Forethinker

The Future of News Delivery: Integrating AI, Big Data, and Cloud Services to Revolutionize Journalism

Abijith Trichur Ramachandran[1], Tyler Cranmer[1], Jayant Duneja[1], Max Zhou[1], Julian Bouchard[1], Rahul AlGahtani[1], Jeff Jones[2], Laura Yan[2], Claire Holman[2], Mehak Gupta[2], Anubhav Dubey[2], Kanishka Ghodke[2], Annika Chen[2], and Cinthia Reynoso[3]

[1]College of Engineering and Applied Sciences - Masters of Computer Science
[2]Leeds School of Business - Masters of Science in Business Analytics
[3]Leeds School of Business - Masters of Business Adminstration
[*]Advisor: Lucas Thelosen

April 26, 2024

Abstract

Forethinker is an advanced tool developed through a collaborative effort between graduate students at the University of Colorado, Boulder, and a Google Advisor. This application utilizes a sophisticated large language model for synthesizing and summarizing current news trends, specifically designed to support the informational needs of C-level executives. It strategically enhances decision-making capabilities by filtering and aggregating relevant news content tailored to industry-specific interests, thereby enabling executives to navigate the rapidly evolving business landscape effectively. Forethinker employs state-of-the-art natural language processing algorithms that adaptively refine output based on user interactions, thus providing personalized insights and facilitating streamlined access to critical news. This paper details the development process, architectural innovations, and the strategic implications of Forethinker, highlighting its significant utility in bridging the informational gap for high-level decision-makers in a concise manner.

1. Introduction

In the rapidly evolving business landscape, timely and relevant information is critical for strategic decision-making, especially at the executive level. The Forethinker application, developed through a partnership between graduate students at the University of Colorado, Boulder, and a Google Advisor, harnesses cutting-edge technology to address this need. This paper presents the architectural design, implementation challenges, and the operational efficiencies of Forethinker, which leverages large language models (LLMs) like OpenAI's GPT-4 for synthesizing and summarizing pertinent news content to bolster the strategic awareness of C-level executives.

2. Data Source

2.1 Overview

For this project, the Data Source team worked towards finding and connecting to relevant Data Sources that would ultimately feed into the product. We approached this task by starting with identifying Categories of data that might be valuable to CEOs.

Financial: Google Finance, Bloomberg Terminal

News: Google News, New York Times, Wall Street Journal, Economist, Politico

Social: X, Facebook, Instagram, LinkedIn, Reddit, Quora, TikTok, Medium, Netbase, Meltwater, Brandwatch, Hootsuite

Economic: World Bank, IMF Data, BEA

Consumer: Acxiom, Yodlee, MRI Simmons, Kantar, Nielsen, Google Trends, Google Insights Finder

2.2 Data Audit

Within each category, individual data sources were identified and audited with the goal of understanding information accessibility, refresh rates, data structures, potential costs, and technical feasibility. In total, there were 29 data sources added to the review-pipeline while prioritizing News and Social sources as these were hypothesized to be most relevant for the product. Over the course of the 2 months, 10 data sources were appropriately audited while work continued pursuing an MVP.

2.3 Lessons Learned

2.3.1 Social

Two approaches were considered for accessing information from social platforms, vendor aggregators vs direct from platforms. Vendor Aggregators, such as Netbase Quid and Meltwater accept boolean keyword queries and return publicly available content across sites such as X, Facebook, Reddit as well as Blogs, Tumblr, Editorial sites. The aggregated access came at a significant cost, finding reviews and estimates between $13k-$23k per year. Alternatively, in reviewing direct data pipelines, the main benefit was in reducing financial costs at the expense of development resources and capabilities. It is much more labor intensive to connect to data sources as each platform has varying pricing structures, endpoints, and processes. For example, LinkedIn requires the configuration of a Business Page and application approval while TikTok is only available to research partners.

2.3.2 News

News Sources were found to be more easily accessible with multiple free options. Similar to social media, there was a news aggregator considered (Google News) as well as direct connections (New York Times, Wall Street Journal). The pipelines most easily configurable were using the Google News API and New York Times API, both of which were available at no cost. New York Times has 10 different endpoints, 7 of which are still supported, with successful data extractions were made with 3 of them.

2.3.3 Development

For any development work, Jupyter Notebook files were created using Google Colab, a collaborative platform with easy access to libraries and computational resources. Python packages google.cloud and pandas_gbq were found to be very helpful in passing data securely and efficiently to BigQuery. This data warehousing solution served as a holding place to set the stage for further analysis and insights to be drawn from the collected information. With Google's comprehensive cloud ecosystem our first product iteration was successfully managed in one place.

