REPORT.IT

ITALIAN DISCRIMINATION COMMUNITY

Test Plan

Report.it

Riferimento	
Versione	1
Data	10/07/2024
Presentato da	Simona Grieco, Maria Concetta Schiavone

Sommario

1. Introduzione

L'attività di testing è una componente fondamentale nello sviluppo di un prodotto software efficiente e funzionale perché permette di rilevare possibili errori all'interno del sistema, che potrebbero verificarsi in seguito alla scrittura del codice sorgente.

In questo documento vengono pianificate e descritte in modo dettagliato le attività di testing da effettuare relativamente alle change request.

2. Funzionalità da testare

Le funzionalità da testare solo le seguenti:

CR1 – Creazione nuovo utente
Descrizione
L'implementazione di questa CR ha come obiettivo la creazione della nuova entità "Amministratore" che va a creare una nuova prospettiva al sistema, quella diagnostica. Questo nuovo utente potrà visualizzare una dashboard dedicata che mostri statistiche generali in merito alle denunce inoltrate in tutta Italia e suddivise per regione geografica.
Funzionalità da testare
Login standard utente Amministratore

CR2 – Ottimizzazione denuncia di discriminazione
Descrizione
L'obiettivo della seguente CR è quello di migliorare la sezione delle denunce di una discriminazione, in quanto attualmente l'utente durante la compilazione del modulo di denuncia può solamente allegare prove in formato testuale. Pertanto, si è pensato di proporre un sistema più dettagliato per consentire all'utente di segnalare e tracciare le discriminazioni subite, fornendogli la possibilità di caricare prove anche attraverso foto, video o documenti.
Funzionalità da testare
Intera sezione "Inoltro denuncia"

CR4 – Sezione discussione in tendenza
Descrizione
Lo sviluppo della seguente CR mira a potenziare il sistema di voto per i post della community, rendendo più facile per gli utenti individuare i contenuti più rilevanti e popolari grazie a una sezione dedicata.
Funzionalità da testare
Ordinamento delle discussioni in base al punteggio

3. Approccio

Le tecniche di testing utilizzate si basano sull'approccio di tipo:

- BlackBox, per definire i test cases utilizzando la tecnica del **Category Partition**, specificata nel documento TDS_Report.it.
- WhiteBox, per il testing che necessita di conoscere i dettagli implementativi.

3.1. Unit Testing

L'unit testing riguarda la verifica delle singole unità di codice (funzioni, metodi, classi) per assicurarsi che funzionino correttamente in isolamento. Viene utilizzato per identificare bug a livello di codice sorgente e per verificare che ogni unità esegua correttamente le operazioni per cui è stata progettata.

3.2. Integration Testing

L'Integration Testing ha l'obiettivo di verificare l'interazione tra le nuove componenti e quelle preesistenti, concentrandosi sull'integrazione tra varie unità del sistema per assicurarsi che lavorino correttamente insieme.

Questo tipo di test identifica problemi nell'interazione tra componenti integrate.

3.3. System Testing

Il system testing è una fase di verifica completa del sistema nel suo complesso, eseguendo test end-to-end per assicurarsi che l'intero sistema soddisfi i requisiti specificati.

3.4. Regression Testing

Il regression testing garantisce che le modifiche al codice (correzioni di bug, nuove funzionalità) non abbiano introdotto nuovi difetti nel software esistente. Viene eseguito ripetutamente durante lo sviluppo.

L'approccio del testing di regressione sarà basato sui test di unità e di integrazione che saranno sviluppati sul sistema originale. In questo modo, si garantirà che le modifiche apportate al sistema o alle sue componenti non introdurranno nuovi bug o errori.

4. Strumenti per il testing

Gli strumenti per il testing che si adoperano sono i seguenti:

- **mockito.dart:** Libreria di Dart che permette di creare oggetti finti (mocks) per i test. Viene utilizzata principalmente per simulare il comportamento di oggetti complessi o dipendenze esterne in modo che possano essere testati senza effettivamente eseguire il loro codice reale.
- **firebase_auth_mock.dart**: Libreria che fornisce mock per il pacchetto firebase_auth di Flutter, permettendo di testare le funzionalità di autenticazione senza doversi connettere a Firebase.
- **flutter_test.dart**: Pacchetto di Flutter che fornisce una serie di strumenti e utilità per il testing delle applicazioni Flutter. Include funzionalità per scrivere test widget, test unitari e test di integrazione.
- **integration_test.dart**: Un pacchetto Flutter specifico per l'esecuzione di test d'integrazione. Questi test verificano il comportamento dell'applicazione intera, inclusi i widget, i servizi back-end e altre dipendenze.
  Tale pacchetto è pensato per eseguire test end-to-end sull'applicazione. Questo significa che può essere utilizzato non solo per testare l'integrazione di vari componenti

dell'app, ma anche per convalidare il comportamento complessivo dell'applicazione, il che include i test di sistema.

5. Criteri Pass/Fail

Lo scopo del testing è quello di dimostrare la presenza di fault. La fase di test avrà successo se individuerà una failure, ossia se l'output osservato sarà diverso da quello atteso. Al contrario, il testing fallirà se l'output osservato sarà uguale all'oracolo.