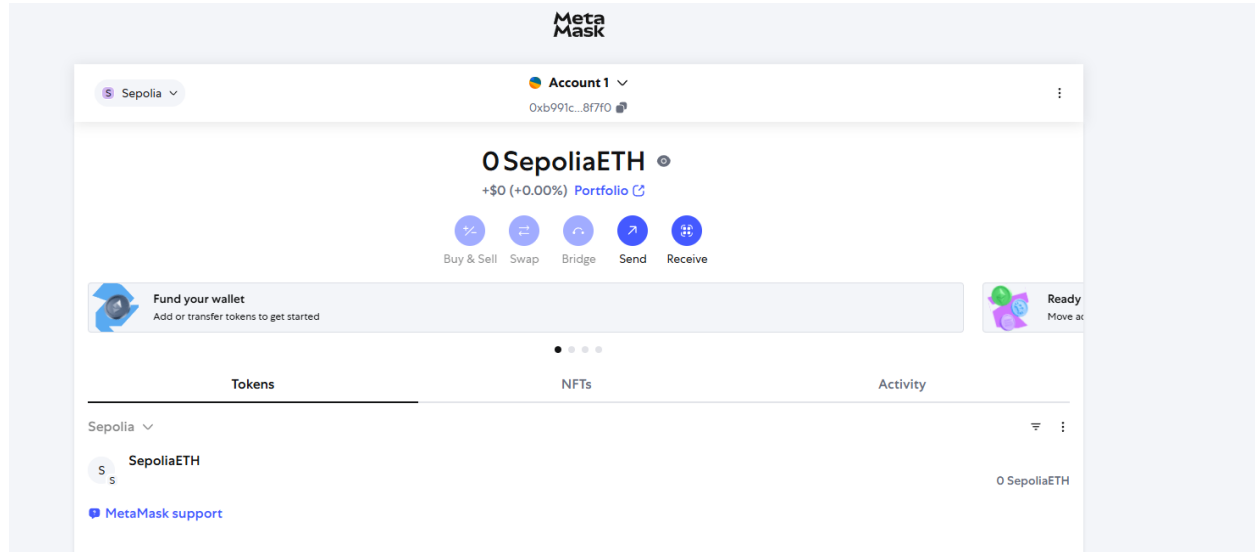


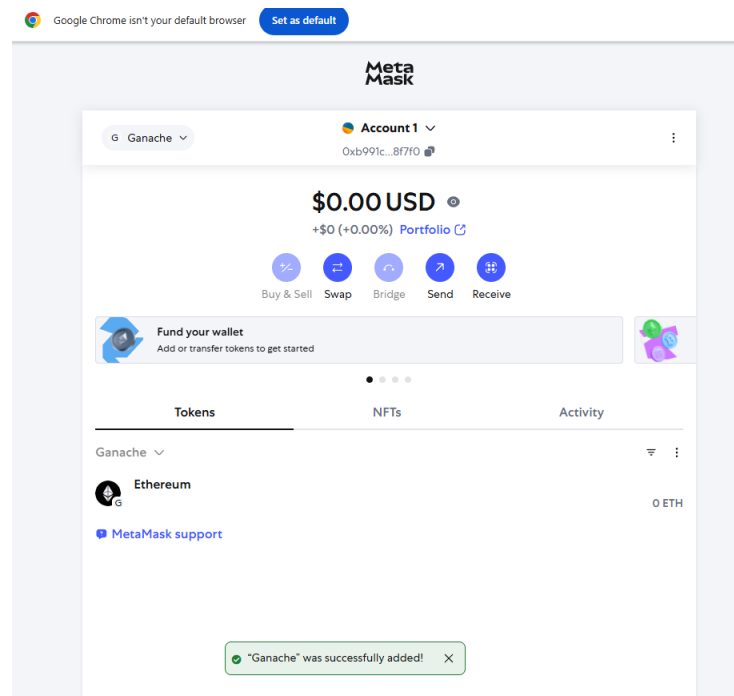
Pintero, Adrian Joriz G.

