

B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU



OOMD Mini Project Report

FAKE NEWS DETECTION SYSTEM

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

OWAIS ALI KHADIM BATTI (1BM23CS221)
PARTH B PATIL (1BM23CS227)
PIYUSH SAMDARIYA(1BM23CS230)

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2025-26

B.M.S. COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



DECLARATION

We, **OWAIS ALI KHADIM BATTI (1BM23CS221)** , **PARTH B PATIL (1BM23CS227)** and **PIYUSH SAMDARIYA(1BM23CS230)** students of 5th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this OOMD Mini Project entitled "**FAKE NEWS DETECTION SYSTEM**" has been carried out in Department of CSE, B.M.S. College of Engineering, Bangalore during the academic semester August 2025- December 2025. I also declare that to the best of our knowledge and belief, the OOMD mini Project report is not from part of any other report by any other students.

Signature of the Candidate

OWAIS ALI KHADIM BATTI (1BM23CS221)

PARTH B PATIL (1BM23CS227)

PIYUSH SAMDARIYA(1BM23CS230)

B.M.S. COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING



CERTIFICATE

This is to certify that the OOMD Mini Project titled “**FAKE NEWS DETECTION SYSTEM**” has been carried out by **OWAIS ALI KHADIM BATTI (1BM23CS221)** , **PARTH B PATIL (1BM23CS227)** , **PIYUSH SAMDARIYA(1BM23CS230)** during the academic year 2025-2026.

Signature of the Faculty in Charge

DR. ADARSHA SAGAR H V

Table of Contents

Sl No	Title	Pageno
1	Ch 1: Problem statement	
2	Ch 2: Software Requirement Specification	
3	Ch 3: Class Diagram	
4	Ch 4: State Diagram	
5	Ch 5: Interaction diagram	
6	Ch 6: UI Design with Screenshots	

Chapter 1: Problem Statement

The rapid growth of digital communication technologies has transformed the way people consume information. Social media platforms, online news websites, blogs, and messaging applications enable users to access news instantly and share it widely within seconds. While this accessibility has numerous advantages, it also creates a major vulnerability: the uncontrolled spread of *fake news*. Fake news refers to fabricated, misleading, or exaggerated information presented as legitimate news with the intention to deceive readers, provoke emotional reactions, or manipulate public perception. As digital content continues to expand, distinguishing between trustworthy information and misinformation has become increasingly difficult for ordinary users.

The consequences of fake news are significant and far-reaching. In the political sphere, misinformation can influence elections, shape public opinion, and destabilize democratic processes. In the social domain, fake news can cause panic, promote violence, fuel hatred, and damage reputations. During crises such as pandemics or natural disasters, false information can mislead citizens, hinder emergency responses, and even put lives at risk. The economic impact is also considerable, as fabricated stories can manipulate stock markets, harm businesses, and mislead consumers. Because of these risks, combating fake news has become a global priority for governments, media authorities, and technology companies.

However, manual verification of news is time-consuming and often impractical given the massive volume of online content continuously being generated. Human fact-checkers cannot keep pace with the speed at which misinformation spreads. Moreover, many users lack the digital literacy skills needed to critically evaluate online content. As a result, there is a strong need for automated solutions that can assist in detecting and limiting the spread of fake news before it reaches large audiences.

To address this challenge, the development of an intelligent **Fake News Detection System** is essential. By using machine learning (ML), natural language processing (NLP), and data mining techniques, the system can analyze news articles, detect patterns associated with fake content, and classify information as “real” or “fake” with high accuracy. Such a system can process large amounts of data quickly, identify linguistic and semantic cues in text, and learn from verified datasets to continuously improve its performance. The system aims to provide users with a reliable and efficient tool for evaluating the authenticity of news, enabling them to make informed decisions and reducing the impact of misinformation.

The primary problem this project seeks to solve is the **lack of an automated, scalable, and reliable mechanism to detect fake news across digital platforms**. The Fake News Detection System will contribute to improving the quality of information consumed by the public, supporting media organisations in fact-checking processes, and promoting safer digital communication environments. Ultimately, this system aims to enhance information integrity, reduce the spread of harmful content, and build greater trust in digital media ecosystems.

