NewsSum

AI-powered news summarization

Submitted by: Ayush Pratap

Organisation: Condigence LLP

Role: Full Stack Developer Intern

Under the guidance of: Mr. Vishal Aryan

Github Repository: https://github.com/Ayush-Pratap-Tripathi/Condigence-LLP---Full-Stack-development-Intern/tree/main/newsSum

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to **Condigence LLP** for providing me with the invaluable opportunity to work as a **Full Stack Developer Intern** during the period **October 1, 2025 – January 31, 2026**. This internship has been an immensely enriching and transformative experience, allowing me to bridge the gap between theoretical knowledge and real-world software development practices. It offered me hands-on exposure to designing, developing, and deploying a complete full-stack application while adhering to industry-level standards.

I am profoundly thankful to **Mr. Vishal Aryan**, my respected mentor and guide, for his continuous supervision, insightful feedback, and unwavering encouragement throughout the development of this project. His technical expertise, patience, and in-depth understanding of full-stack development, backend architecture, and system design played a pivotal role in shaping this project from its initial concept to its successful implementation. His guidance helped me understand not only *how* to build systems, but also *why* certain architectural and design decisions are crucial in scalable and maintainable applications.

I would also like to extend my sincere appreciation to the entire team at **Condigence LLP** for their collaborative environment, constructive discussions, and valuable suggestions. Their cooperative spirit and willingness to share knowledge significantly contributed to overcoming technical challenges and improving the overall functionality, performance, and reliability of the system.

The project titled **“newsSum – An AI-Powered News Summarization Platform”** was developed during my internship tenure as part of my professional learning and contribution to the organization. This project provided me with the opportunity to explore the practical integration of **Artificial Intelligence**, **Natural Language Processing**, and **Full-Stack Web Development**. Through this work, I gained hands-on experience with technologies such as **Django**, **Django REST Framework**, **MySQL**, **MongoDB**, **React.js**, and **Hugging Face Transformer models**, while also learning how to integrate third-party APIs and manage real-world data pipelines.

Finally, I would like to acknowledge the numerous online resources, official documentation, open-source libraries, and developer communities whose contributions played an essential role in the successful completion of this project. Their shared knowledge and tools greatly supported my learning and problem-solving process.

This internship journey has been a significant milestone in my professional development. The skills, experience, and insights gained through this project will continue to guide and inspire me in my future endeavors as a software developer.

Submitted By: Ayush Pratap

Full Stack Developer – Intern, Condigence LLP

ABSTRACT

In today's information-driven digital era, users are exposed to an overwhelming volume of news content from multiple online sources, making it difficult to quickly extract meaningful insights from lengthy articles. Reading full news articles is often time-consuming and inefficient, especially for users seeking concise and accurate information. To address this challenge, **newsSum – An AI-Powered News Summarization Platform** has been developed as an intelligent web-based system that automates the process of news aggregation and summarization.

newsSum fetches real-time news articles from external news sources and generates concise, human-readable summaries using advanced **Artificial Intelligence (AI)** and **Natural Language Processing (NLP)** techniques. The system leverages an abstractive summarization approach, enabling it to generate summaries that capture the core meaning of the article rather than merely extracting key sentences. This significantly enhances readability and comprehension while preserving contextual relevance.

The application is built using a **Django and Django REST Framework** backend, a **React.js (Vite)** frontend, and a hybrid database architecture consisting of **MySQL** and **MongoDB**. MySQL is used for structured user authentication and relational data management, while MongoDB stores unstructured data such as news articles, summaries, and user interaction history. The platform integrates the **WorldNews API** to fetch real-time news content and utilizes the **Hugging Face Inference API** with the **google/pegasus-xsum** transformer model to generate high-quality abstractive summaries.

newsSum features a secure authentication system, user-specific data handling, theme management, and an intuitive user interface that allows users to browse news, generate summaries on demand, and manage their reading history efficiently.

By automating the news summarization process, newsSum significantly reduces the time required to consume news content, improves information accessibility, and enhances user productivity. The platform provides a fast, intelligent, and user-centric solution for modern news consumption, demonstrating the effective application of AI-driven summarization in real-world web applications.

CONTENTS

1. Problem Statement

Overview

With the rapid growth of digital media and online journalism, users today are exposed to an overwhelming amount of news content across multiple platforms. News articles are often lengthy, detailed, and published at a very high frequency, making it difficult for readers to consume, analyze, and extract essential information efficiently. In many cases, users are forced to skim through entire articles or rely on headlines, which may not always convey the complete or accurate context of the news.
This information overload leads to reduced productivity, increased cognitive fatigue, and the risk of missing critical insights hidden within long-form news articles. The challenge becomes even more significant for professionals, students, and researchers who need quick yet reliable summaries to stay informed without spending excessive time reading full articles.

Core Challenges Identified:

- **Information Overload**: News platforms publish thousands of articles daily, making it impractical for users to read full-length content for every topic of interest.
- **Time Constraints**: Users often lack sufficient time to read complete articles, especially when news content is verbose or highly detailed.
- **Inefficient Summarization Methods**: Manual summarization is subjective, inconsistent, and not scalable for large volumes of content.
- **Lack of Intelligent Tools**: Many existing platforms either provide extractive summaries or rely on headlines, which may omit important context or nuances.
- **Poor Personalization and Accessibility**: Users require a system that allows them to generate summaries on demand rather than consuming fixed or pre-generated content.

Limitations of Existing Systems

Most existing news platforms focus on content delivery rather than content comprehension. While some platforms offer short descriptions or highlights, they often:

- Depend on **extractive summarization**, which simply selects sentences from the article without rephrasing or contextual understanding.
- Do not leverage advanced **AI-based abstractive summarization**, resulting in summaries that may feel disjointed or incomplete.
- Lack user-centric features such as summary history, structured data storage, and seamless integration with modern web interfaces.

As a result, users still spend considerable effort navigating through content to understand the core message of news articles.

Need for an Automated and Intelligent Solution

To address these challenges, there is a clear need for an intelligent system that can:

- Automatically fetch real-time news from reliable sources.
- Generate **concise, coherent, and context-aware summaries** using advanced AI models.
- Reduce the time and effort required to consume news while preserving accuracy.
- Provide a clean, responsive, and user-friendly interface for seamless interaction.
- Ensure scalability, reliability, and secure handling of user data.

newsSum's Solution Intent

The **newsSum** project aims to solve these challenges by introducing an **AI-powered news summarization platform** that automates the process of news aggregation and summarization. By leveraging state-of-the-art **Natural Language Processing (NLP)** techniques and **abstractive summarization models**, newsSum transforms lengthy news articles into meaningful summaries that retain essential context and intent.
The system is designed as a full-stack web application that integrates real-time news APIs, AI-driven summarization, and a modern frontend interface. It enables users to quickly understand news content, stay informed efficiently, and manage summarized information effectively — thereby addressing the core limitations of traditional news consumption methods.

2. Objectives

Primary Objective

The primary objective of the **newsSum** project is to design and develop an **AI-powered web application** that automates the process of news aggregation and summarization, enabling users to quickly understand lengthy news articles by generating concise, accurate, and context-aware summaries using advanced Natural Language Processing (NLP) techniques.

The system aims to reduce information overload and time consumption associated with traditional news reading, while ensuring that the summarized content preserves the original article's intent, relevance, and key insights.

