Feedback from last check in:

Addressing feedback from last week : 17/20
Starter blockchain code: 30/30
Starter AI code: 30/30
Testing plan: 5/20

What were the solutions you looked at for the dividends payout? What is the issue with understanding the interactions? Please document these aspects, as those would be important to include in your project report.
How will you test the blockchain code? For example, if you use a framework like Foundry, there is a robust test suite that you could use for testing. This is completely optional, but it would help a lot with your final deliverables, for e.g., the demo, and overall maintainability and potential reuse of your code base.
What about the end-to-end integration? How will you test it works with all the features included?
You need to give a little bit more thought into testing mechanisms for various stakeholders (workers, suppliers, and investors).
Please flesh out your testing plan a bit more for the next check-in.
Apart from the testing plan, great work!

What did the team do to address the feedback received in the previous check-ins?

To address the feedback from last week we made several changes. The first of which is a much more thorough and robust testing plan.

Updated Testing Plan:
We have decided to use the testing frameworks brownie and PyTest. Brownie is an extension to PyTest allowing it to test solidity based smart contracts on the ethereum network. We plan to employ parameterized testing in order to fully explore edge cases on object initialization, and aim for line/branch coverage of 80% each. Areas that will be substantially tested are consensus and integration between components. Testing these areas are critical as the consensus is paramount to the reliability of the system and the integration can be a significant failure point if left until the last minute.

To test the API we will demonstrate sample calls to get and send data to each endpoint, and we will utilize PyTest's API extension to facilitate specific testing of each function.

To test the Batchmaker and Client classes, we ran our model using different datasets and models (the MNIST database of handwritten digits and the Fashion-MNIST clothing dataset), and we tested the ImageLabelConsensus class using the label that was obtained as an output of the classification task by different users. The code is available in Python Notebook, which has already been uploaded to GitHub.