Chapter 2: Software Requirement Specification

1. Problem Statement

The rapid spread of misinformation and fake news on digital platforms poses a major threat to society, influencing public opinion and decision-making. Detecting fake news manually is time-consuming, biased, and unreliable.

This project aims to develop a **Fake News Detection System** that uses **Natural Language Processing (NLP)** and **Machine Learning (ML)** to classify news as *fake* or *real* with high accuracy, providing users with trustworthy information and reducing misinformation.

2. Purpose

The purpose of the Fake News Detection System is to automatically detect fake or misleading news articles based on textual analysis and metadata. The system provides accurate predictions with confidence scores, explanations, and credibility assessments to assist readers, journalists, and organizations in validating information authenticity.

3. Scope

The Fake News Detection System will:

- . Accept input in the form of text or URLs.
- . Preprocess the data to extract meaningful features.
- . Use trained machine learning/NLP models to predict the authenticity of news.
- . Display the results with confidence scores and highlighted reasoning.
- . Support admin/fact-checker feedback to improve model accuracy over time.

The system can be expanded to multiple languages, integrated with browsers, or adapted for social media content verification.

4. Overview

The project involves creating a **web-based system** integrated with a **machine learning model** trained on labelled datasets of fake and real news.

Users can enter a news article or link, and the system will analyse it to predict whether the content is fake, real, or uncertain. A confidence score and key reasoning are displayed.

5. General Description

- . **Users:** General public, journalists, researchers, and administrators.
 - . **Platform:** Web-based (with potential for future mobile version).
 - . **Technology Stack:** Python, Flask/Django, HTML/CSS, JavaScript, TensorFlow/PyTorch, NLP libraries (NLTK, SpaCy, Transformers).
 - . **Database:** MySQL or MongoDB.
 - . **Operating Environment:** Compatible with Windows, Linux, and macOS.
-

6. Functional Requirements

1. User Module

- . Users can input news text or URL for analysis.
- . System displays results with label (Fake/Real/Uncertain).
- . Users can provide feedback or report errors.

2. Admin Module

- . Admin can view logs, update datasets, and retrain the model.
- . Admin can review flagged or uncertain results.

3. Prediction Module

- . Preprocess text (tokenization, stop word removal, stemming).
- . Extract features using TF-IDF or embeddings.
- . Classify using a trained ML/NLP model.

4. Feedback Module

- . Allows user or fact-checker feedback.
 - . Stores corrections for model retraining.
-

7. Non-Functional Requirements

1. Performance:

- . The system should predict the result within **3–5 seconds** for standard input length.

2. Scalability:

- .Should handle concurrent user requests efficiently.
3. **Accuracy:**
 - .Target model accuracy $\geq 90\%$.
 4. **Security:**
 - .Secure database access and input validation.
 - .Prevent injection or misuse of API endpoints.
 5. **Usability:**
 - .User-friendly and minimal UI with clear result visualization.
 6. **Maintainability:**
 - .Modular code structure for easy updates and retraining.
-

8. Interface Requirements

- . **User Interface:**
Clean and responsive web interface built with HTML, CSS, and JavaScript.
 - . **Hardware Interface:**
Standard system with at least 8GB RAM for local deployment or server hosting.
 - . **Software Interface:**
Interfaces with trained ML models, databases, and cloud storage (if applicable).
-

9. Software Requirements

Type	Requirement
Operating System	Windows 10+, Linux (Ubuntu 20.04+), macOS
Language	Python 3.10+, HTML, CSS, JS
Framework	Flask / Django
ML Libraries	TensorFlow / PyTorch / Scikit-learn
NLP Libraries	NLTK, SpaCy, Transformers
Database	MySQL / MongoDB
IDE	VS Code / PyCharm
Hosting	AWS / Render / Localhost

10. Design Constraints

- . The system must use an open-source dataset or self-curated labelled dataset.
 - . The ML model should be explainable and interpretable.
 - . Deployment must comply with data privacy and ethical AI standards.
-

