Veteran Sentiment Analysis System: Final Report

Deloitte GPS

Alexis Herbert

Abdoulay Lashley

Vaishnavi Deepak Patil

Sameer Kumar

Lakshminarayana Natarajan

Shreya Pavadashettar Jayanna

December 7, 2025

Introduction and Context

System Overview

The Veteran Service Organization (VSO) Sentiment Analysis System is an integrated analytics platform designed to help VSOs monitor, understand, and respond to evolving concerns within the veteran community. The system ingests real-time online discussions from Reddit and X, cleans and processes the text, performs sentiment and thematic analysis, identifies emerging topics, and visualizes findings through an interactive dashboard. It also includes an AI Copilot that enables real-time, context-aware question answering grounded in processed data.

The VSO struggles to extract insights from large amounts of unstructured online conversations. Traditional surveys generate delayed, static feedback, which leaves emerging issues like mental health, financial strain, and minority/women veteran experiences undetected until they worsen.

Purpose

The system was created to address key challenges:

- Delayed and insufficient feedback loops
- Difficulty identifying negative trends early
- Lack of accessible tools for real-time insights
- No centralized view of sentiment, themes, and personas

Intended Audience/Stakeholders:

- Deloitte review and product teams
- VSO decision-makers
- Policy analysts
- UMD faculty/students evaluating technical implementation

The system aims to:

- Provide situational awareness of veteran concerns.
- Reduce latency between emerging issues and VSO responses.
- Give VSOs a data-driven foundation for targeted outreach.
- Improve understanding of veteran personas and thematic drivers.

The Development Process

Research Phase

During the research phase, the team conducted a comprehensive analysis of challenges faced by Veteran Service Organizations, focusing particularly on recurrent pain points communicated by veterans such as mental health concerns, financial stress, and support gaps. The team also evaluated the availability and quality of relevant public conversation data on Reddit and X, assessing which platforms provided sufficient volume and representativeness for sentiment analysis. To ensure the project aligned with stakeholder expectations, the team held discussions with Deloitte mentors, using their feedback to shape user stories and define the core functionality needed for a system that could deliver meaningful, actionable insights to the VSO.

Design Phase

In the design phase, the team translated research insights into tangible system components. Low-fidelity wireframes were developed to outline dashboard structure, user flows, and key interactions. Architecture diagrams were created to illustrate the full data pathway, from API ingestion to processing, JSON generation, dashboard visualization, and AI Copilot integration, ensuring clarity across technical layers. Thematic categories were defined through iterative analysis of conversation patterns, and UI styling choices were guided by usability

expectations and VA-inspired accessibility principles. These materials provided a strong foundation for system development and ensured alignment between technical design and stakeholder needs.

CRUD Matrix

Process / Entity	DataScientist	Request Data	Collect Process	Social Media	Raw Data	Pre-processed Data	Analyze Process	Analyzed Data	Generate Insights	Insight	VSO
Submits a request	CRUD	CRUD									
Configure API & data sources	R	R	CRU	R							
Extract and store raw data			R	R	C						
Clean and preprocess data			R		R	C					
Run sentiment analysis	R	R				R	CRU	C			
Update analysis parameters	U	U					U	U			
Generate Insights	R	R					R	R	CR	CR	
View and filter insights	R								R	R	R
Delete outdated insights										D	D

Development Phase

The development phase involved building the system's three integrated layers: the data processing pipeline, the interactive dashboard, and the AI Copilot. The data pipeline was implemented in Python using spaCy, Transformers, and KeyBERT to handle text cleaning, normalization, sentiment scoring, keyword extraction, and synthetic text generation. In parallel, the dashboard was built with HTML, JavaScript, and CSS, providing interactive visualizations, filtering capabilities, thematic insights, and persona analysis. The backend Copilot was built using FastAPI and a Retrieval-Augmented Generation (RAG) framework, allowing users to ask contextual questions grounded in the processed dataset. Together, these components formed a cohesive, end-to-end analytical system capable of generating real-time insights.

Testing Phase

Throughout testing, the team performed multiple layers of validation to ensure reliability, accuracy, and consistency. Unit tests were run on individual pipeline components to confirm that cleaning, scoring, and extraction functions behaved as expected. The dashboard underwent structured QA checks to verify correct rendering of charts, accuracy of sentiment and theme metrics, and proper interaction among filters. The AI Copilot was evaluated using scripted prompts to validate that its responses were grounded in the dataset and free from hallucinations. Finally, regression testing was conducted each time the dataset was updated to ensure system stability across the pipeline, dashboard, and Copilot components.

Deployment Phase

Deployment focused on ensuring the system could be easily hosted, run, and accessed by both technical and non-technical users. The full application was deployed locally using lightweight Python servers, allowing the data pipeline, dashboard, and AI Copilot to run

seamlessly on any machine with Python installed. Users begin by setting up a virtual environment, installing required Python libraries, and configuring necessary API keys. Once the environment is prepared, the data pipeline is executed to generate the processed datasets used throughout the system. The AI Copilot backend is then launched using a FastAPI server running on port 8001, while the interactive dashboard is made accessible through a simple local web server running on port 8000. This deployment structure ensures that all system components are operational through a standard local hosting setup, making the solution both reproducible and easy to navigate during client demonstrations and testing.

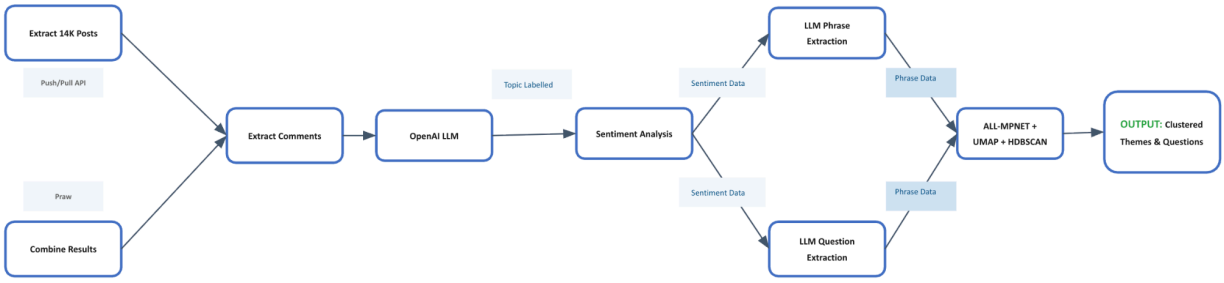
Challenges and Solutions

During the project, the team encountered several challenges that shaped the system's evolution. One major challenge was the high variability and noise present in Reddit and X data, which required a multi-step cleaning and normalization strategy to ensure consistent processing. The team also discovered that traditional sentiment analysis approaches often produced overly neutral outputs, leading them to integrate LLM-enhanced scoring and synthetic examples to better capture emotional nuance. Additionally, visualizing numerous themes and personas in a way that remained intuitive for nontechnical users required multiple iterations of dashboard layout and filtering logic. Finally, preventing the AI Copilot from hallucinating was a critical challenge; this was resolved through a strict RAG pipeline that constrained responses to the processed dataset. These solutions collectively improved accuracy, usability, and trustworthiness across the system.

System Functionality

This section explains how the Veteran Sentiment Analysis System works from end to end, tying together the data pipeline, dashboard, and AI Copilot into a single experience that Deloitte reviewers, VSOs, and faculty can understand without opening the prototype. Where relevant, the description mirrors the actual screens and features available in the dashboard_v5 interface.

