CORIOLIS TECHNOLOGIES

Automated Task Orchestration Using LLMs

Authored By:
Bhuvnesh Sahu

Table of Contents

Acknowledgement

I would like to sincerely thank **Mr. Besant Rajan**, CEO of Coriolis Technologies, for giving me the opportunity to work on this industry project.

I am deeply grateful to **Mr. Sudhir Kumar** and **Mr. Rohan Nandode** for their continuous mentorship and guidance throughout this journey, especially during moments of challenge and saturation. I am also thankful to my peers and the entire Coriolis team for their collaboration, encouragement, and feedback that helped shape this work.

This internship has been a valuable experience, giving me the opportunity to expand my knowledge, explore new concepts, and apply my learning to real-world problems.

Abstract

This project presents an intelligent assistant powered by Large Language Models (LLMs) that automates complex, multi-step tasks and enables users to interact with unfamiliar systems through simple natural language queries. The assistant interprets user-defined tasks, maps them to the system's API documentation, and generates executable scripts (e.g., curl or Python commands). It is designed to support both technical and non-technical users, enabling seamless task execution without prior system knowledge. The solution has been tested across multiple platforms—including EigenServ, GitLab, and the Coriolis AI App—and shows promising results with large LLMs such as GPT-4o-mini and LLAMA3-70B, Phi-4-7B, Mistral 7B.

Introduction

In DevOps and system administration, users often perform repetitive tasks such as configuring virtual machines, initiating builds, or querying logs. These tasks require knowledge of API specifications and scripting, which creates friction for two key user groups:

Clients and Non-Technical Users

While they know *what* they want to achieve (e.g., "increase RAM of a VM"), they often lack knowledge of *how* to perform the task within the system. This leads to over-reliance on manuals or support engineers, creating bottlenecks and increasing support overhead.

Developers and DevOps Engineers

Even technical users must repeatedly write scripts for routine operations by consulting fragmented documentation and manually constructing API calls. This reduces productivity and introduces scope for errors.

To solve these challenges, we built a task orchestration system using Large Language Models (LLMs) that allows users to describe their goals in plain English. The system reasons over structured API documentation (Swagger/OpenAPI JSON) to identify the required steps and either:

- **Provides step-by-step guidance for clients**, minimizing dependency on support teams.
- **Generates complete automation scripts for developers**, enabling fast and accurate task execution.

This unified solution reduces friction, increases productivity, and democratizes access to complex systems.

Problem Definition

This project addresses two core problems in infrastructure management and automation:

1. **Guided Task Execution for Non-Technical Users**
    - Clients often understand the outcome (e.g., "increase RAM") but not the technical process.
    - Documentation is fragmented, outdated, or overly technical for novice users.
    - This results in frequent support queries, increasing overhead and slowing resolution.
2. **Automation Support for Developers and Technical Teams**
    - Developers must repeatedly script similar workflows (e.g., VM management, CI/CD configuration).
    - These tasks require manually navigating API references and composing scripts.
    - This leads to inefficiencies, increased error rates, and wasted engineering effort.

Architecture Overview

The system architecture consists of the following components:

1. User Interface (Streamlit App): Accepts task descriptions in natural language.
2. Knowledge Base: Structured API documentation (Swagger/OpenAPI JSON).
3. LLMs (e.g., GPT-4o-mini, LLAMA3, Phi-2): Understand task intent and map to APIs.
4. Script Generator: Converts the API calls into executable `curl`, shell, or Python commands.

System Workflow

Workflow 1: Client-Side Task Guidance (Step-by-Step Instruction Generation)

This workflow is designed to assist non-technical users by generating a clear sequence of API steps they need to follow, without requiring direct scripting or execution.

