TP
Test Plan

BeYourChoice

Riferimento	BYC_RAD_V_1.5, BYC_SDD_V_1.2, BYC_ODD_V_1.1
Versione	1.1
Data	15/12/2024
Destinatario	Prof.ssa Filomena Ferrucci, Prof. Fabio Palomba
Presentato da	Marco Ciano, Giuseppe D'Avino,Antonio Rapa, Iari Normanno, Rocco Cione, Marco Acierno
Approvato da	Marco Ciano, Giuseppe D'Avino

Revision History

Data	Versione	Descrizione	Autori
18/11/2024	1.0	Prima stesura del documento	Antonio Rapa Iari Normanno Rocco Cione Marco Acierno
19/11/2024	1.0	Revisione e completamento del documento	Antonio Rapa Iari Normanno Rocco Cione Marco Acierno
20/11/2024	1.0	Revisione e del documento	Iari Normanno Rocco Cione Marco Acierno
22/11/2024	1.1	Aggiunta tabella relativa ai possibili rischi	Marco Ciano Giuseppe D'Avino

Sommario

1 Introduzione

BeYourChoice mira a creare un ambiente di apprendimento dinamico e partecipativo, dove studenti e docenti possono interagire attivamente, sfruttando la tecnologia, per approfondire la comprensione dei principi democratici e delle istituzioni politiche. Il documento di Test Plan si occupa di analizzare le attività di Testing effettuate nel tempo per garantire il corretto funzionamento della piattaforma.

Nel documento sono descritte le strategie di testing adottate, le funzionalità soggette a verifica, e gli strumenti selezionati per individuare eventuali errori. L'obiettivo è fornire una piattaforma priva di malfunzionamenti.

Considereremo solo alcune delle funzionalità aventi requisiti con priorità alta presenti all'interno di ciascuna macro categoria:

- Gestione autenticazione e registrazione;
- Gestione profilo;
- Gestione classe virtuale;
- Gestione materiale didattico;
- Gestione e generazione domande quiz;
- Creazione scenario virtuale;

2 Documenti correlati

Il presente documento è strettamente collegato ai materiali prodotti fino al rilascio della versione 1.0(ovvero RAD, SDD, ODD) e manterrà tale connessione anche con i documenti che saranno sviluppati e pubblicati in futuro(ovvero Category partition, Test Case Specification, Test Summary Report). Pertanto, sarà soggetto a revisioni e aggiornamenti nel tempo. I test case saranno sviluppati sulla base delle funzionalità individuate nel documento di raccolta e analisi dei requisiti.

2.1 Collegamento con il documento di raccolta e analisi dei requisiti (RAD)

Il collegamento tra il Test Plan (TP) e il RAD riguarda sia i requisiti funzionali che quelli non funzionali del sistema. Il RAD fornisce una descrizione dettagliata delle funzionalità, specificando anche il livello di priorità.

2.2 Collegamento con il System Design Document (SDD)

Nel System Design Document (SDD) è descritta l'architettura del sistema basata sul modello MVC, insieme alla struttura dei dati e ai servizi forniti dai vari sottosistemi.

2.3 Collegamento con l'Object Design Document (ODD)

L'Object Design Document (ODD), conterrà informazioni sui package e sulle classi che compongono il sistema.

3 Panoramica del Sistema

Il sistema proposto consiste in un'applicazione web, accessibile dagli studenti e dai docenti, che mira a facilitare l'insegnamento e l'apprendimento dell'educazione civica negli istituti di scuola superiore in Italia. Gli utenti della nostra applicazione saranno: Studente e Docente.

Tutti gli utenti dovranno poter registrarsi e autenticarsi alla piattaforma tramite credenziali personali. Una volta effettuato il login saranno disponibili diverse funzionalità a secondo del tipo di utente. Il docente potrà gestire classi virtuali che avranno come funzionalità: la

creazione e l'inserimento manuale degli studenti, oltre alla possibilità di caricare materiale didattico. I docenti potranno inoltre gestire il materiale didattico, e avranno a disposizione strumenti per creare quiz personalizzati e scenari interattivi. Questi scenari permettono agli studenti di assumere ruoli specifici e di partecipare a simulazioni di eventi. Gli studenti, a loro volta, potranno accedere al materiale didattico caricato dai docenti, partecipare attivamente a quiz e scenari simulativi, e monitorare i propri progressi tramite una dashboard dedicata. Questa dashboard offre agli studenti un quadro complessivo delle loro attività e dei risultati ottenuti mentre, per il docente offre un quadro generale dell'andamento di ogni sua singola classe. Gli utenti della piattaforma potranno inoltre comunicare attraverso una chat integrata, per scambiarsi messaggi in tempo reale.

