

Blockchain Technologies

Journal

Submitted By

Pratik Jawalsingh Tiruwa

Roll No : 31031523035

MSc CS – Part II

Semester : III

**Department Of Computer
ScienceSomaiya Vidyavihar
University SK Somaiya College**

INDEX:

Practical No	Title
1.	A. Creating a simple blockchain to calculate sum of two number. B. Creating a simple blockchain to calculate factorial of a number.
2.	Creating a simple blockchain to check whether a number is happy or not.
3.	Creating a simple blockchain to check and validate a Kaprekar number.
4.	Create a simple blockchain to store only automorphic number also secure your automorphic number by DES Algorithm and validate the block before adding it to the blockchain.
5.	Create a simple blockchain to record deposit and withdrawal transactions.
6.	A. Create a smart contract for addition of two numbers. B. Write a simple auction contract where a user can bid on an item and the highest bidder wins.
7.	A. Write a smart contract to display factorial of a number. B. Write a smart contract to display nth term of Fibonacci series. C. Write a smart contract to check if the number is prime or not. D. Write a smart contract to deposit and withdraw money.
8.	A. Create a smart contract to calculate mean of n numbers. B. Create a smart contract to calculate median of n numbers. C. Create a smart contract to create a student portal and register a new student having the details Name, IdNo, Address, 3 subject marks, percentage and grade.
9.	Write a smart contract to create a voting application.
10.	A. Write a smart contract for Single Level Inheritance. B. Write a smart contract for Multi-Level Inheritance. C. Write a smart contract for Multiple Inheritance. D. Write a smart contract for Hierarchical Inheritance.

11.	Creating a simple DApp for performing basic mathematical operations on two numbers.
12.	Create a DApp to calculate factorial of a number. Create a DApp to implement transactions between two accounts. Create a DApp to implement elections.
13.	Storing and Retrieving files using IPFS.

Practical 1

A Creating a simple blockchain to calculate the sum of two number

Blockchain.js

```
const c = require('crypto');

class Block {
  constructor(i, t, n1, n2, ph='') {
    this.i = i;
    this.t = t;
    this.n1 = n1;
    this.n2 = n2;
    this.sum = n1 + n2;
    this.ph = ph;
    this.h = this.calhash();
  }

  calhash() {
    return c.createHash('sha256')
      .update(this.i + this.t + this.sum + this.ph)
      .digest('hex');
  }
}

class Blockchain {
  constructor() {
    this.chain = [this.createGBlock()];
  }

  createGBlock() {
    return new Block(0, '23/10/2024', 0, 0, '0');
  }

  getcBlock() {
    return this.chain[this.chain.length - 1];
  }

  addBlock(nb) {
    nb.ph = this.getcBlock().h;
    nb.h = nb.calhash();
    this.chain.push(nb);
  }
}
```

```
}  
  
module.exports.Block = Block;  
module.exports.Blockchain = Blockchain;
```

test.js

```
const {Block, Blockchain} = require('./blockchain');  
  
let mb = new Blockchain();  
  
console.log("Pratik Tiruwa")  
console.log("31031523035")  
  
console.log("First transaction");  
mb.addBlock(new Block(1, '23/10/2024', 23, 5));  
mb.addBlock(new Block(2, '23/10/2024', 2, 15));  
  
console.log(JSON.stringify(mb, null, 3));
```

Output :

```
PS C:\Users\Asus\Desktop\MSc Computer Science (Somaiya)\SEMESTER 3\BlockChain\Practicals> node '..\Practical 1\test.js'
Pratik Tiruwa
31031523035
First transaction
{
  "chain": [
    {
      "i": 0,
      "t": "01/08/2024",
      "n1": 0,
      "n2": 0,
      "sum": 0,
      "ph": "0",
      "h": "f0a3fc7bac56c13fb952d1f062337fe9d5c66953df566885bc504270d31f2aa8"
    },
    {
      "i": 1,
      "t": "01/08/2024",
      "n1": 23,
      "h": "2ed2dc7c51627b2e92b1b8c14722299ceeacde0b91f9af470a97bf304810c502"
    },
    {
      "i": 2,
      "t": "01/08/2024",
      "n1": 2,
      "n2": 15,
      "sum": 17,
      "ph": "2ed2dc7c51627b2e92b1b8c14722299ceeacde0b91f9af470a97bf304810c502",
      "h": "4e273c9274a8762f39ddf3949bea60af17b4f9781a2fe65fe59c917baeae29cc"
    }
  ]
}
```

