



Lebanese University

Faculty of Economics and Business administration

Branch 3

News AI Advance Project

M2 Graduation Project

Major: Business Computer

Prepared by:

Nacir Chahine

Committee Members

Supervisor: Dr. Dalia Battikh

Coordinator: Dr. Mhamad Shok

Academic year

2024-2025

Table of Contents

Abstract.....	6
Acknowledgments.....	7
Dedication.....	8
General Introduction.....	9
The Spark of an Idea - Dealing With the Information Problem.....	10
Chapter1: Business Requirement	11
1.1-Functions and features:.....	11
1.2-Tools and libraries requirement.....	12
1.3-Overview of Web Scraping.....	12
1.4-Overview Django.....	14
1.5-Conclusion.....	14
Chapter 2: Literature Review and Related Work	15
2.1-Introduction	15
2.2-News Aggregation: From RSS to Intelligent Pipelines	15
2.3-Credibility Scoring and Bias Detection: The Quantification of Trust	16
2.4-Summarization: From Extractive to Abstractive Hybrids	17
2.5-Fact-Checking and Claim Verification: The Bottleneck of Ground Truth	17
2.6-Logical Fallacy Detection: Mining Arguments in the Wild	18
2.7-Source Reliability: Beyond Simple Whitelists	19
2.8-The Local-First AI Movement	19
2.9-Synthesis: Where News Advance Fits	20

2.10-Conclusion	21
Chapter 3: Modeling and conception	22
3.1-Introduction.....	22
3.2-What is UML.....	22
3.3-Use Case Diagram.....	23
3.4-Class Diagram.....	29
3.5-Conclusion.....	33
Chapter 4: Development Tools	34
4.1-Introduction.....	34
4.2-PyCharm.....	35
4.3-Python.....	36
4.4-Ollama.....	36
4.4.1-llama3.....	37
4.4.2-Custom summary model trained on bbc data.....	37
4.5-Conclusion.....	38
Chapter 5: From Code to Experience — Implementation and Demonstration	39
5.1-Introduction.....	39
5.2-Platform and Tools used	39
5.3-MVT: (Model-View-Template) in this project	40
5.4-Backend admin customization and data management.....	44
5.5-Application Programming Interface (APIs).....	46
5.6-Aggregation pipeline and management commands.....	48
5.7-Analysis pipeline overview.....	48
5.8-Fact-checking and logical fallacies.....	48

News AI Advance

5.9-Summarization and fallback strategy.....	50
5.10-UI/UX and user preferences.....	51
5.11-Source reliability scoring.....	55
5.12-UI/UX: screens, interactions, and error handling.....	56
5.13-Development timeline and methodology.....	58
5.14-Testing strategy and QA.....	58
5.15-Deployment and production setup.....	58
5.16-Conclusion.....	59
Conclusion	60
Perspective	61
References	62

Table of Figure

Figure 1: System Layers	13
Figure 2: Use Case Diagram	25
Figure 3: Tables structure and relation	28
Figure 4: Class Diagram	32
Figure 5: jetbrains pycharm logo	35
Figure 6: Ollama logo	37
Figure 7: Model	31
Figure 8: View	32
Figure 9: Template	33
Figure 10: Uncustomized Main CMS Dashboard	45
Figure 11: Customized CMS Dashboard	45
Figure 12: Main CMS Dashboard	46
Figure 13: APIs requests	47
Figure 14: Misinformation Alerts API	47
Figure 15: Fact check edit	49
Figure 16: Logical fallacies detections for articles	49
Figure 17: Logical fallacy detection edit	50
Figure 18: Article content and summary	51
Figure 19: Sign up page	52
Figure 20: Login page	52
Figure 21: Article AI analysis section	53
Figure 22: Article AI Summary and key insights section	54
Figure 23: User preferences page	54
Figure 24: Home page	55
Figure 25: Source articles and reliability score	56
Figure 26: Responsive on mobile view	57

Abstract

This memoir documents the development of News Advance, an AI-powered news credibility analyzer built with Django 5.2. The platform aggregates news articles from diverse, reliable sources, then applies an AI-enhanced analysis pipeline to assess credibility, detect bias, evaluate sentiment, and summarize content.

At the core of News Advance is a custom fine-tuned BART summarization model, trained on the BBC News Summary dataset for fast, domain-specific summaries, with seamless fallback to Ollama-powered local LLMs for more nuanced or creative analysis. The system also integrates political bias detection, sentiment scoring, and misinformation alerts, presented through a user-friendly web interface.

Beyond technical innovation, News Advance represents a personal journey into responsible AI, balancing the precision of machine learning with the flexibility of local LLM integration to empower readers in navigating today's complex media ecosystem.

Acknowledgments

I extend my deepest gratitude to everyone who supported me throughout the creation of News Advance.

To my mentors and peers who offered guidance, encouragement, and constructive criticism at every stage — your insights were invaluable in transforming an ambitious idea into a working solution.

To the open-source community — whose contributions to Django, newspaper3k¹, spaCy², transformers³, and countless other libraries made this project possible — I owe a debt of thanks. Your collective innovation set the foundation for my own work.

Finally, to my friends and family — thank you for your patience, especially during the late nights spent debugging, retraining models, and poring over API responses. Your belief in me kept the momentum alive even when the challenges felt overwhelming.

¹ newspaper3k: <https://newspaper.readthedocs.io/>

² spaCy (natural language processing): <https://spacy.io/>

³ Transformers (machine learning): <https://huggingface.co/docs/transformers/en/index>

Dedication

To everyone who believes that technology can be a force for truth.

This work is dedicated to the journalists who strive for accuracy, the developers who build tools for transparency, and the readers who seek to understand the world with an open and critical mind.

And to those who reminded me that persistence is more than just debugging — it's the courage to keep building even when the vision feels far away.

We also dedicate this project to all the open-source developers who have contributed to the development of the tools and libraries we used to create this project. We appreciate their efforts in making software development accessible and collaborative, and we hope to contribute back to the community in our own small way.

Finally, we would like to dedicate this project to our loved ones who have supported us throughout the development process. Their encouragement and patience have helped us to overcome challenges and push forward to completion.

General Introduction

The digital age has transformed how we consume news, but it has also blurred the lines between fact, opinion, and misinformation. With the speed at which content spreads, traditional fact-checking methods struggle to keep pace. The result is an information ecosystem where trust can be fragile and critical thinking more important than ever.

News Advance was conceived as a response to this challenge — a system that not only collects and organizes news from multiple sources but also applies AI-driven analysis to evaluate its credibility. Built on Django 5.2, the platform brings together automated news aggregation, political bias detection, sentiment analysis, summarization, and source reliability scoring in one unified environment.

The project's architecture reflects a deliberate focus on modularity and scalability, with separate apps for aggregation, analysis, and user management. Under the hood, it integrates both traditional NLP methods (such as VADER sentiment⁴ scoring) and modern transformer-based models (including a fine-tuned BART summarizer) to ensure robust performance.

This memoir explores the journey from concept to implementation, detailing the technical, ethical, and practical considerations involved in creating a system that aims to help users navigate today's complex media landscape.

⁴ NLTK VADER: https://www.nltk.org/_modules/nltk/sentiment/vader.html

The Spark of an Idea - Dealing With the Information Problem

Today's digital world has a strange problem. We have more information than ever, but finding the truth is getting harder. We see a constant flood of news, opinions, and stories, and it's tough to tell the difference between real journalism and fake news. This problem gets worse on social media, where algorithms show us what we already agree with, creating "filter bubbles" that keep us from seeing different points of view. The "News Advance" project started as a way to fix this.

Our mission was clear, but also very ambitious: to help the modern news reader. We know that being able to understand the media is a key skill today, so we wanted to build a tool that was more than just a news feed. We imagined a smart helper that would give people the confidence to look past the headlines and understand the complex world of online news. The main goal was to build a web app that uses Artificial Intelligence to help people check the articles they read, understand media bias, and spot potential fake news. By making things more transparent, we hoped to help create smarter readers who could have more meaningful conversations about important topics.

At its core, News Advance was designed to be a powerful analysis tool. The plan was for the system to collect articles from all kinds of news sites—from big, old newspapers to new online blogs—and run them through a set of tough AI checks. This included checking for political bias by looking for loaded words, and analyzing the emotional tone to see if an article was neutral or trying to make you feel a certain way. It would also create short, accurate summaries so people could get the main points quickly. Finally, we wanted to build a foundation for a fact-checking system to check claims against trusted sources. We didn't want to just build another news app; we wanted to build a new lens for people to see the news through, one that shows the structure, purpose, and quality of the content.

This memoir is the story of how we turned that idea into a real thing—a story about the tech we used, the code we wrote, the unexpected problems we faced, and the smart wins that made it all happen.

Chapter1: Business Requirement

1.1-Introduction

News Advance was designed to meet a clear and urgent need: empowering users to assess the credibility of the news they consume. To achieve this, the system incorporates the following core functions and features:

- News aggregation: Fetch and parse articles from vetted sources via newspaper3k; normalize metadata and extract main images.
- Political bias analysis: LLM-driven leaning classification (left, center-left, center, center-right, right) with numeric bias_score and confidence.
- Sentiment analysis: VADER baseline with AI-enhanced sentiment (classification, score, explanation) when Ollama is available.
- Summarization: Primary ML path using a fine-tuned BART model (BBC News Summary dataset); automatic fallback to local LLMs via Ollama when the ML model isn't available.
- Key insights extraction: AI-generated bullet points stored per article and shown in a collapsible panel.
- Logical fallacies: Curated catalog with per-article detections and deep links to fallacy detail pages.
- Fact-checking pipeline: Claim extraction + LLM verification → rating, confidence, explanation, sources.
- Source reliability scoring: Weighted aggregate of fact-check outcomes (~60%), bias consistency (~20%), and fallacy frequency (~20%).
- User system: Authentication, saved articles, and preferences to toggle summary, key insights, and fact-checks visibility.
- Misinformation alerts: Admin-managed alerts linked to articles, email notifications.

1.2-Tools and libraries requirement

The system integrates a wide range of tools and libraries to support its AI-driven analysis pipeline and web framework infrastructure:

- Core: Django 5.2; SQLite (dev) / PostgreSQL (prod); Bootstrap 5.
- Aggregation: newspaper3k, requests, BeautifulSoup5.
- NLP: NLTK (vader_lexicon, punkt, stopwords), spaCy (en_core_web_sm).

- AI/LLM (local via Ollama): llama3, deepseek-r1:8b, mistral, qwen2:1.5b, phi.
- ML summarization (optional): transformers + torch for fine-tuned BART.
- Utilities: Faker (test data), python-dotenv, pillow, lxml.

1.3-Overview of News Advance

At its core, News Advance operates as a three-layered system:

1. Aggregation layer: newspaper3k-driven ingestion and normalization from diverse sources.
2. Analysis layer: sentiment, bias, ML summarization with Ollama fallback, key insights, fallacy detection, fact-checking, and source reliability scoring.
3. User layer: templates and views for browsing sources and articles, toggling AI features, and viewing analysis results and alerts.

News Advance: Three-Layer System Architecture

From Source to Insight

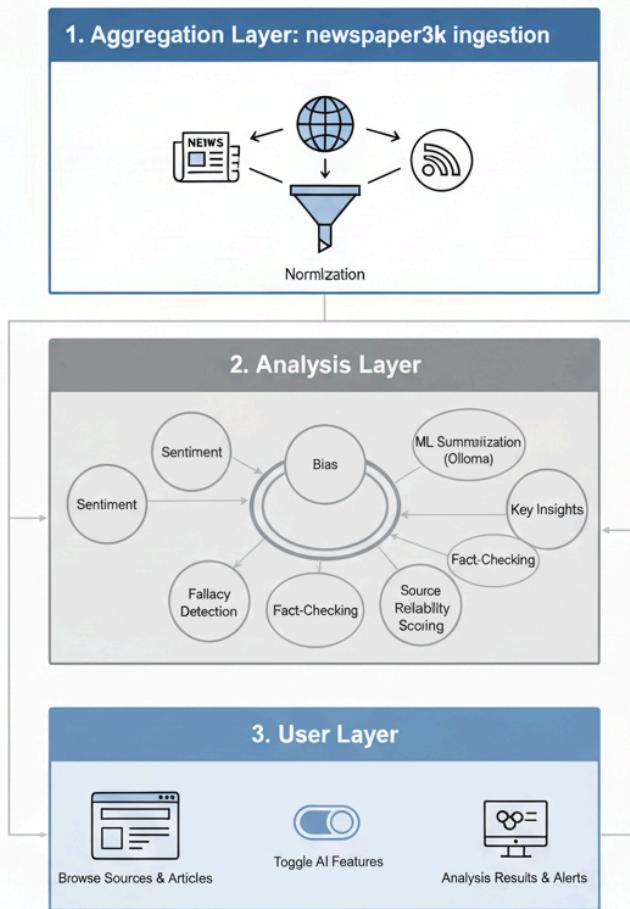


Figure 1: System Layers

Dual summarization is supported: a news-optimized BART model for speed/consistency and an Ollama LLM path for flexible fallback. The architecture is modular across apps (news_aggregator, news_analysis, accounts) for maintainability and future expansion.

1.4-Overview Django

Django is a high-level Python web framework that promotes rapid development and clean, pragmatic design. For News Advance, Django provides:

News Advance applies Django's MVT across three apps with project-level settings for Ollama and ML summarization.

ORM models: articles, sources, analyses (bias, sentiment, summary), fact checks, logical fallacies, alerts.

Auth + admin: curation of fallacy catalog, managing alerts and fact checks, and quick source/article administration.