Specific Objectives

The specific objectives of the newsSum project are as follows:

1. **To develop a full-stack web application for news summarization**
   Design and implement a complete full-stack system using a modern frontend and a robust backend architecture that allows users to browse news articles and generate AI-based summaries seamlessly.
2. **To integrate real-time news fetching from external sources**
   Utilize a third-party news API to fetch up-to-date news articles dynamically, ensuring that users always have access to current and relevant information across multiple domains.
3. **To implement AI-based abstractive text summarization**
   Integrate an advanced transformer-based NLP model to generate abstractive summaries that rephrase and condense news articles instead of merely extracting sentences, resulting in more natural and readable summaries.
4. **To leverage cloud-based AI inference services**
   Offload computationally intensive summarization tasks to a managed AI inference platform, enabling efficient processing without requiring local model hosting or heavy infrastructure.
5. **To design a secure user authentication and authorization system**
   Implement a secure authentication mechanism that allows users to register, log in, and access personalized features while ensuring proper access control and data protection.
6. **To support efficient data storage and management**
   Store structured user data and unstructured news-related data using appropriate database technologies, ensuring scalability, fast retrieval, and reliable persistence of summaries and user interactions.
7. **To provide a clean, responsive, and user-friendly interface**
   Develop an intuitive frontend interface that enables smooth navigation, clear presentation of news content, and easy interaction with summarization features across different screen sizes and devices.
8. **To ensure modularity and maintainability of the system**
   Follow a modular design approach for both frontend and backend components so that the system can be easily extended, maintained, or enhanced in the future.
9. **To enhance productivity and information accessibility**
   Enable users to consume large volumes of news content efficiently by significantly reducing reading time while maintaining comprehension and accuracy.

Outcome Expectation

By achieving these objectives, the newsSum project aims to deliver a **scalable, intelligent, and user-centric news summarization platform** that demonstrates the practical application of AI and NLP in real-world web systems. The project is expected to improve news consumption efficiency, reduce cognitive load on users, and serve as a strong foundation for future enhancements such as personalization, category-based summaries, and advanced analytics.

3. Technology Stack

Overview

The **newsSum** project is implemented as a **full-stack web application** that combines a modern JavaScript-based frontend, a Python-based backend, multiple databases, and cloud-hosted AI inference services. The selected technology stack ensures scalability, maintainability, secure data handling, and efficient integration of Artificial Intelligence for news summarization.

Each technology was chosen based on its suitability for real-world production systems, ease of integration, and alignment with the project's functional and non-functional requirements.

Frontend Technologies

- **React.js**
  - Used for building the single-page application (SPA) that handles user interaction, routing, and dynamic UI updates.
  - Implements component-based architecture for better modularity and reusability.
  - Key directories:
    frontend/src/pages/, frontend/src/components/, frontend/src/services/
- **Vite**
  - Serves as the frontend build tool and development server.
  - Chosen for its fast cold-start time, efficient bundling, and hot module replacement (HMR).
  - Configuration file: frontend/vite.config.js
- **JavaScript (ES6+)**
  - Used throughout the frontend for application logic, API integration, and state handling.
- **CSS (Global Styling)**
  - Styling is handled through global CSS files ensuring consistent theming across all pages.
  - Main stylesheet: frontend/src/index.css
- **Axios**
  - Used as the HTTP client for all frontend-to-backend API communication.
  - Encapsulated inside service files for clean separation of concerns.
  - Handles request configuration, base URL management, and error handling.

Frontend Environment Configuration

- **VITE_API_BASE**
  - Defines the base URL for backend authentication and API endpoints.
  - Enables easy switching between development and production environments without code changes.

Backend Technologies

Primary Technologies

- **Python**
  - Core programming language used for backend logic, API handling, and AI orchestration.
- **Django**
  - Serves as the primary backend framework.
  - Provides built-in security features, ORM support, and a structured project layout.
  - Handles user authentication, request routing, and business logic.
- **Django REST Framework (DRF)**
  - Used to build RESTful APIs consumed by the React frontend.
  - Enables clean serialization, validation, and API endpoint structuring.
  - Supports scalable API development and standardized response formats.

Backend Environment Configuration

- **SECRET_KEY**
  - Used by Django for cryptographic signing and security-related operations.
- **DEBUG**
  - Enabled during development for detailed error reporting.
- **CORS_ORIGIN**

    - Configured to allow secure communication between the frontend (running on a different port) and the backend.

Database Technologies

Relational Database

- **MySQL**
    - Used for storing structured and relational data such as user authentication details.
    - Provides strong consistency, transactional integrity, and reliable query performance.
    - Environment variables:
        - MYSQL_DB
        - MYSQL_USER
        - MYSQL_PASSWORD
        - MYSQL_HOST
        - MYSQL_PORT

NoSQL Database

- **MongoDB Atlas**
    - Used for storing unstructured and semi-structured data such as AI-generated summaries
    - Chosen for its schema flexibility and scalability.
    - Cloud-hosted to ensure high availability and easy scaling.
    - Connection handled via:
        - MONGO_URI

This hybrid database approach allows the system to efficiently manage both relational and document-based data.

External APIs and AI Integration

News Data Source

- **WorldNews API**
    - Used to fetch real-time news articles from external sources.
    - Provides up-to-date and structured news content for summarization.
    - API key configured using:
        - WORLDNEWS_API_KEY

AI and NLP Integration

- **Hugging Face Inference API**
    - Used to perform abstractive text summarization.
    - Eliminates the need for local model hosting and heavy compute resources.
- **Transformer Model: google/pegasus-xsum**
    - A state-of-the-art abstractive summarization model.
    - Specifically optimized for summarizing news articles into concise outputs.
    - Produces human-readable summaries while preserving semantic meaning.
    - Configured using:
        - HF_API_KEY
        - HF_MODEL

Security and Configuration Management

- **Environment Variables**
    - All sensitive information such as API keys, database credentials, and secret keys are stored in .env files.
    - Prevents hardcoding of secrets in the source code and improves security.
- **CORS Configuration**
    - Ensures controlled cross-origin communication between frontend and backend.
- **Authentication Mechanism**
    - Django’s authentication system is used as the foundation for user management and secure access control.

Summary

The technology stack of **newsSum** reflects a carefully chosen combination of modern web technologies, scalable databases, and cloud-based AI services. By integrating Django and React with external APIs and transformer-based NLP models, the system achieves a balance between performance, scalability, and intelligence. This stack enables newsSum to function as a production-ready AI-powered news summarization platform while remaining extensible for future enhancements.

4. System Architecture

Overview

The **newsSum** system is designed using a **client–server architecture** combined with **external AI and data services**. The architecture emphasizes modularity, scalability, and clear separation of concerns between presentation, business logic, data storage, and AI inference.

At a high level, the system consists of:

- A **frontend client** responsible for user interaction and presentation
- A **backend server** that handles authentication, data processing, API orchestration, and persistence
- **Databases** for structured and unstructured data storage
- **External APIs** for news retrieval and AI-based summarization

This layered architecture ensures that each component operates independently while communicating through well-defined REST APIs.

High-Level Architectural Components

1. Client Layer (Frontend)

- Implemented using **React.js with Vite**
- Runs in the user's web browser
- Responsible for:
    - Rendering UI components
    - Handling user interactions
    - Sending requests to backend APIs
    - Displaying fetched news articles and generated summaries

The frontend communicates with the backend exclusively through **HTTP-based REST APIs** using Axios. All business logic, AI processing, and data persistence are delegated to the backend.

