# Agent Forge – Documentation & YAML Source (Docs Folder + Sample Definitions)

This document contains the full, detailed content of every file inside the project's `docs/` folder, as well as the sample Ethics and Team YAML definitions referenced by the system.

## definitions/ethics/basic\_framework.yaml

id: basic\_framework  
description: Default ethical framework that blocks hateful or violent content  
implementation: forge\_core.safety\_guardrails  
principles:  
 - id: no\_hate  
 statement: The agent must not generate hateful or harassing language.  
 keywords\_check:  
 forbidden:  
 - hate  
 - racist  
 - sexist  
 - slur  
 - id: no\_violence  
 statement: The agent must not promote or facilitate violence.  
 keywords\_check:  
 forbidden:  
 - bomb  
 - kill  
 - terrorist  
 - attack  
 - id: no\_private\_data  
 statement: The agent must not reveal private or personal data about real individuals.

## definitions/teams/sample\_team.yaml

id: math\_helper\_team  
description: Demonstrates coordinator/worker pattern for simple arithmetic tasks  
implementation: agents.coordinator\_agent.CoordinatorAgent  
coordinator\_agent\_id: simple\_agent\_v5  
worker\_agents:  
 math\_worker: simple\_agent\_v5  
coordination\_protocol: Hierarchical  
shared\_state\_schema:  
 type: object  
 properties:  
 intermediate\_results:  
 type: array  
 items: {type: number}

## docs/architecture.md

# Architecture Overview  
  
```  
+--------------------------+  
| forge\_ui.py | ← Streamlit front-end  
+--------------------------+  
 ↓ HTTP / CLI  
+--------------------------+  
| Agents | ← SimpleAgent, BTAAgent, CoordinatorAgent  
+--------------------------+  
 ↓ Calls  
+--------------------------+  
| Skills / Tools | ← Capabilities that perform work  
+--------------------------+  
 ↓ Registry look-ups  
+--------------------------+  
| forge\_core |  
| • ComponentRegistry |  
| • AgentBuilder |  
| • BehaviourTree engine |  
| • Evaluation Harness |  
| • Logging & Guardrails |  
+--------------------------+  
 ↓ I/O  
+--------------------------+  
| External Services |  
| • LLM via Ollama |  
| • ChromaDB Vector DB |  
| • DuckDuckGo API |  
+--------------------------+  
```  
  
## Data Flow  
  
1. \*\*Prompt\*\* enters via CLI or UI.  
2. UI/CLI asks `AgentBuilder` to instantiate the requested agent.  
3. The agent runs `run(prompt)` – may tick a behaviour-tree that triggers Skills/Tools.  
4. Each component emits a JSONL trace line (`forge\_logging`).  
5. Output is returned; Evaluation harness (optional) scores and logs results.  
  
## Extension Points  
  
| Layer | How to Extend |  
|-------|--------------|  
| \*\*Tool\*\* | Add a class under `capabilities/` inheriting `BaseTool`, declare YAML under `definitions/tools/`. |  
| \*\*Skill\*\* | Same as Tool, but inherit `BaseSkill`, can orchestrate multiple tools. |  
| \*\*Agent\*\* | Create class under `agents/`, implement `run()`; reference it in a YAML under `definitions/agents/`. |  
| \*\*Behaviour-Tree\*\* | Write YAML under `strategies/` using `Sequence`, `Selector`, `Action`. |  
| \*\*Ethics\*\* | Define principles & keyword checks under `definitions/ethics/`, assign IDs to agents or teams. |  
  
## Sequence Diagram (Agent with Skill)  
  
```mermaid  
sequenceDiagram  
 participant UI/CLI  
 participant Agent  
 participant Skill  
 participant Tool  
  
 UI/CLI->>Agent: run(prompt)  
 Agent->>Skill: execute(run\_id, parameters)  
 Skill->>Tool: execute(run\_id, parameters)  
 Tool-->>Skill: results  
 Skill-->>Agent: structured data  
 Agent-->>UI/CLI: final answer  
```

## docs/getting\_started.md

# Getting Started  
  
Follow this hands-on guide to spin up \*\*Agent Forge\*\* on your machine.  
  
