B.Tech 6th Semester Minor project Presentation

“Octopus - Your AI-Powered Digital Company”

Presented by: -

Anarv Vasavada(2203051051122)

Meet Patel(2203051050757)

Vedant Jha(2203051050870)

Kevin Patel(2203051050755)

Supervisors:

Mr. P. Rajakumar

Content of Presentation

Abstract

Octopus is a secure, AI-powered orchestration platform designed to function as your digital company, fundamentally redefining how complex projects are executed. It brings together a coordinated team of specialized AI and API agents that operate like virtual employees, each handling distinct responsibilities while working collaboratively toward a common goal. Starting from a single natural language instruction, Octopus intelligently parses the request, plans an optimized execution strategy, and seamlessly delegates tasks across its multi-agent ecosystem. By unifying natural language understanding, autonomous task planning, and containerized agent collaboration into a single cohesive system, Octopus eliminates the inefficiencies of fragmented workflows and tool silos. This architecture empowers innovation teams to automate sophisticated, multi-step projects end-to-end, achieving new levels of productivity, clarity, and operational oversight, all while maintaining secure and isolated execution environments.

Introduction

What is Octopus?

- Octopus is a **multi-agent AI collaboration platform** that transforms high-level human instructions into autonomous, secure execution across different tools and APIs.
- Unlike traditional tools that require manual coordination, Octopus acts like your own **digital company**, where agents handle different domains ,docs, code, sheets, communication seamlessly.
- A single instruction like

  "Create a Notion doc for Q3 goals, update Google Sheets, raise GitHub issues, and notify Slack**"**

  gets fully executed without manual hand-offs.

Front-End Technology

• React.js, Tailwind CSS, HTML, Javascript

Back-End Technology

• FastAPI / Node.js, Docker, Redis, ChromaDB / Weaviate, Machine Learning, NLP, VectorDB

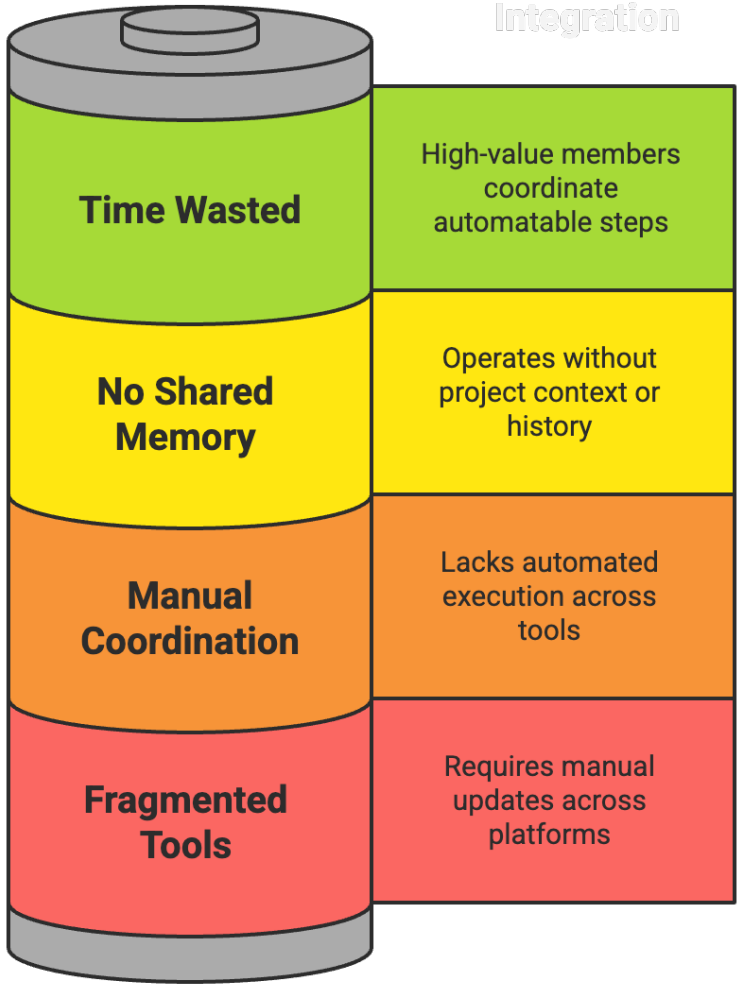
Aim:

To build a multi-agent AI infrastructure that replaces fragmented tools and manual coordination with secure, intelligent automation.