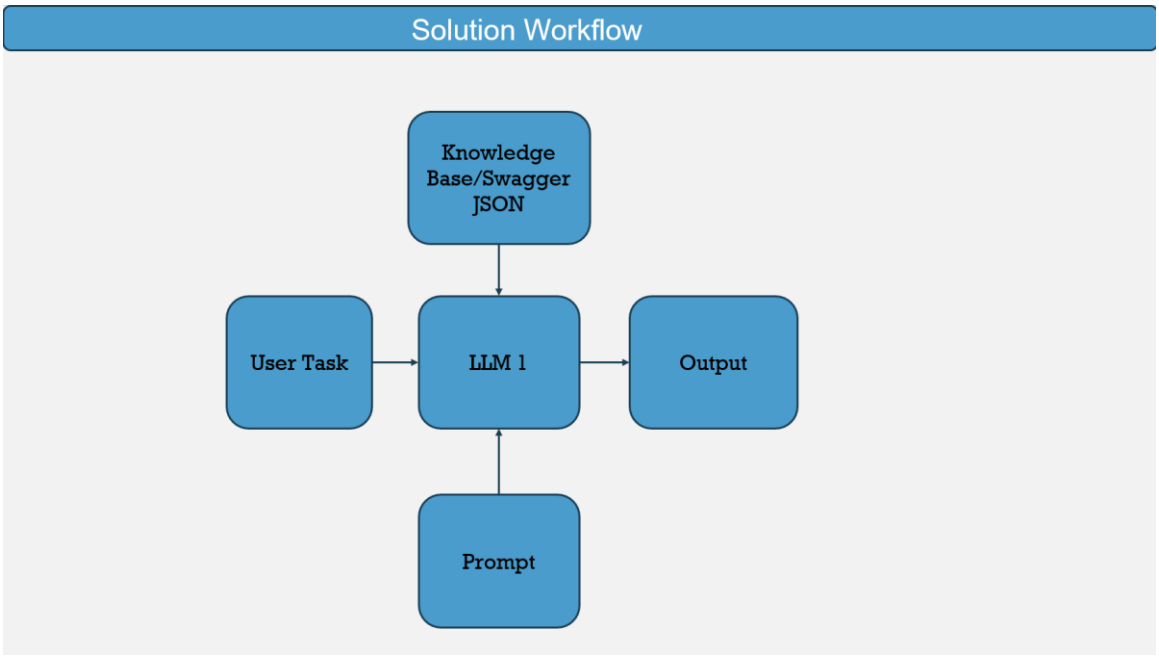
Step 1: Task Description Input
The user provides a plain-English query (e.g., "List all machines with server name 'es3' and then logout").

Step 2: Provide Swagger JSON
Provide the Swagger JSON of the system, you are working with

Step 3: Intent Interpretation
The LLM analyzes the input to understand the goal and decomposes it into actionable subtasks.

Step 4: API Sequence Planning
Using the Swagger/OpenAPI knowledge base, the assistant identifies the relevant endpoints, request structure, and execution order.

Step 5: Instruction Generation
The assistant outputs a human-readable step-by-step guide with mapped API calls (method, endpoint, payload, sequence).

