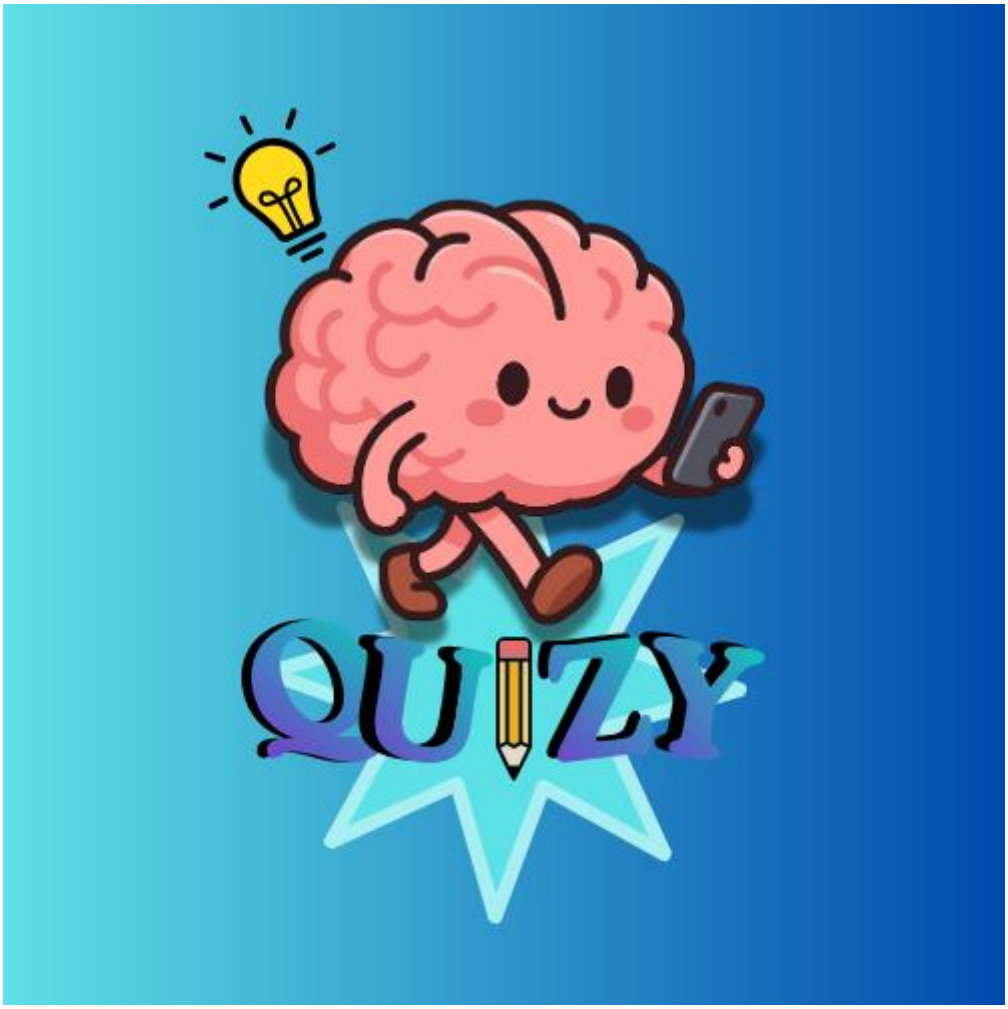
# Università degli Studi di Salerno

**Corso di Ingegneria del Software**

# Quizy

# Test Plan

# Versione 0.4




**Data: 22/12/2025**

| | |
|---|---|
| Progetto: Quizy | Versione: 0.4 |
| Documento: Test Plan | Data: 21/12/2025 |

## Coordinatore del progetto

| **Nome** | **Matricola** |
|---|---|
| Davide Nino Longobardi | 0512119971 |

## Partecipanti

| **Nome** | **Matricola** |
|---|---|
| Roberto Caiazza | 0512116335 |
| | |

| **Scritto da:** | Roberto Caiazza |
|---|---|

## Revision History

| **Data** | **Versione** | **Descrizione** | **Autore** |
|---|---|---|---|
| 20/12/2025 | 0.0 | Stesura dei primi 4 punti del documento | Davide Nino Longobardi, Roberto Caiazza |
| 21/12/2025 | 0.1 | Stesura completa del documento | Davide Nino Documento, Roberto Caiazza |
| 17/12/2025 | 0.2 | Aggiunta del punto 9 | Davide Nino Longobardi, Roberto Caiazza |
| 03/02/2026 | 0.3 | Revisione del documento | Davide Nino Longobardi, Roberto Caiazza |
| 08/02/2026 | 0.4 | Riscritture del punto 8 e Revisioni finale del documento | Davide Nino Longobardi, Roberto Caiazza |

# 1.INTRODUZIONE

Il documento viene redatto per fornire una panoramica sui test che vengono effettuati sull'applicazione Quizy. L'obiettivo di questo documento è definire l'ambito, l'approccio, le risorse e la pianificazione delle attività di testing necessarie per validare il sistema. Questo piano è stato redatto per fornire un framework operativo condiviso tra sviluppatori e tester, assicurando che le verifiche siano eseguite in modo sistematico, efficiente ed economico.