11. Preliminary Schedule

Phase	Tasks	Duration
Phase 1	Research & Data Collection	2 weeks
Phase 2	Preprocessing & Feature Engineering	2 weeks
Phase 3	Model Development & Training	3 weeks
Phase 4	Web Interface Development	2 weeks
Phase 5	Testing & Evaluation	2 weeks
Phase 6	Final Deployment & Documentation	1 week
Total Duration: ~12 weeks (3 months)		

12. Preliminary Budget (in INR)

Item	Description	Estimated Cost (₹)
Data Storage & Processing	Cloud storage, datasets	₹8,000
Development Tools	Software, licenses (if any)	₹3,000
Hosting Services	Cloud server (AWS/Render)	₹5,000
Model Training Resources	GPU or Colab Pro+ usage	₹4,000
Miscellaneous	Internet, maintenance, testing	₹2,000
Total Estimated Budget		₹22,000

Total Estimated Budget ₹22,000

13. Conclusion

The Fake News Detection System aims to provide an efficient, scalable, and user-friendly platform for detecting fake news using modern NLP and ML methods. It promotes a more informed society by enhancing content verification and credibility across online platforms.

At the core of the system is the **Database**, which stores articles, prediction results, and

The process begins with an **Article** submitted by a **User**. The article flows into the

These embeddings are passed to the **ModelService**, which uses the machine learning model to

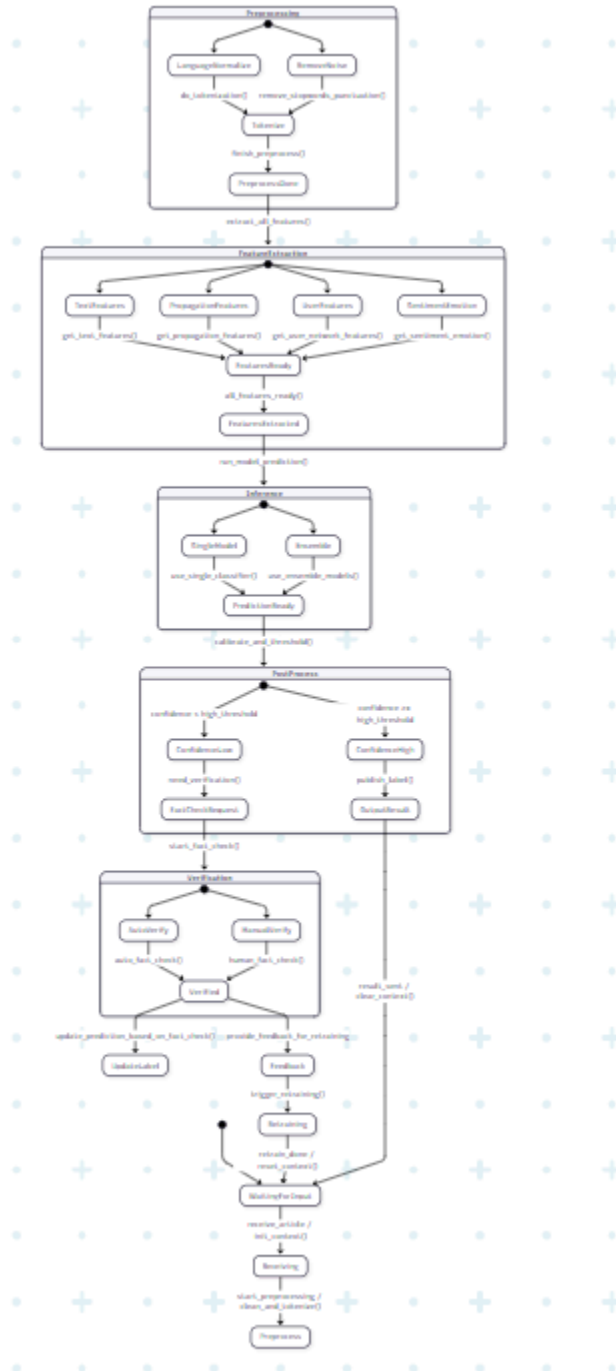
To improve transparency, the **ExplanationEngine** generates explanations for the output. It

