IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# TRUTHCHECK AI: AN INTEGRATED MULTI-MODEL FAKE NEWS DETECTION PLATFORM LEVERAGING RETRIEVAL-AUGMENTED GENERATIVE AI, OCR, AND REAL-TIME WEB EVIDENCE

[1]Anish Dhanaji Thakur, [2]Priyansh Rakesh Gangan, [3]Aaditya Ranjeetsingh Thakur, [4]Priyanshu Durgvijay Jaiswar, [5]Anupam Choudhary

[1]Student, [2]Student,[3]Student, [4]Student, [5]Faculty

[1] Department of Computer Engineering, [2] Department of Computer Engineering, [3] Department of Computer Engineering [4]Department of Computer Engineering, [5]Department of Computer Engineering

[1] Rizvi College of Engineering, Mumbai, India, [2] Rizvi College of Engineering, Mumbai, India, [3] Rizvi College of Engineering, Mumbai, India, [4] Rizvi College of Engineering, Mumbai, India, [5] Rizvi College of Engineering, Mumbai, India

***Abstract:*** The unchecked propagation of misinformation across digital media ecosystems has emerged as a critical societal challenge, undermining public health discourse, democratic processes, and financial market integrity. Existing automated fact-checking solutions are largely confined to research environments, lack multi-modal input support, and offer no user-facing infrastructure for persistent history or administrative oversight. This paper presents TruthCheck AI, a deployable web-based platform that enables registered users to submit news claims in either textual or image form and receive a structured, AI-generated verdict in real time. The system orchestrates four distinct processing layers: Optical Character Recognition (OCR) via Tesseract for image-to-text extraction; sentiment and subjectivity analysis through TextBlob; live evidence retrieval from the web using SerpAPI; and verdict generation via Google 2.5 Flash operating under a Retrieval-Augmented Generation (RAG) paradigm. Each verification is persisted in a MySQL relational database tied to a secure user account, while an administrator dashboard provides platform-wide analytics and user management. Experimental results confirm that well-established factual claims receive verdicts with confidence scores exceeding 80%, whereas ambiguous or temporally sensitive claims are correctly classified as UNVERIFIED. The platform demonstrates the feasibility of combining contemporary AI services into a cohesive, production-ready application for misinformation detection at scale.

***Index Terms*** - Fake News Detection, Retrieval-Augmented Generation, Natural Language Processing, Optical Character Recognition, Generative AI, Flask, Google Flash 2.5, SerpAPI, TextBlob, MySQL, Misinformation.

## I. INTRODUCTION

The digitisation of information delivery has fundamentally altered how news is produced, shared, and consumed. Social media platforms and instant messaging applications allow content — regardless of its veracity — to propagate to millions of users in minutes, often preceding any form of editorial scrutiny. The consequences of unchecked misinformation span multiple critical domains: fabricated health advisories have been shown to impede vaccination uptake and delay emergency medical responses; politically motivated disinformation campaigns have demonstrably influenced electoral outcomes; and financially motivated fake reports have artificially distorted asset valuations in capital markets [1].

Traditional fact-checking mechanisms rely on domain experts who manually cross-reference claims against authoritative sources. While such approaches yield high accuracy, they are resource-intensive, slow in execution, and fundamentally incapable of scaling to match the volume of content generated across contemporary digital platforms. There is therefore a well-defined need for automated systems capable of providing rapid, transparent, and accessible preliminary verdicts on submitted information.

Existing automated solutions present several practical shortcomings. Most operate as command-line research prototypes rather than deployable consumer applications. They frequently impose restrictions on input modality, accepting only plain text while ignoring the increasingly prevalent practice of disseminating news as image screenshots. Furthermore, they tend to omit user-facing features such as authentication, personal verification history, and administrative monitoring — all of which are essential for institutional or public deployment.

TruthCheck AI addresses these gaps by delivering a complete, multi-modal, web-based fact-checking platform. The system accepts both plain text and image inputs, applies NLP-based linguistic profiling, retrieves corroborating or contradicting evidence in real time, and leverages a state-of-the-art large language model (LLM) — Google 2.5 Flash — to generate a structured, human-readable verdict accompanied by a confidence score and a concise explanatory statement. All outputs are persisted in a relational database associated with authenticated user accounts, and an administrative dashboard provides platform-wide oversight.

