AI Engineer Technical Assessment - One Day Challenge

Objective:
Build a multi-agent AI system with real-time streaming responses to demonstrate your core AI engineering skills. PLEASE DO NOT USE AI GENERATOR TOOLS FOR THIS TEST, DO IT YOURSELF EVEN IF THE RESULT IS WORST.

Task Requirements (6-8 hours)

Core Challenge: Smart AI Assistant with Agent Specialization

Create a FastAPI backend that routes user queries to specialized AI agents based on query type detection.

Part 1: Multi-LLM Integration (50 points)

1. LLM Router System
   ```python
   # Required endpoints
   POST /chat - Main chat interface
   GET /models/status - Available models health
   POST /chat/stream - Streaming responses
   ```

2. Model Integration (Choose 2 minimum)
   - OpenAI GPT
   - Google Gemini
   - Claude API
   - Any open-source model

3. Intelligent Routing
   - Route based on query complexity
   - Implement fallback mechanisms
   - Cost optimization logic

Part 2: Agent Specialization (35 points)

Create 3 specialized response modes:

1. Code Assistant
   ```python
   # Handles: code analysis, debugging, explanations
   trigger_keywords = ["code", "function", "debug", "programming"]
   ```

2. Research Assistant
   ```python
   ```

```python
   # Handles: information gathering, analysis, summaries
   trigger_keywords = ["research", "analyze", "compare", "find"]
```

3. Task Helper
   ```python
   # Handles: step-by-step guidance, how-to questions
   trigger_keywords = ["how to", "steps", "guide", "tutorial"]
   ```

Part 3: Real-time Streaming (15 points)

- Implement Server-Sent Events (SSE)
- Show progressive response building
- Handle connection interruptions

Technical Requirements

Technology Stack:
```yaml
Backend: FastAPI + Python 3.9+
AI APIs: Any 2+ LLM providers
Streaming: SSE or WebSockets
Database: SQLite (for session storage)
Testing: Basic pytest coverage
```

API Response Format:
```json
{
  "agent_used": "code|research|task",
  "response": "AI response content",
  "model": "gpt-4|gemini|claude",
  "confidence": 0.85,
  "processing_time": 1.2,
  "token_count": 150
}
```

Test Scenarios

Test 1: Agent Detection

Input: "Explain this Python function: def add(a,b): return a+b"

Expected: Routes to Code Assistant

Test 2: Model Fallback

Scenario: Primary model fails
Expected: Automatic fallback to secondary model

Test 3: Streaming Response

Input: "Write a detailed explanation of machine learning"
Expected: Progressive response chunks via SSE

Test 4: Cost Optimization

Input: "What is 2+2?"
Expected: Routes to cheaper/faster model

Deliverables (3 files maximum)

1. Source Code
- Single Python file or small project structure
- Clean, documented code(NON AI)
- Docker setup (optional but preferred)

2. Demo Script
```python
# demo.py - Automated demonstration
# Should show all features working
# Include sample API calls
```

3. Quick README
```markdown
# Setup (max 10 lines)
# API Usage (3-5 examples)
# Architecture Overview (brief)
```

Timeline: 8 Hours Maximum

Hours 1-2: Setup + Basic FastAPI + Single LLM
Hours 3-4: Agent detection + routing logic
Hours 5-6: Second LLM + fallback mechanism
Hours 7-8: Streaming + testing + documentation

Submission (VMNEBULA@gmail.com)
Email with:
1. Code files (ZIP or GitHub link)
2. Brief demo video (2-3 minutes, optional)
3. Setup instructions (if non-standard)

Subject:`AI Engineer Assessment - YOUR NAME`

Allowed Resources

Documentation, Stack Overflow
AI APIs official docs
Your existing code/projects

NOT Allowed:

AI coding/Writing assistance
Copy-paste solutions

Focus on demonstrating your ability to integrate multiple AI models, implement intelligent routing, and handle real-time responses. Quality over quantity!

Good luck!