Project Charter:Multi-Agent LLM System for Automating Engineering Research Workflows

Project Information

Project No. & Project Title: AutoResearchLab: Multi-Agent LLM System for Automating Engineering Research Workflows

Students’ Names: [Wang Chenweiyi, Yang Yalin, Zeng Deyang ,Chen Zidan ,Zhang Chengyuan]

Supervisor’s Name: Assoc Prof. Chau Yuen

Date Prepared: [January 2026]

Project Supervisor: Assoc Prof. Chau Yuen

Team Leader: [Wang Chenweiyi]

Team Members: [Zeng Deyang ,Chen Zidan ,Yang Yalin,Zhang Chengyuan]

Project Purpose or Justification

Project Description

Engineering research workflows traditionally involve many repetitive steps, such as literature review, experimental planning, coding of simulations, and report drafting. However, most existing tools are single-agent "chatbots" that lack the structure and reliability required for serious research use.

The purpose of **AutoResearchLab** is to address these limitations by designing a multi-agent Large Language Model (LLM) system. This system will coordinate specialized AI agents to support EEE research tasks in a systematic and traceable way, allowing researchers to focus on critical oversight and high-level engineering rather than repetitive routine tasks.

Traditional engineering research workflows—ranging from idea conception and code implementation to paper drafting—rely heavily on repetitive human labor. Existing AI tools are mostly single-agent chatbots that lack the systematic structure required for complex research, often leading to logical inconsistencies and irreproducibility.

AutoResearchLab is a multi-agent AI framework developed based on the **Agent Development Kit (ADK)**.

Core Modules & Technical Challenges:

1. **Idea Generation & Evaluation Module:** Generates research ideas. Challenges include establishing a quantitative metric system to score ideas on Novelty, Feasibility, and Application Value.
2. **Code Execution & Debugging Module:** Handles simulation code construction and execution. Challenges include implementing **Git-based version control** to manage code evolution, preventing logical dead ends via standardized research plans, and generating robust experimental data.
3. **Conclusion Synthesis & Paper Output Module:** Handles data cleaning and paper drafting. Challenges include using **RAG (Retrieval-Augmented Generation)** to ensure logical consistency between experimental data and written conclusions.

High-Level Technical Roadmap

The project will orchestrate a closed-loop workflow across four phases using ADK:

1. **Initialization & Orchestration :**
   - The user inputs a vague research intent .
   - The ADK Core Controller decomposes the task into sub-goals and dispatches them to specialized agents.
2. **Phase 1: Idea Verification (Idea Agent & Metric Evaluation):**
   - **Input:** Task description + External Literature Database.
   - **Process:** Idea Agent generates proposals -> Evaluator scores them based on "Novelty, Feasibility, Value" (0-10 scale).
   - **Output:** The highest-scoring, standardized Research Plan.
3. **Phase 2: Iterative Code Construction (Iterative Coding with Git):**
   - **Environment:** Isolated Python Execution Sandbox.
   - **Process:** Coding Agent generates code based on the plan -> Attempts execution.
     - **Git Mechanism:** Major changes trigger auto-commits; if errors or infinite loops occur, the system triggers a **Rollback** to the previous stable branch and retries with a adjusted strategy.
   - **Output:** Executable simulation scripts + Experimental Data Logs (Logs/CSV).
4. **Phase 3: Evidence-Based Writing (Writing Agent + RAG):**
   - **Logic Check:** Writing Agent reads experimental logs and retrieves relevant theory via RAG.

- **Consistency:** Verifies that "Research Conclusions" are supported by "Experimental Data," eliminating hallucinations.
- **Output:** Technical report in LaTeX/Markdown with plots and citations.

Summary Milestones

Milestone	Target Date
ADK Fundamentals Team study of Agent Development Kit (ADK) core concepts and documentation.	Week 1
Open Source Research Investigation and analysis of relevant open-source multi-agent projects.	Week 2
Architecture & Planning Finalize overall system architecture design and assign specific tasks to team members.	Week 3
Framework Setup Establish the project skeleton and define API interfaces between agents.	Week 4
Standardization Define data output structures (Schemas) for key workflow nodes and pass criteria for testing.	Week 5
Module Development Development of individual sub-modules (Idea, Coding, Writing) and passing verification tests.	Week 6 - 8
System Integration Full system integration and joint debugging (End-to-End testing).	Week 9
Optimization Address key technical bottlenecks and optimize issues found during development.	Week 10 - 11
Reporting Synthesize project conclusions, organize data, and draft the technical paper.	Week 11
Final Delivery Project defense and system demonstration.	Week 12

Team Member Roles

Team Member	Key Activity & Responsibilities
Wang Chenweiyi	**Team Leader & System Architect** Project Management and overall ADK architecture design. Manages cloud infrastructure, API integration, and the foundational runtime environment for code execution.
Yang Yalin	**Idea Generation Module Lead** Develops the "Idea Agent" and designs quantitative evaluation metrics (Novelty/Feasibility) to filter and optimize research directions.
Zeng Deyang	**Code Generation & Git Workflow** Develops the "Coding Agent" focusing on prompt-to-code translation and pre-processing. Implements the Git-based version control and "rollback" logic for debugging.
Chen Zidan	**Execution & Data Synthesis** Monitors simulation execution and develops the "Data Agent." Responsible for structuring simulation logs, generating visualizations, and formatting results for the writing module.
Zhang Chengyuan	**Paper Synthesis & RAG System** Develops the "Writing Agent" and RAG knowledge base. Responsible for synthesizing academic papers from structured data and verifying logical consistency.

Summary Budget

Funding is primarily for ADK-related network services and computing resources:

1. **API Costs:** Token usage for Gemini API or other LLMs.
2. **Support Components:** RAG vector database storage, Git hosting services.
3. **Compute Resources:** Cloud server costs for online compilation, debugging, and simulation.

Risk Assessment

1. **Risk:** Subjective idea metrics leading to low-value proposals. **Mitigation:** Introduce a Reviewer Agent to cross-check novelty against paper databases.
2. **Risk:** Code debugging entering infinite loops. **Mitigation:** Use Git to force Rollback and reset context.
3. **Risk:** Mismatch between paper conclusions and data. **Mitigation:** Enforce citation of experimental logs and cross-verify via RAG.