## II. REVIEW OF LITERATURE

Research into automated fake news detection has advanced substantially over the past decade. Early methodologies relied on handcrafted linguistic features — including n-gram distributions, punctuation patterns, and syntactic parse trees — fed into classical machine learning classifiers. Contemporary approaches have shifted toward deep neural architectures and, more recently, toward leveraging the reasoning capabilities of large pre-trained language models.

Pérez-Rosas et al. [1] conducted one of the earliest systematic evaluations of automated fake news detection, applying Naïve Bayes, Logistic Regression, and Support Vector Machines to surface-level linguistic cues extracted from multi-domain news corpora. Their findings established that stylistic signals carry meaningful discriminative power, with cross-domain accuracy reaching approximately 74%. A key limitation of the study is its dependence on a static training distribution, which constrains generalisation to novel misinformation patterns.

Wang [2] introduced the LIAR benchmark — a large-scale dataset of over 12,700 manually labelled statements sourced from PolitiFact, annotated with fine-grained truthfulness categories ranging from "pants-fire" to "true." Experiments with Convolutional Neural Networks and Bidirectional LSTM architectures demonstrated that incorporating speaker credibility metadata alongside claim text measurably improves classification performance, establishing a standard evaluation benchmark for subsequent work.

Shu et al. [3] provided a comprehensive survey of fake news detection from a data-mining perspective, taxonomising approaches into knowledge-based, style-based, propagation-based, and source-credibility methods. Their analysis highlighted that multi-modal evidence fusion — combining textual content with social propagation signals — yields superior detection accuracy, directly motivating the multi-source evidence retrieval design adopted in TruthCheck AI.

More recent empirical work [4] has investigated the application of large language models such as GPT-3 and GPT-4 to zero-shot and few-shot fake news detection. These studies demonstrate that when LLMs are supplied with retrieved web snippets as grounding context — a pattern formalised as Retrieval-Augmented Generation (RAG) by Lewis et al. [10] — the incidence of hallucination is substantially reduced and verdict reliability is meaningfully improved. TruthCheck AI operationalises this insight as its core verification mechanism.

A notable gap that persists across the reviewed literature is the absence of end-to-end deployable systems that integrate multi-modal input, secure user authentication, persistent storage, and administrative oversight into a single cohesive application. The present work directly addresses this gap.

## III. METHODOLOGY AND SYSTEM DESIGN

### A. System Architecture

TruthCheck AI follows a monolithic web application architecture implemented with the Python Flask framework, organised into four logical layers:

1. **Presentation Layer** — Jinja2-templated HTML/CSS pages rendered server-side, employing CSS custom properties for a consistent dark-themed responsive interface.
2. **Application Layer** — Flask route handlers and business logic encapsulated in a single application module, with authentication decorators enforcing access control.
3. **AI and External Services Layer** — Tesseract OCR for image-to-text conversion, TextBlob for NLP analysis, SerpAPI for live web search retrieval, and Google Gemini 2.5 Flash for generative verdict production.
4. **Data Layer** — MySQL 8 database accessed via the mysql-connector-python driver, storing user accounts and all verification records with full referential integrity.

### B. Technology Stack

| Layer | Technology | Purpose |
|---|---|---|
| Backend Framework | Python 3, Flask 3.0.3 | Web server, routing, session management |
| Database | MySQL 8, mysql-connector-python 8.4.0 | Persistent user and verification storage |
| Authentication | bcrypt 4.1.3, Flask sessions | Secure password hashing and login state |
| Generative AI | Google Gemini 2.5 Flash (google-genai) | AI-powered verdict generation |
| Web Search | SerpAPI (google-search-results 2.4.2) | Real-time evidence retrieval |
| NLP | TextBlob 0.18.0 | Sentiment and subjectivity analysis |
| OCR | Tesseract, pytesseract 0.3.13, Pillow 10.4.0 | Text extraction from uploaded images |
| Frontend | Jinja2, CSS3, CSS Variables | Responsive templated UI |

Figure 3.1 Technology Stack

C. Verification Algorithm

The verification pipeline executes in six sequential stages upon receiving a user submission:
**Stage 1 — Input Ingestion:** If the submission is an image file, Tesseract OCR is invoked to extract the textual content; the extracted string is then treated identically to a direct text input. Empty submissions after extraction trigger a validation error.

**Stage 2 — NLP Analysis:** TextBlob computes sentiment polarity in the range [−1.0, +1.0], classified as Positive (> 0.1), Negative (< −0.1), or Neutral otherwise. Subjectivity is computed in [0.0, 1.0] and expressed as a percentage; values above 70% indicate potentially opinion-driven content.

