### **4+1 View Model for News Aggregation System (NAS)**

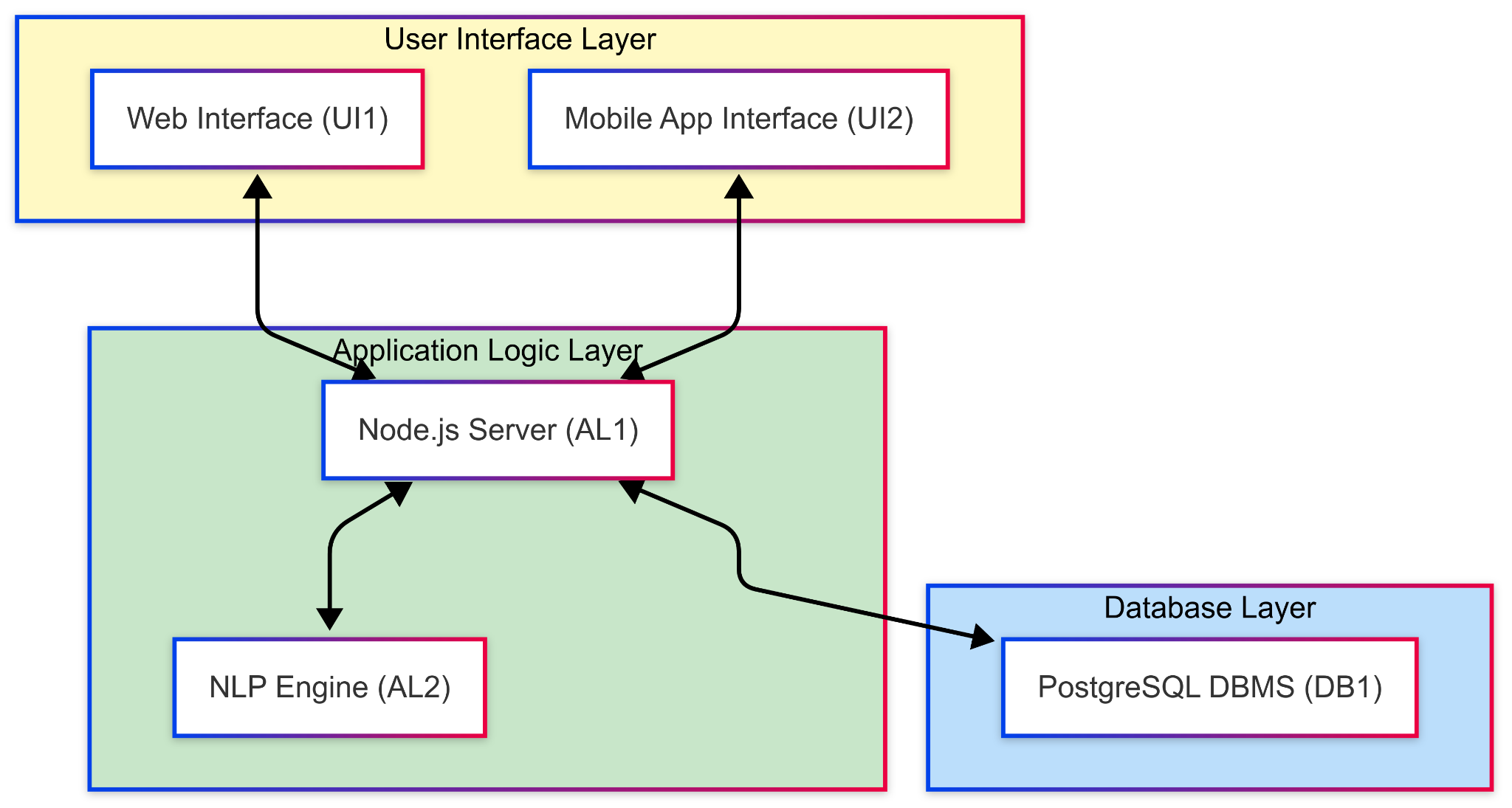
The **4+1 View Model** is a structured way to represent the architecture of the **News Aggregation System (NAS)**. It provides multiple perspectives (logical, process, development, physical) and scenarios to illustrate how the system is designed and how its components interact. Below is the detailed **4+1 View Model** for NAS:

#### 

#### **1. Logical View**

The **Logical View** describes the system's functionality and how it is organized into components. It focuses on **user interfaces**, **application logic**, and **data storage**.

