessati, sfruttando il meccanismo definito come digital delivery che rende più immediata la fruizione.
Ogni utente ha quindi la possibilità di recensire i film in catalogo, sia che ne abbia acquistato una copia sulla piattaforma che altrimenti.

4. FUNZIONALITÀ DA TESTARE/NON TESTARE

Verranno testate alcune delle funzionalità individuate nell'analisi dei requisiti e che la versione finale di 7Movies presenterà. Non verranno eseguiti testing sulle prestazioni dei vari moduli. In particolare, saranno testate:

- Gestione del Sistema: sarà testata la funzionalità di ricerca dei film.
- Gestione Utente: verranno testate le funzionalità di Registrazione e Login.
- Gestione Film: sarà testata la funzionalità di inserimento film.
- Gestione Carrello: sarà testata la funzionalità di aggiunta di un film al carrello.

5. PASS/FAIL CRITERIA

Il testing ha successo se l'output osservato è diverso dall'output atteso: ciò significa che la fase di testing avrà successo se individuerà una failure. In tal caso questa verrà analizzata e, se legata ad un fault, si procederà alla sua correzione. Sarà infine iterata la fase di testing per verificare che la modifica non abbia impattato su altri componenti del sistema.
Al contrario, il testing fallirà se l'output osservato sarà uguale all'oracolo.

6. APPROCCIO

Nella sessione di testing della piattaforma verrà utilizzato un approccio di tipo “BLACK BOX”, che prevede che i test vengano effettuati in maniera da non scendere nei dettagli del codice, ma basandosi sulle specifiche delle funzionalità da testare. L’approccio alla fase di testing si compone di tre fasi:

● Testing di unità, che controlla i singoli moduli;

● Testing di integrazione, che utilizza un approccio di tipo “bottom-up” in modo da non dover utilizzare gli stub ma solamente i driver;

● Testing di sistema, che controlla se il software soddisfa tutte le funzionalità specificate.

6.1. *Testing di unità*

Con il testing di unità verrà effettuato un controllo delle varie classi e metodi del sistema, quindi saranno ricercate le condizioni di fallimento andando ad evidenziare gli errori. Il testing di unità, sarà eseguito dal team di sviluppo attraverso l’implementazione di classi di test utilizzando il framework JUnit. In particolare, per ogni classe che esegue operazioni complesse sarà sviluppata la relativa classe JUnit.

6.2. *Testing di integrazione*

Il testing di integrazione consente di individuare i problemi che si verificano quando due unità si combinano. Se si utilizza un piano di test che prevede il test di ciascuna unità e la verifica della validità di ognuna prima di combinarle, sarà evidente che gli errori rilevati nella combinazione delle unità sono probabilmente legati alla relativa interfaccia. Questo metodo consente di ridurre notevolmente il numero di possibilità e di semplificare in larga misura l'analisi. Verrà utilizzato un approccio di tipo “bottom-up” che prevede il solo utilizzo dei driver, senza dover ricorrere all’utilizzo degli stub. Sarà realizzato con il supporto dello strumento Selenium e i risultati del test saranno descritti nel documento ITR (Integration Test Report).

6.3. *Testing di sistema*

Con il testing di sistema verrà effettuato un controllo della conformità dell'intero sistema con i requisiti dell'utente finale. Verranno testate manualmente tutte le componenti realizzate e le funzionalità previste e specificate nei requisiti funzionali e non funzionali.

7. PIANIFICAZIONE DEL TESTING

Il testing verrà effettuato utilizzando il metodo del Category Partition: è un tipo di test combinatorio che permette di identificare attributi, valori rilevanti e possibili combinazioni, permettendo la separazione dell'identificazione dei valori che caratterizzano lo spazio di input dalla combinazione di valori diversi in casi di test completi; fornisce una stima del numero dei casi di test molto presto.
Sono previste 3 fasi diverse:

- Decomporre le specifiche in feature testabili indipendentemente : in questo passo, il test designer deve identificare le feature che devono essere testate in modo separato, identificando i parametri e qualunque altro elemento dell'ambiente di esecuzione da cui dipende (ad esempio un database). Per ciascun parametro e elemento dell'ambiente si identificano le caratteristiche (elementari) del parametro, dette categorie.
- Identificare Valori Rappresentativi : questo passo, prevedere che il test designer identifichi un insieme di valori rappresentativi (classe di valori, detta choices) per ciascuna caratteristica di ogni parametro definito nella fase precedente.
- Generare Specifiche di Casi di Test : Per ogni Test Case realizzato, verranno prodotti i relativi Test Case Specifications contenenti gli esempi di tutte le casistiche individuate per ogni funzionalità con i relativi risultati.