**Stage 3 — Web Evidence Retrieval:** SerpAPI is queried using the first 150 characters of the submission. The top three organic search results are extracted, yielding title-snippet pairs and source URLs that serve as grounding context for the subsequent AI stage.

**Stage 4 — AI Verdict Generation:** A structured prompt is assembled incorporating the original claim and the retrieved search snippets. This prompt is submitted to Google Gemini 2.5 Flash, which is instructed to return a response in a fixed machine-parseable format: VERDICT (TRUE / FALSE / UNVERIFIED), CONFIDENCE (0–100), and EXPLANATION (one sentence). The system parses the response deterministically by scanning for labelled prefixes, implementing a lightweight RAG pattern that grounds model output in retrieved evidence rather than relying solely on parametric memory.

**Stage 5 — Database Persistence:** The complete verification record — comprising user identifier, input text, verdict, confidence score, explanation, sentiment, subjectivity, and source URLs — is inserted into the verifications table.

**Stage 6 — Result Display:** The verification result is rendered to the user alongside NLP metrics, sources consulted, and the AI-generated explanation.
D. Database Design

The database, named truthcheck_db, comprises two tables. The users table stores account credentials (bcrypt-hashed passwords), email addresses, role assignments (user or admin), and account creation timestamps. The verifications table stores each fact-check outcome and references the users table via a foreign key on user_id with an ON DELETE CASCADE constraint, ensuring that a deleted user's verification history is automatically removed. The relationship is one-to-many: a single user may accumulate zero or more verification records over time.

E. Authentication and Authorisation

Passwords are hashed using bcrypt with a per-record cryptographic salt, providing resistance against rainbow table attacks; plaintext passwords are never stored or transmitted. Upon successful authentication, the user's identifier, username, and role are stored in a server-side Flask session signed with the application's secret key. Access control is enforced by two Python decorator functions: login_required, which redirects unauthenticated requests to the login page, and admin_required, which additionally verifies that the session role equals "admin" before permitting access to administrative routes. This decorator pattern cleanly separates access control logic from route handler implementation.

F. Application Routes

| URL Route | HTTP Method | Function | Access Level |
|---|---|---|---|
| / | GET | Verification home page | Authenticated users |
| /verify | POST | Verification form handler | Authenticated users |
| /login | GET, POST | User login | Public |
| /register | GET, POST | User registration | Public |
| /logout | GET | Session clearance | Authenticated users |
| /history | GET | Personal verification history | Authenticated users |
| /admin | GET | Administrator dashboard | Admin only |
| /admin/delete_user/ | POST | User account deletion | Admin only |
| /admin/history/ | GET | Any user's history view | Admin only |

Figure 3.2 Application routes

## IV. RESULTS AND DISCUSSION

A. Functional Performance

TruthCheck AI was evaluated across all defined functional requirements, with the following observations:

Multi-modal Input: The OCR pipeline correctly processes image submissions containing news headlines and article screenshots, provided inputs are of adequate resolution and contrast. Tesseract reliably extracts textual content for downstream processing in these conditions.

NLP Analysis: Opinion pieces and editorial content consistently receive subjectivity scores above 60%, while factual reporting yields markedly lower values. Inflammatory or sensationalist language is correctly identified by the sentiment classifier as Negative, which frequently co-occurs with fabricated content in the test corpus.

Web Search Integration: SerpAPI successfully retrieves real-time search results for the submitted claims. The top-three snippet extraction supplies sufficient contextual grounding for the Gemini model to anchor its reasoning in current, retrieved information rather than relying exclusively on parametric knowledge.

AI Verdict Accuracy: The Google Gemini 2.5 Flash model produces well-structured, deterministically parseable verdicts. Established factual claims receive confident TRUE or FALSE verdicts with confidence scores typically above 80%. Obscure, recent, or inherently ambiguous claims are appropriately classified as UNVERIFIED with reduced confidence scores, indicating that the model correctly distinguishes verifiable from unverifiable content.

Authentication and Authorisation: Registration, login, logout, session management, and role-based access control all function correctly across tested scenarios. Bcrypt hashing ensures that no plaintext credential is stored or exposed at any point.

Database Persistence: All verification records are correctly stored in MySQL and associated with the submitting user. The history page renders personal records in reverse chronological order, and cascading deletion removes all associated records upon account removal.

