1 Introduzione

Il testing è una parte fondamentale del processo di sviluppo di software. L’obiettivo di questa fase è la verifica delle funzionalità, dell’affidabilità e delle performance del software prima di consegnare il prodotto agli utenti finali.

1.1 Tipi di test

- Test whitebox: in questa tipologia, la struttura interna, la progettazione e l'implementazione del software sono note al tester. I tester esaminano la logica del codice, il flusso di dati e il flusso di controllo.
- Test blackbox: in questo caso, il funzionamento interno del software è nascosto al tester. Il tester conosce solo gli input e gli output previsti e verifica se il software si comporta correttamente in base alle sue specifiche.

1.2 Livelli di test

- Test unitari: si concentrano su singole unità o componenti del software.
- Test di integrazione: assicurano che i diversi moduli o servizi del software funzionino insieme come previsto.
- Test di sistema: una fase di test completa che convalida l'intero sistema.
- Test di accettazione: la fase finale di test prima che il software venga rilasciato al cliente.

1.3 Metodi di test

- Test GUI: si concentra sul test dell'interfaccia utente grafica del software per garantire che tutti gli elementi (pulsanti, menu, finestre di dialogo, ecc.) funzionino correttamente e offrano un'esperienza utente fluida.
- Test delle prestazioni: verifica le prestazioni del sistema in diverse condizioni, come test di carico o stress test.

2 Target dei test

2.1 Requisiti funzionali

- Modalità single player: testare che il gioco permetta all'utente di giocare qualunque test
- Sfida giornaliera: testare che la sfida giornaliera cambi ogni giorno e che possa venire giocata una sola volta al giorno
- Homepage: testare il corretto funzionamento di tutti gli elementi della pagina homepage
- Pannello di controllo per insegnanti: testare il corretto funzionamento delle operazioni CRUD e di export/import dei quiz
- Confermare il corretto funzionamento dell'applicazione su computer e sistemi operativi diversi

2.2 Requisiti non funzionali

- Confermare la responsività dell'applicazione. Deve avere minimi ritardi tra l'azione dell'utente e l'esecuzione del sistema.
- Testare l'intuitività dell'interfaccia utente. Deve essere facile da usare per ogni tipo di utente, senza bisogno di un manuale utente.

3 Test schedule

Il codice deve essere eseguito e testato manualmente dopo l'implementazione di ogni funzionalità. Se una parte di codice, in particolare una funzione, non è banale e può essere testata in maniera isolata, è necessario scrivere un test unitario per essa. Al termine di ogni sprint, tutte le principali funzionalità dell'applicazione dovrebbero essere testate per garantire il corretto funzionamento del livello di comunicazione, per verificare che le diverse applicazioni si integrino correttamente e per verificare se la correttezza sia diminuita nel processo di aggiunta di nuove funzionalità.

4 Responsabilità dei test

Ogni sviluppatore dovrebbe testare il proprio codice e assicurarsi che venga compilato ed eseguito senza errori critici che possano causare l'interruzione del programma prima di sottoporlo al controllo di versione. Tutti i test unitari dovrebbero essere superati, oppure nel messaggio di commit dovrebbe essere specificata la causa del mancato superamento e le misure adottate per porvi rimedio. I test di fine sprint dovrebbero essere svolti come una riunione, per dare a tutti gli sviluppatori l'opportunità di analizzare la situazione e le condizioni in caso di errore e di fornire un feedback più completo sui codici di errore generati.

5 Analisi dei rischi

5.1 Requisiti incompleti o ambigui

Requisiti poco chiari o mancanti possono portare a casi di test incompleti, con conseguenti funzionalità non testate o bug trascurati. Funzionalità critiche potrebbero non funzionare come previsto, causando ritardi nei progetti e insoddisfazione del cliente.

5.2 Copertura dei test insufficiente

L'ambito di test limitato a causa di vincoli di tempo o risorse potrebbe lasciare alcune funzionalità non testate. Alcuni bug potrebbero non essere scoperti fino a quando il software non è in produzione.

5.3 Strategia di riduzione dei rischi

- Monitoraggio regolare: i progressi devono essere esaminati regolarmente per identificare tempestivamente eventuali rischi emergenti.
- Approccio di test agile: verrà adottato un approccio di test iterativo, che consentirà un feedback continuo e rapidi aggiustamenti per mitigare i rischi.

6 Test e risultati

ID	Scenario	Prerequisiti	Descrizione	Risultati previsti	Conferma risultato
1	Creazione test	Utente nella scena CRUD dei quiz	L’utente crea un nuovo test, assegna un nome, crea qualche domanda e conferma	Il test viene creato e aggiunto alla lista dei test	✔
2	Creazione test con nome già esistente	Utente nella scena CRUD dei quiz	L’utente prova a creare un nuovo test dandogli un nome già presente nella cartella quiz	Messaggio di errore e impossibilità di completare la creazione	✔
3	Creazione test senza nome	Utente nella scena CRUD dei quiz	L’utente prova a creare un nuovo test senza scrivere un nome	Messaggio di errore e impossibilità di completare la creazione	✔

4	Eliminazione test	Utente nella scena CRUD dei quiz	L’utente clicca il tasto di eliminazione di un test	Il test viene eliminato con successo	✔
5	Modifica test	Utente nella scena CRUD dei quiz	L’utente modifica un test, cambiando le domande o risposte	Il test viene modificato con successo	✔
6	Giocare a quiz	Utente nella scena homepage	L’utente sceglie un quiz e clicca il tasto per iniziarlo	La scena cambia e viene iniziato il test	✔
7	Giocare a daily già completato	Utente nella scena homepage e daily del giorno completato	L’utente clicca il tasto per giocare il daily, quando durante lo stesso giorno lo ha già giocato	Il tasto non è attivo (non si può premere)	✔
8	Risposta cambia colore in base alla correttezza	Utente nella scena quiz	L’utente seleziona una risposta	Il testo della risposta diventa verde se corretta o rosso se sbagliata	✔ ✔