2.4 Future Considerations

This first iteration product was successfully completed using entirely free versions and sources. There were many things found to consider for future iterations when scaling.

Data Sources

User Research Interviews suggest that start-up CEOs are more interested in understanding information related to operational efficiencies than publicly facing news. This leads us to believe sources such as bug tracking and support-ticketing systems would be valuable data sources to feed into the pipeline. The expectation is that solutions to current problems can often be found buried in archives of previous projects. In the case specific solutions do not directly apply to updated issues, potential retrieval from these sources could be domain knowledge experts that have resolved similar issues.

API Limits and Computational Resources

The New York Times API provided us access to historical article data (url, section, keywords, title, abstract, story facets, and others) with no defined limits. This was sufficient in allowing our team to design the data pipeline and structure a database. To improve model performance, the expectation is to vectorize more than just the abstract and keywords, and to include the full article content. Doing so would need to utilize paid API Tiers, and consider API call limits which vary across sources.

Vertex AI

When integrating development notebooks from Google Colab with BigQuery, notebooks were passed through a third cloud product, VertexAI. With this service being used primarily as a repository for notebooks to be passed to the BigQuery interface, many more features were found that could help with future builds. VertexAI is a machine learning platform that lets you train models, build applications, and customize large-language models for AI Applications. This platform also has an AI Agent development suite that is low-code friendly. While this iteration had already started custom architecture and development, a brief integration with VertexAI was a successful introduction to an all-in-one solution to be considered for future builds.

3. Backend

The backend architecture consists of four FastAPI microservices. The `user_service` hosts the Forethinker user backend application, the `news_service` is responsible for aggregating and processing news articles, the `social_media_service` handles the sourcing of tweets that correlate with news articles, and the `authentication_service` manages user credentials and controls access to the other backend services. In this paper, we will discuss all services that interact with Large Language Models (LLMs).

3.1 News Service

The news service workflow consists of eight submodules, each integral to the delivery of curated news content. The process initiates with the Scheduler, which activates the

News Controller, serving as the central command module that orchestrates the flow of operations based on predefined daily triggers. User interests are then inputted into the system, prompting the News Controller to interact with the New Article Sourcer. This submodule is tasked with fetching pertinent articles from established news providers such as The New York Times (NYT) and Google News, ensuring comprehensive collection of current news stories based on a user's interest.

Once the articles are sourced, they are processed by each of the news article sub routines. The News AI Agent(s) takes control and utilizes LangChain and OpenAI's GPT-4 to perform advanced AI subroutines, including summarization, sentiment analysis, article classification and Named Entity Recognition (NER) tagging. The processed data are stored in two distinct repositories for different purposes: BigQuery archives the article content along with the summaries and any additional enriched data, whereas Pinecone is utilized to manage the vector representations of the article content and summaries, optimizing for efficient retrieval and similarity searches. Figure 1