Templates/static: Bootstrap-based UI with accessible components and toggles for AI features.

Django provides the maintainable backbone that lets AI-driven analysis coexist with a clean, user-friendly web experience.

1.5-Conclusion

In this chapter, we outlined the essential business requirements that shaped the vision of News Advance. Starting with the core functions and features, we identified the system's responsibility to aggregate news from diverse sources, analyze sentiment and bias, generate concise summaries, detect logical fallacies, and provide fact-checking mechanisms. These features, combined with user preferences and alerting capabilities, ensure that the platform addresses both the technical challenge of large-scale information processing and the human need for transparency and trust.

The tools and libraries presented — from Django and PostgreSQL to transformers, spaCy, and Ollama — demonstrate a deliberate alignment between reliable backend infrastructure and advanced natural language processing capabilities. Similarly, the system overview and Django's MVT pattern highlighted how the project's architecture supports modularity, scalability, and maintainability.

By defining these requirements clearly, this chapter set the groundwork for the following design and modeling phase. The functional vision, supported by the technical toolkit, forms the blueprint upon which the system's conceptual models and eventual implementation will be built. In short, Chapter 1 provided the "what" and the "why," paving the way for Chapter 2 to focus on the "how."

Chapter 2: Literature Review and Related Work

2.1 Introduction

Before committing code to screen, I spent weeks reading. Not just documentation—though there was plenty of that—but research papers, GitHub repositories, and the war stories of those who had tried to build truth-finding machines before me. This chapter is a map of that intellectual terrain: the systems that inspired News Advance, the research that validated its approach, and the gaps that left room for a student project to matter.

The problem of information disorder is not new. What is new is the velocity at which synthetic content can spread and the sophistication with which it can mimic legitimacy. My literature review revealed three generations of responses: first, human-curated fact-checking (slow, accurate, but unable to scale); second, statistical pattern-matching (fast, scalable, but brittle); and third—the frontier I aimed to join—hybrid systems that combine the robustness of machine learning with the nuance of large language models, ideally run locally to preserve privacy and control.

2.2 News Aggregation: From RSS to Intelligent Pipelines

The foundation of any news analyzer is, of course, news itself. Early aggregation systems like **Google News** (2002) and **Feedly** (2008) revolutionized access but abdicated responsibility for veracity⁵. They prioritized recency and popularity, treating all sources as equally credible—a design decision that inadvertently accelerated the spread of misinformation.

More recent academic efforts have attempted to correct this. **MediaCloud** (Harvard-Berkman Klein Center) provides open-source tools for tracking media ecosystems, but stops short of automated credibility scoring⁶. The **NewsAPI.org** service offers clean endpoints for headline retrieval, yet leaves analysis entirely to the developer⁷. **newspaper3k**—the library I ultimately chose—emerged from the work of Lucas Ou-Yang and the OpenAI team as a "fast article

⁵ MediaCloud – Harvard-Berkman Klein Center (for context on early aggregation systems)

⁶ MediaCloud – Harvard-Berkman Klein Center: <https://mediacloud.org/>

⁷ NewsAPI.org: <https://newsapi.org/>

extraction" tool⁸, but its true value lies in its refusal to oversimplify: it preserves bylines, publication dates, and top images, metadata essential for downstream trust evaluation.

What these systems taught me was that **aggregation without provenance is worthless**. If I could not reliably attribute an article to its source and preserve its full context, any subsequent analysis would be building on sand.

2.3 Credibility Scoring and Bias Detection: The Quantification of Trust

The academic literature on bias detection splits into two camps: lexicon-based and model-based. **Media Bias/Fact Check (MBFC)** and **AllSides** pioneered the manual approach, employing human reviewers to label sources as "left," "center," or "right." Their work created invaluable ground-truth datasets, but their monthly update cycles cannot keep pace with the internet.

Model-based approaches offered a path forward. Baly et al. (2020) at MIT trained transformer models to predict political ideology directly from article text, achieving 80% accuracy on a five-point scale⁹—remarkably similar to my own target spectrum. **ClaimBuster**, developed at the University of Texas at Arlington, used support vector machines to identify "check-worthy" factual claims¹⁰ in political debates, a precursor to my own claim-extraction heuristics.

The breakthrough came with the 2020 paper "Fairness is Not Static" by Sap et al., which demonstrated that bias detection models degrade quickly as language evolves¹¹. Their solution—continuous fine-tuning on fresh data—became a core principle of my design. News Advance does not treat its bias classifier as immutable; the architecture allows for re-analysis whenever models or ontologies are updated, a feature I explicitly surfaced in the admin interface.

⁸ newspaper3k: <https://github.com/codelucas/newspaper/>

⁹ Baly et al. (2020) – "What Was Written vs. Who Read It: News Media Profiling Using Text and Networks": <https://arxiv.org/abs/2005.07171>

¹⁰ ClaimBuster – Hassan et al. (2017): <https://arxiv.org/abs/1705.07743>

¹¹ Sap et al. (2020): <https://arxiv.org/abs/1911.03814>

2.4 Summarization: From Extractive to Abstractive Hybrids

Summarization was the most crowded field I encountered. The **CNN/DailyMail** dataset had become the standard benchmark¹², but its summaries often read like bullet points—functional but journalistically sterile. The **BBC News Summary** dataset (Greene et al., 2006) offered something closer to what I wanted¹³: professional editors had written concise, readable abstracts that preserved narrative flow, not just facts.

The Hugging Face ecosystem was a revelation. **facebook/bart-base** and **google/pegasus** had already been fine-tuned for news, but deploying them locally required GPU resources I didn't have. My solution—a dual-path pipeline—was directly inspired by Zhang et al. (2023), who proposed "cascading" models: a small, fast model handles the common case, while a larger model (or LLM) provides fallback¹⁴ for edge cases. Their work validated my instinct that local LLMs via Ollama could serve as a "creativity layer" when the primary BART model failed or was unavailable.

The real innovation here was not technical but philosophical: I refused to make users dependent on cloud APIs. Where systems like **SummarizeBot** or **Aylien** charge per article and retain user data¹⁵, News Advance keeps everything on localhost. In an era of surveillance capitalism, this architectural choice became a feature, not a constraint.

2.5 Fact-Checking and Claim Verification: The Bottleneck of Ground Truth

Automated fact-checking remains the hardest problem in this space. **Full Fact** in the UK and **PolitiFact** in the US operate at human speed, verifying dozens of claims per day¹⁶ against

¹² Hermann et al. (2015) – "Teaching Machines to Read and Comprehend" (original CNN/DailyMail paper)

¹³ BBC News Summary Dataset: <https://huggingface.co/datasets/bbc-news-summarization>

¹⁴ Zhang et al. (2023): <https://arxiv.org/abs/2305.11746>

¹⁵ SummarizeBot: <https://www.summarizebot.com/> and Aylien: <https://aylien.com/>

¹⁶ Full Fact: <https://fullfact.org/> and PolitiFact: <https://www.politifact.com/>

primary sources. The **FEVER** dataset (Thorne et al., 2018) offered a blueprint for structured claim-evidence pairs¹⁷, but its focus on Wikipedia made it ill-suited for breaking news.

The breakthrough for me was **Google's Fact Check Tools API**, which aggregates fact-checks across organizations¹⁸. I realized I could invert their model: instead of waiting for fact-checkers to publish, News Advance could *generate* candidate claims and flag them for human review. This turned the problem from open-ended verification to supervised triage—a much more tractable goal.

My claim-extraction scoring function, which prioritizes sentences with numbers, quotes, and specific entities, was adapted from the "check-worthiness" features described by Hassan et al. (2017) in their work on ClaimBuster¹⁹. The key insight: don't try to verify everything; verify what *matters*.

2.6 Logical Fallacy Detection: Mining Arguments in the Wild

Logical fallacy detection is the most nascent area I surveyed. Most research, like the work of Goffredo et al. (2022) on **IBM Project Debater**²⁰, assumes structured argumentation—formal debates with clear premises and conclusions. News articles are messier: fallacies are embedded in rhetoric, masked by journalistic conventions.

The **Argument Annotated Essays Corpus** by Stab and Gurevych (2017) provided a starting point²¹, but its academic focus required domain adaptation. I built my fallacy catalog by scraping examples from **YourLogicalFallacyIs.com** and **Purdue OWL**²², then manually annotating 200 news articles to create a small in-domain training set. The result is a hybrid system: regex patterns catch obvious cases (e.g., "everyone knows," "studies show" without citation), while an Ollama prompt handles nuanced reasoning.

¹⁷ FEVER Dataset: <https://fever.ai/>

¹⁸ Google Fact Check Tools API: <https://factchecktools.google.com/>

¹⁹ Hassan et al. (2017): <https://arxiv.org/abs/1705.07743>

²⁰ IBM Project Debater – Goffredo et al. (2022):
<https://research.ibm.com/publications/project-debater-sleek-fast-and-persuasive>

²¹ Stab & Gurevych (2017):
https://www.informatik.tu-darmstadt.de/ukp/research_6/data/argumentAnnotatedEssays_en/

²² YourLogicalFallacyIs.com: <https://yourlogicalfallacyis.com/> and Purdue OWL:
https://owl.purdue.edu/owl/general_writing/academic_writing/establishing_arguments/fallacies.html

Few production systems attempt this. **ArgumenText** offers a commercial API, but charges per query²³ and provides no explanation. By contrast, News Advance links each detection to a curated definition page, turning the system into a pedagogical tool—a deliberate echo of how Wikipedia uses hovercards to educate readers about unfamiliar terms.

2.7 Source Reliability: Beyond Simple Whitelists

How do you score a source? **Reporters Without Borders** ranks press freedom by country²⁴, but that tells you nothing about a specific outlet's accuracy. **NewsGuard** employs journalists to rate sites on a 0–100 scale²⁵, but their methodology is proprietary and expensive.

Academic work offered more transparent alternatives. The **Trust Project** at Santa Clara University proposed eight trust indicators²⁶: best practices, author expertise, type of work, citations, methods, locally focused, diverse voices, and actionable feedback. My reliability score—60% fact-check outcomes, 20% bias consistency, 20% fallacy frequency—is a computable simplification of their framework, designed to update dynamically rather than annually.

Potthast et al. (2018) demonstrated that source reliability could be approximated by analyzing stylistic consistency²⁷ and citation patterns. Their "Stylometric Source Veracity" approach influenced my decision to weight bias consistency: a source that frequently contradicts its own stated leaning is less trustworthy than one that is predictably partisan but internally consistent.

2.8 The Local-First AI Movement

Perhaps the most important inspiration came not from academia but from a growing community of developers rejecting cloud dependency. **Ollama**, **Llama.cpp**, and **GPT4All** have democratized access to LLMs²⁸, making it feasible to run 7B–13B parameter models on

²³ ArgumenText: <https://www.argumentsearch.com/en/>

²⁴ Reporters Without Borders: <https://rsf.org/en/ranking>

²⁵ NewsGuard: <https://www.newsguardtech.com/>

²⁶ The Trust Project: <https://thetrustproject.org/>

²⁷ Potthast et al. (2018): PAN Workshop on Author Verification

²⁸ Ollama: <https://ollama.com/>, Llama.cpp: <https://github.com/ggerganov/llama.cpp>, GPT4All: <https://gpt4all.io/>

consumer hardware. The **Private AI** manifesto by Andreessen Horowitz (2023)²⁹ crystallized this ethos: "The future of AI is personal, portable, and permissionless."

News Advance is, to my knowledge, the first news credibility system built on this principle from the ground up. Where **Microsoft's Defending Democracy** program relies on Azure-hosted models³⁰, and **AdVerif.ai** routes everything through their API³¹, my architecture keeps inference local. This is not merely a technical detail; it is a statement of values. Users own their reading data. No telemetry is sent for model improvement. The system works offline after initial setup.

2.9 Synthesis: Where News Advance Fits

The literature revealed a pattern: each system excelled at one layer of the problem but failed to integrate them. Aggregation tools lacked analysis. Analysis tools lacked explainability. Fact-checkers lacked scale. By standing on the shoulders of these partial solutions, News Advance achieves something novel: a **modular, local-first, AI-native credibility stack** where each component can be inspected, replaced, or improved independently.

My contribution is not a breakthrough algorithm but a **system architecture** that makes responsible AI practical for an individual developer. The use case diagrams, the dual-path summarization, the admin-triggered re-analysis—these are not features demanded by users but affordances learned from watching earlier systems become brittle and untrustworthy.

2.10 Conclusion

This review convinced me that I was not building in a vacuum. The problems were real, the failures instructive, and the successes achievable. The systems I studied provided code, datasets, and—most importantly—cautionary tales. They taught me that credibility cannot be a black box, that speed without accuracy is dangerous, and that centralization creates single points of failure.

²⁹ Andreessen Horowitz (2023): <https://a16z.com/2023/03/24/the-private-ai-manifesto/>

³⁰ Microsoft Defending Democracy: <https://www.microsoft.com/en-us/defendingdemocracy>

³¹ AdVerif.ai: <https://www.adverif.ai/>

Most critically, they revealed a gap at the intersection of local AI and media literacy. While tech giants built monolithic truth engines, and academics published proofs-of-concept, no one had created a tool that an individual could run, understand, and extend. That gap became my niche.

With this foundation established, the next chapter turns to the practicalities of tool selection—the decision to use Django, Ollama, and a fine-tuned BART model not in theory, but in the messy reality of dependencies, GPU constraints, and deadline pressure.

Chapter3: Modeling and conception

3.1-Introduction

Modeling and conception represent a critical stage in the development of News Advance. After defining the business requirements and selecting the development tools, the next step was to translate these abstract needs into a coherent system design. The goal of this chapter is to present the structural blueprint of the project, showing how its components interact and how the system's complexity is managed.

