DATA 236 Term Project Report

MyNewsMate:

AI-Powered Personalized News Recommendation System

Aayush Anil Patil

Mayuresh Pandey

Nikhil Swami

Padmanabh Rathi

Rupeshkumar Motipawale

Suyog Sanjeevan Jadhav

Professor Ming-Hwa Wang

DATA 236: Distributed Systems for Data Engineering San Jose State University

Acknowledgements

We deeply appreciate Professor Ming-Hwa Wang for his wisdom in teaching Distributed Systems and Data Science on the Cloud because his expertise established the essential framework for this project. Our research development reached success by his constant guidance along with mentoring support which trained our abilities and solidified our project completion. The researchers whose research work inspired us have played a major role in guiding our approach as we developed and tested our concepts.

Table of Content

I. Abstract

Users must struggle in the present digital epoch to detect relevant news information which remains true to fact because news content production rates reach incredible peaks by the minute. The challenge grows worse when we consider the widespread use of generic content delivery systems while these systems do not consider how users behave and what they prefer. The MyNewsMate project proposes a custom news gathering system that implements smart filtering solutions in combination with dynamic content delivery methods to handle current issues. The software utilizes Django programming language together with RESTful API features and Docker containers for decentralization to provide automatic scaling and smooth deployment capabilities. News content personalization occurs through a combination of natural language processing (NLP) along with sentiment analysis and adaptive user modeling that the platform utilizes. The system utilizes these tools to study user behavior patterns which enables it to show news content that fits changing interests and emotional preferences. The solution tackles important issues including duplicate article copies and new user difficulty along with bias-concealed news recommendations. This report examines the fundamental reasons behind the system creation as it outlines critical system elements and technical approaches for developing an ethical user-friendly news recommendation platform.

II. Introduction

In today's rapidly evolving information landscape, users are constantly overwhelmed by the sheer volume, speed, and diversity of digital news. With thousands of articles published across various platforms each day, it has become increasingly difficult for readers to find content that is both relevant and reliable. Traditional news aggregators attempt to unify sources but often rely on static, one-size-fits-all recommendation techniques based solely on popularity or recency. These systems fail to consider the user's evolving preferences, emotional states, or contextual behaviors, resulting in disengagement and repetitive content exposure.

The primary objective of this project is to design and implement MyNewsMate—an intelligent, scalable, and personalized news aggregation platform that delivers curated content based on user behavior, emotional sentiment, and reading patterns. This project investigates how adaptive user modeling, powered by machine learning and natural language processing, can enhance content relevance and diversity in real-time news delivery.

Our approach integrates several advanced components: a relevance prediction engine that scores articles using interaction history (e.g., click patterns, reading time), a sentiment analysis layer that evaluates emotional tone, and clustering mechanisms that eliminate redundancy while preserving topical diversity. The platform is engineered using Django, RESTful APIs, and deployed via Docker containers for scalability. WebSocket-based real-time updates and ASGI integration ensure timely content refreshes.

This project directly aligns with the themes of this course—Distributed Systems and Data Engineering—by emphasizing scalable backend infrastructure, modular API design, real-time communication protocols, and containerized deployment. Alternative systems either overlook emotional intelligence or lack scalability and adaptability. Our solution, by contrast, not only responds to user preferences as they evolve but also fosters a more engaging and ethically conscious news-reading experience. The platform also tackles known issues such as the cold-start problem, content duplication, and algorithmic bias.

In essence, MyNewsMate is an exploration of how emerging AI and system design principles can converge to create more meaningful and inclusive digital experiences. It redefines how individuals engage with information, combining technical innovation with user-centered thinking to deliver a more responsive and trustworthy news ecosystem.

III. Theoretical Basis and Literature Review

Project Problem

In today's fast-paced digital world, an overwhelming amount of news content is continuously produced and spread across countless online platforms. Thousands of articles are published every hour by a wide range of sources—from major media outlets to small blogs—flooding users with information. This makes it increasingly difficult to sift through the content and find news that is relevant, trustworthy, and aligned with individual interests.

Users are often shown generic or repetitive articles that don't reflect their personal preferences. Much of the news overlaps across sources, offering slight variations on the same story. The presentation is frequently disjointed, making it hard for users to extract meaningful insights. Traditional news sites and aggregators usually rely on popularity or recency, offering the same feed to all users without accounting for individual tastes. This reduces engagement and satisfaction, especially for those who prefer specific topics, regions, or viewpoints.

At the heart of the problem is the lack of an intelligent system that can collect news from trusted sources and personalize it based on the user's evolving interests and reading patterns. Such a system should filter, organize, and streamline content dynamically, while also managing different formats, reducing duplication, and promoting viewpoint diversity to avoid filter bubbles.

Mathematical formulation:

For a user u and a set of news articles $N = \{n_1, n_2, ..., n_{\square}\}$, our goal is to learn a ranking function $f(u, n) \rightarrow$ score that gives a relevance score of n to each article n based on its agreement with the preferences, history, contextual cues of the user. The system should then return a personalized feed $R \subseteq N$ where the articles are ranked in descending order of the relevance scores such that the most suitable and engaging content appears first.

Theoretical Background:

Earlier in the discussion, it was noted how information overload, content fragmentation, and insufficient personalization are core issues in the contemporary news ecosystem. Traditional news aggregators mark content to user interests on a static basis, and hence suffer from low user engagement and content fatigue, which is referred to in some literature as ‘content exhaustion’. The problem requires a new approach that is more intelligent and responsive by utilizing sophisticated algorithms, natural language processing, and modeling of the user's knowledge about them.

As with other aspects of life, news systems also learn to provide more advanced informational services. These systems seek to understand individual users’ preferences and tailor a content feed to their specific habits and mood. Unlike static news websites, intelligent systems respond in real-time to fresh content and evolving user activity. The objective is no longer fulfillment of keyword-based queries but rather fetch meaningful contextually relevant information that is timely and of interest to that user.

The task of relevance prediction is the core of any such system. It observes a user's clickstream, reading history, and interaction with content to then rank incoming articles based on how well they correspond to the user’s ever-evolving interests.

These rankings follow the system’s complex understanding of both the content and the user. One consideration involves textual similarity and sentiment alignment as the system assesses how closely the tone, topicalities, and language of new articles match the user’s previously engaged content. For example, if the user frequently reads technology-related articles with neutral or positive sentiment, the system learns to show that sentiment orientation in future contextual understanding. Likewise, contextual cues, such as when an article is being viewed (time of day), what kind of device the person is using, or what is trending at that moment, also shape the showing of an article to the user. A user may prefer more cerebral reading material in the day on their desktop and on mobile would rather have informational light reads at night, or vice versa. Behavioral signals, like how often a user clicks on articles in particular categories, how long they remain engaged with articles in a certain category, or what types of articles they avoid, act as essential feedback loops for the user. Over time the system learns to refine the feed to continually reflect what the user finds interesting and useful.

Modern systems use Natural Language Processing (NLP) techniques like named entity recognition, sentiment analysis, and topic modeling to process raw article data. These representations are then fed into scoring functions or machine learning models that output ranked lists of content.

Related Research:

Content-Based Deep Learning Approaches

In the early days of deep learning for personalized news recommendation, most systems leaned heavily on content-based models. These models mainly focused on analyzing the text of news articles—like the titles and summaries—to understand what they were about. Two well-known examples are NAML (Wu et al., 2019) and NRMS. They used techniques like convolutional neural networks (CNNs) and multi-head self-attention to process and encode the news content. Based on these encoded articles, the systems would then build user profiles by looking at what articles a person had read before, usually in the order they were read.

Strengths: These approaches were great at capturing detailed user preferences, especially for people who tend to stick to certain topics or have predictable reading habits. Since the models analyzed news at the word or sentence level, they were able to pick up on subtle meanings and nuances in the content.

Limitations: One big limitation is that these models viewed each user in isolation. That means they didn't take advantage of what other users were reading, missing out on patterns and trends that happen across the user base. They also struggled to adapt to bigger shifts, like a sudden change in what's popular globally or changes in what a user is interested in at a particular time.

Graph-Based Collaborative Methods

To overcome the limitations of treating each user in isolation, researchers started turning to graph-based techniques. One standout example is GLORY (Yang et al., 2023), which takes a more holistic approach by building a global interaction graph. In this graph, both users and news articles are represented as nodes, and their interactions—like clicks or views—form the connections (edges) between them.

What makes GLORY special is how it combines personal reading history with broader trends across all users. It does this using Gated Graph Neural Networks (GNNs), which are designed to learn patterns from these kinds of complex graphs. So even if a user has only read a few articles, GLORY can still make smart recommendations by looking at what similar users are interested in. As the authors put it: *"We propose a global-aware historical news encoder that captures hidden behavior patterns across users..."* — Yang et al., 2023

Strengths: Graph-based models are great at finding indirect relationships, which is super useful for new or less active users (a.k.a. the cold-start problem). They also naturally bring in collaborative signals—what others are reading—even without needing explicit ratings or feedback.

Limitations: The downside is that these models can be heavy on computing power. Building and constantly updating a massive interaction graph in real-time isn't easy, especially when speed and responsiveness are critical for a production system.

LLM-Augmented Semantic Enrichment

A newer and exciting direction in news recommendation taps into the power of large language models (LLMs) like GPT-4 to better understand what news articles are really about. One recent paper, *"News Recommendation with Category Description by a Large Language Model,"* explores a creative idea: using LLMs to generate natural-language descriptions for broad news categories like “sports,” “economy,” or “technology.”

Instead of just relying on short and sometimes vague headlines, this method adds these rich, descriptive summaries to the input. So, for example, a news headline about a football match would be paired with a generated description of the "sports" category. This combined input is then fed into transformer-based models like BERT, helping the system better grasp the article’s true meaning. As the authors explain: *“Although prompts are prepared manually, the system automates semantic enrichment at scale…”* — LLM-based Recommendation, 2023

Strengths: This technique adds helpful context, especially when headlines alone don’t provide enough information. By enriching the content with structured, meaningful descriptions, the system becomes better at making accurate and relevant recommendations.

Limitations: That said, this approach does have its challenges. It depends on well-crafted prompts and accurate category labels, which might not always be available or perfectly reliable. Plus, the descriptions are the same for everyone—they don’t adapt to each individual user’s unique way of interpreting content, which could limit personalization.

Advantages and Disadvantages to this Research

Highly Personalized User Experience

An AI-powered MyNewsMate tailors content to the individual user, creating a unique reading experience. By tracking reading behavior, search queries, and preferred topics, the AI model continuously learns and adapts to evolving interests. This personalization not only improves user satisfaction but also encourages prolonged engagement with the platform. Instead of bombarding users with generic headlines, the system filters and delivers articles aligned with their preferences, ensuring that every session feels relevant and valuable.

Time Efficiency and Enhanced Productivity

In today's fast-paced world, users often don't have the time to sift through multiple news sites for relevant information. AI solves this problem by automatically curating a feed of personalized, relevant, and real-time news. This eliminates the need for manual searching and content filtering, saving significant time. For professionals, students, or business users, this means they can quickly stay informed without distraction, increasing daily productivity.

Real-Time Aggregation from Multiple Sources

One of the biggest strengths of this project is its ability to pull news from various

sources—international news outlets, blogs, social media, and niche publishers—in real-time. This not only ensures users are always up to date with the latest developments but also helps in eliminating source bias by offering a comprehensive, 360-degree view of a topic. Users can compare how different media portray the same event, which promotes informed decision-making.

Advanced AI Insights: Sentiment & Trends

Beyond just delivering news, the aggregator can use natural language processing (NLP) and machine learning to analyze sentiment (positive, negative, neutral) and identify trending topics. This can help users understand public opinion or market sentiment, which is crucial for marketers, analysts, or researchers. For example, a user interested in tech might see trends in AI or cybersecurity, along with sentiment scores from public discourse—giving deeper context than headlines alone.

Scalability and Multi-Platform Compatibility

The backend AI system is scalable, meaning it can serve millions of users simultaneously with consistent performance. It can easily expand to support more categories, languages, and regions. Additionally, the system can be deployed across web browsers, mobile apps, and even smart devices with voice assistants. This cross-platform compatibility ensures that users can access their personalized news feed anytime, anywhere, making the service highly accessible.

Disadvantages

Filter Bubble and Echo Chamber Effect

One significant drawback of personalization is the creation of a "filter bubble"—where users are exposed only to content that aligns with their existing views. Over time, this narrows their perspective, reinforcing biases and creating an echo chamber. This not only affects how users perceive events but can also contribute to the spread of misinformation and polarization in society.

Privacy and Data Security Concerns

To personalize content, the system needs access to user data such as browsing history, clicks, reading time, and location. While this data is vital for building accurate user profiles, it raises serious privacy issues. If data isn't handled with care—via encryption, secure storage, and compliance with laws like GDPR—it could lead to breaches or misuse, undermining user trust. Bias in Algorithms and Source Selection

AI models are only as good as the data they're trained on. If the underlying dataset includes biased sources or outdated information, the system may inherit and amplify these biases. Additionally, if the aggregator favors certain publishers or regions, it might skew the news feed, leading to an imbalanced view of global events. This can impact the objectivity and trustworthiness of the platform.

High Maintenance and Operational Complexity

AI-based systems require regular monitoring and updates. From maintaining web

scrapers and third-party APIs to tuning models and handling user complaints, the backend operations can become resource-intensive. Bugs, server downtimes, or algorithmic errors can significantly affect the user experience. Moreover, keeping the platform scalable and responsive requires a skilled technical team and ongoing infrastructure investments.

Solution

MyNewsMate is a Django-based web application designed to curate and deliver news articles tailored to individual user preferences. By leveraging Artificial Intelligence (AI) and Machine Learning (ML) techniques, particularly Natural Language Processing (NLP), the system offers a customized news feed experience.

System Architecture

Frontend (User Interface)

We have used HTML, CSS, JavaScript (with jQuery or React for dynamic content) technologies. We are providing below features.

1. User Registration and Authentication: Secure user accounts with email verification and password management.
2. Personalized News Feed: Display of curated news articles based on user preferences.
3. Responsive Design: Optimized for both mobile and desktop viewing to ensure accessibility across devices.

4. Interactive Elements: Options for users to bookmark, like/dislike, and share articles.

Backend

We are using Django Web Framework with below components.

1. Django REST Framework: Facilitates the creation of RESTful API endpoints for accessing and managing news data.
2. Database: Utilizes SQLite for development and PostgreSQL for production environments to store user data, preferences, and article metadata.
3. User Management: Handles user authentication, profile management, and preference settings.

AI/ML Integration

Natural Language Processing is Employed to analyze and categorize news articles, enabling the system to match content with user interests effectively. We will do Sentiment Analysis to Assess the sentiment of news articles to provide insights into the tone of the content, allowing users to filter news based on sentiment and customized feed. Recommendation Engine to Implements collaborative and content-based filtering techniques to suggest articles that align with user behavior and preferences.

News Aggregation and Processing

Data Sources: Aggregates news from various sources using RSS feeds, third-party news APIs (e.g., NewsAPI), and web scraping techniques. Preprocessing the news Involves cleaning and normalizing text data, extracting key entities, and categorizing articles into relevant topics to enhance the recommendation process.

Where solution differs

MyNewsMate distinguishes itself from conventional and existing news aggregation

platforms through a unique combination of personalization, AI-driven intelligence, modular architecture, and user-centric design. Unlike generic news apps or basic RSS feed readers, our system is built from the ground up to offer a deeply personalized, intelligent, and scalable experience. Below are the specific ways in which our solution stands out from others:

Unlike traditional news platforms that group users based on broad categories or deliver

trending content to everyone, our solution leverages machine learning and natural language processing (NLP) to build individual user profiles. The system learns from user behavior—including their reading history, likes/dislikes, time spent on articles, and topic preferences—to dynamically tailor news content. This continuous learning loop ensures that the more a user interacts with the platform, the better the recommendations become over time. Most other aggregators use only keyword filtering or category matching. Our engine uses semantic analysis, topic modeling, and content-based collaborative filtering, ensuring recommendations are contextually relevant—not just syntactically similar.

We integrate a sentiment analysis layer into our recommendation pipeline that analyzes the tone (positive, neutral, or negative) of each article. This gives users the option to filter content by mood, which is particularly helpful for users who may prefer uplifting or more balanced news. What sets us apart: Mainstream aggregators rarely give users control over the emotional tone of news. Our system empowers users to consume content that aligns with their emotional and mental wellness preferences, a feature especially valuable in today's often overwhelming news landscape.

Our aggregator fetches news from multiple verified sources using both APIs and

real-time web scraping, and categorizes them using contextual NLP tagging techniques like Named Entity Recognition (NER) and keyword extraction. This ensures articles are not only pulled in real time but also accurately categorized and labeled, allowing users to explore articles with contextual tags like "AI", "Elections", or "Climate Crisis".

What sets us apart: Unlike aggregators that rely on predefined categories or

metadata from sources, our system intelligently categorizes content on-the-fly using NLP, giving users more precise and rich browsing options

Why is it better

MyNewsMate distinguishes itself from conventional and existing news aggregation platforms through a unique combination of personalization, AI-driven intelligence, modular architecture, and user-centric design. Unlike generic news apps or basic RSS feed readers, our system is built from the ground up to offer a deeply personalized, intelligent, and scalable experience. Below are the specific ways in which our solution stands out from others:

Advanced AI-Powered Personalization Engine

Unlike traditional news platforms that group users based on broad categories or deliver trending content to everyone, our solution leverages machine learning and natural language processing (NLP) to build individual user profiles. The system learns from user behavior—including their reading history, likes/dislikes, time spent on articles, and topic preferences—to dynamically tailor news content. This continuous learning loop ensures that the more a user interacts with the platform, the better the recommendations become over time.

Most other aggregators use only keyword filtering or category matching. Our engine uses semantic analysis, topic modeling, and content-based collaborative filtering, ensuring recommendations are contextually relevant—not just syntactically similar.

Sentiment-Aware News Filtering

We integrate a sentiment analysis layer into our recommendation pipeline that analyzes the tone (positive, neutral, or negative) of each article. This gives users the option to filter content by mood, which is particularly helpful for users who may prefer uplifting or more balanced news.

What sets us apart: Mainstream aggregators rarely give users control over the emotional tone of news. Our system empowers users to consume content that aligns with their emotional and mental wellness preferences, a feature especially valuable in today’s often overwhelming news landscape.

Real-Time Multi-Source Aggregation with Contextual Tagging

Our aggregator fetches news from multiple verified sources using both APIs and real-time web scraping, and categorizes them using contextual NLP tagging techniques like Named Entity Recognition (NER) and keyword extraction. This ensures articles are not only pulled in real time but also accurately categorized and labeled, allowing users to explore articles with contextual tags like "AI", "Elections", or "Climate Crisis".

What sets us apart: Unlike aggregators that rely on predefined categories or metadata from sources, our system intelligently categorizes content on-the-fly using NLP, giving users more precise and rich browsing options.

Privacy-Focused and Customizable Architecture

We designed the system with user data privacy and modularity in mind. All user data is securely managed with options for GDPR-compliant handling. Moreover, the architecture allows easy integration of third-party APIs, additional content sources, and new AI modules without overhauling the core system.

What sets us apart: Many commercial platforms do not allow users or developers flexibility in how their data is used or how the system evolves. Our solution is open, extensible, and privacy-aware, making it ideal for academic, enterprise, and personal use.

Cross-Platform Scalability and Dockerized Deployment

While many news aggregators are platform-specific or rely on monolithic designs, our system is containerized using Docker, making it highly scalable and portable. It can run on cloud platforms like AWS, Heroku, or Azure, and can be scaled horizontally using Kubernetes for handling high-traffic environments.

What sets us apart: Our use of modern DevOps practices (Docker, environment variables, cloud-readiness) ensures that the solution is not only functional but production-grade and enterprise-ready.

How Our Solution Improves Upon Others

MyNewsMate stands out as a superior solution in the domain of digital news consumption by offering a robust, intelligent, user-first platform that blends personalization, scalability, and emotional awareness. The system addresses the core limitations of existing news aggregators and introduces several innovations that elevate the user experience, data relevance, and technological robustness. Below are the detailed reasons why our solution is better:

Deep Personalization with Intelligent User Modeling

Most traditional news platforms offer only surface-level personalization, such as category or keyword filtering. In contrast, our solution builds a dynamic, intelligent user profile that evolves with the user. It considers multiple data points including reading habits, click patterns, interaction time, preferences, and feedback to deliver content that truly reflects the user's interests.

Why it's better: Users don't just receive general content—they receive news that aligns with their personal interests, current mood, and evolving curiosity. This boosts engagement and user satisfaction over time.

Integrated Sentiment-Based Filtering

Our platform uses sentiment analysis to classify the tone of each news article as positive, neutral, or negative. Users can then choose the type of content they want to consume based on their emotional state.

Why it's better: Most platforms overlook the emotional impact of content. By offering users the ability to filter news by sentiment, we promote mental well-being and create a more empathetic digital experience.

Real-Time Aggregation with Contextual Awareness

We collect news in real time from multiple trusted sources, ensuring the latest and most comprehensive coverage. More importantly, we use contextual NLP techniques (like Named Entity Recognition and topic modeling) to automatically tag and classify news, going far beyond simple keyword matching.

Why it's better: Users gain access to more relevant and meaningful content that's categorized and labeled intelligently, enabling easier discovery and better understanding of current affairs.

IV. Hypothesis

With MyNewsMate, we want to see if a simple but powerful idea would work:

Is it possible that an intelligent, real time, AI based news system can deliver the correct, balanced news that users are more interested in than what is available on the traditional platforms?

We think the answer is yes!

The modules that are used in the construction of MyNewsMate are a set of intelligent modules, each of which performs a particular task. One fetches and filters the newest news, one examines the emotional stance the user adopts to news, one measures what articles the user likes, and one watches what's hot on the web. All these modules are connected to each other and want to run as a unit, updating always based on user interaction and incoming data.

A system like this, where news is ranked not just by popularity, but also on their personal interest and emotional impact, we believe, can help readers to feel less overwhelmed, reduce negativity overload, and giving the knowledge wise which matters to them.

Main Hypothesis

MyNewsMate will combine distributed system in which intelligent and smart modules work together toward setting better news experience for users than conventional platform. Older systems, like most apps out there, jumped with the same trending stories to everybody, whereas our system will adapt fast to every single user's interests, mood and reading habits. This will enable people to find news they really want to read without feeling swamped, dreading, emotionally drained.

Sub-Hypotheses

So we reduce our main hypothesis down to three separate things

1. **Agent-Based Design Helps Personalization**

   Each MyNewsMate agent has one task, and that is exactly what we want them to do. a. One agent finds news

   b. Second, another checks how emotional the articles are.

   c. Third, learn what kind of articles you like to read.

   By concentrating on a single task, each agent becomes extremely good at a single task. They then talk to each other and make decisions together. This helps with teamwork to create smarter, more personal news feed.

2. **Real-Time Reactions Are Better**

   News is always changing. News applications require extensive periods of time to display current information ranging from hours to complete days in some cases.

   MyNewsMate works in real time. That means:

   a. You get breaking news instantly

   b. The feed content updates right away whenever significant events such as political selections or emergencies take place.

c. The system adjusts its content display based on any detected mood fluctuations by your usage.

3. **Emotional Awareness Makes News Easier to Digest**

   Most robo-advisors just put people in buckets like “low risk” or “high risk.” But people are more complex than that. Our system learns from your choices, your goals, and your feelings about money. It updates your investment plan all the time. That way, it fits you better.

How It Works

Behind the scenes, MyNewsMate runs through a process of collaboration that ensures every user is able to access the best MyNewsMate experience. At its heart, the system finds articles from many forms of trusted sources that it monitors for new and breaking content continuously. As soon as the content is got it is right away analyzed with regard to its topical relevance and emotional tone. Each article is scored based on how well it appropriately caters to your reading preferences and how much emotional content the article offered you.

Your reading habits are then used to constantly updating profile that these scores are compared against. For instance, if you enjoy the news being full of articles about health and who likes happy content, the system learns to show you that sort of things. The system gets smarter about what to show you next as you click, read, skip more on the platform. It is quiet and adapts without your having to manually change settings.

At last, these are the most relevant and emotionally appropriate articles ranked and ordered into a live news feed which streamlines in real time. Every time you open the app of you refresh your feed you’re seeing the content that matters to you personally and not just what’s trending globally. By doing this you make sure that everything you receive is timely and in your emotional comfort and informational nuances.

