Solution Design Document: AI Social Media Assistant

Version: 1.1
Date: May 5, 2024
Author: Samuel Dagne

1. Introduction

Chosen Role: AI Assistant for Social Media Content Curation & Draft Generation

Industry/Niche: Technology Marketing / Content Marketing (with a specific focus on sourcing and repurposing news related to Artificial Intelligence and Machine Learning).

Problem Statement: Social Media Managers (SMMs) and Content Marketers in the tech space, particularly AI/ML, face the constant challenge of staying updated with rapidly evolving news and translating relevant findings into engaging content for various social platforms. This involves time-consuming tasks like:

- Continuously monitoring numerous news sources.
- Sifting through information to find truly relevant and impactful content.
- Reading and understanding articles to extract key messages.
- Crafting distinct posts tailored to the constraints and audience expectations of different platforms (e.g., Twitter vs. LinkedIn).
- Generating relevant hashtags.
  This manual process limits the SMM's ability to focus on higher-level strategy, engagement, and performance analysis.

Proposed Solution: This project proposes an AI-driven agent designed to automate key parts of the content curation and drafting workflow. The agent leverages the speed and power of the Groq API with advanced language models (e.g., Llama3) to:

- Proactively monitor specified news sources (via NewsAPI) for recent articles related to AI/ML keywords.
- Select one or more top relevant articles based on predefined criteria.
- Utilize Groq API to automatically summarize the core content of the selected article(s).
- Employ Groq API to create distinct draft posts based on the summary, tailored for both Twitter and LinkedIn, including relevant hashtags.
- Deliver these generated drafts (including the source article link) via Telegram for review and potential use by the SMM.
- Operate on an automated schedule, ensuring a consistent flow of content ideas.

2. Key Tasks & Workflows

The AI agent will execute the following workflow automatically on a defined schedule (e.g., daily at the time specified in config.py):

Scheduled Trigger: The workflow is initiated by an internal scheduler (Python schedule library).

Fetch Relevant Content:

- Query the NewsAPI (https://newsapi.org/v2/everything) endpoint.
- Search parameters: Keywords (e.g., "artificial intelligence", "machine learning", "LLM"), language (English), recency (e.g., last 24 hours), sort order (e.g., relevancy). (Configurable via config.py).

Select Top Article(s):

- From the NewsAPI results, filter for articles with usable text content.
- Select the top N articles (as defined in config.py, e.g., N=1) based on the API's sort order and content availability.

Process Each Selected Article:

- Extract the article title, URL, and available text (description or content snippet from NewsAPI payload).
- **AI Summarization (Groq):** Send the extracted text to a designated model on the Groq API (e.g., llama3-8b-8192). Construct a prompt to request a concise summary (e.g., 2-3 sentences) suitable for social media context.
- **AI Draft Generation (Twitter - Groq):**
  - Construct a prompt for a model on the Groq API (e.g., llama3-8b-8192).
  - The prompt instructs the model to create a short (under 280 characters), engaging Tweet based on the article's title and generated summary, including the source URL and 2-3 relevant hashtags.
-
- **AI Draft Generation (LinkedIn - Groq):**
  - Construct a separate prompt for a model on the Groq API (e.g., llama3-8b-8192).
  - The prompt instructs the model to create a slightly longer, professional LinkedIn post discussing the article's significance (based on title and summary), including the source URL and 3-4 relevant hashtags.

Format & Output:

- Combine the original article title, URL, Groq-generated summary, Tweet draft, and LinkedIn draft into a structured output.
- **Deliver the output:**
  - Primary: Print formatted output to the console.
  - Secondary: Send a formatted message (MarkdownV2) to a predefined Telegram chat via the Telegram Bot API.

Logging: Log key steps, successes, API interactions (including Groq token usage), and failures for monitoring and debugging.