Problem Statement

THE PROBLEM:

- **Fragmented Tools:** Teams rely on multiple disconnected platforms like GitHub, Notion, Google Sheets, and Slack, requiring manual hopping and updates.
- **Manual Coordination:** No single system breaks down a big-picture instruction and auto-executes across these tools, causing delays and errors.
- **No Shared Memory:** Each tool works in isolation, with no global understanding of project context or history.
- **Time Wasted on Micromanagement:** High-value team members waste time coordinating steps that could be automated.



Proposed Methodology

1. NLP-Based Instruction Parsing

- Uses spaCy + scikit-learn to turn natural language commands into structured task trees with dependencies.

2.Task Orchestration Engine

- Prioritizes tasks, plans parallel vs sequential execution, and routes them to the right agent pods.

3. Containerized Multi-Agent Pods

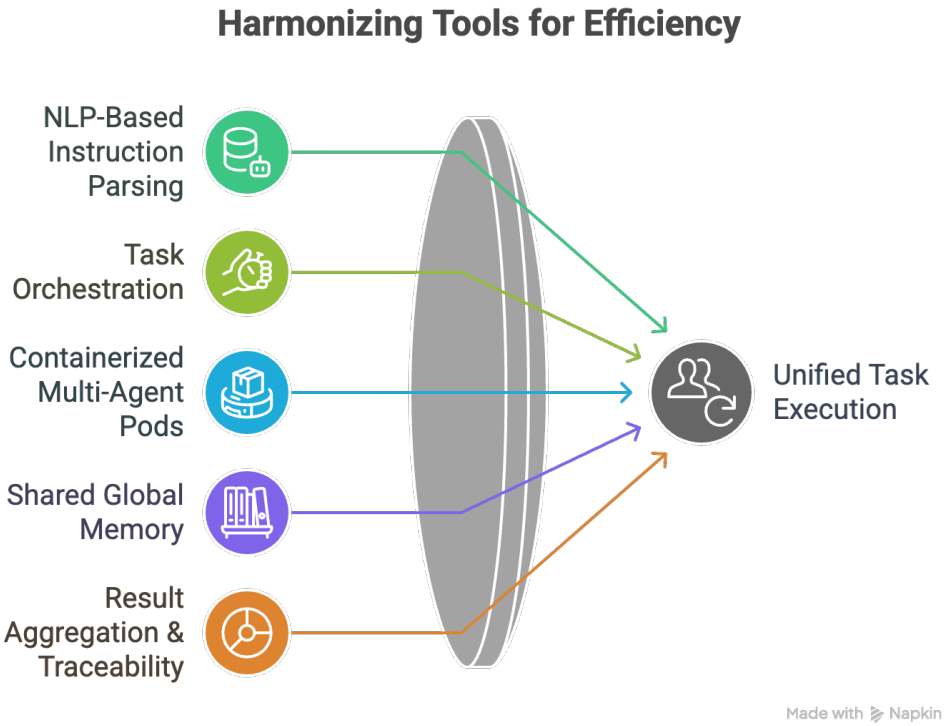
- Runs each agent (Notion, GitHub, Slack, Sheets) in secure, isolated Docker containers.

4. Shared Global Memory

- Employs Redis or a vector DB to track task history, intermediate results, and context.

5. Result Aggregation & Traceability

- Merges outputs into a final result with clear logs for easy debugging and review.



Project Module

1. **Instruction Parser:**

• Interprets natural language commands and breaks them down into a structured task tree with clearly defined sub-tasks and dependencies, ensuring nothing gets missed or misinterpreted.

2. **Task Orchestrator:**

• Analyzes the task tree to build the most efficient execution sequence, intelligently deciding which tasks can run in parallel and which need to be completed sequentially for smooth workflows.

3. **Agent Pods:**

• Dedicated, containerized agents connect to tools like Notion, GitHub, Slack, and Google Sheets. Each operates independently, executing tasks specific to its domain, just like specialized employees in a company.

