Take-Home Assignment: GenAI News Assistant MCP Server

1. Introduction

Welcome! This take-home assignment is designed to help us understand your skills in backend development, integrating with external APIs, applying generative AI, and working with specific communication protocols relevant to our systems.

We appreciate you taking the time to complete it.

The goal is to build a backend service that acts as a "Tool Server" conforming to the Model Context Protocol (MCP). This server will provide tools for fetching and analyzing news information using a third-party API and a Large Language Model (LLM).

We estimate this task should take approximately **4 hours of focused** work. Please reach out if you anticipate needing significantly more time.

We know this assignment requires a notable time investment, and we sincerely appreciate your effort. A practical task like this is crucial for us to fairly evaluate the skills essential for the role, and we truly value the time you're dedicating.

2. The Core Task: Build an MCP Server

You will build a backend MCP server that listens for requests conforming and executes specific "tools" based on those requests. **The server should be containerized using Docker**. We have supplied a repository with two Docker files containing all you need for either a Python or TypeScript setup. Please make sure that you test that it is working.

3. Technology Stack & Setup

- **Languages:** You can use either **Python** or **TypeScript**.
- **Template Repository:** https://github.com/latentsp/redesigned-octo-engine provides a template repository containing:
  - Basic project structure for both Python and TypeScript.
  - A Dockerfile for containerization.
  - A basic README template for you to fill out.
- **Dependencies:** You are free to use standard libraries for your chosen language.

4. External API: NewsAPI.org

This assignment requires integrating with a third-party news service.

- **API:** Please use **NewsAPI.org**.

- **Signup:** You will need to register for a **free API key** on their website: https://newsapi.org/
- **Documentation:** Familiarize yourself with their API documentation, particularly the /v2/everything and /v2/top-headlines endpoints: https://newsapi.org/docs
- **Configuration:** Your application should expect the NewsAPI.org API key to be provided via an environment variable named **NEWS_API_KEY**.
- **Rate Limit Awareness:** Please be mindful of the rate limit. You are hard-capped at 100 requests per day, consider caching responses while testing or employing a TDD methodology with injected data.

5. LLM API Access

The assignment involves using a Large Language Model for analysis.

- **API Key:** We will provide you with an API key for OpenAI shortly after you receive this assignment. This key will have a small pre-paid budget for you to use.
- **Budget Awareness:** Please be mindful of the budget. Use the LLM API calls efficiently, especially for the tools involving LLM analysis. Test prompts and logic where possible without making excessive API calls.
- **Configuration:** Your application should expect the LLM API key via an environment variable named **LLM_API_KEY**.

6. ModelContextProtocol (MCP)

As mentioned, your server must communicate using **MCP in SSE transport**. MCP allows client applications (Claude Desktop) to discover and execute specific functionalities (tools) exposed by backend services.

- **Specification:** The definitive specification for MCP, including the required JSONRPC structures for requests and responses is available at https://modelcontextprotocol.io/introduction
- **Focus:** Your primary task is to implement the *logic* for the tools defined below, ensuring they correctly parse MCP requests and return properly formatted MCP responses according to the spec.

7. Required MCP Tools to Implement

You need to implement the server-side logic for the following tools, making them available via the MCP interface. Do *not* implement the steps themselves in this document, only define the goals, inputs, and outputs.

Tool 1: `search_news`

- **Goal:** To search for recent news articles matching a specific query.

