# AI Workflow for Multi-Agent Automation

## Overview

This project implements a multi-agent workflow using Python and Streamlit to automate tasks such as:   
1. Fetching research data about companies and industries using the Serper API.  
2. Generating AI-based use cases and descriptions with Hugging Face models.  
3. Presenting the workflow in a user-friendly Streamlit web app.  
  
The workflow can serve businesses, researchers, and developers who need actionable insights based on company and industry data.

## Code Explanation

### app.py  
This is the main Streamlit application that integrates the components and serves as the entry point.

Key Components:  
- Research Agent: Uses Serper API to fetch data.  
- Use Case Agent: Generates use cases using Hugging Face Transformers.  
- Utility Functions: Contains helper functions for efficient processing.  
  
Alternatives: Flask or Django for backend processing, Gradio for UI.

### src/research\_agent.py  
Fetches data using Serper API.

Key Steps:  
- Constructs a search query.  
- Sends the query to the Serper API.  
- Returns industry and company-related data.  
  
Alternatives: Google Custom Search API, Bing Search API.

### src/use\_case\_agent.py  
Generates use cases and detailed descriptions.

Key Steps:  
- Uses Hugging Face BART summarizer for text processing.  
- Returns concise use cases and descriptions based on input data.  
  
Alternatives: OpenAI GPT models, T5 Transformer models.

### src/utils.py  
Contains reusable utility functions for formatting and error handling.

## Libraries Used

1. Streamlit: For creating a web-based interface.  
2. Transformers: For text summarization and natural language processing.  
3. Requests: For API integration.  
4. JSON: For parsing API responses.  
  
Alternatives: Dash (instead of Streamlit), Flask (for backend APIs).

## Serper API

The Serper API is used to fetch real-time data from Google search results.   
Steps to Use:  
1. Sign up at serper.dev.  
2. Generate an API key and add it to `config.py`.  
3. Use it in `research\_agent.py` for sending queries and receiving responses.  
  
Alternatives: Bing API, Google Custom Search API.

## Hugging Face Transformers

The project uses the BART model for summarization tasks.  
Why BART?  
- Pre-trained for summarization tasks.  
- Generates coherent and concise text.  
  
Alternatives: GPT-3, T5, or fine-tuned models for domain-specific tasks.

## Streamlit Integration

Streamlit provides an easy-to-use framework for developing data applications.  
Key Features:  
- Interactive interface for multi-agent automation.  
- Real-time output display.  
  
Alternatives: Gradio (similar simplicity), Flask or Django (more flexibility).

## Project Setup and Deployment

1. Clone the repository: `git clone <repository\_url>`.  
2. Set up a virtual environment and activate it.  
3. Install dependencies: `pip install -r requirements.txt`.  
4. Run the app: `streamlit run app.py`.  
5. Access the app at `http://localhost:8501`.

Deployment Options:  
- Streamlit Sharing  
- Heroku  
- Docker

## Conclusion

This project demonstrates the power of multi-agent automation for generating actionable insights.  
Future Improvements:  
1. Enhance UI/UX with additional frameworks.  
2. Extend API support for diverse use cases.  
3. Add more robust error handling.