Capstone - Sales Agent Prototype Using Multi-Agent GPT Models

short line

Poonam Kajal

29 August, 2025

# 

[Overview 2](#_Toc207402529)

[Steps 4](#_Toc207402530)

[**1.** **Technical Documentation** 4](#_Toc207402531)

[**Code Structure** 4](#_Toc207402532)

[**Design Decisions** 4](#_Toc207402533)

[**System Workflow** 4](#_Toc207402534)

[2. **Time Management** 5](#_Toc207402535)

[3. **Challenges and Solutions** 5](#_Toc207402536)

[4. **Experiments** 5](#_Toc207402537)

[Conclusion 9](#_Toc207402538)

[Appendix 9](#_Toc207402539)

# 

# 

# 

# 

# 

# 

# 

# Overview

This project created a **Sales Assistant Agent** that helps sales reps quickly understand a target company and its competitors. The system uses **Streamlit** for the interface, **Tavily Search API** to collect company and competitor information, and **AI models** (GPT-3.5, GPT-4, or Groq) to generate insights.

The sales rep enters details such as product name, company URL, product category, competitors, value proposition, and target customer. The app then produces a clear **one-page report** that includes company strategy, competitor mentions, leadership information, product/strategy summary, and links to sources.

The one-pager can be viewed in the app, downloaded as **Markdown**, or exported as a **PDF** for easy sharing.

The code is managed in a **GitHub repository**, making it version-controlled and easy to collaborate on. The working prototype was deployed on **Streamlit Cloud**, providing a public link so the tool can be accessed without installation.

The outcome is a working prototype that shows how AI can save time and provide focused, useful insights in a single page.

**GITHUB LINK:** <https://github.com/poonamkajal/SalesAgentsPrototype>

**STREAMLIT APP:** <https://poonamkajal-salesagentsprototype-app-7xsv1f.streamlit.app/>

## 

# Steps

## **Technical Documentation**

The **Sales Assistant Agent** is a Streamlit-based web application that combines **LangChain** and **LangGraph** for workflow orchestration. It integrates the **Tavily Search API** to collect real company and competitor context, and uses **large language models (GPT-3.5, GPT-4, or Groq)** to generate structured insights.

### **Code Structure**

* **app.py** – Main Streamlit interface. Handles user input (product, company, competitors, etc.) and displays the one-pager output.
* **requirements.txt** – Lists project dependencies for reproducibility.
* **.env file** – Stores **API keys and secrets** (e.g., OPENAI\_API\_KEY, GROQ\_API\_KEY, TAVILY\_API\_KEY). This keeps sensitive data out of the code and makes the app easier to configure and deploy.
* **GitHub Repository** – Version-controlled source code for collaboration and transparency.
* **Streamlit Cloud Deployment** – Public deployment that allows users to access the app without installing locally.

### **Design Decisions**

* **Streamlit** was used because it is very easy to build with and lets us quickly create a working web app.
* **Tavily Search API** was added so the AI could use real information from the web, instead of guessing, which makes answers more accurate.
* **Template rules** were used to make sure the AI always writes the report in the same one-page format with the five required sections.
* **Markdown + PDF export** was added so the final report can be easily shared, saved, or downloaded in different formats.

### **System Workflow**

1. **User Input**: Sales rep enters cpmpany name, product name, company URL, competitors, value proposition, and target customer through the Streamlit form.
2. **Research Node**: Tavily collects recent company and competitor information.
3. **Compose Node**: GPT processes inputs and research data to generate a one-pager.
4. **Template Enforcer**: Ensures consistency by structuring the output into: Company Strategy, Competitor Mentions, Leadership Information, Product/Strategy Summary, and Article Links.
5. **Output**: The one-pager is displayed in Streamlit, with options to download as Markdown or PDF.