2. Application Layer (Backend)

- Implemented using **Django and Django REST Framework**
- Acts as the core controller of the system
- Responsibilities include:
    - User authentication and authorization
    - API request validation and response formatting
    - Fetching news data from external APIs
    - Orchestrating AI-based summarization
    - Managing interactions with both relational and NoSQL databases

The backend exposes a set of REST endpoints that the frontend consumes. It also acts as an intermediary between external services (news APIs and AI inference APIs) and internal storage.

3. Data Layer

The newsSum project follows a **hybrid data storage architecture** to efficiently handle different types of data.

Relational Database (MySQL)

- Stores structured data such as:
    - User authentication details
    - User account metadata
- Ensures strong consistency and transactional integrity for critical user-related operations.

NoSQL Database (MongoDB Atlas)

- Stores semi-structured and unstructured data such as News articles fetched from external sources
- Chosen for its flexibility, scalability, and suitability for document-based data.

This separation ensures optimal performance and maintainability by using the right database for the right type of data.

4. External Service Layer

News Data Service

- **WorldNews API** is used to retrieve real-time news articles.
- The backend sends authenticated requests using the configured API key.
- Retrieved data is processed and normalized before being stored or sent to the frontend.

AI Inference Service

- **Hugging Face Inference API** is used for text summarization.
- The backend sends article content to the API along with the configured model (google/pegasus-xsum).
- The API returns an abstractive summary, which is then processed and stored by the backend.

By using managed external services, the system avoids the complexity of hosting and scaling large AI models locally.

Detailed Data Flow

1. User Interaction Flow

1. The user accesses the newsSum web application via the browser.
2. The frontend renders the UI and triggers API requests based on user actions.
3. Requests are sent to the backend using Axios with the configured API base URL.

2. News Fetching Flow

1. The backend sends a request to the WorldNews API using the configured API key.
2. The API returns structured news data.
3. The backend processes.
4. The processed news articles are returned to the frontend for display.

3. Summarization Flow

1. The user selects or opens a news article.
2. The frontend sends the article content or identifier to the backend.
3. The backend forwards the article text to the Hugging Face Inference API.
4. The google/pegasus-xsum model generates an abstractive summary.
5. The backend stores the summary in MongoDB and returns it to the frontend.
6. The frontend displays the summarized content to the user.

4. Authentication and Security Flow

1. User authentication requests are handled by Django.
2. User credentials are validated against MySQL-stored data.
3. Authorized users are granted access to protected API endpoints.
4. Cross-origin access is controlled using CORS configuration.

Architectural Characteristics

- **Separation of Concerns**: UI, business logic, data storage, and AI processing are clearly separated.
- **Scalability**: External APIs and cloud-hosted databases allow the system to scale with increasing usage.
- **Maintainability**: Modular backend services and frontend components simplify updates and debugging.
- **Security**: Sensitive configuration is managed via environment variables; direct database access from the frontend is prevented.

Summary

The system architecture of **newsSum** follows a clean, modular, and scalable design that integrates modern web technologies with AI-powered services. By combining a React-based frontend, a Django REST backend, hybrid databases, and external AI inference APIs, the platform efficiently delivers real-time news summaries while maintaining performance, security, and extensibility. This architecture forms a robust foundation for future enhancements and production-level deployment.

5. Functional Requirements

Overview

Functional requirements define the **core capabilities and behaviors** of the **newsSum** system. These requirements specify what the system must do to successfully fetch news, summarize content using AI, manage users, and present information through a user-friendly interface.
The functional requirements of newsSum are derived directly from the implemented frontend–backend workflows, API integrations, database interactions, and AI inference pipeline.

User Authentication and Access Management

1. **User Registration**
   - The system shall allow users to create an account by providing required credentials.
   - User details must be securely stored in the relational database.
2. **User Login**
   - Registered users shall be able to log in using valid credentials.
   - The backend shall authenticate the user and allow access to protected functionalities.
3. **Session Handling**
   - The system shall maintain authenticated user sessions across frontend interactions.
   - Unauthorized users attempting to access restricted APIs must be denied access.
4. **Secure API Access**
   - Only authenticated users shall be allowed to access protected backend endpoints related to news interaction and summarization.

News Retrieval and Management

1. **Fetch Real-Time News**
   - The system shall retrieve real-time news articles from external news sources using the WorldNews API.
   - News data shall be fetched dynamically based on system configuration.
2. **News Data Processing**
   - The backend shall process and normalize the fetched news data before exposing it to the frontend.
   - Relevant article details such as title, content, source, and publication data must be handled correctly.
3. **Display News Articles**
   - The frontend shall display a list of available news articles in a readable and structured format.
   - Users shall be able to select individual articles for detailed viewing.

AI-Based News Summarization

1. **Summarization Request Handling**
   - The system shall allow users to request a summary for a selected news article.
   - The frontend shall send the article content or reference to the backend for processing.
2. **AI Inference Integration**
   - The backend shall invoke the Hugging Face Inference API to perform abstractive summarization.
   - The summarization process shall use the google/pegasus-xsum transformer model.
3. **Summary Generation**
   - The system shall generate concise, coherent, and context-aware summaries of news articles.
   - Generated summaries must retain the key information and intent of the original article.
4. **Summary Storage**
   - The backend shall store AI-generated summaries in MongoDB for persistence and future access.
5. **Summary Delivery**
   - The system shall return the generated summary to the frontend in a structured response format.
   - The frontend shall display the summary clearly to the user.

User Interaction and History Management

1. **User-Specific Data Handling**
   - The system shall associate summaries with the authenticated user.
   - User-specific data must be isolated from other users.

2. **Summary History Access**
    - The system shall allow users to view previously generated summaries.
    - Stored summaries must be retrievable efficiently from the database.
3. **Content Reusability**
    - The system shall support re-accessing stored summaries without regenerating them unnecessarily.

Frontend–Backend Communication

1. **REST API Communication**
    - All communication between the frontend and backend shall occur through RESTful APIs.
    - API responses must follow consistent and predictable formats.
2. **Axios Integration**
    - The frontend shall use Axios for all HTTP requests.
    - API base URLs shall be configurable using environment variables.
3. **Error Handling and Feedback**
    - The system shall return meaningful error responses for failed API requests.
    - The frontend shall display appropriate feedback messages to users.

Configuration and Environment Handling

1. **Environment-Based Configuration**
    - The system shall load sensitive configuration values (API keys, database credentials, secret keys) from environment variables.
2. **CORS Management**
    - The backend shall allow requests only from authorized frontend origins.

Summary

The functional requirements defined above ensure that **newsSum** operates as a complete, secure, and intelligent news summarization platform. These requirements collectively support real-time news retrieval, AI-driven summarization, secure user interaction, and efficient data management. By fulfilling these functional requirements, the system delivers its intended value of simplifying and accelerating news consumption through automation and AI.

6. Non-Functional Requirements

Overview

Non-functional requirements define the **quality attributes and operational constraints** of the **newsSum** system. While functional requirements specify *what* the system does, non-functional requirements describe *how well* the system performs its tasks in terms of performance, security, reliability, usability, scalability, and maintainability.
These requirements ensure that newsSum operates as a **robust, secure, and user-friendly AI-powered application** suitable for real-world usage.

