📝 Instructions

- GitHub Repo link should be submitted through this google form.
- Adding a docker implementation is mandatory.
- Include a readme.md detailing how the code can be executed.
- Use the free services / tiers only to implement the solution.
- Each team should submit their solution only once through google form. In case of submission by more than one team member, only the most recent submission will be considered.
- Solution should be submitted by 11th May, 2025, 11:59 PM. No late submissions will be entertained. Any email or g-chat messages will also not be entertained after this deadline.
- You can post your queries on g-chat spaces or reach out to Irtiza Hussain, Muhammad Shariq, Sana Ullah, Saba Pervez, Nabeel Farooqui or Syed Daniyal Hassan Naqvi and they will respond to your queries as soon as possible.
- Submissions must be original. Plagiarism or completely copying from open-source projects will result in disqualification.
- Teams should cross-check their submission before submitting to avoid incomplete uploads. No changes after the submission deadline will be considered.

🚀 Challenge: AgentDock — Multi-Agent MCP Server with UI & Tool Integrations

Overview

AgentDock is a Model Context Protocol (MCP) server with a clean UI to register, manage, and interact with intelligent agents. It enables multi-agent orchestration, tool integrations (e.g., GitHub, Slack, Jira), and LLM-powered interactions via an LLM (free version / tier).

The goal is to build an extensible platform where users can trigger actions through natural language, monitor agent behavior, and register tools dynamically — all from a web interface.

Core Requirements

UI Capabilities

- **Agent Management**
    - Register/deregister agents with code, description, and config
- **Natural Language Interface**
    - Ask agents questions using Groq (e.g., “Summarize latest PR”)
- **Monitoring & Logs**
    - View recent agent actions and outputs

Backend Functionality

- **MCP-Compatible Agent Server**
- **Multi-Agent Support** (e.g., GitHubSync, SlackAgent, etc.)
- **REST API Tool Registration**
- **Modular & Extensible Architecture**

Sample Integrations

Speech-to-text commands (e.g., "Update Shopify inventory")

Advanced tools or agent chaining

Seamless configuration UI for tool-specific settings

These are some examples, try to make creative ones.

Tool	Example Use
GitHub	PR summaries, repo sync, CI/CD triggers
Jira	List, update, or create tickets
Slack	Send messages, channel updates

Shopify	Inventory updates via natural language
Speech	Transcribe commands into actions

🏆 Evaluation Criteria

1. Core Functionality (30%)

- Are all the required features implemented (Agent registration, Natural Language Interface, Logs, REST API for tools)?
- Does the system support multi-agent orchestration properly?

2. Technical Design and Architecture (20%)

- Is the backend modular, extensible, and MCP-compliant?
- Is the codebase well-organized, documented, and scalable for new integrations?

3. UI/UX and Usability (15%)

- Is the web interface clean, intuitive, and responsive?
- Can users easily register agents, monitor logs, and interact via natural language?

4. Integration Quality (15%)

- How well are the sample tools (e.g., GitHub, Slack, Jira) integrated?
- Are the REST APIs and tool configuration interfaces user-friendly and reliable?

5. Innovation and Extra Features (20%)

- Are there additional creative features like agent chaining, advanced monitoring, or speech-to-text commands? The creative idea of the agent is also a plus point.