Final Year Project Proposal

TU856/4

NewsScope: A Global, Bias-Aware News Aggregator & Analysis Application

Christopher Noblett
C22454222

School of Computer Science
TU Dublin – City Campus

28/10/2025

Table of Contents

Contents

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Christopher Noblett

Christopher Noblett

28/10/2025

Summary

The proposed project, *NewsScope*, will be a cross-platform news application designed to collect and analyse news stories from English-language sources around the world. The core idea is to use artificial intelligence to examine how language is used in reporting, identifying tone, political bias, and framing. The system will also detect and categorise different forms of false or misleading information, including misinformation, disinformation, and malinformation.

NewsScope will operate through a **multi-layer architecture**: a Flutter-based Android frontend with Firebase Authentication, a FastAPI backend hosted on Render, an NLP engine powered by Hugging Face and TruSt, and a Supabase Postgres database with Buckets for archiving. News aggregation will draw from RSS feeds, NewsAPI, and GDELT. Planned features include personalised bias profiles, integrated fact-checking via PolitiFact, and spectrum-based visualisations of media outlets and articles.

The final deliverables will include a fully functional Android application, with a companion website as a possible extension if time allows.

Background (and References)

People today have more access to information than ever before, but this also brings greater exposure to biased and misleading content. With the rise of digital and social media, information disorder—including misinformation, disinformation, and malinformation—has become a major issue (Wardle and Derakhshan, 2017). Many audiences struggle to recognise when reporting is factual and impartial versus when it reflects political, corporate, or regional agendas. For instance, CNN and FOX News are widely seen as representing opposing ideological perspectives within the United States—CNN leaning centre-left and FOX right-leaning—while the BBC is more often regarded as maintaining a centrist and impartial approach to reporting (AllSides, 2025; BBC, 2021; Pew Research Center, 2020). This growing segregation of audiences highlights the need for systems that help readers critically compare how global events are reported across different outlets and viewpoints (Newman et al., 2024).

Existing solutions have tried to address these challenges in different ways. Ground News offers bias comparisons and blind-spot analysis using third-party bias ratings, though its Western focus and subscription model limit accessibility (Ground News, 2025). Research prototypes such as Tanbih (Tanbih Project, no date) and PureFeed (AlKhamissi et al., 2025) demonstrate machine learning approaches to event clustering and sensationalism filtering, but they are not designed for mainstream public use.

Recent academic work provides a strong technical foundation for NewsScope. Spinde et al. (2023) identify key forms of media bias—framing, omission, and labelling—while Rönnback et al. (2025) and Knutson et al. (2024) show that artificial intelligence models can scale to detect bias across large datasets. Datasets such as MBIB (Wessel et al., 2023), LIAR (Wang, 2017), and FEVER (Thorne et al., 2018) enable benchmarking and fact verification. For practical integration, NewsScope will use the Hugging Face Inference API (RoBERTa for bias detection [LLM], DistilBERT for sentiment analysis [LLM]), TruSt for disorder classification, spaCy for claim extraction, and PolitiFact for fact-checking. Political spectrum mapping will combine AllSides source ratings with article-level bias scores, supported by the matous-volf/political-leaning-deberta-large model [LLM] for ideological classification.

Together, these studies and technologies demonstrate both the academic feasibility and social importance of developing NewsScope — a free, accessible platform that combines bias detection, misinformation analysis, and global news visualisation.

Proposed Approach

The proposed development of NewsScope will follow a structured, research-driven process divided into five main stages: research, requirements gathering, analysis and design, implementation, and testing/evaluation.

Research

Initial research will focus on understanding the current landscape of media bias detection, misinformation analysis, and natural language processing (NLP) techniques.

This will include reviewing existing tools such as Ground News and Tanbih, as well as studying academic frameworks for automated bias detection and sentiment analysis.

Research will also explore the use of transformer-based models via Hugging Face Inference API (RoBERTa for bias detection [LLM], DistilBERT for sentiment analysis [LLM]), TruSt for disorder detection, and spaCy for claim extraction.

