stack8s

stack8s AI Agent (“AI Architect”)

Below is a **full, standalone Project Requirements Document** for the **stack8s AI Agent – “AI Architect”**.
It is structured for a freelancer or AI engineer to take over and implement as a clear, scoped project.

Project Requirements Document – stack8s AI Agent (“AI Architect”)

1. Project Overview

The goal of the **AI Architect** is to provide a conversational AI agent inside stack8s that can guide users to:

- Select the **correct compute** (local + multi-cloud)
- Design an **end-to-end AI/ML architecture**
- Recommend **LLM models** and supporting components
- Propose an optimal **infrastructure stack**, deployable on stack8s

The AI Architect acts as a specialised, domain-aware chatbot that interacts with users through natural language, asking follow-up questions, validating assumptions, and producing actionable recommendations.

It must pull information from **four knowledge domains**: - WE WILL PROVIDE YOU THESE

1. **Multi-cloud Compute Availability & GPU Pricing (from DB) - pgsql**
2. **Marketplace of LLM Models (HuggingFace) - scrape from here**
3. **Marketplace of Kubernetes-Native Components (from DB) or as founder here**
   (databases, API gateways, pipelines, inference endpoints, vector DBs, etc.)
4. **Local Cluster Compute Availability (Optional)**

---

2. Functional Objectives

The agent should:

2.1 Understand user intent from natural language

Example input:

> "I have 2TB of MRI images and want to detect lung cancer. What GPUs should I choose? What architecture do I need?"
>
> [This is just an example..it should be clever enough to answer any Ai stack related question]

The AI should extract key requirements:

- Dataset size
- Problem domain (medical imaging → CNN, ViT, fine-tuning)
- Required compute characteristics
- Supporting tools (object storage, pipelines, model registry)

2.2 Engage in multi-turn conversation

The agent **must ask clarifying questions** and refine answers iteratively:

- What framework do you want (PyTorch, TensorFlow)?
- Do you require HIPAA/GDPR compliance?
- Are you training from scratch or fine-tuning?
- Do you prefer on-prem, cloud, hybrid?

The conversation must persist across turns reliably.

2.3 Recommend GPU instances + compute stack

The agent will suggest compute based on:

- Local cluster GPU availability & utilisation
- Cloud GPU types (A100, H100, L40s, MI300, H200, T4, etc.)
- Your pricing dataset
- Performance requirements (memory, FLOPs, PCIe vs NVLink, multi-GPU scaling)

It must output:

- Best GPU type
- Best cloud provider or local node
- Price/hour
- Estimated training/inference cost
- "Equivalent alternatives" (fallback options)

2.4 Recommend LLMs or Model Architectures

Using the Hugging Face marketplace metadata:

- Suggest classification / segmentation models
- Suggest foundation models for fine-tuning
- Suggest appropriate tokenizers, adapters, quantisation options
- Suggest LLMs for orchestration and reasoning

Output includes:

- Top 3–5 recommended models
- Licensing notes
- GPU compatibility
- Model size and expected memory usage

2.5 Recommend Supporting Components

Using the stack8s Kubernetes marketplace:

For AI/ML workflows:

- Vector DB (Weaviate / pgvector / Milvus)
- Object storage (MinIO)
- API Gateway (Kong, Ambassador)
- Inference framework (vLLM / Triton / Ray Serve)
- CI/CD pipeline (Argo Workflows / Tekton)

The AI should also output:

- Why the component fits
- What alternatives exist
- The minimal architecture diagram (text-based if needed)

2.6 Produce a Deployment Plan

Final output should include:

- Architecture summary
- GPU recommendation

- Training/inference infrastructure
- Kubernetes components required
- Estimated cost
- Deployment approach for stack8s

---

3. Data & Knowledge Base Integration

The freelancer must integrate **four knowledge bases** into the agent.

3.1 Local Cluster Compute Availability

API or DB source will contain:

- GPUs available
- GPU memory
- GPU utilisation
- Node labels (storage, networking, locality)
- Active jobs & reservations
- Queueing and scheduling rules

The AI must know whether:

- The workload *fits locally*,
- Needs *hybrid scaling*,
- Or should *burst to cloud*.

3.2 Cloud Compute + GPU Pricing

Dataset includes:

- Provider → instance types
- Number of GPUs
- GPU type (H100/H200/A100/T4/etc.)