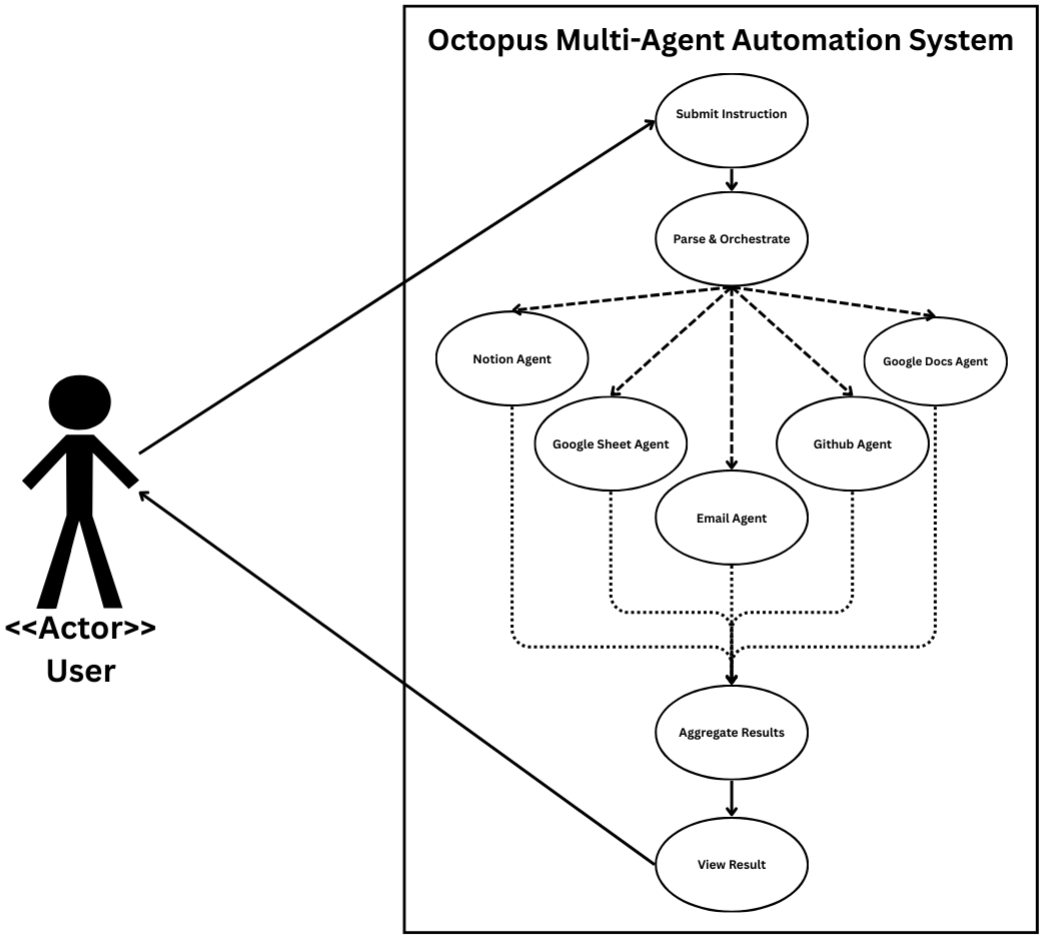
4. **Shared Memory Layer:**

• Acts as a global knowledge base that stores essential data such as document links, sheet IDs, or pull request numbers. This allows agents to share context and build on each other's outputs without redundant work.

