Università degli Studi di Salerno
Corso di Ingegneria del Software

FakeBuster
Problem Statement
Versione 3.0

Data: 12/01/2026

Progetto: FakeBuster	Versione: 3.0
Documento: Problem Statement	Data: 12/01/2026

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
13/10/2025	1.0	Creazione e modifica sulla base delle indicazioni del prof in aula	Bruno Santo & Emiliano Di Giuseppe
19/10/2025	2.0	Modifica sulla base delle indicazioni del prof in aula	Bruno Santo & Emiliano Di Giuseppe
12/01/2026	3.0	Revisione Finale	Bruno Santo & Emiliano Di Giuseppe

Indice

1. Problem Domain

La disinformazione online continua a rappresentare una minaccia significativa per l'informazione pubblica e la fiducia nelle fonti digitali.
Molti social network consentono la diffusione incontrollata di notizie senza un controllo di veridicità, amplificando il rischio di fake news e manipolazione dell'opinione pubblica.
FakeBuster Social nasce come piattaforma sperimentale che unisce intelligenza artificiale e partecipazione umana per creare un ecosistema informativo più affidabile.
L'obiettivo principale è ridurre la diffusione di notizie false alla radice, bloccandone la pubblicazione già in fase di inserimento.
Il sistema valuta automaticamente le notizie proposte dagli utenti e decide se pubblicarle o meno in base a una stima di attendibilità.
Parallelamente, un fact-checker può analizzare i log delle decisioni dell'IA, verificando la correttezza delle valutazioni e contribuendo al miglioramento continuo del modello.
Il progetto si propone di:

- Creare un social network controllato dove la qualità delle informazioni è prioritaria.
- Sviluppare un modello di AI in grado di stimare in tempo reale la veridicità dei contenuti testuali.
- Garantire trasparenza e tracciabilità grazie ai log delle decisioni dell'IA, accessibili ai verificatori.

In questo modo, FakeBuster Social si distingue non come un semplice rilevatore di fake news, ma come una piattaforma proattiva che previene la diffusione di contenuti falsi, mantenendo al contempo un equilibrio tra automazione e supervisione umana.

2. Scenarios

• Scenario 1 — userSubmitsNews

Attori Partecipanti
maria:User, AIService, FakeBusterSystem

Flusso di eventi

- Maria accede alla pagina "Nuovo Post", inserisce titolo, testo e carica un'immagine (opzionale).
- Maria clicca "Pubblica".
- **AIService** riceve il testo, esegue la pulizia (rimozione URL/spazi) e verifica che non sia *gibberish* (testo senza senso).
- Il modello restituisce uno score di attendibilità (es. 0.75).
- Poiché lo score è superiore alla soglia critica di blocco calcolata dal sistema (es. ≥ 0.7), il **FakeBusterSystem** considera il contenuto accettabile e lo imposta su pubblicato.
- Il post appare nel Feed.

Precondizione
L'utente è autenticato.

Postcondizione
Il post è visibile nel Feed pubblico.

Quality requirements
• Decisione e pubblicazione entro 5 secondi.

• Scenario 2 — aiBlocksSubmission

Attori partecipanti
luca:User,AIService, FakeBusterSystem

Flusso di eventi

- Luca invia una notizia controversa tramite il form.
- **AIService** analizza il testo e restituisce uno score basso (es. `0.10`), inferiore alla soglia di accettazione.

- Il sistema imposta lo stato del post su `bloccato`.

- Luca **non** viene reindirizzato al feed; rimane sulla pagina e visualizza immediatamente un **Popup Modale** rosso: *"Post bloccato dall'IA. Il contenuto non ha superato la verifica automatica."*

- Nel popup, Luca compila il campo "Motivazione" e clicca "Fai appello".

- Il sistema registra un oggetto `Appello` nel database collegato al post e lo stato del post rimane bloccato in attesa di revisione.

Precondizione
Utente autenticato.

Postcondizione
Il post è salvato ma non visibile nel feed (stato bloccato); è stato creato un record di appello per il Fact Checker.

Quality requirements
• Notifica all'utente entro 5s.
• Motivo sintetico (es. "score 0.18 — alta probabilità di contenuto non verificato") mostrato all'utente; non esporre feature sensibili.

- ## Scenario 3 — userAppeal

Attori partecipanti
anna:User, fact checker:moderator1,AIService, FakeBusterSystem

Flusso di eventi

- Il Fact Checker effettua il login e viene indirizzato alla **Dashboard**.

- Il Fact Checker visualizza l'appello nella Dashboard.

- Valuta il testo originale e la motivazione dell'utente.

- Ha due opzioni:

  - **Pubblica:** Accetta l'appello, il post diventa visibile a tutti.
  - **Mantieni Bloccato:** Respinge l'appello, il post resta nascosto (stato `bloccato`) con esito `respinto`. *(Nota: Il Fact Checker non modifica il testo della notizia per preservarne l'integrità originale).*

- Il Fact Checker clicca sul pulsante **"Pubblica"**.

- Il sistema aggiorna lo stato del post a `pubblicato`, chiude l'appello con esito `accettato` e imposta il Fact Checker come revisore.

- Il post diventa visibile nel Feed pubblico.

Precondizione
Post in stato revisionee appello inviato.

Postcondizione
Post pubblicato o definitivamente respinto.

Quality requirements
• Fact checker deve visualizzare tutte le informazioni rilevanti (testo, log IA) in un'unica schermata.