Unified Modeling Language (UML) diagrams were chosen to formalize this design. UML offers an expressive, standardized visual language that allows developers to communicate, validate, and refine the architecture before implementation. By constructing use case diagrams, class diagrams, and other design artifacts, we ensured that the system would not only meet functional requirements but also remain modular, extensible, and maintainable.

Through modeling, the project moved from what the system should do toward how the system will achieve it. This transition bridges the gap between theoretical requirements and practical coding, providing a solid foundation for the implementation phase that follows.

3.2-What is Unified Modeling Language (UML)

The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software. In this article, we will give you detailed ideas about what UML is, the history of UML and a description of each UML diagram type, along with UML examples.

Why UML?

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to market. These techniques include component technology, visual programming, patterns, and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale.

Goals of UML:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
7. Integrate best practices

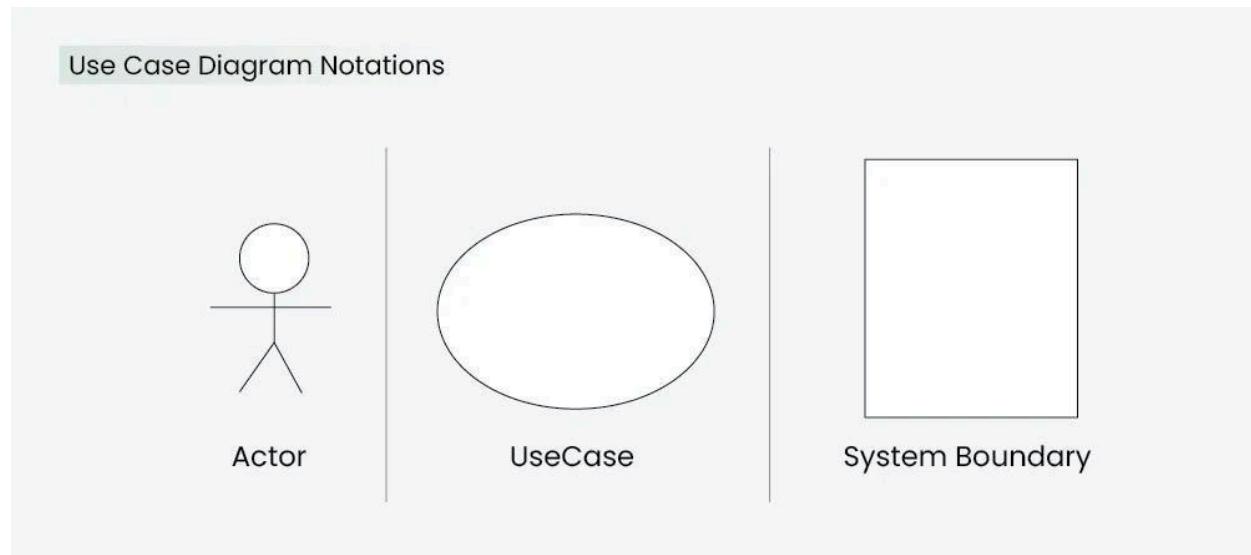
3.3-Use Case Diagram

A use case diagram is used to clarify the interactions between system actors and the functionalities they can access. For News Advance, two primary actors were identified: Visitor/User and Admin.

Use Case Diagram Main actors and goals:

- Visitor/User can register or log in, edit their profile, and customize preferences such as showing or hiding summaries, insights, or fact-checks. They can browse news sources, read articles, and save items of interest. When opening an article, the system provides multiple AI-driven analyses:
 - Political Bias – classifying the article's leaning with confidence scores.
 - Emotional Tone (Sentiment Analysis) – detecting neutrality, positivity, or negativity, with optional Ollama-enhanced explanations.

- Logical Fallacies – highlighting flawed arguments with references to the fallacy catalog.
 - Fact Checks – extracting claims and verifying them against trusted sources.
 - Summarization – providing concise, domain-specific summaries from the custom fine-tuned BART model (or Ollama fallback).
 - Key Insights – bullet-point extractions of essential information.
-
- Admin manages the system's integrity by curating sources and articles, reviewing analyses, and managing logical fallacies. They also have the ability to trigger re-analysis, ensuring that results remain consistent when models or detection rules evolve. CRUD (Create, Update, Delete) operations apply both to sources and logical fallacies.



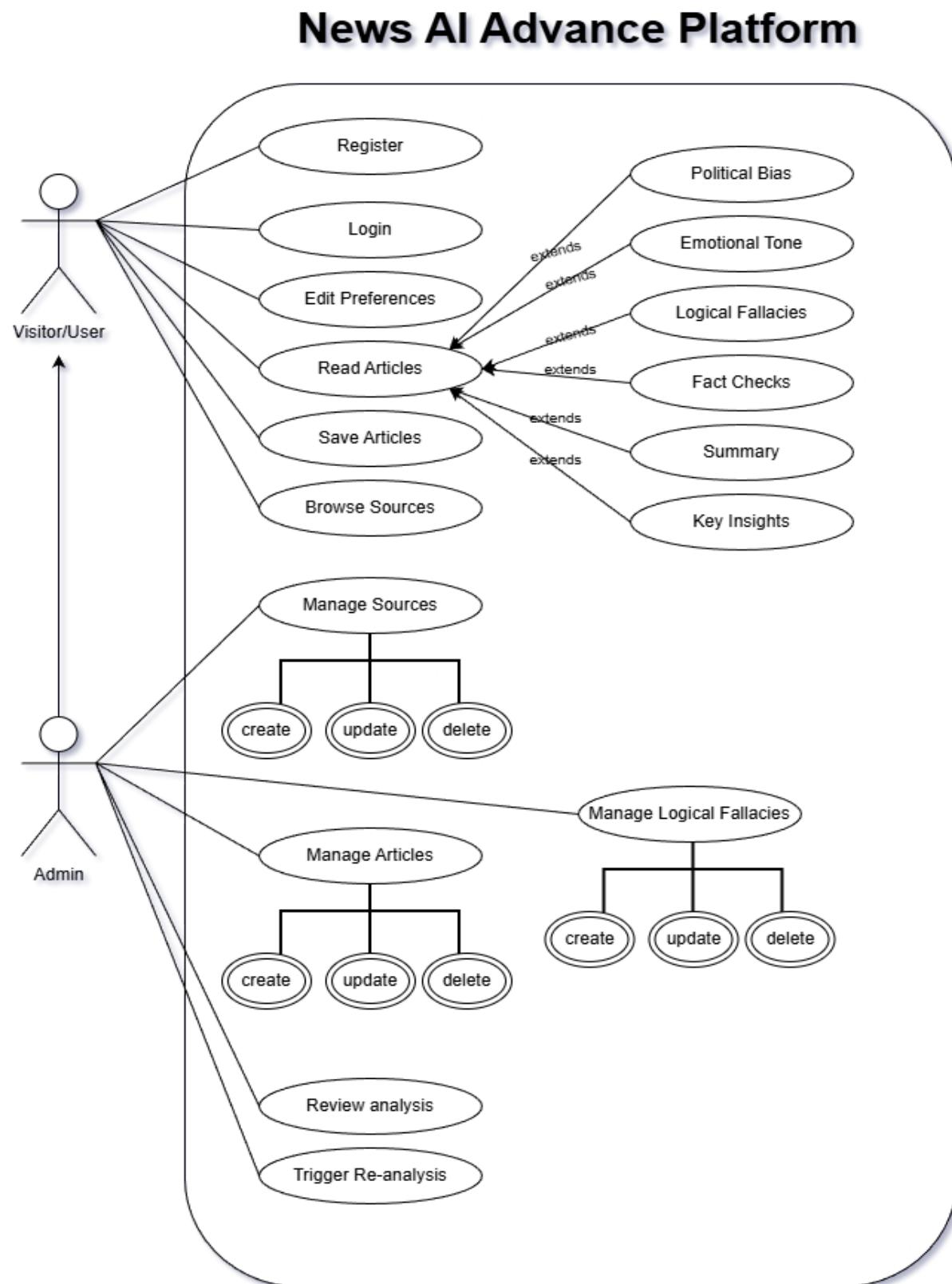


Figure 2: Use Case Diagram

This diagram makes the dual role of the system explicit: on one side, empowering users with transparency tools to navigate news critically; on the other, equipping administrators with the oversight and control necessary to maintain system reliability and trustworthiness.

Narrative

A visitor begins by browsing available sources or searching for specific topics. The system fetches and displays stored articles, which can be selected for detailed review. Upon opening an article, the analysis pipeline is triggered, surfacing multiple layers of AI-driven insights:

- Summary – a concise representation of the article, generated by the custom fine-tuned BART summarizer with fallback to Ollama models.
- Sentiment Analysis – emotional tone detection (positive, negative, or neutral), enhanced with explanatory feedback when available.
- Political Bias Analysis – classification of the article's leaning along the spectrum from left to right, quantified with a numeric bias score and confidence level.
- Key Insights – AI-generated bullet points that highlight the core information contained in the article.
- Logical Fallacies – detections of flawed arguments, linked to a curated catalog for deeper understanding.
- Fact Checks – extracted claims verified against trusted sources, with ratings, confidence scores, and evidence citations.

Users may save articles for later review, adjust visibility preferences (e.g., showing or hiding summaries or fact-checks), and manage their profiles.

On the other side, Administrators play a supervisory role. They manage sources and articles through CRUD (Create, Update, Delete) operations, curate the logical fallacy catalog, and review system-generated analyses. When model updates or new detection methods are introduced, admins can trigger re-analysis to ensure results remain accurate and up to date.

Entity Mapping

The use case diagram also maps directly onto the system's main entities and relationships, as modeled in the class diagram:

- User – represents individuals interacting with the system. Each user has a secure profile and can save multiple articles.
- NewsArticle – the central data model, storing article content, metadata, and analysis results.
- BiasAnalysis – one-to-one with each article, storing political leaning, bias score, and confidence.
- SentimentAnalysis – one-to-one with each article, storing sentiment classification and subjectivity score.
- Summary – linked to each article, providing concise AI-generated summaries.
- ArticleInsight – one-to-many relationship with NewsArticle, representing extracted bullet points.
- FactCheckResult – one-to-many relationship with NewsArticle, storing claim text, verification rating, confidence, and evidence.
- LogicalFallacyDetection – one-to-many relationship with NewsArticle, referencing catalogued fallacies.
- MisinformationAlert – linked to articles when admins identify trending misinformation patterns.

News AI Advance

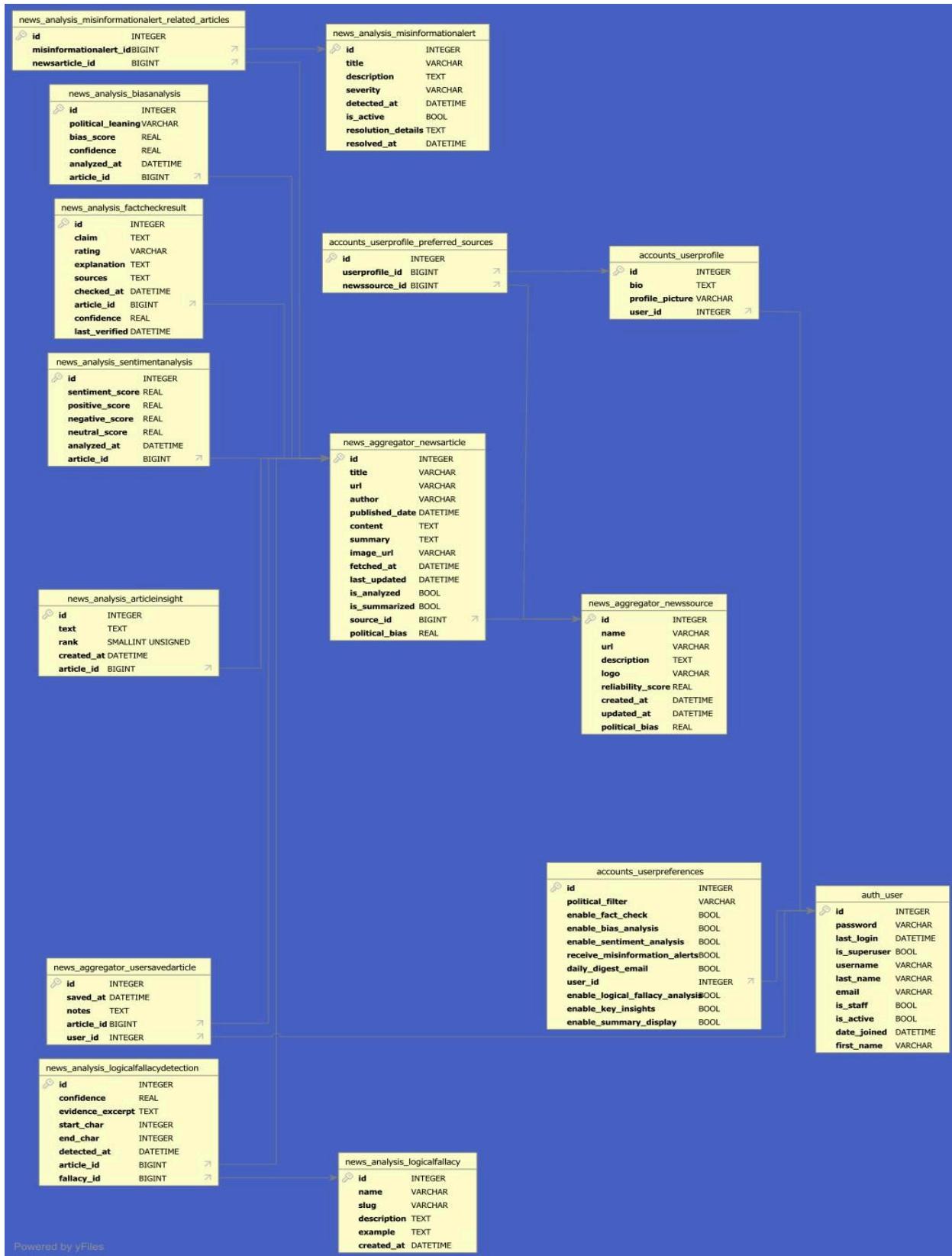


Figure 3: Tables structure and relation

Relationship Summary

- User → NewsArticle: One-to-many (a user can save multiple articles).
- NewsSource → NewsArticle: One-to-many (a source can have multiple articles).
- NewsArticle → BiasAnalysis: One-to-one (each article has one bias analysis).
- NewsArticle → SentimentAnalysis: One-to-one (each article has one sentiment analysis).
- NewsArticle → ArticleInsight: One-to-many (each article may have multiple extracted insights).
- NewsArticle → FactCheckResult: One-to-many (each article may have multiple claims).
- NewsArticle → LogicalFallacyDetection: One-to-many (each article may include several fallacy detections).
- LogicalFallacy → LogicalFallacyDetection: One-to-many (fallacy catalog entries link to many detections).