Poiché le query sono state implementate nelle classi DAO, il testing di integrazione si assicurerà del corretto funzionamento di queste, andando a testare le funzionalità del sistema che utilizzano un’interfaccia per collegare due o più componenti. In questo modo ci assicureremo che le classi DAO eseguano le query in una maniera corretta, e simultaneamente, che non ci siano problemi legati all’interfaccia. Il test di integrazione sarà realizzato attraverso il framework Selenium, con utilizzo del ChromeWebDriver per automatizzare la fase di testing.

8. TEST CASE

Per sviluppare i test cases sarà utilizzato il metodo del Category Partition. Questo metodo consiste nell'identificare per ogni funzionalità da testare dei parametri; per ogni parametro verranno individuate delle categorie, le quali poi saranno suddivise in scelte. Alle scelte verrà assegnato un valore.

9. SPECIFICA DEI TEST CASE

Di seguito sono riportati tutti i test cases realizzati raggruppati in packages.

9.1. *TC_1: GESTIONE SISTEMA*

9.1.1. *TC_1.1: Ricerca film*

Parametro: titolo, regista, anno **Formato: [a-zA-z0-9.%+-] {1-50}**	
Categorie	**Scelte**
lunghezza lt	1: lunghezza = = 0 [errore] 2: lunghezza <=50 [property lunghezzaLTok]
lunghezza lr	1:lunghezza = = 0 [errore] 2: lunghezza <=50 [property lunghezzaLRok]
lunghezza la	1:lunghezza == 0 [error] 2:lunghezza <= 4 [property lunghezzaLAok]

Codice	**Combinazione**	**Esito**
FC_1.1.01	lt1	Errore
FC_1.1.02	lr1	Errore
FC_1.1.03	la1	Errore
FC_1.1.04	combinazione di due o più eventi precedenti	Errore
FC_1.1.05	lt2.lr2.la2	Ricerca film

9.2. *TC_2: GESTIONE UTENTE*

9.2.1. *TC_2.1: Registrazione*

Parametro: Nome **Formato: [a-zA-Z] {3-30}**	
Categorie	**Scelte**
lunghezza ln	1: lunghezza == 0 [errore] 2: lunghezza <=30 [property lunghezzaLNok]

Parametro: Cognome **Formato: [a-zA-Z] {3-30}**	
Categorie	**Scelte**
lunghezza lc	1: lunghezza == 0 [errore] 2: lunghezza <=30 [property lunghezzaLCok]

Parametro: Password **Formato: [a-zA-z0-9._%+-]{6,30}**	
Categorie	**Scelte**
lunghezza lp	1: lunghezza == 0 [errore] 2: lunghezza <=30 [property lunghezzaLPok]

Parametro: email **Formato: [A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{10,50}**	
Categorie	**Scelte**
lunghezza le	1: lunghezza == 0 [errore] 2: lunghezza <=50 [property lunghezzaLEok]
Formato fe	1: rispetta il formato [if lunghezzaLEok] [property formatoFEok] 2: non rispetta il formato [if lunghezzaLEok] [errore]

Codice	Combinazione	Esito
FC_2.1.01	ln1	Errore
FC_2.1.02	lc1	Errore
FC_2.1.03	le1	Errore
FC_2.1.04	fe2	Errore
FC_2.1.05	lp1	Errore
FC_2.1.06	combinazione di due o più eventi precedenti.	Errore
FC_2.1.07	ln2.lc2.lp2.le2.fe1	Registrazione

TC_2.2: Login

Parametro: email **Formato: [A-Z0-9._%+-]{2-15}**	
Categorie	Scelte
lunghezza le	1: lunghezza == 0 [errore] 2: lunghezza <=50 [property lunghezzaLEok] 3: lunghezza >50 [errore]
Esiste EM	1: non esiste nel DB [if lunghezzaLEok] [property esisteEMok] 2: esiste nel DB [if lunghezzaLEok] [errore]
Formato fe	1: rispetta il formato [if lunghezzaLEok] [property formatoFEok] 2: non rispetta il formato [if lunghezzaLEok] [errore]

Parametro: Password **Formato: [a-zA-z0-9._%+-]{6,30}**	
Categorie	Scelte
lunghezza lp	1: lunghezza == 0 [errore] 2: lunghezza <=30 [property lunghezzaLPok]
Corrisponde cp	1: uguale alla password esistente nel DB per l'username inserito. [if lunghezzaLPok] [property corrispondeCPok] 2: non uguale alla password presente nel DB. [if lunghezzaLPok] [errore]

Codice	**Combinazione**	**Esito**
FC_2.2.01	le1	Errore
FC_2.2.02	le3	Errore
FC_2.2.03	EM1	Errore
FC_2.2.04	fe2	Errore
FC_2.2.05	lp1	Errore
FC_2.2.07	cp2	Errore
FC_2.2.08	combinazione di due o più eventi precedenti	Errore
FC_2.2.09	le2.EM2.fe1.lp2.cp1	Login

