■ NeoAI Insight Engine – Design Documentation

Date: August 6, 2025

Version: v1.0

Author: NeoAI Engineering Team

■ 1. System Architecture

The system follows a modular agent-based architecture where each agent is responsible for a distinct research task.

Component	Function
Frontend (Streamlit)	Collects user input, displays results, manages UI/UX
Backend (FastAPI)	Routes requests and orchestrates agents
DataCollectorAgent	Fetches real-world articles using GNews API
InsightAgent	Extracts high-confidence keywords using spaCy NLP
IdeaGeneratorAgent	Calls Groq LLMs to generate a detailed Markdown report

■ 2. Agent Communication Protocol

- User submits query and LLM choice from Streamlit interface.
- FastAPI receives POST request at `/generate`.
- Backend calls `DataCollectorAgent` to pull 3 GNews articles.
- Keywords are extracted from titles/descriptions using `InsightAgent`.
- `IdeaGeneratorAgent` sends structured prompt to Groq and returns a proposal.
- All results are returned to the frontend and displayed in Markdown format.

■ 3. Custom Algorithms

Keyword Extraction via `spaCy`:

The agent uses spaCy's `en_core_web_sm` pipeline to extract meaningful terms. It combines noun_chunks and named entities, removes duplicates, and ranks based on token length and position.

```
from collections import Counter def extract_keywords(text): doc = nlp(text) keywords = Counter(chunk.text.lower() for chunk in doc.noun_chunks if len(chunk.text) > 3) return keywords.most_common(10)
```

Prompt Engineering:

The keywords are inserted into a structured prompt requesting Markdown output and rich formatting.

■ 4. Key Innovations

- Multi-agent simulation of a real R&D; workflow
- Groq integration for ultra-fast open LLM access
- Modular architecture for easy scaling
- Minimal UI latency using Streamlit + FastAPI

- Fully Markdown-supported reports
- Customizable LLM selector with fallback support

■■ 5. Known Limitations

- Agents are called sequentially — no async or multi-threaded optimization yet.
- `spaCy`'s base model is limited in domain-specific understanding.
- No persistent user history or feedback learning.
- API failures only trigger fallback message, no retry mechanism.

■ 6. Future Enhancements

- Add agent chat interface with memory
- Switch to transformer-based extractors (e.g., KeyBERT, LLM summarizers)
- Support PDF & DOCX export of research outputs
- Integrate live logs panel and API analytics dashboard