3. High-Level Architecture & Tech Stack

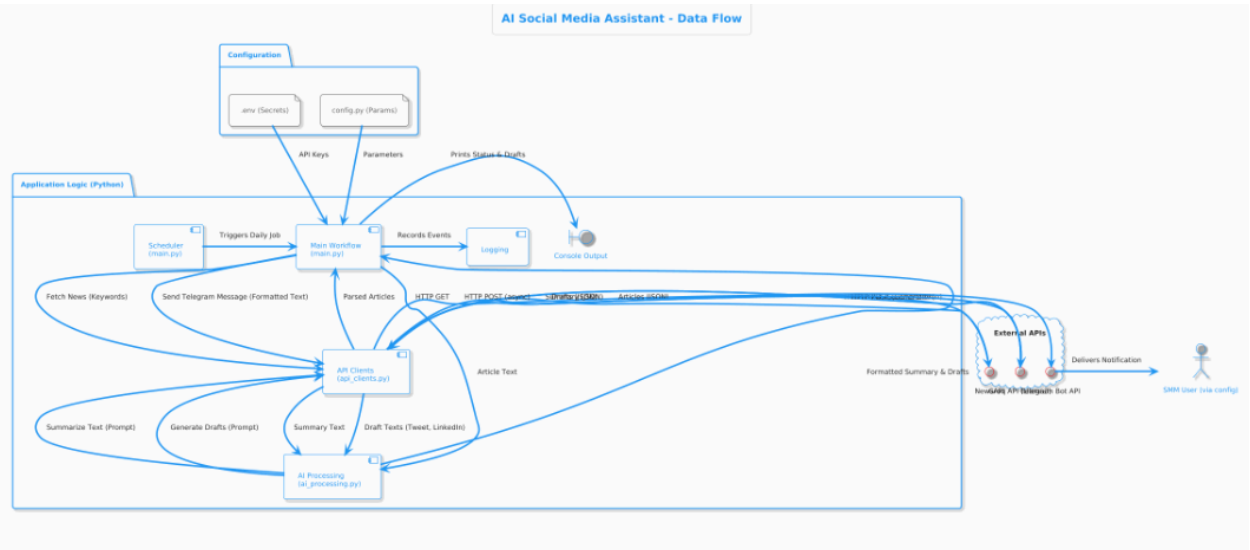
3.1 Language: Python 3.9+

3.2 Core Libraries:

- requests: For making HTTP calls to NewsAPI and Groq API.
- python-telegram-bot: For interacting with the Telegram Bot API.
- schedule: For easy implementation of in-script job scheduling.
- python-dotenv: For managing API keys and sensitive configuration securely via a .env file.
- asyncio: For handling asynchronous operations with the Telegram Bot API.
- datetime, time, os, logging, re: Standard Python libraries.

3.3 Execution Environment: The agent is designed to run on a local machine or any environment with Python and internet access.

3.4 Data Flow:



3.5 Data Storage:

- **Configuration (API Keys, Parameters):** Stored in a .env file (for keys) and config.py (for operational parameters), loaded at runtime.

4. Third-Party APIs & Integrations

4.1 NewsAPI (newsapi.org):

- **Purpose:** Primary source for discovering recent news articles relevant to specified keywords (AI/ML).
- **Integration:** Via requests library making calls to the REST API, managed in api_clients.py.
- **Authentication:** API Key (NEWS_API_KEY) loaded from .env.
- **Endpoint:** https://newsapi.org/v2/everything.

4.2 Groq API (groq.com):

- **Purpose:** Provides access to high-speed, large language models (e.g., Llama3) for advanced NLP tasks. This is the core AI engine.
  - **Summarization:** To condense article content.
  - **Text Generation:** To create social media post drafts based on prompts.
- **Integration:** Via requests library making POST calls to the Groq chat completions endpoint (https://api.groq.com/openai/v1/chat/completions), managed in api_clients.py.
- **Authentication:** API Token (GROQ_API_KEY as a Bearer Token) loaded from .env.
- **Models Used:** Configurable via config.py (e.g., llama3-8b-8192).

4.3 Telegram Bot API (core.telegram.org/bots/api):

- **Purpose:** Provide a convenient channel for delivering the generated drafts directly to the user/SMM.
- **Integration:** Via python-telegram-bot library, managed in api_clients.py using asyncio for non-blocking operations.
- **Authentication:** Bot Token (TELEGRAM_BOT_TOKEN) loaded from .env.
- **Method:** sendMessage with ParseMode.MARKDOWN_V2.

4.4 Scheduler (schedule library):

- **Purpose:** Enables proactive, automated execution of the content generation workflow.
- **Integration:** Used within main.py to run the main job function at defined daily times.

5. Automation & Action

5.1 Task Automation: This agent automates:

- Searching news sources for relevant AI/ML content.
- Extracting key information.
- Generating concise summaries tailored for social media.
- Drafting initial posts for Twitter.
- Drafting initial posts for LinkedIn.
- Adapting tone and length for different platforms.
- Suggesting relevant hashtags.

5.2 Meaningful Action: The agent takes the meaningful action of generating tangible, ready-to-review social media draft posts, powered by advanced LLMs via Groq, tailored for specific platforms. This provides direct value by significantly reducing content creation time and offering high-quality initial drafts.

5.3 Proactive Action: The agent acts proactively by utilizing the schedule library to automatically run the entire discovery, AI processing, and drafting process at regular daily intervals. It pushes content ideas to the user without requiring manual triggers for each run.

6. Setup & Running Instructions (Summary - Full details in README.md)

1. Ensure Python 3.9+ is installed.
2. Set up a virtual environment and install dependencies using pip install -r requirements.txt.
3. Create a .env file in the project root with NEWS_API_KEY, GROQ_API_KEY, TELEGRAM_BOT_TOKEN, and TELEGRAM_CHAT_ID.
4. Run the agent using python main.py.
5. (Optional) Modify operational parameters like SCHEDULE_TIME in config.py.