# QType

**QType is a domain-specific language (DSL) for rapid prototyping of AI applications.**  
It is designed to help developers define modular, composable AI systems using a structured YAML-based specification. QType supports models, prompts, tools, retrievers, and flow orchestration, and is extensible for code generation or live interpretation.

## 🚀 Quick Start

Install QType:

pip install qtype[interpreter]



Create a file hello\_world.qtype.yaml that answers a question:

id: hello\_world

flows:

- id: chat\_example

description: A simple chat flow with OpenAI

mode: Chat

steps:

- id: llm\_inference\_step

model:

id: gpt-4

provider: openai

auth:

id: openai\_auth

type: api\_key

api\_key: ${OPENAI\_KEY}

system\_message: |

You are a helpful assistant.

inputs:

- id: user\_message

type: ChatMessage

outputs:

- id: response

type: ChatMessage



Put your openai api key into your .env file:

echo "OPENAI\_KEY=sk...." >> .env



Validate it's semantic correctness:

qtype validate hello\_world.qtype.yaml



You should see:

INFO: ✅ Schema validation successful.

INFO: ✅ Model validation successful.

INFO: ✅ Language validation successful

INFO: ✅ Semantic validation successful



Launch the interpreter:

qtype serve hello\_world.qtype.yaml`



And go to <http://localhost:8000/ui> to see the user interface for your application:

Example UI

See the [full docs](https://bazaarvoice.github.io/qtype/) for more examples and guides.

## 🤝 Contributing

Contributions welcome! Please follow the instructions in the [contribution guide](https://bazaarvoice.github.io/qtype/contributing/).

## 📄 License

This project is licensed under the **MIT License**.  
See the [LICENSE](http://./LICENSE) file for details.

## 🧠 Philosophy

QType is built around modularity, traceability, and rapid iteration. It aims to empower developers to quickly scaffold ideas into usable AI applications without sacrificing maintainability or control.

Stay tuned for upcoming features like:

* Integrated OpenTelemetry tracing
* Validation via LLM-as-a-judge
* UI hinting via input display types
* Flow state switching and conditional routing

Happy hacking with QType! 🛠️

[Generate JSON Schema](https://github.com/bazaarvoice/qtype/actions/workflows/github_workflows_generate-schema.yml) [Publish to PyPI](https://github.com/bazaarvoice/qtype/actions/workflows/publish-pypi.yml)