B. Creating a simple blockchain to calculate factorial of a number.

Blockchain.js

```
const c = require('crypto');
const { blob } = require('stream/consumers');

class Block {
  constructor(i, t, n, ph = '') {
    this.i = i;
    this.t = t;
    this.n = n;
    this.fact = this.factorial();
    this.ph = ph;
    this.h = this.calHash();
  }

  factorial(n) {
    let ans = 1;
```

```
        if (n === 0)
            return 1;
        for (let i=2; i<=n; i++)
            ans = ans * i;
        return ans;
    }

    calHash() {
        return c.createHash('sha256')
            .update(this.i + this.t + this.sum + this.ph)
            .digest('hex');
    }
}

class Blockchain {
    constructor() {
        this.chain = [this.createGBlock()];
    }

    createGBlock() {
        return new Block(0, '24/10/2024', 0, 0, '0');
    }

    getcBlock() {
        return this.chain[this.chain.length - 1];
    }

    addBlock(nb) {
        nb.ph = this.getcBlock().h;
        nb.h = nb.calHash();
        this.chain.push(nb);
    }
}

module.exports.Block = Block;
module.exports.Blockchain = Blockchain;
```

test.js

```
const {b, bc, Blockchain, Block} = require('./blockchain');

let mb = new Blockchain();
console.log("Name : Pratik Tiruwa , Seat No : 31031523035");

mb.addBlock(new Block(1, '24/10/2024', 5));
mb.addBlock(new Block(2, '24/10/2024', 12));

console.log(JSON.stringify(mb, null, 3));
```

Output :

```
PS C:\Users\Asus\Desktop\Blockchain Practical> node '.\Practical 1\Factorial\test.js'
Name : Pratik Tiruwa , Seat No : 31031523035
{
  "chain": [
    {
      "i": 0,
      "t": "24/10/2024",
      "n": 0,
      "fact": 1,
      "ph": 0,
      "h": "f07ac694dcdbd2c432551a1e2cbb9b370d4a8a81262088241b54915d45cedc519"
    },
    {
      "i": 1,
      "t": "24/10/2024",
      "n": 5,
      "fact": 1,
      "ph": "f07ac694dcdbd2c432551a1e2cbb9b370d4a8a81262088241b54915d45cedc519",
      "h": "0bb6ec3eccb46c67fd18ca4293471dafaf257cf6f645f17406bded4b9b452237"
    },
    {
      "i": 2,
      "t": "24/10/2024",
      "n": 12,
      "fact": 1,
      "ph": "0bb6ec3eccb46c67fd18ca4293471dafaf257cf6f645f17406bded4b9b452237",
      "h": "f7b153df266dc19f56f0cf3eb46dc3102c3d10b5997b0e9f5eb8899deb811dae"
    }
  ]
}
```

Practical 2 : Creating a simple blockchain to check whether a number is happy or not.

```
const c = require('crypto');

class Block {
  constructor(i, t, n, ph = '') {
    this.i = i;
    this.t = t;
    this.n = n;
    this.ph = ph;
    this.f = this.calHappyNumber();
    this.h = this.calhash();
  }

  calHappyNumber() {
```



```
while(this.n > 9) {
  let sum = 0;
  while(this.n > 0)
  {
    let reminder = this.n % 10;
    this.n = Math.floor(this.n / 10);
    let sqr = reminder*reminder;
    sum += sqr;
  }

  this.n = sum;
}

if (this.n == 1)
  console.log("Happy Number")
else
{
  console.log("Not happy Number");
}
}

calhash() {
  return c.createHash('sha256')
    .update(`${this.i} ${this.t} ${this.n} ${this.f}
    ${this.ph}`)
    .digest('hex');
}

}

class Factchain {
  constructor() {
    this.chain = [this.genesisBlock()];
  }

  genesisBlock() {
    return new Block(0, new Date().toISOString(), 0);
  }

  getBlock() {
    return this.chain[this.chain.length - 1];
  }

  addBlock(nb) {
    nb.ph = this.getBlock().h;
    nb.h = nb.calhash();
    this.chain.push(nb);
  }
}
```

```
}  
  
module.exports.Block = Block;  
module.exports.Factchain = Factchain;
```

