Project Proposal Report

AgentOS: An MCP-Native Operating System for Autonomous Multi-Agent AI Systems

1. Introduction

Large Language Models (LLMs) are rapidly evolving from passive dialogue systems to **active tool-using agents** that perform reasoning, retrieval, planning, and real-world tasks. However, existing multi-agent systems lack a unified, standardized mechanism to access external tools, maintain long-term context, exchange information, and coordinate tasks reliably.

The **Model Context Protocol (MCP)**—a new open standard for tool/agent interaction—provides a universal interface for LLMs to securely access resources such as files, databases, browsers, and APIs. Despite its potential, current MCP applications are limited to isolated tools or lightweight integrations.

This project proposes **AgentOS**, a novel "operating system" built entirely on MCP principles. AgentOS treats tools as system resources and agents as processes, delivering a robust and scalable environment for autonomous multi-agent LLM workflows.

The goal is to design, implement, and evaluate a **research-grade agent operating system** that can become a standout contribution in applied AI and potentially form the basis for a research paper.

2. Problem Statement

Current AI agent frameworks face several limitations:

1. **Fragmented Context Access:**
   Tools and data sources are siloed; agents cannot share context efficiently.
2. **Lack of Standardized Tool Interaction:**
   Every system uses proprietary APIs, making interoperability difficult.
3. **Unreliable Long-Running Agents:**
   When agents operate over extended periods, they face context overflow, state inconsistency, and memory loss.
4. **No Unified "Agent Kernel":**
   There is no OS-like abstraction to manage:
   - Scheduling
   - Tool permissions
   - Memory
   - Resource allocation
   - Task delegation
   - Safety enforcement

To address these challenges, AgentOS aims to create a **systematic, operating-system-inspired structure** for multi-agent AI systems using the MCP ecosystem.

3. Proposed Solution: AgentOS

3.1 Overview

AgentOS is an MCP-native operating system for AI agents.
It provides a unified environment in which:

- **Agents are processes**
- **MCP tools are system resources**
- **The orchestrator is the kernel**
- **Context layers function as memory management modules**
- **Task scheduling and stateful execution mimic OS-level designs**

The system ensures that each agent has controlled, secure, auditable access to resources while maintaining stable long-running operations.

4. Research Objectives

The primary objectives of the project are:

1. **Design an OS-style architecture** using MCP tools and multi-agent orchestration methods.
2. **Develop a modular MCP tool ecosystem** (filesystem, search, web, DB, logging, vector memory).
3. **Implement an Agent Scheduler** for cooperative multitasking between LLM agents.
4. **Build a Hierarchical Memory Layer**:
    - Short-term conversation memory
    - Episodic vector memory
    - Structured JSON-based memory
    - Long-term cold storage
5. **Integrate a Trust & Safety Firewall** for tool access regulation.
6. **Create benchmark tasks** to evaluate system stability, latency, and autonomy.
7. **Produce a research paper** demonstrating novelty, results, and insights.

5. System Architecture

5.1 High-Level Design

AgentOS consists of the following components:

1. **Agent Kernel (Coordinator)**
    - Manages agent lifecycle
    - Handles scheduling, context routing, and tool permissions
    - Maintains system-level logs and metrics
2. **MCP Tool Layer (Drivers)**
    - Filesystem
    - Browser / HTTP client
    - Vector memory store
    - Database client
    - Evaluation harness
    - Logging and audit
      Each tool acts like an OS “driver” accessible via MCP.
3. **Multi-Agent Process Layer**
   Agents perform tasks such as:
    - Retrieval Agent
    - Planning Agent
    - Reasoning Agent
    - Execution Agent
    - Review / Safety Agent
      They run simultaneously but cooperatively.
4. **Hierarchical Memory System**
   Inspired by OS memory tiers:
    - **L1:** Dialogue buffer
    - **L2:** Episodic memory (vector DB)
    - **L3:** Structured memory (JSON schemas)
    - **L4:** Cold storage (disk via MCP FS tool)
5. **Zero-Trust Tool Firewall**
    - Every incoming tool call is validated
    - Risk scoring & policy checks

- Logging for reproducibility

6. **Monitoring & Observability Layer**
    - Performance metrics
    - Token usage
    - Memory hits/misses
    - Latency breakdowns

6. Expected Outcomes

1. **A functioning prototype of AgentOS** capable of running fully autonomous multi-agent workflows.
2. **A research paper** containing:
    - System design
    - Architecture diagrams
    - Quantitative benchmarks
    - Analysis of memory performance and agent stability
3. **New technical skills**, including:
    - MCP protocol development
    - Multi-agent system orchestration
    - Hierarchical memory design
    - Trust and safety in tool-using agents
    - Distributed architecture concepts
4. **A publishable contribution** suited for workshops or conferences in:
    - AI systems
    - LLM engineering
    - Autonomous agents

7. Evaluation Plan

The system will be evaluated using the following metrics:

7.1 Performance Metrics

- Tool-call latency
- Inter-agent communication overhead
- Memory retrieval accuracy
- Task completion time

7.2 Reliability Metrics

- State consistency under long-running workflows
- Memory tier hit-rates
- Recovery after agent failure

7.3 Safety Metrics

- Number of blocked unsafe tool calls
- Policy violation detection accuracy
- False positives/negatives in tool auditing

7.4 Comparative Analysis

AgentOS will be compared against:

- LangChain Agents
- LangGraph workflows
- CrewAI multi-agent systems

8. Innovation and Contribution

AgentOS represents a meaningful academic contribution because:

- It is **the first OS-style architecture built on MCP**.
- Introduces **hierarchical memory management** for LLM agents.
- Implements **secure, policy-driven tool usage** across agents.
- Offers **fail-safe scheduling** for multi-agent autonomy.
- Bridges OS concepts (processes, drivers, kernel) with agent systems—an underexplored research area.

This type of architecture is highly relevant for future AI applications in:

- Robotics
- Autonomous reasoning
- Real-time multi-agent coordination
- Industrial automation
- AI personal assistants

9. Project Timeline

Week	Milestone
1	Literature review & system design
2–3	MCP tool implementation (filesystem, vector store, browser)
4–5	Agent Kernel + Scheduler
6–7	Multi-agent layer + orchestration
8	Hierarchical memory system
9	Trust & Safety Firewall
10	Benchmarking & experiments
11	Research paper drafting
12	Final report, documentation & presentation

10. Conclusion

AgentOS aims to push the boundaries of what LLMs can do by providing a coherent, OS-inspired platform for multi-agent autonomy. By leveraging MCP as the foundational communication layer, this project will build a scalable, secure, and academically significant system. It not only deepens practical engineering skills but also contributes novel research insights that can support a conference-ready paper.

I request approval to proceed with this project as my semester-long research assignment.