PROJECT 7 — SMART MANUFACTURING PLANT MONITORING & DIAGNOSTICS ASSISTANT

A real industrial AI assistant capable of reading SOPs, interpreting machine logs, diagnosing faults, and supporting operators on the factory floor

---

1. Problem Scenario

A large manufacturing company wants an AI assistant that can support:

- Plant operators
- Maintenance engineers
- Quality control personnel
- Shift supervisors

The assistant should be able to:

- Retrieve **Safety SOPs, Machine Manuals, Work Instructions** using RAG
- Interpret machine sensor logs
- Diagnose faults using rule-based or ML-based tools
- Provide step-by-step troubleshooting
- Retrieve spare part information
- Suggest preventive maintenance
- Notify escalation if issue is severe
- Track production KPIs (OEE, cycle time, downtime)
- Maintain context about ongoing machine issues
- Integrate with machine log systems through tools or mock APIs

The system must be built using:

- **Ollama + Mistral**
- **LangChain**
- **RAG**
- **Tools for sensors, logs, diagnostics, parts**
- **Streamlit UI + FastAPI backend**
- **Memory**
- *(Bonus)* Agentic planning, caching, autonomous alerting

---

2. Functional Requirements

A. Core Capabilities

1. Answer operator questions about machines
2. Provide real-time monitoring for:
    - Temperature
    - Vibration
    - RPM
    - Pressure
    - Power consumption
3. Provide troubleshooting steps for faults
4. Retrieve SOPs and manuals from RAG
5. Interpret logs uploaded as CSV or text
6. Provide safety guidelines
7. Notify when escalation is needed
8. Guide operator through safety procedures
9. Suggest spare parts
10. Provide predictive maintenance overview

---

3. RAG Requirements

RAG Dataset Should Include:

- Machine SOPs
- Preventive maintenance schedules
- Troubleshooting handbooks
- Calibration guidelines
- Spare part catalogs
- Safety rules & PPE instructions
- Quality management documents
- Failure mode manuals (FMEA)

Vector store:

- FAISS (recommended for fast on-site usage)
- ChromaDB (if ease of setup is preferred)

Students must justify the choice.

Expected RAG Outputs:

- Safety steps
- Machine operational steps
- Troubleshooting guidelines

- Maintenance intervals
- “What to do when X error appears”

---

4. Memory Requirements

✔ Entity Memory

Store:

- Machine ID currently being discussed
- Fault codes mentioned
- Recent sensor readings
- Operator name & shift
- Previously attempted troubleshooting steps

✔ Summary Memory

Summarize long diagnostic conversations.

✔ Optional: Vector Memory

For long-term history of machine issues.

---

5. Tools / Functions Students Can Build

Students must implement **5–8 tools**, since industrial setups are tool-heavy.

---

1. Sensor Data Fetch Tool

Input:

- machine_id

Output:

- current temperature
- vibration
- rpm
- power
- pressure

Students can:

- Use mock sensor JSON data
- Simulate time-series readings

---

2. Fault Diagnosis Tool

Input:

- sensor values
- fault code

Output:

- suspected failure mode
- risk severity
- first-level steps

This can be:

- Rule-based
- LLM-based classification
- A small ML model

---

3. Spare Parts Lookup Tool

Input:

- machine_id
- issue type

Output:

- recommended spare parts
- availability
- price
- supplier info

Dataset: JSON/CSV catalog.

---

4. Maintenance Scheduler Tool

Given:

- machine usage hours
- last maintenance date

Output:

- next maintenance due
- type of maintenance (A/B/C level)
- tasks checklist

---

5. Log File Analyzer Tool

Students upload CSV logs containing:

- timestamps
- sensor readings
- error codes
- cycle time

Tool should:

- detect anomalies
- extract patterns
- summarize

Optional: use simple threshold-based anomaly detection.

---

6. Safety Rule Checker Tool

Given:

- current issue
- machine type

Output:

- PPE required
- lockout/tagout instructions
- hazard level

---

7. Production Metrics Tool

Compute:

- OEE (Overall Equipment Effectiveness)
- Downtime percentage
- Throughput rate
- Cycle time deviation

From mock production logs.

---

8. Notification / Escalation Tool

If severity is high:

- send email
- notify supervisor
- or simulate via console

---

6. APIs Students May Use (Optional)

IoT / Sensor APIs

- Students simulate/ mock them
- Or use:
    - Thingspeak API
    - Azure IoT Hub REST APIs
    - Real industrial datasets from Kaggle

Maintenance Data

Students create static JSON.

---

7. Agentic AI Design (Bonus)

Agent Roles

1. **Sensor Monitoring Agent**
    - Poll sensors
    - Detect anomalies
    - Trigger alarms
2. **RAG Retrieval Agent**
    - Retrieve relevant SOP section

3. **Diagnostics Agent**
    - Use fault tool
    - Suggest first-level fixes
4. **Maintenance Planning Agent**
    - Plan preventive maintenance
    - Check spare parts
5. **Escalation Agent**
    - Detect when to escalate
    - Ensure operator safety

Autonomous Behavior Examples

- If temperature > threshold → warn operator
- If error code repeated → suggest deeper maintenance
- If no operator response → escalate

---

8. Caching Strategy (Bonus)

Cache:

