AI Agent for Engineering Execution Support (AEES)

1. Project Title

AEES – AI-Powered Agent for Engineering Execution & Decision Support

---

2. Problem Statement

In large engineering and industrial organizations, Execution and Engineering teams spend a significant amount of time on:

- Repetitive technical documentation
- Requirements analysis
- Equipment specification lookup
- SOP / manual searching
- Status reporting
- Root cause analysis of failures
- Information retrieval from large internal repositories

These tasks are often manual, non-automated, and time-consuming. Information is usually scattered across multiple documents, PDFs, spreadsheets, tickets, and emails.

Current challenges:

- Slow information access
- Human error
- Lack of unified knowledge system
- No AI assistance for engineers
- No intelligent automation in daily engineering workflow

---

3. Proposed Solution

Build an **AI-powered multi-agent system** that assists Engineering & Execution teams using:

- Natural Language Processing
- Multi-agent reasoning
- Tool use
- Knowledge retrieval (RAG)

- Document understanding
- Decision support
- Automation

The AI agent can:

✅ Answer technical project questions
✅ Explain engineering documents
✅ Search SOPs automatically
✅ Suggest actions based on issues
✅ Generate reports
✅ Track project status
✅ Flag risks
✅ Automate repetitive tasks
✅ Act as an engineering co-pilot

---

4. Project Objective

The goal of this project is to create an **intelligent Engineering Co-Pilot** that:

- Reduces time spent on documentation
- Increases productivity
- Improves accuracy
- Supports decision-making
- Automates repetitive work
- Centralizes information access

---

5. System Architecture

The system consists of four main layers:

1. Frontend Layer

Stack: React + Tailwind CSS
Purpose:

- User chat interface

- Dashboard view
- Upload engineering documents
- View AI responses
- Task activity panel

---

2. AI Orchestration Layer

Stack: LangChain + OpenAI GPT-4

Handles:

- Prompt logic
- Tool selection
- Multi-agent coordination
- Workflow routing
- Memory
- Reasoning

Agents:

- Engineering Assistant Agent
- Research Agent
- Report Generator Agent
- Risk Analysis Agent
- Automation Planner Agent

---

3. Knowledge Layer (RAG)

Stack: Embeddings + Vector Database (FAISS / Pinecone / Chroma)

Stores:

- Technical manuals
- SOPs
- Project plans
- Equipment specs
- Past tickets
- Failure reports

Used for:

- Context injection
- Accurate domain responses
- Retrieval-Augmented Generation

---

4. Backend API Layer

Stack: Python (FastAPI / Flask)

Handles:

- User requests
- File uploads
- AI calls
- Vector search
- Agent logic
- Secure OpenAI integration

---

6. Technology Stack

Frontend:

- React (Vite)
- Tailwind CSS
- Axios
- React Router
- Framer Motion (optional - animations)

Backend:

- Python
- FastAPI
- LangChain
- OpenAI GPT-4 API
- FAISS/Pinecone
- PyMuPDF/PyPDF
- Pandas

- Docx/CSV parsers

Database:

- Vector DB: FAISS / Chroma
- Optional: PostgreSQL for logs

Deployment:

- Frontend: Vercel/Netlify
- Backend: Render/AWS
- Vector DB: Local or cloud
- Docker

---

7. Dataset Details

The dataset is multi-source and mimics engineering knowledge:

Types of Data Used:

1. SOP Documents (PDF)
2. Equipment manuals
3. Maintenance logs (CSV)
4. Engineering project reports
5. Failure reports
6. Issue tickets
7. Configuration files
8. Past project timelines

Each file is chunked and embedded into the vector database.

Example dataset structure:

```
/data
  /manuals
  /sop
  /reports
  /logs
  /schemas
```

---

8. Features

👨‍💼 Engineering Query Assistant

Ask:

"Explain the cooling system failure in plant 3"
"Which SOP applies to high temperature anomaly?"
"Recommend components for a 200kW output"

📄 Auto Report Generation

Generates:

- Daily engineering summary
- Maintenance reports
- Risk reports
- Root cause analysis

🚨 Risk Assessment Agent

Flags:

- System vulnerabilities
- Project delays
- Anomalous values
- Unsafe conditions

🔁 Workflow Automation Agent

Automates:

- SOP recommendations
- Ticket categorization
- Priority tagging
- Step generation

---

9. Sample Use Cases

User Role	Example Use Case
Engineer	Ask for design specs
Supervisor	Get system status summary
Maintenance	Troubleshoot failure
Manager	Generate performance report
New Joiner	Learn systems via AI guide

---

10. Metrics of Success

Metric	Target
Time saved	60–70%
Error reduction	40%
Response speed	< 5 seconds
Productivity boost	2x
Knowledge accessibility	Instant

---

11. Future Enhancements

- Voice-based agent
- Autonomous multi-agent workflows
- IoT data integration
- Predictive maintenance AI
- Real-time plant monitoring
- Integration with SAP / Jira
- Robotics control
- 3D visualization support

---

12. Why This Project is Valuable to John Cockerill

Perfectly aligned with their focus on:

- Engineering innovation
- Industrial execution
- Smart automation
- AI in operational workflow
- Industry 4.0

This project proves ability to:

- Build AI agents
- Apply AI to engineering
- Understand industrial context
- Support real-world execution use cases

---

13. Conclusion

AEES is not just a chatbot.
It is an **intelligent engineering co-pilot** capable of transforming how engineers work daily.

It demonstrates a powerful blend of:

AI + Engineering + Automation + Execution + Decision Intelligence

---

14. Author

Developed By: Prakriti Sharma
Specialization: AI, Full Stack, Automation, ML Systems
Target Role: AI Intern – Engineering & Execution Support