NAME : ANSHIKA SINGH

Roll_no : 230163

TITLE : BLOCKBLOOM_ASSIGNMENT_2

Q1) MetaMask Account Address:- 0x975a41448D0cb12Efa1E533D18B14EA5f7F4dcf1

Q2) Summary on Ethereum Whitepaper:-

The Ethereum whitepaper, written by Vitalik Buterin , white paper as the second generation of cryptocurrencies, Ethereum is not only a cryptocurrency used as a means of transactions but also encompasses wider possibilities. The creation of a so-called 'world computer', whereby it is possible for people to create and execute smart contracts and decentralized applications dApps, is what Ethereum is all about.

In conclusion, Turing completeness and smart contracts takes centre stage in the Ethereum blockchain ecosystem. Users can use native scripts on the Ethereum blockchain as a means to create a diverse range of smart contracts with intricate terms and conditions in them. Smart contracts are digital agreements that are enacted when different events occur on a blockchain. The significance of such contracts is the facilitation of several functions from funding transactions to governance mechanisms within decentralized systems.

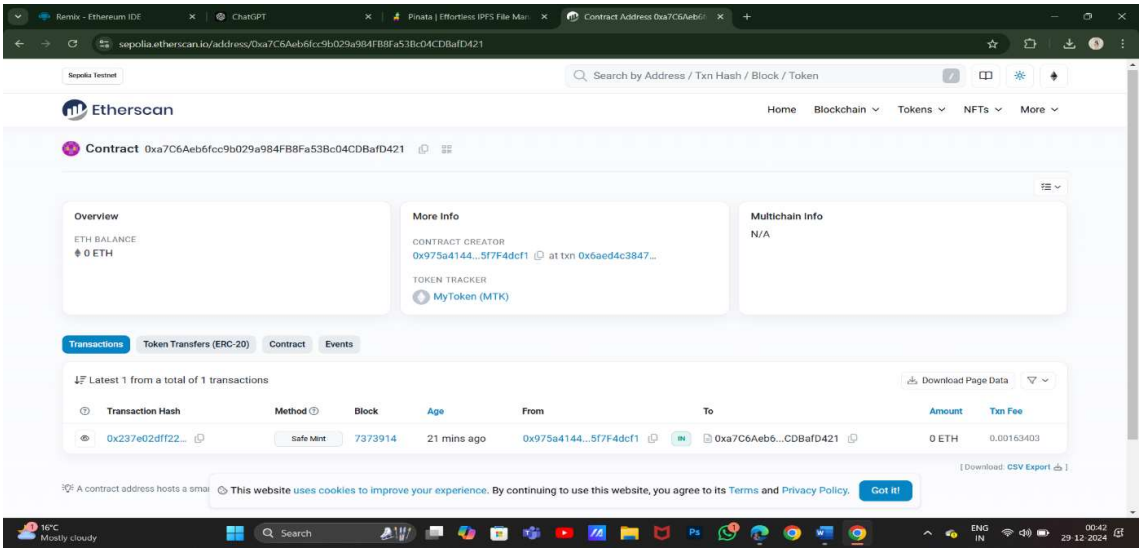
As such Ethereum gives rise to DAOs (Decentralized Autonomous Organizations) which can be smart contracts on the Ethereum network and automatically execute the decisions made by them. With that such funds can be deployed or with DAOs regulations enforced.

Q3) Contract Address using ERC 721:- 0xa7C6Aeb6fcc9b029a984FB8Fa53Bc04CDBafD421

Verification link on Opensea :-

https://testnets.opensea.io/assets/sepolia/0xa7C6Aeb6fcc9b029a984FB8Fa53Bc04CDBafD421/0

Attached Screenshot of Sepolia Etherscan for Minted NFTs:-



Q4)

A cryptocurrency wallet is certainly the easiest way to interact with cryptocurrencies since, without it, it is impossible to own and have access to cryptocurrencies. All that the creator of the cryptocurrency needs to do is sign off on a transaction using a private key. The block is then added to a blockchain network and turns on the cryptocurrency. A crypto wallet does not in fact possess the physical coins, rather it serves as an authorization token to carry through necessary transactions on the cloud-based network.