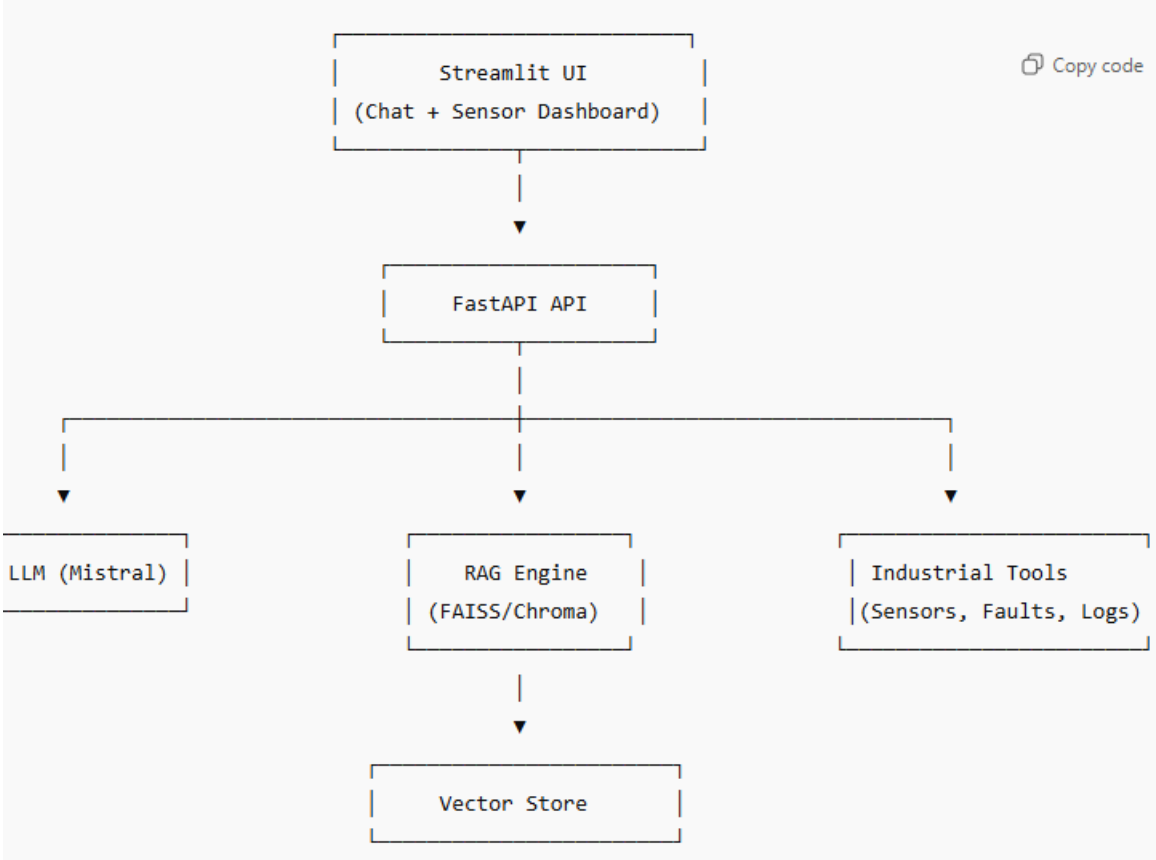
- SOP sections
- Spare part catalog
- Sensor readings for past 5 minutes
- Common fault explanations

Benefits:

- Faster response for frequently asked troubleshooting
- Reduced sensor API read load
- Quick safety instruction retrieval

---

9. ASCII Architecture Diagram



10. Folder Structure

```
manufacturing-assistant/
│
├── backend/
│   ├── main.py
│   ├── tools/
│   │     ├── sensor_fetch.py
│   │     ├── fault_diagnose.py
│   │     ├── spare_parts.py
│   │     ├── maintenance.py
│   │     ├── log_analyzer.py
│   │     ├── safety_checker.py
│   │     └── metrics.py
│   ├── rag/
│   │     ├── retriever.py
│   │     ├── embedder.py
│   │     └── vector_store.py
│   ├── memory/
│
├── frontend/
│   └── app.py (Streamlit)
│
├── evaluation/
│   ├── rag_eval.ipynb
│   ├── bleu_rouge.ipynb
│   ├── sensor_eval.ipynb
│   ├── fault_diagnose_eval.ipynb
│   └── latency_tests.ipynb
│
└── data/
    ├── sops/
    ├── manuals/
    ├── sensors/
```

11. Evaluation Tasks

A. RAG Evaluation

Queries like:

- “What to do when Error Code E23 occurs?”
- “Safety steps for pump maintenance?”

Metrics:

- Precision@k
- Recall@k
- Diversity

B. Fault Diagnosis Quality

Compare tool output vs expected outcomes.
Evaluate:

- Correct diagnosis
- Correct severity
- Correct first steps

C. Response Quality

Metrics:

- ROUGE-L
- BLEU
- BERTScore

Applied to:

- troubleshooting
- safety instructions

D. Sensor Data Evaluation

Simulate:

- normal vs abnormal values
- evaluate anomaly detection logic

E. Latency Tests

Measure:

- sensor query time
- fault diagnosis tool time
- RAG time
- end-to-end response

F. Stress Tests

- 50 machines polled every 5 seconds
- multiple log uploads
- large SOP document queries

---

12. Deployment Requirements

Streamlit UI Must Include:

- Chat interface
- Real-time sensor dashboard
- Fault detection alerts
- SOP/RAG viewer
- Spare parts & maintenance tabs
- CSV log upload & visualization

FastAPI Backend Must Include:

- `/chat`
- `/sensors`
- `/fault`
- `/parts`
- `/maintenance`
- `/rag`
- `/analyze_log`

---

13. Student Submission Checklist

Students must deliver:

✔ RAG pipeline
✔ Tools for sensors, faults, maintenance
✔ Streamlit dashboard
✔ FastAPI backend
✔ Memory strategy
✔ Optional agent system
✔ Evaluation notebooks
✔ Test logs
✔ Usability demonstration