fetch.ai

Hackathon Guide

Fetch.ai

Agenda

1. Introduction
2. uAgents, AI Engine, Agentverse, DeltaV
3. Walkthrough

Let’s talk about AI, today.

AI -> AGI -> ASI

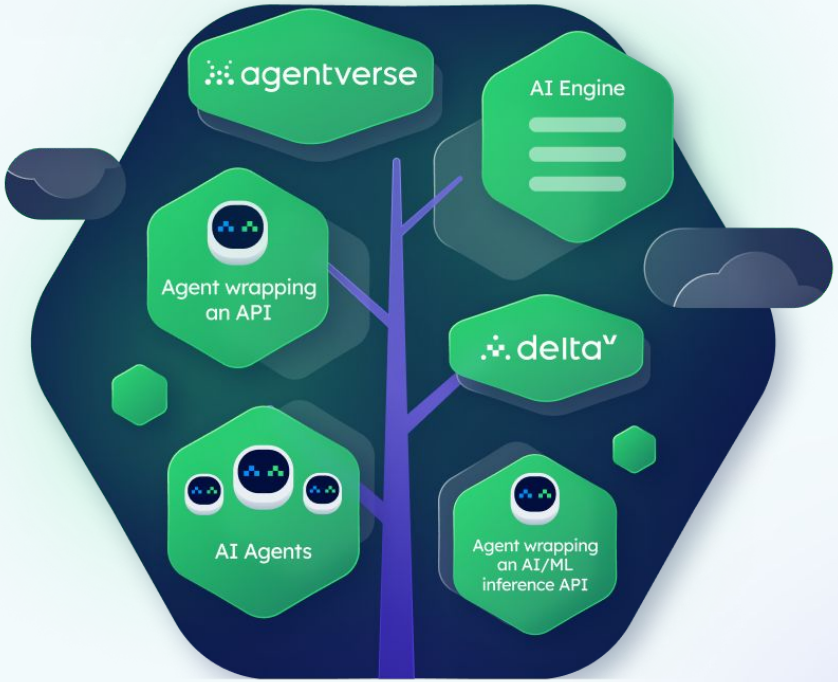
LLMs lack ‘action’

- LLMs can’t *act*
- Example: ChatGPT
- LLMs need *to act*.
- To act = *Agentia* (latin)

What are AI Agents?

- Pieces of code that ***perceive*** their environment and ***take actions*** in order to ***achieve specific goals***
- Usually, interact autonomously with other agents

What is Fetch.ai, and where does it fit in?



Components

1. **AI Agents** are independent decision-makers that connect to the network and represent data, APIs, services and people
2. **Agentverse** serves as a development platform for these agents
3. **AI Engine** links human input to AI actions
4. **DeltaV** is a simple user interface for non-technical users
5. **Fetch Network** underpins the entire system, ensuring smooth operation and integration

Integrations, Solutions, Applications

- **Integration**: Third party tools/apps/APIs that can be directly integrated with uAgents + DeltaV. Example: Google Docs, RapidAPI, etc.

Integrations, Solutions, Applications

- **Integration**: Third party tools/apps/APIs that can be directly integrated with uAgents + DeltaV. Example: Google Docs, RapidAPI, etc.
- **Solution**: Combination that ‘chains’ multiple integrations to achieve a use-case on DeltaV. Example: Fetching latest data from a particular API from RapidAPI and feeding it into Google Docs.

Integrations, Solutions, Applications

- **Integration**: Third party tools/apps/APIs that can be directly integrated with uAgents + DeltaV. Example: Google Docs, RapidAPI, etc.
- **Solution**: Combination that ‘chains’ multiple integrations to achieve a use-case on DeltaV. Example: Fetching latest data from a particular API from RapidAPI and feeding it into Google Docs.
- **Application**: An end-to-end application that uses uAgents. Doesn’t use the DeltaV interface. Is a standalone app (example: Blockagent)

Quick Walkthrough: From Zero to DeltaV

Step One: uAgents

Step Two: AgentVerse

Step Three: DeltaV/App

Step One: uAgents

1. Install uAgents library (Start here)
2. Try creating your first uAgent (Start Here)
3. Create two agents and start an interaction between them (Start Here)

You can stop following the guide when you reach the section ‘Booking a table at a restaurant’.

Step Two: Agentverse

1. Create an Agentverse account on https://agentverse.ai
2. Create an Agent on online IDE of Agentverse (Link to guide)
3. Use the mailbox service to create a local agent (Link to guide)

You need to use mailbox service if you are using a python module apart from the ones listed here. It is IMPORTANT to note that due to security reasons, Agentverse doesn't allow import of all modules. But you can run your agent locally and connect it to Agentverse via Mailbox.

Step Three: DeltaV

1. Register a Hugging face API agent as a service (Link)
2. Now try your API/Problem Statement and connect it to DeltaV.

For more examples, you can check the integrations repository. Remember that you DO NOT need to build your own UI. Your project must be accessible from DeltaV.

If you get stuck, want to explore more, visit fetch.ai/docs

Your Objective

a. Build a solid application using uAgents using custom frontend

OR

b. Build a solid solution (chained integrations) that works on DeltaV

Examples

If you’re using your custom frontend

- You don’t need to raise a PR
- You just need to submit the link of your project (you can host it on free cloud services) in a Google Form (to be shared later)
- You will need to demo the project at Paradox (date TBD)

If you’re using Agentverse/DeltaV

- You need to raise a PR
- You just need to submit the link of your repo in a Google Form (to be shared later)
- You will need to demo the project at Paradox (date TBD)

Here’s what we would be judging you on

- **Functionality**: Is your application functional end-to-end?
- **Creativity**: Are you merely ‘integrating’ something - or are you presenting a business use-case as well?
- **Code Quality**: Is your code clean, organized, and well-commented?

Directory Structure


```
├── poetry.lock
├── pyproject.toml
├── README.md
└── src
    ├── agents
    │   ├── module_x
    │   │   ├── model_x.py
    │   │   └── __init__.py
    │   └── __init__.py
    ├── __init__.py
    ├── main.py
    ├── messages
    │   ├── message_x.py
    │   ├── __init__.py
    └── utils
        └── utility_x.py
```

README file

- Project name
- Description of the project
- Instructions to run the project
- Use-case example
- Special considerations, if any

Remember, a good README will help judges arrive to a decision quickly!

API Keys and Environment Variables

If your project involves the use of API keys or other sensitive data:

- Do not push sensitive keys or passwords to the public repository.
- Instead, include a sample .env file with placeholders for these values and detailed instructions on where to get the actual values and how to set them up.

FAQs

> Do we have to use uAgents?

Yes, it’s a must.

> Agentverse doesn’t import X library

Yes, because there are security restrictions. You can create a local agent and connect it to a mailbox

> Can I use any additional API/integration and connect two agents

It would be AWESOME and would earn you extra brownie points.

FAQs

> But do we **really** have to use uAgents?

Yes, it’s a must.

FAQs

> Can we skip using uAgents?

Nope.