## **Time Management**

I planned and worked on this project over several days. Each day focused on a different part of the work:

* **Day 1 – Setup (about 6 hours):** Installed the tools, set up Streamlit, LangChain, and Tavily. Tested that the AI and APIs worked together.
* **Day 2 – Core Prototype (about 8 hours):** Built the main workflow with LangGraph, connected Tavily for research, and created the Streamlit form to collect user inputs.
* **Day 3 – Testing and Improvements (about 6 hours):** Improved the prompts, added PDF export, and made sure the one-pager always follows the right format.
* **Day 4 – Documentation (about 2 hours):** Wrote the final report, cleaned up the code, and added outputs for submission.

## **Challenges and Solutions**

* **Challenge: GPT made up information (Guessing).**  
  Solution: Connected the model to **Tavily Search API** so it uses real company and competitor data instead of guessing.
* **Challenge: Installing and connecting different libraries caused version errors.**  
  Solution: Fixed this by listing all correct packages (Streamlit, LangChain, Tavily, Groq) in the **requirements.txt** file.
* **Challenge: Sharing results was limited.**  
  Solution: Added **PDF export** so reports can be downloaded and shared easily.
* **Challenge: Groq API gave authentication errors (Invalid or missing API key).**  
  Solution: Created a **.env file** to safely store the GROQ\_API\_KEY and loaded it with dotenv. This fixed the error and made it easier to switch between Groq and OpenAI keys when testing.

## **Experiments**

* **Different Models**
  + GPT-3.5: Fast and cheaper but sometimes gave shallow answers.
  + GPT-4: Slower but gave more detailed and accurate insights.
  + Groq (LLaMA): Very efficient and low cost but needed stricter prompts to get good results.
* **Prompt Styles**
  + One big prompt: Sometimes messy and missed sections.
  + Section-by-section prompts (Map-Reduce): Much better, always produced the 5 required sections in the right order.
* **Outputs**
  + Markdown was the easiest to show in Streamlit.
  + Added **PDF export** so the report could be downloaded and shared more easily.
* **Testing with Inputs**
  + Tried different companies and products (e.g., Snowflake, BigQuery, Redshift).
  + Found that Tavily search results gave stronger, more reliable reports compared to AI alone.

**Example**:

* + Product Name: Cloud Data Platform X
  + Company URL: https://www.snowflake.com/
  + Product Category: Cloud Data Platform
  + Competitors: BigQuery, Redshift, Synapse
  + Value Proposition: Real-time analytics and governed AI
  + Target Customer: Chief Data Officer – Acme Corp

1. **System Output**

The system was tested with real company inputs to show how the one-pager works. Below is an example generated output when the **target company was Snowflake**, with competitors BigQuery, Redshift, and Synapse.

A black and white text on a black background

AI-generated content may be incorrect.

A screenshot of a black and white screen

AI-generated content may be incorrect.A screenshot of a black and white website

AI-generated content may be incorrect.

# Conclusion

The Sales Assistant Agent project shows how AI and web search can be combined to make a practical tool for sales representatives. The app was built with Streamlit for the interface, LangChain for workflow design, and Tavily Search API for collecting real information from the web. Large language models such as GPT-3.5, GPT-4, or Groq were used to turn this information into a structured one-page report.

The one-pager is easy to understand and always includes the same five sections: company strategy, competitor mentions, leadership information, product or strategy summary, and article links. This format helps sales reps get a quick overview of a target company without needing to do long research on their own. The report can be previewed inside the app and downloaded as either Markdown or PDF, making it simple to share and reuse.

During the project, challenges such as model hallucinations, API key errors, and formatting issues were solved with search grounding, .env files for secrets, and template enforcement. The project was version-controlled with GitHub and deployed on Streamlit Cloud for easy access. Overall, this prototype proves that AI can save time and give sales reps focused, useful insights.

# Appendix

* Streamlit Documentation: [https://docs.streamlit.io](https://docs.streamlit.io?utm_source=chatgpt.com)
* LangChain Documentation: <https://python.langchain.com>
* Tavily API Docs: <https://docs.tavily.com>
* Groq API Docs: https://console.groq.com