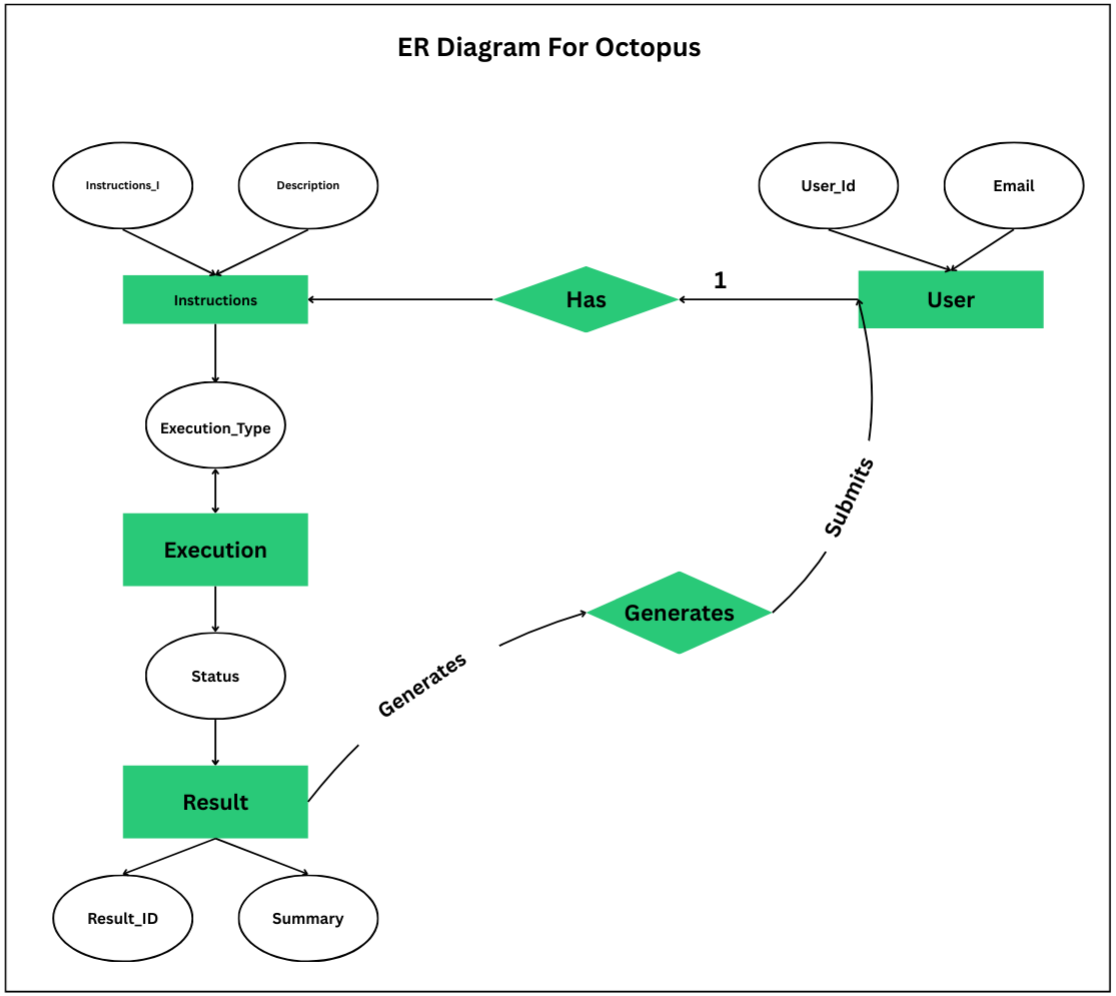
5. **Result Viewer:**

• Compiles all outcomes into a clear summary, accompanied by a concise execution log that shows step-by-step what was done, giving users full transparency and confidence in the process.

