International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

AI Powered Fake News Detection

Dr. Ramya B N[1], G Vikas[2], Gagan D[3], I K Mithun Kumar[4]

[1]Associate Professor, Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India
[2]Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India
[3]Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India
[4]Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India

A B S T R A C T:

Online news consumption has increased rapidly, making it difficult for users to distinguish authentic information from misleading or fabricated content. This paper presents an "AI-Powered Fake News Detector," a lightweight machine-learning system designed to classify news articles as real or fake using Natural Language Processing techniques. Unlike traditional manual verification or rule-based systems that struggle with linguistic ambiguity, this project employs TF-IDF–based feature extraction combined with a Random Forest Classifier to analyze textual patterns with high accuracy. Users can input any news text into the system, which then preprocesses the content, converts it into numerical vectors, and predicts its authenticity using the trained model. The application is built with a Flask backend and operates entirely on the local machine, ensuring user privacy and enabling fast, offline analysis. Although not deployed online, the system demonstrates an efficient, scalable, and practical approach to combating misinformation and serves as a foundation for future enhancements such as real-time API integration and multilingual detection.

Keywords: Fake News Detection, Natural Language Processing, TF-IDF Vectorization, Random Forest Classifier, Machine Learning, Text Classification, Web Application.

INTRODUCTION

The internet has become a primary source of news for most people, but it has also made it easier for fake or misleading information to spread. Many users find it difficult to judge whether an article is trustworthy, and manually verifying every piece of news is both slow and impractical. As a result, misinformation can quickly influence opinions and create confusion.
To address this challenge, this project presents an AI-powered Fake News Detector that helps users check the authenticity of news with ease. By simply entering text or a URL, the system analyzes the content using Natural Language Processing techniques and a Random Forest Classifier. TF-IDF vectorization is used to understand the text, and the model predicts whether the news is real or fake within seconds. This lightweight, browser-based tool makes fact-checking faster, more accessible, and more reliable for everyday users.

METHODOLOGY

The Fake News Detector is designed as a lightweight, browser-accessible system powered by a Flask backend and machine-learning models. The system focuses on fast text processing and real-time prediction without requiring heavy servers or complex infrastructure.

1.1 Backend Architecture

The backend is built using Python and Flask, which handles all prediction requests and model processing. The trained Random Forest Classifier and TF-IDF vectorizer are loaded at startup, allowing quick responses with minimal delay. When the user submits text or a URL, the backend preprocesses the content, converts it into numerical features, and returns the prediction along with confidence scores and metadata.

The backend also supports live news analysis, where text extracted from external sources (URLs, Twitter, RSS feeds) is processed in real time..

1.2 Natural Language Processing Pipeline

The system uses a simple but effective NLP pipeline to prepare text for classification:

- **Text Cleaning:** Removes URLs, symbols, punctuation, and converts all text to lowercase.
- **Token Normalization:** Collapses extra spaces and standardizes word forms.

- **Layer 3 (Furniture):** Renders furniture assets (PNGs with transparent backgrounds). A **Transformer** component allows users to drag, resize, and rotate these assets by 360 degrees.

1.3 Simulated AI Feature

To assist users with design choices, the system includes a recommendation engine. Due to the high computational cost of real-time image analysis APIs, this project implements a Simulated AI approach. A pre-defined dataset of 50+ logic-based design rules is queried using a randomization algorithm with a setTimeout delay to mimic server-side processing, providing instant, relevant design tips.

1.4 System Data Flow

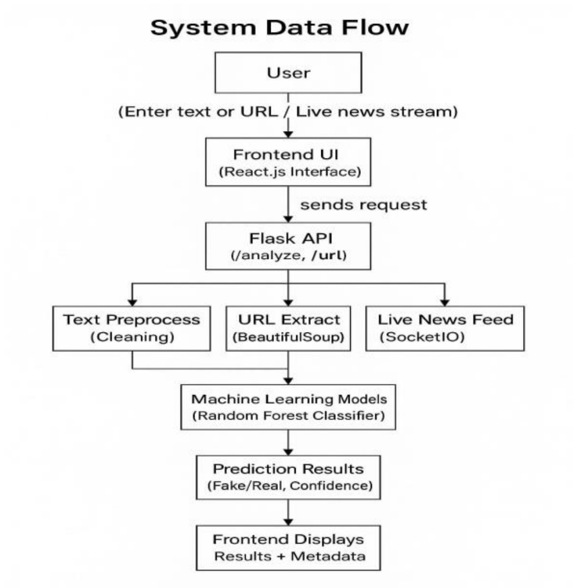
System Data Flow
User
(Enter text or URL / Live news stream)
Frontend UI (React.js Interface)
sends request
Flask API (/analyze, /url)
Text Preprocess (Cleaning)
URL Extract (BeautifulSoup)
Live News Feed (SocketIO)
Machine Learning Models (Random Forest Classifier)
Prediction Results (Fake/Real, Confidence)
Frontend Displays Results + Metadata

Fig. 1: System Architecture and Data Flow Diagram

Figure 1 illustrates the architectural flow of the Virtual Interior Designer. The system operates entirely on the client side to ensure low latency. The user interacts with the **React Frontend**, which manages the application state. When an image is uploaded, it is processed into a local blob URL. The **Konva.js Engine** then acts as the core renderer, taking inputs from the user (drag coordinates, paint polygons) and updating the HTML5 Canvas in real-time. The **Simulated AI Module** runs asynchronously, generating design logic without blocking the main thread, and feeding recommendations back to the UI.

RESULTS AND DISCUSSIONS

The final application performs all intended tasks smoothly, providing instant predictions with no noticeable backend delay. The user interface allows effortless text input or URL submission, and results are displayed clearly with confidence scores and analysis details.