Performance Requirements

Attribute	Description
Response Time	The system should return fetched news articles within a few seconds under normal network conditions. AI-generated summaries should typically be returned within **5–10 seconds**, depending on article length and Hugging Face API latency.
AI Inference Latency	Summarization performance depends on the Hugging Face Inference API; the backend must handle inference calls efficiently without blocking unrelated requests.
Database Access Time	MySQL and MongoDB queries for individual user data or summaries should complete within a reasonable time frame (< 300 ms per operation under normal load).
Frontend Load Time	The React + Vite frontend should load quickly, with optimized bundles and minimal blocking scripts to ensure smooth initial rendering.

Reliability and Availability

Attribute	Description
System Availability	The system should remain available as long as backend and database services are running. Cloud-hosted MongoDB Atlas improves availability and fault tolerance.
Error Recovery	Backend services must gracefully handle failures such as external API downtime, invalid requests, or inference errors and return meaningful error messages.
Data Persistence	Stored news articles and summaries must persist reliably across sessions and system restarts.
External Dependency Handling	The system must handle failures from WorldNews API or Hugging Face API without crashing the application.

Security Requirements

Attribute	Description
Authentication Security	User authentication must ensure that only valid users can access protected resources.
Data Protection	Sensitive data such as API keys, database credentials, and secret keys must never be hard-coded and should be managed via environment variables.
CORS Control	Backend must restrict API access to authorized frontend origins only, preventing unauthorized cross-origin requests.
Database Security	Database access credentials must be secured and database access limited to backend services only.
API Security	Direct access to external APIs (WorldNews, Hugging Face) must be handled exclusively by the backend to avoid key exposure.

Usability Requirements

Attribute	Description
User Interface Simplicity	The UI must be intuitive and easy to navigate, allowing users to access news and summaries with minimal effort.
Responsiveness	The frontend must adapt smoothly to different screen sizes, including desktops and mobile devices.
Feedback and Messaging	Users must receive clear feedback for loading states, errors, and successful operations.
Readability	Generated summaries must be presented in a clear and readable format for easy comprehension.

Maintainability and Extensibility

Attribute	Description
Modular Code Structure	Backend logic should be modularized into views, serializers, services, and utilities. Frontend logic should follow component-based design.
Ease of Updates	Individual modules (news fetching, summarization, UI components) should be updatable without affecting the entire system.
Technology Maintainability	Widely adopted frameworks (Django, React) ensure long-term support and community-driven improvements.
Future Enhancements	The system should support future features such as category-based news, personalization, caching, or analytics with minimal restructuring.

Portability and Deployment

Attribute	Description
Platform Independence	The application should run on any modern browser without OS-specific dependencies.
Environment Configuration	The system must support environment-specific configurations using .env files for development and production.
Cloud Readiness	MongoDB Atlas and external APIs enable cloud-based deployment without major configuration changes.

Compliance and Best Practices

Attribute	Description
Coding Standards	Backend and frontend code should follow framework-recommended best practices and clean coding standards.
Security Best Practices	Sensitive data must follow secure handling practices aligned with common web security guidelines.
AI Usage Responsibility	AI-generated summaries should be treated as assistive content and not as authoritative sources of truth.

Summary

The non-functional requirements outlined above ensure that **newsSum** is not only functionally complete but also **secure, reliable, scalable, maintainable, and user-centric**. These qualities collectively strengthen the system's readiness for real-world usage and provide a solid foundation for future expansion and performance optimization.

7. Module Description

Overview

The **newsSum** system is organized into multiple well-defined modules, each responsible for a specific functional area of the application. This modular design ensures clear separation of concerns, improves maintainability, and allows individual components to be extended or modified independently.
The major modules of the system include:

1. Authentication and User Management Module
2. News Fetching Module
3. AI-Based Summarization Module
4. News and Summary Management Module
5. Frontend User Interface Module
6. Configuration and Utility Module

Each module is described in detail in the following sections.

Authentication and User Management Module

Description

This module handles user registration, login, and access control within the system. It ensures that only authenticated users can access protected functionalities such as viewing news summaries or maintaining personal interaction history.

Key Functionalities

- User registration and credential storage
- User login and authentication
- Access control for protected API endpoints
- Association of user activity with authenticated user accounts

Backend Components

- Django authentication mechanisms
- Django REST Framework views for authentication-related endpoints
- MySQL database for storing user credentials and related structured data

Frontend Components

- Login and registration pages
- Axios-based API calls for authentication requests
- Session handling on the client side

Data Flow

1. User submits login or registration data via the frontend.
2. Backend validates and processes credentials.
3. User information is stored or verified against MySQL.
4. Authenticated access is granted to subsequent API requests.

News Fetching Module

Description

The News Fetching Module is responsible for retrieving real-time news articles from external sources. It serves as the primary data ingestion layer of the system.

Key Functionalities

- Fetching live news data from the WorldNews API
- Handling API authentication using a secure API key
- Processing and normalizing external news data

Backend Components

- Django service or utility functions to interact with the WorldNews API
- Environment-based configuration for API keys
- Error handling for failed or invalid API responses

Data Flow

1. Backend sends a request to the WorldNews API.
2. News data is returned in structured JSON format.
3. Backend processes and prepares data for frontend delivery.

AI-Based Summarization Module

Description

This module implements the intelligence layer of **newsSum**. It generates abstractive summaries of news articles using advanced NLP techniques.

Key Functionalities

- Integration with Hugging Face Inference API
- Abstractive summarization of long news articles
- Handling AI inference responses and formatting summaries

Backend Components

- API client logic for Hugging Face Inference API
- Configuration using HF_API_KEY and HF_MODEL
- Summarization orchestration within Django services

AI Model Used

- **Model Name**: google/pegasus-xsum
- **Purpose**: Abstractive summarization of news content
- **Output**: Concise, coherent summaries preserving original context

Process Flow

1. Backend receives article content for summarization.
2. Content is sent to Hugging Face API.
3. The model generates an abstractive summary.
4. Summary is returned to backend for storage and response.

News and Summary Management Module

Description

This module manages the persistence and retrieval of news articles and AI-generated summaries. It ensures that summarized content can be reused and accessed efficiently.

Key Functionalities

- Storing summaries in MongoDB
- Associating summaries with specific users
- Retrieving stored summaries on demand

Backend Components

- MongoDB Atlas integration
- Data models for summaries
- CRUD operations through Django services

Data Storage

- **Database**: MongoDB Atlas
- **Data Types**:
    - News article metadata
    - Full article content
    - Generated summaries

Frontend User Interface Module

Description

The Frontend UI Module is responsible for presenting news content and summaries to the user in an intuitive and responsive manner.

Key Functionalities

- Displaying lists of news articles
- Viewing full articles and summaries
- Triggering summarization requests
- Managing user interactions and navigation

Frontend Components

- React pages and reusable components
- Axios service layer for backend communication
- Global styling and layout configuration

User Flow

Login → View News → Select Article → Generate Summary → Read Summary

Configuration and Utility Module

Description

This module handles environment configuration and cross-cutting utilities required across the system.