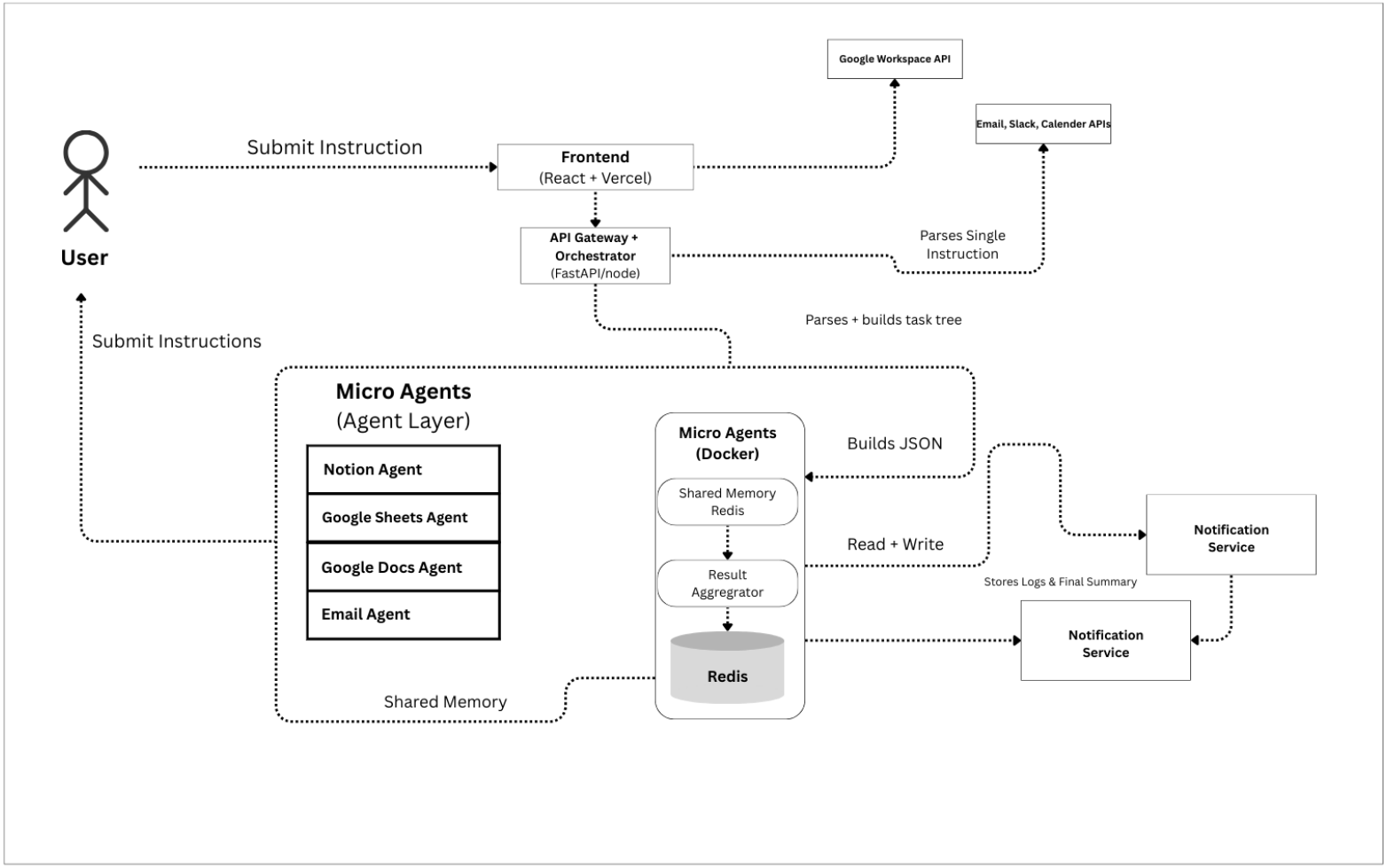
Use-Case Diagram



E-R Diagram


ER Diagram For Octopus
Instructions_I
Description
User_Id
Email
Instructions
Has
1
User
Execution_Type
Execution
Submits
Generates
Generates
Status
Result
Result_ID
Summary

Dataflow Diagram

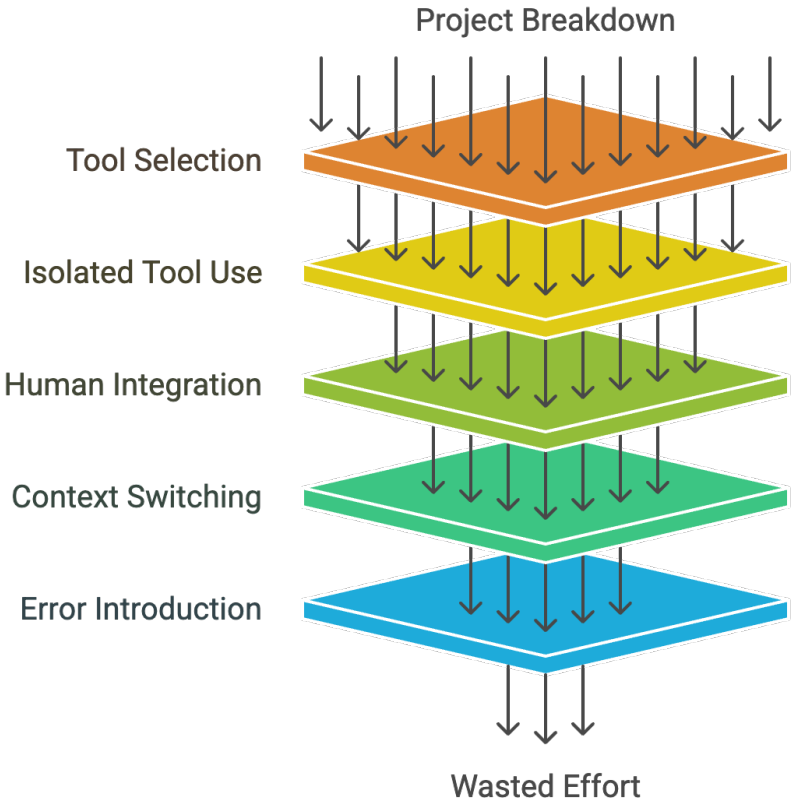


Difference Between Traditional Tools & Octopus

Traditional Tools	Octopus Platform
Use separate, disconnected tools for documents, code, and communication, leading to fragmented workflows.	Provides a unified multi-agent execution system where all tasks are coordinated within a single platform.
Require manual breakdown of projects and assignment of tasks to different people or tools.	Automatically parses instructions and orchestrates tasks across specialized agents, reducing human intervention.
Have no shared memory, so each tool operates in isolation without context from others.	Maintains a global memory across all agents, allowing shared context and continuity throughout the workflow.
Demand more human effort to link outputs between tools, often requiring copy-paste or manual syncing.	Agents coordinate and pass outputs directly to each other, ensuring seamless, automated handoffs.

Traditional vs Octopus Collaboration Process

Traditional Project Workflow Inefficiency



Made with Napkin

Octopus Workflow Process

- User Instruction
- NLP Parsing
- Task Orchestration
- Agent Assignment
- Task Execution
- Inter-Agent Coordination
- Report Aggregation

Made with Napkin

How Octopus is Better?

1. Single Instruction → Full Workflow:

One natural language input triggers the entire process no manual splitting or tool-hopping needed.

2. Specialized Agents:

Each agent handles its domain like an expert employee, boosting speed and precision.

3. Shared Memory:

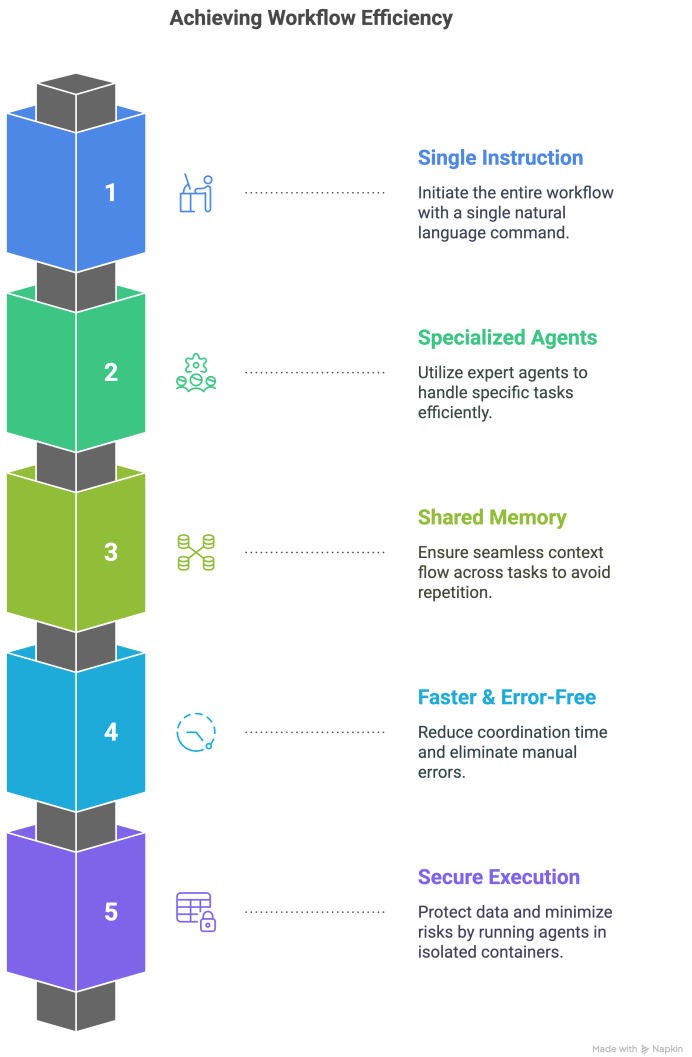
Context flows across tasks automatically, avoiding repeated setups and data entry.

4. Faster & Error-Free:

Cuts human coordination time and eliminates common manual mistakes.

5. Secure Execution:

Agents run in isolated containers, protecting data and minimizing risks.



Octopus - Project Modules Overview

1. NLP Instruction Parser:

Converts natural language commands into structured task plans, fully understanding even complex inputs and breaking them into actionable steps.

2. Task Orchestrator:

Determines optimal execution flows, identifies parallel vs sequential tasks, and assigns work to the right agents for maximum speed and efficiency.

3. Agent Pods:

Containerized services dedicated to domains like Notion, GitHub, Slack, or Sheets, each executing tasks independently like specialized team members.