High-Level System Architecture



At a high level, the system follows a linear but modular flow:

1. Data ingestion from Reddit (and future X integration) into a raw text store
2. Text processing and enrichment that transforms posts into structured sentiment, themes, and personas
3. Dataset generation into Excel and JSON files used by downstream components
4. Interactive visualization through the web dashboard
5. Natural language querying through the AI Copilot, which is grounded in the same processed data

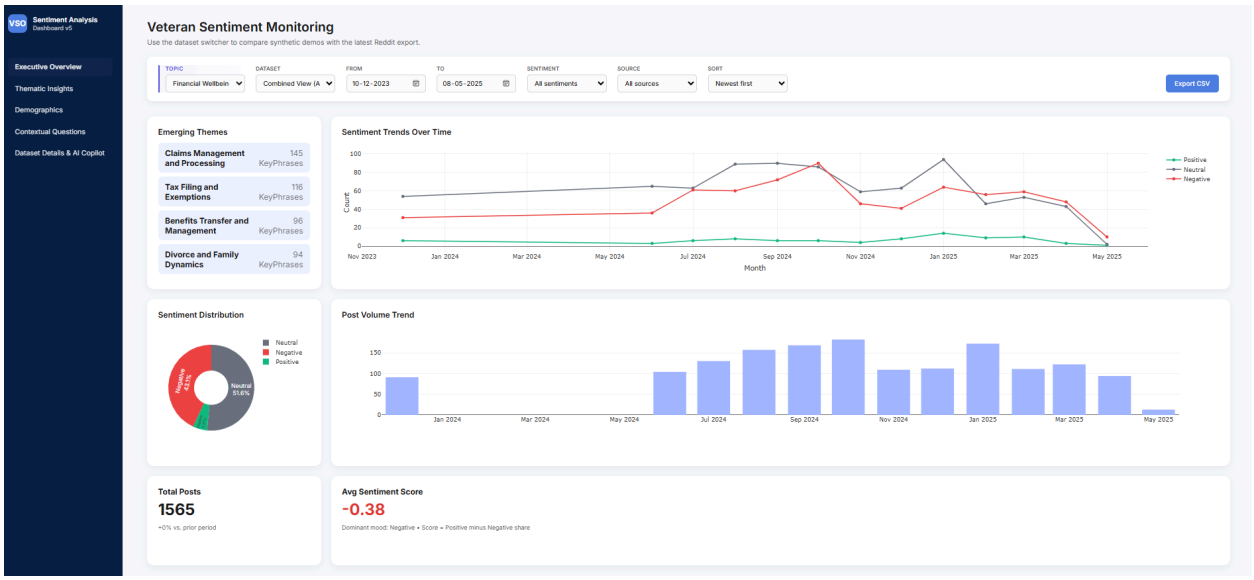
---

Core Features and Their Purpose

The dashboard is organized into a small set of focused views that all share the same global filters and datasets. Across all screens, a global filter bar at the top controls dataset, date range, sentiment, source, and sort order, providing a consistent way to move from a broad overview to very focused questions.

Executive Overview

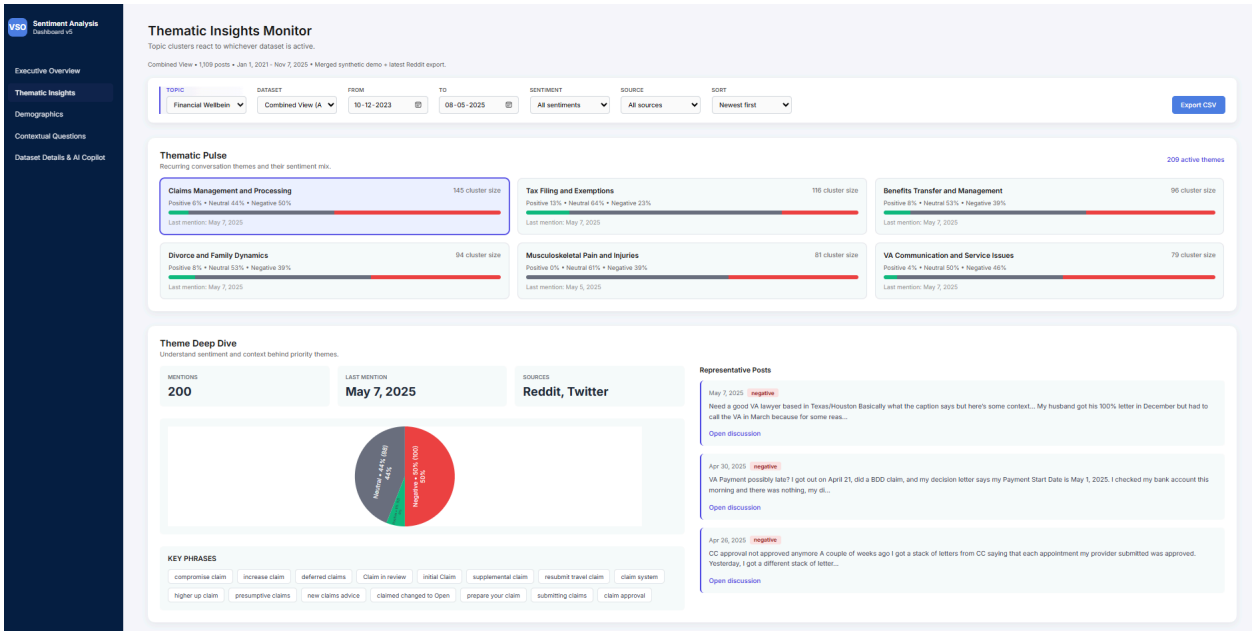
- Provides a single-screen snapshot of post volume, average sentiment score, sentiment mix, and emerging topics for a chosen time window.
- KPI tiles, sentiment trend lines, and a Post Explorer panel give decision makers an immediate sense of *"how things feel"* and whether anything has shifted recently.



Thematic Insights

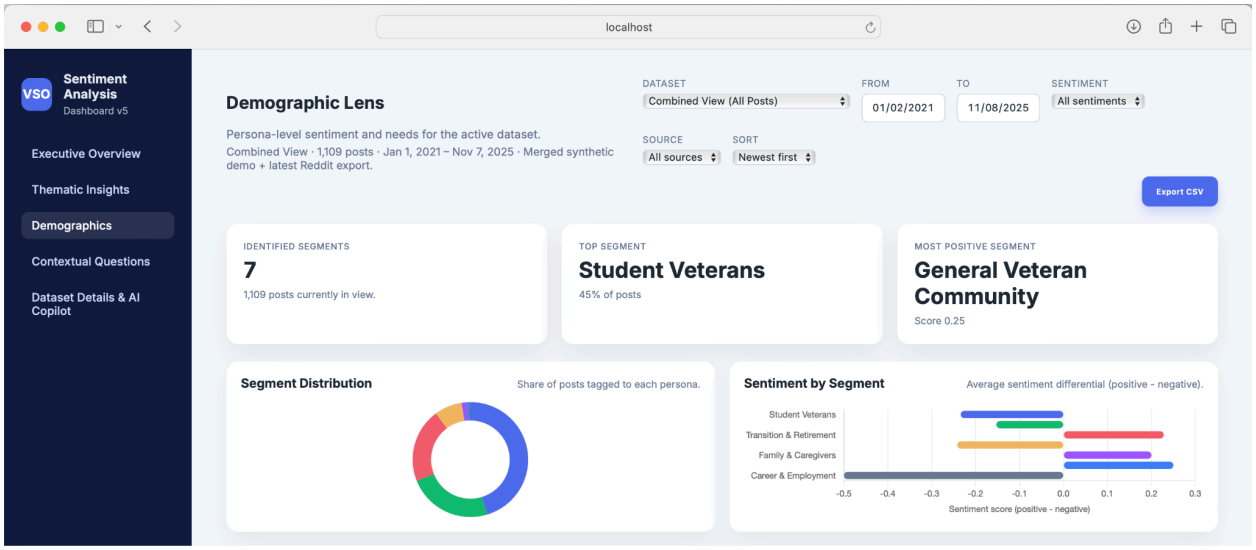
- Group conversations into themes such as Financial Resilience, Financial Stress, Family Caregivers, Voc Rehab, and Student Veterans.