- **MCP Input (Example):**

  `query` - Search news query. Example: latest OpenAI model

  `language` - News language. Example: en

  `pageSize` - Amount per page. Example. 5

- **MCP Output (Success Example):**

```
{
  "articles": [
   {
    "title": "New AI Model Released by...",
    "description": "A brief description of the article...",
    "url": "https://example.com/article1",
    "source_name": "Example News Source",
    "published_at": "2025-04-08T10:00:00Z"
   }
   // ... more articles up to pageSize
  ]
}
```

Tool 2: `extract_information_from_article`

- **Goal:** To fetch the single *latest* news article matching a query and then extract specific structured information (like people, organizations, locations, key quotes) from its content (title and description/snippet) using an LLM. The goal is not to parse the information, just return the JSON response.
- **MCP Input (Example):**
  `query` - Search news query. Example: bitcoin
  `language` - News language. Example: en

- **MCP Output (Success Example):**

```
{
 "result": {
  "fetched_article_title": "Global Tech Summit Announces Major Partnership", // Title of the article processed
  "people": ["Jane Doe", "John Smith"],
  "organizations": ["TechCorp Inc.", "Innovate Solutions", "Global Tech Summit"],
  "locations": ["Metropolis"],
  "key_quotes": ["This collaboration marks a new era"]
 }
```

}

Tool 3: `extract_key_info_and_sentiment`

- **Goal:** To analyze recent news articles about a topic (obtained via their titles/descriptions fetched using a query) to extract key entities and determine the overall sentiment, using an LLM. This tool requires server-side GenAI processing using prompt engineering.
- **MCP Input (Example):**
  `query` - Search news query. Example: renewable energy investment trends
  `language` - News language. Example: en
  `max_articles_to_analyze` - Maximum articles to analyze. Example: 5
- **MCP Output (Success Example):**

```
{
 "status": "success",
 "result": {
  "query": "renewable energy investment trends",
  "analyzed_article_count": 5, // Actual number of articles processed
  "overall_sentiment": "Positive", // e.g., Positive, Negative, Neutral
  "sentiment_confidence": "Medium", // e.g., Low, Medium, High
  "key_entities": {
   "people": ["Person X", "Person Y"],
   "organizations": ["Company A", "Investment Fund B"],
   "locations": ["Region C"]
  },
   "key_takeaway_summary": "LLM generated 1-2 sentence summary highlighting the main points or findings from the analyzed articles."
 }
}
```

8. Deliverables

Please submit your solution by providing:

1. A link to a private Git repository (e.g., GitHub, GitLab) containing your complete source code. Please invite `gmaliar` to the repository.
2. Ensure the repository includes an updated README.md file with:
   - Clear instructions on how to build and run your Docker container.
   - Do not include your environment variables in the repository.

9. Evaluation Criteria

We will evaluate your submission based on the following criteria:

- **Correctness & Functionality:** Does the server correctly implement the MCP protocol? Do the tools work as specified, interacting correctly with NewsAPI.org and the LLM?
- **We will test your application in a Claude Desktop application using our `remote-mcp` tool (https://github.com/latentsp/remote-mcp).** It is crucial that you create an MCP server that uses the SSE transport and that the Docker container is working properly.
- **Code Quality:** Is the code well-structured, readable, maintainable, and idiomatic for Python/TypeScript? Is error handling robust?
- **Prompt Engineering:** (Specifically for `extract_information_from_article` and `extract_key_info_and_sentiment`) How effective and efficient are the prompts designed to instruct the LLM for the required analysis and structured output?
- **API Integration:** Correct usage of the NewsAPI.org endpoints and parameters.
- **Documentation:** Clarity of the README file regarding setup and design choices.

10. Academic Honesty

We expect the final submission to represent your problem-solving effort and understanding. We recognize that modern development workflows often include AI coding assistants (e.g., GitHub Copilot, Cursor), and using these tools is permitted for this assignment. However, your submission should demonstrate thoughtful integration, adaptation, and a clear grasp of the underlying code and logic. Please ensure the final work reflects your ability to build and reason about the solution, not just assemble generated snippets (à la vibe-coding).

11. Questions & Deadline

- **Deadline:** Please submit your solution by the specified date in the email.
- **Questions:** If you have clarifying questions about the requirements, please feel free to contact us. We aim to provide clarifications but cannot provide help in solving the task itself.

We look forward to seeing your solution!