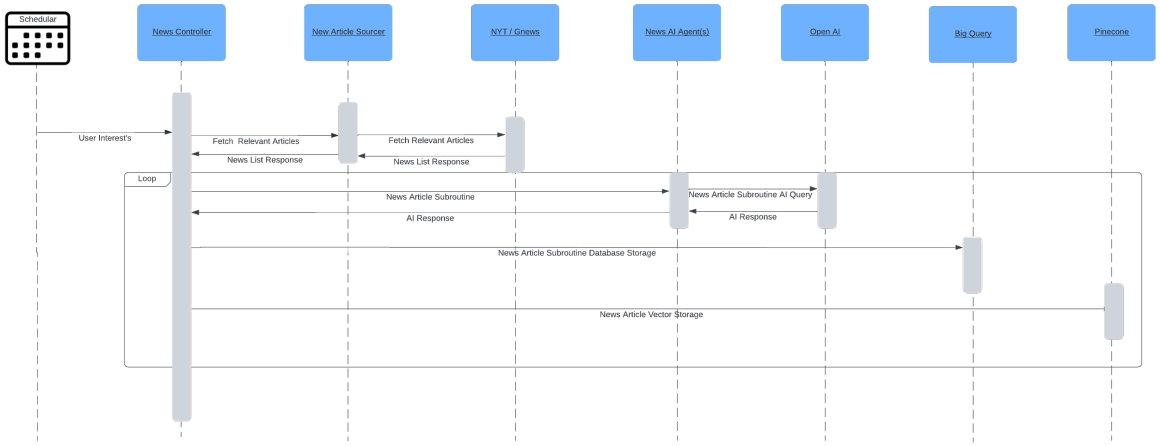

Figure 1: News Service Sequence Diagram

3.1.1 AI Agent

The News AI agent was designed to process articles from the NYT and Google News APIs. Its primary function is to condense news articles into concise summaries, enabling readers to grasp the key points in less than 30 seconds. Additionally, the system includes subroutines that collect metadata for each article, such as sentiment analysis, classification into categories like business or technology, and Named Entity Recognition (NER) for extracting company names. After a user's news article was processed the article content and summary was stored in a Pinecone vector database for future retrieval in the application.

Architecture for the agent was developed using LangChain's LCEL (LangChain Expression Language), OpenAI's GPT-4 model, BigQuery and Pinecone's vector database. These technologies are considered cutting-edge in their respective fields. LCEL facilitated

the seamless composition of multiple LLM chains, offering extensive tools for integrating with the OpenAI GPT-4 API and connecting with Pinecone's vector database. The combination of these technologies was selected for their capabilities of handling AI-driven tasks efficiently.

3.1.2 Challenges

AI Agent Architecture

The development of the AI News Agent involved several architectural revisions to address the variability in OpenAI LLM's generative outputs. Initially, the architecture featured a solitary AI Agent responsible for orchestrating and executing various news processing subroutines. At the beginning of each process, a prompt would guide the AI by specifying the subroutine to execute, leveraging LangChain's Pydantic OpenAI functions. This setup allowed the AI to internally reason and determine the appropriate function to invoke.

During our testing phase, we found that having the AI agent autonomously deciding which function to invoke based on input data achieved an 80-90% reliability rate. However, this also led to a 10-20% incidence rate where incorrect JSON outputs from the LLM would disrupt the subsequent data processing submodules, causing errors. This level of uncertainty was problematic for our production-level application, which necessitated 100% reliability in code execution. The risk of the AI selecting an incorrect function and the consequent disruption to the logic of downstream AI News subroutines drove us to seek a more reliable design approach that could guarantee deterministic results with every subroutine call.

The original architecture also employed a single invocation of the LLM chain across all subroutines, meaning that the same LLM token context window was applied to every task for an article. This approach inadvertently allowed the context window of OpenAI's LLM to accumulate prompts and content from each article being processed. We theorize that this aggregation of data within the LLM's context may have contributed to inconsistencies in output and erroneous function calls, as it overwhelmed the LLM with too much information, leading to potential confusion in processing subsequent articles

We revised the News AI architecture to include separate AI chains, each designated for specific subroutines. The News Controller now selectively triggers the appropriate AI chain according to the task at hand. Not including the custom prompts, each individual AI chain shares a common architecture, incorporating LCEL, OpenAI's GPT-4 chat model, Pydantic OpenAI functions, and a Pydantic Output Parser. This design ensures consistency in processing while providing the precision required for each unique subroutine.

Real-Time News Data Acquisition

Securing real-time access to news articles via company APIs presents numerous obstacles.

Major news sources like The Wall Street Journal (WSJ) and The New York Times (NYT) provide limited API features for real-time news, which hampers our ability to gather timely and comprehensive data. Additionally, the costs associated with accessing the necessary level of data from these sources are substantially high. The free API tiers offered by companies such as New York Times and Wall Streets Journal provide only limited access, falling short of meeting our requirements for detailed article content. Furthermore, potential partnerships with these media giants for enhanced data access prove to be prohibitively expensive, adding to the challenges faced in data aggregation and evaluation.

Human Evaluation of LLM Outputs
Currently, evaluating the answers remains our most significant hurdle. We require human intervention to assess the outputs generated for individual users, which demands some level of business acumen and time. Our system actively records data in the database, but human reviewers are essential to scrutinize and annotate the performance of each summary, sentiment analysis, NER, and article classification produced by the LLMs. This human evaluation is critical to ensuring the accuracy and relevance of our AI-generated content.

3.1.3 Improvements

Advanced RAG Techniques
To improve the relevance and precision of the content provided to users, it is crucial to delve deeper into advanced Retrieval-Augmented Generation (RAG) techniques. These techniques can significantly enhance the capability of our system to retrieve articles that closely match user preferences. By refining our methods in similarity retrieval, we can ensure that users receive the most pertinent and engaging content tailored to their interests.