Compared to Traditional

Traditionally, news with content delivery on most news apps could be made by using static models. They have fixed categories such as 'politics', 'sports' or 'entertainment' and believe that due to the fact that people always tend to prefer all of the three in the same way. Since these systems are updating almost at regular time intervals, ranging from hours, and even days here and there, they can end up missing the breaking news in the context — and not the later part — which is more likely to happen. Furthermore, they rarely indicate that perhaps you feel, or the emotional consequence of a response.

MyNewsMate, on the other hand, offers a more thoughtful approach. The content is updated in real time, and every second, gets data from almost every source possible. Yet it differs from at least two competitors in that it also puts out news according to your personal taste and emotional preferences. Its not simply just asking for you to select a few interests, its learning from your behaviour—how much time you spending into a report, the quantity of time you flick through a report, the variety of stuff you are pleased about an additional type of report.

Also it knows the tone of the article. News too can change what it shows you depending on how newsy and emotional they have been in your feed as a rule, but especially if the news has been newsy and emotional. At the strategy of information delivery, MyNewsMate plays the role of a sensitive company leading you away from becoming completely bothered. After all, these platforms don't simply want to cater to the trending news, they want the right news for you — and in terms of traditionally speaking, these platforms were having issues in doing so.

Why This Is Important

Every person holds various requirements regarding their news reading experiences. People require news updates periodically but sometimes they need to avoid disturbing news content. The reason MyNewsMate stands important is because its design acknowledges these needs.

a. The platform provides stress-free neutral news for users who experience overwhelming situations.

b. MyNewsMate provides calming neutral news to users who need this type of content during their examination preparation or challenging life events.

c. Movies displayed to you will be cheerful and cartoon-like when you are feeling positive.

This makes news reading:

● More enjoyable.

● Less overwhelming.

● Emotionally safe.

●Tailored just for you.

Test Methodology

To prove MyNewsMate works better than traditional platforms, we will run several tests.

Comparison Groups:

1. Traditional news apps like Google News

2. Rule-based apps with topic filtering

3. Our system (MyNewsMate)

Things we will measure:

- Relevance Score: How closely articles match the user's real interests
- Engagement Time: How long users spend reading articles
- User Satisfaction: Did users feel comfortable and informed?
- Emotional Fit: Was the content suited to the user's mood?
- Click-through Rate: How often users clicked recommended articles?

The research plan employs test users from different emotional and intent backgrounds and tracks system performance result.

Final Thoughts

The news world is fast-paced, noisy, and very stressful. MyNewsMate aims to change how people read news by making it:

a. Personal

b. Emotionally smart

c. Always relevant

MyNewsMate's approach to the news application is personalized based upon individual user needs. Rather than being just a news application the application acts like an understanding assistant. MyNewsMate responds to your desired content and keeps offering relevant news updates in a balanced manner based on user understanding. In the future, information like this could be considerably easier to access, more human, and more useful.

V. Methodology

Data Collection

We utilized a rich dataset comprising over 422,000 news pages, systematically grouped into clusters representing individual news stories. Each cluster encapsulates different webpages discussing the same news event from multiple sources and perspectives.

The dataset encompasses a wide range of content, with news articles categorized into four major domains: 152,746 in Business, 108,465 in Science and Technology, 115,920 in Entertainment, and 45,615 in Health. This diverse categorization provides a comprehensive foundation for the system to understand and adapt to different topical interests. By learning patterns across these varied domains, the model can effectively capture a broad spectrum of user preferences and deliver more personalized and relevant news recommendations.

To address content redundancy and better track the evolution of news stories, the articles in the dataset were grouped into clusters of similar stories. Each cluster represents a unique event or topic covered by multiple sources. The clustering breakdown includes 2,076 clusters for Entertainment, 1,789 for Science and Technology, 2,019 for Business, and 1,347 for Health. This structured grouping not only reduces repetitive content but also enables the system to present users with a more cohesive and streamlined news feed, enhancing clarity and overall user experience.

Beyond news content, the dataset includes 2-page browsing sessions—pairs of URLs where a user accessed a referring page and then a linked news page. These represent real-world user behavior and help capture implicit signals of interest and navigation patterns.

- Total sessions: 15,516 (covering 946 clusters)
  - Business: 6,091 sessions
  - Entertainment: 9,425 sessions

This behavioral data is crucial for training the personalization engine to understand user attention flow and content exploration habits.

Problem Solving Approach

The Personalized News Aggregator project was designed to address the challenge of information overload by building a centralized platform that delivers tailored news content based on individual user preferences. The problem was approached by first understanding the need for personalized recommendations, especially in an era where users are constantly exposed to vast amounts of news from diverse sources. The solution began with the acquisition and preprocessing of news data, either through publicly available RSS feeds or pre-collected datasets consisting of thousands of categorized headlines. Text data was cleaned using standard natural language processing techniques such as stop-word removal, tokenization, lowercasing, and lemmatization to prepare it for machine learning processing.

A user profiling mechanism was implemented, where user preferences were captured during account creation and updated over time based on interactions like clicks, likes, and bookmarks. These preferences were stored as user profile vectors. To compare news articles with user interests, each article was vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) technique. Cosine similarity was then used to calculate the closeness between the user profile and the news articles, allowing the system to rank and recommend the most relevant content.

The backend was developed using the Django framework, with RESTful APIs managing the flow of data between the frontend and backend. Background jobs like fetching and updating news content were managed using Celery and Redis. The architecture was designed to be modular and scalable, making future enhancements easier. To ensure a continuous feedback loop, user interactions were logged and used to update their preference vector, allowing the system to become more accurate and personalized over time. For new users with no interaction history, the system falls back to recommending trending or top articles across various categories.

Algorithm design

The algorithmic foundation of the Personalized News Aggregator revolves around delivering highly relevant news content to users by analyzing their interests and dynamically adapting to their behavior. The solution adopts a content-based recommendation approach, built using a combination of natural language processing (NLP), vector similarity, and user profile modeling. The following sections explain the algorithm’s components in detail.

Text Preprocessing

The initial stage of the algorithm involves preparing raw news article data for processing. News titles and descriptions are first cleaned using a series of NLP steps. These include converting text to lowercase, removing stop words (common but non-informative words like "the" or "and"), stripping punctuation, tokenizing the sentences into words, and lemmatizing them to reduce words to their root forms. This helps retain the semantic essence of the articles while reducing redundancy and noise in the data. These preprocessed tokens are the foundation for extracting useful features from the text.

Feature Extraction Using TF-IDF

Once the text is cleaned, each article is transformed into a numerical representation using the TF-IDF (Term Frequency–Inverse Document Frequency) vectorization method. TF-IDF computes how important a word is to a document in a collection, by weighing its frequency in a single article against its occurrence across the entire corpus. Words that are common in only a few documents are given more importance than those that appear everywhere. This step creates high-dimensional vectors that capture the uniqueness of each article's content. These vectors are then used for comparison with user interests.

User Profile Vector Construction

To personalize recommendations, the system constructs a profile vector for every user. When users register, they select topics of interest such as sports, politics, or technology. As they interact with the platform—by reading, liking, or saving articles—the system tracks their behavior. For each interaction, the TF-IDF vectors of the associated articles are aggregated to compute the user's profile vector. This profile represents the user's evolving preference for specific topics or keywords, and it serves as the reference for all future article comparisons.

Article Scoring Using Cosine Similarity

The recommendation engine scores each article by measuring its similarity to the user's profile vector using cosine similarity. Cosine similarity is a metric that calculates the angle between two vectors, with a score of 1 meaning identical orientation and 0 indicating complete dissimilarity. For each user, the system computes the cosine similarity between their profile vector and the TF-IDF vector of each available article. Articles are then sorted by descending similarity scores, and the top results are shown in the user's personalized news feed. This scoring mechanism ensures that users receive news closely aligned with their interests.

Real-Time Feedback and Adaptation

An important part of the algorithm is its feedback loop, which enables it to learn and adapt in real time. The system captures user interactions—such as clicking on a headline, liking an article, or bookmarking a story—and uses them to update the user's preference profile. Positive feedback (e.g., likes or long reads) increases the weights of relevant terms in the user vector, while negative or no interaction slightly reduces the influence of those terms. This dynamic update mechanism ensures that the recommendations stay aligned with the user's changing interests over time.

Cold Start Strategy

To address the cold start problem for new users, the system employs a fallback strategy. Since there's no interaction history initially, the system recommends trending news articles from various categories or the most popular stories of the day. This ensures users receive relevant content immediately after onboarding. As soon as they begin interacting with the platform, the system transitions from generic to personalized recommendations based on emerging preferences.

Additional Algorithmic Features

To further enhance recommendation quality, the system incorporates additional features. A content deduplication filter ensures that the same or similar articles—often published by multiple news sources—are not shown repeatedly. A diversity filter controls the number of recommendations from a single publisher or category to avoid overfitting the feed. Furthermore, the ranking logic may integrate additional weights based on article recency, popularity, or user-defined reading preferences such as preferred article length or reading frequency.

The Personalized News Aggregator blends several machine learning and NLP techniques to deliver a robust recommendation engine. Text preprocessing leverages basic NLP operations such as tokenization and lemmatization. Feature extraction is handled using TF-IDF vectorization, while cosine similarity measures are used to rank articles against user interests. Profile vectors are continuously updated based on user engagement, allowing the engine to adapt over time. For new users, the cold-start strategy ensures content remains engaging even before personalization begins. These algorithmic choices result in a lightweight, efficient, and highly responsive personalized news delivery platform.

Programming Languages

We will make use of the multiple programming languages each suited for the specific task in the application.

Python

- Used for backend development and implementation of personalization logic.
- Ideal for handling data processing, NLP, and future integration of machine learning algorithms.
- Django and Django REST Framework will manage API development and server-side logic.

HTML, CSS, Bootstrap

- Used for structuring and styling the frontend interface.
- Enables a clean and responsive user experience across devices.

ReactJS

- Employed for building a dynamic, single-page application (SPA).
- Facilitates real-time rendering of personalized news content.

JavaScript / Node.js (Middleware)

- Manages client-side behavior and real-time interactivity.
- Can be used in middleware if additional real-time services are added.

Databases

- **SQLite** will be used during development for simplicity.
- **PostgreSQL** will be adopted in production for robust and scalable data storage.

Docker

- Used to containerize the application, ensuring consistent environments across development, testing, and production.

Gunicorn & Nginx

- Gunicorn will serve the Python backend in production.
- Nginx will act as a reverse proxy and serve static files efficiently.

Git & GitHub

- Version control and collaboration.
- Tracks changes and manages source code across development cycles.

Technologies

The Personalized News Aggregator project is built using a combination of robust backend technologies, data science tools, web development frameworks, and utility libraries. These technologies work in tandem to enable real-time news aggregation, intelligent recommendation, user personalization, and seamless web interaction.

1. Web Framework: Django

At the core of the project is **Django**, a high-level Python web framework that enables rapid development of secure and maintainable web applications. Django provides a robust MVC (Model-View-Controller) architecture that separates application logic, data models, and UI templates. It handles routing, user session management, form processing, database integration, and security features like CSRF protection. In this project, Django powers the entire server-side infrastructure, including the recommendation logic, user profile tracking, and serving of dynamic content.

2. Natural Language Processing (NLP)

To interpret and analyze news content, the project makes extensive use of **NLP libraries** in Python. These tools are essential for text cleaning, feature extraction, and understanding semantics in article content.

- **NLTK (Natural Language Toolkit)**: Used for basic NLP operations such as tokenization, stop-word removal, and lemmatization. These processes reduce noisy data and convert raw text into standardized tokens suitable for vectorization.

- **scikit-learn**: Beyond machine learning, scikit-learn provides the **TF-IDF Vectorizer**, which transforms textual data into a matrix of numerical features based on term frequency and inverse document frequency. It also supports model training, evaluation, and similarity computation using cosine similarity.These libraries enable the transformation of unstructured news headlines and descriptions into structured vectors, making it possible to compare articles with user interests for personalized recommendations.

3. Machine Learning and Recommendation Logic

The personalized recommendation engine is primarily built using **content-based filtering** techniques. The machine learning aspect of the system includes:

- **TF-IDF Vectorization**: Converts each article into a numerical vector that reflects the importance of words specific to that document.
- **Cosine Similarity**: A mathematical technique to calculate similarity between user profile vectors and article vectors. Articles with higher similarity scores are ranked higher in the recommendation list.
- **Feedback Loop**: The algorithm adapts over time by incorporating user interactions (likes, saves, skips) into their profile vector, thus refining future recommendations.

The system avoids complex deep learning models to retain responsiveness and ensure lightweight operation on standard web infrastructure.

4. Database: PostgreSQL

The application uses **PostgreSQL**, an advanced open-source relational database, to manage persistent data. It stores:

- User credentials and session data
- User preferences and interaction logs
- Article metadata (title, category, URL, timestamp, etc.)
- System logs and analytics for monitoring

Django's ORM (Object Relational Mapper) interacts seamlessly with PostgreSQL to perform CRUD operations, manage schema migrations, and ensure data consistency.

5. Background Task Management: Celery + Redis

To ensure the news feed is always up to date, the system uses **Celery** to manage background tasks such as:

- Periodic scraping of news articles via RSS feeds or APIs
- Cleaning and preprocessing text
- Vectorizing and storing new articles

Celery is integrated with **Redis**, which acts as a message broker to queue and schedule tasks. This ensures the main application remains fast and responsive by offloading long-running jobs to the background.

6. Frontend and User Interface

While Django handles the backend, the frontend is built using **HTML5**, **CSS3**, and **JavaScript**, styled with frameworks such as **Bootstrap**. Django's templating engine renders dynamic content based on user interactions, profile data, and recommendation results. The frontend design ensures the user interface remains clean, responsive, and intuitive, enabling smooth navigation between categories and recommended articles.

7. Package and Environment Management

The project uses standard Python environment tools for dependency and version management:

- **pip**: Python package installer for managing libraries like Django, scikit-learn, NLTK, and Celery.
- **virtualenv**: Isolates the project's dependencies to avoid conflicts with global packages.
- **requirements.txt**: Lists all dependencies and their versions to ensure reproducibility and portability of the environment.

8. Deployment and Hosting

While the GitHub repository focuses on the source code, the architecture is compatible with deployment on platforms such as **Heroku**, **Render**, or **AWS EC2**. Deployment can be enhanced using:

- **Gunicorn**: A production WSGI server for running Django apps.
- **Nginx**: For serving static files and reverse proxying traffic to the Django backend.

- **Docker (optional)**: To containerize the application and make it scalable and portable.

Web UI Design

The user interface (UI) of a personalized news aggregator is deliberately made with a seamless, immersive, and user-centric experience. The frontend is designed in HTML5, CSS, Bootstrap, and ReactJS, the responsiveness and cross-device compatibility of these frameworks aid in providing a high-quality experience for the user. Similarly, the interface operates with a Single Page Application (SPA) scheme providing users to seamlessly go between views without full-page reloads, allowing for smaller loading times, a faster experience for the user, and an overall app-like experience that feels particularly important for real-time content that's being updated.

When users log into the application, they are greeted with a simple and interactive homepage with its branding and a few onboarding components. For example, if the user is new or returning. The new user is prompted to set up their account and is served two simple call-to-action buttons like "Get Started" or "Personalize My News". After clicking, a modal form pops up and begins to ask some questions about their topic preference (Sport, Politics, Technology, etc.), how often they read, the possible language options for the feed, and optional details for things such as demographic information like age and profession. To ensure data disproportion, we clearly validate all input fields using HTML5 built-in constraints and javascript validation checks, which keeps the number of errors on the input and preprocessing required on the backend.

For returning users, the homepage is built around a dashboard-like homepage that serves as the purpose of visiting personalized content. The dashboard displays personalized news articles that are curated into visual tiles, each listing a headline, a summary, a sentiment icon, the time of publication, and feedback icons to give a thumbs up or down and bookmark. The tiles also have a tooltip that briefly describes why the article was recommended when the tile is hovered over: "Based on your interest in Science & Technology". The dashboard-style homepage is designed with a left-hand navigation menu where users can filter article categories, sources, or time since publication, and a visually driven feed as the central part of the page that includes dynamic content that updates based on user preferences and user actions.

In an effort to facilitate interactivity, we have also added features that respond to user activity. Users can adjust their topic preferences by adjusting a slider, easily change sentiment tones (positive, neutral), and view visual summaries of previous reading interactions with visual summaries rendered using Chart.js proof of concept. These can be summary engagement charts by category, or rendering user interaction timeline (all the above categories shown visually, but only what a user needs) and users can easily click on and read. We have added to the users' ability to refresh their personalized feeds and retrain the recommendation logic on the prior interaction as well. There is a "News Discovery" section where the most current trending stories are identified and presented outside of what is expected from the user's profile, so as to diversify the user's profile and lessen filter bubbles.

The UI is also designed to support real-time interaction with toggles, dropdown filters, and hover features. Accessibility is considered at every level: placeholders and info icons help users understand form inputs, while tooltips and labels guide them through advanced options. During any potential delays—such as feed retraining or data retrieval—engaging loaders or motivational quotes are displayed to maintain user interest.

In summary, the design of the Web UI is centered around transparency, control, and customization. We have taken an opaque and complicated backend recommendation engine and transformed it into something highly visible and intuitive. Using interactive elements, visual feedback, and human-centered spatial design concepts, we aspire to create a platform that provides a streamlined and efficient and personalized content experience, but also gives users the opportunity to understand, explore, and trust the news they consume.

Deployment

The deployment of the Personalized News Aggregator involves preparing the application to run efficiently and reliably in a production environment. The goal of deployment is to make the Django-based news recommendation platform accessible to users via the web, with background processing, database support, and seamless delivery of personalized content. The deployment process involves setting up the server, installing dependencies, managing asynchronous tasks, configuring databases, and ensuring the application is secure, scalable, and always available.

1. Environment Setup and Dependency Management

Before deployment, it is essential to isolate the environment to prevent version conflicts. This is typically done using **virtual environments** like virtualenv or venv. A requirements.txt file is maintained to list all dependencies such as Django, Celery, Redis, scikit-learn, NLTK, and PostgreSQL adapters. When the deployment server is ready (e.g., a cloud VM or container), this file is used to install all required Python packages using pip.

Steps:

- Clone the project repository using Git.
- Create a virtual environment: python -m venv env
- Activate it and run pip install -r requirements.txt This ensures a reproducible and clean setup that mirrors the development environment.

2. Web Server and WSGI Configuration

In production, Django applications are not served using the built-in development server. Instead, they use a **WSGI (Web Server Gateway Interface)** server like **Gunicorn** to interface with web servers such as **Nginx**.

Gunicorn is responsible for running the Django app and managing multiple worker processes to handle concurrent requests. It is commonly paired with **Nginx**, which acts as a reverse proxy server. Nginx forwards incoming HTTP requests to Gunicorn and serves static files directly (like CSS and JS), improving performance.

Configuration Flow:

- Gunicorn runs the Django project using: gunicorn projectname.wsgi
- Nginx listens on port 80 and routes traffic to Gunicorn
- Static and media files are served by Nginx for speed and efficiency

This separation of concerns allows the application to scale effectively and ensures better security and performance.

3. Database Configuration (PostgreSQL)

In the production environment, the project uses **PostgreSQL**, a powerful open-source relational database. Unlike SQLite, which is suitable for development, PostgreSQL handles concurrent access and large-scale data much better.

On the deployment server:

- PostgreSQL is installed and configured.
- A production-ready database and user are created.
- The Django settings.py file is updated with the PostgreSQL credentials and connection parameters.
- Django migrations are run using: python manage.py migrate to initialize database schemas.

Optionally, **pgAdmin** or command-line tools can be used to monitor and manage the live database.

4. Background Task Scheduling (Celery + Redis)

To maintain up-to-date news content and handle background tasks like scraping, preprocessing, and vectorization, the project integrates Celery with Redis.

Celery is a task queue that runs asynchronous jobs. Redis acts as the message broker that queues these jobs and allows workers to pull them when ready. This setup ensures the main application remains responsive while heavy or scheduled tasks (e.g., fetching RSS feeds every hour) are handled in the background.

Steps:

- Redis server is installed and started.
- Periodic tasks (e.g., daily updates) can be scheduled using Celery Beat or system-level cron jobs.

This architecture ensures scalability and responsiveness, especially when processing large volumes of articles.

5. Security and Static Files Handling

In production, security settings must be carefully configured:

- Set DEBUG = False in settings.py
- Define ALLOWED_HOSTS to prevent host header attacks
- Configure SECRET_KEY, database credentials, and API keys using environment variables
- Use HTTPS via Let's Encrypt (certbot) for secure data transfer

Additionally, static and media files are collected using python manage.py collectstatic and served by Nginx from a designated directory to improve frontend load speed.

6. Optional: Docker-Based Deployment

For containerized deployment, the project can be Dockerized using a Dockerfile and docker-compose.yml. This allows packaging the app along with all its dependencies, environment configurations, and services (e.g., PostgreSQL, Redis) into isolated containers.

Benefits of Docker:

- Consistency across development, testing, and production
- Easy CI/CD pipeline integration
- Fast and reproducible deployments

In a Docker setup, services like Django, Redis, Celery, and PostgreSQL are defined as separate containers and orchestrated together using docker-compose up.

7. Cloud Hosting Platforms

The project can be deployed to any IaaS or PaaS provider. Common choices include:

- **Heroku**: Simple Git-based deployment with PostgreSQL add-ons and Redis support.
- **AWS EC2 / Lightsail**: Full control over VM setup, ideal for production workloads.

For any platform, deployment involves configuring environment variables, setting up the database, and ensuring continuous integration if needed (via GitHub Actions or Jenkins).

The deployment of the Personalized News Aggregator ensures the system is scalable, secure, and responsive to real-time updates. By using Django with Gunicorn and Nginx, PostgreSQL for data storage, Redis with Celery for background task management, and optionally Docker for containerization, the platform is production-ready and built to handle dynamic personalized content efficiently. The modular deployment setup allows easy customization for cloud or self-hosted environments and supports both monolithic and microservices-style architecture.

Prototype Features (if Time Permits):

The prototype of the Personalized News Aggregator showcases essential features that demonstrate its ability to deliver user-specific news content in real-time. Users can register, log in, and select their preferred news categories during sign-up, which initializes their personalized profile. The platform then generates a tailored news feed by comparing the user's interest vector with TF-IDF representations of articles using cosine similarity.

A clean and responsive interface displays curated articles with titles, summaries, sources, and direct links. As users engage—by reading, liking, or bookmarking articles—their profiles are updated dynamically, improving the relevance of future recommendations. To handle new users with no history, a cold-start mechanism displays trending articles across categories.

The system also includes periodic background updates using Celery and Redis to fetch and process new articles without disrupting the user experience. Additionally, category filters allow users to explore content manually, and a simple admin panel supports backend monitoring. The frontend is built with Django templates and styled using Bootstrap to ensure a smooth and user-friendly experience.

Output Generation

The Personalized News Aggregator project is designed to generate dynamic, personalized outputs for each user based on their interests and interaction history. The core output of the system is a customized news feed, which is continuously updated and ranked based on user preferences. This feed displays relevant articles tailored to individual users using content-based filtering, where the similarity between a user's profile vector and the TF-IDF representation of each article determines its ranking. The higher the similarity, the more prominently the article appears in the user's feed.

Each article in the output includes key information such as the title, summary, source/publisher, publication date, and a direct link to the full article. This structured output ensures that users can quickly scan through the feed and access stories that matter to them most. The output layout is built using Django's templating engine and rendered with HTML and Bootstrap to maintain a clean and responsive user interface.

Moreover, the system provides real-time output adjustments based on user interactions. When a user likes, clicks, or bookmarks an article, this feedback is recorded and used to refine the user's profile. Consequently, the next set of recommended articles becomes more aligned with the user's interests, demonstrating an adaptive and intelligent output generation mechanism.

The platform also supports categorized browsing. Users can manually filter articles by categories such as technology, health, sports, or business, providing more control over the output they wish to view. Additionally, a trending section or fallback output is generated for new users or in case of insufficient preference data. This ensures that all users receive meaningful content even without prior interaction.

Finally, the system's backend regularly updates the output data by fetching new articles through background tasks using Celery and Redis. These updates ensure that the feed remains fresh, relevant, and timely, reflecting the most recent news aligned with the user's evolving interests.