- Thematic Pulse cards summarize mentions, sentiment split, and last activity, while a Theme Deep Dive panel shows detailed sentiment bars, key phrases, and representative posts.



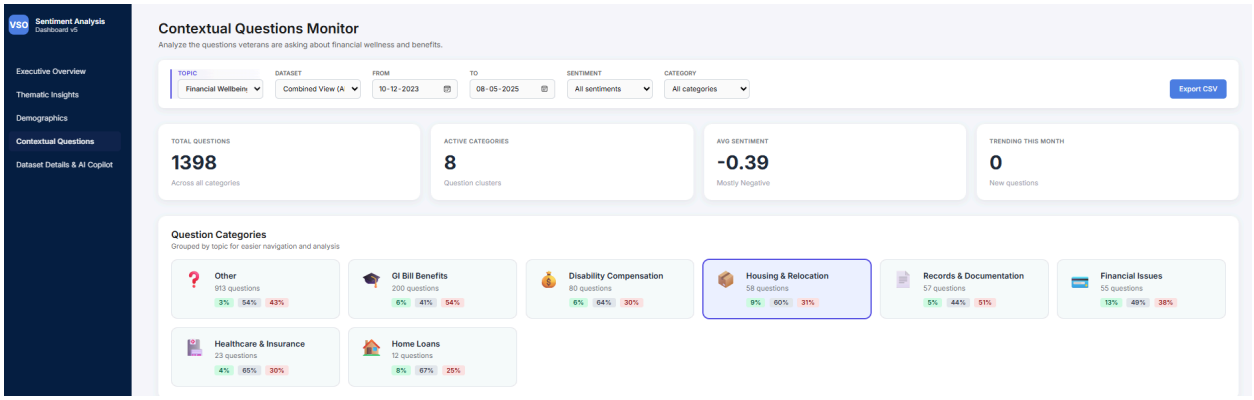
Demographics (Demographic Lens) - *only applicable in the case of Synthetic posts*

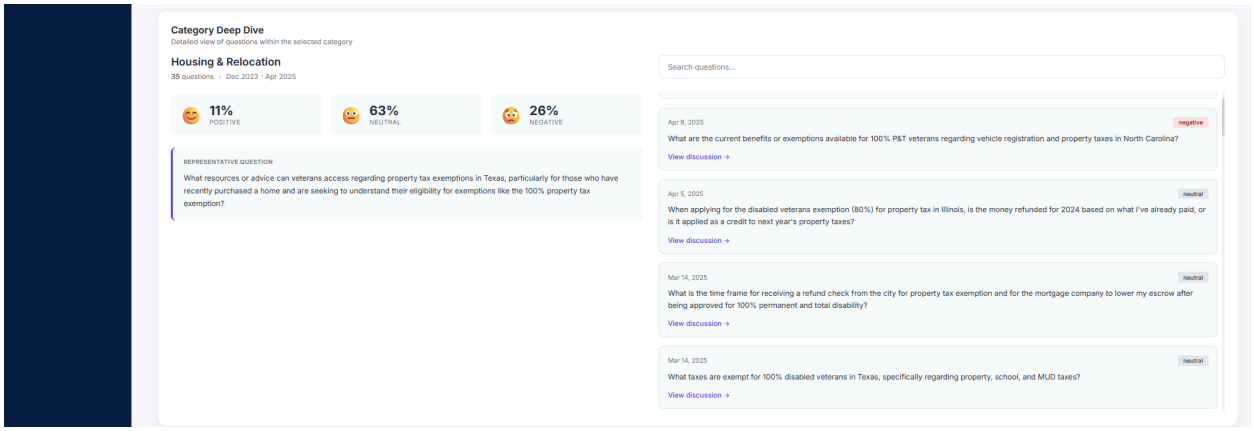
- Surfaces persona segments inferred from the data (e.g., Student Veterans, Disability & Medical Support, Transition & Retirement, Family & Caregivers, Housing & Living).
- Shows segment distribution, sentiment by segment, and snapshot cards with key phrases and example posts so VSOs can see which groups are most affected.



Contextual Questions

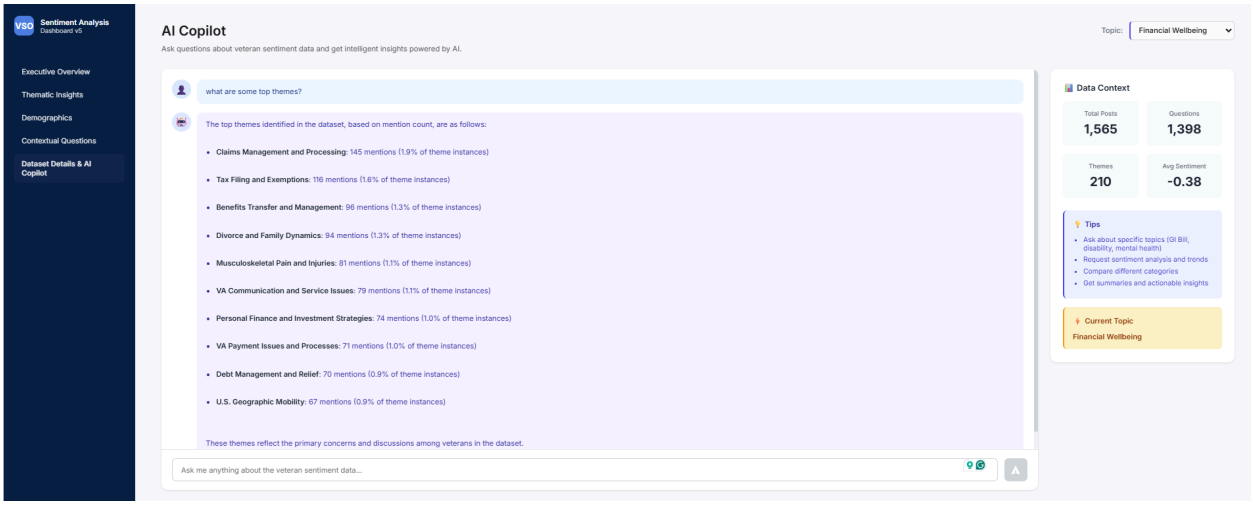
- Offers one-click cards like *"Investigate recent negative sentiment"* or *"Check persona leaders this quarter."*
- Each card applies a preset filter combination and routes the user to the most relevant view, helping nontechnical users get to useful cuts of the data without guessing which filters to select.