Il sistema proposto si basa sull'architettura MVC (Model-View-Controller), implementata utilizzando Python. Questa scelta risulta particolarmente adatta alle nostre necessità, poiché la separazione delle componenti tra modello, vista e controllore offre numerosi vantaggi, tra cui:

- **Modularità**: suddivide il codice in blocchi indipendenti, facilitando la gestione e l'organizzazione del progetto.
- **Manutenibilità**: il codice è facilmente aggiornabile e gestibile, consentendo modifiche rapide a ciascuna componente senza influenzare le altre.
- **Testabilità**: separando la logica di business (Model) dalla logica di presentazione (View), è possibile testare singolarmente ogni componente, migliorando l'affidabilità del sistema.
- **Estendibilità**: l'architettura facilita l'aggiunta di nuove funzionalità senza compromettere il funzionamento delle componenti esistenti.

Per la logica di business e gestione dei dati sarà utilizzato **Python**, con il supporto del framework **Flask** per la gestione del controllore e delle interazioni utente.

Per migliorare l'esperienza utente e rendere l'interfaccia grafica della piattaforma più dinamica e interattiva, sarà integrata la tecnologia **JavaScript**. Questo linguaggio verrà utilizzato per implementare funzionalità avanzate come l'aggiornamento in tempo reale dei contenuti, la gestione degli eventi dell'utente e l'ottimizzazione delle transizioni visive, contribuendo a creare un'interfaccia fluida e intuitiva.

Per la gestione del database sarà utilizzato **MongoDB**, un database NoSQL flessibile e scalabile, perfetto per gestire dati non strutturati e facilmente integrabile con Python.

Per la generazione dei commenti all'interno dei file che comporranno l'implementazione della piattaforma, verrà utilizzato **Qodo** come strumento per la gestione delle docstring.

Per la logica di presentazione verranno utilizzati **HTML** e **CSS**. L'HTML sarà impiegato per strutturare le pagine e i contenuti, il CSS garantirà uno stile visivo chiaro e piacevole.

4 Possibili Rischi

Qui di seguito vengono riportati i rischi che hanno una media o alta probabilità e che potrebbero presentare un maggiore impatto, causando il ritardo della consegna o il fallimento del progetto.

ID	Rischio	Categoria	Rank	Impatto
R1	Non rispetto delle scadenze	Organizzativi	0,56	Catastrofico
R2	Poca comunicazione tra TM	Persone	0,16	Catastrofico
R5	Problemi di integrazione tra software e hardware	Tecnologie	0,35	Catastrofico
R11	Mancanza di skill tecniche adeguate nel team	Persone	0,24	Catastrofico
R12	Deadline fissate ottimistiche.	Stime	0,24	Catastrofico
R20	Eccessivo carico di lavoro sui membri del team.	Persone	0,48	Catastrofico

R25	Il team fatica a tradurre i requisiti in soluzioni concrete.	Requisiti	0,21	Catastrofico
R17	Modifiche ai requisiti durante lo sviluppo.	Requisiti	0,15	Catastrofico
R24	I membri del team perdono motivazione, riducendo produttività e qualità del lavoro.	Persone	0,56	Catastrofico

5 Funzionalità da testare

Come indicato in precedenza, verranno testati esclusivamente i requisiti con priorità alta. Di seguito è riportato l'elenco dei requisiti da testare, suddivisi per ciascuna gestione.

- Gestione autenticazione e registrazione
  - Login
  - RegistrazioneStudente
  - RegistrazioneDocente
- Gestione profilo
  - ModificaProfilo
  - ModificaPassword
- Gestione classe virtuale
  - CreazioneClasseVirtuale
  - InserimentoClasseStudente
- Gestione materiale didattico
  - CaricamentoMateriale
  - RimozioneMateriale
- Gestione e generazione domande quiz
  - CreazioneQuiz
  - EsecuzioneModulo
- Creazione scenario virtuale
  - CreazioneScenario

Le attività di testing non coinvolgeranno le funzionalità correlate a requisiti funzionali di bassa e media priorità.

