# Capstone Project Report

Building a Simple ReAct Agent with Web Search and Weather Tools

Janhavi Naik

January 16, 2026

# 1 Introduction

## 1.1 What is the ReAct Framework

ReAct (Reasoning and Acting) is a framework that combines logical reasoning with the ability to perform actions using external tools. Instead of directly answering every question, the agent can decide whether it needs additional information and then call tools such as web search or weather APIs.

The ReAct framework follows a structured loop:

- Think about the problem
- Decide an action (use a tool or answer directly)
- Observe the tool output
- Produce a final answer

## 1.2 Why Combining Reasoning and Acting is Powerful

Combining reasoning with tool usage allows the agent to:

- Answer real-world, up-to-date questions
- Avoid hallucinating facts
- Dynamically decide the best way to solve a query

This makes the agent more reliable and practical than a standalone language model.

# 2 Environment Setup

## 2.1 Required Libraries and API Keys

The following Python libraries were used:

- langchain-core
- langchain-community

- langchain-groq
- tavily
- requests
- python-dotenv

The following API keys were required:

- Groq API Key (LLM backend)
- Tavily API Key (Web Search)
- OpenWeather API Key (Weather data)

All keys were stored securely using a `.env` file.

### 2.2 LLM Initialization Test

The Groq LLM was initialized using LangChain.

## 3 Agent Design

### 3.1 Prompt Template Used

The agent was guided using a ReAct-style prompt that instructed it to:

- Think step-by-step
- Decide whether a tool is required
- Call the appropriate tool if needed
- Return a final answer

The agent could choose between:

- Web Search Tool
- Weather Tool
- Direct response (no tool)

### 3.2 Decision-Making Rules

- Factual or common knowledge questions are answered directly
- Real-time or external information triggers tool usage
- Weather-related queries always invoke the Weather tool
- News or information-based queries invoke the Web Search tool

# 4 Tool Development

## 4.1 Web Search Tool

The Web Search tool was implemented using the Tavily API.

**Input:** Search query string **Output:** Top 2–3 search results including title and snippet

The tool retrieves recent and relevant information and returns summarized results, allowing the agent to reason over them before answering.

## 4.2 Weather Tool

The Weather tool was implemented using the OpenWeatherMap API.

**Input:** Location and time (e.g., "Mumbai today", "Boisar after 2 days") **Output:** Current or predicted weather conditions

The tool supports:

- Current weather
- Forecast up to 5 days

Basic natural language parsing was applied to extract city names from user queries.

# 5 Agent Loop

## 5.1 Decision–Action–Observation Flow

The agent follows this loop:

1. User asks a question
2. Agent reasons about the query
3. Agent selects a tool or answers directly
4. Tool returns an observation
5. Agent produces the final response

## 5.2 Example Interactions

**Example 1: Weather Query**

- User: What is the weather in Gujarat today?
- Action: Weather Tool
- Output: Current weather details

**Example 2: Knowledge Query**

- User: What is the color of apple fruit?

- Action: Web Search Tool
- Output: Summarized factual explanation

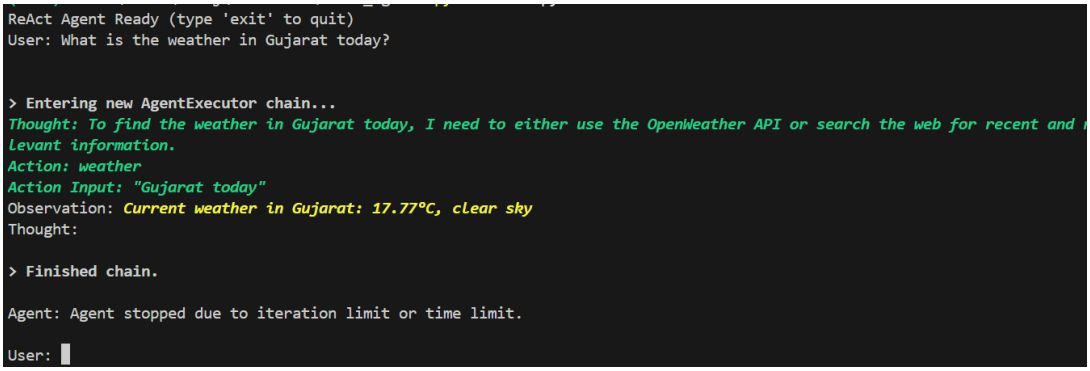


Figure 1: Weather Tool Interaction Log

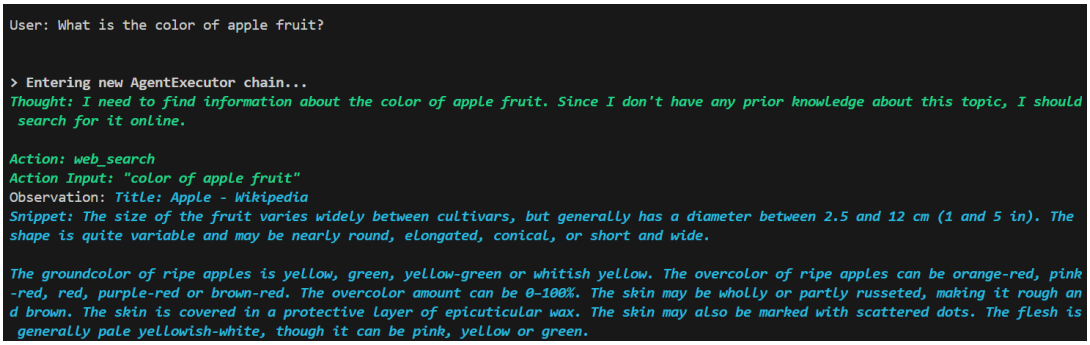


Figure 2: Web Search Tool Interaction Log

# 6 Conclusion

## 6.1 Key Learning Outcomes

Through this project, the following concepts were learned:

- How ReAct agents work
- Tool integration with LLMs
- API-based real-time data retrieval
- Error handling and fallback reasoning

## 6.2 Potential Extensions

Future improvements could include:

- Calculator tool for numerical queries
- Database lookup tool

- Multi-language support
- Improved natural language parsing