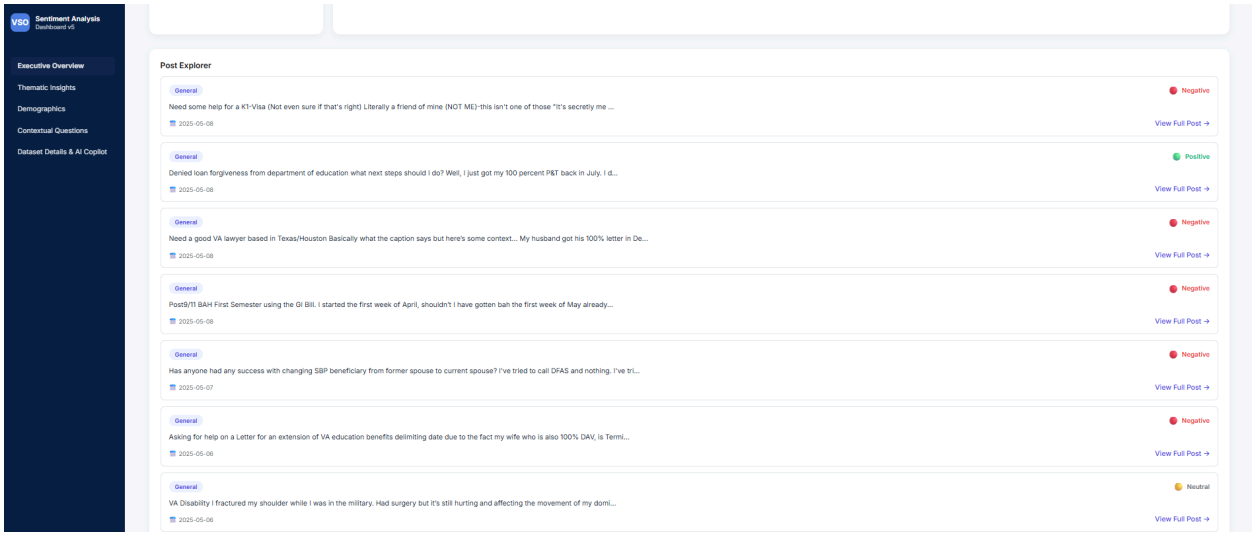
Dataset Details & AI Copilot

- Summarizes the active dataset (Synthetic Demo, Reddit Export, or Combined View) and shows total posts, positive sentiment share, date range, and active themes.
- Hosts the Copilot interface where users ask questions in natural language and receive grounded explanations based on the currently selected dataset and filters.



Export & Post Explorer

- Appears below the primary charts and tables on major screens.
- Lists the individual posts that sit behind the metrics, including text excerpts, dates, sentiment labels, and sentiment scores, with links back to Reddit or to a synthetic conversation template.
- A global Export CSV button exports the currently filtered slice of data for offline analysis.



Backend Processes and Data Handling

The backend is implemented primarily in Python and FastAPI and is responsible for ingesting, processing, storing, and serving the data that powers both the dashboard and the AI Copilot.

1. *Data ingestion and storage*
    - Reddit posts are collected via the Reddit Python API and Old Reddit scraping modules.
    - Raw posts are stored in intermediate files in the Data Extraction and Analysis directories, preserving original text and metadata (timestamps, subreddit names, URLs).
2. *Preprocessing and enrichment*
    - A preprocessing module cleans and normalizes text by removing URLs, emojis, markup, and standardizing casing and punctuation.

- Sentiment analysis runs through two complementary approaches: a fine-tuned sentiment model and an LLM-based scorer. Their outputs are reconciled into a final sentiment label and numeric score for each post.
- Key phrase extraction uses statistical and LLM-driven methods to identify meaningful phrases, which are later used to define themes and to populate phrase chips on the dashboard.
- Clustering algorithms group related phrases and questions into themes; personas are inferred from language patterns in posts and combined with sentiment scores and volumes.

3. *Dataset generation*
   - The synthetic_generator.py script creates a synthetic dataset of posts that mimic realistic veteran conversations while protecting privacy.
   - Pipeline scripts convert all processed outputs into:
     - Excel workbooks for offline review, and
     - JSON files (e.g., synthetic_data.json, combined_data.json) that the dashboard consumes directly.
4. *AI Copilot backend (FastAPI + RAG)*
   - The ai-chat endpoint in server.py FastAPI application provides an HTTP endpoint that accepts user questions and the current dashboard context (selected dataset, filters, and an internal summary of available metrics).
   - A retrieval component identifies the most relevant themes, context questions, and posts from the processed dataset and passes them into an LLM prompt.

- The LLM generates a concise answer that references actual metrics and examples from the retrieved subset, and returns it as a JSON response to the dashboard.

Frontend–Backend Interaction

Although the dashboard serves as a single-page application, it interacts with static and dynamic resources behind the scenes to keep the experience responsive and grounded.

1. *Static data loading*
    - When the user opens Dashboard/index.html, the frontend loads files for the active datasets (Reddit export).
    - Charts, KPIs, and tables are rendered entirely in the browser using HTML, CSS, and JavaScript, so filter changes feel instantaneous.
2. *Filter propagation*
    - When the user changes the dataset, date range, sentiment, or source in the global filter bar, JavaScript filter logic recomputes:
        - Aggregate metrics (total posts, average sentiment, distributions)
        - The active subset of posts shown in the Post Explorer
        - Counts and scores shown in Thematic Pulse cards and deep dive snapshots
    - The same filtered subset is also used to build context for calls to the Copilot backend.
3. *Copilot requests*
    - When a user submits a question in the Copilot input box, the dashboard sends a request to the ai-chat endpoint containing:
        - The question text

        - Current dataset selection and filter parameters
        - A compact summary of visible metrics, where applicable
    - The backend responds with a narrative answer and optional bullet points. The frontend renders this in the Copilot chat panel, maintaining a conversational history for the session.
4. *External navigation and exports*
    - Clicking a post in the Post Explorer opens the original Reddit thread or synthetic conversation template in a new browser tab.
    - Clicking the Export CSV button triggers a JavaScript function that packages the currently filtered subset into a CSV file and initiates a browser download.

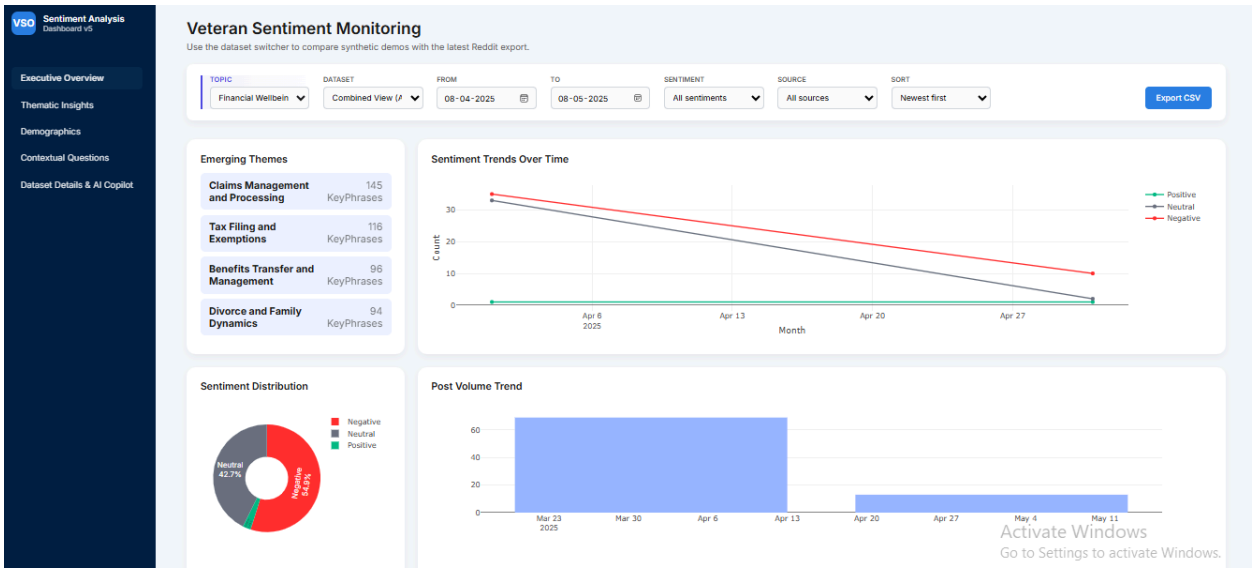
Input and Output Walkthrough

Core Workflow 1: Exploring Sentiment and Themes

Scenario: A VSO analyst wants to understand why negative sentiment has increased over the past month.

1. *User input*
    - Opens the dashboard and selects Combined View (All Posts) in the dataset switcher.
    - Sets Date Range to *"Last 30 days"* and Sentiment to *"All."*
2. *System processing*
    - The frontend filters the combined dataset to posts within the chosen date range.
    - It recomputes total post count, average sentiment score, sentiment distribution, and highlights the top Emerging Topics.

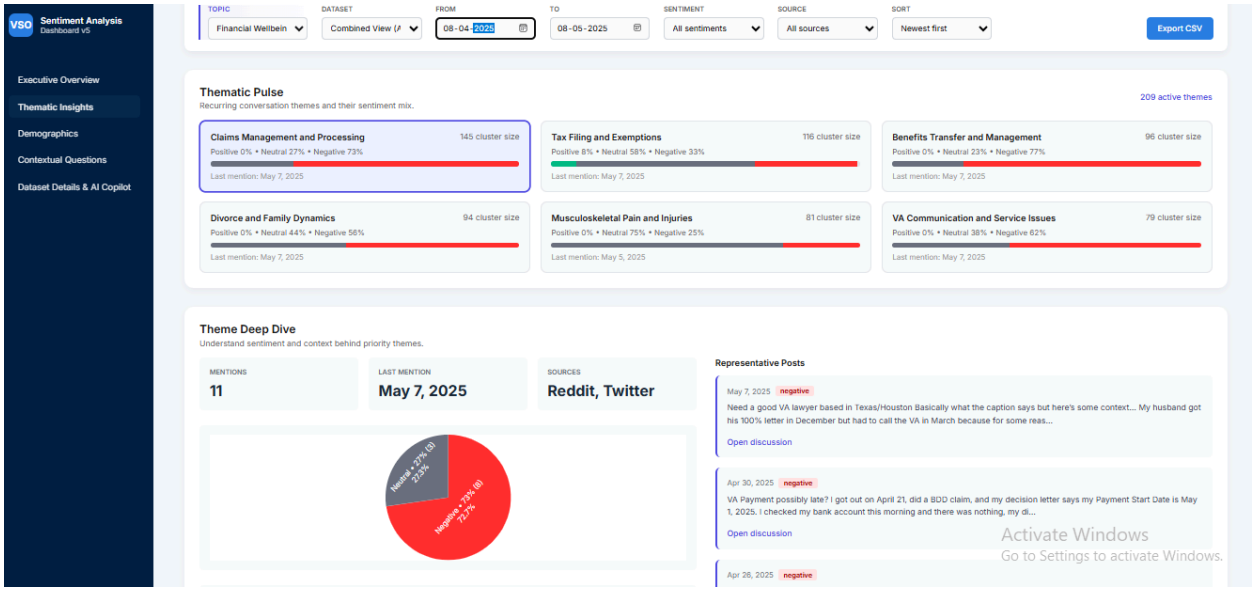
3. *Outputs*
    - Executive Overview updates to show:
        - A lower average sentiment score compared to prior periods
        - A shift in the Post Volume chart
        - Emerging topics such as Financial Stress or Education Benefits rising to the top



4. *Deeper exploration*
    - The analyst clicks Thematic Insights in the left navigation and inspects Thematic Pulse cards.
    - Selecting *"Financial Stress"* opens the Theme Deep Dive panel with sentiment breakdown and key phrases like *"rent increase," "benefit del*ays," or *"credit card debt."*

5. *Final step and evidence*

- The analyst scrolls to the Post Explorer section to read individual posts, then clicks Export CSV to download the filtered slice as supporting evidence for a leadership briefing.



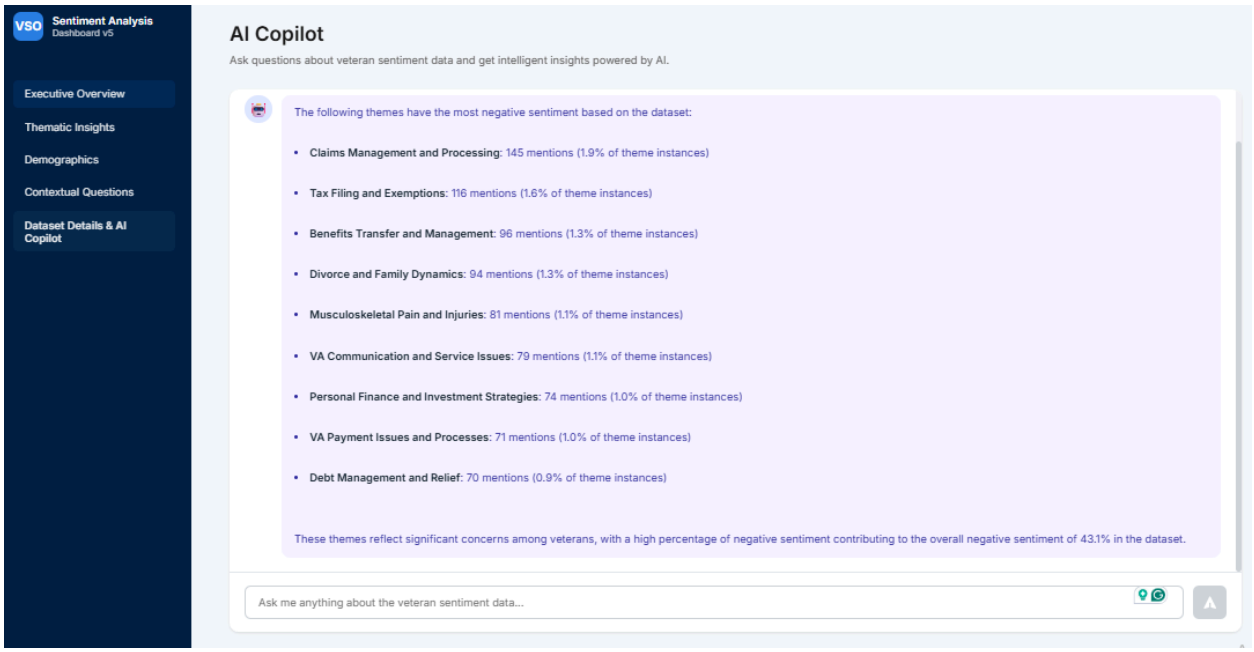
Core Workflow 2: Asking the AI Copilot

Scenario: A VSO leader wants a concise explanation rather than manually exploring filters.

1. *User input*
    - On Dataset Details & AI Copilot, confirms that the Combined View is active.
    - Types: *"Which themes are most negative in the last 30 days?"*
2. *System processing*
    - The dashboard passes the question, dataset selection, and current date range to the Copilot API.
    - The backend retrieves the subset of themes and personas that fit the last 30 days, ranked by negative sentiment and volume.

- The LLM synthesizes a short explanation that highlights the most negative themes and mentions the key personas associated with each theme.

3. *Outputs*
    - The Copilot returns a narrative answer such as:
        - A ranked list of themes with negative sentiment scores
        - A short explanation of which persona segments are driving those trends
        - Recommendations to inspect specific screens (e.g., *"Review Student Veterans in the Demographics view for more context"*).