Additionally, the matous-volf/political-leaning-deberta-large model [LLM] will be evaluated for its ability to classify political leaning (left, centre, right) in news content.

Integration methods for fact-checking APIs (primarily PolitiFact, with Snopes and IFCN as optional) will also be assessed.

Requirements Gathering

Functional requirements will include:

– News aggregation from RSS feeds, NewsAPI, and GDELT

– Bias and sentiment classification

– Detection of misinformation, disinformation, and malinformation

– Ideological spectrum visualisation (left → right, with potential future expansion to a 2D compass)

– Personalised bias profile for users

Non-functional requirements will address system responsiveness, scalability, data security, and accessibility across devices.

User personas will also be developed to identify how different audiences (e.g. casual readers, researchers, journalists) engage with the app's bias insights.

Analysis and Design

The system will use a multi-layer architecture.

– **Frontend**: Flutter (Dart) Android app with Firebase Auth

– **Backend**: FastAPI (Python) hosted on Render, with APScheduler for ingestion jobs

– **NLP Engine**: Hugging Face Inference API [LLM], TruSt, spaCy, PolitiFact, and the matous-volf DeBERTa model [LLM] for political leaning classification

– **Database**: Supabase (Postgres) for articles, analysis results, and user profiles; Supabase Buckets for archiving

– **Infrastructure**: Cloudflare CDN for static assets, GitHub Actions for CI/CD

The UI will present interactive ideological spectrum visualisations, positioning media outlets or articles along a left–right scale based on detected bias. Additional components will include sentiment indicators, fact-checking summaries, and user bias profiling.

Implementation

Implementation will follow an iterative development cycle.

– Phase 1: Backend setup — FastAPI server, Supabase schema, ingestion pipeline, Hugging Face bias/sentiment integration [LLM]

– Phase 2: Frontend creation — Flutter interface linked to backend through RESTful APIs

– Phase 3: Advanced features — user bias profiles, PolitiFact integration, refined visual analytics

All development will be version-controlled using Git and GitHub, with automated deployment via GitHub Actions.

Testing and Evaluation

Testing will combine functional, usability, and performance assessments.

– Unit and integration testing will ensure backend accuracy and API stability

– Frontend usability testing will evaluate design clarity and accessibility across platforms

– NLP models will be evaluated using precision, recall, and F1-scores

– User feedback gathered during prototype demonstrations will guide final refinements before deployment

Deliverables

1. **Interim Report & Prototype**

   – Initial backend ingestion pipeline (RSS, NewsAPI, GDELT) and prototype bias/sentiment analysis via Hugging Face Inference API.

2. **Backend + NLP Engine**

   – FastAPI backend with Supabase Postgres integration.

   – Ingestion scheduler (APScheduler).

   – Deduplication and caching.

   – NLP engine: Hugging Face (RoBERTa for bias, DistilBERT for sentiment), TruSt (disorder detection), spaCy (claim extraction), PolitiFact API (fact-checking).

3. **Frontend App**

   – Flutter UI integrated with backend.

   – Firebase Authentication (email/Google OAuth).

   – Interactive spectrum visualisations (left–right).

4. **Advanced Features**

   – User bias profiling (based on liked articles).

   – Supabase Buckets for archiving older articles.

   – Cloudflare CDN integration for performance.

5. **Evaluation**

   – Model performance (precision, recall, F1).

   – Usability testing of Flutter UI.

   – System performance under continuous ingestion.

6. **Final Report, Dissertation, Technical Manual, Presentation**

Project Schedule

Overview of initial schedule

(September 2025 – April 2026)

September

– Project planning and background research

– Literature review and refinement of proposal

October

– Research NLP techniques (bias, sentiment, disorder detection)

– Identify APIs (RSS, NewsAPI, GDELT, PolitiFact)

– Confirm datasets (MBIB, LIAR, FEVER)

– Get Proposal signed off

– Steps 1-15 of the Feasibility Study completed

November

– Requirements gathering

– System architecture and design diagrams

– Define Supabase schema and retention policy

– Backend setup (FastAPI + Supabase Postgres)

