Overall Project Plan – Multi-Agent AI System (HO, MEC, Context, Coordinator)

This document provides an integrated practical plan for the full multi-agent AI system.
It details each agent's purpose, inputs, outputs, interactions, simulation workflow, and training approach — all written concisely and without unnecessary complexity.
This project will implement a simulation-based proof-of-concept, not a real-network deployment. The goal is to demonstrate agent cooperation and context-aware decision intelligence using LangGraph.

1. Coordinator Agent (Rule-Based Engine)

Nature:
A deterministic agent with fixed rules ensuring safe, conflict-free coordination between the Handover (HO) Agent and the MEC Offloading Agent. No AI or learning is used in this module.

Inputs

• HO proposal: action (stay/handover), target cell, predicted quality, adjusted HOM/TTT, validity period, and reasoning.
• MEC proposal: action (local/offload), target MEC node, predicted latency/energy/CPU-BW use, validity period, and reasoning.
• Shared state: RSRP/SINR trends, cell/MEC loads, UE mobility, backhaul status.
• Context outputs: preference weights (α, β, γ), flags (high mobility, overload), forbidden cells/nodes, policy mode.
• Timing: cooldown windows, timestamps, proposal expiration.

Outputs

• Final decisions: accept / revise / reject for each proposal.
• Revised suggestions: alternate cell/MEC node or adjusted constraints.
• Updated system flags: cooldown activation, resource locks, stability thresholds.
• Validity duration: time window for final approved actions.
• Explanation summary for logs and traceability.

2. Handover Agent (HO)

Nature:
A hybrid agent built from:
• Rule-Based Safety Shield (mandatory)
• Contextual Bandit for simple learning
• DDPG for continuous HOM/TTT tuning and HO action selection
Based on the AHO-DDPG (2025) research model.

Inputs

Type A (direct measurements):
• RSRP/RSRQ/SINR of serving and neighboring cells.
• UE mobility: speed, direction, distance to cells, time since last HO.
• Cell load and backhaul status.
• Current HOM and TTT values.

Type B (context-derived):
• Preference weights: performance, energy, reliability.
• Recommended HOM/TTT ranges.
• Flags: high ping-pong risk, high mobility, unstable target cell.
• Coordinator constraints: forbidden cells, cooldown, restricted transitions.

Outputs

• HO action: stay OR handover to selected target cell.
• Tuned HOM and TTT values (continuous within safety bounds).
• Predicted effect: expected RSRP/SINR after HO, ping-pong risk, expected HO latency.
• Validity window for the proposal.
• Justification string explaining the selection.

3. MEC Offloading Agent

Nature:
A hybrid agent using:
• Rule-Based Safety Shield
• Bandit learner
• DDPG-DTLCM for continuous offloading decisions
Based on the 2025 DDPG-DTLCM research paper.

Inputs

Type A (direct measurements):
• Task details: size, CPU cycles, deadline, task type.
• Device status: battery %, local CPU power, signal quality.
• MEC nodes: CPU/BW availability, load, queue length, backhaul delay.
• Distance-to-node and capability matching (DTLCM metrics).

Type B (context-derived):
• Preference weights: α (performance), β (energy), γ (reliability).
• Suggested CPU/BW caps, energy-saving flags, task urgency level.
• Coordinator restrictions: forbidden MEC nodes, deadline strictness, resource masks.

Outputs

• Offloading action: execute locally OR offload to selected MEC node.
• Resource request: CPU share and bandwidth share required on the target node.
• Predicted metrics: expected delay, expected energy consumption, task success probability.
• Suitability score based on DTLCM.
• Validity window and justification string.

4. Context Analyzer (Tele-LLM)

Nature:
An LLM-based agent (Tele-LLM) that converts raw measurements and logs into structured Type B context parameters
used by HO, MEC, and the Coordinator.

Inputs

- Radio logs: signal trends, HO failures, ping-pong history.
- Mobility patterns: speed, stability, movement trajectory.
- MEC data: delays, queue sizes, node load distribution.
- User preferences: performance vs energy vs reliability.
- System logs: past KPIs, anomalies, coordinator alerts.
- Network mode: number of nodes, congestion, energy level.

Outputs

- Type B parameters: α, β, γ weights.
- HO tuning hints: TTT/HOM ranges, cells to avoid.
- MEC hints: preferred nodes, CPU/BW caps, energy mode.
- System context labels: "high mobility", "battery saving", "congested cell", etc.
- Coordinator assistance: warnings about stability or priority changes.
- Outputs always packaged in structured JSON for LangGraph.