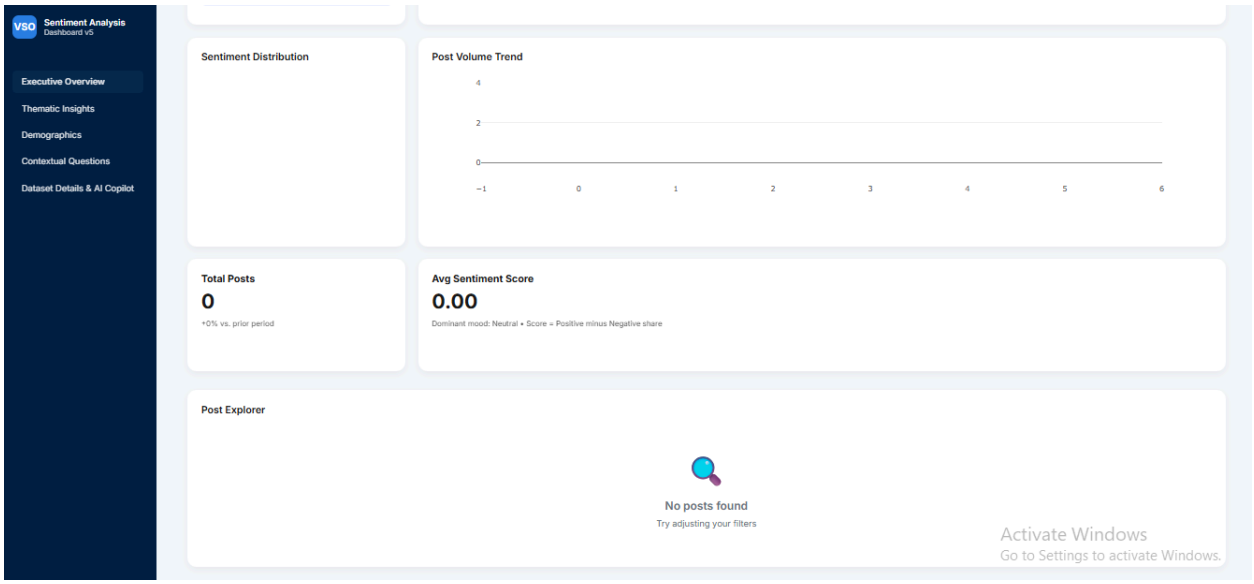
Error Handling and User Feedback

The Veteran Sentiment Analysis System relies on a controlled set of user inputs, limited to dashboard filters for topic, sentiment, source, and date range selection.

1. *Empty or overly restrictive filters*

a. If the combination of dataset, sentiment, and date range yields no posts, KPI tiles display zeros, and charts show *"No data for current filters."*

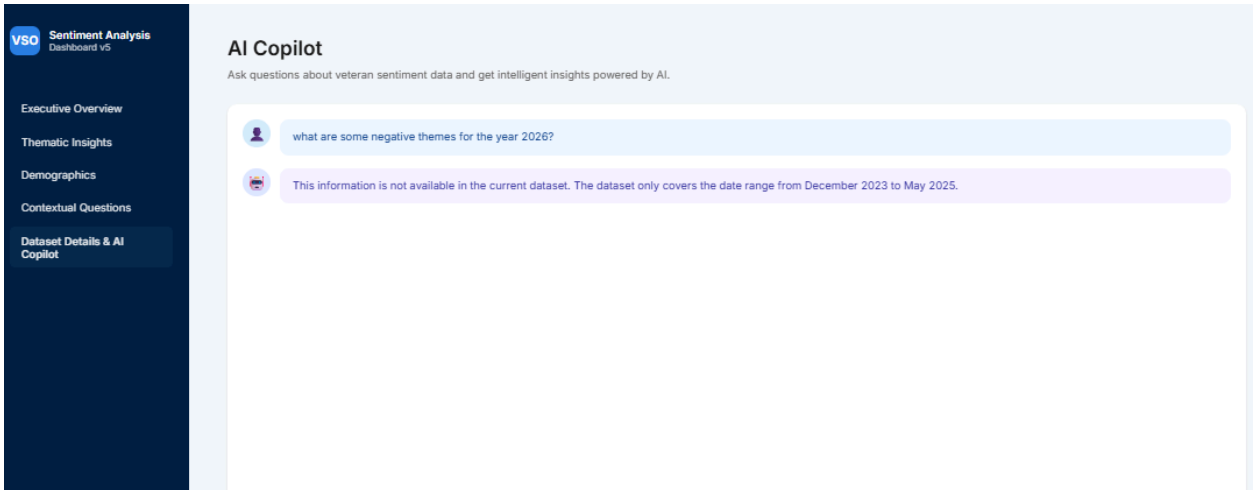
b. The Post Explorer table shows an inline message such as *"No posts found."*



2. *AI Copilot Response Constraints*

a. When the Copilot receives a question, but there are very few posts that match the current filters, the backend still returns a response. Still, it includes language that indicates low sample size, such as "*This information is not available in the current dataset. The dataset only covers the date range from December 2023 to May 2025.*"

b. The frontend surfaces this explanation directly in the chat to encourage cautious interpretation.



Standalone Understanding

This section provides a complete, self-contained understanding of the Veteran Sentiment Analysis System, ensuring readers can fully grasp how it operates without accessing the prototype. The system's workflows, user interactions, and technical components are described in clear, structured detail, supported by conceptual diagrams and definitions of key terms to remove ambiguity and reduce technical barriers.

Logical Workflow Breakdown

The system follows a linear, end-to-end workflow beginning with data collection and ending with insight visualization and natural-language querying. The workflow proceeds as follows:

1. **Data Ingestion**: Reddit posts are collected via API calls, which retrieve publicly available conversations on veteran topics.
2. **Text Processing**: The posts undergo cleaning, normalization, sentiment scoring, keyword extraction, and thematic clustering to transform raw text into structured insights.

3. **Dataset Creation**: Processed outputs are exported to Excel files, which serve as the standardized inputs for the dashboard and the AI Copilot.
4. **Visualization & Exploration**: Users interact with a multi-page dashboard that displays themes, sentiment trends, personas, KPIs, and individual posts.
5. **AI Querying**: Users can ask natural-language questions to the Copilot, which retrieves relevant pieces of processed data and returns structured explanations grounded in the dataset.

This workflow illustrates the system's progression from unstructured online data to actionable insights that support VSO decision-making.

User Journey Overview

A typical user journey is designed to be intuitive and insight-driven:

1. **Entry Point**: The user opens the dashboard and views high-level metrics on the Executive Overview page, gaining an immediate sense of volume, sentiment distribution, and emerging concerns.
2. **Exploration**: Through filters (platform, theme, sentiment, date), the user tailors the view to specific populations or issues, such as negative sentiment about mental health among student veterans.
3. **Deep Dive**: The user navigates to Thematic Insights to understand conversation drivers or to Persona/Demographic Insights for group-specific trends.
4. **Evidence Review**: The user opens the Post Explorer to read actual posts, validate patterns, and export examples for reporting.