– Ingestion scheduler (APScheduler)

– Prototype NLP integration (Hugging Face bias/sentiment)

– Interim report submission

December

– Flutter UI setup

– Firebase Auth integration

– API integration with backend

– Early testing of frontend/backend connectivity

January

– Implement user bias profiling

– Fact-checking integration (PolitiFact)

– Supabase Buckets archiving

– Cloudflare CDN + Render deployment

– Integration testing and performance tuning

February

– Final testing and evaluation

– Begin dissertation writing

– Finalise visualisations and user experience

March

– Complete dissertation

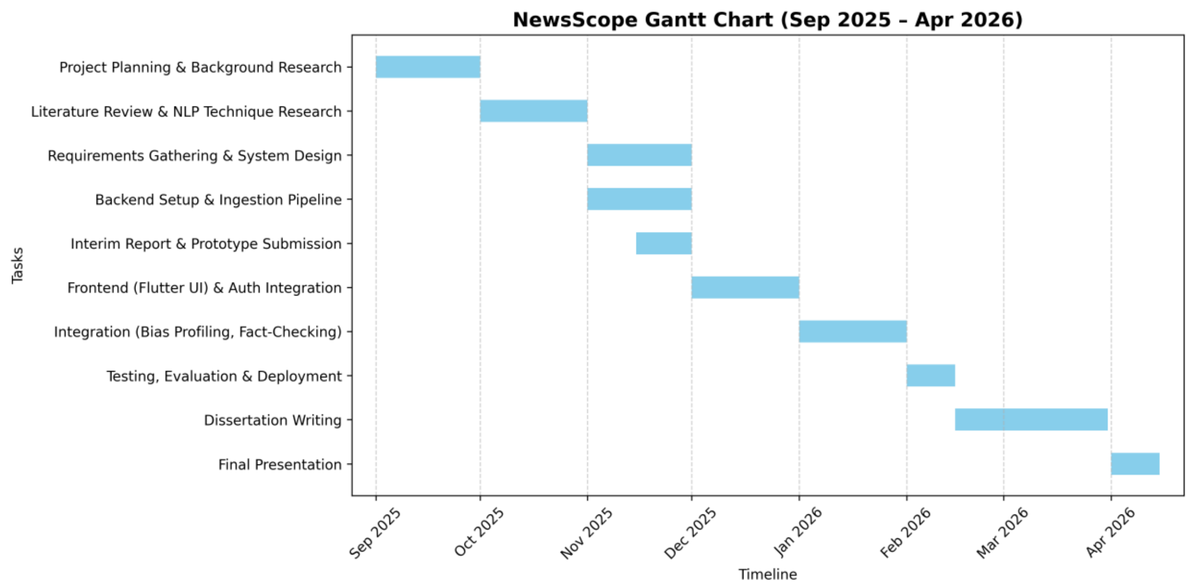
– Final documentation and technical manual