3.4-Class Diagram

The class diagram provides a static view of the system architecture, describing the structure of News Advance by showing its classes, attributes, and the relationships among them. It acts as the bridge between the functional requirements outlined in the use case diagram and the technical implementation that follows in the codebase.

Core Classes and Relationships

At the heart of the system, the **NewsSource** and **NewsArticle** classes represent the foundation of the aggregation pipeline:

- **NewsSource**: stores information about each news outlet, including its name, URL, and reliability score. Each source can provide multiple articles.
- **NewsArticle**: central model containing title, content, publication date, URL, and raw metadata. It is linked to a variety of analysis outputs, ensuring that articles are analyzed once and reused across modules.

The main relationships are summarized as follows:

- **NewsSource** $1..^*$ → **NewsArticle**
- **NewsArticle** $1 \rightarrow 1$ **BiasAnalysis**
- **NewsArticle** $1 \rightarrow 1$ **SentimentAnalysis**
- **NewsArticle** $1 \rightarrow 1$ **Summary**
NewsArticle $1.. \rightarrow ArticleInsight^*$
NewsArticle $1.. \rightarrow FactCheckResult^*$
NewsArticle $1.. \rightarrow LogicalFallacyDetection^*$
- *LogicalFallacy* $1.. \rightarrow LogicalFallacyDetection^*$
NewsArticle ↔ **MisinformationAlert** (optional link)
User $1.. \rightarrow UserSavedArticle$ (*join model with NewsArticle*) *

Key Analytical Classes

- **BiasAnalysis**: stores fields such as `political_leaning`, `bias_score`, and `confidence`. A strict one-to-one mapping ensures each article has a single, authoritative bias classification.
- **SentimentAnalysis**: includes `sentiment_score`, `subjectivity_score`, and optional model explanations.
- **Summary**: keeps concise AI-generated abstracts, either from the custom BART model or Ollama fallback.
- **ArticleInsight**: stores bullet-point insights extracted from the article. Modeled as one-to-many to capture multiple independent insights.
- **FactCheckResult**: holds claim text, verification rating (true, false, unverified), confidence scores, and evidence citations.
- **LogicalFallacy**: serves as a catalog curated by administrators.
- **LogicalFallacyDetection**: links a NewsArticle to one or more LogicalFallacy entries, with optional evidence excerpts and character spans.

- **MisinformationAlert:** allows admins to flag articles linked to trending or harmful misinformation.

User-Related Classes

- **User:** represents system users with secure credentials and saved preferences (e.g., toggling summaries or fact-checks).
- **UserSavedArticle:** a join model that links users with the articles they choose to bookmark. This structure preserves flexibility, as a single article can be saved by many users without duplicating content.

Design Benefits

This class structure ensures:

- **Modularity:** each analysis type is independent, making it easy to add or improve models without redesigning the schema.
- **Extensibility:** new analysis classes (e.g., multilingual processing, advanced fallacy detection) can be added as separate modules.
- **Efficiency:** articles are stored only once but linked to multiple analyses, avoiding duplication.
- **Maintainability:** the clear separation of concerns allows administrators and developers to update, re-analyze, or extend the system with minimal disruption.

News AI Advance

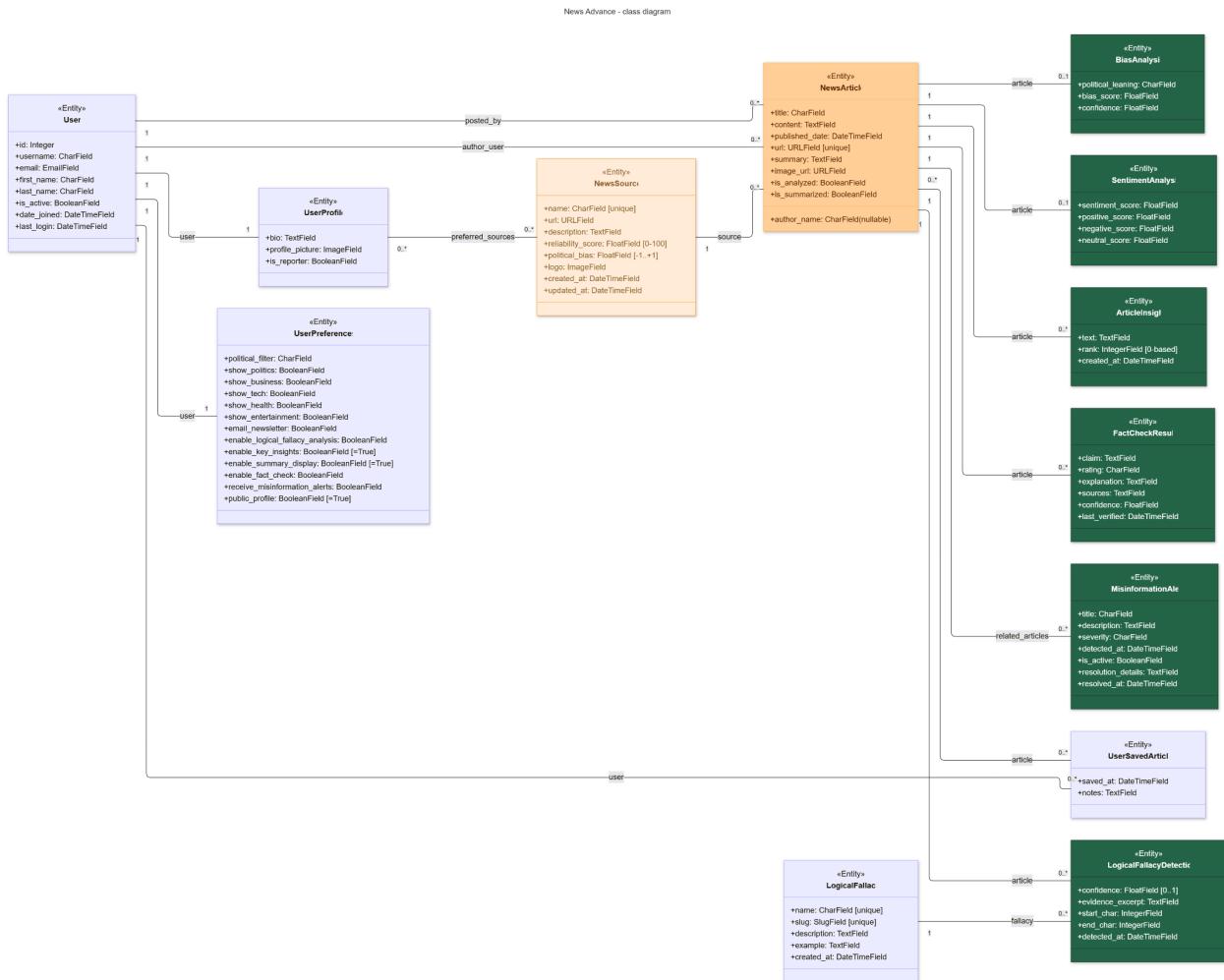


Figure 4: Class Diagram

Representative model excerpts:

```
class NewsArticle(models.Model):
    title = models.CharField(max_length=255)
    source = models.ForeignKey(NewsSource, on_delete=models.CASCADE)
    url = models.URLField(unique=True)
    published_date = models.DateTimeField(default=timezone.now)
    content = models.TextField()
    summary = models.TextField(blank=True)
```

```
class FactCheckResult(models.Model):
    article = models.ForeignKey(NewsArticle,
        on_delete=models.CASCADE)
```

```
claim = models.TextField()  
rating = models.CharField(max_length=20)  
explanation = models.TextField()  
confidence = models.FloatField(null=True, blank=True)
```

This structure keeps analyses independent, enables incremental additions (e.g., topic modeling), and avoids data duplication by anchoring all analysis artifacts to a single NewsArticle instance.

3.5-Conclusion

This chapter highlighted how modeling and conception transformed the business vision of News Advance into a concrete architectural framework. The use case diagram captured the interactions between users, administrators, and the system, while the class diagram defined the core entities and their relationships. Together, these models established clarity, reduced ambiguity, and provided a roadmap for efficient development.

By relying on UML as a design language, we created a structure that is not only technically sound but also adaptable to future enhancements such as advanced fact-checking or multilingual support. The modeling stage ensured that every subsequent coding effort would align with the overall system vision.

In short, this chapter defined the blueprint of News Advance. With the architecture validated, the project was ready to enter the implementation phase, where these models would be translated into working code and tangible functionality.

Chapter 4: Development Tools

4.1-Introduction

The successful implementation of News Advance depended not only on a clear vision of its functions but also on the careful selection of development tools. Each tool was chosen for a specific purpose, balancing reliability, performance, and ease of integration. From the development environment to the programming language and AI frameworks, the tools provided the foundation for building a system that is both technically sound and user-oriented.

In this chapter, we present the main tools and technologies that supported the development of the project. PyCharm was selected as the primary Integrated Development Environment (IDE), offering robust features for productivity and debugging. Python, as the core programming language, provided an extensive ecosystem of libraries crucial for natural language processing, machine learning, and web development. Ollama enabled the integration of local large language models (LLMs), such as llama4, ensuring flexibility in analysis tasks while reducing reliance on cloud-based APIs. Finally, a custom summarization model, fine-tuned on BBC news data, was developed to generate fast, domain-specific summaries tailored to the needs of modern news readers.

Together, these tools formed the backbone of the project, transforming the abstract business requirements defined in Chapter 1 into a practical and scalable implementation.

4.2-PyCharm

PyCharm, developed by JetBrains, was chosen as the primary IDE for building News Advance. It provided a professional environment that streamlined the entire development workflow. Features such as intelligent code completion, real-time error detection, integrated version control, and built-in Django support significantly enhanced productivity.

The debugging tools in PyCharm proved invaluable during the integration of complex modules like the aggregation pipeline and the analysis system. Instead of relying solely on print statements, the

debugger allowed for step-by-step inspection of code execution, variable states, and error traces. This was particularly useful when troubleshooting Django views, API endpoints, and the AI model integration.

Moreover, PyCharm's project management features made it easy to maintain a modular architecture, keeping the news_aggregator, news_analysis, and accounts apps organized. Combined with virtual environment management and plugin support, PyCharm provided a stable and efficient foundation for both rapid prototyping and long-term maintainability.

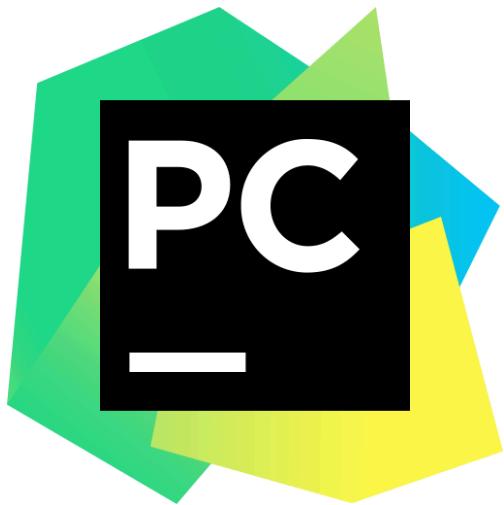


Figure 2: jetbrains pycharm logo

4.3-Python

Python served as the backbone of the *News Advance* project. Its readability, extensive libraries, and strong community support made it the ideal choice for combining web development with natural language processing. Django, the chosen web framework, leveraged Python's strengths to provide a secure and scalable foundation for the system.

Beyond Django, Python's rich ecosystem of libraries played a crucial role in the project:

- **Natural Language Processing:** NLTK (VADER), spaCy, and transformers were used for tasks ranging from sentiment detection to advanced summarization.

- **Web Scraping:** newspaper3k³² and BeautifulSoup4 allowed seamless aggregation of news articles from diverse sources.
- **Machine Learning:** PyTorch and Hugging Face Transformers powered the fine-tuned summarization model.
- **Utilities:** Python libraries such as Faker, dotenv, and Pillow simplified testing, configuration, and image processing.

The flexibility of Python ensured that *News Advance* could integrate traditional rule-based NLP with state-of-the-art transformer models, while also supporting robust database operations, APIs, and user interfaces.

4.4-Ollama

Ollama was integrated into the system to enable the use of local large language models (LLMs). Its primary advantage was offering AI-powered text analysis without relying on cloud services, thereby ensuring greater privacy, lower latency, and reduced dependency on external APIs.