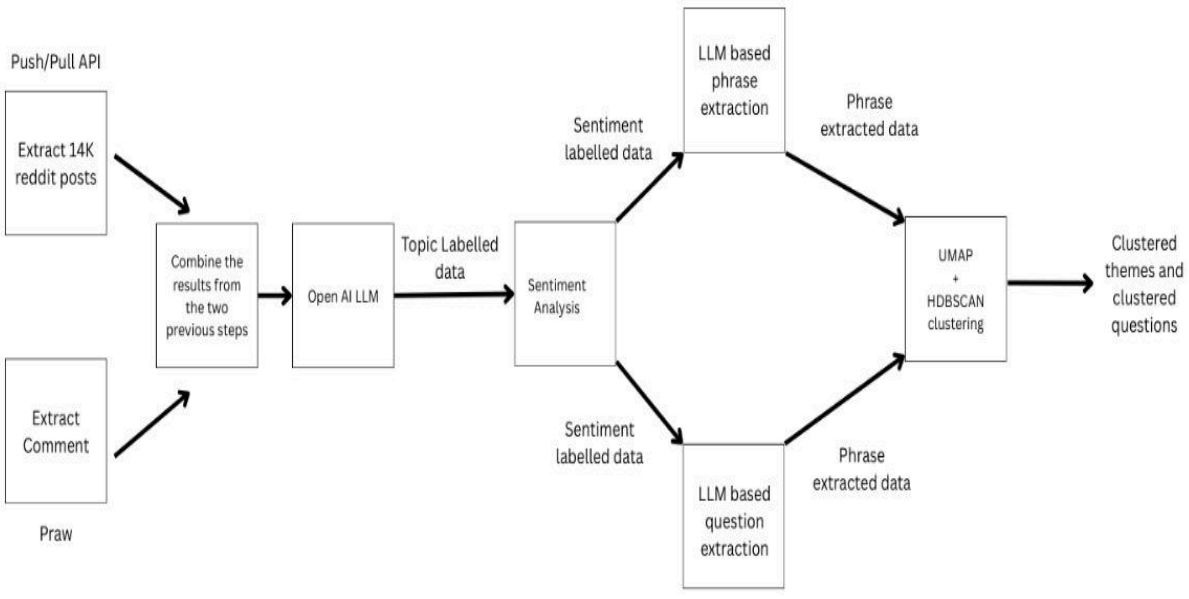
5. **Question Asking**: For more synthesized insights, the user enters a query into the AI Copilot, such as *"What are the fastest-growing concerns in the last 14 days?"* The Copilot analyzes relevant data and provides a clear, narrative answer.
6. **Decision Making**: Based on insights, a VSO stakeholder may identify emerging issues, prioritize outreach topics, or flag areas that require policy attention.

This narrative outlines how a typical stakeholder interacts with the system from first glance to final decision support without requiring technical knowledge.

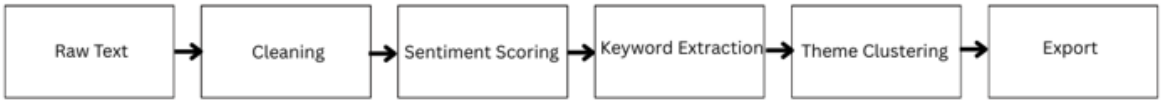
Visual Representations of Processes and Features

To support comprehension, the system can be summarized through simple diagrams and screenshots that visually reinforce how each layer interacts:

1. *System Architecture Diagram:*
    - This diagram shows the high-level flow of information through the system.



2. *Data Pipeline Flowchart:*



    - Highlights the transformation from raw to structured data.

---

3. *Dashboard Page Previews:*
    - Executive Overview Cards
    - Thematic Unsights
    - Persona Snapshot
    - Post Explorer Table

---

4. *AI Copilot Interface Example:*
    - User Query → Dataset Retrieval → Structured Answer
    - Visualizing how queries are processed strengthens understanding of backend logic.

These visuals help nontechnical readers quickly grasp the purpose and operation of each major feature.

Definitions of Technical Terms

To make the report accessible to all audiences, the following technical terms are clearly defined:

- API (Application Programming Interface): A tool that allows the system to automatically retrieve data from Reddit and X.

- Cleaning/Normalization: Processes that remove noise (URLs, emojis, irrelevant characters) and standardize text for analysis.
- Sentiment Scoring: Assigning a positive, negative, or neutral value to a post based on its emotional content.
- Keyword Extraction: Identifying the most important phrases in a post using models like KeyBERT or NLP libraries.
- Thematic Clustering: Grouping posts into related categories (e.g., mental health, financial stress) using machine learning algorithms.
- JSON (JavaScript Object Notation): A structured, lightweight format used to store data for the dashboard and Copilot.
- RAG (Retrieval-Augmented Generation): An AI method that retrieves relevant data before generating an answer, ensuring accuracy and grounding.
- FastAPI: The backend framework that powers the AI Copilot server.
- Dashboard Filters: Controls that allow users to refine insights by theme, sentiment, platform, and date.
- Personas: Categories of veterans (e.g., student veterans, caregivers) used to segment insights.

Providing these definitions reduces reliance on external technical knowledge and ensures clarity.

Deployment Instructions

Deployment is intentionally local: run the FastAPI backend on http://localhost:8000, serve the final dashboard from Dashboard/ (e.g., python -m http.server 5501 →

http://localhost:5501), and optionally serve the legacy dashboard_v5/ build (e.g., python -m http.server 5500 → http://localhost:5500) with its chat proxy on http://localhost:8001/api/chat.

Set Up and Run Guide

Tech Stack Used:

- *Backend & Data Processing*
  - Python 3.10+
  - Pandas
  - spaCy
  - TextBlob
  - KeyBERT
  - Transformers / LLM-enhanced components

    Custom Python scripts for extraction, sentiment scoring, keyword extraction, and question generation
- *Dashboard*
  - HTML
  - CSS
  - Vanilla JavaScript
  - Chart.js (visualizations)
- *AI Copilot*
  - Python-based chatbot server

Prerequisites

1) Python 3.10 or later

   Ensure Python is added to PATH and check the version using the command :

   python --version

2) Git
3) Visual Studio Code (VS Code)
4) A terminal you can use inside VS Code:
   - Windows: PowerShell or Command Prompt

     macOS / Linux: Bash / Zsh
5) OpenAI API Key

   If you don't have one, create it here (you need to buy it):

   https://platform.openai.com/api-keys

Installation steps

Since we already have the output file ready, you can skip steps 7, 8, and 9. If you want to generate the output files, you will need to follow all the steps.

Step 1: Clone the Repository

Use the command

git clone https://github.com/UMDMSISCapstone/F2025-501-Deloitte.git

cd F2025-501-Deloitte

Step 2: Open the project in VS Code

- Open VS Code → File → Open Folder
- Select the cloned F2025-501-Deloitte directory

The folder structure looks like:

Directory flow

F2025-501-Deloitte/

├─ Analysis/ # NLP scripts: cleaner.py, sentiment*.py, keyword_extraction*.py, question_generation.py, etc.

