Market Research & Use Case Generation Agent

Goutam Munda

April 2025

1 Introduction

The objective of the project was to build a "Market Research and Use Case Generation Agent", designed to automate the process of generating relevant AI and GenAI use cases for any given company or industry. The system searches the web for information related to a specified company name or industry vertical to understand its context, offerings, and strategic priorities.

Traditionally, identifying suitable AI applications requires a lot of manual research, domain expertise, and time. Often leading to missed opportunities or generic suggestions. The Project addresses this challenge by implementing a sophisticated multi-agent system to perform targeted market research, analyze industry specific AI trends, generate creative and relevant use cases, and identify supporting resources like code repository and datasets.

1.1 Core Functionality

The Market Research & Use Case Generation Agent system offers the following core functionalities:

- **Research Company/Industry:** Looks up up-to-date information about the given company or industry about what they do and what is important to them.
- **Finds AI Trends:** Identifies how AI or similar technologies is being used among the industry.
- **Suggests AI Use Case:** Lists specific ideas or use cases about how companies in the same industry can use and benefit from them.
- **Finds Helpful Resources:** Searches websites like Kaggle and GitHub for datasets or code examples related to the suggested AI uses.
- **Creates a Report:** Puts all the findings together into a clear Markdown report, including the best ideas and links to resources.

1.2 Technology Stack

- **Python, CrewAI, Google-Gemini, Tavily, Redis, Streamlit**

2 System Architecture

Central to the system's design is its collaborative multi-agent architecture. To effectively tackle the multifaceted challenge of market research and use case generation, we employ four specialized agents. Each agent is assigned a distinct role and task, creating a division of labor that makes the complex process of generating a comprehensive report manageable. This allows each agent to focus on a specific part of the problem – from initial web research and trend analysis to resource collection and final synthesis. The agentic system is defined below with more details about each agent, tools used and the user interface:

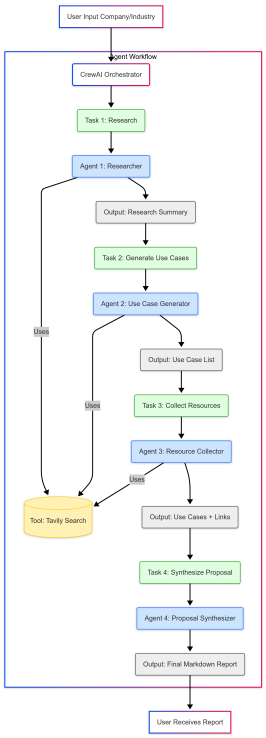

Figure 1: Simplified Flow Diagram of the system

2.1 Agents

- **Agent 1 - Researcher:** Uses the `Tavily Web Search Tool` to find company/industry context, key offerings, strategic focus areas, and business functions.
  - Role: Senior Industry Analyst
  - Goal: Understand the target's context, strategy, and operations.
- **Agent 2 - Use Case Generator:** Takes the research summary, identifies relevant AI/ML trends using web search, and generates specific use cases tailored to the target.
  - Role: AI Innovation Strategist
  - Goal: Propose relevant, formatted AI/ML/GenAI use cases based on research and industry trends.
- **Agent 3 - Resource Collector:** Takes the generated use cases and uses web search to find relevant public datasets (Kaggle, HuggingFace) and code repositories (GitHub).
  - Role: Data & Resource Scout
  - Goal: Discover publicly available datasets and code examples relevant to the proposed use cases.
- **Agent 4 - Proposal Synthesizer:** Consolidates the research, prioritized use cases, and resource links into a final, formatted report.
  - Role: Senior Proposal Manager
  - Goal: Create a clear, professional, and actionable final report in Markdown.

2.2 Tools

Access to up-to-date information is crucial for effective market research, making web search a core requirement for the system. To facilitate this, the agents responsible for research, use case generation, and resource collection are equipped with a specialized web search tool.

The Tavily Search API was selected for this purpose. Tavily is a ready to use third-party API to provide LLMs the capability of web searching. Setting up Tavily saved the time to create a web search tool from scratch.

Furthermore, to optimize costs and performance, a custom caching layer was implemented using Redis. Before making a live call to the Tavily API, the tool checks the Redis cache for results corresponding to the exact same query. If a valid, non-expired result exists in the cache, it is returned directly, bypassing the API call. This significantly reduces the consumption of Tavily credits, especially during development, testing, or when running similar queries repeatedly, while also speeding up response times for cached searches.

2.3 User Interface (UI) - Streamlit

We used Streamlit to build a simple interface for the system. This interface lets users easily try out the report generation process.

A user just needs to type in the name of a company or industry they are interested in and click the `'Generate Report'` button. While the agents work in the background to research and write the report, the user sees a loading message.

Once the report is ready, it appears directly on the screen within the Streamlit app. Below the report, buttons allow the user to download the full report either as a Markdown file (.md) or as a PDF file (.pdf), whichever format they prefer. Helper functions were used behind the scenes to clean up the report text and create the PDF version for download.

3 Setup and Usage

1. **Clone the Respository:** git clone https://github.com/goutam-kul/market-research-agent
2. **Install & Run Redis:** The backend requires a running Redis instance for session management. Please refer to this page of installation guide: https://redis.io/docs/latest/operate/oss_and_stack/install/install-redis
3. **Install Dependencies:** Install the required Python packages using pip:
   `pip install -r requirements.txt`
4. **Configure .env file:** Create a .env file in root directory and setup these variables:
   - `GEMINI_API_KEY=YOUR_API_KEY`
   - `TAVILY_API_KEY=YOUR_API_KEY`
   - `REDIS_PASSWORD` (Leave blank if none, or add your password)
5. **Run the application:** `streamlit run src/app.py`