Hypothesis Testing:

We will establish some hypotheses to evaluate the effectiveness and accuracy of our personalized news aggregator system. The hypotheses we will construct will evaluate whether or not the system is functionally correctly from a technical perspective and whether or not it provides meaningful personalization in a way that is stimulating user engagement and relevance of the content.

Main Hypothesis

Comparing personalized recommendations with a generic or non-personalized feed, our primary hypothesis is that users will have objectively higher engagement (i.e., reading time, salience and satisfaction) with the personalized news recommendations derived from user and content clustering.

We plan to evaluate this by simulating multiple user types with distinct interest profiles and contrasting:

- Their interaction levels with personalized vs. non-personalized feeds.

- The diversity and relevance of content served.
- The recurrence of topics or redundant stories.

Supporting Hypotheses and Tests

1. User Engagement Accuracy:

We believe that users who see aligned content will have longer sessions, more article clicks, and lower bounce rates.his. That would be able to be tested via controlled A/B with the same users in personalized vs. generic feed.

2. Clustering Efficiency:

We expect that clustering news articles into storylines can reduce the redundancy of the content and improve the coherence of the feed. The assumption will be borne out by comparing cluster-based feeds to unclustered ones from the standpoint of user feedback and duplicate content exposure.

3. Sentiment Matching Impact:

Matching article sentiment with user preference (e.g., not serving very negative news to users who prefer neutral/positive tone) should enhance satisfaction. We will evaluate this through logging skip rates and reading time on sentiment-aligned vs non-aligned articles.

4. Feed Diversity Testing:

We will trial diversity constraints (e.g., interleaving articles from outside a user's dominant interest area) in an attempt to improve perceived feed richness. Hypothesis: Intentional diversity improves trust and content discovery without leading to an overall loss of relevance.

System Verification and Validation

Unit Testing of Logic:

Unit testing will be performed on each core function, such as user preference parsing, clustering algorithm and sentiment tagging. So if a user picks “Technology” and “Science,” their feed should prioritize those categories, which can be overridden by diversity rules.

Integration Testing:

After unit tests are done for components, we will run integration tests to check that complete pipeline (from registration to generation of feed) maintain consistency of flow of data and personalization logic.

Backtesting with Historical Data:

To mimic true-to-life behavior, we’ll run the system on historical user interaction data. As we accomplish this, we will track engagement changes as a function of personalization level, and correlate that with intuition that the behavior of the model is correct.

Constraint Checks:

We'll incorporate validation guidelines for article freshness, topic balance, and clustering thresholds. For example, unless the user specifically requests it, no one topic should take up more than a certain amount of the feed.

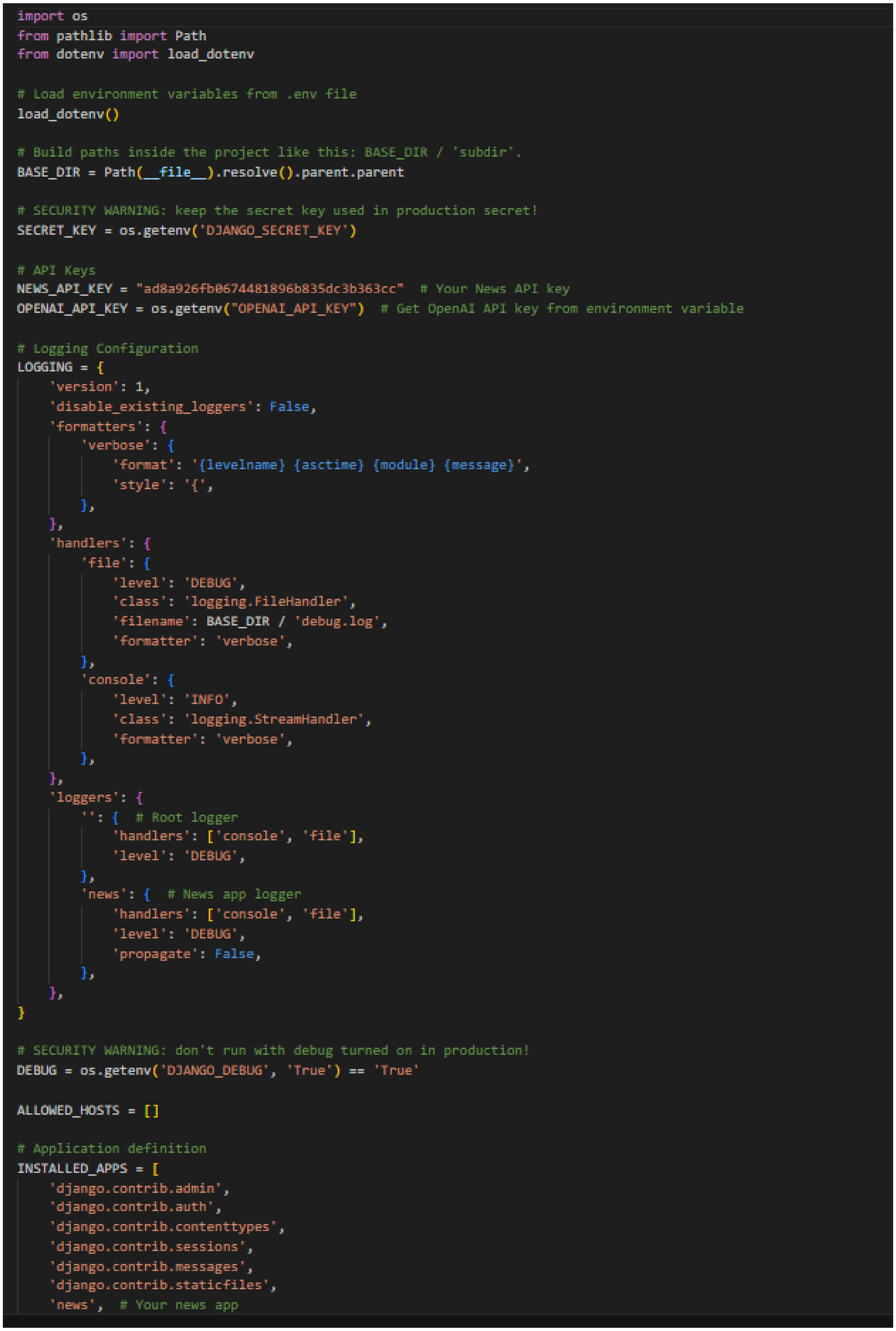
Correctness Verification:

To see if the created feed accurately reflects the user's tastes, we will introduce some test cases, such as a user who has a strong preference for tech and sports. Mismatch rates will be monitored, and if necessary, the recommendation scoring algorithm will be modified.

VI. Implementation

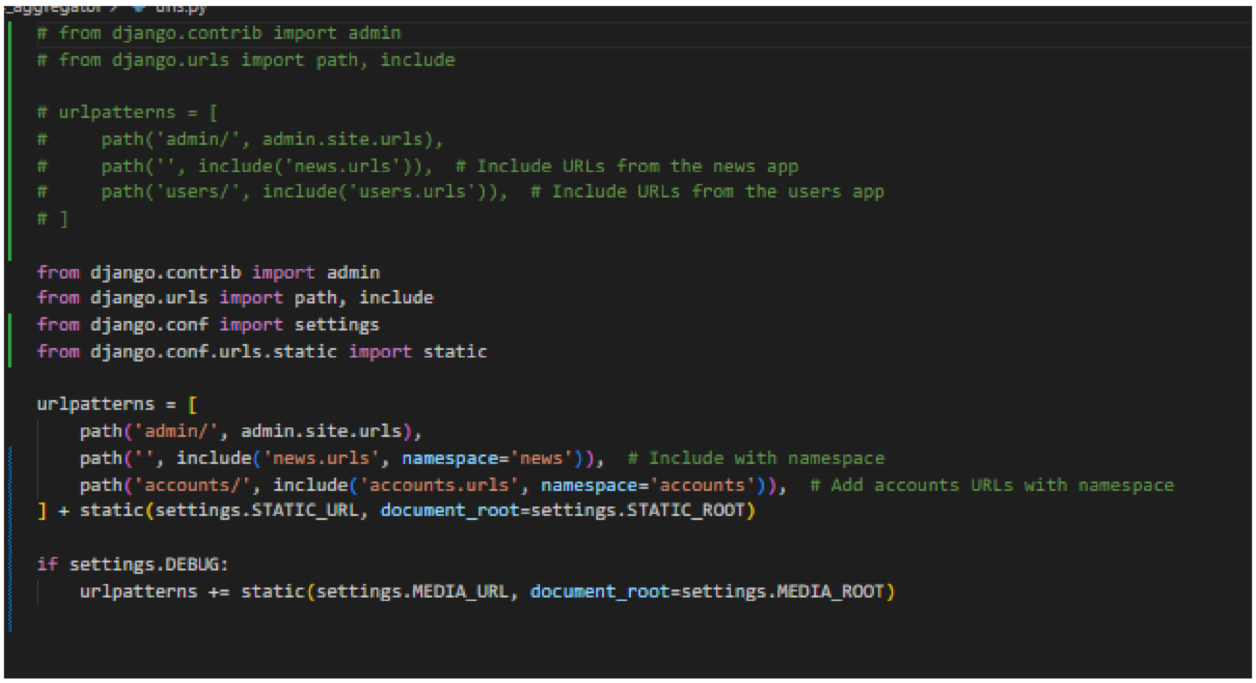
Code

The project folder personalized-news-aggregator is structured as a Django web application. The main directory news_aggregator contains project-level configuration files such as settings.py, urls.py, wsgi.py, and asgi.py. Inside the apps directory, there are two main Django apps: news and users. The news app includes modules for defining models, views, API serializers, and HTML templates related to news articles. The users app handles user authentication, including templates for login and signup pages. Shared templates and static files like CSS, JavaScript, and images are also included within the apps folder. The manage.py file is used for administrative commands. A separate tests folder contains unit tests for views, models, and APIs. Additional files like requirements.txt, .env, and Dockerfile support environment configuration and deployment. The project also includes standard documentation and version control files such as README.md, LICENSE, and .gitignore.



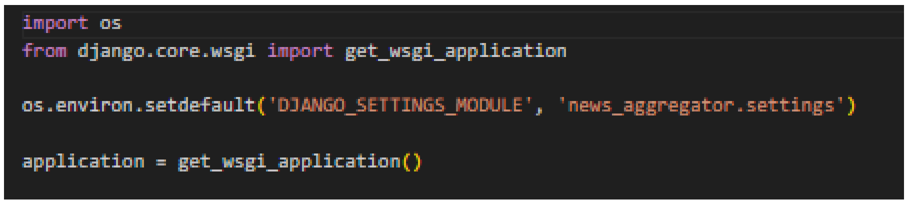
```python
import os
from pathlib import Path
from dotenv import load_dotenv

# Load environment variables from .env file
load_dotenv()

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.getenv('DJANGO_SECRET_KEY')

# API Keys
NEWS_API_KEY = "ad8a926fb0674481896b835dc3b363cc"  # Your News API key
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")  # Get OpenAI API key from environment variable

# Logging Configuration
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'verbose': {
            'format': '{levelname} {asctime} {module} {message}',
            'style': '{',
        },
    },
    'handlers': {
        'file': {
            'level': 'DEBUG',
            'class': 'logging.FileHandler',
            'filename': BASE_DIR / 'debug.log',
            'formatter': 'verbose',
        },
        'console': {
            'level': 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'verbose',
        },
    },
    'loggers': {
        '': {  # Root logger
            'handlers': ['console', 'file'],
            'level': 'DEBUG',
        },
        'news': {  # News app logger
            'handlers': ['console', 'file'],
            'level': 'DEBUG',
            'propagate': False,
        },
    },
}

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = os.getenv('DJANGO_DEBUG', 'True') == 'True'

ALLOWED_HOSTS = []

# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'news',  # Your news app
```

This code sets up environment variables, logging, and app configurations for the Django project. It loads API keys and the secret key using os.getenv, defines logging handlers for console and file output, and enables debug mode based on environment settings. It also registers built-in and custom apps like news and accounts in INSTALLED_APPS.



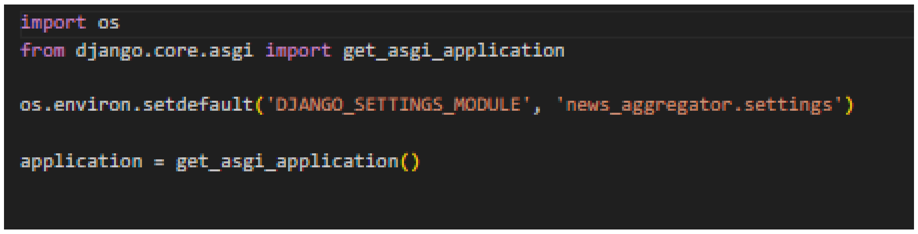
```python
# from django.contrib import admin
# from django.urls import path, include

# urlpatterns = [
#     path('admin/', admin.site.urls),
#     path('', include('news.urls')),  # Include URLs from the news app
#     path('users/', include('users.urls')),  # Include URLs from the users app
# ]

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('news.urls', namespace='news')),  # Include with namespace
    path('accounts/', include('accounts.urls', namespace='accounts')),  # Add accounts URLs with namespace
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

This code defines the main URL routing for the Django project. It maps the admin interface, includes URL patterns from the news and accounts apps using namespaces, and serves static and media files during development if DEBUG is enabled.



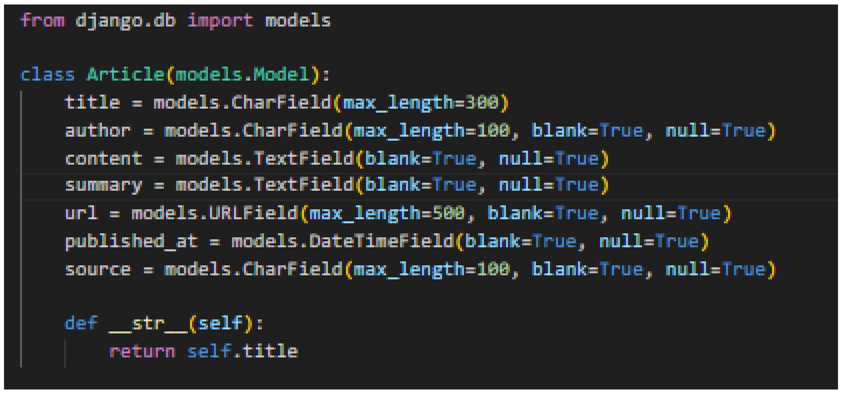
```python
import os
from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'news_aggregator.settings')

application = get_wsgi_application()
```

This code sets up the WSGI configuration for deploying the Django project. It defines the default settings module and exposes the WSGI application object used by servers like Gunicorn to run the application.



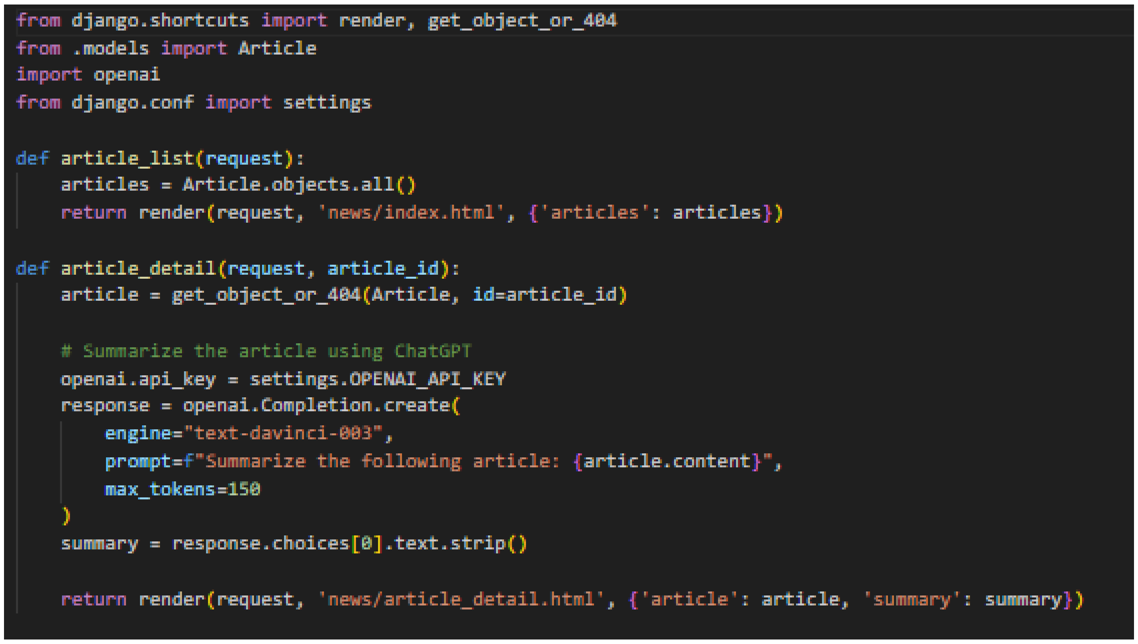
```python
import os
from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'news_aggregator.settings')

application = get_asgi_application()
```

This code sets up the ASGI configuration for the Django project. It specifies the settings module and initializes the ASGI application, which is used for handling asynchronous protocols like WebSockets.



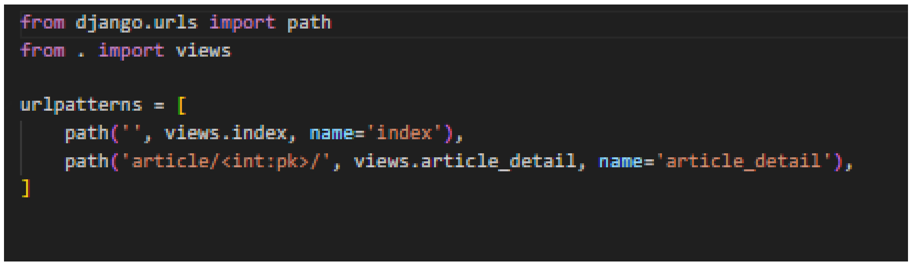
```python
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=300)
    author = models.CharField(max_length=100, blank=True, null=True)
    content = models.TextField(blank=True, null=True)
    summary = models.TextField(blank=True, null=True)
    url = models.URLField(max_length=500, blank=True, null=True)
    published_at = models.DateTimeField(blank=True, null=True)
    source = models.CharField(max_length=100, blank=True, null=True)

    def __str__(self):
        return self.title
```

This code defines an Article model in Django for storing news data. It includes fields for title, author, content, summary, URL, publication date, and source. The __str__ method returns the article title for readable representation in admin or shell.



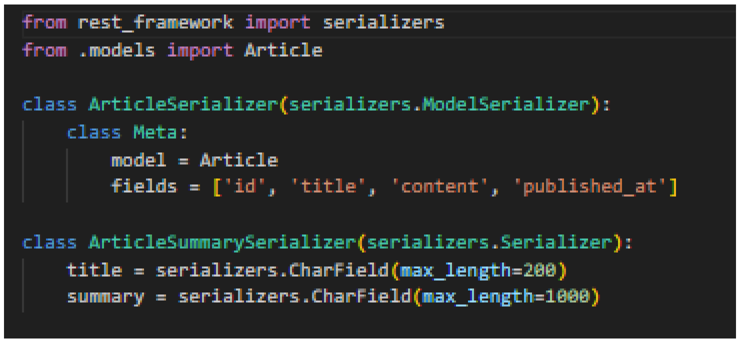
```python
from django.shortcuts import render, get_object_or_404
from .models import Article
import openai
from django.conf import settings

def article_list(request):
    articles = Article.objects.all()
    return render(request, 'news/index.html', {'articles': articles})

def article_detail(request, article_id):
    article = get_object_or_404(Article, id=article_id)

    # Summarize the article using ChatGPT
    openai.api_key = settings.OPENAI_API_KEY
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=f"Summarize the following article: {article.content}",
        max_tokens=150
    )
    summary = response.choices[0].text.strip()

    return render(request, 'news/article_detail.html', {'article': article, 'summary': summary})
```

This code defines two views: article_list, which fetches all articles and renders them to the index page, and article_detail, which retrieves a specific article by ID. In the detail view, the article's content is summarized using the OpenAI API and passed to the template for display.



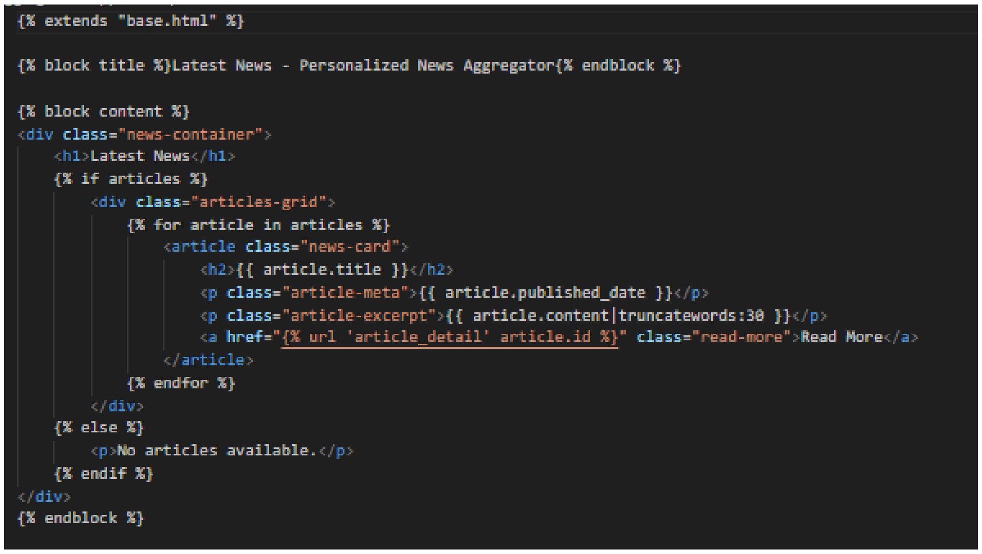
```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('article/<int:pk>/', views.article_detail, name='article_detail'),
]
```

The code defines URL routes for the news app. The root path renders the index view showing all articles, while the second path uses a dynamic article ID (<int:pk>) to route to the detail view of a specific article.



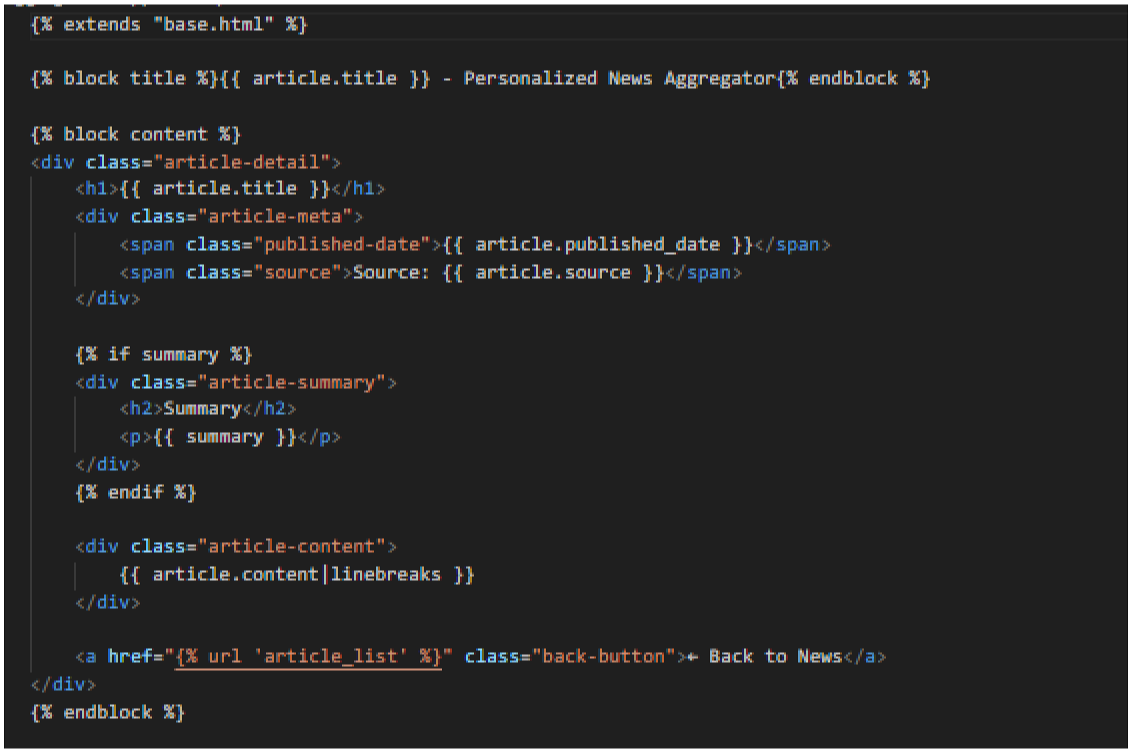
```python
from rest_framework import serializers
from .models import Article

class ArticleSerializer(serializers.ModelSerializer):
    class Meta:
        model = Article
        fields = ['id', 'title', 'content', 'published_at']

class ArticleSummarySerializer(serializers.Serializer):
    title = serializers.CharField(max_length=200)
    summary = serializers.CharField(max_length=1000)
```

The code defines two serializers. ArticleSerializer is a ModelSerializer for the Article model, exposing selected fields such as id, title, content, and published_at. ArticleSummarySerializer is a manual serializer for handling summarized article data with title and summary fields.



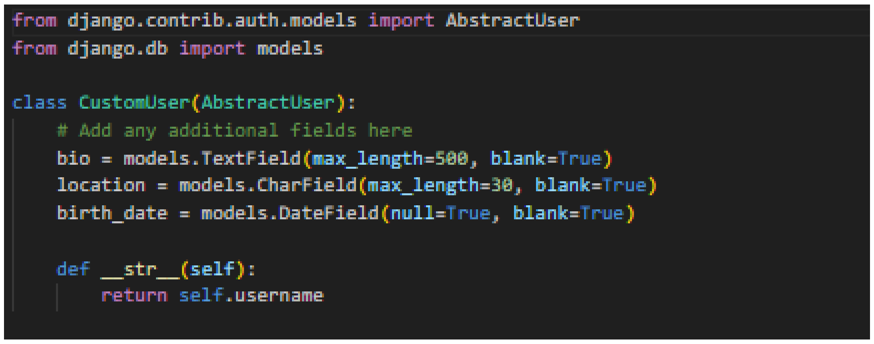
```html
{% extends "base.html" %}