AI Agent Architecture
There is a need to devote additional resources to the research and development of more advanced architectures for our LLM chains within AI agents. A more advanced architectural approach will contribute to building a more robust and resilient system. Emphasizing fault tolerance and system stability in our AI agent designs will help mitigate potential errors and enhance the overall reliability and efficiency of our service.

3.1.4 Successes

Langchain Integration: Langchain has significantly simplified the process of building applications with LLMs by providing a comprehensive framework that includes chains, agents, and retrieval augmented generation (RAG). This framework facilitates complex tasks involving sequences of LLM calls and interactions with external data sources, thereby enhancing the LLM's understanding and use of data not available during its training.

LCEL (LangChain Expression Language): The use of LCEL provided an easy way to compose multiple LLM chains together. LCEL offered various debugging tools for reading the complete chain input and output during execution.

Controlling LLM function calls with Pydantic: A notable success was the use of Pydantic to control the output of LLMs. By defining a Pydantic class, it became possible to specify the desired structure and content of LLM outputs, leading to more reliable and predictable results. Pydantic is a data validation library for Python, but what makes Pydantic so powerful in this setting is how LangChain has integrated it into validating prompts and output from large language models.

GPT-3.5 vs GPT-4: News summarization was successfully achieved using both the GPT-3.5 and GPT-4 models. We observed no significant differences in the quality of summaries produced by these two models.

Vector Database: Setting up vector databases and querying involves using semantic search, which operates with vectors through a similarity metric, such as cosine similarity. The issue at hand is that the method does not truly understand the query or what has been stored in the database; it simply converts text to vectors and compares them. Initially, we planned to convert an entire article into a single vector to determine which was most similar to the query vector we created. However, this posed a challenge as articles are long texts and queries are typically short sentences, making the comparison suboptimal. To address this, we chunked the article into smaller parts and implemented additional metadata filtering. We developed two methods for working with these chunks: the first method retrieves the top 50 results and sorts them based on the number of chunk matches for each article; the second method sorts articles by the highest score any of its chunks received, without considering how many chunks matched. Ultimately, a combination of these two methods proved most effective.

For metadata filtering, we perform Named Entity Recognition (NER) to capture the names of companies mentioned in the article before querying. This ensures that searches are checked against these company names in the metadata to confirm their mention in the article. Surprisingly, NER performed well, even though we used a decoder-only GPT model, which was unexpected because NER tasks are traditionally suited for encoder-decoder models. This effectiveness highlights the adaptability and potential of the models we are working with.

3.2 Social Media Service

Our decision to leverage Twitter for this proof of concept (PoC) is a strategic one, capitalizing on the platform's distinctive strengths. Twitter's vast user base actively generates a real-time pulse of information, making it a rich data source for various research and development endeavors. Twitter is considered as a global public space, hosting conver-

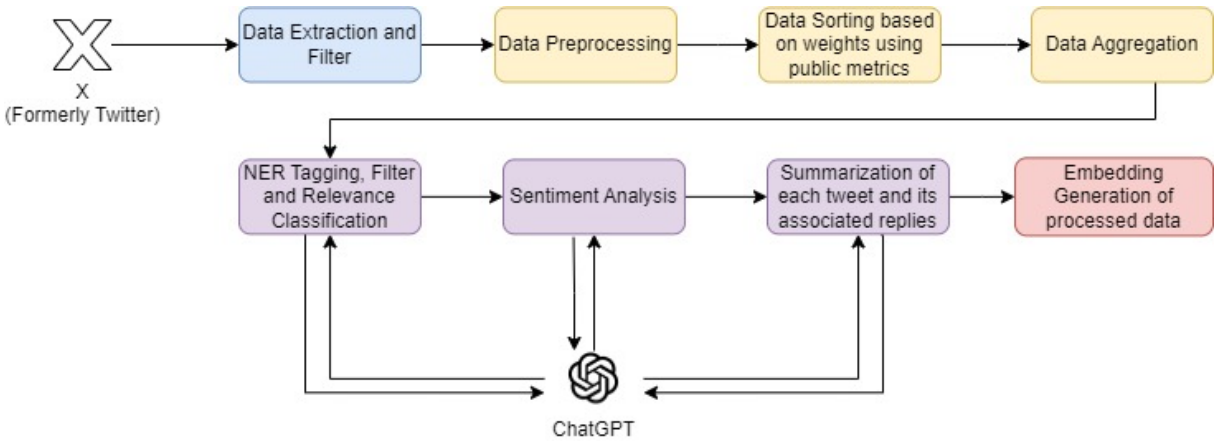

Figure 2: News Service Sequence Diagram

sations on a wide range of topics, from breaking news and entertainment to technology, finance and politics. Its real-time nature allows users to share and receive updates instantaneously, making it a rich source of current and diverse information. For a proof of concept aiming to demonstrate feasibility, gather insights, or test algorithms, Twitter provides a valuable dataset due to its:

Immediate Access to Trends and Events: Twitter's real-time feed offers instant access to emerging trends, news, and public opinion, providing a timely snapshot of global discourse.