Key Functionalities

- Managing environment variables
- Configuring CORS settings
- Centralized error handling and logging
- API base URL and request configuration

Components

- .env files for frontend and backend
- Django settings for CORS and database connections
- Axios configuration for API calls

Inter-Module Interaction Summary

Module	Depends On	Provides
Authentication	Database	Secure access control
News Fetching	External News API	Real-time news data
AI Summarization	Hugging Face API	Generated summaries
Frontend UI	Backend APIs	User interaction layer
Configuration	All Modules	Secure and flexible setup

Summary

The modular architecture of **newsSum** enables clear separation between authentication, news ingestion, AI summarization, data management, and user interaction. Each module is designed to operate independently while contributing to a cohesive and scalable system. This approach improves maintainability, simplifies debugging, and prepares the application for future enhancements

8. User Interface (UI) Design and Screenshots

Overview

The User Interface (UI) of **newsSum** is designed with a strong focus on **simplicity, clarity, and usability**, ensuring that users can consume news and access AI-generated summaries with minimal effort. The frontend is implemented using **React.js with Vite**, enabling a responsive, fast, and dynamic single-page application experience.

The UI design aligns closely with the system's functional modules, guiding users smoothly through authentication, news browsing, and summarization workflows. Emphasis is placed on readability, intuitive navigation, and clean presentation of content.

Design Principles Followed

The following design principles guided the development of the newsSum user interface:

- **Minimalistic Layout**: Avoids unnecessary visual clutter to keep focus on news content and summaries.
- **Clear Navigation Flow**: Logical transitions between pages and actions reduce cognitive load.
- **Responsiveness**: Ensures usability across desktops, laptops, and smaller screen devices.
- **Consistency**: Uniform styling, fonts, and layout patterns across all pages.
- **User Feedback**: Visual cues for loading states, errors, and successful actions.

Authentication Screens

Login Page

Purpose

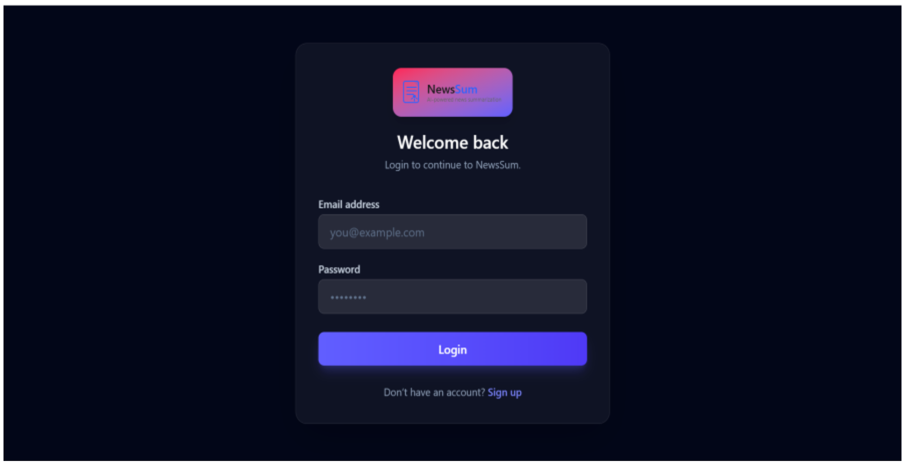
Allows registered users to authenticate and access the system.

UI Features

- Input fields for user credentials
- Clear call-to-action for login
- Validation feedback for incorrect or missing inputs
- Seamless redirection upon successful login

User Flow

Login → Authentication → Access to News Dashboard



Registration Page

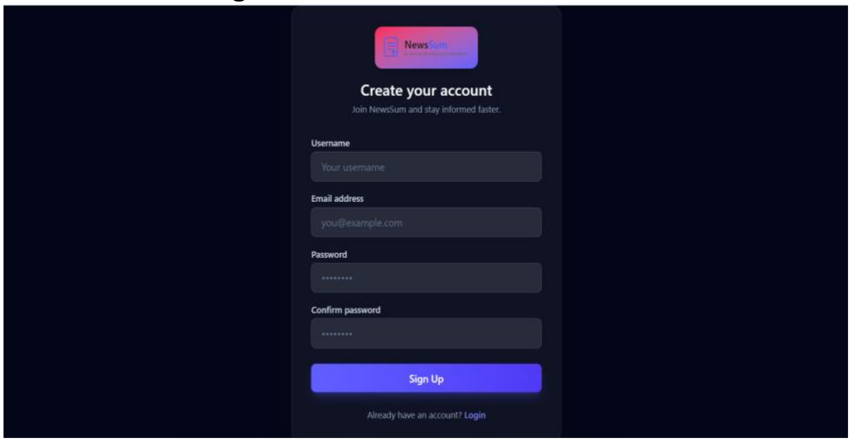
Purpose

Enables new users to create an account and gain access to newsSum features.

UI Features

- Form fields for required user information
- Client-side validation for input correctness
- Feedback messages for successful or failed registration

- Navigation option to return to login


Create your account

Join NewsSum and stay informed faster.

Username: Your username

Email address: you@example.com

Password: ••••••••

Confirm password: ••••••••

Sign Up

Already have an account? Login

News Dashboard / Home Interface

Purpose

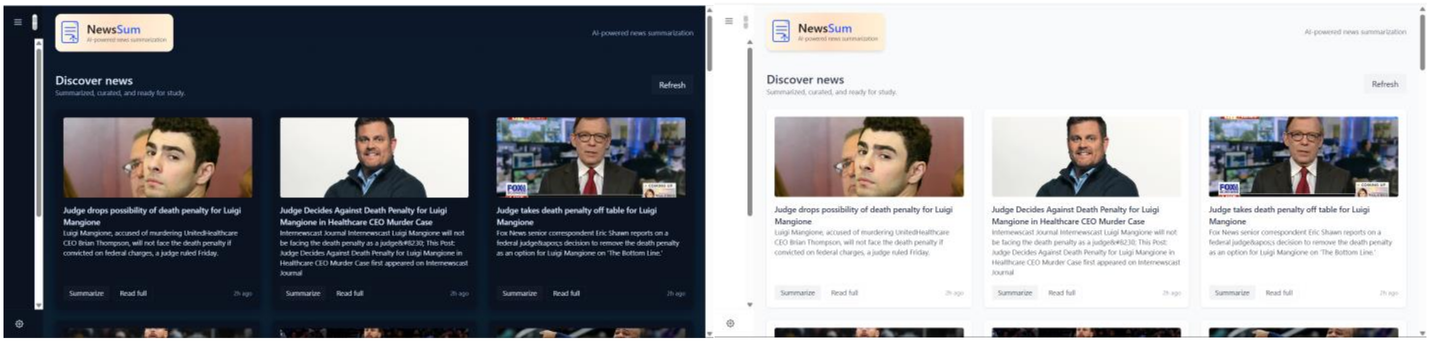
Acts as the central interface where users can browse available news articles fetched from external sources.

UI Features

- Structured list or feed of news articles
- Display of essential article metadata such as title and source
- Clickable articles for detailed viewing
- Clean layout optimized for reading and navigation

User Interaction

- Users scroll through news items
- Select an article to view detailed content
- Trigger summarization when needed


NewsSum
AI-powered news summarization

AI-powered news summarization

Discover news

Summarized, curated, and ready for study.