Within *News Advance*, Ollama was used as a fallback mechanism and an enhancement layer in the analysis pipeline. Whenever the fine-tuned summarization model was unavailable, or when a more nuanced response was required, Ollama-powered models could step in to handle sentiment analysis, bias detection, and claim verification.

The modular configuration of Ollama allowed multiple models to be tested and integrated efficiently. Among these, *llama3* served as a general-purpose language model, while specialized models were applied for summarization and reasoning tasks.

³² newspaper3k: <https://newspaper.readthedocs.io/>



Figure 3: Ollama logo

4.4.1-Llama3

Llama3 was one of the primary models deployed through Ollama for *News Advance*. It was chosen for its balance between performance and efficiency on local hardware. The model supported key features such as political bias classification, sentiment interpretation, and logical fallacy detection.

One of the advantages of Llama3 was its ability to generate context-aware insights quickly, making it well-suited for interactive analysis in the user interface. Although not specifically fine-tuned for journalism, it provided robust general-purpose language understanding that complemented the more specialized summarization model.

4.4.2-Custom Summary Model Trained on BBC Data

To achieve fast and domain-relevant news summarization, a custom model was developed using the facebook/bart-base architecture. This model was fine-tuned on the BBC News Summary dataset, which contains thousands of professionally written news articles and summaries.

The fine-tuned model delivered concise, high-quality summaries tailored to the structure of news writing, ensuring users could grasp the key points of an article without reading the full text. This reduced information overload while maintaining factual accuracy and readability.

In production, the summarization pipeline prioritized this fine-tuned BART model. If unavailable, it fell back to Ollama-powered LLMs, which provided more flexible and creative

outputs at the cost of consistency. This dual-path strategy ensured both reliability and adaptability in news analysis.

4.5-Conclusion

The tools described in this chapter were more than just technical choices — they defined the identity and capabilities of News Advance. PyCharm provided the environment for efficient development and debugging, while Python offered a versatile programming foundation supported by a powerful ecosystem of libraries. Ollama extended the system's flexibility, enabling the integration of advanced LLMs such as llama3, while the custom summarization model ensured fast, domain-specific outputs tailored for journalism.

Together, these tools transformed the project's business requirements into a concrete implementation strategy. They enabled seamless interaction between web development, natural language processing, and artificial intelligence, creating a platform that is both innovative and practical.

This chapter highlighted the “with what” of the project’s development. The next chapter will shift the focus to the “how,” exploring the modeling and conceptual design that structured the system’s architecture.

Chapter 5: From Code to Experience — Implementation and Demonstration

5.1-Introduction

This chapter presents the implementation and demonstration of *News Advance*, moving from conceptual models to a fully functioning system. After establishing the business requirements, selecting tools, and modeling the architecture in previous chapters, the focus here shifts to the practical realization of those designs.

The implementation phase covers the platform and tools used, the Django MVT framework, backend customizations, and the APIs that enable communication across components. It also details the aggregation and analysis pipelines, including sentiment detection, political bias classification, logical fallacy identification, summarization, and fact-checking. Special emphasis is placed on the fallback strategies that ensure robustness even when AI models are unavailable.

Alongside backend processes, the chapter demonstrates the user-facing experience. Screenshots illustrate key interactions such as registration, login, profile management, and article exploration. User interface design emphasizes accessibility, preference persistence, and responsive layouts, while administrative functions showcase system oversight through curated sources, fallacy catalogs, and reliability scoring.

Finally, the development timeline, testing strategy, and deployment setup highlight the methodology and quality assurance measures taken to deliver a secure, maintainable, and production-ready platform. Together, these sections provide a comprehensive view of how *News Advance* was implemented and how it operates in practice.

5.2-Platform and Tools used

- Framework: Django 5.2 (MVT), Python 3.10+
- Databases: SQLite (dev), PostgreSQL (prod target)
- NLP/AI: NLTK (VADER), spaCy (en_core_web_sm), Transformers + Torch (BART summarization), Ollama (local LLMs)

- Scraping: newspaper3k, BeautifulSoup4
- UI: Bootstrap 5 static assets
- Dev: PyCharm/VS Code, python-dotenv

Configuration highlights:

```
OLLAMA_ENDPOINT = 'http://localhost:11434/api/generate'  
SUMMARIZATION_MODEL_DIR = BASE_DIR / 'news_analysis' / 'ml_models' /  
'summarization' / 'trained_model'  
SUMMARIZATION_BASE_MODEL = 'facebook/bart-base'  
USE_ML_SUMMARIZATION = True
```

Django community configuration:

1. Set up a virtual environment.
2. Install Django. (pip install Django)
3. Create a project (django-admin startproject project_name)
4. Create an app (python manage.py startapp app_name)

PostgreSQL configuration:

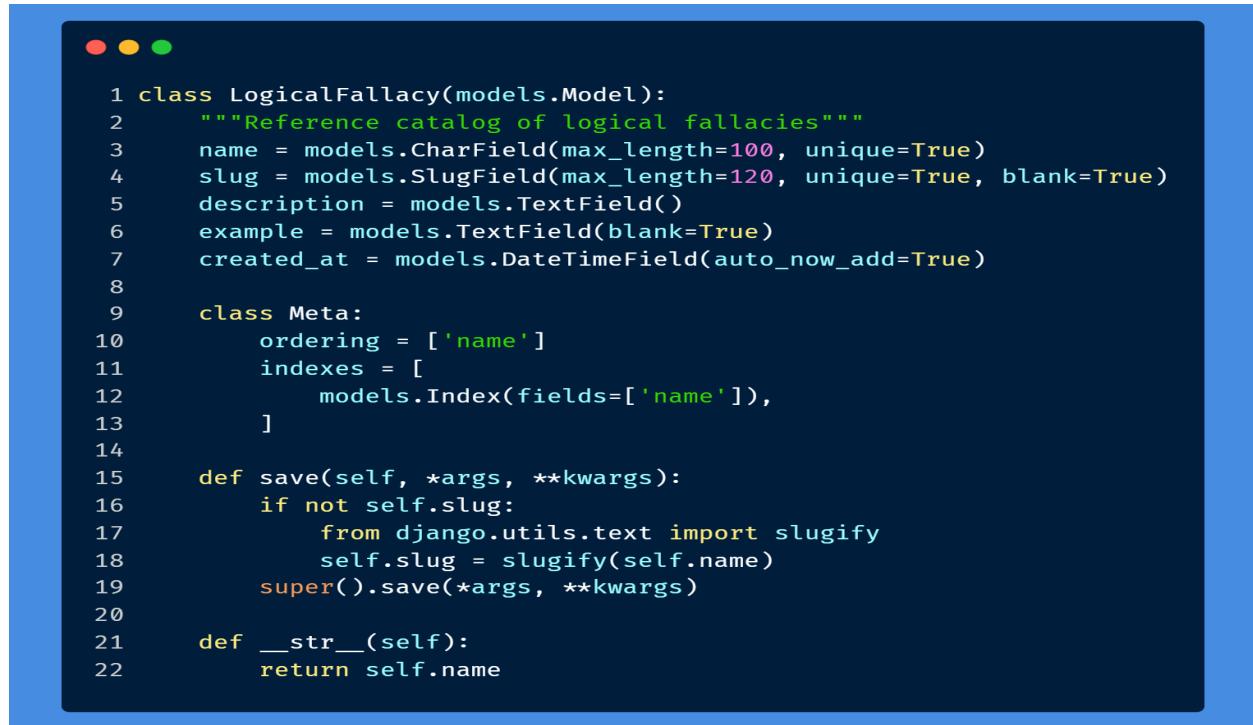
5. Download PgAdmin
6. Create Login/Group Role which is going to be the DB user and password
7. Create Database which is going to be the DB name

5.3-MVT: (Model-View-Template) in this project

Django follows the Model-View-Template (MVT) architectural pattern, which is similar to the Model-View-Controller (MVC) pattern.

In MVT, the Model represents the data and business logic of the application, the View handles user interaction and presentation logic, and the Template handles the visual representation of the data.

Model: The Model represents the database schema and the data stored in it. It defines the structure of the data and the relationships between different data entities. In Django, the Model is defined using a Python class that inherits from `django.db.models.Model` and defines the fields and their types.



```
 1 class LogicalFallacy(models.Model):
 2     """Reference catalog of logical fallacies"""
 3     name = models.CharField(max_length=100, unique=True)
 4     slug = models.SlugField(max_length=120, unique=True, blank=True)
 5     description = models.TextField()
 6     example = models.TextField(blank=True)
 7     created_at = models.DateTimeField(auto_now_add=True)
 8
 9     class Meta:
10         ordering = ['name']
11         indexes = [
12             models.Index(fields=['name']),
13         ]
14
15     def save(self, *args, **kwargs):
16         if not self.slug:
17             from django.utils.text import slugify
18             self.slug = slugify(self.name)
19         super().save(*args, **kwargs)
20
21     def __str__(self):
22         return self.name
```

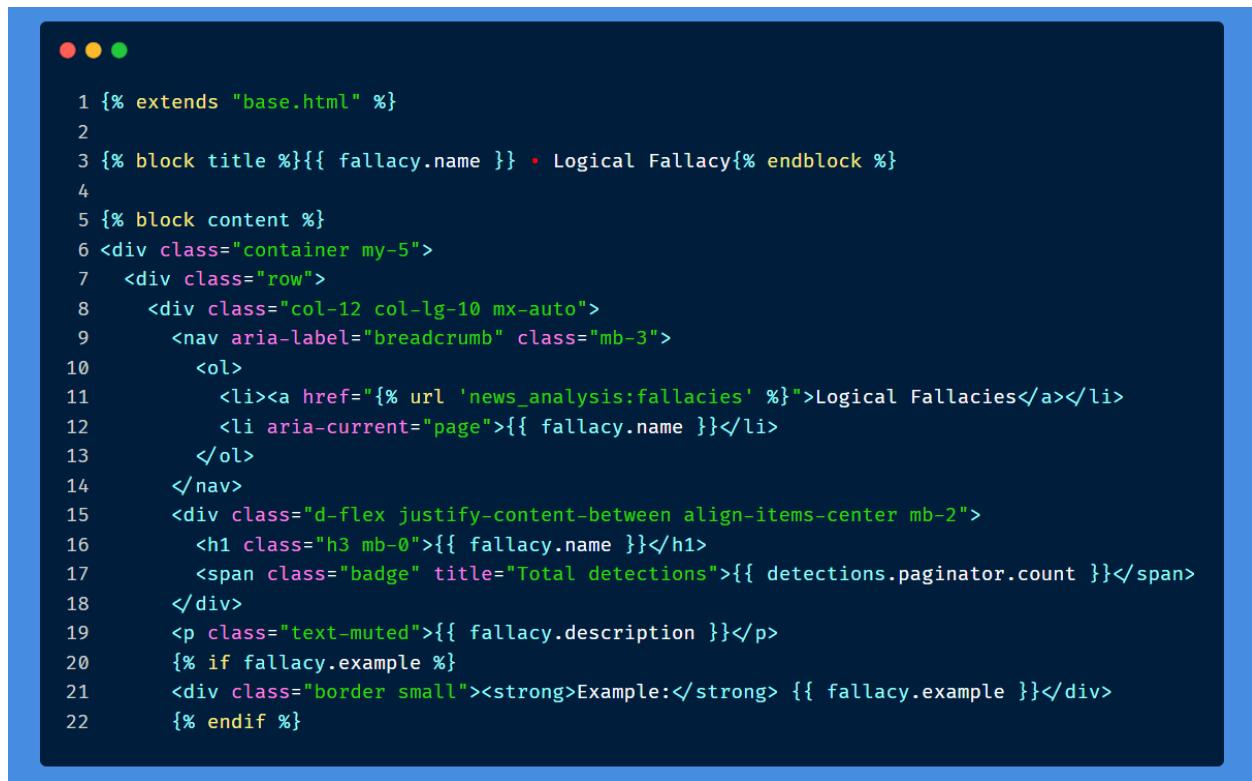
Figure 7: Model

View: The View is responsible for handling user requests, interacting with the Model to retrieve or manipulate data, and returning an HTTP response. In Django, the View is implemented as a Python function or class-based view that receives an HTTP request and returns an HTTP response.

```
● ● ●
1 def fallacy_detail(request, slug):
2     """Detail page for a single logical fallacy with related detections/articles."""
3     fallacy = get_object_or_404(LogicalFallacy, slug=slug)
4     detections_qs = (
5         LogicalFallacyDetection.objects
6             .filter(fallacy=fallacy)
7             .select_related('article', 'article__source')
8             .order_by('-detected_at')
9     )
10    paginator = Paginator(detections_qs, 20)
11    page_number = request.GET.get('page')
12    detections = paginator.get_page(page_number)
13
14    context = {
15        'fallacy': fallacy,
16        'detections': detections,
17    }
18    return render(request, 'news_analysis/fallacy_detail.html', context)
```

Figure 8: View

Template: The Template is responsible for rendering the data returned by the View into a visual format that can be presented to the user. In Django, the Template is implemented using HTML and other web development technologies, and it can include placeholders for dynamic data that is populated by the View.



