# Phase 5: Build a User Interface with Streamlit

**Goal:**  
Create an interactive and user-friendly web interface using **Streamlit** to allow users to view trending news by date and category. The app integrates the entire ML pipeline — from fetching articles to summarizing and storing them — and displays trends in a clean, tabbed layout.

## Steps:

### **1. Setup and Streamlit Configuration**

Installed and configured Streamlit on the system. Set the app title, layout, and structure. Initialized the app with a clear title and headers.

### 2. **Integrated the Full Pipeline in the UI**

Replaced main.py by moving the entire automated pipeline logic into app.py. The app now automatically performs the following operations when first launched:

* Fetch fresh news articles from 3 APIs.
* Preprocess the articles (cleaning text).
* Detect trending topics per category using TF-IDF.
* Generate summaries for top-trending articles using the OpenAI API.
* Store the summarized articles into a PostgreSQL database.

### 3. **Loading and Status Display**

Used st.status() to visually show each step of the pipeline with real-time progress messages. Once all steps are completed, a final success message is displayed. This only runs once per session.

4. Dynamic Date Selection

Added a st.date\_input() at the top of the UI that allows users to select any date. When the date is changed, the UI updates immediately without needing a manual refresh or button click.

5. Tabbed Category View

Used Streamlit's st.tabs() to create one tab per category (Business, Sports, Technology, Entertainment, Health). Each tab automatically loads and displays articles for the selected date and category.

6. Database Query Integration

Integrated the read\_trends\_from\_database() function to load relevant articles from the PostgreSQL database based on the selected date and tab category.

### **7. Refactored Code Structure**

Moved all database reading logic into write\_read\_database.py for better code separation.

Kept app.py focused on UI rendering and user interaction.