**Instruction**

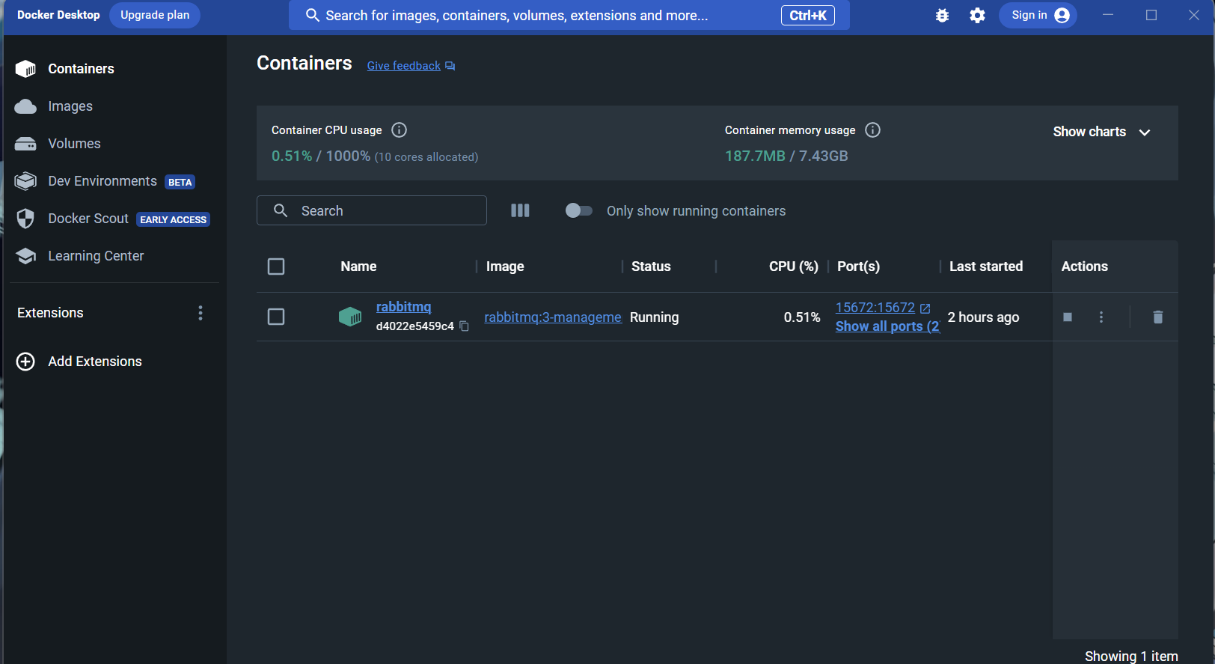
Docker:

1. Open terminal window and run:

docker run -it --rm -d --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3-management

(This create the container in a long string code in a docker that will be used throughout the Prototype)

1. Open the docker file and make sure the container is running all the time



Important libraries needed to run the application:

1. Pika : pip install pika
2. Tkinter: pip install tkinter
3. Json : pip install json
4. Virtual Environment: pip install virtualenv
5. Parser: pip install argparse
6. Rabbit: pip install rabbit