– Prepare presentation materials

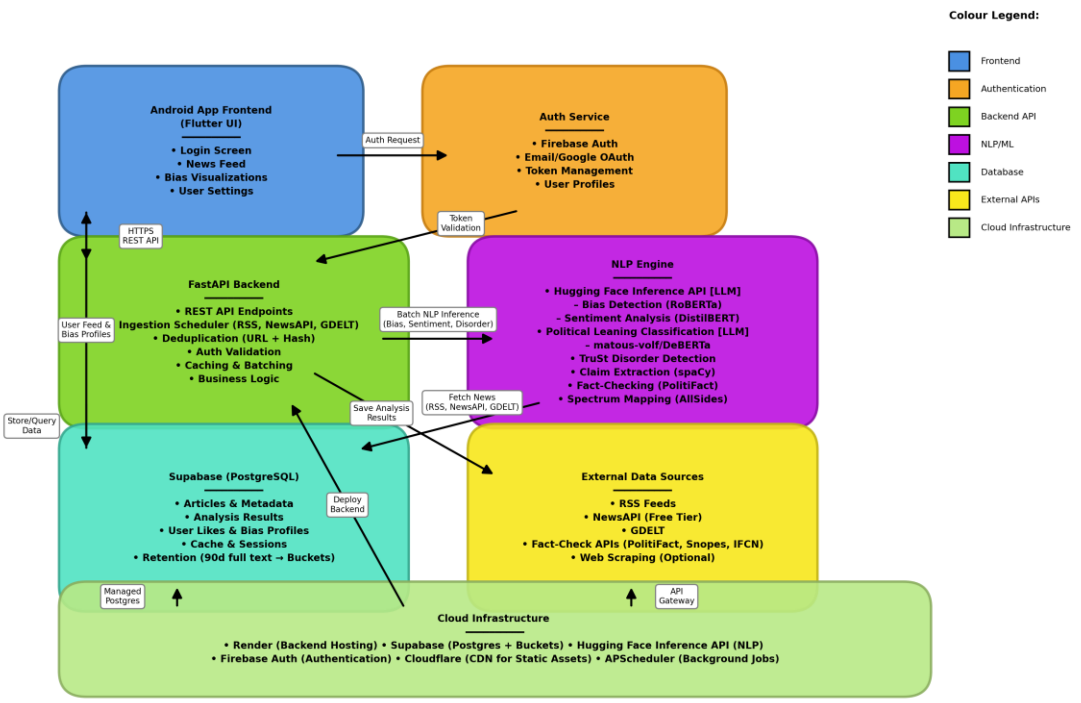
April

– Final presentation and project submission

Gantt Chart



NewsScope System Architecture


Android App Frontend (Flutter UI)
• Login Screen
• News Feed
• Bias Visualizations
• User Settings
Auth Request
Auth Service
• Firebase Auth
• Email/Google OAuth
• Token Management
• User Profiles
Token Validation
HTTPS REST API
FastAPI Backend
• REST API Endpoints
Ingestion Scheduler (RSS, NewsAPI, GDELT)
• Deduplication (URL + Hash)
• Auth Validation
• Caching & Batching
• Business Logic
User Feed & Bias Profiles
Batch NLP Inference (Bias, Sentiment, Disorder)
NLP Engine
• Hugging Face Inference API [LLM]
– Bias Detection (RoBERTa)
– Sentiment Analysis (DistilBERT)
• Political Leaning Classification [LLM]
– matous-volf/DeBERTa
• TruSt Disorder Detection
• Claim Extraction (spaCy)
• Fact-Checking (PolitiFact)
• Spectrum Mapping (AllSides)
Store/Query Data
Save Analysis Results
Fetch News (RSS, NewsAPI, GDELT)
Supabase (PostgreSQL)
• Articles & Metadata
• Analysis Results
• User Likes & Bias Profiles
• Cache & Sessions
• Retention (90d full text → Buckets)
Deploy Backend
External Data Sources
• RSS Feeds
• NewsAPI (Free Tier)
• GDELT
• Fact-Check APIs (PolitiFact, Snopes, IFCN)
• Web Scraping (Optional)
Managed Postgres
API Gateway
Cloud Infrastructure
• Render (Backend Hosting) • Supabase (Postgres + Buckets) • Hugging Face Inference API (NLP)
• Firebase Auth (Authentication) • Cloudflare (CDN for Static Assets) • APScheduler (Background Jobs)
Colour Legend:
Frontend
Authentication
Backend API
NLP/ML
Database
External APIs
Cloud Infrastructure

Technical Requirements

Software:

– Flutter, Dart, VS Code, Android Studio.

Backend:

– Python 3.11+ (stable release).

– FastAPI.

– APScheduler (background ingestion jobs).

– NLP libraries: Hugging Face Inference API, spaCy, TruSt.

Database:

– Supabase (Postgres + Buckets).

– pgAdmin/DBeaver for management.

Data Sources:

– RSS feeds.

– NewsAPI (Free tier).

– GDELT.

– PolitiFact API (primary fact-checking).

– Snopes/IFCN (optional).

Deployment:

– Render (backend hosting).

– Supabase (database + storage).

– Firebase Auth (authentication).

– Cloudflare (CDN for static assets).

– GitHub Actions (CI/CD).