B. Sample Verification Results

| Submitted Claim | Verdict | Confidence | Sentiment | Subjectivity |
|---|---|---|---|---|
| "The Earth revolves around the Sun." | TRUE | 98% | Neutral | 12% |
| "Drinking bleach cures viral infections." | FALSE | 95% | Neutral | 18% |
| "The government is hiding alien contact." | UNVERIFIED | 30% | Negative | 78% |
| "Scientists discover water on Mars." | TRUE | 82% | Positive | 35% |
| "Stock markets will crash next week." | UNVERIFIED | 20% | Negative | 65% |

Figure 4.1 Sample Results

The results illustrate a clear correspondence between claim verifiability and confidence score, and between subjectivity level and the likelihood of an UNVERIFIED or FALSE outcome.

C. Limitations

Several limitations were identified during evaluation. The system is dependent on third-party APIs (SerpAPI and Google Gemini), both of which impose rate limits and require active credentials; the SerpAPI free tier permits 100 queries per month, which is restrictive for sustained public use. OCR accuracy degrades on images with low contrast, non-standard typefaces, or significant background noise. Very recent events may receive UNVERIFIED verdicts not due to actual unverifiability but because retrieved search snippets lack sufficient context. The platform currently supports English-language content only, limiting accessibility for multilingual user populations. Finally, API keys and database credentials are embedded in the source code, which is unsuitable for production deployment without migration to environment variable management.

## V. CONCLUSION

TruthCheck AI demonstrates that a practical, accessible, and deployable fake news detection platform can be realised by intelligently orchestrating multiple contemporary technologies: OCR for multi-modal input normalisation, NLP for linguistic profiling, real-time web search for evidence retrieval, and Generative AI operating under a RAG paradigm for explainable verdict generation. The system successfully achieves all stated design objectives and delivers a fully functional web application with secure authentication, role-based access control, persistent storage, and administrative oversight.

The integration of Google Gemini 2.5 Flash via the RAG pattern is particularly significant: by grounding model output in retrieved evidence rather than parametric memory alone, the system substantially improves verdict reliability and transparency. Users receive not merely a binary classification but a confidence score, a one-sentence explanation, and references to the sources consulted — a design philosophy that encourages informed engagement rather than uncritical reliance on automated outputs.

Future development directions include extending OCR and NLP support to Hindi, Marathi, and other regional languages; integrating a dedicated fine-tuned classification model trained on established benchmarks such as LIAR; packaging the core logic as a browser extension for inline verification; enabling social media URL ingestion for automatic claim extraction; migrating to a hardened production deployment with WSGI serving, HTTPS, and environment-managed credentials; and enriching the administrator dashboard with time-series analytics and verdict distribution visualisations.

## IV. ACKNOWLEDGMENT

I am profoundly grateful to Prof. Anupam Choudhary for his expert guidance and continuous encouragement throughout to see that this project rights its target.

I would like to express deepest appreciation towards Dr. Varsha Shah, Principal RCOE, Mumbai and Prof. Mohd Juned HOD Computer Engineering Department whose invaluable guidance supported me in this project.

At last I must express my sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped us directly or indirectly during this course of work.

## REFERENCES

**[1]** V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, "Automatic Detection of Fake News," in *Proc. 27th Int. Conf. Computational Linguistics (COLING)*, 2018, pp. 3391–3401.

**[2]** W. Y. Wang, "LIAR: A Benchmark Dataset for Fake News Detection," in *Proc. 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017, pp. 422–426.

**[3]** K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake News Detection on Social Media: A Data Mining Perspective," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

**[4]** S. Hoes, J. Altay, and J. Acerbi, "Evaluating the Feasibility of Large Language Models for Fake News Detection," *PLOS ONE*, 2023.

**[5]** Pallets Projects, *Flask Documentation*, 2024. [Online]. Available: https://flask.palletsprojects.com

**[6]** Google LLC, *Google Generative AI Python SDK*, 2024. [Online]. Available: https://ai.google.dev

[7] SerpAPI, *SerpAPI Documentation*, 2024. [Online]. Available: https://serpapi.com/docs

**[8]** TextBlob Contributors, *TextBlob Documentation*, 2024. [Online]. Available: https://textblob.readthedocs.io

**[9]** Google LLC, *Tesseract OCR*, 2024. [Online]. Available: https://github.com/tesseract-ocr/tesseract

**[10]** P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Proc. NeurIPS*, 2020.

**[11]** Oracle Corporation, *MySQL 8.0 Reference Manual*, 2024. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/