9.3. *TC_3: GESTIONE FILM*

9.3.1. *TC_3.1: Inserimento film*

Parametro: Titolo **Formato: [a-zA-z0-9._%+-]{1,50}**	
Categorie	Scelte
lunghezza lt	1: lunghezza = =0 [errore] 2: lunghezza>=1 [property lunghezzaLTok]

Parametro: regista **Formato: [a-zA-z0-9._%+-]{1,50}**	
Categorie	Scelte
lunghezza lr	1: lunghezza = =0 [errore] 2: lunghezza>=1 [property lunghezzaLRok]

Parametro: Categoria **Formato: [a-zA-z0-9._%+-]{1,255}**	
Categorie	Scelte
lunghezza lc	1: lunghezza = =0 [errore] 2: lunghezza>=1 [property lunghezzaLCok]

Parametro: Immagine **Formato: [a-zA-z0-9._%+-]{1,255}**	
Categorie	Scelte
lunghezza li	1: lunghezza = = 0[errore] 2: lunghezza>=1 [property lunghezzaLIok]

Parametro: framePath **Formato: [0 -9] {1,999}**	
Categorie	Scelte
lunghezza lfp	1: lunghezza == 0[errore] 2: lunghezza>=1 [property lunghezzaLFPok]

Parametro: lingue **Formato: [a-zA-z0-9._%+-]{1,255}**	
Categorie	Scelte
lunghezza lLin	1: lunghezza == 0 [errore] 2: lunghezza>=1 [property lunghezzalLinok]

Parametro: Descrizione **Formato: [0 -9] {1,2048}**	
Categorie	Scelte
lunghezza ld	1: lunghezza == 0 [errore] 2: lunghezza >0 [property lunghezzaLDok]

Parametro: Prezzo **Formato: [0 -9] {1,4} +\.[0-9]{0,2}**	
Categorie	Scelte
lunghezza lp	1: lunghezza == 0 [errore] 2: lunghezza >0 [property lunghezzaLPok]
Formato fp	1: rispetta il formato [if lunghezzaLPok] [property formatoFPok] 2: non rispetta il formato [if lunghezzaLPok] [errore]

Codice	Combinazione	Esito
FC_3.1.01	lt1	Errore
FC_3.1.02	lr1	Errore
Fc_3.1.10	lp1	Errore
FC_3.1.11	la1	Errore
FC_3.1.03	lc1	Errore
FC_3.1.04	li1	Errore
FC_3.1.05	lfp1	Errore
FC_3.1.06	lLin1	Errore
FC_3.1.07	ld1	Errore
FC_3.1.08	combinazione di due o più eventi precedenti	Errore
FC_3.1.09	lt2.lr2.lc2.li2.lpf2.lLin2.ld2.fp1	Inserimento film

9.4. *TC_4: GESTIONE CARRELLO*

9.4.1. *TC_4.1: Aggiunta film al carrello*

Parametro: id	
Categorie	**Scelte**
esiste ei	1: non esiste nel DB [errore] 2: esiste nel DB [property esisteEIok]

Codice	**Combinazione**	**Esito**
FC_4.1.01	ei1	Errore
FC_4.1.02	ei2	Aggiunta film

10. RELEASE CRITERIA

Per ogni esecuzione del test tutte le informazioni saranno riportate nel documento TER e per ogni esecuzione automatizzata ci sarà un documento ITR dove vengono riportati le info sui test case che hanno successo. Il processo è iterativo per ogni esecuzione del test fin quando non ci saranno esclusivamente test case che avranno successo.

11. GLOSSARIO

Definizioni:

- **ODD**: Documento che riporta e analizza gli oggetti che compongono il sistema analizzando le componenti a più basso livello, riportandole così come saranno implementate.
- **RAD**: Documento di Raccolta e analisi dei Requisiti che contiene l'elenco dei requisiti funzionali e non funzionali individuati in fase di individuazione dei stessi e la loro analisi sotto forma di scenari e casi d'uso. I mock-up mostrano una possibile implementazione dell'interfaccia del sistema.
- **SDD**: Documento che riporta la progettazione del sistema come risultato di una prima fase di modellazione: contiene una suddivisione ad alto livello del sistema nei sottosistemi che lo comporranno.
- **TCS**: Documento che specifica i casi di test in tutti i loro dettagli.
- **TP**: Documento che descrive il piano di testing adottato nel progetto e la definizione dei casi di test.

Acronimi:

- **ODD**: Object Design Document;
- **RAD**: Requirement Analysis Document;
- **SDD**: System Design Document;
- **TC**: Test case;
- **TCS**: Test case specification;
- **TP**: Test Plan;