Università degli Studi di Salerno
Corso di Ingegneria del Software

FakeBuster
Problem Statement
Versione 2.0

Data: 19/10/2025

Progetto: FakeBuster	Versione: 2.0
Documento: Problem Statement	Data: 19/10/2025

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
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

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

Flusso di eventi

1. Maria (User) apre la form “Nuovo Post” e incolla titolo + testo + eventuale immagine.
2. Maria clicca “Invia”.
3. AIService analizza il testo: estrae features, calcola embedding e produce una stima di attendibilità (score = 0.87).
4. AIService invia la decisione e il log (features principali, score, timestamp) a FakeBusterSystem.
5. FakeBusterSystem, con soglia predefinita (es. publish_if_score ≥ 0.7), pubblica automaticamente il post nello stream pubblico.
6. Maria riceve notifica “Post pubblicato” con link.

Precondizione
L’utente è autenticato e non supera vincoli di rate-limit.

Postcondizione
Il post è pubblicato e la decisione dell’IA è registrata.

Quality requirements
• Decisione e pubblicazione entro 5 secondi.

• Scenario 2 — aiBlocksSubmission

Attori partecipanti
luca:User,AIService

Flusso di eventi

1. Luca invia una notizia (titolo + testo + immagine).
2. AIService valuta e produce score = 0.18 (basso).
3. FakeBusterSystem cambia stato del post in `bloccato` e invia notifica a Luca: “Il tuo contenuto non è pubblicato per possibile bassa attendibilità. Puoi modificare e riprovare o presentare appello.”
4. Luca può modificare il testo o allegare fonti; se modifica e reinvia, il ciclo riparte dalla valutazione AI.

Precondizione
Utente autenticato; contenuto analizzabile (testo o testo+link).

Postcondizione
Post rimane bloccato o viene reinviato dopo modifica.

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

Flusso di eventi

1. Anna trova il suo contenuto bloccato e clicca "Presenta appello"
2. FakeBusterSystem registra l'appello e mette in coda per revisione umana.
3. Un Fact checker(moderator1) apre l'interfaccia che mostra: testo del post, score IA.
4. Moderator1 valuta e sceglie fra: pubblica, mantieni bloccato.
5. Se pubblica → FakeBusterSystem pubblica il post. Se mantieni bloccato → notifica all'utente con motivazione sintetica.

Precondizione
Post in stato revisione e 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

Flusso di eventi

- user clicca **Segnala** su un post → compila motivo (+ commento.).

- system crea report.
- system assegna la task a un factChecker e invia notifica.
- factChecker valuta e decide.
- Autore può vedere la decisione del fact checker guardando i suoi post.

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 (Publisher / Reader)

- L'utente può registrarsi e accedere tramite l'apposito form
- Ricevere conferma immediata di ricezione.
- Presentare appello su post bloccati.
- Segnalare altri post (report) con motivo e commento.
- Ricevere notizie sullo stato del post (pubblicato, bloccato, in revisione, esito appello).
- Creare nuovo post: titolo, testo, link e allegati.

3.1.2 Funzionalità dell'AIService / Processo automatico

- Valutazione automatica per ogni post
- SLA: supporto alla pubblicazione automatica entro 5s quando previsto.

3.1.3 Funzionalità per il FactChecker

- Accedere alla dashboard di revisione con visuale unificata: testo, score IA e segnali utente. Può anche vedere le segnalazioni e i post bloccati
- Azioni possibili su un caso: pubblica, mantieni bloccato, rimuovi, edit.

3.1.4 Operazioni di pubblicazione e stato

- Stati supportati: in attesa, pubblicato, bloccato, rimosso.
- Transizioni determinate da regole: IA + soglie + azione FactChecker.
- Azioni automatiche visibili al pubblico (banner + motivo sintetico) quando applicate.

3.1.5 Requisiti di qualità chiave (breve)

- Decisione e pubblicazione automatica entro 5s quando applicabile.
- Notifica all'utente (es. blocco) entro 5s.

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
- HTTPS obbligatorio; autenticazione forte

3.2.4 Protezione abuso

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

3.2.5 Privacy & conformità

- Supporto per richieste di rimozione;
- 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

Poiché il progetto è in fase iniziale, le tecnologie esatte non sono ancora definitive. L'ambiente di sviluppo previsto include IDE come Visual Studio Code o PyCharm, controllo versione su GitHub e modellazione UML Lo stack tecnologico ipotizzato è: backend in Python (Flask o FastAPI) o in alternativa Node.js; frontend basato su

HTML/CSS (con JavaScript minimo o framework leggero se necessario); database relazionale MySQL per metadati e stato delle notizie. Per funzionalità AI/NLP si prevede l'uso di librerie Python consolidate come scikit-learn, spaCy e transformers per preprocessing, embedding e modelli; il servizio di inferenza sarà esposto via API dal backend.

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 deve essere in grado di analizzare con accuratezza le notizie, deve permettere agli utenti di fare un appeal o segnalazione per eventuali errori dell'IA. Deve permette la revisione umana da parte del fact-checker.