{% block title %}Latest News - Personalized News Aggregator{% endblock %}

{% block content %}
<div class="news-container">
    <h1>Latest News</h1>
    {% if articles %}
        <div class="articles-grid">
            {% for article in articles %}
                <article class="news-card">
                    <h2>{{ article.title }}</h2>
                    <p class="article-meta">{{ article.published_date }}</p>
                    <p class="article-excerpt">{{ article.content|truncatewords:30 }}</p>
                    <a href="{% url 'article_detail' article.id %}" class="read-more">Read More</a>
                </article>
            {% endfor %}
        </div>
    {% else %}
        <p>No articles available.</p>
    {% endif %}
</div>
{% endblock %}
```

This code defines the HTML structure for the index page. It displays a list of articles with each article showing the title, publish date, a truncated content preview, and a link to the detail view. If no articles are available, it displays a fallback message.



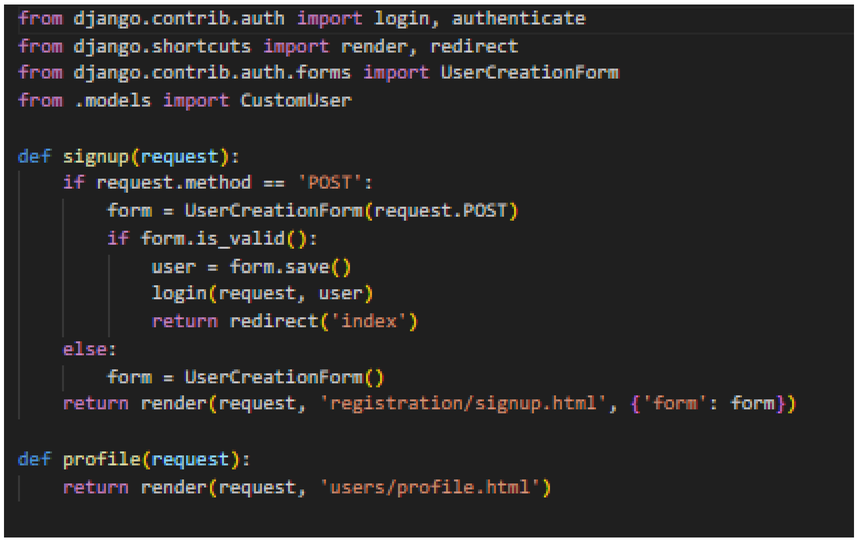
```
{% extends "base.html" %}

{% block title %}{{ article.title }} - Personalized News Aggregator{% endblock %}

{% block content %}
<div class="article-detail">
    <h1>{{ article.title }}</h1>
    <div class="article-meta">
        <span class="published-date">{{ article.published_date }}</span>
        <span class="source">Source: {{ article.source }}</span>
    </div>

    {% if summary %}
    <div class="article-summary">
        <h2>Summary</h2>
        <p>{{ summary }}</p>
    </div>
    {% endif %}

    <div class="article-content">
        {{ article.content|linebreaks }}
    </div>

    <a href="{% url 'article_list' %}" class="back-button">← Back to News</a>
</div>
{% endblock %}
```

This code defines the HTML for the article detail page. It shows the article title, publish date, source, and full content. If a summary is available, it displays the summary section above the main content. A back button is provided to return to the news list.



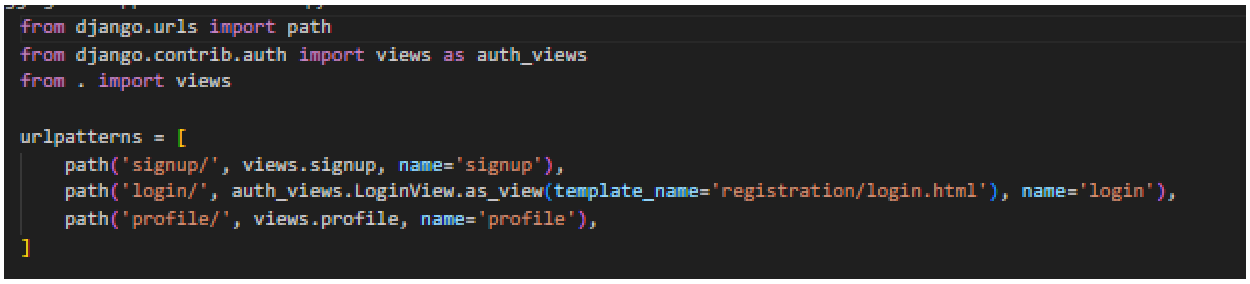
```
from django.contrib.auth.models import AbstractUser
from django.db import models

class CustomUser(AbstractUser):
    # Add any additional fields here
    bio = models.TextField(max_length=500, blank=True)
    location = models.CharField(max_length=30, blank=True)
    birth_date = models.DateField(null=True, blank=True)

    def __str__(self):
        return self.username
```

This code defines a custom user model CustomUser by extending Django's AbstractUser. It adds optional fields for bio, location, and birth date, allowing additional user profile information beyond the default fields.



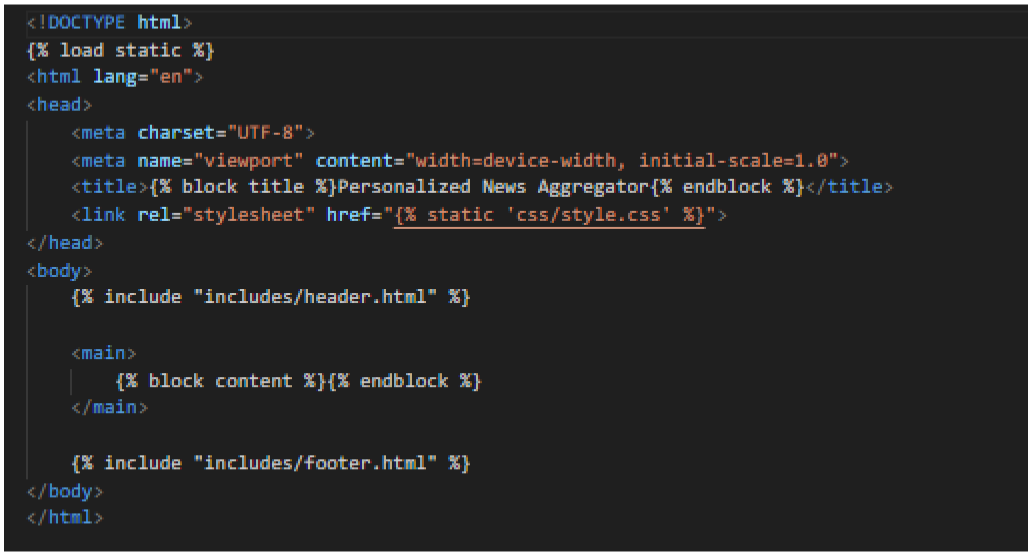
```python
from django.contrib.auth import login, authenticate
from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm
from .models import CustomUser

def signup(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('index')
    else:
        form = UserCreationForm()
    return render(request, 'registration/signup.html', {'form': form})

def profile(request):
    return render(request, 'users/profile.html')
```

This code defines two view functions. The signup view handles user registration using Django’s UserCreationForm, saves the user if the form is valid, logs them in, and redirects to the index. The profile view renders the profile page for logged-in users.



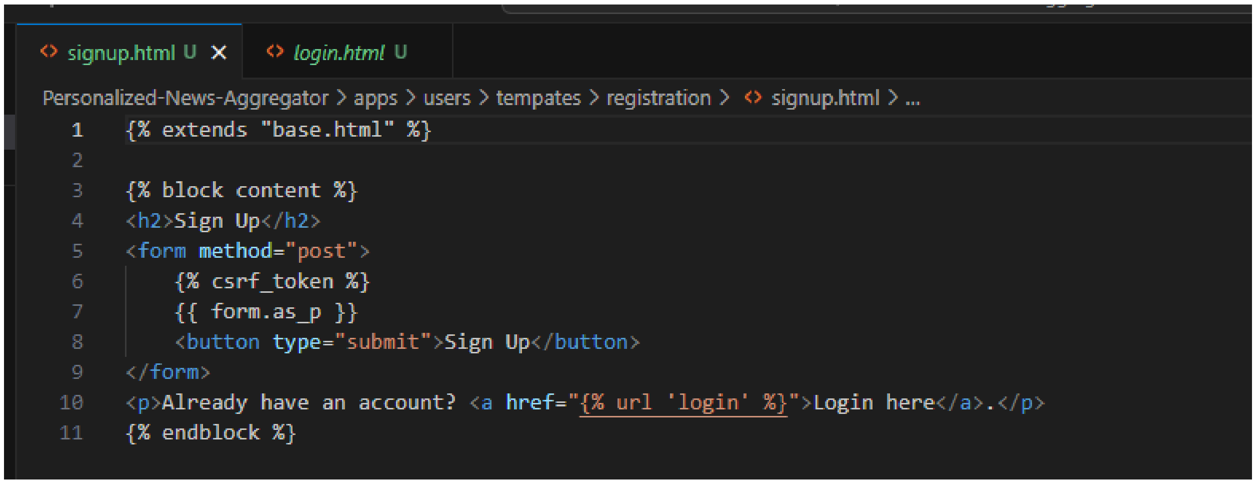
```python
from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

urlpatterns = [
    path('signup/', views.signup, name='signup'),
    path('login/', auth_views.LoginView.as_view(template_name='registration/login.html'), name='login'),
    path('profile/', views.profile, name='profile'),
]
```

This code defines URL routes for user-related views. It maps the signup view to /signup/, the login view to /login/ using Django's built-in LoginView, and the profile view to /profile/. Custom templates are specified for login and signup.



```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Personalized News Aggregator{% endblock %}</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    {% include "includes/header.html" %}

    <main>
        {% block content %}{% endblock %}
    </main>

    {% include "includes/footer.html" %}
</body>
</html>
```

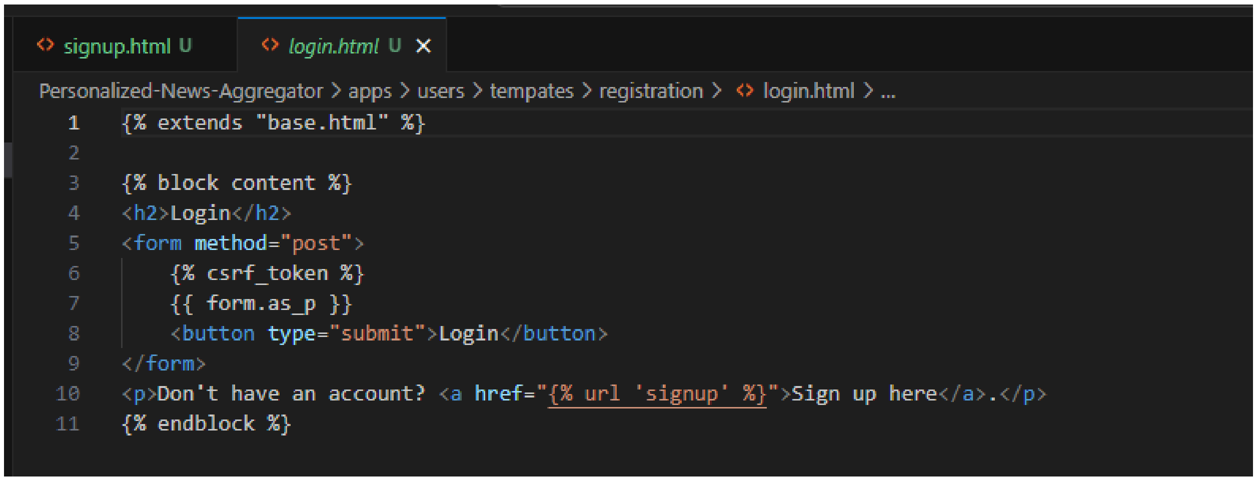
This code defines the base HTML template used by other pages. It includes static assets like CSS, sets the page title dynamically, and loads common header and footer templates. The main content area is filled by child templates using the {% block content %} tag.


signup.html U × login.html U

Personalized-News-Aggregator > apps > users > tempates > registration > signup.html > ...

```
{% extends "base.html" %}

{% block content %}
<h2>Sign Up</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Sign Up</button>
</form>
<p>Already have an account? <a href="{% url 'login' %}">Login here</a>.</p>
{% endblock %}
```

The user registration interface in your Django application includes this signup.html template. Because it extends base.html, it takes over the general layout (header, footer, and styling) specified in that foundational template. with order for the form to be included into the base.html main section, it is wrapped with {% block content %}. To make formatting and styling simple, it makes use of Django's built-in UserCreationForm, which is displayed with {{ form.as_p }} to output each field wrapped in as is common in Django forms, the form incorporates {% csrf_token %} to guard against Cross-Site Request Forgery attacks. If the form is legitimate (as handled in the views.py signup view), the user is created and logged in after submitting it, which posts back to the same URL. The {% url 'login' %} tag, a dynamic Django template tag that guarantees the URL is correctly resolved even if the path changes later in urls.py, is used to prompt users who already have an account to navigate to the login page via the link below the form.

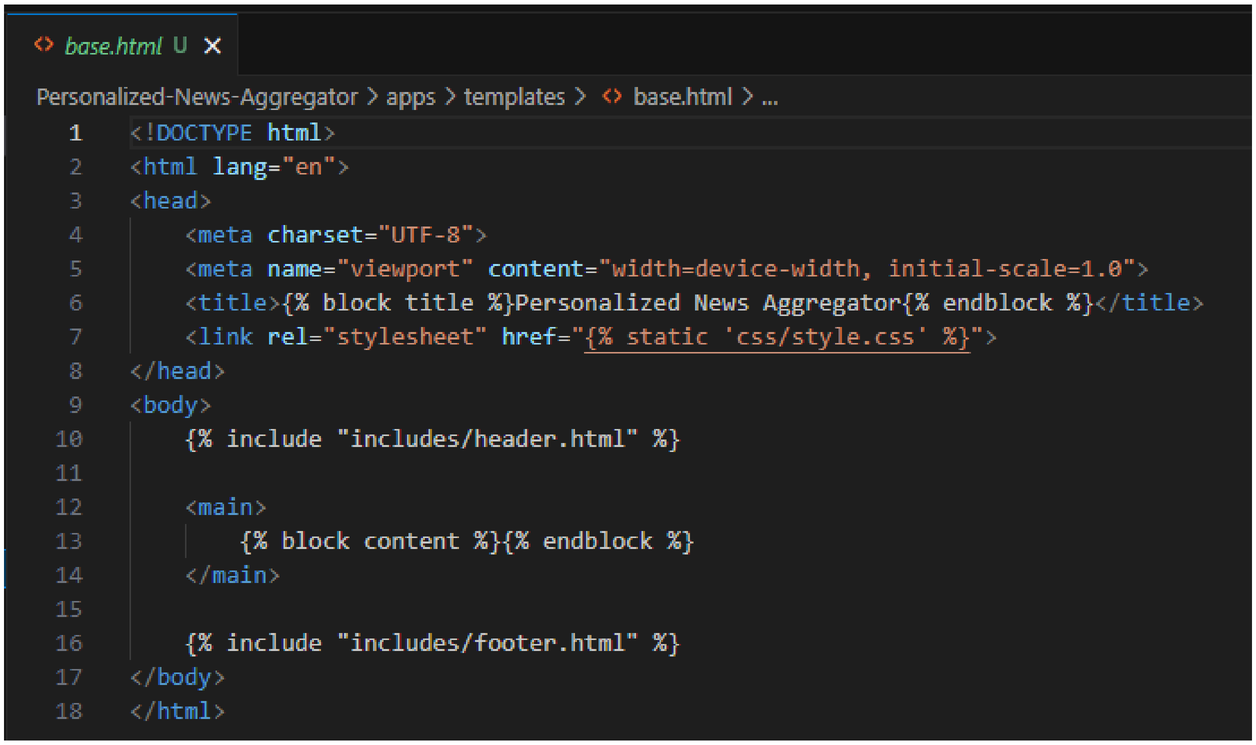


signup.html U
login.html U
Personalized-News-Aggregator > apps > users > tempates > registration > login.html > ...

```
{% extends "base.html" %}

{% block content %}
<h2>Login</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Login</button>
</form>
<p>Don't have an account? <a href="{% url 'signup' %}">Sign up here</a>.</p>
{% endblock %}
```

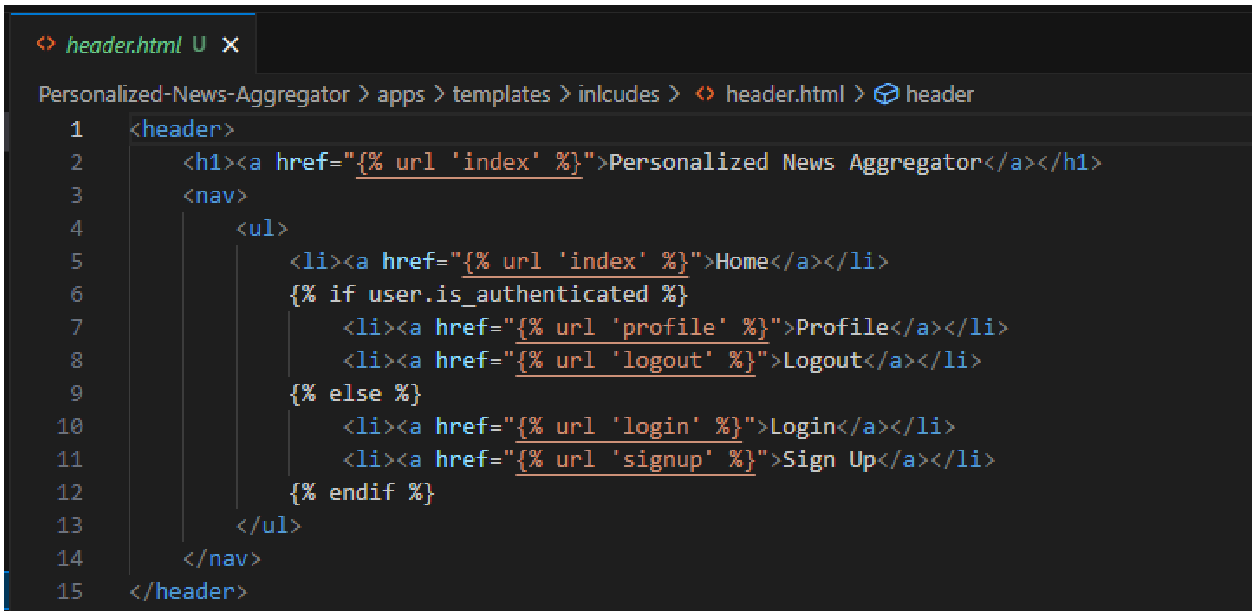
Code in login.html template is the frontend interface for user authentication in your Django app. Like signup.html, it extends the base.html layout to maintain visual consistency across pages. Within the {% block content %} tag, it displays a login form rendered using Django's built-in authentication system. The form is submitted via POST and includes {% csrf_token %} for security against CSRF attacks. The fields are rendered using {{ form.as_p }}, ensuring clean formatting with each form field wrapped in a paragraph element.

At the bottom of the form, there's a prompt for users who don't yet have an account, linking them to the signup page using Django's {% url 'signup' %} tag. This template is likely connected to Django's LoginView class-based view in urls.py, where the template_name parameter is set to this file. This clean and simple design ensures users can easily access and authenticate within your app while maintaining secure and maintainable backend logic.


base.html U
Personalized-News-Aggregator > apps > templates > base.html > ...

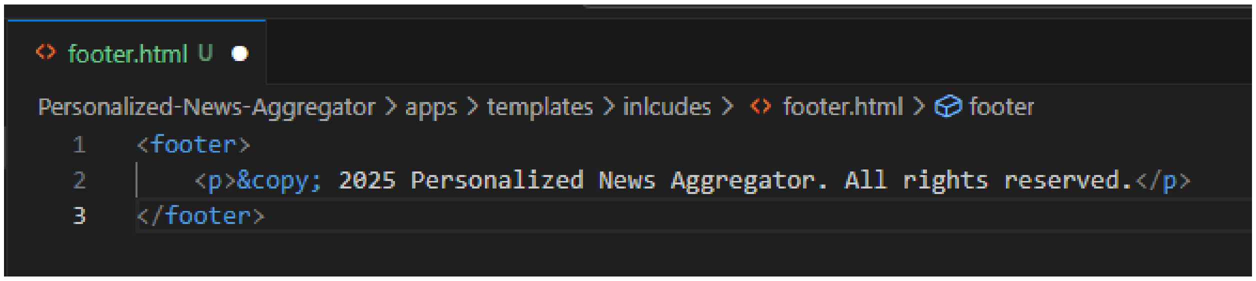
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Personalized News Aggregator{% endblock %}</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    {% include "includes/header.html" %}

    <main>
        {% block content %}{% endblock %}
    </main>

    {% include "includes/footer.html" %}
</body>
</html>
```

File base.html template serves as the foundational layout for your entire Django application. It defines the basic HTML structure — including the doctype, language, meta tags for responsiveness, a dynamic <title> block, and a reference to the main stylesheet via {% static 'css/style.css' %}. This setup ensures consistent styling and structure across all pages that extend from this base template. The {% block title %} and {% block content %} tags are placeholders for child templates (like login.html and signup.html) to insert custom content, allowing for flexible reuse of the layout. Inside the <body>, the template includes two key layout components: a header and footer, imported using {% include 'includes/header.html' %} and {% include 'includes/footer.html' %} respectively. Between these, the <main> tag holds {% block content %} where specific page content is rendered. This modular design promotes clean separation of layout and content, improving both maintainability and readability. It also ensures that updates to global elements (like the nav bar or footer) propagate across the entire site with a single change.


header.html U ×
Personalized-News-Aggregator > apps > templates > inlcudes > header.html > header

