### **System Architecture Document - News Aggregation System (NAS)**

#### **1. Introduction**

The **News Aggregation System (NAS)** is designed to streamline the process of collecting, categorizing, and presenting news articles from multiple sources under a unified headline. The system leverages advanced technologies such as **Natural Language Processing (NLP)**, **web scraping**, and **real-time updates** to provide users with a comprehensive and multi-perspective view of global events. This document outlines the rationale for selecting the **Model-View-Controller (MVC)** architectural pattern for NAS and provides insights into its implementation.

#### **2. Rationale for Choosing MVC**

The **MVC (Model-View-Controller)** architectural pattern has been selected for the NAS project due to its proven benefits in managing complex systems with diverse functionalities. Below are the key reasons for choosing MVC:

##### **2.1 Separation of Concerns**

* **MVC** enforces a clear separation of concerns by dividing the system into three distinct components: **Model**, **View**, and **Controller**.
* In NAS, this separation allows for modular development, where the **Model** handles data processing, the **View** manages the user interface, and the **Controller** orchestrates the interaction between the two.
* This separation is particularly beneficial for NAS, as it involves multiple functionalities such as **news collection**, **headline categorization**, **sentiment analysis**, and **user engagement**.

##### **2.2 Scalability and Flexibility**

* The **MVC** pattern provides a flexible structure that supports scalability. As NAS is expected to grow over time, the ability to scale individual components (e.g., adding new news sources or improving NLP algorithms) without affecting the entire system is crucial.
* For example, the **model** can be scaled independently to handle larger volumes of news data, while the **View** can be updated to support new user interfaces (e.g., mobile apps).

##### **2.3 Reusability and Extensibility**

* The modular nature of **MVC** allows for the reuse of components across different parts of the application. In NAS, functionalities such as **user authentication**, **data filtering**, and **real-time updates** can be reused across multiple modules.
* This reusability reduces development time and effort, making it easier to extend the system with new features in the future.

##### **2.4 Clear User Interface Development**

* **MVC** provides a clear distinction between the **user interface (View)** and the **application logic (Controller)**. This separation simplifies the design and development of user interfaces, enabling web and mobile interfaces to be built separately while using the same application logic.
* In NAS, this is particularly valuable as the system needs to support multiple platforms (web, mobile) with consistent functionality.

#### 

#### **3. Implementation of MVC in NAS**

The implementation of the **MVC** architectural pattern in NAS will follow these principles:

##### 

##### **3.1 Models (Application Logic)**

* **Models** will encapsulate the core application logic, including **data handling**, **validation**, and **business rules**.
* In NAS, models will manage functionalities such as:
  + **News Collection**: Fetching articles from APIs, RSS feeds, and web scraping tools.
  + **Headline Categorization**: Grouping articles under unified incident-based headlines using NLP.
  + **Source Verification**: Verifying the credibility of news sources.
  + **Sentiment Analysis**: Analysing the sentiment of news articles using NLP.
  + **Summary Generation**: Generating concise summaries of articles.
* Models will interact with the **database** (e.g., PostgreSQL) for data storage and retrieval.

##### 

##### **3.2 Views (User Interface)**

* **Views** will handle the **presentation layer**, focusing on the user interface and user experience.
* Separate views will be developed for:
  + **Web Interface**: A responsive web application for desktop and mobile browsers.
  + **Mobile App Interface**: Native mobile applications for iOS and Android.
* Views will interact with the respective **controllers** to retrieve data and present it to users in an intuitive and accessible manner.

##### **3.3 Controllers (Application Logic)**

* **Controllers** will act as intermediaries between **models** and **views**, managing user interactions and application flow.
* In NAS, controllers will handle:
  + **User Authentication**: Managing user login and access control.
  + **Data Processing**: Coordinating the flow of data between models and views.
  + **Real-Time Updates**: Periodically checking for new articles and updating the system.
  + **Filtering and Sorting**: Processing user requests for filtering and sorting articles.
* The **Node.js** server will serve as the primary controller for NAS, handling requests and coordinating the application logic.

#### **4. Conclusion**

The selection of the **MVC architectural pattern** for the **News Aggregation System (NAS)** is based on its well-established benefits, such as **separation of concerns**, **scalability**, **reusability**, and **clear user interface development**. By implementing MVC by the project's specific requirements, we aim to develop a system that is **modular**, **maintainable**, and capable of accommodating future enhancements and changes. This architectural choice aligns with our objective of creating a **comprehensive news aggregation platform** that provides users with a **multi-perspective view of global events** while ensuring a seamless and intuitive user experience.

**MVC Model for NAS**

