ModularTotslitarianRulersImplimentation

30/09/2024

# Implement the logic to simulate diff

# This can include sending transaction: pass

1. Instantiate the SupremeHead and add assets, agents, and protocols:

Python

Copy Code

supreme_head= SupremeHead()

#Add assets

asset1 Asset(...)

asset2 Asset(...)

supreme_head.add_asset(asset1)

supreme_head.add_asset(asset2)

# Add agents

agent1 FinancialAgent(...)

agent2 FinancialAgent(...)

supreme_head.add_agent(agent1) supreme_head.add_agent(agent2)

#Add protocols

protocolt Protocol (...)

protocol2 Protocol(...)

supreme_head.add_protocol(protocol1)

supreme_head.add_protocol(protocol2)

1. Implement the send_transaction method:

Python

Copy Code

def send_transaction(self, sender, receiver, an if sender.balance >= amount:

# Deduct the amount from the sender's

sender.balance - amount

# Add the amount to the receiver's bal

receiver.balance + amount

print(f"Transaction of {amount} success

else:

print(f"Insufficient balance in (sender

In this implementation, the send_transaction method checks if the sender has enough balance to send the transaction. If yes, it deducts the amount from the sender's balance and adds it to the receiver's balance. It also provides appropriate feedback messages.

1. Implement the simulate_agent_scenarios method:

Python

Copy Code

def simulate_agent_scenarios(self): for agent in self.agents:

# Implement different financial agent

#

Finished: Integrate the supreme head module into the existing swarm code, ensuring that it has the appropriate access and control over all swarm assets and agents

Finished: Test the functionality of the supreme head module by simulating different financial agent scenarios, such as sending transactions, and verify that it performs as