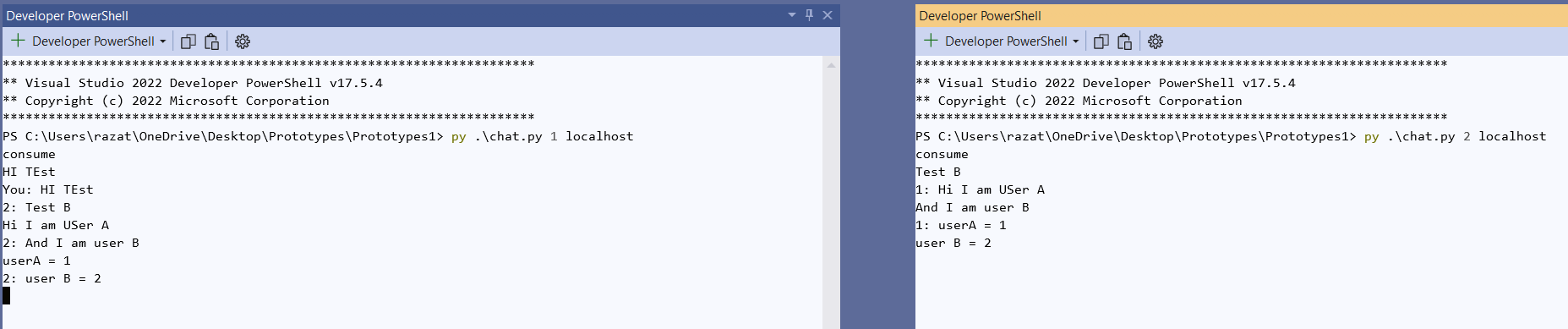
**Prototype1 :**

1. Open the Prototype1 folder in terminal of windows or Powershell of Visual studio code
2. Run the chat.py file by entering following commands in terminal:
3. py .\chat.py 1 localhost (to create user no. 1)
4. Open separate terminal window and run :

py .\chat.py 2 localhost (to create user no.2)

1. You can use : py .\chat.py n localhost (for n number of users)

Now the prototype can be used as chatting application.

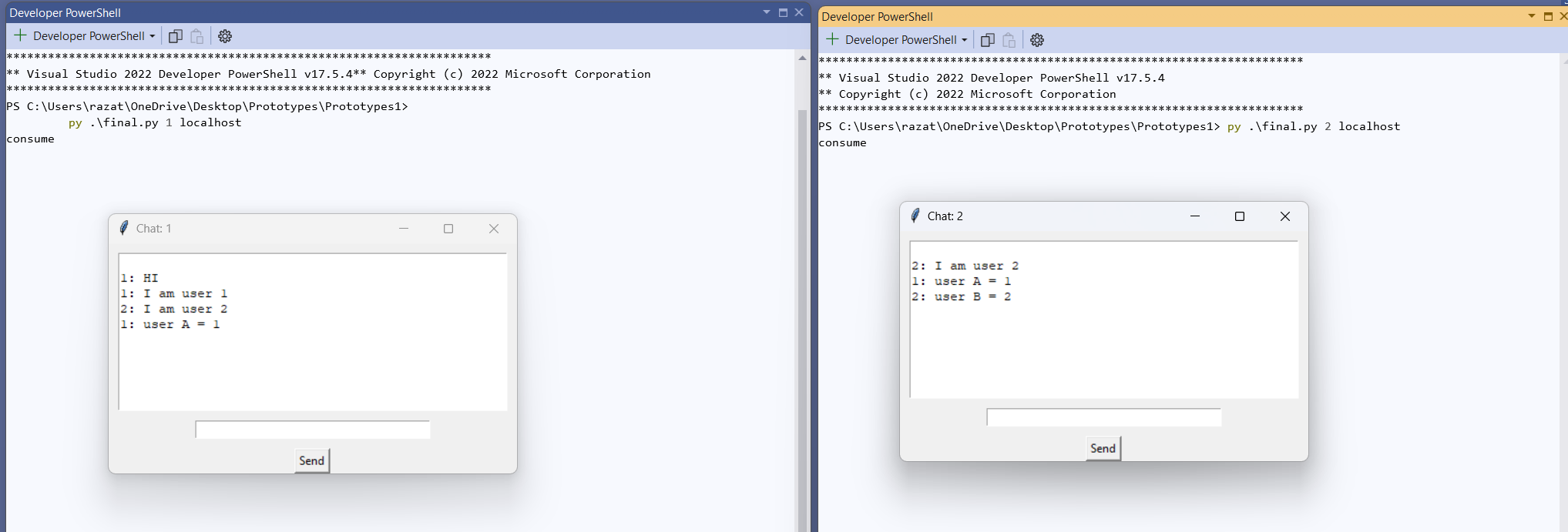


To run GUI program of chatting Application

Run a. py .\final.py 1 localhost

b. py .\final.py 2 localhost

on 2 separate terminal to open GUI chatting application



**Prototype 2: Trading Application**

(always first) In terminal 1 run : py .\exchange.py localhost

Then in terminal 2 : py .\sendOrder.py ‘user’ localhost ‘action’ ‘quantity’ ‘price’

Eg: py .\sendOrder.py 1 localhost buy 100 110

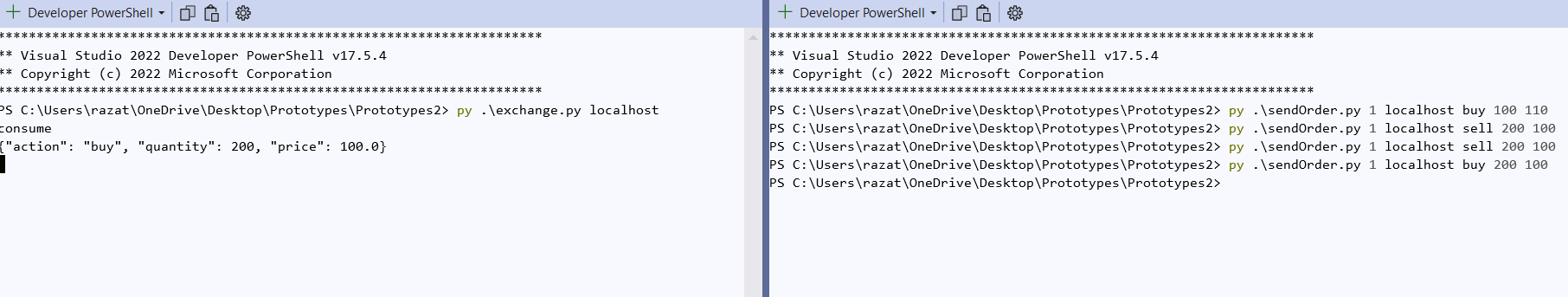
Eg: py .\sendOrder.py 1 localhost sell 200 100

The in Exchange terminal (Terminal 1), the trade action is printed only if the trade is successful i.e

existing\_order.action == "buy" and existing\_order.price >= current\_order.price

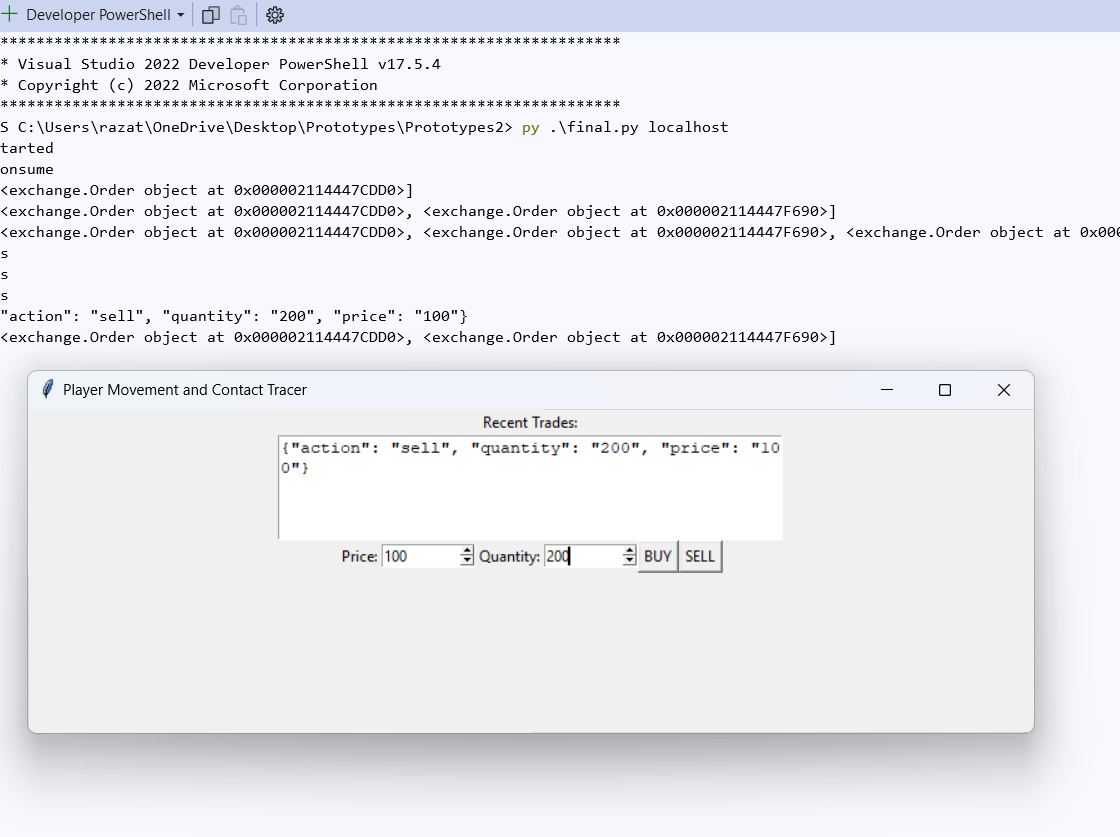
or

existing\_order.action == "sell" and existing\_order.price <= current\_order.price



