Università degli Studi di Salerno

Corso di Ingegneria del Software

UNISA-TCG

Test Plan

Data: 22/12/2025

Progetto: UNISA-TCG	Versione: 1.0
Documento: Test Plan	Data: 22/12/2025

Coordinatore del progetto:

Nome	Matricola
Sepe Giuseppe	0512119386

Partecipanti:

Nome	Matricola
Senatore Francesco	0512120568
Sepe Giuseppe	0512119386
Sullo Diego	0512119455

Scritto da:	Senatore Francesco, Sepe Giuseppe, Sullo Diego

Revision History

Data	Versione	Descrizione	Autore
22/12/2025	**1.0**	**Test Plan**	**Francesco Senatore, Diego Sullo, Giuseppe Sepe**

Indice

1. Introduzione

Il presente documento descrive la strategia di testing per la piattaforma UNISA-TCG. L'obiettivo è validare le funzionalità core di navigazione e ricerca per garantire un'esperienza utente fluida e priva di errori.

1.1 Obiettivi del test

- Garantire che la visualizzazione dei prodotti sia coerente con i dati presenti nel database.
- Verificare l'accuratezza della ricerca tramite filtri.
- Assicurarsi che la navigazione tra catalogo e pagina prodotto sia priva di interruzioni.

1.2 Scope del test

Lo scope è limitato al catalogo, attraverso le classi Categoria, DBConnection e prodotto specificamente:

- Caricamento della lista prodotti.
- Visualizzazione dettagliata del singolo prodotto.
- Funzionamento dei filtri (Rarità, Set, Prezzo).

1.3 Analisi del sistema

Il sistema è un'applicazione web basata su architettura Model-View-Controller (MVC). Il front-end comunica con un database relazionale per recuperare le informazioni sui prodotti e le relative immagini.

2. Approccio

L'approccio sarà di tipo **Black Box** per i test funzionali (testando l'interfaccia come un utente finale) e **White Box** per la validazione delle query di ricerca a livello di codice.

2.1 Vincoli

- I test devono essere completati entro la data di consegna del progetto.
- Limitazione dei dati: i test verranno eseguiti su un set di dati controllato (popolamento DB di test).

2.2 Copertura

La copertura sarà garantita tramite Weak Equivalence Class Testing.

- Si identificano le classi di equivalenza per gli input (es. Prezzo, Rarità, Nome).
- Si sceglie un rappresentante per ogni classe valida.
- Ogni caso di test coprirà un valore rappresentativo per ogni variabile di input, assumendo che i valori della stessa classe si comportino allo stesso modo.

2.3 Test Tools

- **Browser (Chrome/Firefox):** Per l'esecuzione manuale o automatizzata dei test.
- **Selenium IDE / Katalon Recorder:** Per registrare e riprodurre le azioni dell'utente sul browser.
- **Fogli di calcolo (Excel/Google Sheets):** Per la tracciabilità della matrice dei casi di test e dei risultati.

2.4 Test Type

- **Functional Testing:** Verifica che i filtri mostrino i prodotti corretti.
- **Boundary Value Analysis (Integrata):** Test sui limiti dei filtri (es. prezzo minimo e massimo).

2.5 Test Data

Un set di prodotti pre-caricati nel sistema con caratteristiche note (nomi, rarità, prezzi) per confrontare l'output del sistema con il risultato atteso.

3. Test Plan

3.1 Test Team

- **Responsabile Testing:** Giuseppe Sepe
- **Sviluppatori:** Francesco Senatore, Diego Sullo, Giuseppe Sepe

3.2 Major Task

1. Predisposizione dell'ambiente di test e del DB.
2. Scrittura dei casi di test per filtri e visualizzazione.
3. Esecuzione dei test manuali e automatizzati.
4. Reportistica dei bug e ri-test dopo il bug-fixing.

3.3 Ambiente di Test

- **Hardware:** PC con architettura x64.
- **Software:** Server locale (Apache/Tomcat), Browser (Chrome, Firefox).
- **DB:** MySQL.

4. Feature da Testare

- **Visualizzazione Catalogo:** Visualizzazione corretta della griglia prodotti.
- **Visualizzazione Prodotto:** Correttezza dei dati tecnici nella pagina singola.
- **Ricerca e Filtri:** Precisione dei risultati filtrati.

5. Procedure di Testing

In questa fase, il sistema viene sollecitato simulando le azioni di un utente reale che naviga su **UNISA-TCG**. Utilizzeremo casi di test che combinano diverse classi di input per massimizzare l'efficienza.

5.1 Definizione delle Classi di Equivalenza (EC)

Prima di eseguire i test, abbiamo mappato il dominio degli input:

- **Gioco (G):** ECG1: Pokémon, ECG2: Magic, ECG3: Yu-Gi-Oh!.
- **Tipo Prodotto (T):** ECT1: Booster Box, ECT2: Deck Box, ECT3: Accessorio (es. bustine protettive).

5.2 Test Case: Filtri di Ricerca Incrociati

Applichiamo la strategia **Weak Equivalence** per coprire ogni classe almeno una volta con il minor numero di test.

ID	Azione Utente (Input)	Classi Coperte	Risultato Atteso (Output)
TC_01	Filtra: **Pokémon** + **Booster Box**	G1,T1	Visualizzazione di box sigillati Pokémon (es. *Set Base*).
TC_02	Filtra: **Magic** + **Deck Box**	G2,T2	Visualizzazione di porta-mazzi a tema Magic.
TC_03	Filtra: **Yu-Gi-Oh!** + **Accessorio**	G3,T3	Visualizzazione di bustine protettive (sleeves) Yu-Gi-Oh!.

5.3 Test di Visualizzazione Prodotto (Deep Dive)

In questo scenario, testiamo se la "scatola nera" mostra i metadati corretti quando si esce dal catalogo per entrare nel dettaglio.

- **Scenario:** L'utente clicca su un prodotto "Booster Box Pokémon Evoluzioni".
- **Dati attesi in uscita:**
    1. Immagine nitida del box.
    2. Descrizione che specifica il numero di bustine (es. 36 pack).
    3. Etichetta di disponibilità (In Stock / Out of Stock).
    4. Prezzo aggiornato comprensivo di IVA.

5.4 Test di "Nessun Risultato"

Fondamentale per la robustezza del Black-Box è capire come il sistema gestisce l'assenza di match nelle classi di equivalenza.

- **Azione:** Applicare un filtro paradossale (es. Gioco: Yu-Gi-Oh! + Tipo: Booster Box).
- **Risultato Atteso:** La pagina non deve rompersi, non deve apparire nessun prodotto del Catalogo.

6. Rischi e contingenze

- **Rischio:** Mancato caricamento delle immagini delle carte da server esterni.
- **Contingenza:** Utilizzo di immagini segnaposto (placeholder) locali per i test di layout.
- **Rischio:** Prestazioni lente della ricerca con molti utenti.
- **Contingenza:** Ottimizzazione degli indici del database.