Hardware:

– Windows PC (development).

– Android phone + emulator (testing).

Conclusion

The proposed project, *NewsScope*, aims to design and develop a cross-platform news analysis application that empowers users to recognise bias and misinformation in

English-language journalism. Through artificial intelligence and NLP, it will detect patterns of sentiment, framing, and ideological leaning, presenting them in a user-friendly and interactive interface.

By uniting strong technical foundations (FastAPI, Supabase, Hugging Face, Firebase, Render, Cloudflare) with social relevance, NewsScope will contribute both an innovative technological solution and a meaningful step toward improving public media literacy. The completed system — available as an Android app, with a companion website as a possible extension — will showcase how automation can support more transparent and informed news consumption worldwide.

References

AllSides (2025) *Media bias ratings: CNN, Fox News, BBC*. Available at: https://www.allsides.com/media-bias/media-bias-chart (Accessed: 6 October 2025).

BBC (2021) *BBC editorial values and impartiality guidelines*. Available at: https://www.bbc.co.uk/editorialguidelines (Accessed: 6 October 2025).

Pew Research Center (2020) *U.S. media polarization and the 2020 election: A nation divided*. Available at: https://www.pewresearch.org (Accessed: 6 October 2025).

Newman, N., Fletcher, R., Schulz, A., Andi, S. and Nielsen, R. K. (2024) *Reuters Institute digital news report 2024*. Reuters Institute for the Study of Journalism. Available at: https://reutersinstitute.politics.ox.ac.uk (Accessed: 6 October 2025).

Ground News (2025) *Ground News: Compare headlines from different perspectives*. Available at: https://ground.news (Accessed: 6 October 2025).

Tanbih Project (no date) *Tanbih news aggregator and analysis system*. Qatar Computing Research Institute. Available at: https://tanbih.org (Accessed: 6 October 2025).

AlKhamissi, B., AlKhamissi, M., Ismail, A., Elbassuoni, S. and Farghaly, A. (2025) *PureFeed: A machine learning-driven news aggregator for unbiased and sensationalism-free journalism*. ResearchGate. Available at: https://www.researchgate.net/publication/391814829 (Accessed: 6 October 2025).

Spinde, T., Hinterreiter, S., Haak, F., Ruas, T., Giese, H., Meuschke, N. and Gipp, B. (2023) *The media bias taxonomy: A systematic literature review on the forms and automated detection of media bias*. arXiv. Available at: https://arxiv.org/abs/2312.16148 (Accessed: 6 October 2025).

Rönnback, R., Achter, V., Hendriks, C., D'Souza, S. and García, A. (2025) 'Automatic large-scale political bias detection of news outlets', *PLOS ONE*, 20(5), e0321418. doi: 10.1371/journal.pone.0321418.

Knutson, B., Hsu, T. W., Ko, M. and Tsai, J. L. (2024) 'News source bias and sentiment on social media', *PLOS ONE*, 19(3), e0305148. doi: 10.1371/journal.pone.0305148.

Wessel, M., Horych, T., Ruas, T., Spinde, T. and Gipp, B. (2023) *MBIB: The media bias identification benchmark task and dataset collection*. arXiv. Available at: https://arxiv.org/abs/2304.13148 (Accessed: 6 October 2025).

Wardle, C. and Derakhshan, H. (2017) *Information disorder: Toward an interdisciplinary framework for research and policymaking*. Council of Europe. Available at: https://rm.coe.int/information-disorder-report-november-2017/1680764666 (Accessed: 6 October 2025).

Wang, W. Y. (2017) '"Liar, liar pants on fire": A new benchmark dataset for fake news detection', *ACL / arXiv*. Available at: https://arxiv.org/abs/1705.00648 (Accessed: 6 October 2025).

Thorne, J., Vlachos, A., Christodoulopoulos, C. and Mittal, A. (2018) 'FEVER: A large-scale dataset for fact extraction and verification', *Proceedings of NAACL-HLT 2018*. Available at: https://aclanthology.org/N18-1074/ (Accessed: 6 October 2025).