4. Shared Memory:

Stores global context, task history, and intermediate data so agents work with full awareness, avoiding duplication and errors.

5. Result Aggregator & UI:

Compiles all outputs into a clear summary with live progress logs, giving users complete visibility and confidence in the process.

Hardware & Software Requirements

Hardware Stack:

• **Hardware:**

Laptop or PC with at least a dual-core 2.5 GHz processor, 8 GB RAM, and an SSD for smooth operation.

• **Operating System:**

Windows, macOS, or Linux.

• **Browser:**

Latest versions of Chrome, Edge, or Firefox.

• **Network:**

Stable internet connection with a minimum speed of 5 Mbps.

Software Stack:

- Docker for containerized deployment.
- Redis or a vector database for shared memory and context management.
- FastAPI or Node.js as the backend framework.
- Integrations:
- Basic API keys set up for Notion, GitHub, and Slack to enable agent operations.

Expected Outcome of Project

1. **Streamlined Workflows:**

Octopus automates entire multi-step processes that would normally require hours of manual effort, stitching together tasks across documentation, code, and communication tools into one smooth, hands-off flow.

2. **Reduced Manual Oversight:**

With Octopus handling planning, execution, and inter-tool coordination, there's far less need for you to micro-manage details or constantly jump between platforms to keep things aligned.

3. **Smarter Execution:**

Its shared memory ensures agents don't operate in silos each one builds intelligently on the outputs and context of others, leading to more coherent, reliable results.

4. **Faster Delivery:**

By orchestrating parallel execution wherever possible, Octopus dramatically cuts total turnaround time, completing complex projects in a fraction of the usual duration.

Limitations

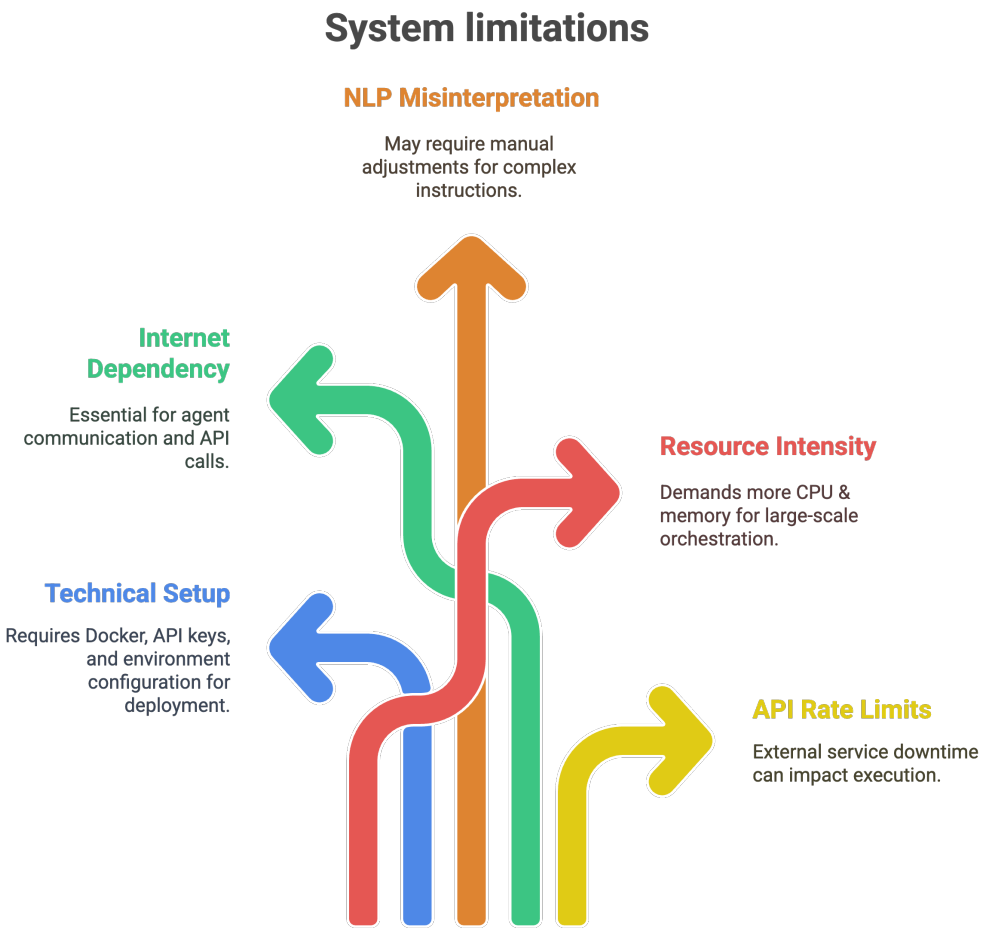
- **Requires Technical Setup:**
  • Needs Docker, API keys, and environment configuration to deploy and run the system.

- **Dependent on Internet & APIs:**
  • Stable internet is essential for agent communication and third-party API calls (e.g., Notion, GitHub, Slack).

- **NLP May Misinterpret:**
  • x or overly complex natural language instructions might not always be parsed correctly, requiring manual adjustments.

- **Resource Intensive for Scaling:**
  • Each agent runs in its own container; large-scale orchestration may demand more CPU & memory resources.

- **API Rate Limits & Downtime:**
  • Relies on external services; any API rate limits or downtime from platforms like GitHub or Notion can impact execution.



References

1. spaCy Documentation
• https://spacy.io/usage

2. scikit-learn Documentation
• https://scikit-learn.org/stable/user_guide.html

3. FastAPI Documentation
• https://fastapi.tiangolo.com/

4. Docker Documentation
• https://docs.docker.com/

5. Redis Documentation
• https://redis.io/docs/

6. Notion API Documentation
• https://developers.notion.com/docs

7. GitHub REST API Documentation
• https://docs.github.com/en/rest

Conclusion & Future Scope

Conclusion : -

• Octopus automates entire workflows from a single instruction, acting like your AI-powered digital company.

• It cuts manual work, speeds up delivery, and reduces errors.

• Ensures transparent, consistent execution essential for teams handling complex projects.

Future Scope: -

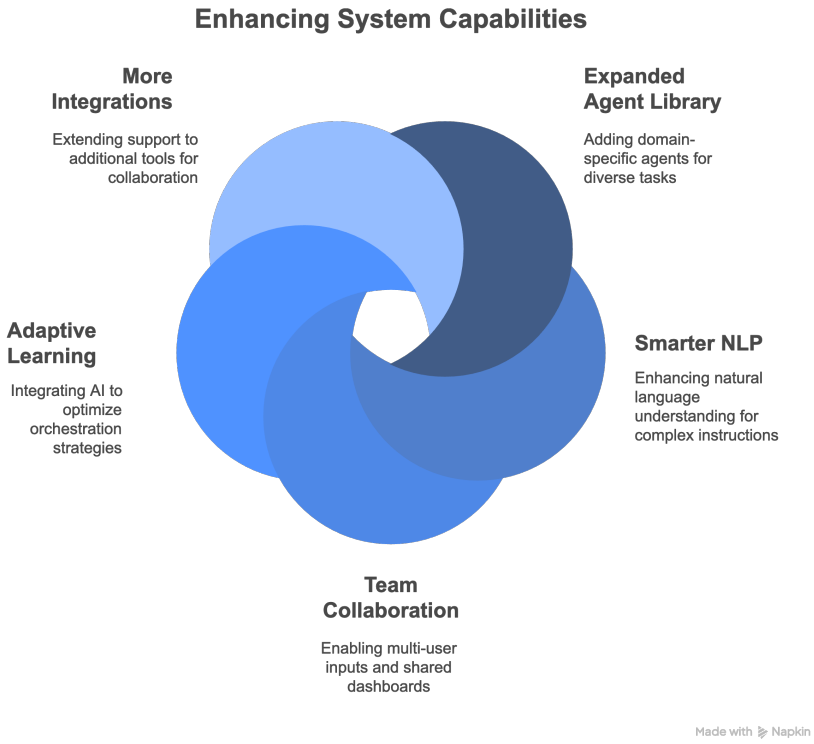
• Expanded Agent Library: Add more domain-specific agents (CRM, analytics, DevOps) to handle diverse tasks.

• Smarter NLP: Enhance natural language understanding to parse even broader, more complex instructions.

• Team Collaboration: Enable multi-user inputs and shared dashboards for real-time team coordination.

• Adaptive Learning: Integrate AI that optimizes orchestration strategies by learning from past projects.

• More Integrations: Extend support to additional tools like Jira, Asana, and custom enterprise APIso collaboration.



THANK YOU