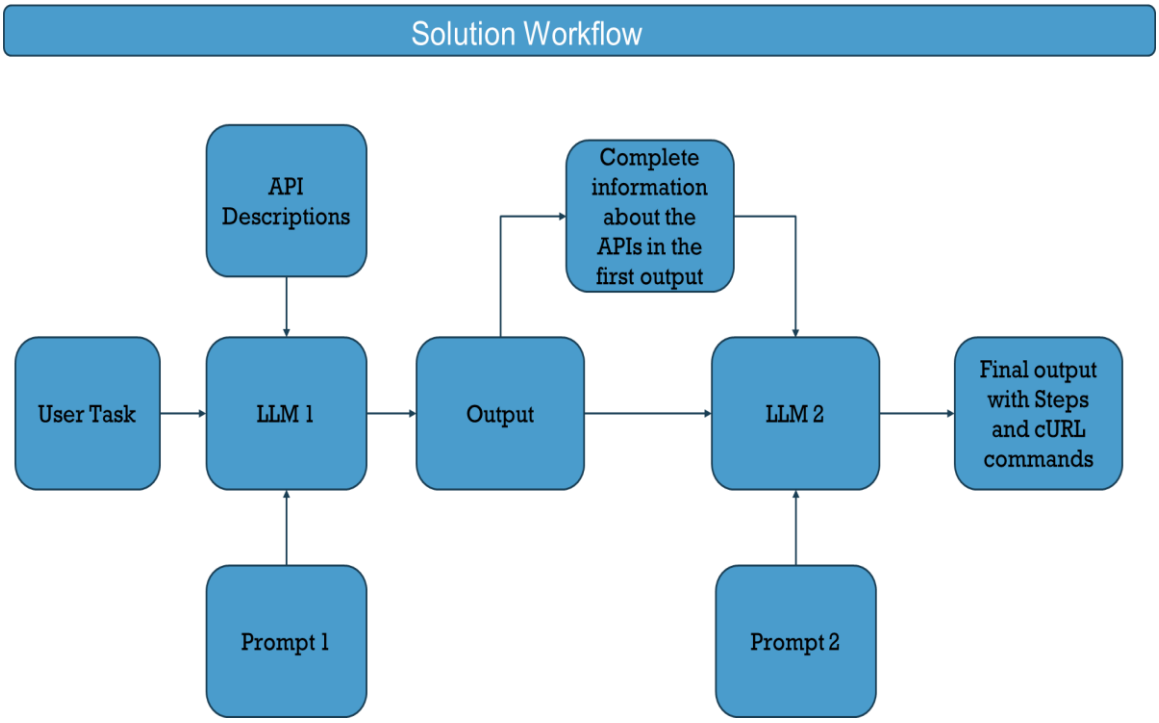


Workflow 2: Automation Support for repetitive Tasks (Script Generation)

This workflow is targeted at developers or DevOps engineers who require fully automated command-line scripts.

1. **Step 1: Task Description Input**
   The user provides a plain-English query (e.g., “List all machines with server name ‘es3’ and then logout”).
2. **Step 2: Provide Swagger JSON**
   Provide the Swagger JSON of the system, you are working with
3. **Step 3: Intent Interpretation**
   The LLM analyzes the input to understand the goal and decomposes it into actionable subtasks.

4. **Step 4: API Sequence Planning**
   Using the Swagger/OpenAPI knowledge base, the assistant identifies the relevant endpoints, request structure, and execution order.
5. **Step 5: Script Construction**
   It generates ready-to-run curl, Python, or shell scripts that perform the task end-to-end.



Knowledge base/Database

The **Knowledge Base** acts as the core reference for the assistant to understand what operations are possible within a target system. It is built using the **Swagger (OpenAPI) JSON file**, which documents all available API endpoints and their specifications.

This file contains:

- **Endpoint URLs** for all available API operations
- **Required HTTP methods** (GET, POST, PUT, DELETE, etc.)
- **Input parameters**, request bodies, and data types
- **Expected responses and status codes**
- **Authentication requirements**, if applicable

The assistant uses this structured documentation to:

- **Understand the capabilities** of the underlying system
- **Map user-defined tasks** to available API calls
- **Generate executable scripts** dynamically using only the information in the Swagger file

Results & Performance

The system was tested with several LLMs:

- GPT-4o-mini
- LLAMA3 70B
- Phi 4 7B
- Mistral 7B

Observations:

- High accuracy in mapping tasks to API workflows
- Enabled non-technical users to complete technical tasks
- Reduced dependency on documentation and manual scripts
- Lower latency with optimized, smaller models

Future Work

- Fine-tuning smaller LLMs (Mistral, TinyLLM) for specific domains
- Multi-turn dialogue handling for step refinement
- Feedback loop for correction and learning
- On-prem deployments for enterprises using Docker

Tools Used

- LLMs: GPT-4o-mini, LLAMA3, Phi-4, Mistral
- APIs: Swagger, GitLab API, EigenServ API
- Languages: Python
- Frameworks: Streamlit, Docker