Refresh

Judge drops possibility of death penalty for Luigi Mangione
Luigi Mangione, accused of murdering UnitedHealthcare CEO Brian Thompson, will not face the death penalty if convicted on federal charges, a judge ruled Friday.
Summarize Read full 2h ago

Judge Decides Against Death Penalty for Luigi Mangione in Healthcare CEO Murder Case
Internewscast Journal Internewscast Luigi Mangione will not be facing the death penalty as a judge&#8230; This Post: Judge Decides Against Death Penalty for Luigi Mangione in Healthcare CEO Murder Case first appeared on Internewscast Journal
Summarize Read full 2h ago

Judge takes death penalty off table for Luigi Mangione
Fox News senior correspondent Eric Shawn reports on a federal judge&apos;s decision to remove the death penalty as an option for Luigi Mangione on 'The Bottom Line.'
Summarize Read full 2h ago

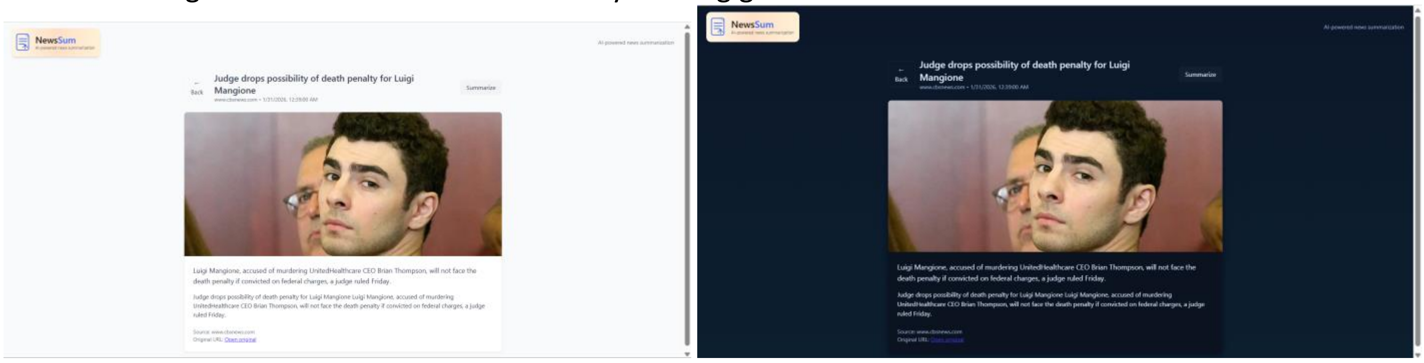
Article View

Purpose

Allows users to read full news articles and generate AI-powered summaries on demand.

UI Features

- Full article content displayed in a readable format
- Dedicated action (button or control) to generate a summary
- Loading indicators while the AI summary is being generated


NewsSum
AI-powered news summarization

AI-powered news summarization

← Back

Judge drops possibility of death penalty for Luigi Mangione
www.cbsnews.com • 1/31/2026, 12:39:00 AM

Summarize

Luigi Mangione, accused of murdering UnitedHealthcare CEO Brian Thompson, will not face the death penalty if convicted on federal charges, a judge ruled Friday.

Judge drops possibility of death penalty for Luigi Mangione Luigi Mangione, accused of murdering UnitedHealthcare CEO Brian Thompson, will not face the death penalty if convicted on federal charges, a judge ruled Friday.

Source: www.cbsnews.com
Original URL: Open original

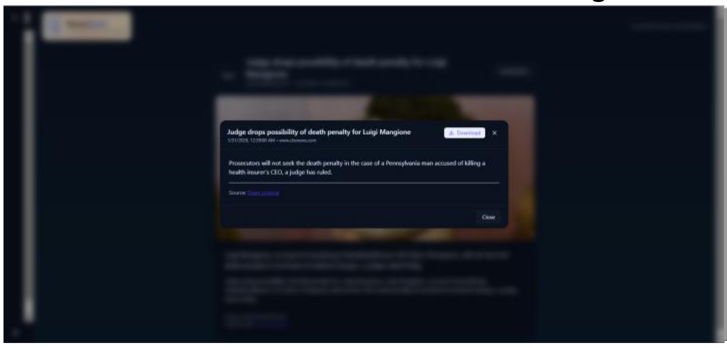
Summary Display Section

Purpose

Presents the AI-generated abstractive summary in a concise and readable manner.

UI Features

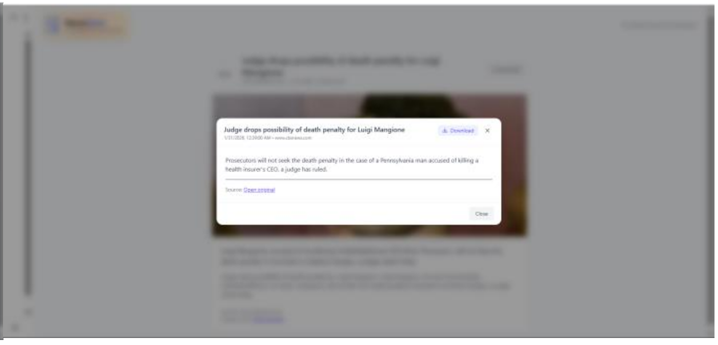
- Highlighted summary section distinct from article text
- Proper spacing and formatting for readability
- Immediate visibility after AI response is received
- Clear indication that the content is AI-generated



Judge drops possibility of death penalty for Luigi Mangione

Prosecutors will not seek the death penalty in the case of a Pennsylvania man accused of killing a health insurer's CEO, a judge has ruled.

Source: Open original



Judge drops possibility of death penalty for Luigi Mangione

Prosecutors will not seek the death penalty in the case of a Pennsylvania man accused of killing a health insurer's CEO, a judge has ruled.

Source: Open original

Responsive Design Considerations

The UI is designed to adapt gracefully across different screen sizes:

- **Desktop View**: Full-width layout with comfortable reading margins
- **Tablet View**: Adjusted spacing and stacking of content
- **Mobile View**: Vertical alignment and simplified navigation

CSS media queries and flexible layout structures ensure consistency across devices.

Error Handling and Feedback

The UI provides clear feedback for different system states:

- **Loading States**: Visible indicators during API calls and AI summarization
- **Error Messages**: Informative messages for network failures, invalid requests, or API errors
- **Success Feedback**: Confirmation of successful actions such as login or summary generation

This feedback mechanism improves transparency and user confidence while interacting with the system.

Summary of UI Flow

The overall UI interaction flow of newsSum can be summarized as:

Login / Register → News Dashboard → Select Article → Generate Summary → View Summary

This linear and intuitive flow ensures that users can access the system's core functionality with minimal learning effort.

Summary

The User Interface of **newsSum** successfully complements the system's AI-powered backend by providing a clean, responsive, and user-friendly frontend experience. Through thoughtful layout design, clear navigation, and effective feedback mechanisms, the UI enables users to interact seamlessly with news content and AI-generated summaries. The design ensures both functional efficiency and a positive user experience, making newsSum accessible and practical for everyday use.

9. Testing and Validation

Overview

Testing and validation were critical phases in the development of **newsSum – An AI-Powered News Summarization Platform**, ensuring that the system functions correctly, reliably, and consistently across all implemented modules. Given that newsSum integrates multiple components — a React-based frontend, a Django REST backend, relational and NoSQL databases, and external AI and news APIs — a comprehensive testing approach was adopted.

The testing process focused on validating:

- Correct functionality of backend REST APIs
- Proper integration between frontend, backend, databases, and external services
- Accurate AI-based summarization behavior
- Secure and reliable data handling
- Graceful error handling and system stability

Backend API testing was primarily conducted using **Insomnia**, while frontend behavior was validated through manual interaction and browser-based testing.