# 2.RELATIONSHIP TO OTHER DOCUMENTS

• Requirements Analysis Document (RAD)

• System Design Document (SDD)

• Object Design Document (ODD)

# 3.SYSTEM OVERVIEW

Questa sezione identifica i componenti software oggetto di verifica durante le fasi di unit testing e integration testing. L'unità minima di test corrisponde alle classi contenute nei seguenti sottosistemi logici, organizzati per livello e la persistenza dei dati.

**Interface Layer**

Rappresenta il front-end dell'applicazione installato sui dispositivi mobili. La componente general_ineterface(UI) gestisce le Activity Android e la cattura degli eventi utente. Il test e la corretta visualizzazione delle schermate.

**Application Layer**

Costituisce il nucleo della logica di business ed è composto da tre elementi principali:

- **AutanticateMenager** Gestisce l'intero ciclo di vita dell'account utente. Questo componente è responsabile delle procedure di autenticazione (autenticate), della registrazione di nuovi utenti (registra), del logout e del cambio password (newPassword).
- **QuizCreatorMenager** Rappresenta il componente dedicato ai creatori di contenuti. Si occupa della logica relativa alla creazione (createQuiz), modifica (aggiornaQuiz) ed eliminazione (deleteQuiz) dei quiz. Gestisce inoltre l'elevazione dei privilegi dell'utente a ruolo "creatore" (upUserRole).
- **QuizUserMenager** Gestisce l'interazione dell'utente standard con i quiz. Questo componente permette di recuperare la lista dei quiz disponibili (getQuizzes), di avviare una sessione di quiz (con o senza password) tramite startQuiz, e di calcolare il punteggio finale al termine dello svolgimento (completaQuiz).

**Persistence Layer**

Livello responsabilità della memorizzazione permanente dai dati. Comprende persistenceManage, gestisce il mapping relazione verso il DBMS MySQL e garantisce l'integrità delle transazioni.

# 4.FEATURES TO BE TESTED

L'approccio di test adottato per **Quizy** è incrementale e segue la classica piramide dei test. La strategia è stata guidata dalle priorità dei requisiti definite nel RAD: abbiamo concentrato la maggior copertura di test sulle funzionalità ad Alta Priorità (Core Business: Login, Creazione e Svolgimento Quiz), garantendo per queste la massima robustezza.

A livello di funzionalità, gli aspetti del sistema che verranno testati sono.

- Registrazione Utente
- Creazione Quiz
- Completa Quiz
- Cambio Ruolo

# 5.PASS/FAIL CRITERIA

Un test viene considerato Passed quando non si sono riscontrate failure durante l'esecuzione, mentre viene considerato Failed quando sono state riscontrate una o più failure.

# 6.APPROACH

L’approccio del testing si divide in:

**Testing di unità:** verrà utilizzata la tecnica di “Black-Box” testing. Si focalizza sul comportamento di I/O, senza preoccuparsi della struttura interna della componente. Se per ogni dato input siamo in grado di prevedere l’output, allora l’unità supera il test. Riduciamo il numero di casi di test effettuando una partizione dividendo le condizioni di input in classi di equivalenza e si scelgono i test case per ogni classe di equivalenza.

**Testing di integrazione**: si utilizza la strategia bottom-up. I sottosistemi al livello più basso della gerarchia sono testati individualmente. I successivi sottosistemi ad essere testati sono quelli che chiamano i sottosistemi testati in precedenza.

**Testing di sistema:** è l’ultimo testing prima della messa in uso del sistema e prevede il controllo delle funzionalità del sistema secondo i requisiti specificati.

# 7.SUSPENSION AND RESUMPTION

La fase di testing sarà sospesa quando si otterranno i risultati attesi, rispettando però i tempi di consegna del progetto. La fase di testing sarà ripresa se si effettueranno modifiche al sistema e verranno rieseguiti di nuovo

i casi di test.

# 8.TESTING MATERIALS

Per garantire l'efficienza e la ripetibilità delle verifiche, sono stati adottati i seguenti strumenti di automazione e supporto:

**Backend (Server - Java):**

- **JUnit 5:** Framework di riferimento per la definizione e l'esecuzione dei test di unità e integrazione.
- **Mockito:** Utilizzato nei test di unità per simulare le dipendenze (mocking) e isolare la logica di business dai componenti esterni.
- **JerseyTest:** Framework specifico per testare le API RESTful in un ambiente controllato, verificando le risposte HTTP senza dover avviare l'intero server Tomcat.

- **H2 Database:** Database in-memory leggero, utilizzato durante i test di integrazione per simulare le transazioni dati senza intaccare il database di produzione (MySQL).

**Frontend (Client - Android):**

- **Maestro:** Framework di automazione UI utilizzato per i Test di Sistema (End-to-End). Permette di simulare i flussi utente reali (tap, scroll, inserimento testo) sull'applicazione Android.

# 9.TEST CASES

UC 1 Registrazione

UC 2 Login

UC 3 Partecipazione Quiz

UC 4 Creazione quiz

# 10.TESTING SCHEDULE

- Responsabilit: Il Team Lead supervisiona; gli sviluppatori eseguono Unit, Integration e System Test.
- Rischi: Ritardi nell'implementazione API Restful potrebbero far slittare i test di integrazione frontend.
- Pianificazione: Settimana 1-2 (Unit Testing), Settimana 3 (Integration Testing), Settimana 4 (System e Regression Testing).