6 Approccio

Il testing dell'intero sistema si concentrerà esclusivamente sul testing di unità che rappresenta la base fondamentale per garantire la correttezza e l'affidabilità di ogni singolo componente. Sebbene non vengano eseguite ulteriori fasi di testing, come il testing di integrazione o di sistema, il nostro obiettivo è verificare approfonditamente le funzionalità elementari del codice, assicurandoci che ogni unità funzioni correttamente e in isolamento.

La progettazione dei casi di test di unità verrà sviluppata parallelamente alla fase di implementazione del sistema, utilizzando un approccio incrementale che permetta di verificare ogni nuova funzionalità appena implementata. Adotteremo un approccio sandwich, combinando tecniche top-down e bottom-up. Questo ci permetterà di testare contemporaneamente i componenti superiori(come interfacce utente) e quelli inferiori (come le singole funzioni e metodi). Questo processo sarà documentato nella presente sezione e perfezionato in fase di esecuzione. Durante il testing, ci concentreremo esclusivamente sul comportamento esterno del sistema, verificando che gli input forniti producano gli output attesi, senza entrare nei dettagli dell'implementazione interna. Utilizzeremo tecniche di partizione di equivalenza e analisi dei valori limite per progettare i test, assicurandoci che tutte le condizioni, comprese quelle limite e di errore, vengano verificate.

6.1 Testing di unità

Il testing di unità rappresenta una fase cruciale per garantire correttezza e affidabilità di ogni singola funzione e metodo sviluppato. Per sviluppare i vari test useremo un approccio black-box unito al metodo sandwich, ci concentreremo quindi singolarmente sui vari componenti del sistema trattandole come "scatole nere", verificando che, da un determinato input, venga prodotto l'output atteso. Per progettare i casi di test, utilizzeremo tecniche come la partizione di equivalenza e l'analisi dei valori limite, che ci permette di esplorare le possibili condizioni operative più significative. Con il metodo sandwich, possiamo testare le unità in modo incrementale, iniziando dalle funzionalità principali e proseguendo verso funzionalità più dettagliate.

7 Criteri di Pass/Fail

Le attività di testing sono mirate a identificare la presenza di faults all'interno del sistema, procedendo con la risoluzione degli stessi.

L'esito di un test case è valutato mediante un oracolo, inteso come il risultato atteso della sua esecuzione, basandosi sui requisiti.

Un test fallisce se, dato un input al sistema, l'output ottenuto è diverso dall'output atteso dall'oracolo.

Un test ha successo se, dato un input al sistema, l'output ottenuto è uguale all'output atteso dall'oracolo.

8 Criteri di sospensione e ripresa

In questa sezione verranno specificati i criteri di sospensione del test e le attività di verifica che dovranno essere ripetute una volta risolti i problemi, prima di riprendere il processo di testing.

Criteri di sospensione

Il processo di testing proseguirà fino alla sua completa conclusione, anche in caso di rilevamento di una failure nel sistema. Il testing potrà essere temporaneamente interrotto esclusivamente nel caso in cui, venga riscontrato un errore nella configurazione o nella definizione di uno o più test, rendendo impossibile la corretta esecuzione.

Criteri di ripristino

Il testing riprenderà una volta che i fault individuati saranno stati corretti e validati, garantendo che le modifiche apportate non abbiano introdotto nuovi problemi e che il sistema sia pronto per proseguire con le attività di verifica pianificate.

9 Test Deliverables

I documenti rilasciati durante e al termine della fase di test:

- Test Plan;
- Category Partition;
- Test Case Specification;
- Test Summary Report;

10 Responsabilità

Ogni team member sarà responsabile del testing della gestione alla quale è stato assegnato. I team members verranno divisi in due sottogruppi che si occuperanno rispettivamente dello sviluppo del lato back-end e del lato front-end.
Ogni team member dovrà effettuare il testing di unità, tramite category partition, dei casi d'uso specificati.

11 Glossario

- **Black-Box Testing:** Tipo di test che verifica la funzionalità di un sistema senza considerare l'implementazione interna, basandosi esclusivamente sugli input e sugli output.
- **Testing:**: Processo o metodo per trovare errori in un'applicazione o un programma software in modo che l'applicazione funzioni in base ai requisiti dell'utente finale;
- **Deliverables:** Documenti o artefatti generati durante la fase di testing, come il Test Plan, Test Case Specification, Test Incident Report e Test Summary Report.
- **Oracolo del Test:** Fonte o meccanismo che specifica i risultati attesi di un test case.
- **Sandwich Testing:** Approccio al testing che combina tecniche top-down e bottom-up, permettendo di testare contemporaneamente componenti di alto livello e di basso livello.