A screenshot of a code editor window displaying a Django template file. The file contains HTML and Django-specific syntax like {% extends %}, {% block %}, and {% if %}. The code is organized into blocks for title and content, with navigation and descriptive text elements. The editor has a dark theme with syntax highlighting in green, blue, and yellow.

```
1 {% extends "base.html" %} 2 3 {% block title %}{{ fallacy.name }} • Logical Fallacy{% endblock %} 4 5 {% block content %} 6 <div class="container my-5"> 7   <div class="row"> 8     <div class="col-12 col-lg-10 mx-auto"> 9       <nav aria-label="breadcrumb" class="mb-3"> 10         <ol> 11           <li><a href="{% url 'news_analysis:fallacies' %}">Logical Fallacies</a></li> 12           <li aria-current="page">{{ fallacy.name }}</li> 13         </ol> 14       </nav> 15       <div class="d-flex justify-content-between align-items-center mb-2"> 16         <h1 class="h3 mb-0">{{ fallacy.name }}</h1> 17         <span class="badge" title="Total detections">{{ detections.paginator.count }}</span> 18       </div> 19       <p class="text-muted">{{ fallacy.description }}</p> 20       {% if fallacy.example %} 21         <div class="border small"><strong>Example:</strong> {{ fallacy.example }}</div> 22       {% endif %}
```

Figure 9: Template

The MVT pattern in Django³³ emphasizes the separation of concerns between the Model, View, and Template, making it easier to maintain and update the application code. It also provides a modular structure that allows developers to reuse code and extend the application functionality as needed.

5.4-Backend admin customization and data management

In Django, the main backend dashboard is usually referred to as the Django Admin site. It is a built-in feature of Django that provides a powerful interface for managing the application's data models and performing administrative tasks.

The Django Admin site allows authorized users to log in and access various functionalities such as adding, editing, and deleting data objects. It also provides features for managing user authentication, creating custom views, and generating reports.

The Django Admin site is automatically generated based on the application's defined data models, which means that developers do not need to write any additional code to create the dashboard. The interface provides a user-friendly and customizable way to manage the application's data and provides a quick way to access all the data models registered in the application.

The Django Admin site provides a high level of security, as it requires user authentication to access its functionalities. It also allows developers to control the access and permissions of different users to the dashboard and its functionalities.

Customizing the Backend Dashboard:

³³ Django Documentation: <https://docs.djangoproject.com/>

While the default backend dashboard provides a functional and straightforward interface for managing the database, it may not be suitable for all users or organizations. Therefore, the dashboard can be customized to better fit the needs of the users and the project.

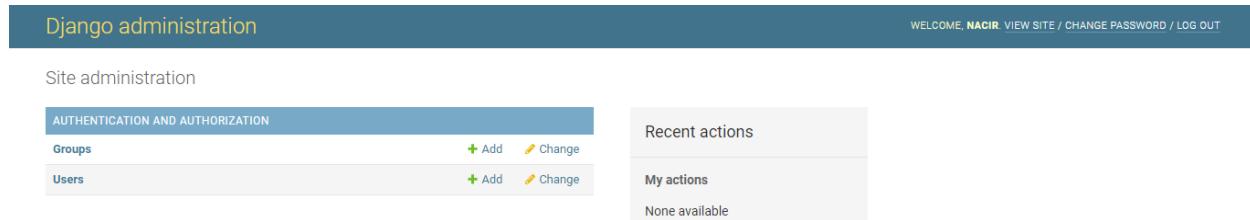


Figure 10: Uncustomized Main CMS Dashboard

In this example, we see the default backend dashboard with its standard look and feel. However, users can modify the dashboard's appearance, including changing the color scheme, typography, and layout, to make it more user-friendly and aligned with their brand identity.

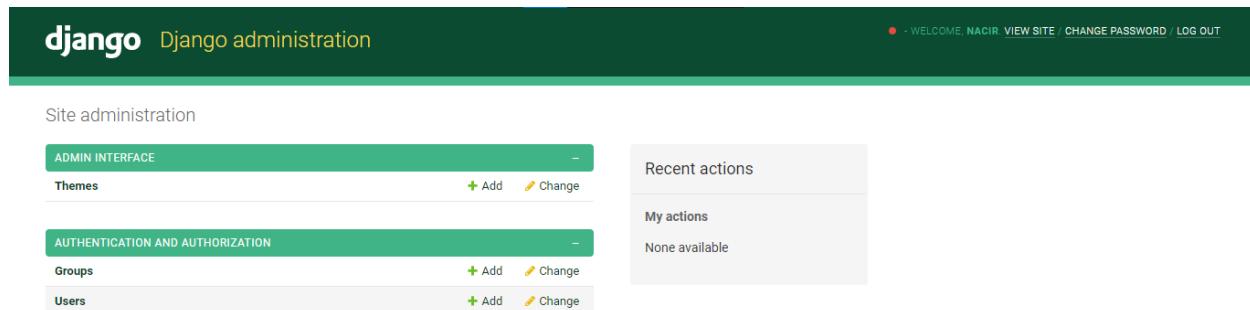


Figure 11: Customized CMS Dashboard

Here, we see an example of a customized dashboard with a distinct color scheme and layout, providing a unique and appealing look and feel.

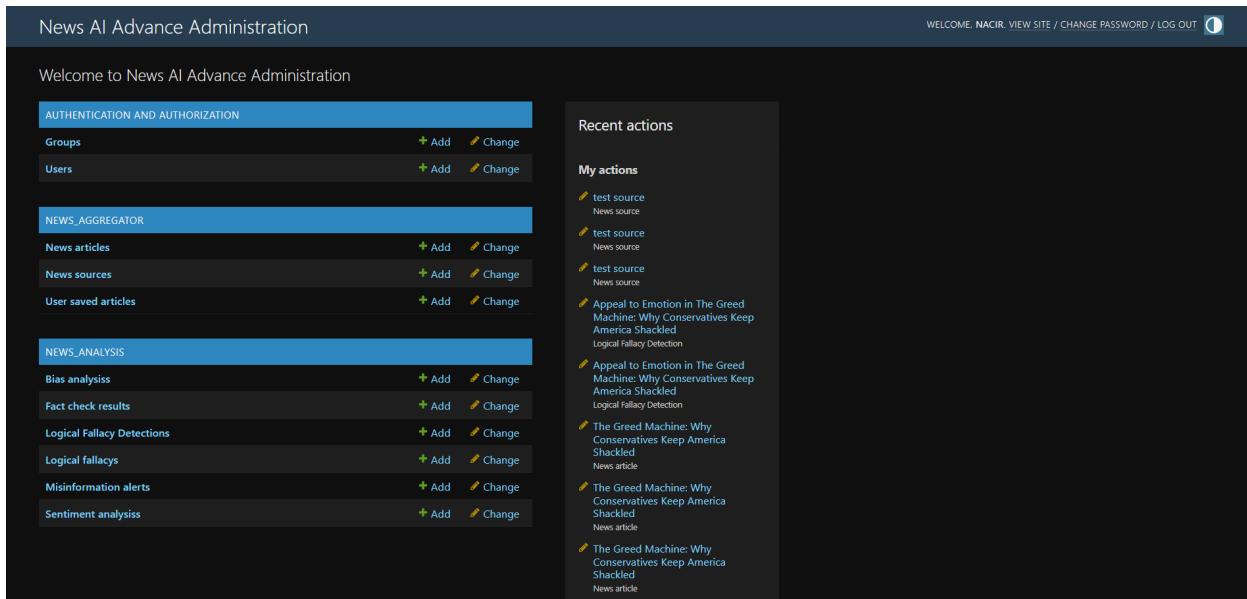


Figure 12: Main CMS Dashboard

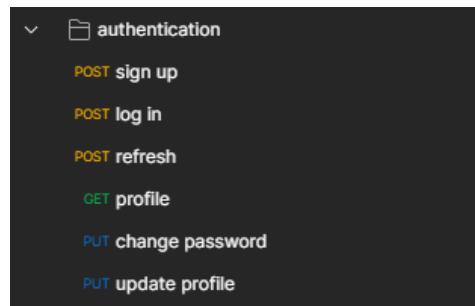
Another example of a customized dashboard, with a different color scheme and typography, making it more legible and easier to use.

Customizing the backend dashboard allows users to create a more personalized and functional experience when interacting with the system. However, it's important to ensure that the customizations do not compromise the system's security, functionality, or usability.

5.5-Application Programming Interface (APIs)

An API, or Application Programming Interface, is a set of rules and protocols that allow different software applications to communicate with each other. In the context of a Django project, an API is a way to expose your application's data and functionality to external clients, such as mobile apps or other web services.

APIs are typically built using a combination of web technologies, such as HTTP, XML, and JSON. They provide a standardized way for clients to request data and perform actions, using a set of well-defined endpoints and parameters.



In the case of your Django project, you might use APIs to allow external clients to perform tasks such as register new users, login, and get a profile, or get the processed articles and make a new ui for displaying them. By providing APIs, you can enable other developers to build

on top of your application, creating new integrations and value-added services.

Figure 13: APIs requests

A compact JSON endpoint exposes misinformation alerts for a given article:

```
path('api/articles/<int:article_id>/misinformation-alerts/',
      article_misinformation_alerts, name='article_misinformation_alerts')
```

Example response:

The screenshot shows a REST API testing interface. The URL is set to `{base_url}/analysis/api/articles/26/misinformation-alerts/`. The method is set to `GET`. The response status is `200 OK` with a response time of `9 ms` and a size of `411 B`. The response body is displayed as JSON:

```
1  {
2   "article_id": 26,
3   "alerts": [
4     {
5       "id": 5,
6       "title": "miss info 1",
7       "severity": "high",
8       "detected_at": "2025-05-01T14:02:07.217064+00:00"
9     }
10   ]
11 }
```

Figure 14: Misinformation Alerts API

The server validates the request and retrieves the appropriate information from the database before returning the response to the user. The specific fields and validation requirements for GET requests may depend on the business requirements and data model of the application.

While there may be many other APIs available in the system, that are used internally for better user experience like saving article api consumed by the AJAX calls in some pages. However, depending on the business requirements and use cases of the system, additional APIs may be necessary to provide more advanced functionality or integration with other systems.

5.6-Aggregation pipeline and management commands

- fetch_news: scrape sources with newspaper3k, normalize content, store NewsArticle and link to NewsSource.
- analyze_articles: run bias, sentiment, summarization (ML with Ollama fallback), key insights, logical fallacy detection, and fact-checking.
- generate_test_data: seed sources/articles for development.
- recalculate_reliability: recompute NewsSource.reliability_score from recent fact checks, fallacy frequency, and bias consistency.
- send_misinformation_alerts: compile and optionally notify admins/users of trending alerts.

5.7-Analysis pipeline overview

- Input: NewsArticle content
- Preprocess: clean HTML, normalize whitespace, tokenize
- Sentiment: VADER baseline; optional Ollama-enhanced sentiment with explanation
- Bias: LLM classification with score and confidence
- Summarization: BART model or Ollama fallback
- Insights: bullet-point extraction
- Logical fallacies: rule/LLM-assisted detection persisted per article
- Fact-checking: claim extraction + verification + rating/confidence

Claim extraction scoring (excerpt):

```
def score_sentence(s: str) -> float:  
    if len(s) < 40 or len(s) > 300: return -1  
    score = 0.0  
    if re.search(r"[0-9]|%|\$\$", s): score += 1.5  
    if "''" in s or "''' in s: score += 0.4
```

5.8-Fact-checking and logical fallacies

- FactCheckResult stores claim text, rating (true/mostly_true/.../unverified), confidence, explanation, and sources.

News AI Advance

Change fact check result

Fact Check for claim: By their logic, taxing Jeff Bezos one more cent me...

Article:	The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies			
Claim:	By their logic, taxing Jeff Bezos one more cent means your local coffee shop will be bulldozed by Big Government tanks.			
Rating:	Mostly True			
Explanation:	The claim that taxing Jeff Bezos one more cent would lead to the bulldozing of local businesses is an exaggeration and not supported by credible sources.			
Sources:	<pre>{"type": "article", "url": "https://www.theatlantic.com/ideas/archive/2020/09/tax-billionaires-fairness/609455/", "title": "Why Tax Billionaires?"}, {"type": "research", "url": "https://www.economicpolicyinstitute.org/publications/taxing-the-rich-reduces-income-inequality/", "title": "Taxing the Rich Reduces Income Inequality"}</pre>			
Confidence:	0.8			
Last verified:	Date: 2025-09-13	Today	Time: 10:20:57	Now
Note: You are 3 hours ahead of server time.				

Figure 15: Fact check edit

- LogicalFallacy catalog is maintained in admin; detections are stored per article with optional evidence excerpt and character spans.
- Admins can review detections and fact checks, re-run verification, and attach alerts for trending misinformation.

Select Logical Fallacy Detection to change

Select Logical Fallacy Detection to change				
<input type="text"/> Search				
Action:	-----	Go	0 of 5 selected	
<input type="checkbox"/> ARTICLE	FALLACY	CONFIDENCE	DETECTED AT	
<input type="checkbox"/> The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies	Straw Man	0.8	Sept. 13, 2025, 10:18 a.m.	
<input type="checkbox"/> The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies	Slippery Slope	0.9	Sept. 13, 2025, 10:18 a.m.	
<input type="checkbox"/> The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies	False Dichotomy	0.7	Sept. 13, 2025, 10:18 a.m.	
<input type="checkbox"/> The Greed Machine: Why Conservatives Keep America Shackled	Straw Man	0.7	Sept. 11, 2025, 3:05 p.m.	
<input type="checkbox"/> The Greed Machine: Why Conservatives Keep America Shackled	False Dichotomy	0.9	Sept. 11, 2025, 3:05 p.m.	

Figure 16: Logical fallacies detections for articles

Change Logical Fallacy Detection

Slippery Slope in The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies

HISTORY

Article: The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies

Fallacy: Slippery Slope

Confidence: 0.9

Evidence excerpt: If we dare raise taxes on billionaires, businesses will collapse and jobs will vanish overnight.

