**Capitolo 3**

**Testing**

1. **Panoramica ai test**

In questo capitolo verrà trattato l’approccio che è stato utilizzato durante la fase di testing dell’applicazione, le difficoltà incontrate durante questa fase e i diversi test effettuati. La fase di testing rappresenta un passaggio cruciale nel ciclo di sviluppo del software, essenziale per garantire che l'applicazione sia stabile, funzionale e risponda ai requisiti degli utenti.

Gli obiettivi principali del testing sono:

* **Verificare la Correttezza Funzionale**: Assicurare che ogni funzione e componente dell'applicazione esegua correttamente il proprio compito, identificando e correggendo eventuali errori logici o di implementazione.
* **Garantire la Stabilità**: Testare l'applicazione in vari scenari e condizioni per assicurare che sia stabile ed evitare crash inaspettati.
* **Migliorare le Prestazioni**: Identificare e risolvere i problemi di performance per garantire un'esperienza utente fluida, anche sotto carico.
* **Assicurare l'Usabilità**: Raccogliere feedback dagli utenti per migliorare l'interfaccia e l'esperienza utente complessiva.
* **Valutare l'Interoperabilità**: Verificare che l'applicazione funzioni correttamente su diverse piattaforme e dispositivi.

1. **Strategia di testing:**

La strategia di testing ha compreso tre diverse tipologie di test, ognuna mirata a verificare specifici aspetti e funzionalità dell'applicazione [1]:

**Test di unità:**

* **Descrizione:** I test di unità sono stati utilizzati per verificare la correttezza delle singole unità di codice, come funzioni e metodi isolati.
* **Obiettivo:** Garantire che ogni singola parte del codice funzioni correttamente e produca i risultati attesi senza errori.

**Test di widget:**

* **Descrizione:** I test sui widget sono stati utilizzati per verificare il comportamento dei componenti dell'interfaccia utente (UI) in isolamento.
* **Obiettivo:** Assicurare che ogni componente UI risponda correttamente agli input e che la sua interfaccia utente sia costruita correttamente secondo le specifiche.

**Test di integrazione:**

* **Descrizione:** I test di integrazione sono stati utilizzati per verificare che diversi moduli e componenti dell'applicazione funzionino correttamente insieme.
* **Obiettivo:** Assicurare che l'interazione tra i vari moduli e componenti sia conforme alle specifiche e che l'applicazione nel suo complesso funzioni come previsto quando i vari componenti sono combinati.

1. **Strumenti e Framework Utilizzati**

Nella realizzazione dei test sono stati utilizzati principalmente i seguenti strumenti e framework:

* **Flutter Test Framework:** Framework di testing fornito da Flutter per eseguire test su widget e integrazione.
* **Mockito:** Libreria per creare mock[[1]](#footnote-1) e stub[[2]](#footnote-2) necessari per i test, facilitando la simulazione di comportamenti e risposte.
* **Firebase Auth Mocks:** Libreria specifica per creare mock di Firebase, i quali permettono di simulare le funzionalità di autenticazione, permettendo di testare la gestione degli utenti senza connessione a un effettivo database.

1. **Descrizione dei Test Effettuati**

In questa sezione verranno descritti i test effettuati per garantire la correttezza e la funzionalità dell'applicazione. I test sono stati suddivisi in due categorie principali: test di registrazione e test di login.