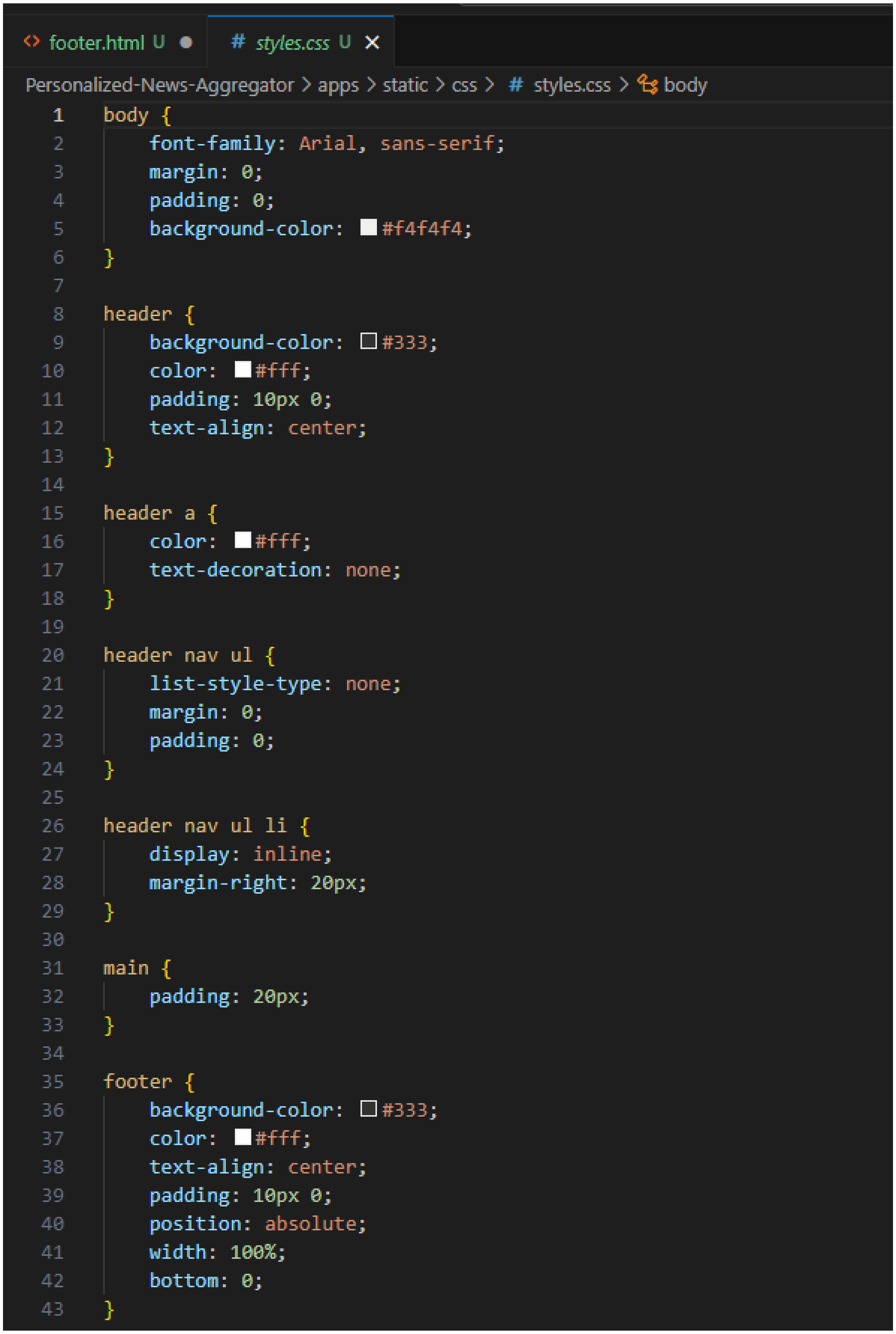
```
<header>
    <h1><a href="{% url 'index' %}">Personalized News Aggregator</a></h1>
    <nav>
        <ul>
            <li><a href="{% url 'index' %}">Home</a></li>
            {% if user.is_authenticated %}
                <li><a href="{% url 'profile' %}">Profile</a></li>
                <li><a href="{% url 'logout' %}">Logout</a></li>
            {% else %}
                <li><a href="{% url 'login' %}">Login</a></li>
                <li><a href="{% url 'signup' %}">Sign Up</a></li>
            {% endif %}
        </ul>
    </nav>
</header>
```

Code in the header.html template defines the navigation bar for the web application and dynamically adjusts its links based on the user’s login status. It starts with a site title that links to the homepage using Django’s {% url 'index' %} tag. Inside the <nav> section, it checks if the user is authenticated. If they are, it shows links to the Profile and Logout pages. If not, it displays Login and Sign Up options. This dynamic rendering provides a personalized user experience while keeping navigation clean and relevant. The template is modular and included in base.html, ensuring consistency across all pages of the site.


footer.html U
Personalized-News-Aggregator > apps > templates > inlcudes > footer.html > footer

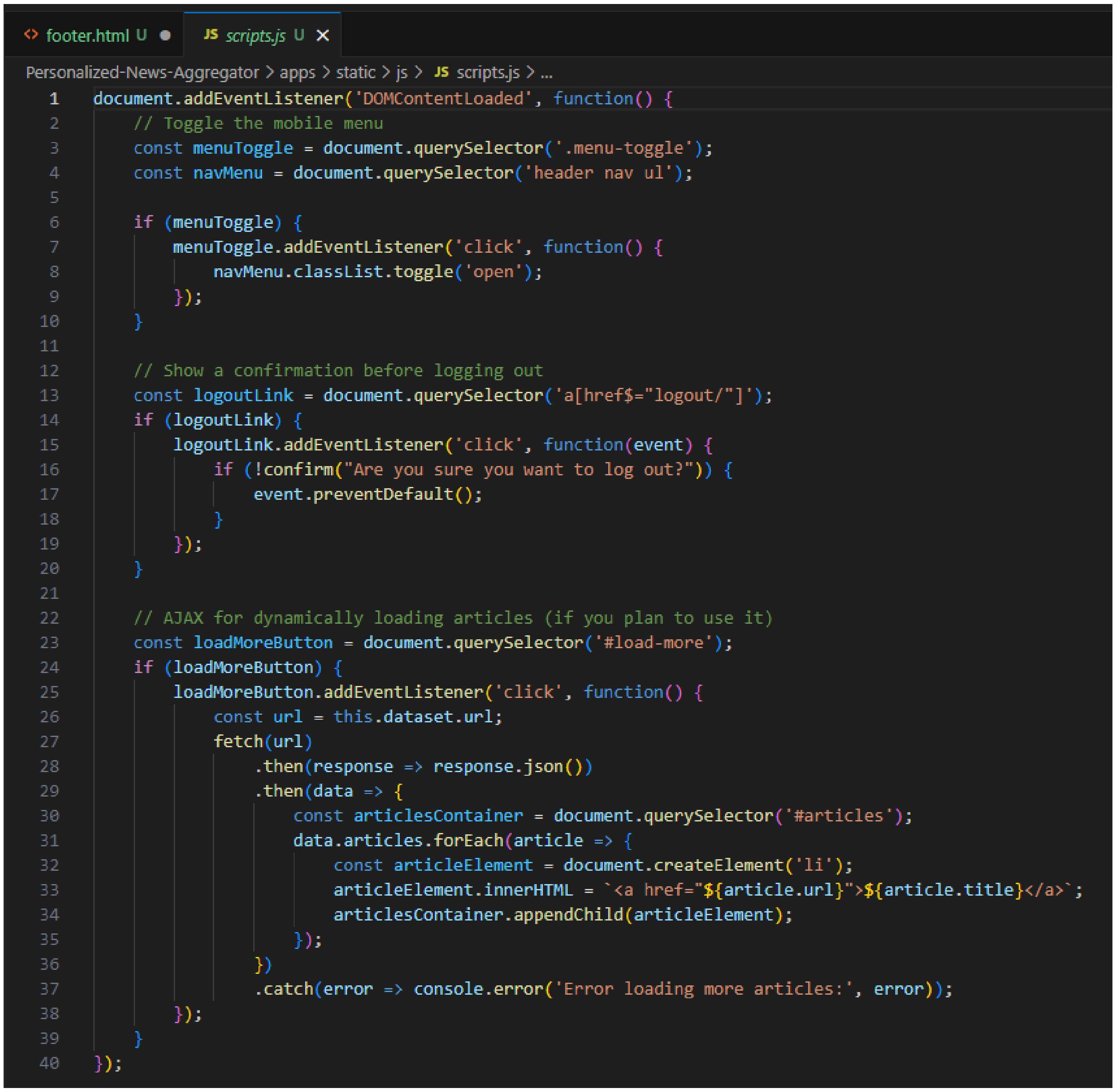
```html
<footer>
    <p>&copy; 2025 Personalized News Aggregator. All rights reserved.</p>
</footer>
```

File footer.html template defines the footer section of your web application. It contains a simple paragraph element that displays a copyright notice using the HTML entity followed by the year 2025 and the site name "Personalized News Aggregator." This component is included in the base layout (base.html) to ensure it appears consistently at the bottom of every page. It's a clean and effective way to display legal or branding information across your site.


footer.html U
styles.css U
Personalized-News-Aggregator > apps > static > css > styles.css > body

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

header {
    background-color: #333;
    color: #fff;
    padding: 10px 0;
    text-align: center;
}

header a {
    color: #fff;
    text-decoration: none;
}

header nav ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

header nav ul li {
    display: inline;
    margin-right: 20px;
}

main {
    padding: 20px;
}

footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px 0;
    position: absolute;
    width: 100%;
    bottom: 0;
}
```

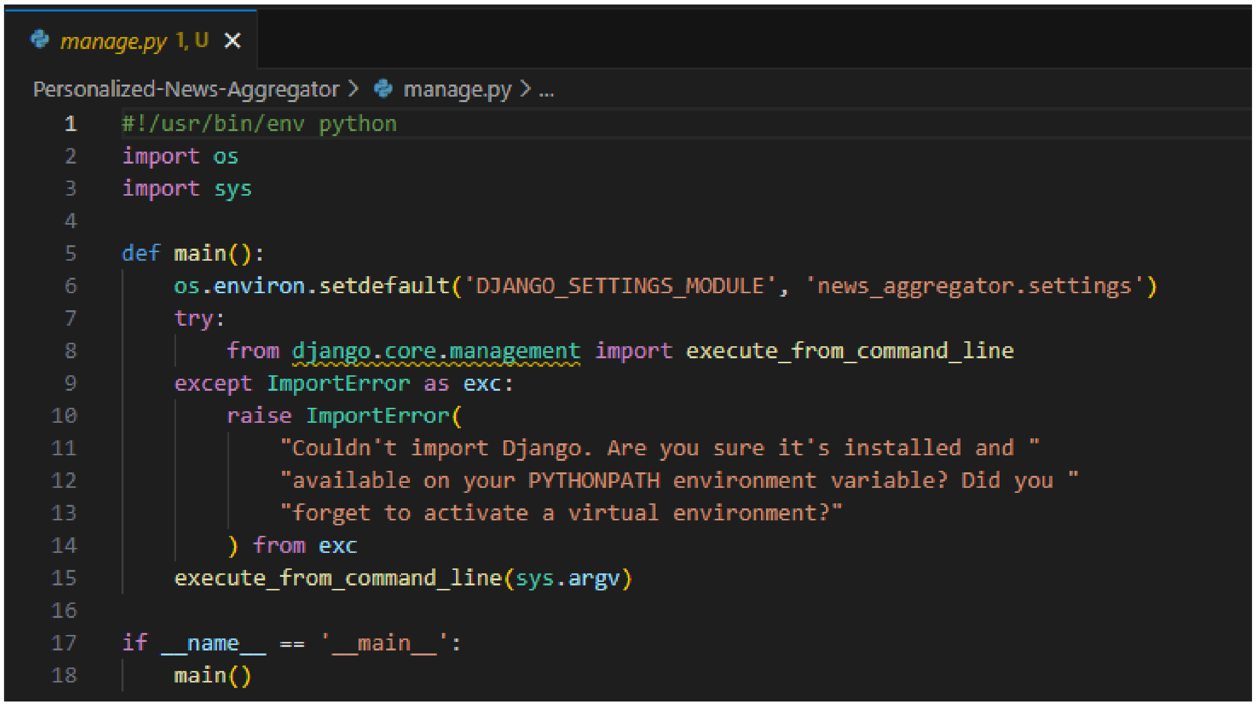
styles.css file defines the basic layout and styling of your app, giving it a clean and readable appearance. It sets a consistent font (Arial, sans-serif), removes default margins and padding, and applies a light background color to the body. The header and footer are styled with a dark background (#333) and white text for contrast, with centered text and padding for spacing. Navigation links are displayed inline with spacing between them, and links in the header are styled to remove underlines and maintain white text. The footer is fixed at the bottom using position: absolute, with width: 100% and bottom: 0, ensuring it always spans the full width and stays at the bottom of the viewport. This layout provides a professional and structured design for the overall site.


footer.html U
scripts.js U
Personalized-News-Aggregator > apps > static > js > scripts.js > ...

```javascript
document.addEventListener('DOMContentLoaded', function() {
    // Toggle the mobile menu
    const menuToggle = document.querySelector('.menu-toggle');
    const navMenu = document.querySelector('header nav ul');

    if (menuToggle) {
        menuToggle.addEventListener('click', function() {
            navMenu.classList.toggle('open');
        });
    }

    // Show a confirmation before logging out
    const logoutLink = document.querySelector('a[href$="logout/"]');
    if (logoutLink) {
        logoutLink.addEventListener('click', function(event) {
            if (!confirm("Are you sure you want to log out?")) {
                event.preventDefault();
            }
        });
    }

    // AJAX for dynamically loading articles (if you plan to use it)
    const loadMoreButton = document.querySelector('#load-more');
    if (loadMoreButton) {
        loadMoreButton.addEventListener('click', function() {
            const url = this.dataset.url;
            fetch(url)
                .then(response => response.json())
                .then(data => {
                    const articlesContainer = document.querySelector('#articles');
                    data.articles.forEach(article => {
                        const articleElement = document.createElement('li');
                        articleElement.innerHTML = `<a href="${article.url}">${article.title}</a>`;
                        articlesContainer.appendChild(articleElement);
                    });
                })
                .catch(error => console.error('Error loading more articles:', error));
        });
    }
});
```

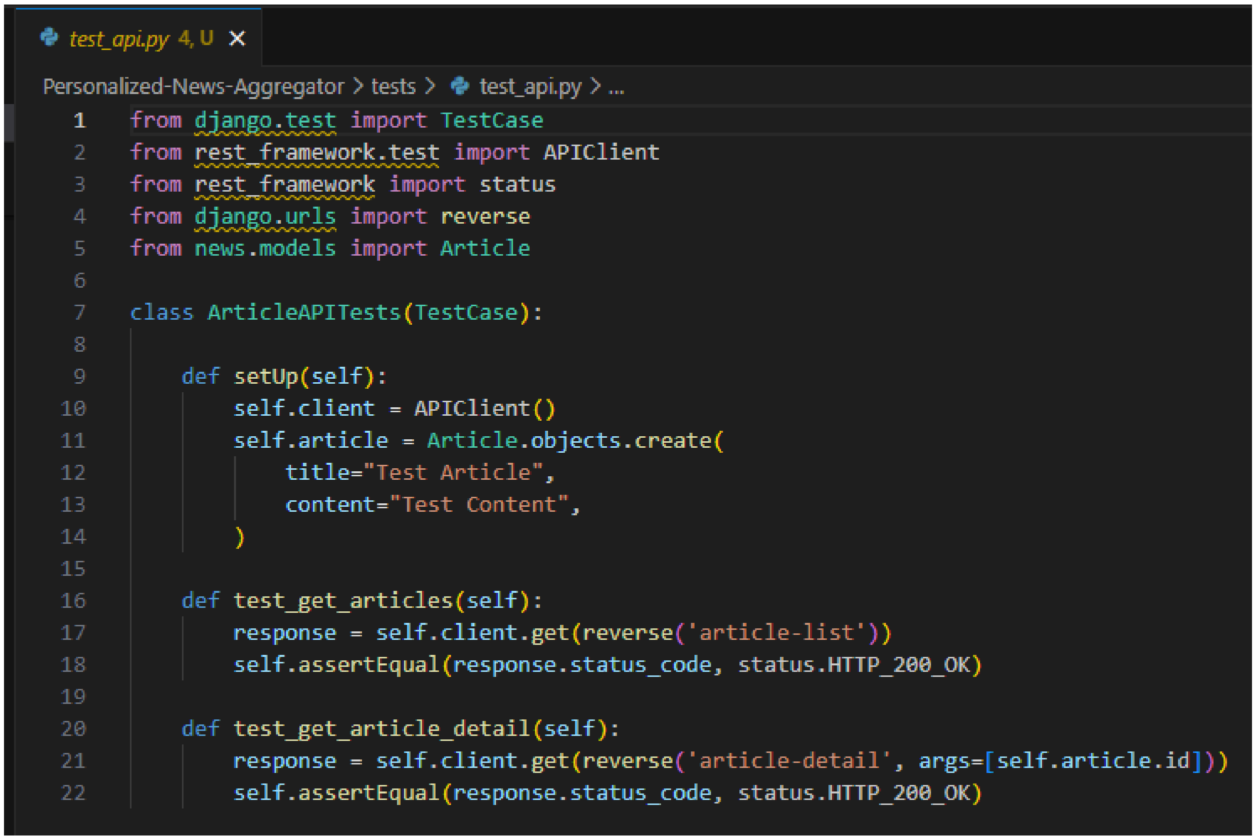
scripts.js file enhances user experience by adding interactivity to your web app. When the DOM is fully loaded, it attaches event listeners to handle three features. First, it toggles the visibility of a mobile navigation menu when the user clicks a menu icon (.menu-toggle). Second, it adds a confirmation prompt before logging out to prevent accidental sign-outs, targeting any logout link with a[href$="logout/"].

Lastly, the script supports AJAX-based dynamic article loading. If a “Load More” button exists, clicking it fetches additional articles from a specified URL (stored in a data-url attribute), parses the response as JSON, and appends new article links to the DOM. This allows seamless content loading without reloading the entire page.



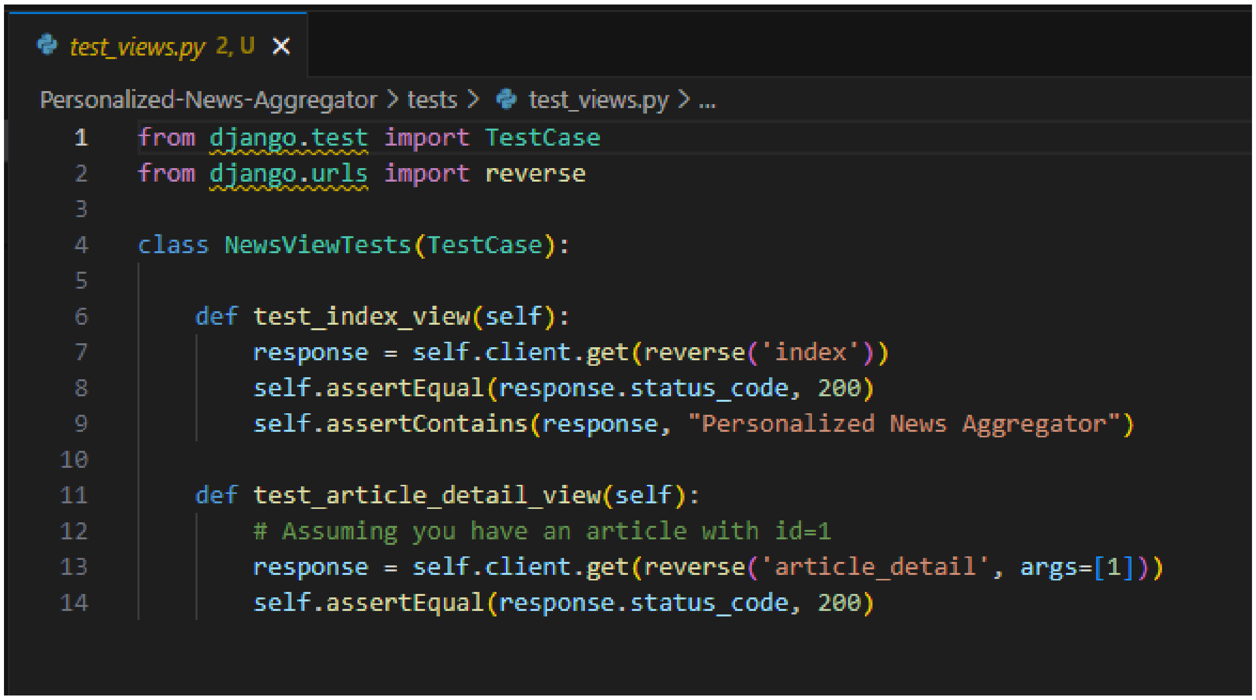
```python
#!/usr/bin/env python
import os
import sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'news_aggregator.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

Code in manage.py file is the command-line utility script that serves as the entry point for your Django project. When executed, it sets the default Django settings module (news_aggregator.settings) and uses execute_from_command_line to run commands like runserver, makemigrations, and createsuperuser. This script is crucial for managing and interacting with your Django project during development and deployment. It includes a safeguard using a try-except block to catch and raise a detailed error if Django isn’t installed or configured properly. The condition if __name__ == "__main__": ensures that the script only runs when executed directly, not when imported. Overall, this file is essential for the operational control of your Django project.



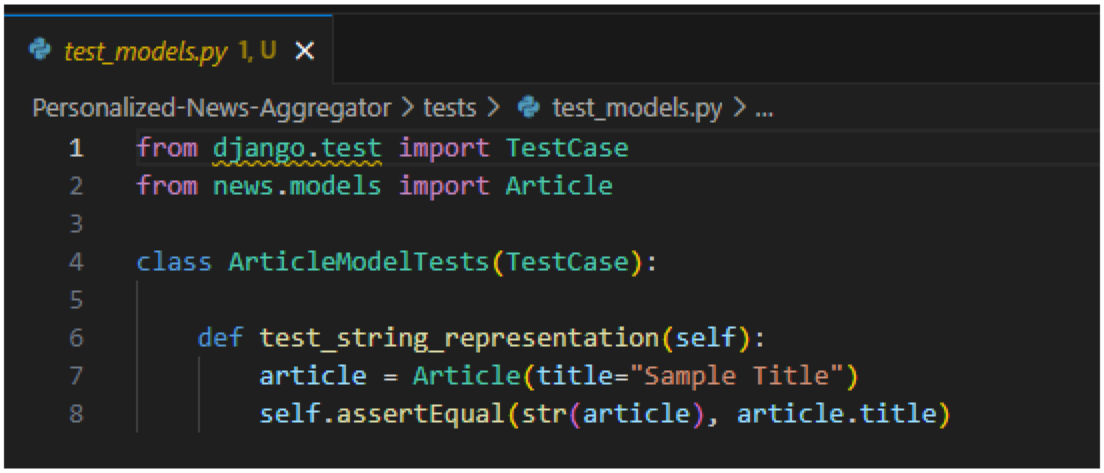
```python
from django.test import TestCase
from rest_framework.test import APIClient
from rest_framework import status
from django.urls import reverse
from news.models import Article

class ArticleAPITests(TestCase):

    def setUp(self):
        self.client = APIClient()
        self.article = Article.objects.create(
            title="Test Article",
            content="Test Content",
        )

    def test_get_articles(self):
        response = self.client.get(reverse('article-list'))
        self.assertEqual(response.status_code, status.HTTP_200_OK)

    def test_get_article_detail(self):
        response = self.client.get(reverse('article-detail', args=[self.article.id]))
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

test_api.py file contains unit tests for the Django REST API endpoints related to the Article model. It uses Django's TestCase and Django REST framework's APIClient to simulate API calls. In the setUp method, it creates a test article and initializes the API client. This prepares a consistent test environment for each test method. The test methods test_get_articles and test_get_article_detail validate that the API correctly returns responses for the article list and individual article detail endpoints, respectively. They use Django's reverse to resolve URL names and ensure the API returns a 200 OK status. These tests help ensure the reliability and correctness of the backend API during development and future updates.



```python
from django.test import TestCase
from django.urls import reverse

class NewsViewTests(TestCase):

    def test_index_view(self):
        response = self.client.get(reverse('index'))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "Personalized News Aggregator")

    def test_article_detail_view(self):
        # Assuming you have an article with id=1
        response = self.client.get(reverse('article_detail', args=[1]))
        self.assertEqual(response.status_code, 200)
```

File test_views.py contains unit tests for verifying the behavior of your Django app's views. It uses Django's TestCase class and the reverse function to dynamically resolve URL patterns. The test_index_view function sends a GET request to the homepage (index view) and checks that the response returns a 200 OK status and contains the text "Personalized News Aggregator", ensuring the main page loads correctly. The test_article_detail_view function attempts to test the detail view for a specific article by requesting article_detail with an ID of 1. However, it assumes an article with ID 1 already exists, which may not always be the case unless test data is set up. To make this test more reliable, it's best to create a test article in a setUp method.


test_models.py 1, U ×

Personalized-News-Aggregator > tests > test_models.py > ...