Types of Wallets:

1-Software Wallets: This includes applications that can run in mobile devices, tablets and desktops. Some of the few are:

Hot wallets: In order to make things easier, there is this wallet that is always online (MetaMask, Trust Wallet, and Exodus).

Advantages: Transactions are very simple and are easy to do since the wallets are easily linked to your devices.

Disadvantages: They are always at risk of being hacked and are prone to getting infected due to the endless stress on the internet.

2-Hardware Wallets: These are wallets in the form of physical devices. The purpose of a hardware wallet is to allow the users the use of offline private key storage (Ledger, Trezor and Keepkey).

Advantages: It offers good security, its hard to hack, and its ideal for users that are storing their assets for a long time.

Disadvantages: It does requires an initial payment to buy for the first time and it can be burdensome to use if making multiple transfers.

Decentralization and Wallets:

Basically any wallet allows you to keep your private keys which could be described as an authorization token to access your account and allows you to operate the entire network without relying on centralized authorities such as traditional banks or exchanges. Non-custodial wallets are especially beneficial since they allow you to have full ownership of your assets without involving a third party.

Example of Wallet(Other than Metamask):-**Trust Wallet**,**Exodus**,**Ledger Nano X**,**Trezor**,**Atomic Wallet**

Q5)

A zero address in Ethereum that blockchain more than just a note is the address 0x0000000000000000000000000000000000000000.

Use as a Smart Contract Address: In the Ethereum ecosystem, it is frequently utilized as an address where a token does not exist for instance:

When a token is mints, a zero addresses serves as a 'from address' which means that the tokens are being created on account of the user instead of being transferred from a user account.

When a token is burned, its 'to address' is changed to a zero address meaning in lay man's terms that the tokens have been destroyed.

Denote an Error For example: To perform some minting or smart contract games Ethereum developers are provided with a zero address that acts as an error or warning. An account should realistically not involve itself with zero addresses and use it trigger an action instead.

Difficulties relating to Guessing the Private Key Derived from a Zero Address

There's no link that connects a zero address to its private key hence there exists no link between them. A private key corresponds to a 256 sequence of bits and a key is drawn by embedding everything into a 256 key. It is more or less impossible to approach or create a private key that is embedded or approaches the 0x0000000000000000000000000000000000000000 address owing to the fact that there exists no such private key. Additionally, it is transactional, secure as a placeholder which means that no individual can inappropriately utilize it since malicious interference is close to impossible.

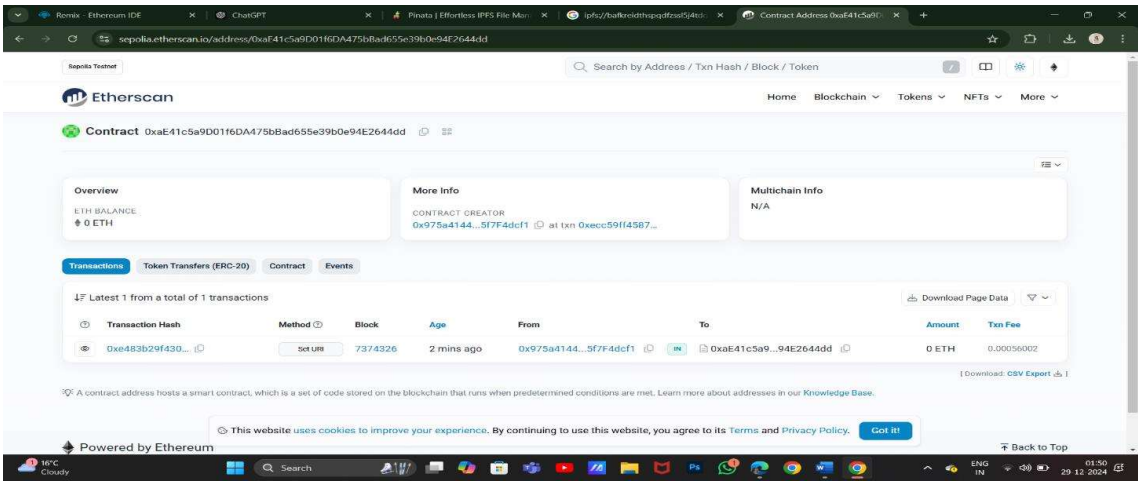
Q6)

With its default settings, the **Brave Browser** is able to provide its users with higher security levels as well as faster browsing since it blocks ads and trackers. In addition, it ensures the privacy of its users. The platform has a built-in feature that allows clients to generate wealth in the form of Basic Attention Tokens (BAT) through viewing targeted ads. It is also possible to use or reward the tokens to content creators and publishers within the platform thus ensuring a borderless and fair environment for digital advertisement.