├─ Data extraction/ # Reddit/Pushshift collectors (all_posts.py, pullshift_api.py), topic_modelling, final input notebooks + XLSX

├─ Output/ # Export artifacts (financial/mental wellbeing Excel files, JSON sentiments, metadata)

├─ Dashboard/ # Current dashboard build (index.html, styles.css, app.js + topic/theme XLSX)

├─ dashboard_v5/ # Earlier dashboard build (index.html, styles.css, app.js/app_v5.js, combined_data.json, synthetic_data.json, synthetic_posts/)

├─ Clustering_Keyphrases/ # clustering.py and themes_generation.py for keyphrase grouping

├─ ClusteringExperiment/ # clustering_exp1.ipynb and related XLSX checkpoints

├─ docs/ # Documentation (Word/PDF), architecture images, flow diagrams, feature showcase

├─ scripts/ # Helper script convert_output_to_json.py

├─ chatbot_server.py # FastAPI/uvicorn endpoint for the AI copilot

├─ config.json # Configuration (API keys/parameters)

├─ gemini_chatbot.log # Copilot log output

├─ main.py # Orchestrator/runner for cleaning/sentiment pipeline

├─ update_ai_copilot.py # Copilot update script

├─ requirements.txt # Python dependencies

```
└─ README.md # Repo overview
```

Step 3: Create a Python environment

Mac/Linux:

python3 -m venv venv

source venv/bin/activate *(To activate the virtual environment)*

Windows:

python -m venv venv

venv\Scripts\activate *(To activate the virtual environment)*

Step 4: Create your .env file

Inside the project root, create a file named .env and add:

OPENAI_API_KEY="<replace with your API KEY>"

HF_API_KEY="<HF_API_KEY>" # optional

Step 5: OPTIONAL - *If you do NOT have HF_API_KEY*

Comment out these two lines in main.py:

#financial_data['key_phrases_using_deepseek'] = financial_data['original_full_text'].apply(lambda x: extract_keywords_structured_open_source(x, model="deepseek-ai/DeepSeek-V3.2-Exp"))

#financial_data['key_phrases_using_mistral'] = financial_data['original_full_text'].apply(lambda x: extract_keywords_structured_open_source(x))

Step 6: Install dependencies

pip install --upgrade pip && pip install -r requirements.txt

(Note: In case there's an error, change the Python version to 3.11, and also if NumPy/pyarrow errors: pip install "numpy<2" "pyarrow<15" then rerun pip install -r requirements.txt)

Step 7: Run the backend data-processing pipeline

python main.py

Step 7 performs:

- cleaning
- sentiment scoring
- keyword extraction
- contextual questions
- Clustering of key phrases and questions
- And saves → Output/financial_data_output_latest.xlsx & in Dashboard folder
- It also saves the financial themes (and other topic themes) in the Dashboard folder.

Step 8: Convert the final Excel into the dashboard JSON (only applicable for dashboard_v5)

python scripts/convert_output_to_json.py --input

Output/financial_data_output_latest.xlsx --output

Output/financial_data_sentiments.json

Step 9: Combine real + synthetic data (only applicable for dashboard_v5)

python scripts/combine_datasets.py

Outputs:

- *dashboard_v5/combined_data.json*

Step 10: Launch the final dashboard frontend (only applicable for the dashboard)

Run python Dashboard/server.py

- Right-click on Index.html and then select *"Open with Live Server"*

Step 11: Start the AI Copilot backend (RAG service) (only applicable for dashboard_v5/dashboard)

Run from the repo root:

uvicorn chatbot_server:app --port 8001

The Copilot will read:

- dashboard_v5/combined_data.json

(Keep this terminal open)

Step 12: Launch the dashboard frontend (only applicable for dashboard_v5)

Open a second terminal (still inside project root):

python -m http.server 8000 (Run this inside venv)

Then visit (in browser):

- http://localhost:8000/dashboard_v5/index.html

The dashboard automatically loads:

- financial_data_sentiments.json

- synthetic_data.json
- combined_data.json

And sends Copilot queries to:

- http://localhost:8001/api/chat

Recommendations for Future Improvements

The Veteran Sentiment Analysis System currently provides a strong foundation for generating insights; however, several enhancements can significantly improve scalability, accuracy, reliability, and organizational value. Future improvements can be categorized into short-term enhancements, long-term strategic upgrades, and intelligent automation opportunities.

Short-Term Enhancements (1–3 Months)

Expand Data Source Coverage:

- While Reddit was used as the primary source for the prototype, future versions should integrate additional platforms such as Twitter/X, specialized veteran forums, and other public datasets. This expansion will produce more representative insights and reduce dependence on a single platform.

Enhance Sentiment Analysis Accuracy:

- Sentiment classification accuracy can be improved by fine-tuning models with veteran-specific terminology, military acronyms, and domain- and context-specific examples. This helps reduce overly neutral predictions and allows the system to better capture emotional nuance.

Introduce Advanced Filtering Capabilities:

- Additional filters, such as geographic location, military branch, and more granular persona categories, would enable deeper segmentation and insight exploration. These refinements strengthen use cases for policy teams and program managers.

Improve Synthetic Demo Data:

- Enhancing the generator used for demonstration datasets will make synthetic posts more realistic and scenario-driven. This allows safe but representative demos for stakeholders without privacy concerns.

Long-Term Enhancements (4–6+ Months)

Cloud Migration (AWS):

Scaling to production usage requires migration of pipeline and dashboard components to a cloud environment. A recommended architecture includes:

- S3 + CloudFront for hosting the dashboard
- AWS Lambda / ECS for the data pipeline and Copilot backend
- API Gateway for endpoint routing
    - This ensures improved reliability, scalability, and low-latency performance.

Automated Data Ingestion & Model Retraining:

- Future versions should incorporate scheduled pipeline runs to automatically refresh datasets daily or weekly. Automated retraining of sentiment and thematic models would ensure accuracy remains aligned with evolving conversational patterns.

Enterprise Security Enhancements:

- To support deployment within a VSO environment, the system should incorporate SSO authentication, role-based access control (RBAC), and audit logging. These enhancements ensure that sensitive insights are accessible only to authorized personnel.

Data Lake Architecture for Scalability:

- Storing processed and raw data in AWS S3 with Glue cataloging and Athena querying will support larger datasets, faster analytics, and integration with future visualization or reporting tools.

AI Copilot Improvements

Expanded RAG Capabilities:

- The AI Copilot can be upgraded with more robust retrieval mechanisms, larger context windows, and improved grounding to minimize hallucinations and ensure that all answers remain fully data-backed.

Intelligent Alerting System:

- Future versions should include automated detection of emerging trends, such as rising negative sentiment in specific themes or personas, and generate alerts for VSO leadership. This transforms the system into a proactive monitoring tool rather than a reactive analytics platform.

Dashboard and User Experience Enhancements

Trend Highlighting & Anomaly Detection:

- Integrating time-series analytics and anomaly detection (e.g., sudden spikes in negative sentiment) would provide users with early warning signals of emerging issues.

Enhanced Cross-Filtering and Visualization Controls:

- More dynamic interactions among sentiment, themes, personas, and time filters can create a smoother, more intuitive exploration experience.

Secure Authentication Layer:

- As the dashboard moves toward an enterprise setting, user authentication and access control become essential for protecting sensitive insights.

Strategic Long-Term Vision

The ultimate direction for this system is to evolve from a static analysis dashboard into a real-time, autonomous listening engine.

This future version would:

- Continuously monitor online discourse.
- Detect emerging crises or concerns early.
- Notify stakeholders automatically.
- Provide real-time Copilot explanations grounded in up-to-date data.

Conclusion

The Veteran Sentiment Analysis System provides a unified, efficient solution for understanding the concerns of the veteran community by real-time analysis of online conversations. By integrating data ingestion, text processing, sentiment, and thematic analysis, interactive visualization, and an AI-driven Copilot, the system transforms unstructured text into clear, actionable insights that support decision-making for Veteran Service Organizations.

Throughout development, the team designed a modular architecture, built an intuitive dashboard, and implemented a grounded RAG-based Copilot to ensure accuracy and usability. The system significantly reduces the manual effort required to identify emerging issues and enables stakeholders to quickly explore trends, personas, and sentiment drivers.

While the prototype offers a strong operational foundation, future enhancements, such as cloud deployment, automated ingestion, improved modeling, and advanced security, will help evolve the platform into a scalable, production-ready intelligence tool. Ultimately, this project demonstrates how AI and data engineering can empower organizations to better understand and respond to the needs of veterans, improving both awareness and impact.