* **Test di Registrazione:**
  + **Registrazione con Successo**
    - **Descrizione:** Questo test verifica che un utente possa registrarsi correttamente utilizzando un’e-mail e una password valide.
    - **Scenario:** Viene fornita un’e-mail valida (test@example.com) e una password valida (password123).
    - **Risultato atteso:** Il metodo *createUserWithEmailAndPassword* restituisce un oggetto *UserCredential*, e la funzione *registration* restituisce il messaggio "Success".
  + **Password Debole:**
    - **Descrizione:** Questo test verifica che il sistema gestisca correttamente il caso in cui la password fornita sia troppo debole.
    - **Scenario:** Viene fornita un’e-mail valida (test@example.com) e una password debole (123).
    - **Risultato atteso:** Il metodo *createUserWithEmailAndPassword* lancia un'eccezione *FirebaseAuthException* con associato il codice *weak-password,* e la funzione *registration* restituisce il messaggio "The password provided is too weak."
  + **E-mail già in Uso:**
    - **Descrizione:** Questo test verifica che il sistema gestisca correttamente il caso in cui l’e-mail fornita sia già associata ad un account esistente.
    - **Scenario:** Viene fornita un’e-mail già in uso (test@example.com) e una password valida (password123).
    - **Risultato atteso:** Il metodo *createUserWithEmailAndPassword* lancia un'eccezione *FirebaseAuthException* con associato il codice *email-already-in-use,* e la funzione *registration* restituisce il messaggio "The account already exists for that email."
  + **Errore Generico:**
    - **Descrizione:** Questo test verifica che il sistema gestisca correttamente eventuali errori generici che possono verificarsi durante la registrazione.
    - **Scenario:** Viene fornita un'email valida (test@example.com) e una password valida (password123).
    - **Risultato atteso:** Il metodo *createUserWithEmailAndPassword* lancia un'eccezione *FirebaseAuthException* con un codice e un messaggio generico associato, e la funzione *registration* restituisce il messaggio corrispondente.
  + **Errore Non di FirebaseAuth:**
    - **Descrizione:** Questo test verifica che il sistema gestisca correttamente errori che non sono specifici di *FirebaseAuthException*.
    - **Scenario:** Viene fornita un’e-mail valida (test@example.com) e una password valida (password123).
    - **Risultato atteso:** Il metodo *createUserWithEmailAndPassword* lancia una generica eccezione *Exception,* e la funzione *registration* restituisce il messaggio corrispondente.
* **Test di login:**
  + **Login con Credenziali Valide:**
    - **Descrizione:** Questo test verifica che un utente possa effettuare il login correttamente utilizzando un’e-mail e una password valide.
    - **Scenario:** Viene fornita un’e-mail valida (1@2.com) e una password valida (123456).
    - **Risultato Atteso:** Il metodo *signInWithEmailAndPassword* restituisce un oggetto *UserCredential,* e la funzione login restituisce il messaggio "Success".
  + **Login con Credenziali Errate:**
    - **Descrizione:** Questo test verifica che il sistema gestisca correttamente il caso in cui le credenziali fornite siano errate.
    - **Scenario:** Viene fornita un’e-mail non valida (invalid@example.com) e una password non valida (invalidpassword).
    - **Risultato Atteso:** Il metodo *signInWithEmailAndPassword* lancia un'eccezione *FirebaseAuthException* con associato il codice *wrong-password*, e la funzione login restituisce il messaggio "Wrong password provided for that user."

Questi test sono stati effettuati per assicurare che le funzionalità critiche di registrazione e login dell'applicazione funzionino correttamente, gestendo sia i casi di successo sia i possibili errori che possono verificarsi.

1. **Sfide e Soluzioni**

Durante il processo di testing, abbiamo incontrato diverse sfide, tra cui la mancanza di una documentazione completa per l'integrazione con Firebase Database. Questa carenza ha reso difficile eseguire test di integrazione accurati per alcune funzionalità chiave dell'applicazione. Inoltre, nonostante l'utilizzo di modelli di linguaggio per ottenere assistenza, le informazioni fornite non sono state sufficienti per risolvere i problemi riscontrati.

Queste problematiche saranno esplorate nei capitoli successivi, dove verrà analizzato il valore aggiunto derivante dall'utilizzo di un LLM come ChatGPT nello sviluppo del progetto. Nonostante l'incapacità di ChatGPT di eseguire test funzionanti per la nostra applicazione, questa informazione è comunque significativa, dato che il nostro obiettivo principale è comprendere l'utilità di un LLM nel contesto dello sviluppo di un'app mobile.

**Bibliografia capitolo**

[1] Testing. *Testing Flutter apps.* Url: <https://docs.flutter.dev/testing/overview>

1. Oggetto simulato che replica il comportamento di un oggetto reale in modo controllato. [↑](#footnote-ref-1)
2. Tipo di oggetto simulato utilizzato nei test unitari e di integrazione per sostituire componenti o dipendenze reali di un sistema con versioni semplificate che forniscono risposte predefinite. [↑](#footnote-ref-2)