Testing Methodology

Type of Testing	Objective	Tools Used
API Testing	Validate correctness of backend REST APIs	Insomnia
Functional Testing	Ensure system features work as intended	Manual testing
Integration Testing	Verify interaction between frontend, backend, databases, and APIs	Manual + Insomnia
UI/UX Testing	Validate layout, navigation, and responsiveness	Browser DevTools
Error Handling Testing	Test system behavior for invalid inputs and failures	Insomnia + Manual

Backend API Testing (Insomnia Validation)

All backend endpoints developed using **Django REST Framework** were thoroughly tested using **Insomnia** during development and after feature completion.

Tested API Categories

Authentication APIs

- **User Registration API**
  - Tested for valid input, missing fields, and duplicate user scenarios
  - Verified correct data storage in MySQL
  - Confirmed appropriate success and error responses
- **User Login API**
  - Tested for valid and invalid credentials
  - Verified successful authentication and access control
  - Ensured unauthorized access attempts were rejected

News Fetching APIs

- Tested backend endpoints responsible for fetching news from the **WorldNews API**
- Verified:
  - Successful retrieval of news data
  - Proper handling of invalid API responses
  - Graceful failure when the external API is unavailable

AI Summarization APIs

- Tested summarization endpoints responsible for invoking the **Hugging Face Inference API**
- Verified:
  - Correct forwarding of article content to the AI model
  - Successful generation of abstractive summaries using google/pegasus-xsum
  - Proper response formatting and error handling
  - Handling of edge cases such as empty or excessively long input text

API Validation Results

Endpoint Category	Expected Behavior	Observed Result	Status
Authentication	Secure access and validation	Correct responses returned	Passed
News Fetching	Real-time news retrieval	Data fetched successfully	Passed
AI Summarization	Accurate summary generation	Coherent summaries returned	Passed
Error Handling	Graceful failure responses	Proper error messages returned	Passed

Integration Testing

Integration testing ensured that all system components worked together as a cohesive unit.

Scenarios Tested

1. **End-to-End News Summarization Flow**
   - User logs in → news fetched → article selected → summary generated
   - Verified correct data flow across frontend, backend, AI service, and database
2. **Database Integration**
   - Verified MySQL correctly stores and retrieves user-related data

    - Verified MongoDB correctly stores news articles and AI-generated summaries
3. **External API Dependency Handling**
    - Tested behavior when external APIs respond slowly or return errors
    - Confirmed backend handles failures without crashing

All integration scenarios executed successfully, confirming stable communication between system layers.

Frontend Functional and UI Testing

Frontend testing was conducted manually using the browser and developer tools.

Validated Areas

- Navigation flow between login, dashboard, article view, and summary view
- Correct rendering of news articles and summaries
- Proper triggering of backend API calls via Axios
- Loading indicators during API and AI processing
- Error messages displayed for failed requests

Responsiveness Testing

- UI tested across different screen sizes using browser developer tools
- Verified layout adapts correctly for desktop and smaller screens

Error Handling and Edge Case Testing

The system was tested against multiple edge cases, including:

- Invalid login credentials
- Missing or malformed API requests
- Empty or incomplete news article content
- External API downtime or delayed responses

In all cases, the system responded with meaningful error messages and maintained stability.

Validation Summary

Validation Area	**Status**	**Remarks**
Backend API correctness	Passed	Verified using Insomnia
AI summarization accuracy	Passed	Summaries coherent and context-aware
Database persistence	Passed	Data stored and retrieved correctly
Frontend-backend integration	Passed	Seamless API communication
UI responsiveness	Passed	Verified manually

Conclusion

The testing and validation phase confirmed that **newsSum** operates reliably across all implemented functionalities. Backend APIs behaved as expected under normal and edge-case conditions, AI-based summarization produced meaningful outputs, and frontend interactions remained smooth and responsive.
The use of **Insomnia for backend API testing** ensured precise validation of request–response behavior, while manual testing validated real-world user interaction flows.
Overall, the system meets its functional and quality expectations and is stable for real-world usage and further enhancement.

10. Results and Discussion

Overview

The **newsSum – AI-Powered News Summarization Platform** was successfully designed, developed, and tested as a full-stack web application integrating real-time news ingestion with AI-based abstractive summarization.
The system achieved its primary goal of reducing the effort and time required to consume lengthy news articles while preserving essential context and meaning.
This chapter discusses the **observed results**, **system behavior**, and **performance characteristics** of newsSum based on functional testing, integration testing, and real usage scenarios.

Functional Results

All core functionalities of newsSum were implemented and validated successfully. The system performed as expected across all major modules.

Functional Validation Summary

Feature	Expected Outcome	Observed Result	Status
User Authentication	Secure login and access control	Users authenticated successfully	Passed
News Fetching	Real-time news retrieval	News fetched consistently via WorldNews API	Passed
Article Display	Clear presentation of news content	Articles rendered correctly in UI	Passed
AI Summarization	Concise abstractive summaries	Meaningful summaries generated	Passed
Data Persistence	Storage of summaries	Data stored and retrieved reliably	Passed
Frontend Integration	Smooth UI interaction	Seamless navigation and updates	Passed

AI Summarization Results

The core intelligence of newsSum lies in its AI-powered summarization capability. The system uses the **Hugging Face Inference API** with the **google/pegasus-xsum** model, which is specifically optimized for news summarization.

Observed Behavior

- Generated summaries were **concise**, typically condensing long articles into a short paragraph.
- The summaries preserved:
    - The main topic of the article
    - Key events or outcomes
    - Contextual meaning without sentence-by-sentence extraction
- The abstractive nature of the model ensured that summaries were **rephrased and coherent**, rather than copied directly from the source text.

Discussion

Compared to extractive approaches, the abstractive summaries produced by Pegasus-XSum offered better readability and a more natural narrative flow. This validates the choice of model for news-centric summarization tasks.

Performance Evaluation

Response Time

- **News Fetching**: Articles were retrieved within a few seconds under normal network conditions.
- **AI Summarization**: Summary generation typically completed within **5–10 seconds**, depending on article length and Hugging Face API response time.

System Stability

- Backend services remained stable during repeated summarization requests.
- External API latency did not block unrelated backend operations.
- Database read/write operations performed reliably without data inconsistency.

Discussion

Performance was largely dependent on external services, particularly the AI inference API. However, by offloading summarization to a managed service, the system avoided local compute bottlenecks and remained responsive from a user perspective.

Frontend Behavior and User Experience

The React-based frontend demonstrated stable and predictable behavior:

- Smooth navigation between login, news listing, article view, and summary display
- Clear visual separation between original articles and AI-generated summaries
- Loading indicators during AI processing improved user awareness
- Error messages were displayed clearly when requests failed

Discussion
The frontend effectively complemented the backend intelligence by providing a clean and intuitive interface. This ensured that users could focus on consuming summarized information without being distracted by technical complexity.

Data Handling and Persistence Results

- **MySQL** reliably stored user authentication data with consistent access control.
- **MongoDB Atlas** efficiently stored AI-generated summaries

Discussion
The hybrid database approach proved effective, allowing structured and unstructured data to coexist without performance or maintenance issues. MongoDB's flexibility was particularly beneficial for storing varying article and summary lengths.

Limitations Observed

While the results were largely positive, a few limitations were observed:

- AI summarization quality is dependent on the clarity and structure of the source article.
- External API availability and rate limits can impact summarization speed.
- The system currently focuses on summarization and does not provide advanced analytics or personalization.

These limitations are inherent to the current scope and are addressed in the Future Enhancements.