• Scenario 4 — userReportsPublicPostToFactChecker

Attori partecipanti
user (utente autenticato),AIServicefactChecker, FakeBusterSystem

Flusso di eventi

- Un utente naviga nel Feed e vede un post che ritiene falso (sfuggito all'IA).

- Clicca il pulsante "Segnala" sul post.

- Si apre un popup dove inserisce il motivo (es. "Contenuto offensivo e falso").

- Il sistema crea una `Segnalazione` nel database.

- Il Fact Checker, nella sua Dashboard, vede la segnalazione nella sezione "Segnalazioni utenti".

- Il Fact Checker decide di bloccare il post cliccando **"Blocca"**.

- Il post scompare dal Feed pubblico (stato `bloccato`).

Precondizione
utente autenticato + post pubblico visibile.

Postcondizione
post bloccato o pubblico a discrezione del fact checker

Quality requirements

- trasparenza pubblica per azioni automatiche (banner + motivo);

3. Requirement

3.1 Functional Requirements

3.1.1 Funzionalità per l'Utente (User)

- Registrazione e Login.
- Creazione Post con upload immagini.
- **Eliminazione dei propri post** (funzionalità di self-moderation).
- Visualizzazione stato post (badge colorati: Pubblicato/Bloccato).
- Invio Appello immediato su blocco IA.
- Segnalazione post altrui (impedita sui propri post).

3.1.2 Funzionalità Sistema & Ai

- **Filtro Gibberish:** Rilevamento e blocco preventivo di testi privi di senso compiuto o troppo brevi.
- **Classificazione IA:** Analisi neurale con soglia di tolleranza (blocco automatico solo per score di attendibilità molto bassi, es. 0.3).
- Generazione Log IA (timestamp, score) per dashboard revisori.

3.1.3 Funzionalità per il FactChecker

- Dashboard unificata (Appelli e Segnalazioni).
- Visualizzazione dettagliata: Autore, Testo, Immagine, Score IA, Motivazione utente.
- **Azioni atomiche:** "Pubblica" (Sblocca) o "Blocca".

3.2 Non Functional

3.2.1 Disponibilità & ridondanza

- Architettura ad alta disponibilità; servizi critici ridondati (stateless + DB).

3.2.2 Scalabilità

- Scalabilità orizzontale per inference e backend (repliche auto-scalabili) senza modifiche architetturali.

3.2.3 Performance / SLA

- Decisione IA + pubblicazione automatica ≤ **5s** quando prevista
- autenticazione forte(hashing)

3.2.4 Protezione abuso

- Rate limiting per submit segnalazioni; meccanismi anti-abuse e dedup dei report.

3.2.5 Privacy & conformità

- Password salvate come hash; Controllo estensioni file immagini; Protezione CSRF (tramite Flask-WTF o gestione sessioni sicura).
- opzioni di archiviazione sicura.

3.2.6 Auditabilità & trasparenza

- Dashboard_checker consultabile;
- non esporre feature sensibili o parametri modello agli utenti finali.

3.2.7 Usabilità & accessibilità

- Interfacce chiare per invio post, appello e coda fact-checker; dashboard unificata per fact-checker; conformità base a standard di accessibilità.

4. Target environment

L'architettura del sistema è definita e implementata con le seguenti tecnologie:

- **Linguaggio Backend:** Python 3.x.
- **Framework Web:** Flask (gestione rotte, templating Jinja2, sessioni).
- **Database:** MySQL (interfacciato tramite `SQLAlchemy` e `pymysql`).
- **Intelligenza Artificiale:**
  - Librerie: `Torch`, `Transformers` (Hugging Face).
  - Modello: `AutoModelForSequenceClassification` pre-addestrato locale.
  - Preprocessing: `langdetect` (rilevamento lingua), `re` (pulizia regex).
- **Frontend:** HTML5, CSS3, JavaScript (Vanilla).
- **Testing:**

    - Unit/Integration: `Pytest`.
    - System/E2E: `Selenium WebDriver` (Chrome).
- **Ambiente di Sviluppo:** Visual Studio Code / PyCharm.

5. Deliverables & Deadlines

1. Problem Statement: 14 ottobre 2025
2. Requisiti e casi d'uso: 28 ottobre 2025
3. Requirements Analysis Document: 11 novembre 2025
4. System Design Document: 25 novembre 2025
5. Specifica delle interfacce dei moduli del sottosistema da implementare e parte dell' Object Design Document
Design Document: 16 dicembre 2025
6. Piano di test di sistema e specifica dei casi di test per il sottosistema da implementare: 16 dicembre 2025
7 Object Design Document: 20 dicembre
8.Implementazione e Documenti di Esecuzione del Test: 10 gennaio

6. Client Acceptance Criteria

Il sistema viene accettato se soddisfa i seguenti criteri, verificati tramite test di sistema automatizzati (Selenium):

1. **Blocco Automatico:** L'inserimento di una notizia fake nota deve innescare il blocco IA e l'apertura del popup di appello (Testato in test_system_ai_block.py).
2. **Flusso Positivo:** L'inserimento di una notizia valida deve portare alla pubblicazione immediata (Testato in test_system_create_post.py).
3. **Revisione Umana:** Il Fact Checker deve poter approvare un appello tramite la dashboard, rendendo il post pubblico (Testato in test_system_fact_checker.py).
4. **Stabilità:** Il sistema deve gestire correttamente input non validi (testo vuoto, lingua errata) senza crashare.