Diverse User Base: With millions of active users worldwide, Twitter encompasses a broad spectrum of perspectives, making it a comprehensive source for gauging public sentiment or analyzing behaviors across different demographics.

Richness of Data: Tweets can contain text, links, images, and videos, offering multidimensional data points for analysis. This variety supports complex analyses, such as sentiment analysis, trend prediction, or event detection.

Public API Access: Twitter's API facilitates the extraction of large amounts of data for analysis, allowing researchers and developers to query specific topics, hashtags, or user activities

Despite the strengths of the various strengths stated there are several challenges when working with Twitter provided in the following section.

3.2.1 Challenges

Cost of Queries and Rate Limits: Accessing social media APIs, such as Twitter, often comes with associated costs and rate limitations. These platforms typically have tiered pricing models, where increased access or higher limits equate to higher costs. Rate limits can significantly slow down data collection, necessitating efficient query management and potentially leading to increased expenses to meet project requirements. As we were using

the free tier, to retrieve data it became a limiting factor to scale.

Relationship in Time Frame with News Articles: Establishing a temporal relationship between tweets and news articles is challenging. News events are rapidly reported on Twitter, but the volume and velocity of tweets can make it difficult to accurately align them with specific news articles' publication times. There are also cases where tweets may talk about an event that has happened in the late past and has now gained traction whereas there is no current news article that matches it. Another interesting case is some tweets predict data or events that further become a sensation in the news which is commonly found in unveiling of the next CPU or GPU for example. Creating a temporal connection and providing data or associated articles to the users preference can be challenging.

The Weighting Function on Identifying Valuable Data: Determining the relevance and value of social media content is inherently challenging due to the vast amount of noise in the data. Developing a weighting function that can effectively identify valuable data involves complex algorithms that can analyze sentiment, credibility, influence of the source, and the content's context. This is crucial for filtering out irrelevant or misleading information and focusing on data that genuinely contributes to understanding the news event.

Data Volume and Velocity: Twitter generates a massive amount of data daily. Processing this data in real-time to parse relevant information and associate it with news articles from established sources can be technically challenging and resource-intensive.

Bias and Misinformation: Social media platforms, including Twitter, can be sources of bias and misinformation. When associating tweets with news content, there's a risk of amplifying incorrect information or biased perspectives, making it critical to implement checks for source credibility and factuality.

Evolution of Language and Topics: The language and topics trending on Twitter can evolve rapidly. Keeping up with these changes, especially when trying to match tweets to news topics, requires continuous updating of NLP models and algorithms to ensure relevance and accuracy.

3.2.2 Improvements

Refined Data Collection: By optimizing query parameters and focusing on specific keywords, hashtags, and user profiles, we managed to reduce the noise in the collected data, leading to more relevant datasets for analysis.

Advanced NLP Techniques: We implemented more sophisticated NLP models for sentiment analysis, entity recognition, and categorization, improving the accuracy and

reliability of our insights.

Efficient API Usage: Strategies were developed to circumvent rate limits and maximize the efficiency of our API usage, ensuring a steady flow of data within the constraints of Twitter's API.

3.2.3 Successes

Effective Sentiment Analysis: Utilizing ChatGPT, we successfully analyzed tweet sentiments and obtained a nuanced understanding of public opinion on various topics. This enabled us to gauge the emotional tone of conversations accurately.

High-Quality Data Filtering: By prioritizing replies based on public engagement metrics, we filtered out the most impactful responses, enriching our data quality and ensuring that our analyses were based on significant interactions.

Insightful NER Results: The application of spaCy for Named Entity Recognition (NER) proved fruitful. By extracting key entities from tweets and replies, we identified major themes, individuals, organizations, and products mentioned in discussions.

3.3 Lessons Learned

In the context of developing an application designed to aggregate data from various sources, including news outlets, social media platforms, and personal information, we are facing a significant challenge regarding the costs associated with accessing this data through established APIs. Unlike in the past, when many of these companies offered educational subscriptions or discounts, such options have become scarce following the rise of LLMs. As a result, the financial burden of procuring data from these sources has become a major consideration for our project.