* **User Interface (UI):**
  + **Web Interface (UI1):** A responsive web application for desktop and mobile browsers.
  + **Mobile App Interface (UI2):** Native mobile applications for iOS and Android.
* **Application Logic (AL):**
  + **Node.js Server (AL1):** Handles the core application logic, including news collection, categorization, sentiment analysis, and summary generation.
  + **NLP Engine (AL2):** External Natural Language Processing (NLP) service used for headline categorization, sentiment analysis, and summary generation.
* **Database (DB):**
  + **PostgreSQL DBMS (DB1):** Used for storing news articles, user data, sentiment scores, and system logs.

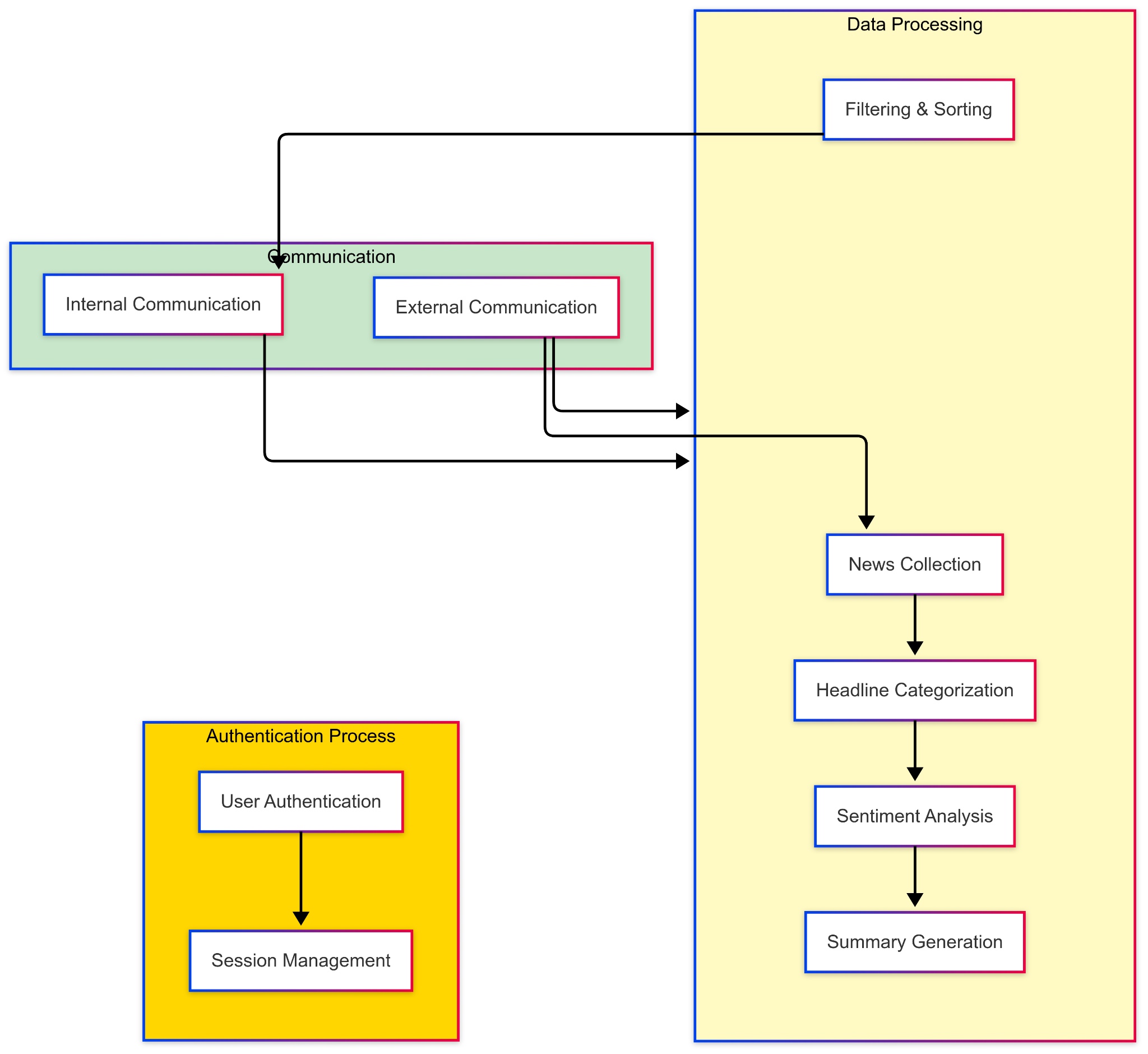


**2. Process View**

The **Process View** focuses on the dynamic aspects of the system, including **user authentication**, **data processing**, and **communication** between components.

* **User Authentication:**
  + Handles user login, access control, and session management.
* **Data Processing:**
  + **News Collection:** Fetches articles from APIs, RSS feeds, and web scraping tools.
  + **Headline Categorization:** Groups articles under unified incident-based headlines using NLP.
  + **Sentiment Analysis:** Analyzes the sentiment of news articles using NLP.
  + **Summary Generation:** Generates concise summaries of articles using NLP.
  + **Filtering and Sorting:** Processes user requests for filtering and sorting articles.
* **Communication:**
  + **Internal Communication:** Between the Node.js server, database, and NLP engine.
  + **External Communication:** With news APIs, RSS feeds, and web scraping tools.

#### 



#### 

#### **3. Development View**

The **Development View** describes the system from a developer's perspective, including the **programming languages**, **technology stack**, and **communication protocols**.

* **Programming Languages:**
  + **JavaScript:** Used for backend development with Node.js.
  + **Python:** Used for NLP tasks (e.g., sentiment analysis, summary generation).
  + **Dart:** Used for mobile app development with Flutter.
* **Technology Stack:**
  + **Backend:** Node.js, Express.js, PostgreSQL.
  + **Frontend:** React.js (for web), Flutter (for mobile).
  + **NLP:** Python with libraries like NLTK, SpaCy, or Hugging Face Transformers.
* **Communication Protocols:**
  + **HTTP/HTTPS:** For communication between the web/mobile interfaces and the backend.
  + **WebSocket:** For real-time updates.
  + **RESTful APIs:** For integrating with external news APIs and RSS feeds.