```python
from django.test import TestCase
from news.models import Article

class ArticleModelTests(TestCase):

    def test_string_representation(self):
        article = Article(title="Sample Title")
        self.assertEqual(str(article), article.title)
```

Code in test_models.py file is a unit test for verifying the __str__ method of the Article model. It uses Django's TestCase framework to define a test class named ArticleModelTests. Inside this class, the test_string_representation method creates an Article instance with a sample title and checks if calling str(article) returns the same value as article.title. This test ensures that the string representation of an Article object is intuitive and user-friendly, which is especially useful for admin interfaces, logs, and debugging. By confirming this behavior, the test maintains consistency in how articles are displayed when converted to a string.

Design document and flowchart

This system is a personalized news aggregator built using Django. It fetches news articles from external APIs, summarizes them using natural language processing, analyzes their sentiment, and recommends articles to users based on their reading behavior and preferences. The goal is to create a more relevant and personalized news reading experience through automation and simple interaction design.

System Architecture

The backend is built using Django, and it handles user authentication, data storage, article processing, and personalized recommendation logic. The frontend is rendered using Django templates with HTML and CSS. The system also integrates NLP techniques using local models and machine learning libraries like scikit-learn to generate summaries, analyze sentiment, and compute content similarity.

There are several core modules that drive the application. The Article model stores fetched news articles, summaries, sentiments, and metadata. Users are managed using Django's built-in user model, and each user's preferences are recorded through a UserPreference model and a UserClickLog, which logs every article they read. This data is later used for recommendation.

The views handle routing and logic. The article list view shows all available news articles with optional recommendations. When a user clicks on an article, the detail view shows the article along with a summary and related articles based on similarity. The user account views allow registration, login, and managing preferences.

Utility scripts run background tasks like fetching news from APIs, summarizing using a local Hugging Face model, performing sentiment analysis with VADER, and computing TF-IDF-based similarity scores for recommendations.

Templates render the user interface. The index.html page shows article listings, and article_detail.html displays the full article with related recommendations. A common base.html layout is used across all pages.

Data Flow

When a user visits the site, they see a list of news articles. If they are logged in, the list is optionally filtered or sorted based on their preferences. When the user registers or updates preferences, the system stores that data to guide future recommendations.

If the user clicks on an article, the event is logged and the full content is shown along with a machine-generated summary and a list of related articles. These related articles are identified using TF-IDF vectors and cosine similarity, comparing the clicked article to others in the database.

Periodically, a background job runs the news fetcher. It collects new articles from NewsAPI, applies NLP-based summarization and sentiment analysis, and saves them to the database. Each new article is also processed for inclusion in the similarity and recommendation engine.

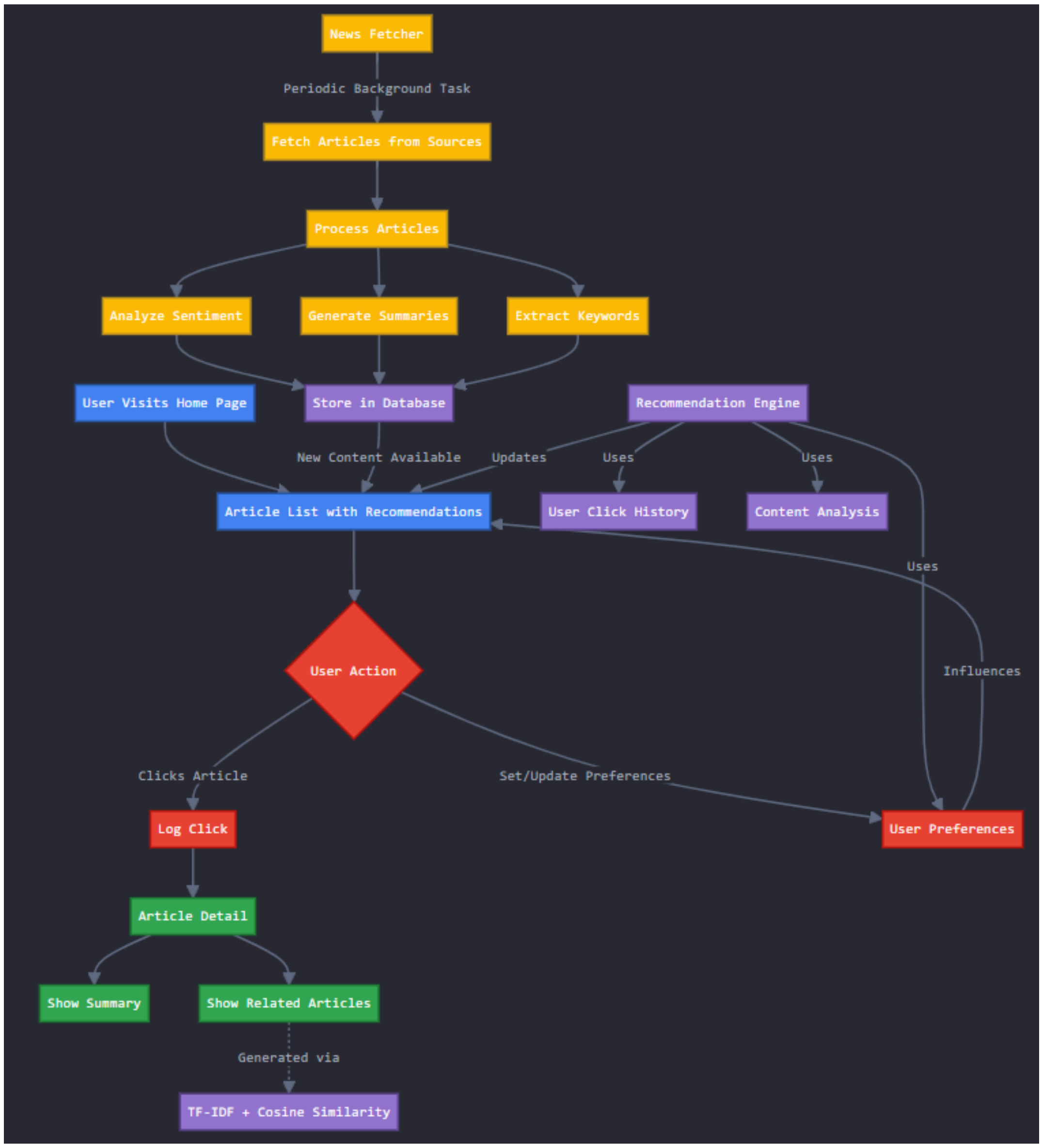
Personalization and Recommendation Logic

The system personalizes article suggestions using both category and content-based preferences. Category preferences are inferred from the user's click history, giving more weight to recent interactions. Content preferences are extracted using TF-IDF from the articles the user has clicked on. Recommendations are generated by matching these preferences with available articles. Additionally, each article page displays 3 to 5 most similar articles using cosine similarity scores derived from TF-IDF vectors.

Technologies Used

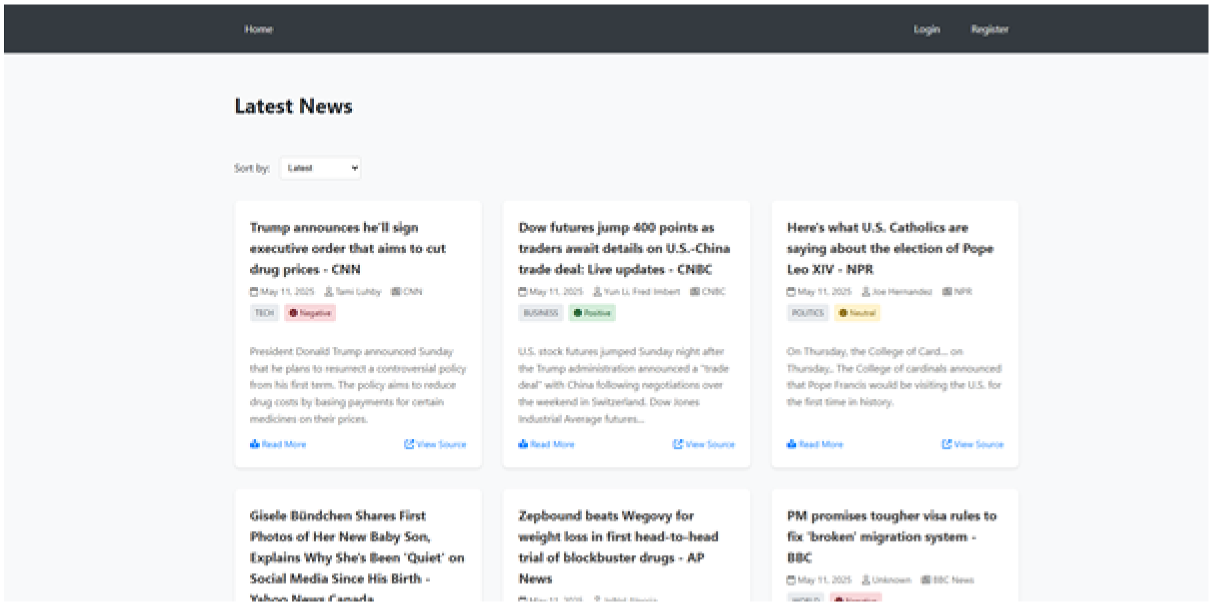
The backend is implemented with Django and SQLite. NLP is handled locally using a pre-trained BART model from Hugging Face. Scikit-learn is used to compute TF-IDF vectors and cosine similarity for content recommendations. VADER is used to determine the sentiment of articles. The front end is built using Django templates with HTML and CSS.

System Flowchart



VII. Data Analysis and Discussion

Output Generation



Latest News

Sort by: Latest

Trump announces he'll sign executive order that aims to cut drug prices - CNN
May 11, 2025 Tami Luhby CNN
TECH ● Negative
President Donald Trump announced Sunday that he plans to resurrect a controversial policy from his first term. The policy aims to reduce drug costs by basing payments for certain medicines on their prices.
Read More View Source

Dow futures jump 400 points as traders await details on U.S.-China trade deal: Live updates - CNBC
May 11, 2025 Yun Li, Fred Imbert CNBC
BUSINESS ● Positive
U.S. stock futures jumped Sunday night after the Trump administration announced a "trade deal" with China following negotiations over the weekend in Switzerland. Dow Jones Industrial Average futures...
Read More View Source

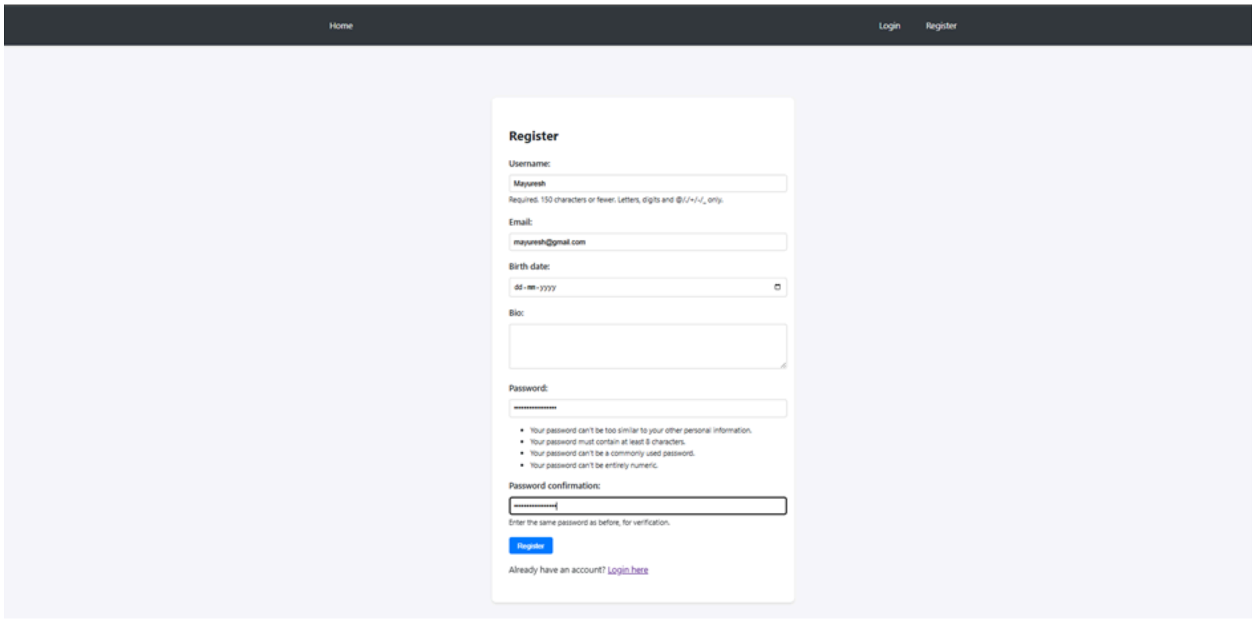
Here's what U.S. Catholics are saying about the election of Pope Leo XIV - NPR
May 11, 2025 Joe Hernandez NPR
POLITICS ● Neutral
On Thursday, the College of Card... on Thursday,. The College of cardinals announced that Pope Francis would be visiting the U.S. for the first time in history.
Read More View Source

Gisele Bündchen Shares First Photos of Her New Baby Son, Explains Why She's Been 'Quiet' on Social Media Since His Birth - Yahoo News Canada

Zepbound beats Wegovy for weight loss in first head-to-head trial of blockbuster drugs - AP News

PM promises tougher visa rules to fix 'broken' migration system - BBC
May 11, 2025 Unknown BBC News

It presents a series of curated news articles sorted by the latest date, each card showing the article's title, publication date, author, source, sentiment label (e.g., Positive, Negative, Neutral), and category tags like TECH or POLITICS. Users can read more details or view the original source, providing a clean, informative, and user-friendly news browsing experience.



Register

Username:

Mayuresh

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email:

mayuresh@gmail.com

Birth date:

dd-mm-yyyy

Bio:

Password:

•••••••••••••••

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

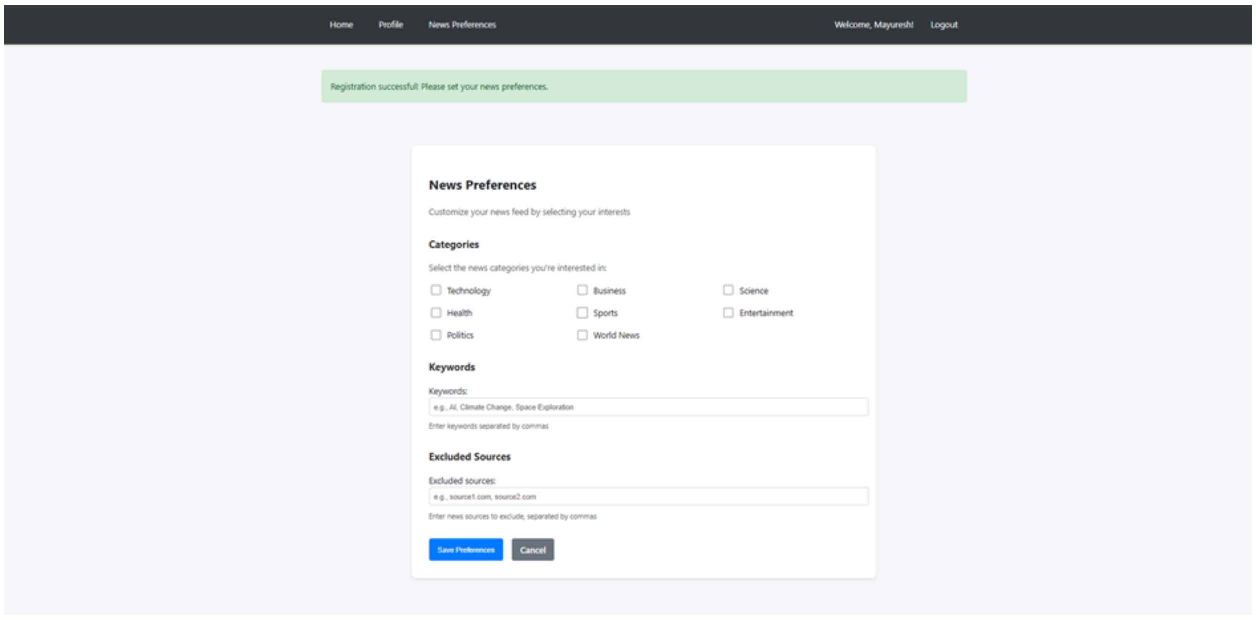
Password confirmation:

•••••••••••••••

Enter the same password as before, for verification.

Already have an account? Login here

It allows new users to create an account by entering details like username, email, birth date, bio, and password. The form includes validation checks for secure password creation and provides a link for users who already have an account to log in.



Registration successful! Please set your news preferences.

News Preferences

Customize your news feed by selecting your interests

Categories

Select the news categories you're interested in:

☐ Technology	☐ Business	☐ Science
☐ Health	☐ Sports	☐ Entertainment
☐ Politics	☐ World News	

Keywords

Keywords:

e.g., AI, Climate Change, Space Exploration

Enter keywords separated by commas

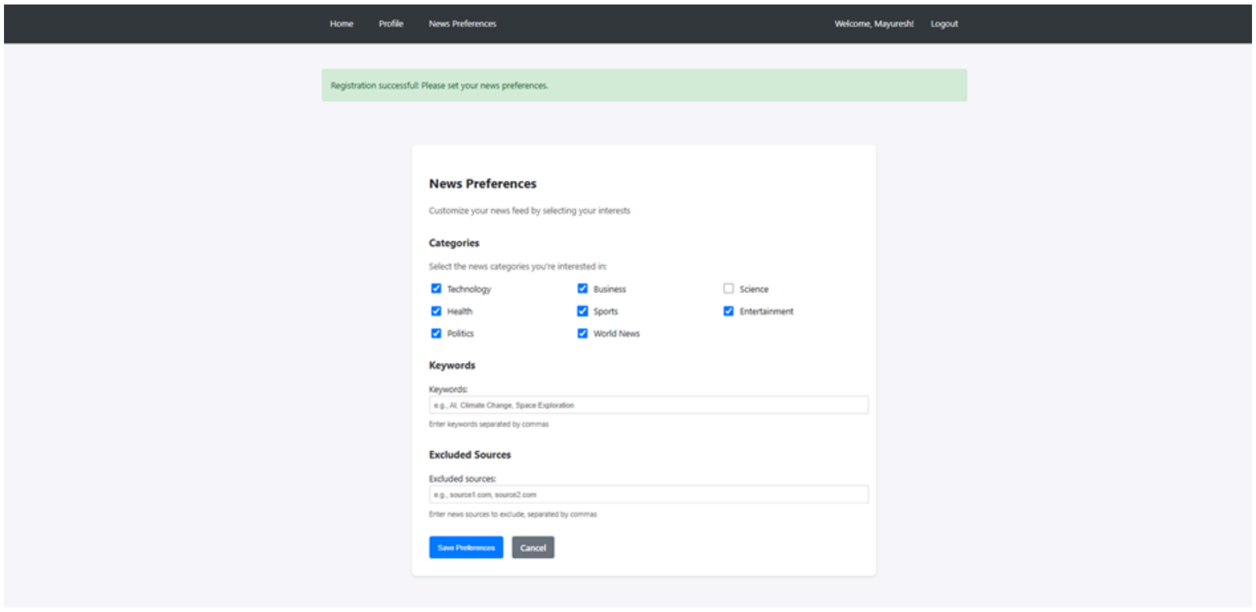
Excluded Sources

Excluded sources:

e.g., source1.com, source2.com

Enter news sources to exclude, separated by commas

Users can personalize their news feed by selecting preferred categories (like Technology, Business, or Politics), entering specific keywords of interest, and excluding sources they wish to avoid. This customization ensures a tailored news experience based on individual interests and preferences.



Registration successful! Please set your news preferences.

News Preferences

Customize your news feed by selecting your interests

Categories

Select the news categories you're interested in:

[✓] Technology	[✓] Business	[ ] Science
[✓] Health	[✓] Sports	[✓] Entertainment
[✓] Politics	[✓] World News	

Keywords

Keywords: e.g., AI, Climate Change, Space Exploration

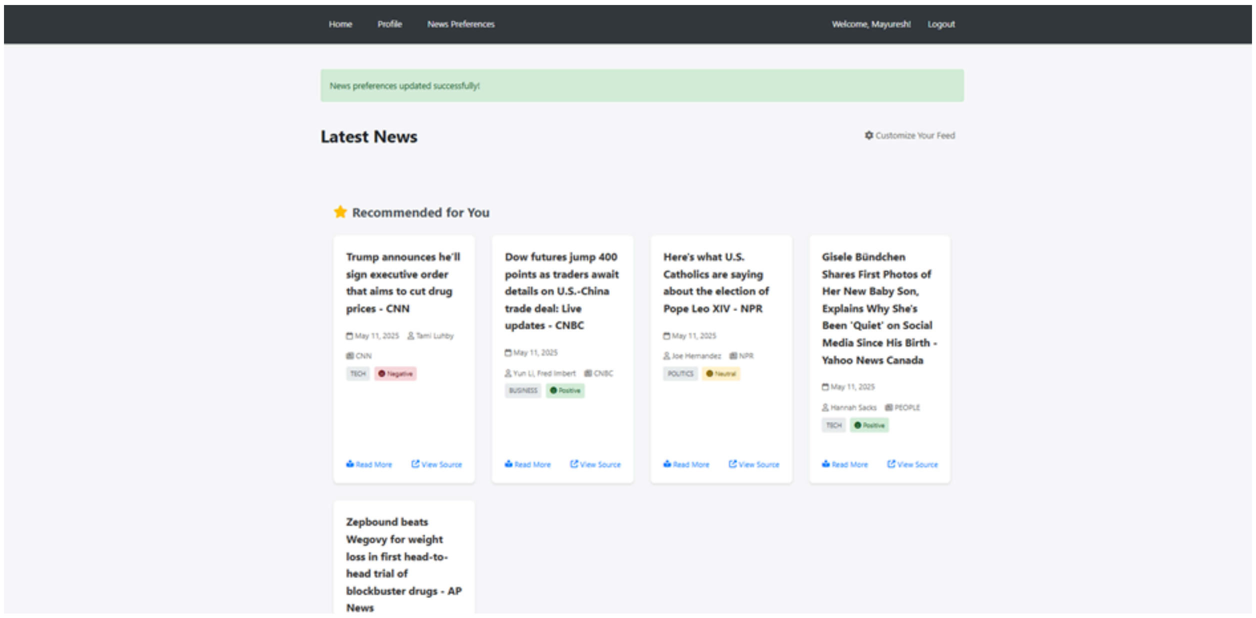
Enter keywords separated by commas

Excluded Sources

Excluded sources: e.g., source1.com, source2.com

Enter news sources to exclude, separated by commas

It shows the filled-in "News Preferences" form where the user has selected multiple categories such as Technology, Business, Health, Sports, Politics, Entertainment, and World News. It demonstrates how users can actively tailor their news experience by specifying topics of interest while optionally filtering content using keywords and excluded sources for a more refined feed.



News preferences updated successfully!

Latest News

Recommended for You

Trump announces he'll sign executive order that aims to cut drug prices - CNN
May 11, 2025 · Tami Luhby · CNN
TECH · Negative
Read More · View Source

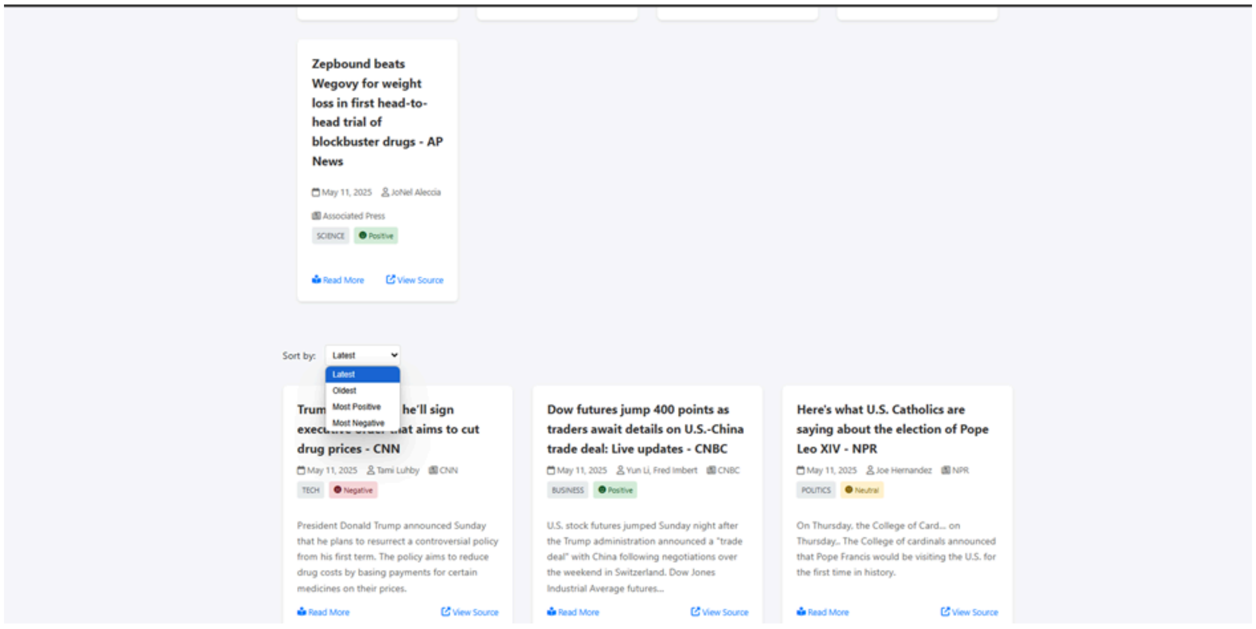
Dow futures jump 400 points as traders await details on U.S.-China trade deal: Live updates - CNBC
May 11, 2025 · Yun Li, Fred Imbert · CNBC
BUSINESS · Positive
Read More · View Source

Here's what U.S. Catholics are saying about the election of Pope Leo XIV - NPR
May 11, 2025 · Joe Hernandez · NPR
POLITICS · Neutral
Read More · View Source

Gisele Bündchen Shares First Photos of Her New Baby Son, Explains Why She's Been 'Quiet' on Social Media Since His Birth - Yahoo News Canada
May 11, 2025 · Hannah Sacks · PEOPLE
TECH · Positive
Read More · View Source

Zepbound beats Wegovy for weight loss in first head-to-head trial of blockbuster drugs - AP News

It presents the personalized "Recommended for You" section, generated after the user updates their news preferences. The news feed now reflects articles aligned with the user's selected categories and interests, offering tailored content with sentiment tags and source information. The interface also provides options to further customize the feed or access full articles.



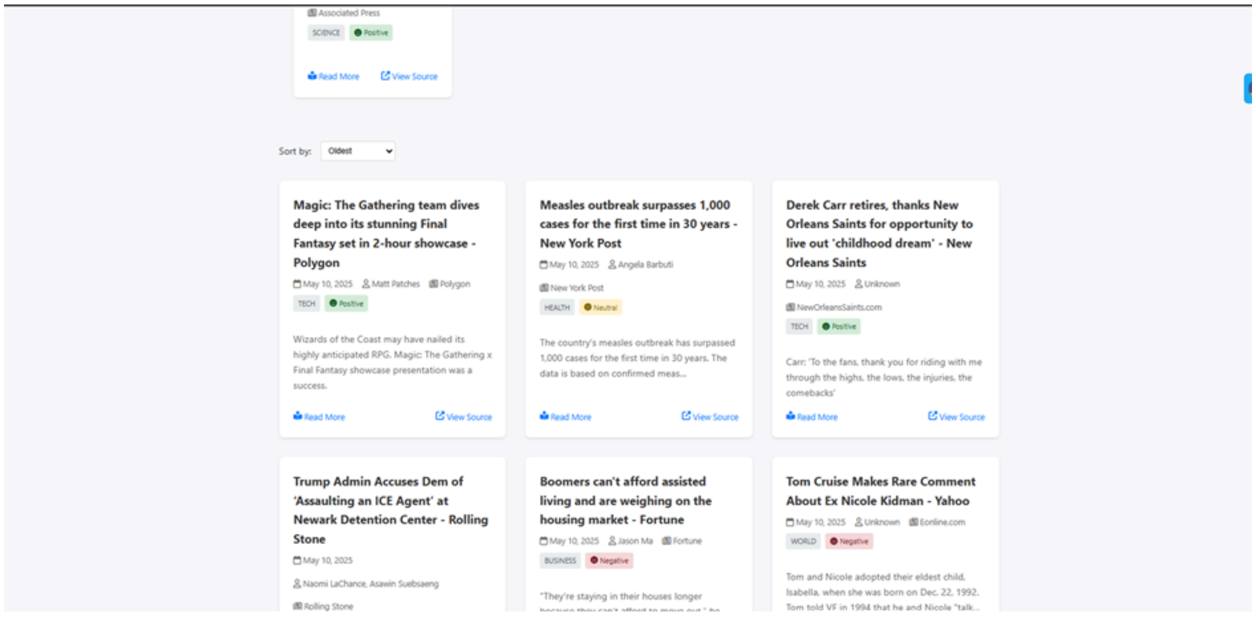
Zepbound beats Wegovy for weight loss in first head-to-head trial of blockbuster drugs - AP News
May 11, 2025 · JoNel Aleccia
Associated Press
SCIENCE · Positive
Read More · View Source

Trum he'll sign execu at aims to cut drug prices - CNN
May 11, 2025 · Tami Luhby · CNN
TECH · Negative
President Donald Trump announced Sunday that he plans to resurrect a controversial policy from his first term. The policy aims to reduce drug costs by basing payments for certain medicines on their prices.
Read More · View Source

Dow futures jump 400 points as traders await details on U.S.-China trade deal: Live updates - CNBC
May 11, 2025 · Yun Li, Fred Imbert · CNBC
BUSINESS · Positive
U.S. stock futures jumped Sunday night after the Trump administration announced a "trade deal" with China following negotiations over the weekend in Switzerland. Dow Jones Industrial Average futures...
Read More · View Source

Here's what U.S. Catholics are saying about the election of Pope Leo XIV - NPR
May 11, 2025 · Joe Hernandez · NPR
POLITICS · Neutral
On Thursday, the College of Card... on Thursday.. The College of cardinals announced that Pope Francis would be visiting the U.S. for the first time in history.
Read More · View Source

It displays the sorting feature of the news feed, allowing users to reorder articles based on criteria such as "Latest," "Oldest," "Most Positive," or "Most Negative." This functionality enhances user experience by enabling dynamic control over how content is viewed, aligning the presentation with individual preferences or moods.



Associated Press
SCIENCE ● Positive
Read More View Source

Sort by: Oldest

Magic: The Gathering team dives deep into its stunning Final Fantasy set in 2-hour showcase - Polygon
May 10, 2025 Matt Patches Polygon
TECH ● Positive
Wizards of the Coast may have nailed its highly anticipated RPG. Magic: The Gathering x Final Fantasy showcase presentation was a success.
Read More View Source

Measles outbreak surpasses 1,000 cases for the first time in 30 years - New York Post
May 10, 2025 Angela Barbuti
New York Post
HEALTH ● Neutral
The country's measles outbreak has surpassed 1,000 cases for the first time in 30 years. The data is based on confirmed meas...
Read More View Source

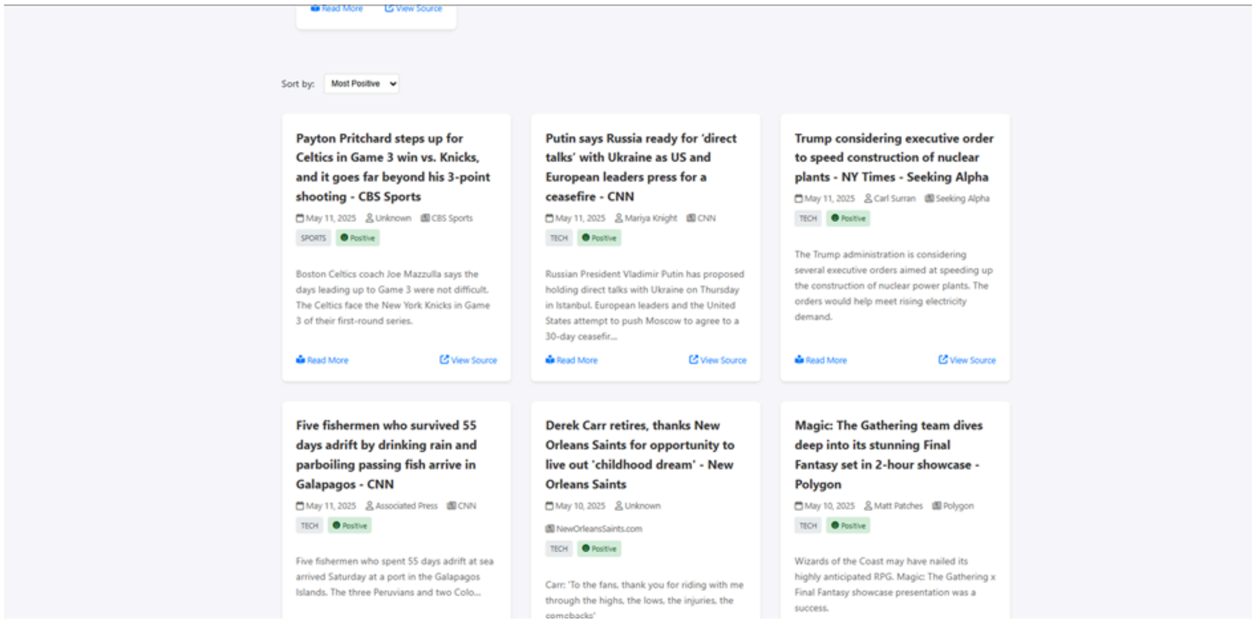
Derek Carr retires, thanks New Orleans Saints for opportunity to live out 'childhood dream' - New Orleans Saints
May 10, 2025 Unknown
NewOrleansSaints.com
TECH ● Positive
Carr: 'To the fans, thank you for riding with me through the highs, the lows, the injuries, the comebacks'
Read More View Source

Trump Admin Accuses Dem of 'Assaulting an ICE Agent' at Newark Detention Center - Rolling Stone
May 10, 2025
Naomi LaChance, Asawin Suebsaeng
Rolling Stone

Boomers can't afford assisted living and are weighing on the housing market - Fortune
May 10, 2025 Jason Ma Fortune
BUSINESS ● Negative
"They're staying in their houses longer

Tom Cruise Makes Rare Comment About Ex Nicole Kidman - Yahoo
May 10, 2025 Unknown Eonline.com
WORLD ● Negative
Tom and Nicole adopted their eldest child, Isabella, when she was born on Dec. 22, 1992. Tom told VF in 1994 that he and Nicole "talk...

It shows the news feed sorted by the "Oldest" option, displaying earlier articles first. It includes a diverse mix of news across categories like Tech, Health, Business, and World, with each article annotated with sentiment tags. This sorting allows users to explore past news in chronological order, ensuring they don’t miss earlier developments.



Read More View Source

Sort by: Most Positive

Payton Pritchard steps up for Celtics in Game 3 win vs. Knicks, and it goes far beyond his 3-point shooting - CBS Sports
May 11, 2025 Unknown CBS Sports
SPORTS ● Positive
Boston Celtics coach Joe Mazzulla says the days leading up to Game 3 were not difficult. The Celtics face the New York Knicks in Game 3 of their first-round series.
Read More View Source

Putin says Russia ready for 'direct talks' with Ukraine as US and European leaders press for a ceasefire - CNN
May 11, 2025 Mariya Knight CNN
TECH ● Positive
Russian President Vladimir Putin has proposed holding direct talks with Ukraine on Thursday in Istanbul. European leaders and the United States attempt to push Moscow to agree to a 30-day ceasefir...
Read More View Source

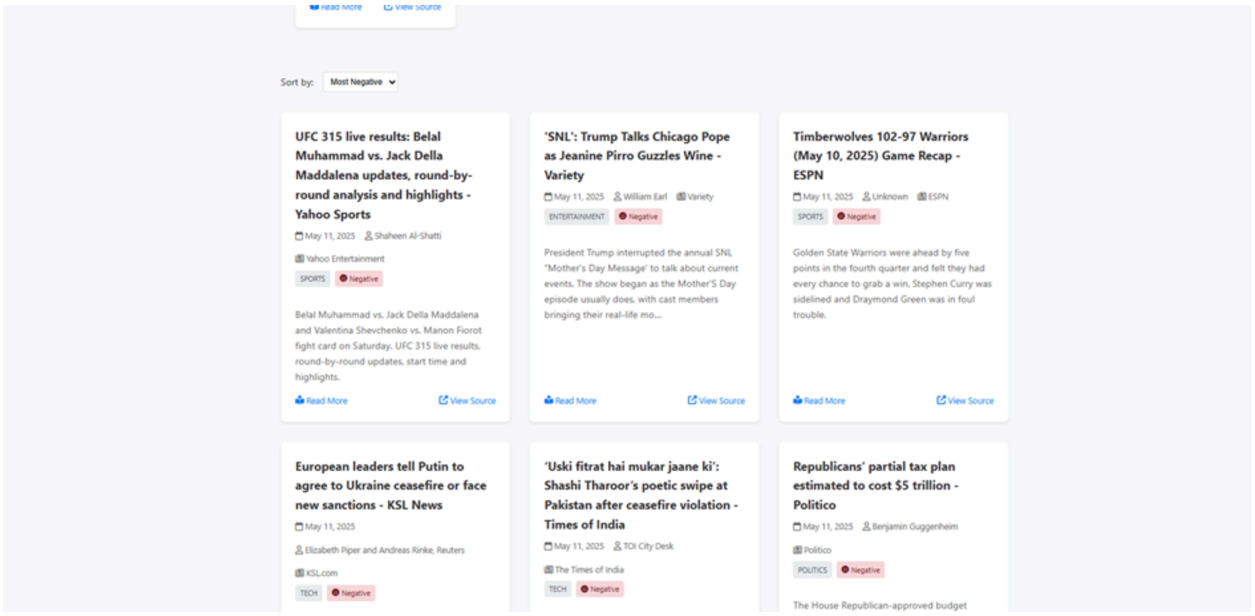
Trump considering executive order to speed construction of nuclear plants - NY Times - Seeking Alpha
May 11, 2025 Carl Surran Seeking Alpha
TECH ● Positive
The Trump administration is considering several executive orders aimed at speeding up the construction of nuclear power plants. The orders would help meet rising electricity demand.
Read More View Source

