**Project Report**

**Market Research & Use case Generation Agent**

*Submitted by*

**Selva Tharun R R**

*Submitted on*

**14-04-2025**

**INTRODUCTION**

In today’s fast-paced industries, companies need to adopt Artificial Intelligence (AI), Machine Learning (ML), and Generative AI (GenAI) to remain competitive. However, discovering relevant use cases, tools, and datasets is time-consuming and often manual. This project solves that by automating the entire AI use case research pipeline using a multi-agent system built on LangChain, Google Gemini, and Tavily, with a frontend powered by Streamlit.

**OBJECTIVE**

To develop a multi-agentic system that, given any company or industry name, automatically:

* Performs web-based industry research
* Generates AI/ML and GenAI use cases
* Collects related datasets and open-source projects
* Suggests GenAI tools such as document search, chatbots, or auto-report generators

**TOOLS AND TECHNOLOGIES**

1. LangChain (agent orchestration)
2. Gemini 1.5 Pro (LLM)
3. Tavily API (real-time web search)
4. Streamlit (UI)
5. Kaggle, Hugging Face (dataset search)
6. Python (overall development)

**SYSTEM ARCHITECTURE**

**AGENTS :-**

1. **Industry Research Agent**

Input : Industry or company name

Action : Uses Tavily API to search the web

LLM : Summarizes key findings

1. **Use Case Generator Agent**

Input : Summary from research agent

Output : Realistic, categorized AI/ML/GenAI use cases

1. **Resource Collector Agent**

Input: Use case titles

Output: Links to related datasets from Kaggle & HuggingFace

1. **GenAI Tool Recommender Agent**

Input: Industry summary

Output: GenAI-based internal and external solution proposals (e.g., chatbots, RAG systems)

****

**METHODOLOGY**

1. The app starts with a user-entered industry (e.g., “Retail”)
2. Tavily fetches live information which Gemini interprets
3. LangChain orchestrates the flow between agents
4. Each agent focuses on a specific domain (research, generation, resources, solution ideas)
5. The frontend shows results in real time