Università degli Studi di Salerno
Corso di Ingegneria del Software

FakeBuster
Test Plan
Versione 1.0

Data: 14/12/2025

Progetto: FakeBuster	Versione: 1.0
Test Plan	Data: 14/12/2025

Coordinatore del progetto:

Nome	**Matricola**
Bruno Santo	0512116161
Emiliano Di Giuseppe	0512119155

Partecipanti:

Nome	**Matricola**
Bruno Santo	0512116161
Emiliano Di Giuseppe	0512119155

Scritto da:	Bruno Santo & Emiliano Di Giuseppe

Revision History

Data	Versione	Descrizione	Autore
14/12/2025	1.0	Creazione Test Plan	Bruno Santo & Emiliano Di Giuseppe

Indice

1. Introduction 4
   1.1. Overview 4
2. Relationship to other documents 4
   2.1. Relazioni con il Requirement Analysis Document(RAD) 4
   2.2. Relazioni con il System Design Document(SDD) 5
   2.3. Relazioni con l'Object Design Document(ODD) 5
3. System overview 5
4. Feature to be tested/not to be tested 5
5. Pass/Fail criteria 6
6. Approach 6
   6.1. Testing di unità 6
   6.2. Testing d'integrazione 6
   6.3. Testing di sistema 6
7. Suspension and resumption 6
8. Testing materials (hardware/software requirements) 7
9. Test cases 7
   9.1. Gestione Utenza 7
      9.1.1. TC_UTE_1: Registrazione Utente 7
   9.2. Gestione Pubblicazioni 9
      9.2.1. TC_PUB_1: Inserimento Nuovo Post 9
   9.3. Gestione Appelli 10
      9.3.1. TC_APP_1: Creazione Appello 10
   9.4. Gestione Segnalazioni 12
      9.4.1. TC_SEG_1: Invio Segnalazione 12
10. Testing schedule 13

1. Introduction

FakeBuster Social nasce come piattaforma sperimentale che unisce intelligenza artificiale e partecipazione umana per creare un ecosistema informativo più affidabile.

1.1. Overview

FakeBuster Social propone un nuovo paradigma di social network, in cui la pubblicazione di una notizia è subordinata a una valutazione automatica di attendibilità effettuata da un modello di intelligenza artificiale.
Il sistema integra inoltre una supervisione umana (fact-checker) che analizza eventuali ricorsi o segnalazioni degli user

Funzionalità principali:

- Analisi automatica del testo prima della pubblicazione.
- Classificazione dei contenuti in *attendibili*, *sospetti* o *falsi*.
- Pannello dedicato ai fact-checker per la revisione e l'addestramento continuo del modello.

2. Relationship to other documents

Questo documento è strettamente collegato ai documenti prodotti fino ad ora: la gran parte delle funzioni previste da implementare sono già state specificate nei documenti precedenti.

2.1. Relazioni con il Requirement Analysis Document(RAD)

I test case pianificati nel Test Plan sono relazionati ai requisiti funzionali (UC1 Pubblicazione, UC2 Appello, UC6 Registrazione), agli scenari e ai mock-up definiti nel RAD .

2.2. Relazioni con il System Design Document(SDD)

I test case pianificati nel Test Plan verificano la corretta integrazione dei sottosistemi (Gestione Pubblicazioni, AI, Persistenza) definiti nell'architettura MVC del SDD .

2.3. Relazioni con l'Object Design Document(ODD)

I test di unità faranno riferimento alle classi (Model) e ai moduli (Controller/Blueprints) specificati nell'ODD, verificando il rispetto delle invarianti e delle pre/post-condizioni .

3. System overview

FakeBuster Social è una piattaforma web progettata per contrastare la disinformazione mediante un approccio preventivo. L'architettura del sistema è basata sul pattern **MVC (Model-View-Controller)** e implementata utilizzando un backend **Python/Flask** e un database relazionale **MySQL** per la persistenza dei dati.
Il cuore del sistema è l'interazione tra il modulo di gestione delle pubblicazioni e il servizio esterno **AIService**, che analizza in tempo reale il testo dei post inviati dagli utenti (User). In base allo score di attendibilità restituito dall'IA, il sistema determina automaticamente lo stato del post (Pubblicato o Bloccato). Il sistema include inoltre interfacce dedicate per la moderazione umana (Fact-checker), permettendo la gestione di Appelli e Segnalazioni in caso di falsi positivi o negativi.

4. Feature to be tested/not to be tested

Di seguito la lista delle features di cui si effettuerà il testing per le varie gestioni:

Sotto sistema	Funzionalità	Riferimento Requisito (RAD)
Gestione Utenza	Registrazione di un nuovo utente con validazione input.	UC6 - Registrazione
Gestione Pubblicazioni	Inserimento di un nuovo post e verifica blocco/pubblicazione in base allo score IA.	UC1 - Pubblicazione notizia
Gestione Appelli	Creazione di un appello per un post bloccato.	UC2 - Appello contenuto bloccato
Gestione Segnalazioni	Invio di una segnalazione per un post pubblico.	UC3 - Segnalazione post

Features da non testare:

- Tutti i Requisiti Funzionali non presenti nella tabella sovrastante.
- L'accuratezza intrinseca del modello di Intelligenza Artificiale (la verifica della qualità dello score è esterna al test funzionale del sistema software).
- Aspetti puramente grafici (CSS) che non influenzano la logica.

5. Pass/Fail criteria

Le attività di testing sono mirate ad identificare la presenza di faults (errori) all'interno del sistema. L'esito di un test case è valutato mediante un oracolo, inteso come il risultato atteso della sua esecuzione, basandosi sui requisiti:

- **Pass:** Un test ha successo se, dato un input al sistema, l'output ottenuto coincide con l'output atteso dall'oracolo.
- **Fail:** Un test fallisce se, dato un input al sistema, l'output ottenuto è diverso dall'output atteso o se il sistema va in crash.

6. Approach

Il test che si effettuerà sul sistema si compone di tre fasi:

6.1. Testing di unità

Per il testing di unità, la strategia prevista consiste nel testare ogni singola unità (funzioni dei Controller e metodi dei Model) all'interno del sistema, selezionate dal modello a oggetti. Poiché il sistema è sviluppato in **Python**, i casi di test saranno documentati e implementati attraverso l'uso del framework **Unittest** o **PyTest**.

6.2. Testing d'integrazione

Lo scopo del testing di integrazione è quello di integrare tutte le componenti di una funzionalità al fine di testarle nel complesso (es. interazione tra Controller, Database e AI Service). Verrà utilizzato un approccio Bottom-up e l'ausilio di librerie di *mocking* (es. `unittest.mock`) per simulare le risposte del servizio IA esterno.

6.3. Testing di sistema

Prima di essere pronto all'uso, il sistema affronterà l'ultima fase di testing, quello di sistema, per dimostrare che siano soddisfatti tutti i requisiti richiesti. Sarà utilizzato il tool **Selenium**, che permette di registrare le azioni che un utente può intraprendere sul browser, simulando l'intero flusso (dal Login alla Pubblicazione).

7. Suspension and resumption

In questa sezione vengono definiti i criteri rigorosi che determinano quando l'attività di testing deve essere sospesa e quando può essere ripresa.
Criteri di Sospensione (Suspension Criteria): Le attività di test verranno sospese

immediatamente se si verifica una delle seguenti condizioni:

- **Blocco critico del sistema:** Presenza di difetti che impediscono le funzionalità base di accesso (Login/Registrazione) o la navigazione principale, rendendo impossibile l'esecuzione del 40% o più dei test case pianificati.
- **Indisponibilità dell'ambiente:** Il database MySQL o il servizio AIService (o il suo mock) non rispondono, impedendo qualsiasi test di integrazione o di sistema.
- **Instabilità dei dati:** Corruzione frequente dei dati nel database di test che rende i risultati non affidabili.

Criteri di Ripresa (Resumption Criteria): Le attività di test potranno riprendere solo quando:

- Tutti i difetti bloccanti (Show-stoppers) sono stati risolti e verificati con uno *Smoke Test* di successo.
- L'ambiente di test (Server Flask, Database, AI Mock) è stato ripristinato ed è stabile.
- È stata rilasciata una nuova build del software contenente le correzioni necessarie.