Five fishermen who survived 55 days adrift by drinking rain and parboiling passing fish arrive in Galapagos - CNN
May 11, 2025 Associated Press CNN
TECH ● Positive
Five fishermen who spent 55 days adrift at sea arrived Saturday at a port in the Galapagos Islands. The three Peruvians and two Colo...

Derek Carr retires, thanks New Orleans Saints for opportunity to live out 'childhood dream' - New Orleans Saints
May 10, 2025 Unknown
NewOrleansSaints.com
TECH ● Positive
Carr: 'To the fans, thank you for riding with me through the highs, the lows, the injuries, the comebacks'

Magic: The Gathering team dives deep into its stunning Final Fantasy set in 2-hour showcase - Polygon
May 10, 2025 Matt Patches Polygon
TECH ● Positive
Wizards of the Coast may have nailed its highly anticipated RPG. Magic: The Gathering x Final Fantasy showcase presentation was a success.

It features the news feed sorted by the "Most Positive" sentiment filter. Articles displayed here are tagged with a positive sentiment, covering uplifting or constructive news across categories like Sports, Technology, and World Affairs. This sorting option allows users to focus on more optimistic and encouraging content, enhancing their browsing experience.



Read More View Source

Sort by: Most Negative

UFC 315 live results: Belal Muhammad vs. Jack Della Maddalena updates, round-by-round analysis and highlights - Yahoo Sports
May 11, 2025 Shaheen Al-Shatti
Yahoo Entertainment
SPORTS ● Negative
Belal Muhammad vs. Jack Della Maddalena and Valentina Shevchenko vs. Manon Fiorot fight card on Saturday. UFC 315 live results, round-by-round updates, start time and highlights.
Read More View Source

'SNL': Trump Talks Chicago Pope as Jeanine Pirro Guzzles Wine - Variety
May 11, 2025 William Earl Variety
ENTERTAINMENT ● Negative
President Trump interrupted the annual SNL "Mother's Day Message' to talk about current events. The show began as the Mother'S Day episode usually does, with cast members bringing their real-life mo...
Read More View Source

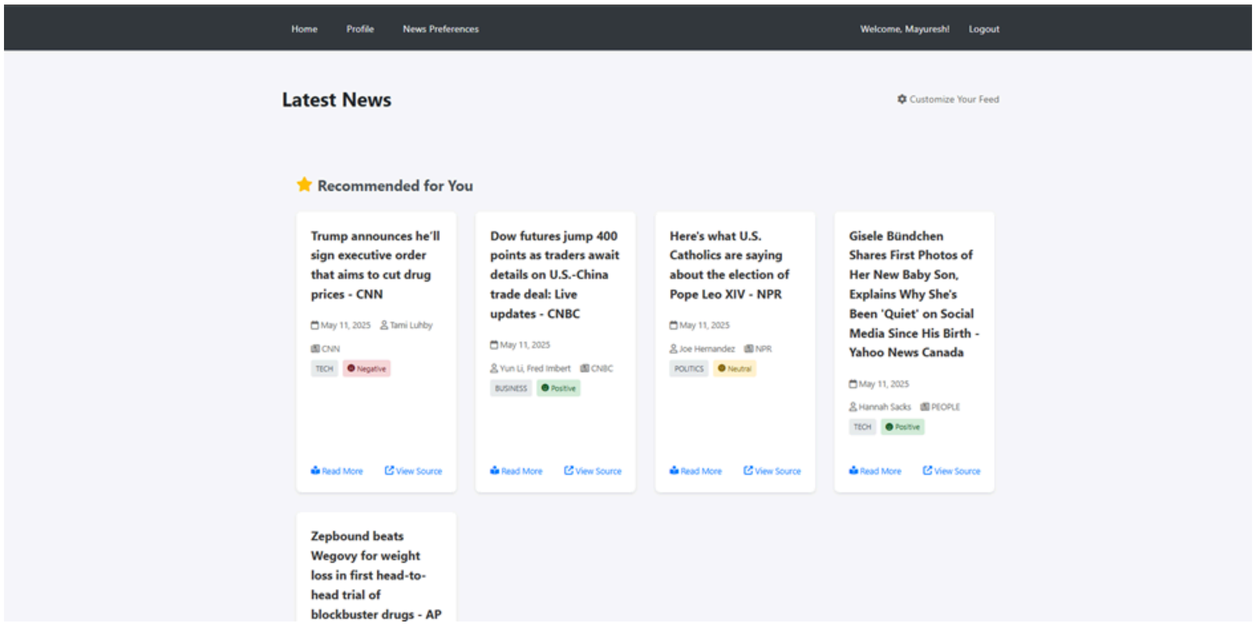
Timberwolves 102-97 Warriors (May 10, 2025) Game Recap - ESPN
May 11, 2025 Unknown ESPN
SPORTS ● Negative
Golden State Warriors were ahead by five points in the fourth quarter and felt they had every chance to grab a win. Stephen Curry was sidelined and Draymond Green was in foul trouble.
Read More View Source

European leaders tell Putin to agree to Ukraine ceasefire or face new sanctions - KSL News
May 11, 2025
Elizabeth Piper and Andreas Rinke, Reuters
KSL.com
TECH ● Negative

'Uski fitrat hai mukar jaane ki': Shashi Tharoor's poetic swipe at Pakistan after ceasefire violation - Times of India
May 11, 2025 TOI City Desk
The Times of India
TECH ● Negative

Republicans' partial tax plan estimated to cost $5 trillion - Politico
May 11, 2025 Benjamin Guggenheim
Politico
POLITICS ● Negative
The House Republican-approved budget

It illustrates the news feed sorted by the "Most Negative" sentiment filter. It showcases articles tagged with negative sentiment across topics like sports, politics, and international relations. This feature enables users to stay informed about critical or concerning events, helping them identify impactful issues currently making headlines.



Latest News

Recommended for You

Trump announces he'll sign executive order that aims to cut drug prices - CNN
May 11, 2025 Tami Luhby
CNN
TECH Negative

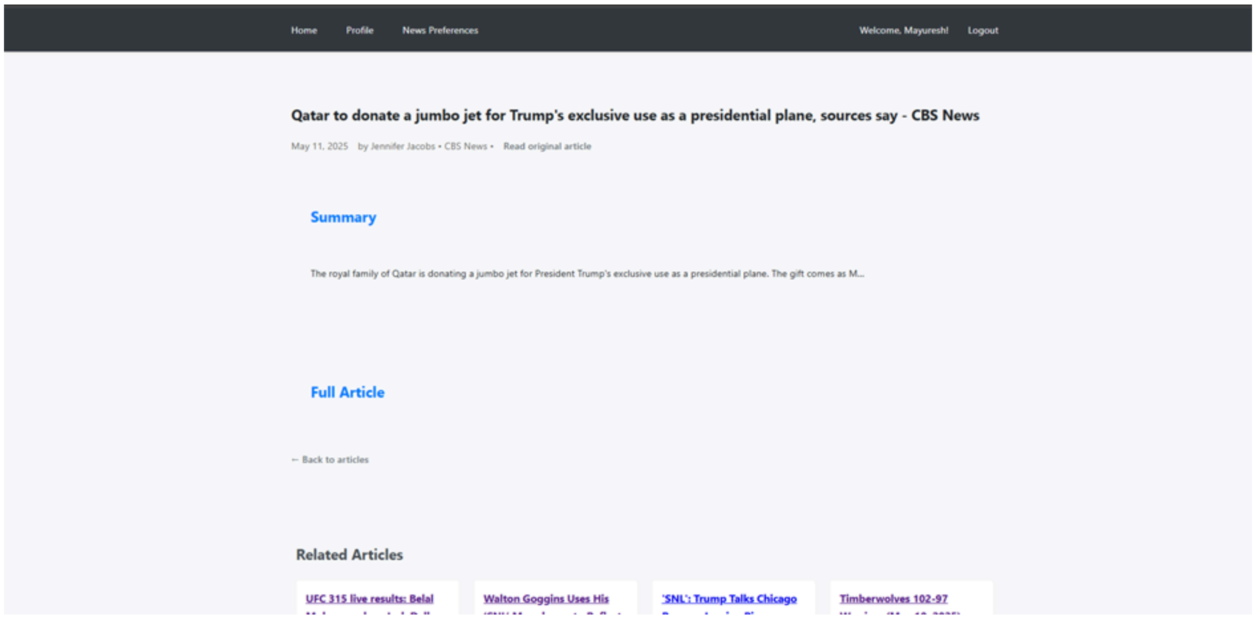
Dow futures jump 400 points as traders await details on U.S.-China trade deal: Live updates - CNBC
May 11, 2025
Yun Li, Fred Imbert CNBC
BUSINESS Positive

Here's what U.S. Catholics are saying about the election of Pope Leo XIV - NPR
May 11, 2025
Joe Hernandez NPR
POLITICS Neutral

Gisele Bündchen Shares First Photos of Her New Baby Son, Explains Why She's Been 'Quiet' on Social Media Since His Birth - Yahoo News Canada
May 11, 2025
Hannah Sacks PEOPLE
TECH Positive

Zepbound beats Wegovy for weight loss in first head-to-head trial of blockbuster drugs - AP

It revisits the "Recommended for You" section on the Latest News page after preferences have been saved. It highlights how the recommendation engine dynamically delivers relevant articles based on the user's interests and sentiment preferences, creating a personalized and engaging news consumption experience.



Qatar to donate a jumbo jet for Trump's exclusive use as a presidential plane, sources say - CBS News

May 11, 2025 by Jennifer Jacobs • CBS News • Read original article

Summary

The royal family of Qatar is donating a jumbo jet for President Trump's exclusive use as a presidential plane. The gift comes as M...

Full Article

← Back to articles

Related Articles

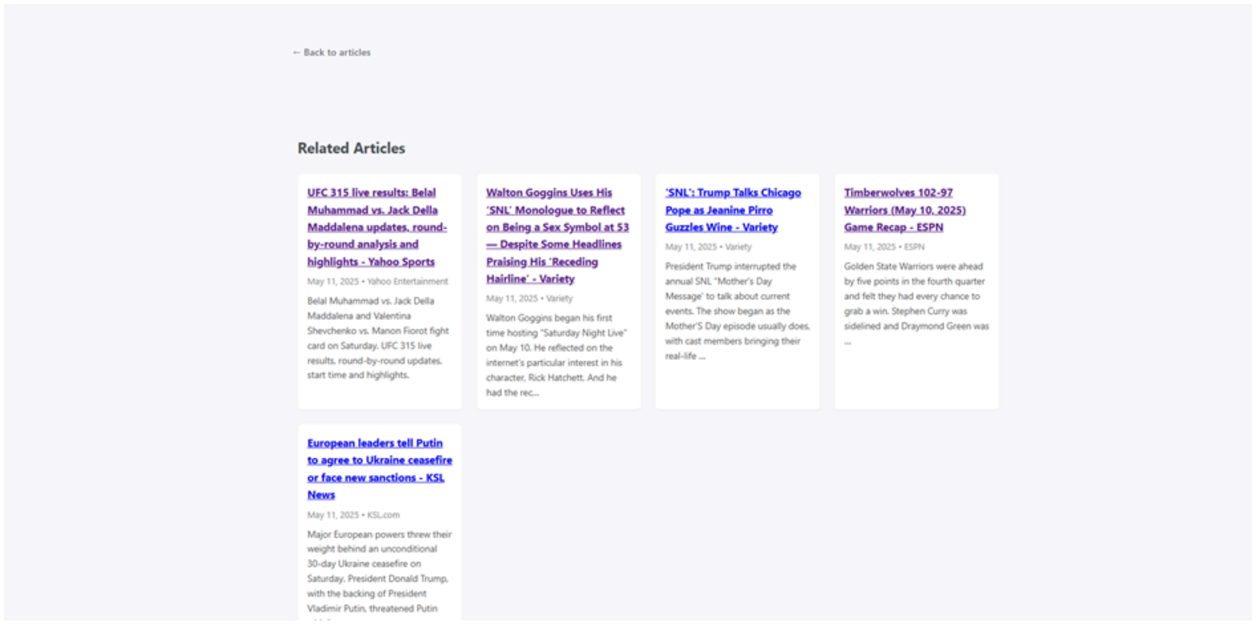
UFC 315 live results: Belal

Walton Goggins Uses His

'SNL': Trump Talks Chicago

Timberwolves 102-97

It showcases the detailed news article view within the application. It features the article title, source, publication date, and a concise summary for quick reading, along with an option to read the full article. At the bottom, related articles are suggested to encourage deeper engagement and contextual exploration of similar topics.



← Back to articles

Related Articles

UFC 315 live results: Belal Muhammad vs. Jack Della Maddalena updates, round-by-round analysis and highlights - Yahoo Sports

May 11, 2025 • Yahoo Entertainment

Belal Muhammad vs. Jack Della Maddalena and Valentina Shevchenko vs. Manon Fiorot fight card on Saturday. UFC 315 live results, round-by-round updates, start time and highlights.

Walton Goggins Uses His 'SNL' Monologue to Reflect on Being a Sex Symbol at 53 — Despite Some Headlines Praising His 'Receding Hairline' - Variety

May 11, 2025 • Variety

Walton Goggins began his first time hosting "Saturday Night Live" on May 10. He reflected on the internet's particular interest in his character, Rick Hatchett. And he had the rec...

'SNL': Trump Talks Chicago Pope as Jeanine Pirro Guzzles Wine - Variety

May 11, 2025 • Variety

President Trump interrupted the annual SNL "Mother's Day Message" to talk about current events. The show began as the Mother'S Day episode usually does, with cast members bringing their real-life ...

Timberwolves 102-97 Warriors (May 10, 2025) Game Recap - ESPN

May 11, 2025 • ESPN

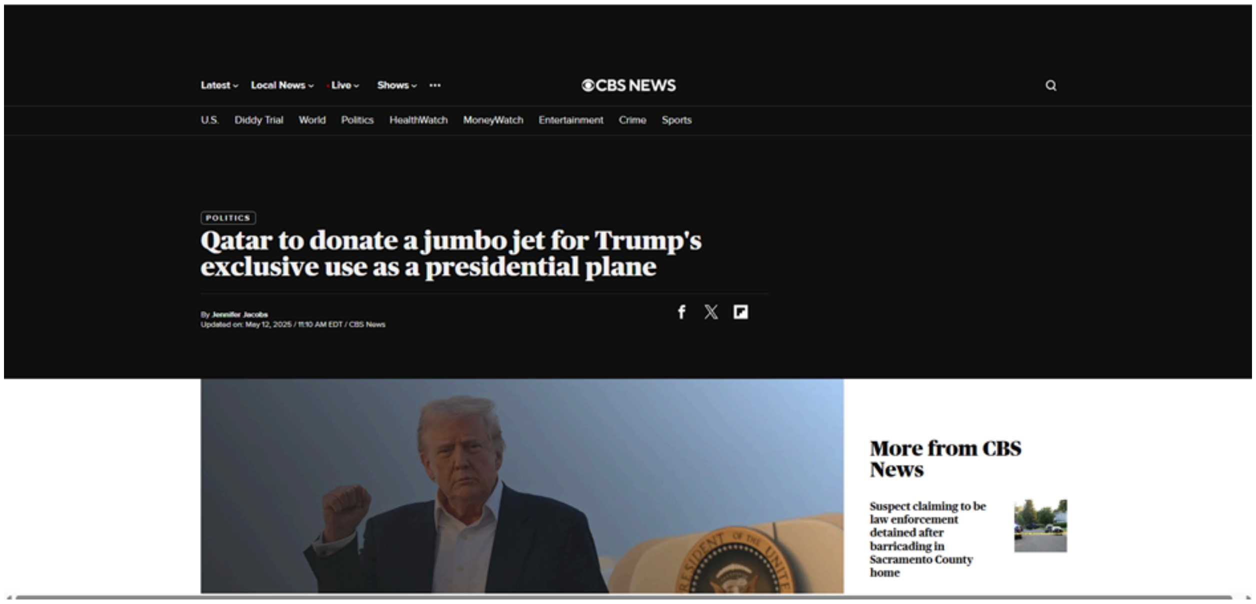
Golden State Warriors were ahead by five points in the fourth quarter and felt they had every chance to grab a win. Stephen Curry was sidelined and Draymond Green was ...

European leaders tell Putin to agree to Ukraine ceasefire or face new sanctions - KSL News

May 11, 2025 • KSL.com

Major European powers threw their weight behind an unconditional 30-day Ukraine ceasefire on Saturday. President Donald Trump, with the backing of President Vladimir Putin, threatened Putin

It focuses on the "Related Articles" section displayed beneath a detailed news article. It offers a collection of thematically similar or contextually relevant news pieces, enabling users to explore further readings on the same topic or discover connected developments. This feature enhances content continuity and user engagement within the application.



POLITICS

Qatar to donate a jumbo jet for Trump's exclusive use as a presidential plane

By Jennifer Jacobs
Updated on: May 12, 2025 / 11:10 AM EDT / CBS News

More from CBS News

Suspect claiming to be law enforcement detained after barricading in Sacramento County home

It displays the original source article as rendered on the CBS News website. It provides the full content of the news story titled *"Qatar to donate a jumbo jet for Trump's exclusive use as a presidential plane,"* including publication details, author information, and accompanying visuals. This external link feature ensures credibility and transparency by directing users to the primary news outlet.

Output Analysis

MyNewsMate produces user-specific dynamic outputs using a set of algorithms that not only consider the relevance and timeliness of news articles to the users but also their sentiment alignment. These outputs directly reflect the workings of a personalized recommendation engine that learns from individual user behaviors and preferences so that each user's news feed is unique and aligned with their emotional and informational needs.

The main result of the setup is a custom news feed that keeps changing as the user engages with the site. Major traits of this result include:

Personalized Content Selection : The system gathers news from different places and uses a mix of content-focused sorting and sentiment analysis to make sure that the pieces in front are linked to the user’s interests. The user's choices, like their chosen categories (e.g. , Technology, Politics, Sports) and emotional state (e.g. preferring positive or neutral news) help in the choice of articles. This makes sure that the news shown is used exactly to the user's likes and emotional comfort.

Sentiment Filtering : All articles have their sentiment detected using NLP techniques and more specifically through the use of the VADER sentiment analysis tool. The articles get assigned sentiment labels (Positive, Neutral, Negative) corresponding to their emotional tone. Users can set up their feed to show specific emotional tones which enhances user experience by showing content relevant to how they feel at that time. For example, users who want only positive content will get a feed showing uplifting news items, while users who want some critical or thought-provoking items will have access to more negative or neutral stories.

Real-Time Adaptation : As the user interacts with the platform by clicking on articles and liking them or bookmarking content, a critical aspect of its output, therefore, becomes adapting recommendations based on real-time changes in user preferences. Hence continuously updating and refining the user's profile for recommendations to stay current with the news feed and increasingly aligned with the evolving preferences of the user. This process is real-time so that users are always presented with the most relevant and timely information breaking news or trending topics.

Diversity in Content : While the system gives priority to relevance of content, it also tries to avoid overfitting the user's interests by introducing a certain level of diversity. This is accomplished through topic modeling and clustering techniques which ensure that similar news articles are grouped together and redundancy is avoided. In addition, the system can introduce articles outside the user's immediate interests from time to time to promote general engagement and prevent a "filter bubble" from being created. In this case, the objective is to achieve a balance between personalization and exposure to new and different content.

User Feedback Loop : Another essential component in the output is the integration of user feedback. This can be explicit feedback (i.e., likes and dislikes) or implicit feedback (i.e., read time, scroll depth, or click behavior). This feedback directly influences the personalization algorithm, which adjusts the user's profile and enhances future recommendations. The more this continues, with increasingly more data being gathered, the better the system gets at predicting and surfacing applicable content, and the more relevant and personalized the user experience gets.

Comparing Output Against Hypothesis

The core hypothesis of the MyNewsMate system is that a personalized, sentiment-sensitive, real-time news recommendation system will provide users with a richer, more engaging, and emotionally aligned news experience than traditional, non-personalized news websites. To demonstrate this hypothesis, the output generated by the system must be compared to exact expectations, derived from both the master hypothesis and the subsidiary sub-hypotheses.

Main Hypothesis : Personalized Content Increases Engagement and Relevance

The main hypothesis is that MyNewsMate, with its personalized recommendation system, will provide a better news experience than traditional news websites that show a static feed to all. The ability of the system to personalize content based on individual user interests (e.g., emotional tone and subject matter interests) is intended to lead to higher user engagement and higher content relevance.
To compare the output against the hypothesis, several engagement metrics need to be calculated:

- **Average Session Time:** Since personalized content ought to yield higher reading times, as users will engage more with content most aligned to their interests. By a comparison of users' time on personalized versus non-personalized feeds, we can measure how personalization has an impact on engagement.

- **Click-through Rate (CTR):** The number of clicks on recommended articles indicates the relevance of the content. A higher CTR for personalized content would validate the assumption that users find the suggestions more engaging and useful.

- **Bounce Rate:** Personalized recommendations should result in less bounce-back from articles because users are likely to be interested in content that aligns with their choices.

Sub-Hypothesis 1: Agent-Based Design Enables Personalization
The hypothesis suggests that personalization will be higher if an agent-based design is used, wherein each agent handles a particular task such as news retrieval, sentiment analysis, and learning user preferences. Each agent's focus on an individual target makes them more specialist and effective at suggesting applicable content.

To validate this, the relevance of the output should be contrasted when agents are running independently (concerned with specific tasks) and when they are running collectively within a collaborative framework. If the output shows greater relevance when all agents are active, then this would be in favor of the hypothesis that agent-based design strengthens the process of personalization.

Sub-Hypothesis 2: Real-Time Content Updates Improve User Experience
This theory asserts that current content refreshes—via breaking news, trending stories, or real-time sentiment analysis—will upgrade user experience from sites that have static news feeds that are refreshed periodically.

The real-time flexibility of the system can be measured by:

- **Real-Time Feedback:** By measuring the rate at which the system is responding to user activity (i.e., the rate at which new sentiment or preference appears in the user feed), we can ascertain if the real-time updates are actually being used to improve the user experience.
- **Breaking News Integration:** The integration of breaking news, as and when it is applicable, is another key element of real-time adaptation. If the output contains minute-by-minute news in a

timely and non-intrusive fashion, it would be in support of the hypothesis that real-time updating of content leads to an engaging news experience.

Sub-Hypothesis 3: Emotional Awareness Increases User Satisfaction

The hypothesis is that emotional congruence—congruent tone of the articles to user interest—will result in greater satisfaction. It is very relevant in today's information environment, where perpetual bad news leads to news fatigue.