GUI application

Run to open GUI app : py .\final.py localhost



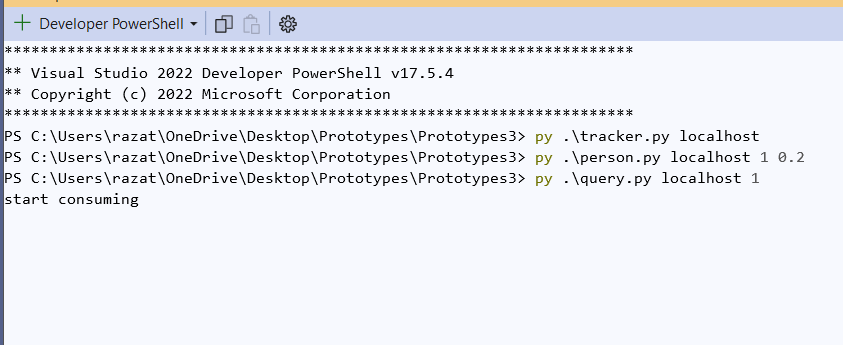
Prototype3 :

In terminal run following commands in order:

py .\tracker.py localhost

py .\person.py localhost ‘playerusername’ ‘speed’

py .\query.py localhost 1

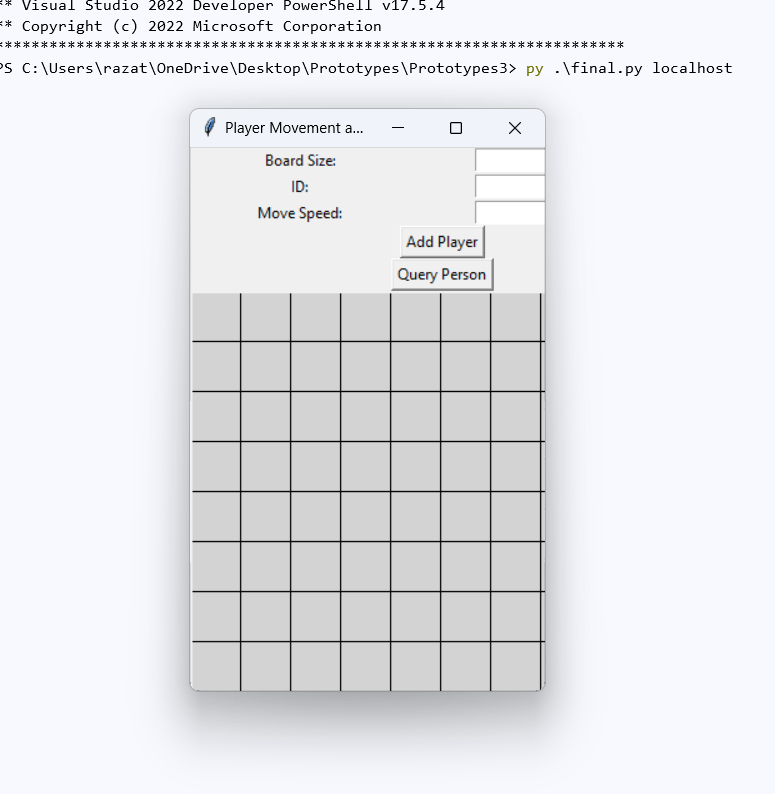


We cant see any results because the prototype is about contract tracing in 2D grids

GUI Application:

Run in terminal to open grid board:

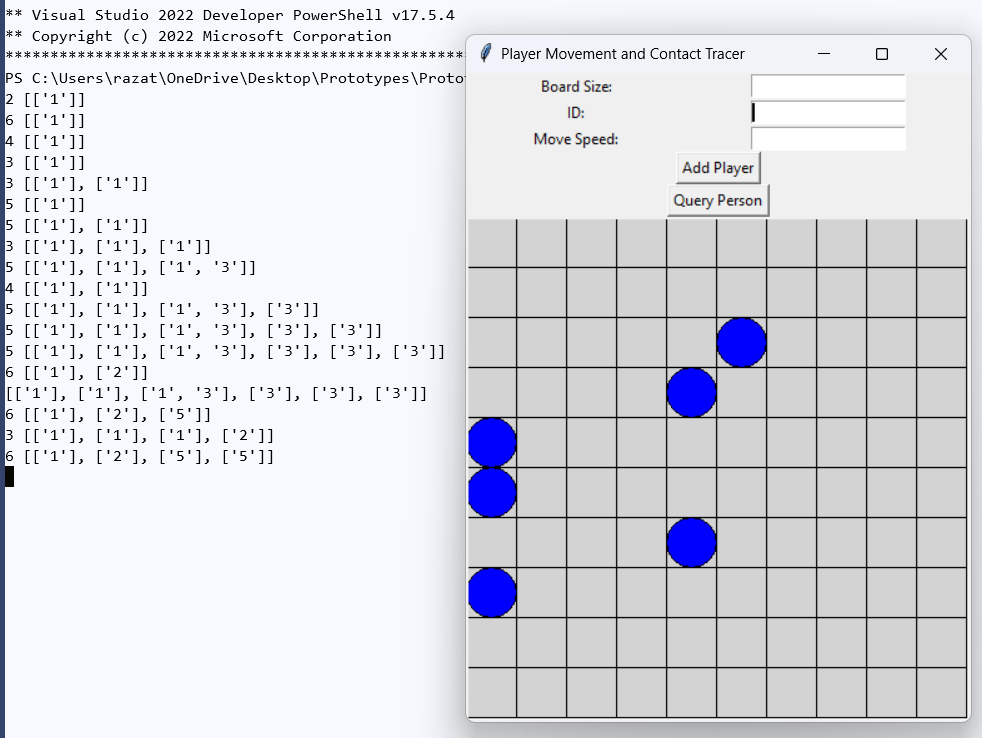
py .\final.py localhost



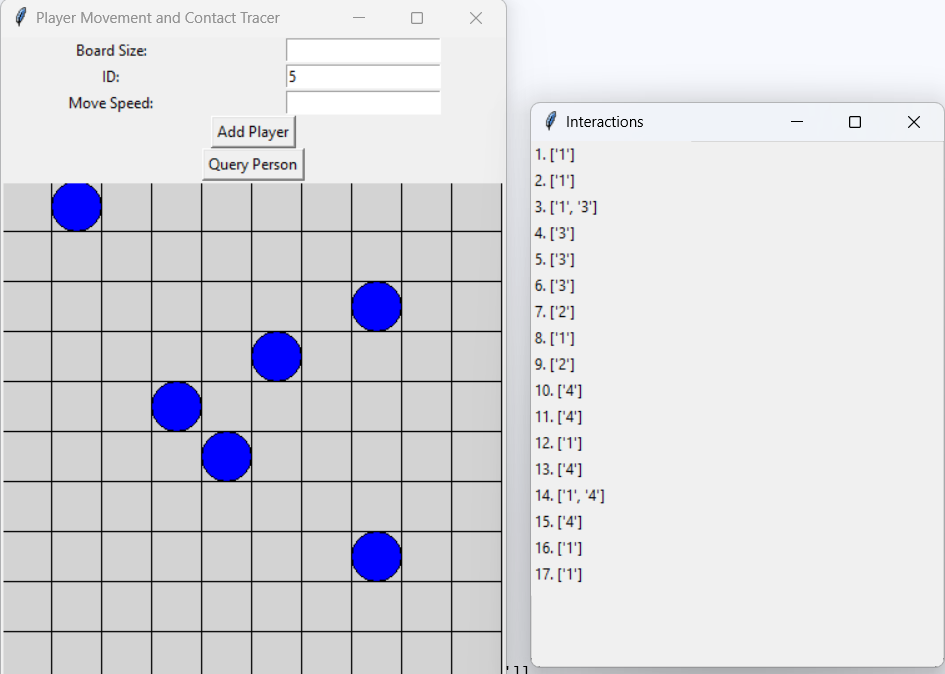
Here you can add players of desired number, change speed and trace their movements

Showcase:

Lets add 6 player with each unique id: 1,2,3,4,5,6



Which player interacted with whom and how many times in sequential order. To get query of specific player, we can put the player id and press ‘query’



Here, we can see that player 5 had 17 interactions with other players in sequential order.