Multi■Agentic Coding Framework – System Architecture & Workflow Created by Saksham Jain

1. Overview
This system is a deterministic multi■agent software factory built using AutoGen and Streamlit. It converts natural lang

2. Agents and Responsibilities
Controller Agent – Initializes the workflow, injects rules, and starts the pipeline.
Requirements Agent – Converts user intent into a formal requirements.md document.
Coding Agent – Produces a production■ready main.py implementation.
Review Agent – Performs correctness, performance, and security validation. Can request fixes.
Documentation Agent – Generates README.md with usage instructions.
QA Agent – Produces test_main.py and ensures test safety using mocks.
Deployment Agent – Creates Dockerfile and run.sh.
UI Agent – Generates app_ui.py Streamlit frontend.

3. File Structure
app.py – Streamlit controller UI
workflow.py – Orchestration engine
agents.py – Agent factory and LLM backend config
utils/logger.py – Centralized logging
utils/test_executor.py – Safe test runner
workspace/ – Generated project output directory

4. Sequential Workflow Design
Unlike default AutoGen group■chat style systems, this framework enforces strict sequential execution. This avoids de

5. Why Sequential over GroupChat Autonomy
We originally experimented with:
- GroupChat auto speaker selection
- LLM■controlled routing
- Manual override selection

All approaches caused:
- Infinite loops
- Empty agent responses
- Inconsistent LLM behavior across providers

Sequential pipelines guarantee termination, deterministic reproducibility, and debugging safety.

6. How to Run the Project
Install dependencies:
pip install -r requirements.txt

Set your LLM provider in agents.py:
- Groq API
- Ollama local
- OpenRouter

Run the system:
streamlit run app.py

7. Output Artifacts
requirements.md – Software specification

main.py – Core application logic
README.md – User documentation
test_main.py – Unit tests
Dockerfile + run.sh – Deployment
app_ui.py – Application UI

8. Logging & Observability
Every agent action is logged to system_logs.log for full traceability.

9. Workspace Execution
All generated files are extracted to /workspace and tested automatically via unittest runner.