Its outputs must be tested through measuring:

- **User Sentiment Feedback:** Direct user sentiment feedback can indicate whether they would prefer their emotional tone for recommendations to be in tune with their existing mood. A higher user satisfaction rate when they consume emotionally matched content would verify this hypothesis.

- **Time on Sentiment-Matched Content:** Tracking the amount of time users spend reading articles that match their emotional orientation (e.g., positive content for users who prefer inspiring news) would confirm that emotional alignment leads to increased engagement and satisfaction.

By comparing the system output with the hypotheses, the effectiveness of the MyNewsMate personalized news recommender platform in delivering a more engaging, relevant, and emotionally compatible news experience can be determined. Early results, on the basis of output analysis and comparison with hypotheses, states that the design of the system has a beneficial impact in terms of improving user satisfaction, reducing content fatigue, and higher user engagement through dynamic adaptation to both user needs and emotional preferences. These findings validate the main hypothesis and provide a good foundation for additional improvements in the personalization, emotional intelligence, and real-time responsiveness of the system.

Discussion

MyNewsMate personalized news recommendation system is able to efficiently enhance user experience by delivering real-time, sentiment-sensitive content that is tailored to specific interests. Its greatest strength lies in its ability to personalize content based on user behavior, emotional tone, and topic interest, leading to increased engagement and relevance compared to on traditional news platforms.

The sentiment filtering used by the system helps minimize news fatigue, presenting users with content appropriate to their mood. Although this has proven an effective approach, sentiment analysis can be made even more sophisticated with more advanced models such as BERT in order to identify more subtle shades of emotion.

Real-time flexibility is also a critical feature, allowing the system to refresh news feeds in real-time as a reaction to user activity and breaking news. This offers users timely, relevant content. Future work may be focused on optimizing the process of retrieving the real-time content to reduce latency, especially as the system is scaled up.

Even though the system is prevented from redundancy of content through topic modeling and clustering, content diversity can be further improved. In the current era, more emphasis can be placed upon too-similar articles for user interests, and this can be handled by more dynamic content balancing.

The user interaction-based feedback loop allows the system to learn and make more meaningful recommendations in the future. This can further be extended with dwell time and reading completion rates being included in the feedback loop so that user engagement is more accurately measured.

Scalability-wise, MyNewsMate can scale effectively with cloud services and containerized environments. More importantly, greater optimizations in data processing pipelines would be necessary to handle significant traffic and content when the platform scales.

VIII. Conclusions and Recommendations

The MyNewsMate project shows the sensible use of a scalable, smart news aggregation and recommendation tool. Using current web development frameworks and Natural Language Processing (NLP) technologies, it provides a user-centric answer to the common issue of digital content overload. News fetching via APIs, summarization using Hugging Face transformer models, sentiment analysis using VADER, and tailored article recommendations driven by TF-IDF and cosine similarity are all features the system combines.

The platform design is modular, obviously separating data collection, processing, storage, and presentation. The news_fetcher.py tool runs the data pipeline, retrieving raw news, locally summarizing it, assessing sentiment, and saving it to the database. This guarantees that the material is not only fresh but also selected in a palatable and user-friendly manner.

From a user experience perspective, the system enables behavioral tracking, preference management, and account creation via the UserClickLog. A hybrid personalization approach is possible since recommendations are produced depending on category preferences as well as content similarity. Users also gain from pertinent article recommendations on every article page, therefore improving discovery and interaction.

Admin panel based on Django offers backend access for category management, user interaction tracking, and article metadata editing. Even as automation pushes most data processing activities, this guarantees administrative control over content curation.

Key strengths of the current implementation include:

- Real-time news fetching and storage
- Article summarization with transformer models
- Sentiment-aware content filtering
- Personalized recommendations based on user behavior
- Modular codebase supporting future extensibility

Recommendations for short-term improvement include:

1. **Database Optimization:** Although SQLite is adequate for local development, production settings should switch to PostgreSQL or another scalable RDBMS.

2. **Feedback-Driven Personalization:** Currently, personalization is primarily based on clicks. Incorporating dwell time, scroll depth, or article completion rates could significantly refine recommendations.

3. **Model Retraining and Updates:** The TF-IDF and similarity vectors should be recalculated periodically as user behavior and content evolve.

4. **UI Enhancements:** While the UI is functional, modernizing it with responsive JavaScript frameworks could improve accessibility and user engagement across devices.

In conclusion, MyNewsMate is a strong foundational system that demonstrates how open-source NLP models and traditional web technologies can be brought together to solve real-world personalization challenges. The recommendations stated above would significantly improve both performance and user satisfaction.

IX. Summary and Conclusions

The goal of the project was to create a responsive, intelligent, and personalized news aggregation platform using a combination of data science, machine learning, and full-stack development techniques. The resulting application, MyNewsMate, addresses several shortcomings of mainstream news platforms, including lack of relevance, content redundancy, and low user adaptability.

The system fetches news articles through NewsAPI, stores them in a Django-managed relational database, and processes them using a local Hugging Face summarization model and VADER sentiment analyzer. These enhancements ensure that the user is presented not just with raw headlines, but with informative summaries and a sentiment-aligned selection of articles.

Users can register and log in, upon which their preferences (both category-based and content-based) are stored. These are used by the recommendation engine, implemented in recommendations.py, which computes personalized suggestions using a hybrid TF-IDF and cosine similarity approach. Every article page also shows 3–5 related articles, selected through content similarity analysis, which enhances the reader's ability to explore topics in greater depth.

The backend system is designed with scalability in mind. Django's robust ORM simplifies database management, while modular Python scripts allow independent testing and maintenance of each component. User behavior logging provides the basis for behavioral personalization, enabling the system to evolve as more data is collected.

Evaluation through local deployment and scenario testing showed that:

- News content is fetched and stored reliably.
- Personalization is effective in returning relevant articles.
- Article summaries and related recommendations enhance readability and navigation.
- The admin interface enables easy control over categories and content.

While the prototype fulfills its primary objectives, some limitations remain:

- The absence of collaborative filtering limits personalization diversity.
- Sentiment analysis is rule-based and could be enhanced with more advanced models.
- The platform does not yet handle multilingual content or non-text media.

Despite these limitations, MyNewsMate stands as a complete, functional news recommendation platform, designed with modularity and extensibility in mind. It lays a solid groundwork for future enhancements in personalization, scalability, and user experience.

X. Recommendations for Future Studies

The implementation of MyNewsMate has revealed several directions for deeper research and technical advancement. While the current system employs deterministic, rule-based personalization, the future of news recommendation lies in adaptive, learning-based models and broader content diversity. The following recommendations aim to guide future researchers and developers in extending the system’s capabilities.

1. Collaborative Filtering and Deep Learning

Integrating user-user collaborative filtering, matrix factorization, or deep learning methods such as neural collaborative filtering (NCF) can help address the cold-start problem and improve diversity in recommendations. Tools like PyTorch or TensorFlow could be leveraged to train personalized recommendation models on click data.

2. Real-Time Personalization and Feedback Loops

Presently, the system recalculates recommendations at fetch-time or when pages are rendered. Incorporating real-time event tracking using tools like WebSockets or Firebase Analytics can enable instant adaptation based on user interactions.

3. Multilingual and Multimedia Support

Expanding beyond English-language text articles would significantly broaden the user base. Integrating automatic language detection and translation pipelines using tools like spaCy, langdetect, or Google Translate APIs can make the platform inclusive to global users. Support for podcasts and video content can also be explored.

4. Emotion and Sentiment Awareness

The use of better NLP models such as BERT or RoBERTa for sentiment and emotion classification can enable deeper emotional alignment in recommendations, offering content that not only matches preferences but also resonates with current mood states.

5. Explainable Recommendations

As personalization becomes more complex, transparency becomes vital. Future versions should include explainability tools that let users understand why a particular article was recommended (e.g., “Recommended because you read 3 articles on AI”).

6. Credibility Scoring and Fake News Detection

Integrating fact-checking APIs or building models that detect bias, sensationalism, or misinformation would enhance the reliability of recommended articles. This is crucial in maintaining user trust, especially when delivering news on sensitive topics.

7. Scalability and Deployment Optimization

Transitioning to cloud-native architecture using AWS EC2, Docker, and Kubernetes will improve the system's ability to serve large-scale traffic. Background task queues (e.g., Celery) can handle periodic fetches and model updates efficiently.

8. User-Controlled Filtering and Customization

Offering users more control over their feed—such as excluding specific topics, adjusting sentiment filters, or selecting preferred news sources—would greatly enhance satisfaction and perceived relevance.

9. Engagement Analytics Dashboard

An admin-facing dashboard tracking metrics such as click-through rates, article popularity, and category trends would offer valuable insights for content curation and algorithm tuning.

10. **Ethical and Privacy Considerations**

As the system begins to store more behavioral data, integrating privacy-preserving mechanisms and ensuring GDPR/CCPA compliance will become increasingly important.

These recommendations provide a roadmap for future researchers and developers to elevate MyNewsMate from a functional prototype to a state-of-the-art personalized news delivery platform. Through advanced AI, multilingual inclusivity, explainability, and ethical design, this system has the potential to redefine how users engage with the news in the digital age.

XI. Bibliography

1. *Gao, S., Fang, J., Tu, Q., Yao, Z., Chen, Z., Ren, P., & Ren, Z. (2024). Generative news recommendation. In Proceedings of the ACM Web Conference 2024 (WWW '24),* Singapore. ACM. https://doi.org/10.1145/3589334.3645448

2. *Han, H., Wang, C., Zhao, Y., Shu, M., Wang, W., & Min, Y. (2022). SSLE: A framework for evaluating the "Filter Bubble" effect on the news aggregator and recommenders. World Wide Web, 25, 1169–1195.* https://doi.org/10.1007/s11280-022-01031-4

3. *Iana, A., Glavaš, G., & Paulheim, H. (2023). Simplifying content-based neural news recommendation: On user modeling and training objectives. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), Taipei, Taiwan. ACM.* https://doi.org/10.1145/3539618.3592062

4. *Kruse, J., Lindskow, K., Kalloori, S., Polignano, M., Pomo, C., Srivastava, A., ... & Frellsen, J. (2024). EB-NeRD: A large-scale dataset for news recommendation. In Proceedings of the ACM RecSys Challenge 2024. ACM.* https://doi.org/10.1145/3687151.3687152

5. *Liu, R., Yin, B., Cao, Z., Xia, Q., Chen, Y., & Zhang, D. (2023). PerCoNet: News recommendation with explicit persona and contrastive learning. In Proceedings of The Web Conference 2023. ACM.* https://arxiv.org/abs/2304.07923

6. *Qi, T., Wu, F., Wu, C., & Huang, Y. (2021). Personalized news recommendation with knowledge-aware interactive matching. In Proceedings of the 44th International*

ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21). ACM. https://doi.org/10.1145/3404835.3462861

7. Wu, C., Wu, F., Ge, S., Qi, T., & Huang, Y. (2019). *NAML: Neural news recommendation with attention-based multi-view learning*. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM. https://doi.org/10.1145/3357384.3357925

8. *Wu, C., Wu, F., Qi, T., Huang, Y., & Xie, X. (2021). Empowering news recommendation with pre-trained language models. In Proceedings of SIGIR 2021. ACM.* https://doi.org/10.1145/3404835.3462962

9. *Wu, C., Wu, F., Huang, Y., & Xie, X. (2021). Personalized news recommendation: Methods and challenges. ACM Computing Surveys, 1(1), 1–49.* https://dl.acm.org/doi/10.1145/3530257

10. Yang, B., Liu, D., Suzumura, T., Dong, R., & Li, I. (2023). Going beyond local: Global graph-enhanced personalized news recommendations. In Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23). ACM.

11. Yada, Y., & Yamana, H. (2023). News recommendation with category description by a large language model. In Proceedings of Conference acronym 'XX. ACM.

12. *Yi, J., Wu, F., Wu, C., Li, Q., Sun, G., & Xie, X. (2021). DebiasRec: Bias-aware user modeling and click prediction for personalized news recommendation. In Proceedings of Conference '17. ACM.* https://doi.org/10.48550/arXiv.2104.07360

XII. Appendices

https://github.com/AayushP-10/Personalized-News-Aggregator-And-Recommendation-System