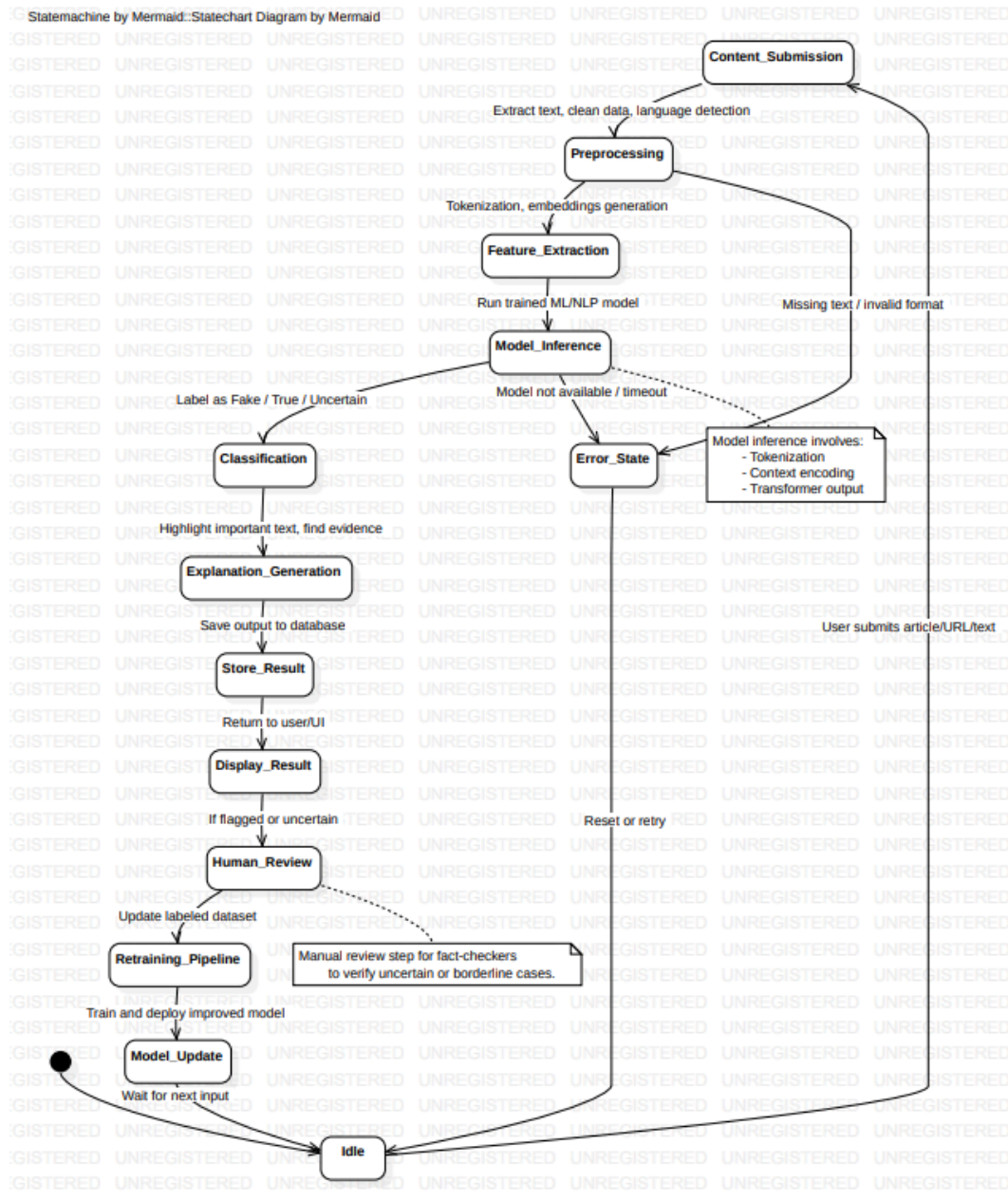
A. FastCheckers plays a nontrivial role by reviewing articles and expressing an opinion

A **FactChecker** plays a supervisory role by reviewing articles and approving or rejecting system-generated labels. This ensures human oversight and improves reliability. Meanwhile, the **Admin** manages system users, monitors overall system health, and updates the machine learning model when needed.

Overall, the diagram represents a structured, modular system where each component performs a specific function. Together, they enable efficient fake news detection with automated analysis, human verification, and administrative control.

Chapter 4: State Modeling





States

- **Content_Submission**

Meaning: User or external source submits an article/text/URL.

Relevance: Entry point to the system — captures user intent and initial metadata. Validates input format before any work begins.

- **Preprocessing**

Meaning: Text extraction, cleaning, language detection, normalization.

Relevance: Removes noise and standardizes input so downstream components (tokenizers, models) get reliable, comparable data. Critical for model accuracy and

multilingual support.

- **Feature_Extraction**

Meaning: Tokenization, embedding/context encoding, generation of model-ready features.

Relevance: Translates raw text into numerical representations the ML model can consume; quality here determines model performance and ability to capture semantics.

- **Model_Inference**

Meaning: Run the trained ML/NLP model to compute logits/labels and confidence scores.

Relevance: Core decision-making step — produces the prediction (fake/real/uncertain) used by all following steps.

- **Error_State**

Meaning: A fault occurred (missing/invalid input, timeout, model offline).

Relevance: Allows graceful handling of failures: retry, notify user, log for ops. Prevents silent incorrect outputs.

- **Classification**

Meaning: Model output mapped to human-friendly labels (Fake / True / Uncertain).

Relevance: Converts technical scores into actionable decisions and drives downstream UI and moderation logic.

- **Explanation_Generation**

Meaning: Produce textual highlights, key phrases, and supporting evidence to justify the prediction.

Relevance: Improves transparency and trust; helps users and fact-checkers understand *why* the system made that decision.

- **Store_Result**

Meaning: Save prediction, explanation, metadata into the database.

Relevance: Persists results for auditing, future retraining, and allowing fact-checkers/admins to review history.

- **Display_Result**

Meaning: Return the prediction and explanation to the submitting user or UI.

Relevance: Final user-facing output; must be clear and actionable so users can make informed decisions.

- **Human_Review**

Meaning: Flagged/uncertain items are forwarded to fact-checkers for manual verification.

Relevance: Ensures quality control for borderline cases, collects labels for improving the

model, and reduces false positives/negatives.

- **Retraining_Pipeline**

Meaning: Incorporate reviewed/curated labels and new data to retrain models.

Relevance: Keeps model up to date with emerging misinformation patterns and reduces concept drift.

- **Model_Update**

Meaning: Deploy the newly trained model to production.

Relevance: Makes improvements live; part of the CI/CD and MLOps loop that ensures continuous improvement.

- **Idle**

Meaning: System waiting for next input or background jobs complete.

Relevance: Represents steady state; resources can be scaled down or scheduled maintenance performed.

Events (what triggers transitions and why they matter)

- **User submits article/URL/text**

Trigger: Start processing — sends system from Idle → Content_Submission → Preprocessing.

Relevance: The external stimulus that drives the entire workflow.

- **Extract text / Clean data / Language detection (Preprocessing events)**

Trigger: Raw content converted to cleaned text.

Relevance: Ensures inputs meet the expected format for Feature_Extraction. A failure here → Error_State.

- **Tokenization / Embeddings generation (Feature extraction events)**

Trigger: Preprocessed text is encoded.

Relevance: Prepares structured features required by the ML model; errors here also lead to Error_State.

- **Run trained ML/NLP model (Model Inference event)**

Trigger: Model executes on embeddings and returns scores/labels.

Relevance: The decision-making action; its availability and latency are critical for UX.

- **Missing text / Invalid format (Error events)**

Trigger: Preprocessing fails or input invalid.

Relevance: Allows early rejection with meaningful feedback and prevents garbage-in/garbage-out.

- **Model not available / Timeout (Error events)**
Trigger: Model service down or overloaded.
Relevance: System can retry, switch to fallback, or notify operators — important for reliability SLAs.
- **Label as Fake / True / Uncertain (Classification event)**
Trigger: Model scores are mapped to labels based on thresholds.
Relevance: Determines subsequent path: uncertain → Human_Review; definite → Store_Result & Display_Result.
- **Highlight important text / Find evidence (Explanation generation event)**
Trigger: After classification, explanation routines run.
Relevance: Produces human-readable justification; critical for interpretability and user trust.
- **Save output to database (Store_Result event)**
Trigger: After explanation generation, results are persisted.
Relevance: Enables auditing, analytics, and training-data collection.
- **Return to user / UI (Display_Result event)**
Trigger: Results are presented in the front-end.
Relevance: User receives actionable feedback; UI should show label, confidence, and evidence.
- **If flagged or uncertain → Human review (Flagging event)**
Trigger: Classification yields Uncertain or confidence below threshold OR user flags result.
Relevance: Invokes human-in-the-loop moderation to improve accuracy and gather labels.
- **Manual review finishes → Update labeled dataset (Human Review completion event)**
Trigger: Fact-checker approves/rejects.
Relevance: Provides high-quality labels for the Retraining_Pipeline.
- **Trigger Retraining / Train and deploy improved model (Retraining event)**
Trigger: Enough new labeled data or scheduled retraining.
Relevance: Keeps model current; reduces errors caused by evolving misinformation tactics.
- **Model update deployed (Model_Update event)**
Trigger: New model promoted to production.
Relevance: Brings improvements into live inference and closes the MLOps loop.

- **Reset or retry (Recovery event from Error_State)**
Trigger: After operator action or automated retry policy.
Relevance: Restores normal flow without manual interruption when transient faults occur.
- **Wait for next input (Idle event)**
Trigger: After finishing processing, system returns to idle.
Relevance: Marks end of a processing cycle and readiness for the next submission.

Chapter 5: Interaction Modeling

SEQUENCE DIAGRAM:

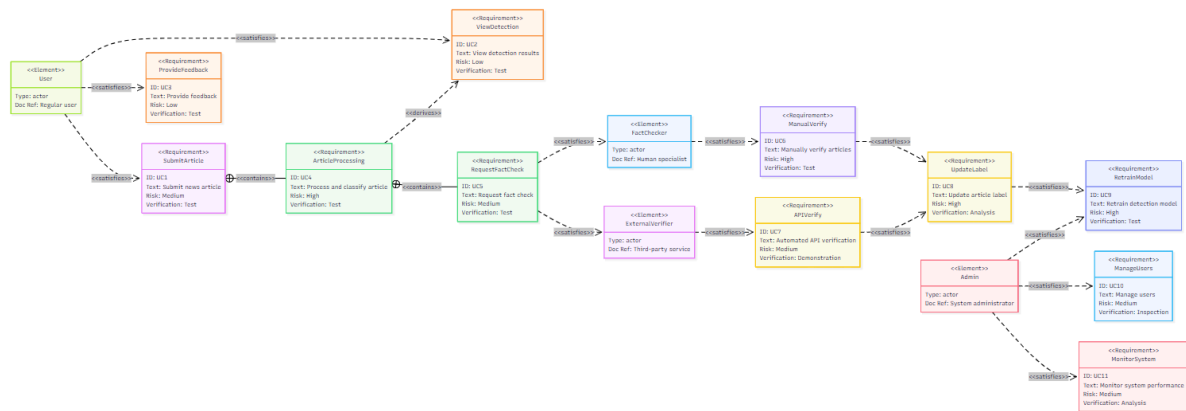
```

sequenceDiagram
    participant User
    participant Frontend
    participant Backend
    participant Preprocessor
    participant FeatureExtractor
    participant ModelService
    participant ExplanationEngine
    participant Database
    participant FactChecker

    User->>Frontend: Submit article/text
    Frontend->>Backend: POST/submit
    Backend->>FeatureExtractor: Tokenize and generate embeddings
    FeatureExtractor->>ModelService: Send embeddings for prediction
    ModelService->>ExplanationEngine: Store result and explanation
    ExplanationEngine->>Database: 
    Backend->>Preprocessor: Clean and extract text
    Preprocessor->>ModelService: Submit corrected label
    ModelService->>Database: Fetch article and model result
    ModelService-->>Backend: Return label + confidence
    Backend->>ExplanationEngine: Generate explanation and evidence
    ExplanationEngine-->>Backend: Confirmation
    Backend->>Frontend: Send final result
    Frontend->>User: Display label, confidence, evidence
    User->>FactChecker: Report incorrect label (optional)
    FactChecker->>ModelService: Trigger retraining (optional)
    ModelService->>Database: Update record
  