Summary of Results

The results demonstrate that **newsSum successfully fulfills its intended purpose**:

- Automates news summarization using AI
- Reduces time required to consume news content
- Provides coherent, context-aware summaries
- Maintains stable performance and data integrity
- Offers a user-friendly and responsive interface

Conclusion of Discussion

The successful implementation and observed results validate the architectural and technological decisions made during the development of newsSum. By integrating real-time news APIs, transformer-based NLP models, and a modern full-stack architecture, the system effectively demonstrates how AI can enhance everyday information consumption. The discussion confirms that newsSum is functionally sound, technically robust, and well-positioned for future expansion.

11. Future Enhancements

Overview

While **newsSum** successfully delivers AI-powered news summarization in its current form, there is significant scope for enhancement and expansion. The existing architecture and modular design allow for future improvements without major restructuring. This chapter outlines potential enhancements that can further improve system performance, usability, intelligence, and scalability.
These enhancements are proposed based on the current implementation, observed limitations, and evolving user needs in modern news consumption platforms.

Advanced Personalization and User Preferences

- Introduce **user-specific preferences** such as preferred news categories (e.g., technology, finance, sports).
- Enable personalized news feeds based on user reading history and interaction patterns.
- Store preference metadata in the database to dynamically tailor news retrieval and summarization behavior.

Category-Based and Topic-Specific Summarization

- Allow users to filter and summarize news by categories or keywords.
- Enable summarization of multiple related articles into a **single consolidated summary** for broader topic understanding.
- Improve content discoverability by grouping similar news items.

Summary Length and Style Customization

- Provide options for users to choose:
  - Short summaries (quick overview)
  - Medium summaries (balanced detail)
  - Detailed summaries (in-depth understanding)
- Adjust summarization parameters dynamically before sending requests to the AI inference API.

Caching and Performance Optimization

- Implement caching mechanisms to store previously generated summaries.
- Avoid repeated AI inference calls for the same article, reducing latency and API usage.
- Introduce asynchronous processing or background tasks for summarization to improve frontend responsiveness.

Multi-Model AI Support

- Extend the system to support multiple summarization models in addition to google/pegasus-xsum.
- Allow experimentation with different transformer models to compare summary quality.
- Enable easy switching or upgrading of AI models without disrupting core functionality.

Improved Analytics and Insights

- Track summary usage metrics such as:
  - Number of summaries generated
  - Most summarized topics
  - Average reading time saved
- Display insights through dashboards to provide users with measurable value from the system.

Enhanced UI and Accessibility Features

- Improve UI accessibility by supporting:
  - Adjustable font sizes
  - Dark/light themes
  - Better contrast and readability
- Enhance mobile experience with optimized layouts and touch-friendly interactions.

Scalability and Deployment Enhancements

- Introduce containerization (e.g., Docker) for easier deployment and environment consistency.
- Add load balancing and monitoring tools to support higher traffic.
- Prepare the system for cloud-native scaling if user volume increases.

Security and Compliance Improvements

- Enhance authentication mechanisms with stronger session management.
- Add rate limiting to prevent abuse of summarization APIs.
- Improve audit logging for tracking user actions and system behavior.

Summary

The future enhancements outlined above highlight the strong extensibility of the **newsSum** platform. With its modular full-stack architecture and AI-driven core, the system can evolve into a more personalized, scalable, and feature-rich news intelligence platform. These enhancements provide a clear roadmap for future development while building upon the solid foundation established by the current implementation.

12. Conclusion

Summary

The **newsSum – AI-Powered News Summarization Platform** was successfully designed, developed, and validated as a full-stack web application during the internship tenure at **Condigence LLP**. The project demonstrates the practical application of **Artificial Intelligence and Natural Language Processing (NLP)** to solve the real-world problem of information overload in modern news consumption.

By integrating real-time news retrieval with transformer-based abstractive summarization, newsSum enables users to consume lengthy news articles efficiently without compromising on context or accuracy. The system effectively bridges the gap between raw news content and user-friendly information consumption.

What Was Achieved

The key achievements of the project include:

- Development of a **complete full-stack system** using React.js (Vite) for the frontend and Django with Django REST Framework for the backend.
- Successful integration of **WorldNews API** for real-time news ingestion.
- Implementation of **AI-based abstractive summarization** using the Hugging Face Inference API and the google/pegasus-xsum model.
- Adoption of a **hybrid database architecture**, utilizing MySQL for structured user data and MongoDB Atlas for unstructured news and summary data.
- Secure handling of sensitive configuration and credentials through environment variables.
- Thorough testing and validation of backend APIs using Insomnia and manual frontend testing.

These achievements collectively validate the technical soundness and functional completeness of the system.

Technical and Learning Outcomes

From a technical perspective, the project provided hands-on experience in:

- Designing RESTful APIs using Django REST Framework.
- Integrating third-party APIs and managed AI inference services.
- Managing structured and unstructured data using multiple databases.
- Building responsive and interactive user interfaces with React.
- Handling real-world concerns such as error handling, API latency, and system reliability.

From a learning standpoint, the project strengthened understanding of **full-stack architecture**, **AI integration in web applications**, and **scalable system design**, aligning closely with industry practices.

Project Significance

newsSum serves as a practical demonstration of how AI-powered systems can enhance everyday digital experiences. By automating the summarization process, the platform reduces cognitive load, saves time, and improves accessibility to information. The project highlights the effectiveness of abstractive NLP models in real-world applications beyond theoretical experimentation.

Final Remarks

In conclusion, **newsSum** successfully meets its intended objectives and stands as a robust, extensible, and production-ready prototype for intelligent news summarization. The modular design, clean architecture, and effective use of AI technologies position the system well for future enhancements and real-world adoption. This project represents a meaningful milestone in the internship journey and reflects the practical application of modern software engineering and artificial intelligence concepts.

13. References

1. **Django**
   Official Django Documentation – Used for backend development, request handling, ORM integration, and security features.
   https://docs.djangoproject.com/
2. **Django REST Framework**
   Django REST Framework Documentation – Used for building RESTful APIs, serializers, and structured API responses.
   https://www.django-rest-framework.org/
3. **React**
   React Documentation – Used for building the frontend user interface and managing component-based architecture.
   https://react.dev/
4. **Vite**
   Vite Documentation – Used as the frontend build tool and development server for fast bundling and hot module replacement.
   https://vitejs.dev/
5. **Axios**
   Axios Documentation – Used for handling frontend-to-backend HTTP requests and API communication.
   https://axios-http.com/
6. **MySQL**
   MySQL Documentation – Used for managing structured relational data related to user authentication.
   https://dev.mysql.com/doc/
7. **MongoDB**
   MongoDB Atlas Documentation – Used for storing unstructured data such as news articles and AI-generated summaries.
   https://www.mongodb.com/docs/
8. **WorldNews API**
   WorldNews API Documentation – Used to fetch real-time news articles from external sources.
   https://worldnewsapi.com/
9. **Hugging Face**
   Hugging Face Documentation – Used for AI model inference and NLP-based summarization.
   https://huggingface.co/docs
10. **Google Research**
    Pegasus-XSum Model Paper – Referenced for understanding abstractive summarization behavior of the google/pegasus-xsum model.
    https://arxiv.org/abs/1912.08777
11. **Insomnia**
    Insomnia Documentation – Used for manual testing and validation of backend REST APIs.
    https://docs.insomnia.rest/
12. **Mozilla Developer Network**
    MDN Web Docs – Used as a reference for JavaScript, HTTP concepts, and frontend best practices.
    https://developer.mozilla.org/