#### 

#### 

#### 

#### 

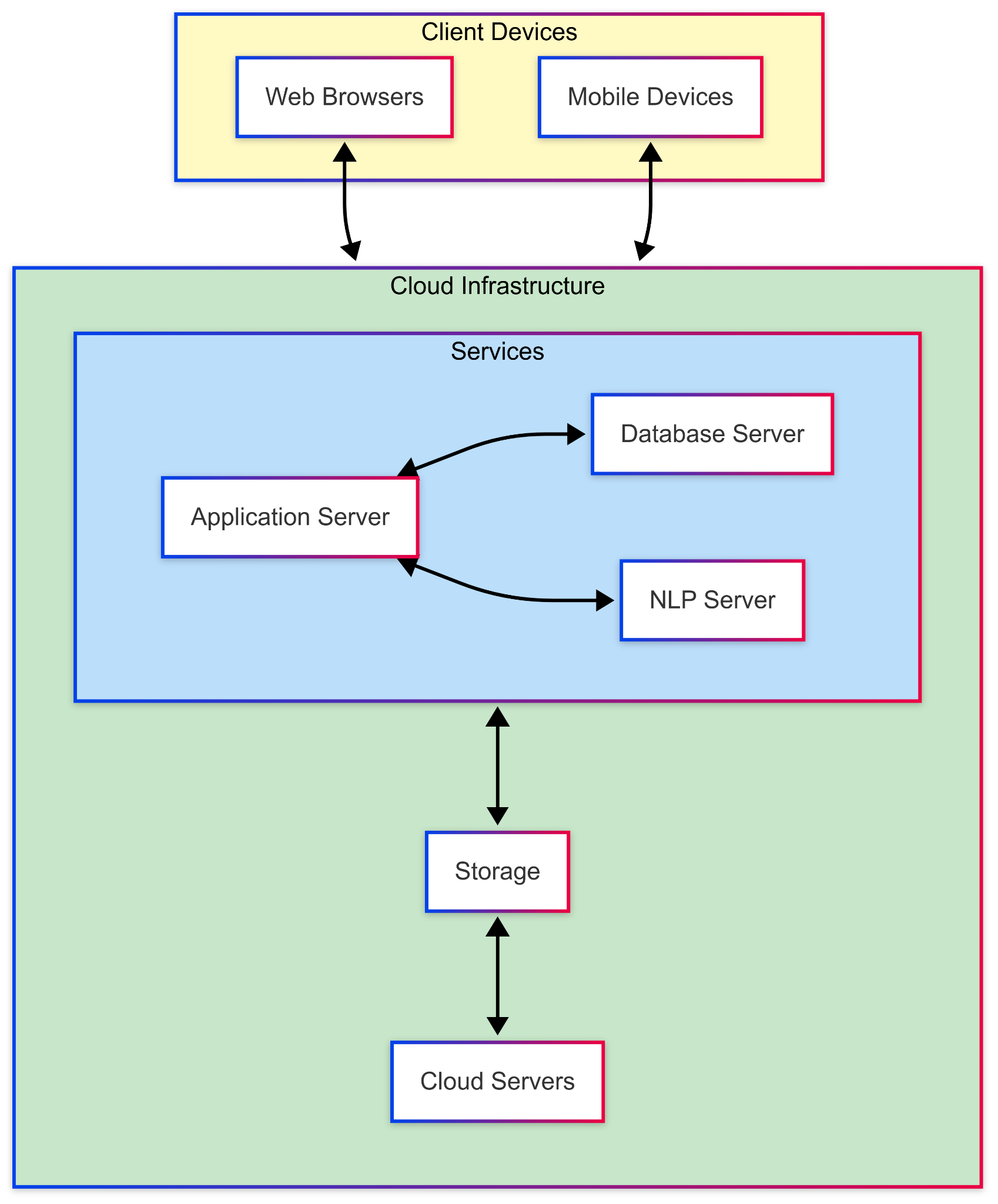
#### 

#### **4. Physical View**

The **Physical View** describes the hardware and infrastructure required to support the system.

* **Server Hardware:**
  + **Cloud Servers:** Hosted on platforms like AWS, Azure, or Google Cloud for scalability and reliability.
  + **Storage:** Cloud-based storage for news articles, user data, and system logs.
* **Client Devices:**
  + **Web Browsers:** Accessed via desktop and mobile browsers.
  + **Mobile Devices:** Accessed via native iOS and Android apps.