```


[illegible]

USE CASE DAIGRAM:




The flowchart illustrates the complete lifecycle of a fake news detection system, beginning with content submission from users, APIs, or external feeds. Once an article or text is submitted, the system validates the input and either returns an error for invalid or missing data or proceeds to preprocessing, where text is extracted, cleaned, and prepared for analysis. After preprocessing, the system performs feature extraction, converting text into tokenized and embedded representations suitable for machine learning. These features are passed into the model inference stage, where a transformer-based model analyzes the content and generates a prediction. If the model is unavailable or times out, an appropriate error is returned; otherwise, the system classifies the content as fake, true, or uncertain. When predictions are confident, the system generates explanations by highlighting key phrases and retrieving evidence, then stores the results in the database and displays them to the user. For uncertain or disputed cases, the workflow shifts to human review, where fact-checkers evaluate the article and make a final decision. Verified outcomes are fed into the retraining pipeline to help improve the model over time, while escalated cases may require additional human or administrative intervention. Administrators also handle model updates, rollbacks, system monitoring, and user management. Overall, the flowchart represents a robust, transparent, and continuously improving fake news detection ecosystem where automated intelligence and human judgment work together.

Chapter 6: UI Design with Screenshots


 **Fake News Detector**


Analyze news articles for credibility using advanced AI-powered fact-checking technology

 Enter News Article Text

At close, the Sensex was up 446.21 points or 0.52 percent at 85,632.68, and the Nifty was up 139.50 points or 0.54 percent at 26,192.15. About 1759 shares advanced, 2221 shares declined, and 165 shares unchanged.


In trade today, the Nifty 50 briefly approached the 26,250 zone, just shy of its all-time high, before settling slightly lower, under the 26,200 mark but still firmly in the green. The Nifty 50 topped the 26,200 for the first time in 14 months.

 Analyze Article

 **Trust Score**

Mostly Credible

85
out of 100

 Key Findings

Key Findings

- Reports specific, verifiable stock market data (Sensex and Nifty).
- Provides quantitative information about advancing, declining, and unchanged shares.
- Mentions specific milestones (Nifty 50 approaching 26,250) with context.

Positive Indicators

- ✓ Presents factual market data.
- ✓ Avoids sensationalism.
- ✓ Uses neutral language.

Recommendation


Cross-reference the reported figures with other financial news sources to confirm accuracy. The information is likely reliable, but verification is always a good practice.

Fake News Detector

Analyze news articles for credibility using advanced AI-powered fact-checking technology

Enter News Article Text

The biggest change to the Q3 Monochrom versus the original is that sensor. While it is only capable of capturing black and white photos, it still retains Leica's Triple Resolution technology that captures images at resolutions of 60, 36, and 18 megapixels across an ISO range from 100 to 200,000. It is also capable of recording monochromatic video at up to 8K resolution. Leica removed the color Bayer filter, and there is also no low-pass filter, which together results in images that Leica says are incredibly sharp, feature impressive dynamic range, and have "exceptional tonal depth."

 Analyze Article

Trust Score

Mostly Credible

85

out of 100

Key Findings

Key Findings

- The article describes technical specifications of a camera.
- Claims are specific and quantifiable (resolution, ISO range).
- The source is likely a tech publication or press release based on the language.

Positive Indicators

- ✓ Specific technical details are provided.
- ✓ Lack of sensationalism or emotional language.

Recommendation

Verify specifications with official Leica documentation for complete accuracy.