5. LangGraph Execution Pipeline

Node Map:
- ContextAnalyzer
- HOAgent
- MECAgent
- Coordinator
- StateStore
- Logger

Data Flow:
1. Input → ContextAnalyzer.
2. ContextAnalyzer → StateStore (Type B).
3. StateStore → HO/MEC in parallel.
4. HO/MEC → Coordinator.
5. Coordinator → StateStore + Actuation.
6. Logger stores KPIs.

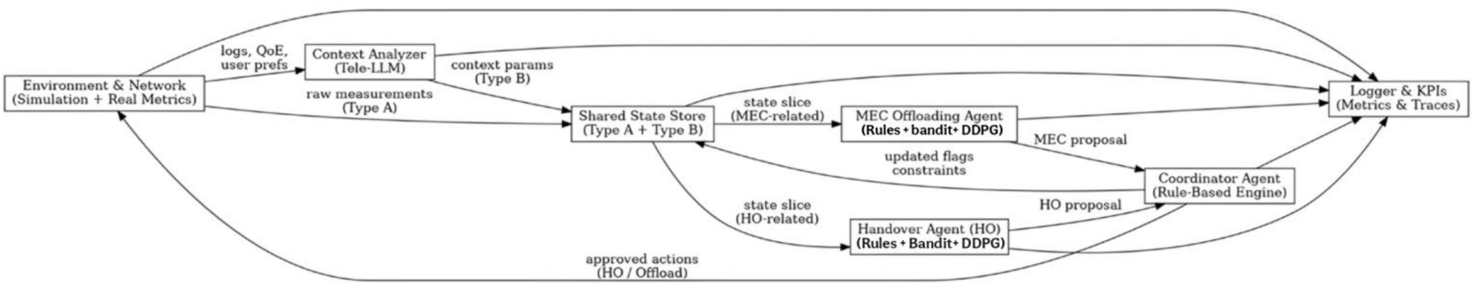
Execution Guards:
- No HO during critical MEC.
- No MEC offload during HO window.
- Strict timeouts: LLM ≤100ms, HO/MEC ≤20ms.

6. Shared State and Messaging

State includes:
- Type A: immediate radio, mobility, load, tasks.
- Type B: context parameters, tuning values, masks.
- Versions: timestamps and TTLs.

Messages:
- Context → StateStore: weights, tuning, caps, masks.
- HO/MEC → Coordinator: decisions, costs, constraints, validity.
- Coordinator → All: accept/revise/reject, reasons.

The following diagram illustrates the top-to-bottom data flow:



7. Simulation Framework

A Python-based Gym-like simulation is used to evaluate the system and train learning components. LangGraph manages the multi-agent workflow at each step.

Simulation Inputs

• Topology setup: number of cells, MEC nodes, node distances.
• Radio model: pathloss, shadowing, noise, instability factor.
• Mobility model: UE paths, speed, direction variability.
• Task generation: task size, rate, deadlines.
• Load scenarios: light / medium / heavy network conditions.

Simulation Outputs

• Radio KPIs: RSRP, SINR, HO triggers.
• MEC KPIs: end-to-end delay, energy use, success rate.
• System KPIs: HOSR, HOF, Ping-Pong, QoE, stability.
• Decision logs: timestamped actions, reasons, validity windows.

8. Training Methodology

Training follows a **two-stage progressive strategy** to keep learning safe, stable, and efficient across both HO and MEC agents.

Stage 1 — Bandit Phase (Safe Initial Learning)

Both agents begin with discrete, low-risk decisions:

- **HO Agent:** selects the best target cell using contextual features (RSRP/SINR, load, mobility).
- **MEC Agent:** selects the best MEC node using **DTLCM scoring** (distance + capability match).

This phase provides quick adaptation, avoids unsafe parameter changes, and establishes a strong baseline before continuous learning begins.

Stage 2 — DDPG Phase (Continuous Optimization)

After Bandit policies stabilize, both agents switch to continuous learning using **DDPG**:

- **HO Agent:** optimizes *HOM* and *TTT* alongside the HO decision.
- **MEC Agent:** optimizes continuous resource allocation (CPU/BW) and refined offloading choices.

The reward combines throughput, delay, energy, failure penalties, and stability measures to ensure balanced behavior.

Training Process

Training is performed across multiple simulation episodes with varying mobility, load, and channel conditions. Convergence is reached when KPIs stabilize or after a fixed episode count.

This Bandit → DDPG progression ensures safe exploration first, then fine-grained optimization, producing reliable and context-aware behavior for both agents.

The reward function is dynamically weighted using the context-derived preference weights (performance, energy, reliability), ensuring that the agent's behavior aligns with the application's high-level goals.