NW-301

Task 1:



Task 2:



Task 3 (Running Ganache)

```
C:\WINDOWS\system32\cmd. x + v
eth_chainId
eth_chainId
eth_accounts
eth_getBlockByNumber
eth_getBalance
eth_chainId
eth_getBalance
eth_chainId
eth_getBalance
eth_chainId
eth_getBalance
eth_blockNumber
eth_chainId
eth_getBalance
eth_chainId
eth_getBalance
eth_chainId
eth_getBlockByNumber
eth_getBalance
eth_chainId
eth_getBalance
eth_chainId
eth_getBalance
eth_chainId
```

Task 4 (Remix Deployment):

The screenshot displays the Remix IDE interface for deploying a smart contract. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is active, showing the 'Injected Provider - MetaMask' environment, the 'Sepolia (11155111) network', and an account with address '0xb99...8f7f0 (0 ETH)'. The 'GAS LIMIT' is set to 'Estimated Gas' and '3000000'. The 'VALUE' is '0 Wei'. The selected contract is 'ServiceFeeCalculator - contracts/Pint', with 'evm version: osaka' and 'Verify Contract on Explorers' checked. The 'Deploy & Verify' button is highlighted. Below, it shows 'At Address' and 'Transactions recorded 0'. The main editor on the right shows the Solidity code for the 'ServiceFeeCalculator' contract, which includes variables for 'firstName', 'middleName', and 'lastName', and a 'calculate' function with conditional logic for service fees. The bottom panel features an 'Explain contract' section with an 'AI copilot' toggle, a search bar for transaction hashes, and a list of accessible libraries including 'ethers.js'.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 contract ServiceFeeCalculator {
5
6     string public firstName = "Jose";
7     string public middleName = "Balsa";
8     string public lastName = "Rizal";
9
10    uint256 public serviceFee;
11    bytes32 public hashValue;
12
13    function calculate(uint8 serviceCode) public {
14        require(serviceCode == 1 || serviceCode == 2, "Invalid serviceCode");
15
16        // Calculate service fee
17        if (serviceCode == 1) {
18            // Consultation: 10,000 + 12% tax
19            serviceFee = 10000 + (10000 * 12 / 100);
20        } else {
21            // Documents: 2,000 + 10% service charge + 12% tax
22            uint256 base = 2000;
23            uint256 serviceCharge = base * 10 / 100;
24            uint256 subtotal = base + serviceCharge;
25            serviceFee = subtotal + (subtotal * 12 / 100);
26        }
27
28        // Build message components
29        bytes memory f = bytes(firstName);
30        bytes memory m = bytes(middleName);
```

Task 5 Deployment:

The screenshot displays the Remix IDE interface during the deployment of a smart contract. The main editor shows the Solidity code for the `ServiceFeeCalculator` contract, which includes variables for `firstName`, `middleName`, and `lastName`, and a `calculate` function that takes a `serviceCode` and returns a `serviceFee` and `hashValue`. The code is compiled, and the deployment process is shown in the left sidebar.

DEPLOY & RUN TRANSACTIONS

- ENVIRONMENT:** Injected Provider - MetaMask
- ACCOUNT:** Sepolia (11155111) network, 0xb99...8f7f0 (0 ETH)
- GAS LIMIT:** Estimated Gas, Custom 3000000
- VALUE:** 0 Wei
- CONTRACT:** ServiceFeeCalculator - contracts/Pint
- evm version:** osaka
- ☒ Verify Contract on Explorers
- Deploy & Verify** button
- At Address** button (Load contract from Address)

Transactions recorded 0 i >

ServiceFeeCalculator.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 contract ServiceFeeCalculator {
5
6     string public firstName = "Jose";
7     string public middleName = "Balsa";
8     string public lastName = "Rizal";
9
10    uint256 public serviceFee;
11    bytes32 public hashValue;
12
13    function calculate(uint8 serviceCode) public {
14        require(serviceCode == 1 || serviceCode == 2, "Invalid serviceCode");
15
16        // Calculate service fee
17        if (serviceCode == 1) {
18            // Consultation: 10,000 + 12% tax
19            serviceFee = 10000 + (10000 * 12 / 100);
20        } else {
21            // Documents: 2,000 + 10% service charge + 12% tax
22            uint256 base = 2000;
23            uint256 serviceCharge = base * 10 / 100;
24            uint256 subtotal = base + serviceCharge;
25            serviceFee = subtotal + (subtotal * 12 / 100);
26        }
27
28        // Build message components
29        bytes memory f = bytes(firstName);
30        bytes memory m = bytes(middleName);
```

Explain contract AI copilot

☐ Listen on all transactions Filter with transaction hash or address

The following libraries are accessible:

- `ethers.js`

Type the library name to see available commands.

Imported Account 1

999.999 ETH

+ETH 0.00 (+0.00%) Sign up for Newsletters

Buy Hello to Bitcoin

Trade Ethereum (ETH) directly on MetaMask

Jan 20, 2028

Contract deployment Confirmed

-0 ETH -ETH 0.00