Finding and utilizing advanced debugging tools is crucial for developing a production-level application that utilize LLMs. The recent launch of Langchain's new tool, Langsmith, offers a promising approach to overcome these challenges. Having access to such tools during development could have conserved time and effort in diagnosing the causes of incorrect outputs from our LLMs, rather than relying on guesswork. This access would have facilitated the development of a more robust system for iterative improvements.

The sheer volume of Twitter data presents both a challenge and an opportunity. Efficient data processing and management techniques are crucial for handling this volume effectively, enabling us to glean valuable insights from the vast streams of data. Strategies such as data partitioning, efficient querying, and scalable storage solutions are integral to managing the flow and storage of data efficiently.

The continuously evolving language and topics on platforms like Twitter underscore the need for adaptable and learning NLP models. These models must be capable of

keeping up with new trends, slang, and the dynamic nature of human communication on social media. Implementing models that can update their parameters in real-time or near-real-time and utilize ongoing learning processes is critical for maintaining the relevance and accuracy of the analysis.

4. Frontend

Front-end developers were responsible for developing a user-friendly, efficient, and intuitive front-end interface for interacting with and visualizing AI-generated data. The team developed a highly intuitive dashboard, login layout, and user-specific sections that personalize the professional and personal information displayed. This customization allows the application's LLM to deliver highly relevant content, significantly improving the users' professional and personal experience.

The team adopted an agile approach for this project. During the inception phase, the team conducted a competitive analysis across various data categories, including News, Social Media, and Economy. This analysis helped shape the initial concept. In the first sprint, the team outlined the application's core features: a sign-up page, profile page, interest page, central dashboard, recently viewed analytics, and a group discussion forum. In the second sprint, the frontend team designed and developed a Minimal Viable Product (MVP) website focusing primarily on the News data category. This sprint also involved collaboration with the Go-To-Market (GTM) team on the product's logo and slogan. By the third sprint, the frontend team had integrated the frontend interface with the backend systems. The frontend team continues to explore and enhance interfaces for other data categories.

4.1 Lessons from Development

Working as a startup with only one frontend engineer, we learned how crucial cross-functional collaboration is and how to optimize communication speeds between backend and frontend systems to enhance application responsiveness and user experience. Continuous monitoring and upgrading of these interactions are crucial.

The frontend team prioritized enhancing the interface speeds between backend and front-end systems. This improvement was crucial for reducing user friction and significantly enhancing satisfaction. Faster interfaces are instrumental in streamlining user interactions and improving the overall efficiency of the application.

4.2 Lessons from Designing

A unified design scheme across all platforms, including mobile apps and websites, emerged as a key learning point. Consistency in design is essential for maintaining a coherent user experience and building a reliable, intuitive user interface.

4.2.1 Functionality

When defining functionality, we faced challenges in deciding how best to integrate social features like friend connections and following mechanisms. Thorough user research was essential to understand the target audience's needs and preferences, which informed the tailoring of features to effectively meet user expectations.

4.2.2 Data Visualization

Additionally, every aspect of data visualization needs to be carefully analyzed to ensure it effectively addresses users' needs. This involves evaluating how data is presented in the application to ensure clarity, usability, and relevance, ultimately facilitating a more intuitive user experience.

The seamless collaboration between the frontend and backend teams significantly enhanced the application's performance, making it more responsive and efficient in handling large volumes of data. The frontend team's adaptability in integrating real-time updates and their proactive approach to incorporating user feedback exemplified their commitment to continuous improvement. Their work addressed the initial challenges and set new standards in user interface design for complex systems.

5. Go To Market

Identifying our niche market was challenging. Through our research, we learned that start-ups CEOs were not ideal candidates for our product as their primary focus is on growing their businesses, which involves tackling critical operational issues and driving growth. Most general news, tweets, or global metrics do not significantly impact their day-to-day business decisions. In contrast, we discovered that executives of well-established companies find this information highly relevant. These executives must stay informed for meetings with clients and public forums where strategic knowledge of industry conditions and global events is crucial.

Furthermore, after working with the other teams, we conclude that our user profile can be broad. The initial sign-up form is designed to effectively filter out likes and dislikes, tailoring the information to be nearly personalized for each user. This customization enhances user engagement by ensuring that the content they receive is directly relevant to their interests and needs.