8. Testing materials (hardware/software requirements)

Questa sezione specifica le risorse hardware e software necessarie per eseguire i test pianificati, in coerenza con l'architettura definita nell'SDD .

Requisiti Hardware:

- **Client di Test:** PC Desktop/Laptop (min. 8GB RAM) per l'esecuzione dei test di sistema tramite Browser.
- **Server di Test (Locale/Staging):** Macchina in grado di ospitare il Backend Flask e il Database MySQL (min. 4 vCPU, 16GB RAM raccomandati per supportare il carico simulato).

Requisiti Software:

- **Sistema Operativo:** Windows 10/11, macOS o Linux (Ubuntu 20.04+).
- **Linguaggio & Framework:** Python 3.9+, Flask 2.x.
- **Database:** MySQL Community Server 8.0.
- **Strumenti di Testing (Test Runners):**
  - **PyTest** o **Unittest**: Per l'esecuzione dei test di unità e integrazione.
  - **Selenium WebDriver**: Per l'automazione dei test di sistema (browser Chrome/Firefox).
  - **Unittest.mock**: Per simulare le risposte dell'AIService durante i test di integrazione.
- **Browser:** Google Chrome (ultima versione stabile) o Mozilla Firefox per i test di interfaccia.

9. Test cases

Di seguito saranno elencati i test case delle varie funzionalità del sistema, utilizzando il metodo del **Category Partition**.

9.1. Gestione Utenza

9.1.1. TC_UTE_1: Registrazione Utente

Funzionalità: Permette a un utente Guest di registrarsi al sistema. **Riferimento:** RAD (UC6), ODD (gestione_utenza.py).

Parametro: Username

Nome Categoria	Scelte per Categoria
Lunghezza [UN_LEN]	1. Lunghezza < 1 (Vuoto) [ERROR]
	2. Lunghezza > 20 [ERROR]
	3. 1 <= Lunghezza <= 20 [PROPERTY UN_OK]

Parametro: Email

Nome Categoria	Scelte per Categoria
Formato [EM_FMT]	1. Formato non valido (manca @ o dominio) [ERROR]
	2. Formato valido [PROPERTY EM_OK]
Unicità [EM_UNI]	1. Email già presente nel DB [if EM_OK] [ERROR]
	2. Email non presente nel DB [if EM_OK] [PROPERTY EM_NEW]

Parametro: Password

Nome Categoria	Scelte per Categoria
Lunghezza [PW_LEN]	1. Lunghezza < 8 caratteri [ERROR]
	2. Lunghezza >= 8 caratteri [PROPERTY PW_OK]

Casi di Test (Test Frames)

Test Case ID	Test Frame	Esito Atteso
TC_UTE_1.1	UN_LEN 1	**Errore** (Username obbligatorio)
TC_UTE_1.2	UN_LEN 3, EM_FMT 1	**Errore** (Formato email non valido)
TC_UTE_1.3	UN_LEN 3, EM_FMT 2, EM_UNI 1	**Errore** (Email già registrata)
TC_UTE_1.4	UN_LEN 3, EM_FMT 2, EM_UNI 2, PW_LEN 1	**Errore** (Password troppo breve)
TC_UTE_1.5	UN_LEN 3, EM_FMT 2, EM_UNI 2, PW_LEN 2	**Corretto** (Utente registrato e loggato)

9.2. Gestione Pubblicazioni

9.2.1. TC_PUB_1: Inserimento Nuovo Post

Funzionalità: Permette a un utente autenticato di pubblicare una notizia, soggetta a verifica IA.
Riferimento: RAD (UC1), SDD (Soglia AI 0.7).

Parametro: Ruolo Utente

Nome Categoria	Scelte per Categoria
Autenticazione [AUTH]	1. Utente non loggato (Guest) [ERROR]
	2. Utente loggato (User) [PROPERTY AUTH_OK]

Parametro: Contenuto Post

Nome Categoria	Scelte per Categoria
Testo [TXT]	1. Testo vuoto o nullo [ERROR]
	2. Testo presente (> 0 caratteri) [PROPERTY TXT_OK]

Parametro: AI Score (Simulato)

Nome Categoria	Scelte per Categoria
Punteggio [SCORE]	1. Score < 0.7 [if AUTH_OK AND TXT_OK] [PROPERTY BLOCKED]
	2. Score >= 0.7 [if AUTH_OK AND TXT_OK] [PROPERTY PUBLISHED]

Casi di Test (Test Frames)

Test Case ID	Test Frame	Esito Atteso
TC_PUB_1.1	AUTH 1	**Errore** (Accesso negato/Redirect Login)
TC_PUB_1.2	AUTH 2, TXT 1	**Errore** (Contenuto mancante)
TC_PUB_1.3	AUTH 2, TXT 2, SCORE 1	**Corretto** (Post creato con stato 'Bloccato')
TC_PUB_1.4	AUTH 2, TXT 2, SCORE 2	**Corretto** (Post creato con stato 'Pubblicato')

9.3. Gestione Appelli

9.3.1. TC_APP_1: Creazione Appello

Funzionalità: Permette all'autore di contestare il blocco di un post. **Riferimento:** RAD (UC2), ODD (gestione_appelli.py).

Parametro: Stato Post

Nome Categoria	Scelte per Categoria
Stato Corrente [P_STAT]	1. Stato = 'Pubblicato' [ERROR]
	2. Stato = 'Bloccato' [PROPERTY STAT_OK]

Parametro: Storico Appelli

Nome Categoria	Scelte per Categoria
Esistenza [APP_HIST]	1. Appello già presente per questo post [if STAT_OK] [ERROR]
	2. Nessun appello precedente [if STAT_OK] [PROPERTY APP_NEW]

Casi di Test (Test Frames)

Test Case ID	Test Frame	Esito Atteso
TC_APP_1.1	P_STAT 1	**Errore** (Impossibile appellare post pubblicato)
TC_APP_1.2	P_STAT 2, APP_HIST 1	**Errore** (Appello già in corso o chiuso)
TC_APP_1.3	P_STAT 2, APP_HIST 2	**Corretto** (Appello creato, stato post 'In Revisione')

9.4. Gestione Segnalazioni

9.4.1. TC_SEG_1: Invio Segnalazione

Funzionalità: Permette a un utente di segnalare un post ritenuto fake o inappropriato. **Riferimento:** RAD (UC3), ODD (gestione_segnalazioni.py).

Parametro: Stato Post

Nome Categoria	Scelte per Categoria
Visibilità [VIS]	1. Post 'Bloccato' o 'In Revisione' [ERROR]
	2. Post 'Pubblicato' [PROPERTY VIS_OK]

Parametro: Storico Segnalazioni

Nome Categoria	Scelte per Categoria
Duplicati [DUP]	1. Utente ha già segnalato questo post [if VIS_OK] [ERROR]
	2. Prima segnalazione per questo post [if VIS_OK] [PROPERTY NEW_REP]

Casi di Test (Test Frames)

Test Case ID	Test Frame	Esito Atteso
TC_SEG_1.1	VIS 1	**Errore** (Post non visibile o non segnalabile)
TC_SEG_1.2	VIS 2, DUP 1	**Errore** (Segnalazione già inviata)
TC_SEG_1.3	VIS 2, DUP 2	**Corretto** (Segnalazione registrata)

10. Testing schedule

Fase di Test	Attività	Data Inizio	Data Fine	Responsabili
1. Unit Testing	Scrittura ed esecuzione test per Models e Controllers (es. Post.py, gestione_utenza.py).	26/12/2025	28/12/2025	Santo, Di Giuseppe
2. Integration Testing	Verifica interazioni Backend-Database e Backend-AI Mock.	29/12/2025	02/01/2026	Santo, Di Giuseppe
3. System Testing	Esecuzione scenari completi (E2E) tramite Selenium (es. Flusso Pubblicazione).	03/01/2026	07/01/2026	Santo, Di Giuseppe
4. Regression Testing	Verifica correzioni bug e stabilità finale.	08/01/2026	09/01/2026	Santo, Di Giuseppe
5. Rilascio Report	Generazione del report finale di test.	10/01/2026	10/01/2026	Santo, Di Giuseppe