Università degli Studi di Salerno
Corso di Ingegneria del Software

CERICinema
Test Plan (TP)
Versione 1.0

Coordinatore del progetto:

Nome	**Matricola**

Partecipanti:

Nome	**Matricola**
Edilio Massaro	0512103218
Ildo Tiberio	0512103134
Raffaele Squeglia	0512103122
Chiara Bufalino	0512102894

Scritto da:	

Revision History

Data	**Versione**	**Descrizione**	**Autore**

Indice

INTRODUZIONE

Lo scopo di questo documento è quello di analizzare e gestire lo sviluppo e le attività di testing riguardanti la piattaforma CERICinema. Questa sessione di lavoro deve verificare il corretto funzionamento della piattaforma nella totalità dei casi, studiati appositamente per mettere alla prova le funzionalità principali 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.

1. RELAZIONE CON GLI ALTRI DOCUMENTI

Per verificare il corretto funzionamento della piattaforma sono stati predisposti dei test case basati sulle 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 TCS_CERICinema), contenente la simulazione di input e output con il conseguente risultato.

2. PANORAMICA DEL SISTEMA

CERICinema nasce per facilitare agli utenti amanti di cinema l'acquisto di un biglietto per la proiezione desiderata. Infatti, di solito sarebbero costretti ad affrontare interminabili minuti di file per poter acquistare il biglietto del film in questione. Grazie a questo software potranno ordinare comodamente da casa, con un semplice click i biglietti per le proiezioni dei loro film preferiti, saltando inutili file e pagando comodamente online.
Lo scopo della piattaforma è dunque offrire una soluzione rapida, comoda e intuitiva alla portata di tutti.

3. *FUNZIONALITA'DA TESTARE/NON TESTARE*

Verranno le funzionalità principali che la versione finale di CERICinema presenterà. Non verranno eseguiti testing sulle prestazioni dei vari moduli. In particolare, saranno testate:

- Gestione delle Proiezioni: verranno testate le funzionalità di aggiunga, visualizzazione e modifica di una proiezione.
- Gestione Utente: verranno testate le funzionalità di Registrazione, Login e Logout.
- Gestione Film: verranno testate le funzionalità di visualizzazione dei film.
- Gestione Prenotazioni: verranno testate le funzionalità di visualizzazione ed inserimento di prenotazione.
- Gestione Acquisti: verranno testale funzionalità di visualizzazione dello storico acquisti.

4. 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.

5. 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.

5.1. 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.

5.2. 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.

5.3. 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 le funzionalità principali presenti nei requisiti funzionali del sistema.

6. 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 feauture testabili indipendentemente : in questo passo, il test designer deve identificare le feauture 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 Servlet, il testing di integrazione verrà effettuato solo per la classe DriverManagerConnectionPool.java, testando la correttezza strutturale delle query con il test sulle Servlet effettuato nel test di sistema.

7. 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.

8. SPECIFICA DEI TEST CASE

Di seguito sono riportati tutti i test cases realizzati.

8.1. TC_1: GESTIONE UTENTE

TC_1.1: Registrazione

Parametro: Nome **Formato: [a-zA-Z] {3-25}**	
Categorie	**Scelte**
lunghezza ln	1: lunghezza <=2[errore] 2: lunghezza >=3 && lunghezza <=25 [property **lunghezzaLNok]** 3: lunghezza >25 [errore]
Formato fn	1: rispetta il formato [if **lunghezzaLNok]** [property **formatoFNok]** 2: non rispetta il formato [if **lunghezzaLNok]** [errore]

Parametro: Cognome **Formato: [a-zA-Z] {3-30}**	
Categorie	**Scelte**
lunghezza lc	1: lunghezza <=2 [errore] 2: lunghezza >=3 && lunghezza <=30 [property **lunghezzaLCok]** 3: lunghezza >30 [errore]
Formato fc	1: rispetta il formato [if **lunghezzaLCok]** [property **formatoFCok]** 2: non rispetta il formato [if **lunghezzaLCok]** [errore]

Parametro: Username **Formato: [A-Z0-9._%+-]{2-15}**	
Categorie	**Scelte**
lunghezza lu	1: lunghezza <=2 [errore] 2: lunghezza >=2 && lunghezza <=15 [property **lunghezzaLUok]** 3: lunghezza >15 [errore]
Esiste eu	1: non esiste nel DB [if **lunghezzaLUok**] [property **esisteEUok**] 2: esiste nel DB [if **lunghezzaLUok]** [errore]

Parametro: Password **Formato: [a-zA-z0-9._%+-]{6,30}**	
Categorie	**Scelte**
lunghezza lp	1: lunghezza <=5 [errore] 2: lunghezza >=6 && lunghezza <=30 [property **lunghezzaLPok]** 3: lunghezza >30 [errore]