On the other hand, the **BAT token** is an ERC-20 token and it is built on the platform of Ethereum. This token system aims at enabling advertisers to focus more on target audience attention measurement across them, the publishers and the consumers. The main aim of BAT is to enhance user experience by ensuring that digital advertisement is less annoying while ensuring users' attention is paid for. The use of BAT negates the challenges posed by target advertisement as all transactions in this ecosystem are uncontrolled but easily validated on the Blockchain.

Q7) Contract Address using ERC 1155 :- 0xaE41c5a9D01f6DA475bBad655e39b0e94E2644dd

Attached Screenshot of Sepolia Etherscan for Minted NFTs:-



Q8)

```
import secrets
import hashlib
def generate_ethereum_address():
    private_key = secrets.token_hex(32)     # Generate a random 32-byte private key
    public_key = hashlib.sha256(private_key.encode()).hexdigest()    # Derive public key using keccak256
    # Ethereum address is the last 20 bytes of the keccak256 hash of the public key
    ethereum_address = '0x' + hashlib.sha256(public_key.encode()).hexdigest()[-40:]
    print(f"Private Key: {private_key}")
    print(f"Ethereum Address: {ethereum_address}")

if __name__ == "__main__":
    generate_ethereum_address()
```

Q9)

```
import sha3
def generate_contract_address(sender_address, nonce):
    # Remove '0x' from sender address and convert to bytes
    sender_bytes = bytes.fromhex(sender_address[2:])
    # RLP encoding is simply concatenation of address and nonce
    rlp_input = sender_bytes + nonce.to_bytes((nonce.bit_length() + 7) // 8 or 1, 'big')
    # Keccak256 hash to derive the address
    keccak = sha3.keccak_256()
    keccak.update(rlp_input)
    # Take the last 20 bytes for the contract address
    contract_address = "0x" + keccak.hexdigest()[-40:]
    return contract_address

# Example usage with hardcoded inputs
```

```
sender = "0xabcdef1234567890abcdef1234567890abcdef12"  # Example sender address
nonce = 1  # Example nonce
print(f"Sender Address: {sender}")
print(f"Nonce: {nonce}")
print(f"Generated Contract Address: {generate_contract_address(sender, nonce)}")
```

Q10)

```
from web3 import Web3
# Initialize Web3 connection (connect to a test network)
w3 = Web3(Web3.HTTPProvider("https://mainnet.infura.io/v3/YOUR_INFURA_PROJECT_ID"))
def sign_transaction(private_key, transaction):
    # Sign the transaction using the sender's private key
    signed_txn = w3.eth.account.sign_transaction(transaction, private_key)
    return signed_txn
def verify_transaction(signed_txn):
    # Recover the sender's address from the signed transaction
    recovered_address = w3.eth.account.recover_transaction(signed_txn.rawTransaction)
    return recovered_address
# Example Usage
if __name__ == "__main__":
    # Hardcoded private key
    private_key = "0x4c0883a69102937d6231471b5dbb6204fe512961708279f6dfeee7d5af11d7ac"
    # Define the transaction details
    transaction = {
        'to': '0xRecipientAddressHere',  # Receiver's address
        'value': w3.to_wei(0.01, 'ether'),  # Amount to send (in wei)
        'gas': 21000,    # Gas limit
        'gasPrice': w3.to_wei(50, 'gwei'),  # Gas price
        'nonce': w3.eth.get_transaction_count(w3.eth.account.from_key(private_key).address),
        'chainId': 1    # Mainnet chain ID
    }
```

```
# Sign the transaction
signed_txn = sign_transaction(private_key, transaction)
print("Signed Transaction:", signed_txn.rawTransaction.hex())

# Verify the transaction
recovered_address = verify_transaction(signed_txn)
sender_address = w3.eth.account.from_key(private_key).address

print("Sender Address:", sender_address)
print("Recovered Address:", recovered_address)
print("Signature Verified:", recovered_address == sender_address)
```