# Agent Academy - Project

We hope that you have become somewhat familiar with the Open AEA and the Open Autonomy Frameworks, and that you have a basic understanding of how all the pieces fit together: AEA components and Agent Services architecture.

The next step in the journey is to engage in building a new, disruptive decentralized service. This will be a more interactive part where we are proposing that Academy participants work together, helped by some of our engineers. This will be the focus of Module 3.

In order to enter this next stage in the Academy, we require that you complete a small project so that we can assess your understanding of the framework and the concepts involved. It does not have to be “perfect”, but we would like to ensure that you have understood the necessary elements to maximize your potential in building new services. The Academy will take place periodically, so you can take the assessment when it better suits you, and you will be assigned to the next available set of lab sessions.

As always, the Autonolas Discord is available for community support if you are stuck at any point, but we would like to see that the submission that you present is the result of your own effort.

## Exercise 1:

At this point, we can look to make modifications to the existing hello world FSM App. The objective is to modify the behavior of the Hello World FSM App in order to add additional functionality. As a reminder, follow these instructions to install the Open Autonomy framework from PyPI.

The additional functionality you are tasked with implementing at this point is to include another round and behavior associated with a new state into the Hello World FSM App.

The purpose of this round will simply be to transition to the next round, so there is no need to add any additional functionality beyond entering into a new round which we shall call “MyNewRound”

In order to accomplish this you will need to implement:

- A Round
- A Payload
- A Behavior
    - Within this behavior we would like to log to the console

      “Sent from MyNewRound!”

And of course, the most important thing of all is an associated test to ensure that the string is correctly printed. This test should be based on the e2e test located here;

## Exercise 2:

Now we have modified an Abci application we will look to further extend its functionality. Next, we will look to take advantage of the composability of the entire framework. We are going to add a new connection to our ABCI application.

Using our work from the previous exercise we will look to:

- Add a http client connection to the application.
- Use this new connection to retrieve a price from ftx.
- Print the price in the format:
    - "Got a new price from FTX: {new_price}”
- Push the component to our IPFS registry.

The composable nature of the framework allows this with remarkably little code. Some guidance and hints are found below.

- You must add the http client to the agents. This is the implementation of the client which we were introduced to previously.
- You might find that our price estimation abci provides inspiration.
- You will need parameters for your http request. Please use the following arguments for messages sent to the client connection:

  args:

  url: https://ftx.com/api/markets/BTC/USD

  api_id: ftx

  response_key: result:last

- When pushing your component, you will need to export a registry url: OPEN_AEA_IPFS_ADDR="/dns/registry.autonolas.tech/tcp/443/https"

Once you have successfully pushed your package, you will receive an output as so;

> Pushed component with:
>
> PublicId: open_aea/signing:1.0.0
>
> Package hash: QmWdCAiXpeT1tgrcf8uofKdQ6fLxMH77D5jwgQbbMo41df

Now you have accomplished these goals, please submit your solution and sign up to our Agent Academy where we will show you how to take this further!