---  
## 1. Prerequisites  
  
\* Python \*\*3.10+\*\*  
\* Git  
\* (Optional) [Ollama](https://ollama.ai/) running locally for offline LLMs  
\* (Optional) Docker Desktop if you prefer containerised setup  
  
---  
## 2. Installation  
  
```bash  
# Clone repository  
$ git clone https://github.com/your-org/agent-forge.git  
$ cd agent-forge/agent\_forge\_project  
  
# Create virtual environment  
$ python -m venv .venv && source .venv/bin/activate  
  
# Install Python dependencies  
$ pip install -r requirements.txt  
```  
  
If you plan to hack on documentation:  
  
```bash  
$ pip install mkdocs-material mkdocs-mermaid2-plugin  
```  
  
---  
## 3. Configuration  
  
Global settings live in \*\*config.yaml\*\* – update LLM model names or vector-store path as needed:  
  
```yaml  
ollama\_api\_url: "http://localhost:11434/api/generate"  
vector\_store\_path: "./chroma\_db"  
log\_level: INFO  
```  
  
---  
## 4. Run Your First Agent  
  
```bash  
# Echo agent smoke test  
$ python run\_forge.py --agent simple\_agent\_v5 --prompt "Hello, Forge!"  
# → Should print: ECHO: Hello, Forge!  
```  
  
Launch Streamlit UI:  
  
```bash  
$ streamlit run forge\_ui.py  
```  
  
---  
## 5. Adding a New Skill  
  
1. Create `capabilities/my\_skill.py` inheriting `BaseSkill`.  
2. Define its YAML under `definitions/skills/my\_skill.yaml`.  
3. List the skill ID in the target agent's `allowed\_skills` array.  
4. Re-run the agent – the registry auto-reloads on startup.  
  
---  
## 6. Running Tests & Evaluation  
  
```bash  
# Install test deps  
$ pip install pytest  
  
# Execute all test cases  
$ pytest -q  
  
# Run evaluation harness against YAML test cases  
$ python -m forge\_core.evaluation  
```  
  
JSONL results are written to \*\*logs/evaluation\_results.jsonl\*\*.  
  
---  
## 7. Troubleshooting  
  
| Symptom | Remedy |  
|---------|--------|  
| `ModuleNotFoundError` for a capability | Check `implementation:` dotted path in YAML, ensure file is on `PYTHONPATH`. |  
| Empty response from agent | Look at `logs/agent\_execution.jsonl` for detailed trace. |  
| ChromaDB errors | Delete `chroma\_db/` folder and re-initialise, or ensure correct embedding model is configured. |  
  
If stuck, open an issue on GitHub with logs attached.

## docs/index.md

# Agent Forge Documentation  
  
Welcome to \*\*Agent Forge\*\* – an open-source workbench for creating, testing and evolving AI agents.  
  
## Key Features  
  
\* \*\*Modular Components\*\* – Agents, Skills, Tools, Ethics, Teams, Strategies all declared in YAML and validated by Pydantic.  
\* \*\*Behaviour-Tree Engine\*\* – Plug agent logic together with declarative control-flow.  
\* \*\*Retrieval-Augmented Generation\*\* – Out-of-the-box document chunking and ChromaDB vector store.  
\* \*\*Evaluation Harness\*\* – Batch test-cases, JSONL logs, LLM-based judging, evolutionary search.  
\* \*\*Streamlit UI\*\* – Prompt, run, and inspect traces in real-time.  
  
## Quick-Start  
  
```bash  
# 1. Clone & create virtual-env  
$ git clone https://github.com/your-org/agent-forge.git && cd agent-forge  
$ python -m venv .venv && source .venv/bin/activate  
  
# 2. Install dependencies  
$ pip install -r requirements.txt  
  
# 3. Run a smoke-test agent  
$ python run\_forge.py --agent simple\_agent\_v5 --prompt "2 + 2?"  
  
# 4. Launch the UI  
$ streamlit run forge\_ui.py  
```  
  
## Repository Layout (excerpt)  
  
```  
agent\_forge\_project/  
├─ forge\_core/ # engines & utilities  
├─ capabilities/ # Skills & Tools  
├─ agents/ # Agent classes  
├─ definitions/ # YAML configuration files  
├─ strategies/ # Behaviour-Tree YAMLs  
├─ test\_cases/ # Automated tests  
└─ docs/ # This documentation site  
```  
  
## Navigating the Docs  
  
\* [Architecture](architecture.md) – Deep dive into layers & data-flow.  
\* [Getting Started](getting\_started.md) – Installation, configuration, and first agent.  
\* Evaluation, RAG, Extending, Roadmap (coming soon)  
  
> Tip: Build the docs locally with \*\*mkdocs-material\*\*:  
>  
> ```bash  
> pip install mkdocs-material  
> mkdocs serve  
> ```