- vCPU, RAM
- Network bandwidth

- Price per hour
- Multi-region availability

The AI must:

- Compare across providers
- Offer cheapest vs fastest choices
- Recommend fallback GPU options when stock is low
- Handle provider-specific constraints (spot, quotas, limits)

3.3 LLM Marketplace (Hugging Face)

Dataset includes:

- Model name
- Domain (NLP, vision, multi-modal)
- Size (parameters)
- Architecture (ViT, Llama, Qwen, Mixtral)
- GPU memory requirement
- Training samples
- Licence
- Downloads / popularity

The agent will use this to match models with user requirements.

3.4 Kubernetes Component Marketplace

This includes deployable stack8s-approved components:

- Databases
- Pipelines (Argo, Tekton)
- ML frameworks (vLLM, Triton, Ray)
- observability (Prometheus, Grafana, Loki)
- Messaging (NATS, Kafka)
- CI/CD
- Secrets managers

- API Gateways

Each must have:

- Supported versions
- Requirements (CPU/RAM)
- Helm charts
- Integration patterns

---

4. Technical Requirements

4.1 Architecture

The agent will be built as:

- A **FastAPI or NodeJS backend** (freelancer can choose)
- Connected to **OpenAI / Anthropic / custom model** for reasoning
- With a **vector database** (Weaviate, pgvector, Chroma, Milvus) to store:
    - Compute specs
    - Pricing
    - Model metadata
    - Marketplace components

4.2 Chat Interface

- Integrated into stack8s-portal UI
- Real-time chat
- Supports multi-turn reasoning
- Supports markdown and formatted outputs
- Can output YAML manifests or architectural diagrams

---

5. Core Deliverables

Phase 1 – Knowledge Layer

- Vector database schema
- Ingestion pipeline for 4 datasets

- API endpoints for querying compute, pricing, models, components
- Documentation of all datasets

Phase 2 – AI Reasoning Engine

- Prompt engineering
- System prompts for:
    - AI compute advisor
    - Architecture designer
    - LLM recommender
    - Kubernetes component planner
- Multi-turn memory
- Guardrails for accuracy

Phase 3 – Chat Agent Integration

- Chat UI inside stack8s-portal
- Streaming responses
- Error handling
- User context handling

Phase 4 – Output Templates

- Standardised outputs:
    - GPU recommendation table
    - Architecture summary
    - Deployment stack
    - Cost breakdown
    - Kubernetes manifest snippets

Phase 5 – Testing

- Test conversations
- Edge case handling
- Performance & latency testing

6. Example Conversation

User:

I have 2TB of MRI images and I want to create a model to detect lung cancer.
Please architect the best stack and find the right GPUs.

AI Architect Response (Expected):

1. Asks clarifying questions:
    - Do you want to train from scratch or fine-tune?
    - Are the images 3D or 2D MRI slices?
    - Do you need HIPAA/GDPR compliance?
    - On-prem or cloud preference?
2. Suggests compute options:
    - H100 80GB (fastest)
    - A100 80GB (balanced)
    - L4 or L40 (cheaper alt for inference)
3. Suggests models:
    - ViT for medical imaging
    - BioMedCLIP
    - MONAI pipelines
    - Or custom segmentation models
4. Suggests Kubernetes stack:
    - Object storage: MinIO
    - Pipeline: Argo Workflows
    - Model registry + tracking
    - Inference: Triton or vLLM
    - Vector DB for metadata
5. Offers deployment plan:
    - YAML to deploy pipeline
    - Cost estimation

- On-prem vs cloud GPU comparison

---

7. Acceptance Criteria

The AI Architect is “complete” when:

- It can hold **multi-turn, context-aware conversations**
- It recommends the **correct GPU compute** based on real resource availability
- It understands and queries **pricing + availability across clouds**
- It recommends **appropriate LLMs or ML models** from Hugging Face
- It recommends **full Kubernetes stacks**
- It can output **a deployable architecture plan**
- It passes 20+ test scenarios:
  - Image classification
  - LLM fine-tuning
  - Multi-GPU training
  - Hybrid cluster scaling
  - Real-time inference deployment

---

8. Skills Required

- Deep knowledge of AI/ML workflows
- Experience building agentic systems
- Prompt engineering
- Python/FastAPI or NodeJS backend
- Vector databases (pgvector, Weaviate, Milvus)
- Retrieval-augmented generation (RAG)
- Kubernetes concepts & DevOps pipelines
- Good understanding of GPU compute

---

9. Deliverable Summary

Phase	Deliverable	Description
1	Knowledge Base Layer	Data ingestion + vector DB
2	Reasoning Engine	Prompts, workflows, RAG
3	Chat Interface	Full UI + backend logic
4	Templates	Architecture, GPU selection, cost estimates
5	Testing	Real-world AI workload scenarios