Parametro: Codice fiscale **Formato: [a-zA-z0-9]{16}**	
Categorie	**Scelte**
lunghezza lcf	1: lunghezza <=15[errore] 2: lunghezza ==16 [property **lunghezzaLCFok]** 3: lunghezza >16 [errore]
Formato fcf	1: rispetta il formato [if **lunghezzaLCFok]** [property **formatoFCFok]** 2: non rispetta il formato [if **lunghezzaLCFok]** [errore]

Parametro: Data di nascita **Formato: [0-9]{1,11}**	
Categorie	**Scelte**
lunghezza ldn	1: lunghezza = =0[errore] 2: la data viene selezionata da una box apposita [property **lunghezzaLDNok**]

Parametro: email **Formato: [A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{10,50}**	
Categorie	**Scelte**
lunghezza le	1: lunghezza <=9 [errore] 2: lunghezza >=10 && lunghezza <=50 [property **lunghezzaLEok]** 3: lunghezza >50 [errore]
Formato fe	1: rispetta il formato [if **lunghezzaLEok]** [property **formatoFEok]** 2: non rispetta il formato [if **lunghezzaLEok]** [errore]

Codice	Combinazione	Esito
FC_1.1.01	ln1	Errore
FC_1.1.02	ln3	Errore
FC_1.1.03	ln2.fn2	Errore
FC_1.1.04	ln3.fn1.lc1	Errore
FC_1.1.05	ln3.fn1.lc3	Errore
FC_1.1.06	ln3.fn1.lc2.fc2	Errore
FC_1.1.07	ln3.fn1.lc3.fc1.lu1	Errore
FC_1.1.10	ln3.fn1.lc3.fc1.lu3	Errore
FC_1.1.12	ln3.fn1.lc3.fc1.lu2.eu2	Errore
FC_1.1.10	ln3.fn1.lc3.fc1.lu2.eu1.lp1	Errore
FC_1.1.11	ln3.fn1.lc3.fc1.lu2.eu1.lp3	Errore
FC_1.1.12	ln3.fn1.lc3.fc1.lu3.eu1.lp2.lcf1	Errore
FC_1.1.13	ln3.fn1.lc3.fc1.lu3.eu1.lp3.lcf3	Errore
FC_1.1.14	ln3.fn1.lc3.fc1.lu3.eu1.lp3.lcf2.fcf2	Errore
FC_1.1.15	ln3.fn1.lc3.fc1.lu3.eu1.lp3.lcf1.fcf1.ldn1	Errore
FC_1.1.16	ln3.fn1.lc3.fc1.lu3.eu1.lp3.lcf1.fcf1.ldn2.le1	Errore
FC_1.1.17	ln3.fn1.lc3.fc1.lu3.eu1.lp3.lcf1.fcf1.ldn2.le3	Errore
FC_1.1.18	ln3.fn1.lc3.fc1.lu3.eu1.lp3.lcf1.fcf1.ldn2.le2.fe2	Errore
FC_1.1.19	ln3.fn1.lc3.fc1.lu3.eu1.lp3.lcf1.fcf1.ldn2.le3.fe1	Registrazione

TC_1.2: Login

Parametro: Username **Formato: [A-Z0-9._%+-]{2-15}**	
Categorie	**Scelte**
lunghezza lu	1: lunghezza <=2 [errore] 2: lunghezza >=2 && lunghezza <=15 [property **lunghezzaLUok]** 3: lunghezza >15 [errore]
Esiste eu	1: non esiste nel DB [if **lunghezzaLUok**] [property **esisteEUok**] 2: esiste nel DB [if **lunghezzaLUok]** [errore]

Parametro: Password **Formato: [a-zA-z0-9._%+-]{6,30}**	
Categorie	**Scelte**
lunghezza lp	1: lunghezza <=5 [errore] 2: lunghezza >=6 && lunghezza <=30 [property **lunghezzaLPok]** 3: lunghezza >30 [errore]
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_1.2.01	lu1	Errore
FC_1.2.02	lu3	Errore
FC_1.2.03	lu2.eu2	Errore
FC_1.2.04	lu3.eu1.lp1	Errore
FC_1.2.05	lu3.eu1.lp3	Errore
FC_1.2.06	lu3.eu1.lp2.cp2	Errore
FC_1.2.07	lu3.eu1.lp3.cp1	Login

TC_1.3: Logout

Codice	Combinazione	Esito
FC_1.3.01		Logout

8.2. TC_2: GESTIONE PROIEZIONI

TC_2.1: Visualizzazione proiezione

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

Codice	Combinazione	Esito
FC_2.1.01	ei1	Errore
FC_2.1.02	ei2	Visualizza proiezione