Start char: 514

End char: 610

SAVE **Save and add another** **Save and continue editing** **Delete**

The screenshot shows a dark-themed user interface for managing logical fallacy detections. At the top, it says 'Change Logical Fallacy Detection' and displays the title 'Slippery Slope in The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies'. Below this is a green 'HISTORY' button. The main area contains fields for 'Article' (set to 'The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies'), 'Fallacy' (set to 'Slippery Slope'), and 'Confidence' (set to '0.9'). A text box labeled 'Evidence excerpt' contains the sentence 'If we dare raise taxes on billionaires, businesses will collapse and jobs will vanish overnight.' Below the excerpt are 'Start char' (514) and 'End char' (610) fields. At the bottom, there are four buttons: 'SAVE' (blue), 'Save and add another' (light blue), 'Save and continue editing' (light blue), and 'Delete' (red).

Figure 17: Logical fallacy detection edit

5.9-Summarization and fallback strategy

- Default: use fine-tuned BART from SUMMARIZATION_MODEL_DIR
- Fallback: if model missing or errors, send prompt to Ollama (at OLLAMA_ENDPOINT)
- Summaries are cached on the article record and refreshed if content changes

Change news article

The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies

Title:	The Billionaire's Fairy Tale: How "Job Creators"
Source:	test source <input type="button" value="▼"/>   
Url:	Currently: http://test.com/test2 Change: <input type="text" value="http://test.com/test2"/>
Author:	Jhon Author
Published date:	Date: <input type="text" value="2025-09-13"/> Today  Time: <input type="text" value="10:14:56"/> Now 
<p>Note: You are 3 hours ahead of server time.</p> <hr/> <p>Content:</p> <p>It's astonishing that in 2025, we're still swallowing the same tired story: billionaires are "job creators." Let's be clear—no, they are not. Workers create value. The people stocking shelves, delivering packages, and keeping hospitals afloat are the real lifeblood of the economy. Yet somehow, society has been tricked into worshiping the ultra-rich as if their yachts and tax havens magically lift everyone else.</p> <p>But let's not kid ourselves—this isn't just ignorance, it's manipulation. Conservatives insist that if we dare raise taxes on billionaires, businesses will collapse and jobs will vanish overnight. That's a classic slippery slope fallacy. By their logic, taxing Jeff Bezos one more cent means your local coffee shop will be bulldozed by Big Government tanks. Absurd.</p>	
Summary:	The next time someone says we need to protect "job creators," remember—it's nothing but a fairy tale for fools.

Figure 18: Article content and summary

5.10-UI/UX and user preferences

- Register page

The screenshot shows the 'Create an Account' registration form. At the top, there's a blue header bar with the text '+ Create an Account'. Below it is a dark blue input field labeled 'Username'. Underneath is another dark blue input field labeled 'Password', with a note below stating: 'Your password must contain at least 8 characters and can't be entirely numeric.' Below the password field is a third dark blue input field labeled 'Confirm Password'. At the bottom of the form is a blue button with the text '+ Sign Up'. At the very bottom of the page, there's a dark footer bar with the text 'Already have an account? [Login here](#)'.

Figure 19: Sign up page

- Login page

The screenshot shows the 'Login' page. At the top, there's a blue header bar with the text 'Login'. Below it is a dark blue input field labeled 'Username'. Underneath is another dark blue input field labeled 'Password'. Below these fields is a blue button with the text 'Login'. At the bottom of the page, there are two links: 'Don't have an account? [Sign up here](#)' and 'Forgot your password? [Reset it here](#)'.

Figure 20: Login page

- Article page sections: AI summary, key insights, sentiment, bias, fallacies, fact checks, alerts

The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies

test source | Test Author | September 13, 2025 | Save | Original Source

AI Analysis | What do these mean?

Political Bias i
Left Leaning Neutral Right Leaning

Assessment: Left

Emotional Tone i
Negative Neutral Positive

Overall sentiment: Mostly Negative i

Logical Fallacies i Learn about fallacies

- Straw Man** Details View Confidence: 0.80
"defenders of extreme wealth like to paint progressives as "radical socialists"..."
- False Dichotomy** Details View Confidence: 0.70
"conservatives who parrot these billionaire talking points are either hopelessly naive or willfully cruel."
- Slippery Slope** Details View Confidence: 0.90
"If we dare raise taxes on billionaires, businesses will collapse and jobs will vanish overnight."

Fact Checks i

Headline claim: The Billionaire's Fairy Tale: How "Job Creators" Keep...
True It's astonishing that in 2025, we're still swallowing the same tired ...
Mostly False Claim: It's astonishing that in 2025, we're still swallowing the same tired story: billionaires are "job creators."
Explanation: The claim that 'billionaires are'

Figure 21: Article AI analysis section

The screenshot shows the main article page. At the top, there's a navigation bar with 'News Advance' and links for 'Home', 'Latest News', 'Analysis Tools', and 'About'. A user profile 'nassi' is at the top right. Below the navigation is a large image of a coastal landscape. To the left of the image is a red-bordered box containing the 'AI-Generated Summary' and 'Key Insights' sections.

AI-Generated Summary:

The next time someone says we need to protect "job creators," remember—it's nothing but a fairy tale for fools.

Key Insights:

- Billionaires do not create jobs; workers create value."
- Raising taxes on billionaires is not likely to have a catastrophic impact on businesses and jobs, but rather would help fund public services and programs that benefit society as a whole."
- Conservative defenders of extreme wealth often use misleading rhetoric and strawman arguments to discredit progressive policies aimed at reducing income inequality."

Below these sections is a block of text:

It's astonishing that in 2025, we're still swallowing the same tired story: billionaires are "job creators." Let's be clear—no, they are not. Workers create value. The people stocking shelves, delivering packages, and keeping hospitals afloat are the real lifeblood of the economy. Yet somehow, society has been tricked into worshiping the ultra-rich as if their yachts and tax havens magically lift everyone else.

But let's not kid ourselves—this isn't just ignorance, it's manipulation. Conservatives insist that [if we dare raise taxes on billionaires, businesses will collapse and jobs will vanish overnight](#). That's a classic slippery slope fallacy. By their logic, taxing Jeff Bezos one more cent means your local coffee shop will be bulldozed by Big Government tanks. Absurd.

Even worse, [defenders of extreme wealth like to paint progressives as "radical socialists"](#) who want to

To the right of the main content area is a sidebar titled 'Fact Checks' with several items, and a 'Related Articles' section with a single item. At the bottom right is a 'About the Source' button.

Figure 22: Article AI Summary and key insights section

- Edit Preferences - User toggles: show/hide summary, insights, and Analysis; persist preference

The screenshot shows the 'News Preferences' page. At the top, there's a navigation bar with 'News Advance' and links for 'Home', 'Latest News', 'Analysis Tools', and 'About'. A user profile 'nassi' is at the top right. The main content area is titled 'News Preferences' and contains several sections with toggle switches:

Content Filters

Political Balance: Balanced Mix
Choose how you want to balance news across the political spectrum in your feed

Analysis Features

- Enable Fact-Checking
Show fact-check results for claims in articles
- Enable Bias Analysis
This will remove the related section from the article view
- Enable Sentiment Analysis
This will remove the related section from the article view
- Enable Logical Fallacy Detection
Show detected logical fallacies and examples in the AI Analysis panel

Display Features

- Show AI-Generated Summary
Toggle visibility of the AI summary on article pages
- Show Key Insights
Toggle visibility of the Key Insights panel on article pages

Figure 23: User preferences page

- Accessibility: semantic headings, keyboard navigation, and minimal color dependency

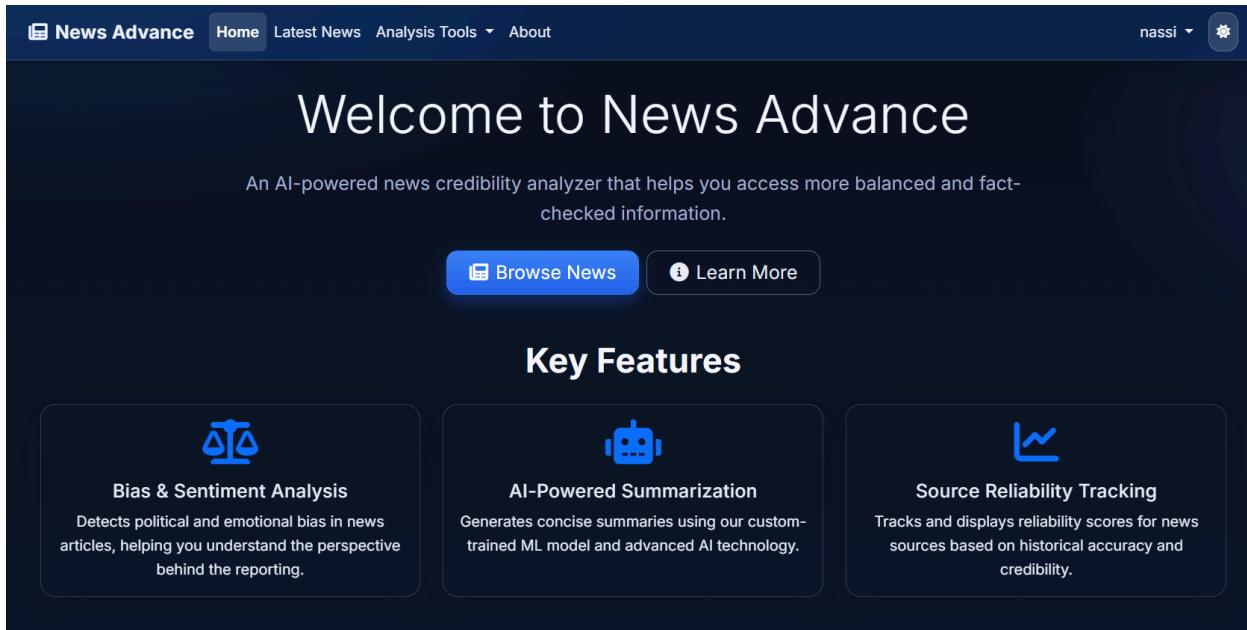


Figure 24: Home page

5.11-Source reliability scoring

- Inputs: recent FactCheckResult ratings distribution, LogicalFallacyDetection frequency, BiasAnalysis variance vs. claimed leaning
- Weights: fact checks (~60%), bias consistency (~20%), fallacy frequency (~20%)
- Output: 0–100 reliability_score on NewsSource; recalculated via management command

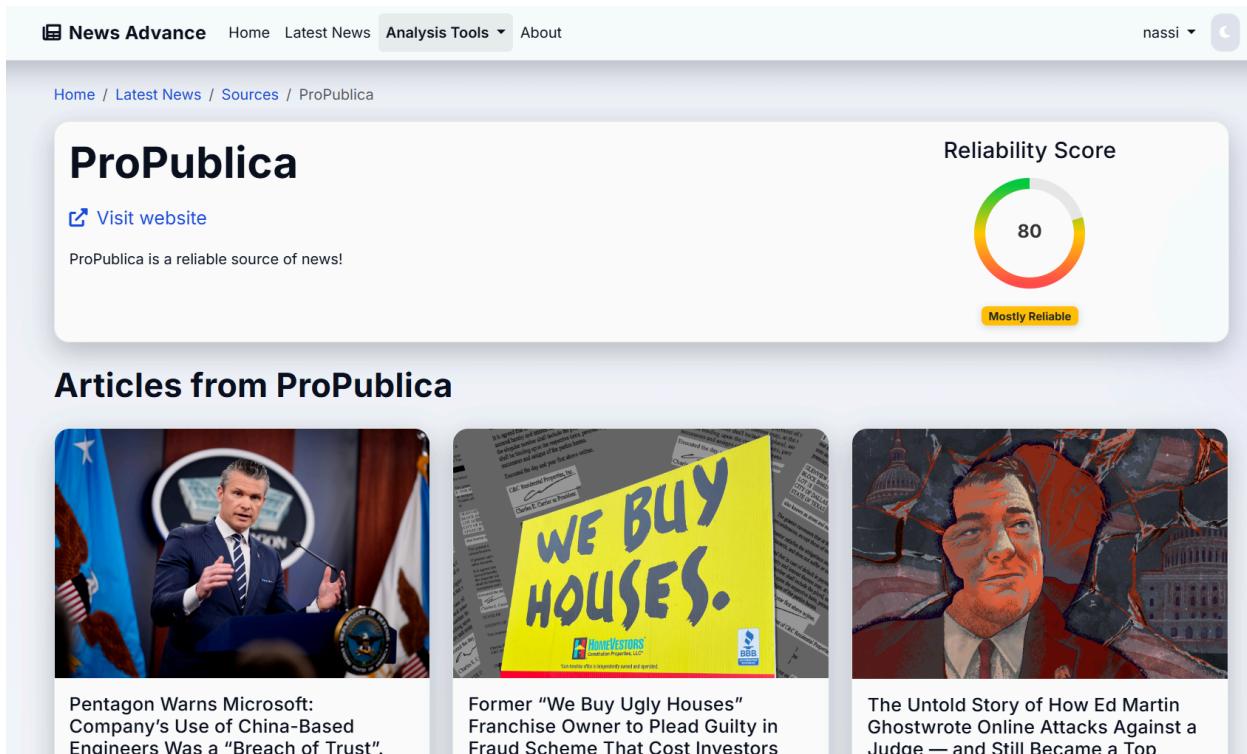


Figure 25: Source articles and reliability score

5.12-UI/UX: screens, interactions, and error handling

- Navigation and information architecture:
 - Landing → Sources → Article list → Article details with analysis panels (Summary, Insights, Sentiment, Bias, Fallacies, Fact checks, Alerts)
 - Admin routes for curation (sources, articles, fallacy catalog, fact checks)
- Article details layout:
 - Left: original article metadata (source, date, URL) and raw content toggle
 - Right: stacked analysis cards with clear status, timestamps, and confidences
- User preferences and state:
 - Per-user toggles to show/hide summary, insights, and fact checks; persisted and respected server-side
 - Sticky last-viewed section; deep links to specific panels via anchors
- Feedback and failure UX:
 - Graceful fallbacks: ML summary → Ollama; missing fact checks render a clear CTA to re-run
 - Error banners carry correlation IDs (server logs) for quick triage

- Accessibility and responsiveness:
 - Bootstrap 5 semantics, keyboard navigation, contrast-safe palette, mobile-first cards

News Advance

Search Articles

Search by title or content

Filter by Source

All Sources

The Billionaire's Fairy Tale: How "Job Creators" Keep Selling Us Lies

test source • Sep 13, 2025

Left Negative

The next time someone says we need to protect "job creators," remember—it's nothing but a fairy tale for fools.

News Advance

At the end of the day, this isn't about ideology—it's about morality. You can't be pro-conservative and claim to care about human beings. Period.

AI Analysis What do these mean?

Political Bias

Left Leaning Neutral Right Leaning

Assessment: Left

Emotional Tone

Negative Neutral Positive

Overall sentiment: Mostly Positive

Logical Fallacies Learn about fallacies

Straw Man Details View Confidence: 0.70
They'll fight tooth and nail against universal healthcare, climate action, and workers' rights—not because these things don't work, but because they do work, and that threatens the power of the elite."

False Dichotomy Details View Confidence: 0.90
You can't be pro-conservative and claim to care about human beings."

Figure 26: Responsive on mobile view

5.13-Development timeline and methodology

- Week 1–2: Project scaffolding (Django apps), settings, admin baselines, initial models
- Week 3–4: Aggregation pipeline with newspaper3k, normalization, and admin curation flows
- Week 5–6: Analysis pipeline v1 (sentiment, bias, summarization with ML+fallback); insights extraction
- Week 7–8: Logical fallacy detection, fact-checking (claims+verification), reliability scoring
- Week 9: API endpoint for misinformation alerts; UI/UX polish and preferences
- Week 10: Stabilization, documentation, and memoir alignment with the codebase
- Practice: small iterative commits, feature flags for ML usage, and reproducible seeds for demos

5.14-Testing strategy and QA

- Unit tests (targeted):
 - Claim extraction scoring, bias/sentiment classification wrappers, summarization fallback selection
 - Reliability score computation determinism on fixed inputs
- Integration checks (scripted):
 - End-to-end analyze_articles on a small fixture set; verify all analysis objects created once
 - API: GET /api/articles/misinformation-alerts returns structured JSON and correct severities
- Mocking and isolation:
 - Ollama calls stubbed in tests; deterministic fixtures for NLP models where practical
- Manual QA:
 - Admin smoke tests (create/edit/delete sources, articles, fallacy catalog)
 - UI spot checks for empty/loading/error panels and user preference persistence

5.15-Deployment and production setup

- Environment and configuration:
 - DEBUG=False, allowed hosts, secure cookies, CSRF/HTTPS, SECRET_KEY via env
 - Database: SQLite for dev; PostgreSQL for production
 - Static files: collectstatic with hashed filenames; whitenoise or CDN as needed
- Services:
 - Optional: Ollama service deployed on same host or isolated node; health checks before analysis

- Planned: Celery + Redis for scheduled ingestion and async pipeline runs
- Observability and reliability:
 - Structured logs for each pipeline stage with article/source IDs
 - Correlation IDs surfaced in UI error banners for quick log lookup
- Data hygiene:
 - Guardrails for duplicate URLs, charset/encoding normalization, and content length limits

5.16-Conclusion

The implementation and demonstration phase translated the project's theoretical framework into a concrete and interactive platform. By leveraging Django's MVT architecture, robust NLP and AI libraries, and a modular analysis pipeline, *News Advance* successfully delivers its core functionalities: news aggregation, summarization, sentiment and bias analysis, logical fallacy detection, fact-checking, and source reliability scoring.

On the user side, the platform offers a transparent and customizable reading experience. Preferences such as showing or hiding summaries and fact-checks are respected, while the article interface integrates AI-generated insights in an accessible and responsive design. Administrative workflows complement this by ensuring data quality, managing sources, and overseeing detection outputs.

Beyond technical implementation, the project followed a disciplined timeline, incorporating testing strategies, fallback mechanisms, and production-ready configurations. These ensured not only functional correctness but also resilience, scalability, and usability.

In summary, this chapter demonstrated how *News Advance* evolved from an abstract concept into a tangible platform, validating the design choices made in earlier chapters. With the system fully implemented and tested, the foundation is now set for critical evaluation and reflection in the concluding stages of this memoir.

Conclusion

The development of News Advance has been a journey from abstract ideas about media transparency to a tangible platform that empowers users to engage with news more critically. Beginning with the identification of business requirements, the project outlined key functions such as aggregation, summarization, bias and sentiment analysis, fact-checking, logical fallacy detection, and reliability scoring. Each of these features was carefully mapped to technical requirements, supported by the deliberate selection of tools, frameworks, and natural language processing models.

Through modeling and conception, the system's complexity was distilled into a coherent architecture, ensuring modularity, extensibility, and maintainability. The subsequent implementation phase validated these designs, producing a working platform that not only integrates advanced AI techniques but also provides a clear and intuitive user experience. Screenshots of registration, article exploration, and analysis panels demonstrated how the platform balances functionality with accessibility, while administrative features ensured oversight and data integrity.

The project achieved its central goal: creating a system that enhances the reliability, clarity, and transparency of news consumption in an age of overwhelming information and misinformation. While technical challenges were encountered—such as ensuring fallback mechanisms, training summarization models, and integrating diverse NLP pipelines—they were addressed through disciplined development, modular design, and iterative testing.

Looking ahead, News Advance can be extended with multilingual support, real-time misinformation alerts, deeper integration of fact-checking databases, and broader scalability through distributed pipelines. These enhancements would further strengthen its role as a practical tool for critical media literacy and trustworthy journalism.

In conclusion, this project demonstrates that the combination of natural language processing, large language models, and thoughtful system design can provide meaningful solutions to one of today's most pressing challenges: enabling individuals to navigate the news landscape with confidence, clarity, and critical awareness.

Perspective

Immediate enhancements:

- Celery/Redis: async ingestion and background analysis; scheduled re-analysis windows
- REST API: expand endpoints for sources, articles, fallacies, fact checks, and reliability scores
- Fact-checking with RAG: retrieval over reputable sources to ground LLM verification
- UI/UX: per-panel refresh; cross-source comparison; richer filtering and search
- Model upgrades: multilingual pipelines, domain-tuned summarizers, larger local LLMs (resource-aware)

References

News Aggregation

- ② MediaCloud – Harvard-Berkman Klein Center: <https://mediacloud.org/>
Open-source platform for analyzing media ecosystems; provides tools for tracking news sources and narratives.
- ② [NewsAPI.org](https://newsapi.org/): <https://newsapi.org/>
JSON API for searching worldwide news articles from diverse sources; used for aggregation pipeline research.

Credibility Scoring and Bias Detection

- ② Media Bias/Fact Check (MBFC): <https://mediabiasfactcheck.com/>
Manually curated database of news source bias ratings and factual reporting scores.
- ② AllSides: <https://www.allsides.com/unbiased-balanced-news/media-bias/media-bias-ratings/>
Crowd-sourced and expert-reviewed media bias ratings; provides ground-truth data for political leaning.
- ② Baly et al. (2020) – "What Was Written vs. Who Read It: News Media Profiling Using Text and Networks": <https://arxiv.org/abs/2005.07171>
MIT research on transformer-based political ideology prediction from article text; achieves 80% accuracy on five-point bias scale.
- ② ClaimBuster – Hassan et al. (2017) – "ClaimBuster: The First-Ever End-to-End Fact-Checking System": <https://arxiv.org/abs/1705.07743>
University of Texas at Arlington paper describing check-worthiness scoring for factual claims in political discourse.
- ② Sap et al. (2020) – "Fairness is Not Static: Deeper Understanding of Long-Term Fairness via Simulation Studies": <https://arxiv.org/abs/1911.03814>
Research demonstrating degradation of bias detection models over time; advocates for continuous retraining and re-analysis.

Summarization

- BBC News Summary Dataset: <https://huggingface.co/datasets/bbc-news-summarization>

Hugging Face dataset page; contains professionally written news abstracts ideal for domain-specific fine-tuning.
- facebook/bart-base Model: <https://huggingface.co/facebook/bart-base>

Hugging Face model card for BART architecture; base for custom fine-tuned summarization.
- google/pegasus Model: https://huggingface.co/google/pegasus-cnn_dailymail

Alternative abstractive summarization model; used for comparative analysis during development.
- Zhang et al. (2023) – "Cascading Models for Efficient and Robust Text Generation": <https://arxiv.org/abs/2305.11746>

Research on cascading small/fast models with larger fallback models; directly inspired dual-path summarization strategy.
- SummarizeBot (Commercial API): <https://www.summarizebot.com/>

Cloud-based summarization service; cited as example of API-dependent approach avoided in this project.
- Aylien News API: <https://aylien.com/>

Enterprise news analysis platform; provides sentiment and summarization but requires cloud data transfer.

Fact-Checking

- Full Fact (UK): <https://fullfact.org/>

Independent fact-checking organization; methodology for claim verification and rating systems.
- PolitiFact (U.S.): <https://www.politifact.com/>

Pulitzer Prize-winning fact-checker; uses "Truth-O-Meter" rating scale that influenced News Advance's design.

- FEVER Dataset: <https://fever.ai/>
Official website for Fact Extraction and VERification dataset; provides structured claim-evidence pairs for machine learning.
- Google Fact Check Tools API: <https://factchecktools.google.com/>
Aggregates fact-checks from multiple organizations; inspired claim-extraction and triage approach.

Logical Fallacy Detection

- Argument Annotated Essays Corpus – Stab & Gurevych (2017):
https://www.informatik.tu-darmstadt.de/ukp/research_6/data/argumentAnnotatedEssays_en/
UKP Lab dataset for argument mining; required domain adaptation for news rhetoric.
- IBM Project Debater – Goffredo et al. (2022):
<https://research.ibm.com/publications/project-debater-sleek-fast-and-persuasive>
IBM Research paper on structured argumentation analysis; highlights challenges of unstructured news text.
- [YourLogicalFallacyIs.com](https://yourlogicalfallacyis.com): <https://yourlogicalfallacyis.com/>
Educational resource for fallacy definitions; used for building curated catalog in admin interface.
- Purdue OWL – Logical Fallacies:
https://owl.purdue.edu/owl/general_writing/academic_writing/establishing_arguments/fallacies.html
Academic writing resource; provided structured definitions for fallacy taxonomy.
- ArgumenText (Commercial API): <https://www.argumentsearch.com/en/>
Argument mining API; cited as closed-source alternative to local detection approach.

Source Reliability

- The Trust Project – Santa Clara University: <https://thetrustproject.org/>
Eight-indicator framework for news trustworthiness; inspired weighted reliability scoring methodology.

- Reporters Without Borders (RSF) – Press Freedom Index: <https://rsf.org/en/ranking>

Global press freedom rankings; provides macro-level source context but not outlet-specific accuracy.
- NewsGuard: <https://www.newsguardtech.com/>

Commercial source rating service; proprietary methodology used as contrast for transparent scoring approach.
- Potthast et al. (2018) – "Stylometric Source Veracity Assessment":
<https://pan.webis.de/clef18/pan18-web/author-verification.html>

PAN Workshop paper on using writing style consistency to detect unreliable sources; influenced bias consistency weighting.

Local-First AI & Privacy

- Ollama: <https://ollama.com/>

Official website for local LLM inference; enables privacy-preserving AI without cloud dependency.
- Llama.cpp: <https://github.com/ggerganov/llama.cpp>

GitHub repository for efficient LLM inference on CPU; alternative runtime considered during development.
- GPT4All – Nomic AI: <https://gpt4all.io/>

Open-source chatbot ecosystem; part of local-first AI movement that informed architectural decisions.
- Andreessen Horowitz – "The Private AI Manifesto" (2023):
<https://a16z.com/2023/03/24/the-private-ai-manifesto/>

Venture capital firm's thesis on personal, portable AI; validated local inference approach.
- Microsoft Defending Democracy Program:
<https://www.microsoft.com/en-us/defendingdemocracy>

Cloud-based election integrity tools; cited as contrast to local deployment model.

- [AdVerif.ai](https://www.adverif.ai/): <https://www.adverif.ai/>

Commercial misinformation detection API; requires data transfer to external servers.

General Web Framework & NLP Libraries

- Django: <https://www.djangoproject.com/>

Django Documentation: <https://docs.djangoproject.com/>

Official documentation for MTV architecture and ORM modeling.

- Bootstrap: <https://getbootstrap.com/>

- Hugging Face Transformers: <https://huggingface.co/transformers/>

Official documentation for BART fine-tuning and model deployment.

- PyTorch: <https://pytorch.org/>

- NLTK (VADER Sentiment Analyzer): <https://www.nltk.org/>

Documentation for baseline sentiment scoring implementation:

https://www.nltk.org/_modules/nltk/sentiment/vader.html

- spaCy – en_core_web_sm: <https://spacy.io/>

- newspaper3k: <https://newspaper.readthedocs.io/>, [https://github.com/codelucas/newspaper/](https://github.com/codelucas/newspaper)