International Fact-Checking Network (no date) *IFCN / Poynter*. Available at: https://www.poynter.org/ifcn/ (Accessed: 6 October 2025).

PolitiFact (no date) *PolitiFact*. Available at: https://www.politifact.com (Accessed: 6 October 2025).

Snopes (no date) *Snopes*. Available at: https://www.snopes.com (Accessed: 6 October 2025).

Volf, M. (2023) *Political leaning classification using DeBERTa*. Hugging Face. Available at: https://huggingface.co/matous-volf/political-leaning-deberta-large (Accessed: 6 October 2025).

Appendix A: First Project Review

Title: Machine Learning & Credit Card Fraud Detection

Student: Hannan Ahmed (C19742501)

Description (brief): A web application that detects fraudulent credit card transactions using machine learning algorithms (primarily Random Forest). The system analyses the synthetic PaySim dataset, performs feature engineering, and provides both real-time transaction simulation and batch dataset analysis with visualization capabilities.

What is complex in this project: The complexity lies in handling a highly imbalanced dataset (only 0.13% fraudulent transactions), implementing multiple ML algorithms for comparison, developing custom feature engineering techniques, and integrating a trained model with a Flask-based web application that supports both simulation and dataset upload functionality.

What technical architecture was used: Two-tier architecture: Python/Flask backend with PostgreSQL for data processing and ML model integration (using scikit-learn, Random Forest); HTML/CSS/JavaScript frontend with visualization libraries (Plotly, D3.js); model serialization via pickle files; deployment considerations included Docker and Heroku.

Explain key strengths and weaknesses of this project, as you see it:

Strengths: Comprehensive approach to data preprocessing and feature engineering; thorough comparison of multiple ML algorithms; effective handling of class imbalance through oversampling; clean, minimalistic UI design; strong documentation of the iterative development process.
Weaknesses: Limited to synthetic data (PaySim) which may not fully represent real-world fraud patterns; model trained on only 60,000 rows rather than full dataset due to computational constraints; no user authentication system; limited to specific data format; potential security vulnerabilities not fully addressed for handling sensitive financial data in production.

Appendix B: Second Project Review

Title: Visualizing and Predicting Golf Performance Using Machine Learning

Student: Brandon Kelly (C19428962)

Description (brief): A web application that visualizes PGA Tour player statistics and predicts tournament performance using machine learning. Features include live leaderboards, player comparison tools (radar/bar charts), historical data analysis (2010-2018), and logistic regression models predicting top 10 finishes and tournament wins with 77% and 86% accuracy respectively.

What is complex in this project: The complexity involves implementing multiple ML algorithms (logistic regression, random forest) with extensive model evaluation, handling imbalanced datasets for binary classification, creating interactive data visualizations (Chart.js radar/bar charts), integrating real-time tournament data via third-party APIs, and managing cross-component communication between React frontend and Django REST backend.

What technical architecture was used: Two-tier architecture: Django REST framework backend with PostgreSQL database (Dockerized), React frontend with Material UI/Bootstrap/Chakra for styling, Chart.js for visualizations, scikit-learn/NumPy/Pandas for ML implementation, News API and Rapid Slash Golf API for external data, AWS for potential deployment.

Explain key strengths and weaknesses of this project, as you see it:

Strengths: Comprehensive ML model evaluation using multiple metrics (accuracy, precision, recall, F1, AUC-ROC, cross-validation); effective data visualization with multiple chart types enabling easy player comparisons; successful implementation of live data integration; well-structured MVT architecture with clear separation of concerns; thorough testing approach including unit, integration, and usability testing; clean, responsive UI design following established design principles.
Weaknesses: Limited to historical data (2010-2018) rather than current season statistics; prediction models require manual user input rather than automated data retrieval; class imbalance issues affecting precision/recall scores; reliance on single third-party APIs creates potential points of failure; comparison limited to two players from same year; overfitting concerns with random forest model (98% test accuracy suggesting memorization rather than generalization).

Appendix C: Prompts Used with ChatGPT

Can you format this reference list to use Harvard Referencing?

Appendix D: