Name: Qiyuan Huang, Ruoyu Gao, Qicheng Li

Date: 5/21/2022

Section: 001

# Final Project Part II

**Total in points** (100 points total):

# Professor’s Comments:

**Affirmation of my Independent Effort: Qiyuan Huang, Qicheng Li, Ruoyu Gao**

**(Sign here)**

Cloud Computing

**NFT Card Game**

Report



### **Diagram Description automatically generated**

## Implementation Steps

* We implemented the Ethereum card smart contract for holding immutable card information on blockchains. The card contract requested ChainLink RNG for a verified random number to ensure the cards’ data would be randomly generated in each mint process.
* We implemented the Ethereum marketplace smart contract to allow each P2P node to trade cards without depending on any centralized server.
* We implemented the P2P node by using React/Javascript/Node.js. Since even the card images are holding on IPFS, Any P2P node can mint/trade cards without depending on any centralized server.
* We designed and implemented a centralized server by using Flask. The server accepts P2P nodes’ PVP battle requests and sends the battle results back to P2P nodes. We deployed the Flask server on EC2.
* We deployed an Azure table for storing the card deck information. The P2P node can pick any set of cards, compile them as a card deck and store them in the Azure table.
* We implemented a weather-related battle system. The weather where the IoT sensor is located would affect the battle. The game server would retrieve the weather data from the IoT hub and use it as a factor for battle result calculation.

## Ethereum Contract deployment

1. Compile Card and Marketplace contracts first.

A computer screen capture

Description automatically generated with medium confidence

1. Deploy the two contracts on the Kovan testnet

## Graphical user interface, text Description automatically generated

Text

Description automatically generated

## EC2 Deployment

1. Create Ubuntu EC2 on AWS

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. Store key pairs, ssh into Ubuntu EC2

Graphical user interface, text

Description automatically generated with medium confidence

1. Setup and activate the required virtual environment

Text, letter

Description automatically generated

1. Run the Flask server on Ubuntu

Graphical user interface, text, application, letter, email

Description automatically generated

## Azure Table Deployment

1. Create Azure Storage and Azure table via Azure portal

Graphical user interface, text, application, email

Description automatically generated

1. Create a python API that store data into the Azure table

Text

Description automatically generated

1. Create a python API that retrieve data into the Azure table

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**GitHub Link: https://github.com/cookythecat/cookythecat-NYU-CC-PROJECT-2**