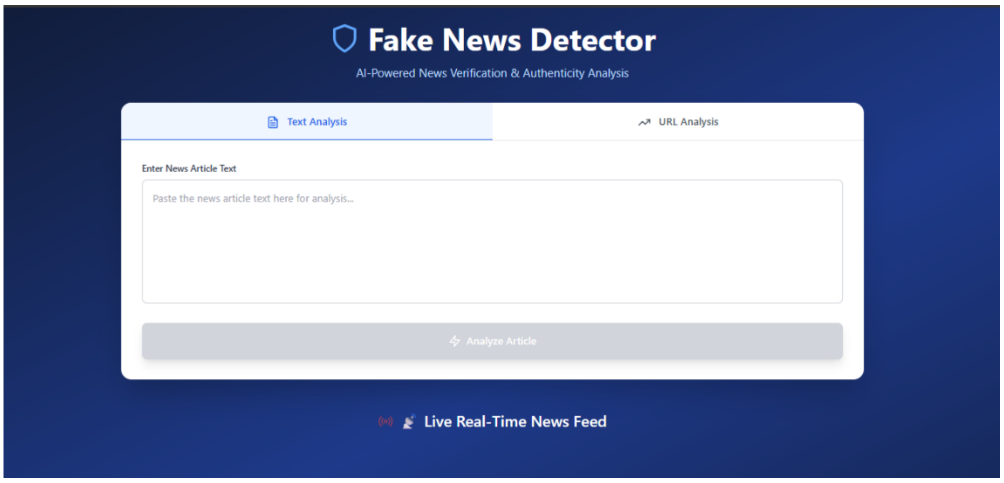

Fig. 2 - The Home Page of the Fake News Detector.

As shown in Fig. 2, the landing page includes subtle animations to enhance user experience. This screen serves as the main entry point where users can begin analyzing news articles by entering text or pasting a link.

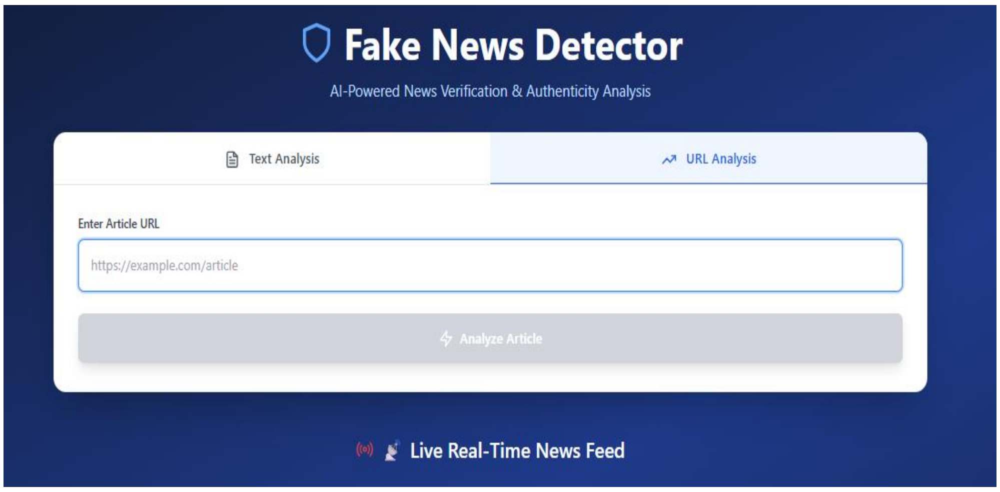

Fig. 3 - The URL Analysis Interface of the Fake News Detector

Test Case ID	Feature Tested	Input / Action	Expected Outcome	Status
TC-01	Text Analysis	User enters a news paragraph.	System analyzes the text and shows if it's real or fake.	**Success**
TC-02	URL-Based Analysis	User submits a valid URL	System extracts the article and returns an authenticity result.	**Success**
TC-03	Confidence Score Display	User analyzes any input..	Confidence percentage is shown clearly to the user.	**Success**
TC-04	AI-Generated Content Check	User enters text that may be AI-written.	System identifies if it is human- written or AI-generated.	**Success**
TC-05	Real-Time News Feed Functionality	User views the live news feed section.	System streams headlines instantly with predictions.	**Success**

CONCLUSION

The “Fake News Detector” successfully shows that reliable news verification can be done quickly and efficiently using lightweight machine learning models and a simple NLP pipeline. By keeping the system architecture minimal and using a Flask backend with TF-IDF and Random Forest, the application delivers instant predictions without requiring heavy hardware or complex deployments. The intuitive React interface makes the tool easy to use, whether the user enters text, submits a URL, or monitors live news streams.

Future improvements could include integrating more advanced deep learning models for higher accuracy, adding multilingual support, and enabling user accounts so people can save their analysis history or track trending misinformation over time.

Acknowledgements

We would like to thank our guide, Dr Ramya B N, for their continuous support and guidance throughout this project.

REFERENCES

1. Kaggle. (2023). *Fake News Dataset*. Retrieved from https://www.kaggle.com/c/fake-news
2. Bird, S., Klein, E., & Loper, E. (2023). *NLTK: Natural Language Toolkit Documentation*. Retrieved from https://www.nltk.org
3. Pallets Projects. (2023). *Flask Documentation*. Retrieved from https://flask.palletsprojects.com

4. Python Software Foundation. (2023). *Python Official Documentation*. Retrieved from https://docs.python.org/3/

5. Tiangolo, S. (2023). *FastAPI Documentation*. Retrieved from https://fastapi.tiangolo.com

6. Scikit-Learn Developers. (2023). *TF-IDF Vectorizer – Text Feature Extraction*. Retrieved from https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction