Laurea Triennale in informatica-Università di Salerno
Corso di *Ingegneria del Software*- Prof. C. Gravino

Test Plan

MatchPoint

Riferimento	
Versione	1.0
Data	15/12/2025
Destinatario	Docente Ingegneria del Software Prof. Carmine Gravino
Presentato da	Gaetano Aprile, Luigi Artuso, Alessandro De Bonis & Marco Galdi
Approvato da	

1. Introduzione

1.1 Scopo del Documento

Il presente documento descrive la strategia, l'ambito e l'approccio pianificato per le attività di testing del sistema software **MatchPoint**. L'obiettivo principale è verificare la correttezza funzionale dei componenti di business logic, assicurando che soddisfino i requisiti specificati nel documento di Analisi dei Requisiti (RAD).

1.2 Metodologia

La metodologia adottata per la progettazione dei casi di test è il **Category-Partition Method** (Ostrand & Balcer, 1988). Questo metodo di testing *Black-Box* permette di decomporre le specifiche funzionali in unità testabili, identificare parametri e ambienti, e generare un insieme sistematico di Test Case.

2. System Overview

Gli elementi oggetto di testing sono i componenti software responsabili della logica applicativa (Service Layer) identificati nel System Design Document (SDD).
Nello specifico, verranno testati i flussi definiti nei seguenti Casi d'Uso del RAD:

- **UC_1**:Registrarsi alla piattaforma (Gestito da: `GestioneUtentiService`)
- **UC_2**: Dare una valutazione ad un partecipante (Gestito da: `GestioneValutazioneService`)
- **UC_3**: Iscriversi ad un evento (Gestito da: `PartecipazioneEventoService`)
- **UC_4**: Creare un evento sportivo (Gestito da: `GestioneEventoService`)

3. Features to be Tested

Verranno testate le funzionalità descritte nei flussi di eventi dei casi d'uso sopra citati, con particolare attenzione a:

- **Validazione degli Input:** Verifica che il sistema accetti solo dati conformi ai vincoli (es. date future per gli eventi, rating tra 1 e 5).
- **Logica di Business:** Verifica delle regole di dominio (es. calcolo dei posti disponibili, transizioni di stato dell'evento).
- **Gestione degli Errori:** Verifica che il sistema sollevi le eccezioni corrette in presenza di condizioni non valide (es. tentativo di iscrizione a evento pieno).
- **Interazioni tra Entità:** Verifica che le associazioni tra oggetti (es. Utente-Evento) vengano create correttamente.

4. Features NOT to be Tested

Considerando la natura di Unit Testing focalizzato sulla logica di business, sono esclusi da questo piano:

- **Interfaccia Utente (GUI)**: Il rendering grafico delle pagine (HTML/CSS) e la User Experience lato browser non sono oggetto di questo piano di test. L'interfaccia delle API (Controller) viene verificata solo funzionalmente tramite Swagger e non tramite test di unità automatizzati.
- **Persistenza Fisica (Database)**: Le interazioni reali con il DBMS (MySQL) saranno simulate tramite tecniche di Mocking. Non verrà testato il funzionamento interno del database o delle query SQL.
- **Requisiti Non Funzionali**: Performance, carico e sicurezza infrastrutturale non sono coperti da questi test unitari.

5. Criteri di Pass/Fail

Un Test Case è considerato **PASSATO** se:

- L'output reale restituito dal metodo sotto test coincide esattamente con l'output atteso (Oracolo).
- Lo stato interno degli oggetti modificati coincide con lo stato atteso.
- Viene sollevata l'eccezione specifica attesa in caso di test di errore.

Un Test Case è considerato **FALLITO** in tutti gli altri casi.

6. Approccio di Testing

L'approccio seguirà rigorosamente i passaggi definiti dal metodo Category-Partition:

1. **Analisi delle Specifiche**: Decomposizione di ogni Caso d'Uso in unità funzionali testabili.
2. **Identificazione delle Categorie**: Individuazione dei parametri di input e delle condizioni ambientali (Environment) che influenzano l'esecuzione.
3. **Partizionamento in Scelte**: Suddivisione di ogni categoria in classi di equivalenza significative (valori validi, valori limite, valori di errore).
4. **Definizione dei Vincoli**: Applicazione di vincoli ([error], [single], [if]) per ridurre il numero di combinazioni ed evitare scenari impossibili.
5. **Generazione Test Frames**: Creazione di scenari astratti di test.
6. **Implementazione Test Cases**: Traduzione dei frame in codice eseguibile.

6.1 Strumenti

- **Linguaggio:** Java
- **Framework di Testing:** JUnit 5
- **Framework di Mocking:** Mockito (utilizzato per isolare i Service dai DAO/Repository).
- **API Testing & Documentation:** SpringDoc OpenAPI (Swagger UI), utilizzato per la verifica manuale degli endpoint.

7. Testing Materials

- Hardware: PC Sviluppatore.
- Software: Java JDK 17+, Eclipse/IntelliJ, JUnit 5 library.

8. Casi di Test

I casi di test specifici, derivati dall'applicazione del metodo Category-Partition, non sono inclusi nel presente documento ma sono dettagliati nel documento **Test Case Specification**.
Il documento TCS contiene:

- L'analisi delle categorie e delle scelte per ogni funzionalità.
- I vincoli identificati.
- Le tabelle complete dei Test Case con input e oracoli (risultati attesi).