TC_2.2: Inserimento Proiezione

Parametro: Nome Proiezione **Formato: [a-zA-z0-9]{1,30}**	
Categorie	**Scelte**
lunghezza lnp	1: lunghezza = =0 [errore] 2: lunghezza>=1 && lunghezza <=30 [property **lunghezzaLTok]** 3: lunghezza>30 [errore]

Parametro: Sala **Formato: [a-zA-z0-9]{1,3}**	
Categorie	**Scelte**
lunghezza ls	1: lunghezza = =0 [errore] 2: lunghezza>=1 && lunghezza <=3[property **lunghezzaLAok]** 3: lunghezza>3 [errore]

Parametro: DataProiezione **Formato: [a-zA-z0-9-]{8}**	
Categorie	**Scelte**
lunghezza ld	1: lunghezza = =0 [errore] 2: lunghezza = 8 [property **lunghezzaLDok** 3: lunghezza>30 [errore]

Parametro: Film **Formato: [a-zA-z0-9]**	
Categorie	**Scelte**
lunghezza lf	1: lunghezza = = 0[errore] 2: lunghezza>=2 [property **lunghezzaLCok**]

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

Codice	Combinazione	Esito
FC_2.3.01	lnp1	Errore
FC_2.3.02	lnp3	Errore
FC_2.3.03	lnp2.ls1	Errore
FC_2.3.04	lnp2.ls3	Errore
FC_2.3.05	lnp2.ls2.ld1	Errore
FC_2.3.06	lnp2.ls2.ld3	Errore
FC_2.3.07	lnp2.ls2.ld2.lf1	Errore
FC_2.3.08	ltnp.ls2.ld2.lf2	Errore
FC 2.3.09	ltnp.ls2.ld2.lf2.lp1	Errore
FC 2.3.09	ltnp.ls2.ld2.lf2.lp2	Inserimento

TC_2.3: Modifica Proiezioni

Parametro: Nome Proiezione Formato: [a-zA-z0-9]{1,30}	
Categorie	**Scelte**
lunghezza lnp	1: lunghezza = =0 [errore] 2: lunghezza>=1 && lunghezza <=30 [property **lunghezzaLTok]** 3: lunghezza>30 [errore]

Parametro: Sala Formato: [a-zA-z0-9]{1,3}	
Categorie	**Scelte**
lunghezza ls	1: lunghezza = =0 [errore] 2: lunghezza>=1 && lunghezza <=3[property **lunghezzaLAok]** 3: lunghezza>3 [errore]

Parametro: DataProiezione Formato: [a-zA-z0-9-]{8}	
Categorie	**Scelte**

lunghezza ld	1: lunghezza = =0 [errore] 2: lunghezza = 8 [property **lunghezzaLDok** 3: lunghezza>30 [errore]

Parametro: Film **Formato: [a-zA-z0-9]**	
Categorie	**Scelte**
lunghezza lf	1: lunghezza = = 0[errore] 2: lunghezza>=2 [property **lunghezzaLCok**]

Parametro: Prezzo **Formato: [0 -9] {1,4} +\.[0-9]{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_3.3.01	lnp1	Errore
FC_3.3.02	lnp3	Errore
FC_3.3.03	lnp2.ls1	Errore
FC_3.3.04	lnp2.ls3	Errore
FC_3.3.05	lnp2.ls2.ld1	Errore
FC_3.3.06	lnp2.ls2.ld3	Errore
FC_3.3.07	lnp2.ls2.ld2.lf1	Errore
FC_3.3.08	ltnp.ls2.ld2.lf2	Inserimento

8.3. TC_3: GESTIONE PRENOTAZIONE

TC_3.1: Visualizza Prenotazione

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

Codice	**Combinazione**	**Esito**
FC_3.1.01	ei1	Errore
FC_3.1.02	ei2	Visualizza Prenotazione

TC_3.2: Effettua Prenotazione

Parametro: posto	
Categorie	**Scelte**
esiste ei	1: posto == null [errore] 2: posto scelto [propety **esisteEIok**]

Codice	**Combinazione**	**Esito**
FC_3.1.01	ei1	Errore
FC_3.1.02	ei2	Visualizza Prenotazione

8.4. TC_4: GESTIONE ACQUISTO

TC_4.1: Visualizzare acquisto

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

8.5. TC_5 GESTIONE FILM

TC_5.1: Visualizzare Film

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

9. RELEASE CRITERIA

Per ogni esecuzione del test tutte le informazioni saranno riportate nel documento TER e per ogni esecuzione ci sarà un documento TIR 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.

10. 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;