#### 



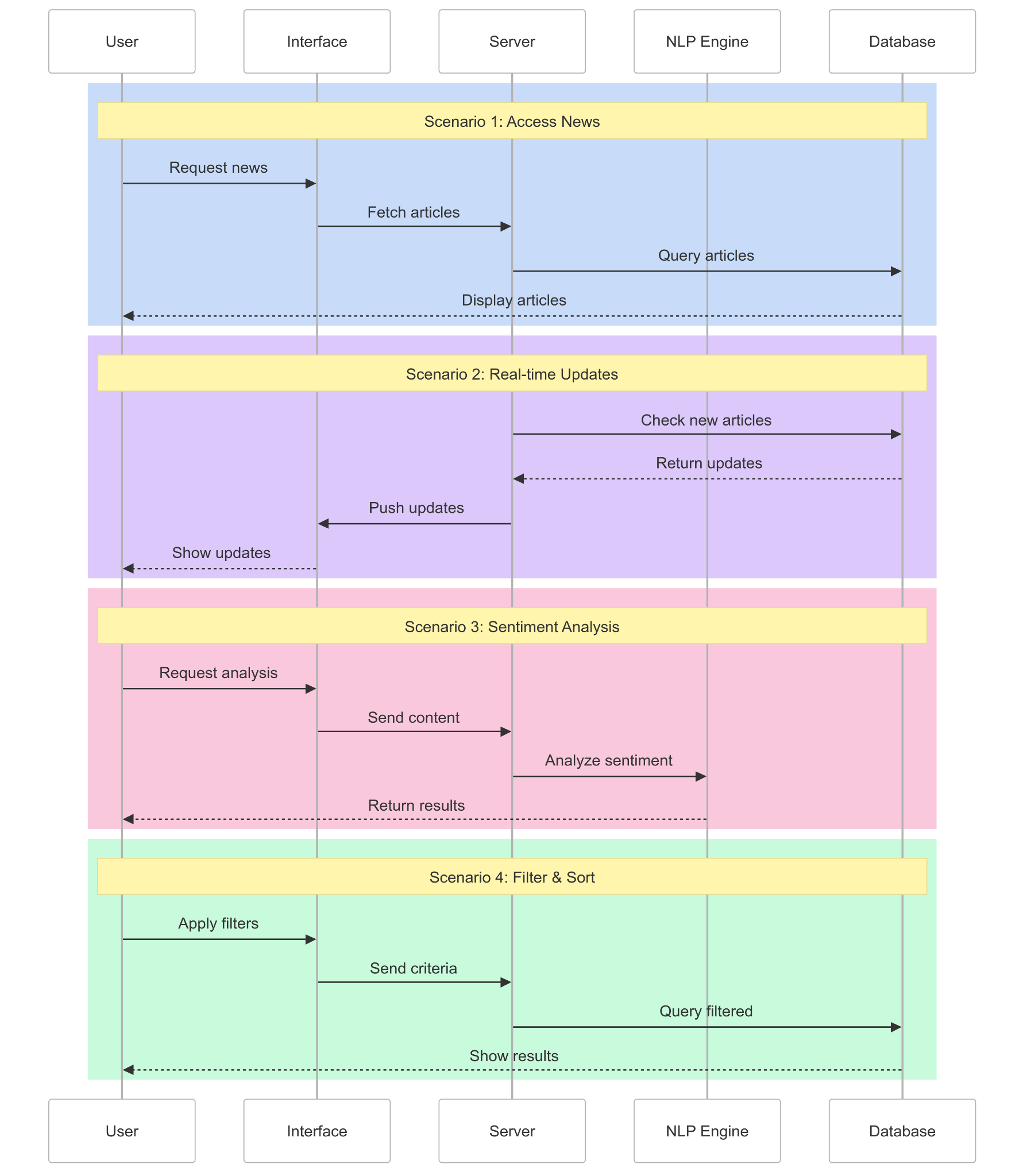
#### 

#### 

#### **5. Scenarios**

The **Scenarios** section provides example use cases to demonstrate how the components interact in different situations.

* **Scenario 1: User Accesses News Articles**
  + The user requests news updates via the web or mobile interface.
  + The Node.js server fetches articles from the database and displays them to the user.
* **Scenario 2: Real-Time News Updates**
  + The system periodically checks for new articles from APIs and RSS feeds.
  + New articles are categorized, analysed for sentiment, and stored in the database.
  + The user receives real-time updates via the web or mobile interface.
* **Scenario 3: Sentiment Analysis**
  + The user requests sentiment analysis for a specific news article.
  + The Node.js server sends the article content to the NLP engine.
  + The NLP engine analyses the sentiment and returns the results to the user.
* **Scenario 4: Filtering and Sorting**
  + The user applies filters (e.g., date, source) and sorting options (e.g., most recent, most relevant).
  + The Node.js server retrieves the filtered and sorted articles from the database and displays them to the user.



This **4+1 View Model** provides a comprehensive overview of the **News Aggregation System (NAS)** architecture, covering its logical, process, development, and physical aspects, along with key scenarios.