test.js

```
console.log("Pratik Tiruwa , 31031523035")  
  
const {Block, Factchain} = require('./Happynumber');  
  
let mb = new Factchain();  
  
mb.addBlock(new Block(1, new Date().toISOString(), 19));  
mb.addBlock(new Block(2, new Date().toISOString(), 7));  
  
console.log(JSON.stringify(mb, null, 1));
```

Output :

```
● PS C:\Users\Asus\Desktop\Blockchain Practical> node '.\Practical 2\Happy Number\test.js'  
Pratik Tiruwa , 31031523035  
Not happy Number  
Happy Number  
Not happy Number  
{  
  "chain": [  
    {  
      "i": 0,  
      "t": "2024-10-24T11:42:16.721Z",  
      "n": 0,  
      "ph": "",  
      "h": "df3fc495b709d996264d59cbbcb7207d0786592034b1b0916f738cc2fda6670"  
    },  
    {  
      "i": 1,  
      "t": "2024-10-24T11:42:16.724Z",  
      "n": 1,  
      "ph": "df3fc495b709d996264d59cbbcb7207d0786592034b1b0916f738cc2fda6670",  
      "h": "e87f4f3e4c4dc4960b7c9d22dbed7dcde1ffd329f67a7706704a2a2ceaf6fd39"  
    },  
    {  
      "i": 2,  
      "t": "2024-10-24T11:42:16.724Z",  
      "n": 7,  
      "ph": "e87f4f3e4c4dc4960b7c9d22dbed7dcde1ffd329f67a7706704a2a2ceaf6fd39",  
      "h": "757c57da6c19a53df2339e93fc23094bccce9d6c6ce59dc0be85a0104460df069"  
    }  
  ]  
}  
● PS C:\Users\Asus\Desktop\Blockchain Practical> □
```

Practical 3 : Creating a simple blockchain to check and validate a Kaprekar number.

Kaprekar.js

```
const c = require('crypto');

class Block {
  constructor(i, t, n1, ph='') {
    this.i = i;
    this.t = t;
    this.n1 = n1;
    this.ph = ph;
    this.h = this.calhash();
  }

  checkaps() {
    var cnt = 0, x = this.n1, sq = x * x;
    var y = x, r, f, b, sum, sq1, rem;

    while (x != 0) {
      rem = x % 10;
      cnt++;
      x = parseInt(x / 10);
    }

    r = Math.pow(10, cnt);
    r = parseInt(r);
    f = parseInt(sq / r);
    b = parseInt(sq % r);
    sum = f + b;

    if (sum == y) {
      return true;
    }
    else {
      return false;
    }
  }

  calhash() {
    return
    c.createHash('sha256').update(this.i + this.t + this.n1 + this.ph)
    .digest('hex');
  }
}

class Blockchain {
```

```
constructor() {
    this.chain = [this.create_GBlock()];
}

create_GBlock() {
    return new Block(0, '11-10-24', 0, '0');
}

Getcurr_block() {
    return this.chain[this.chain.length - 1];
}

Add_Block(nb) {
    if (nb.checkaps() == true) {
        nb.ph = this.Getcurr_block().h;
        nb.h = nb.calhash();
        this.chain.push(nb);
    }
}

validate() {
    for (let i=1; i < this.chain.length; i++) {
        let cb = this.chain[i];
        let pb = this.chain[i - 1];

        if (cb.h != cb.calhash()) {
            return false
        }
        if (pb.h != cb.ph){
            return false
        }
    }

    return true
}
}

module.exports.Block = Block;
module.exports.Blockchain = Blockchain;
```

test.js

```
// Import the Block and Blockchain classes from the kaprekar.js file
const {Block, Blockchain} = require('./kaprekar');

// Create a new blockchain instance
const blockchain = new Blockchain();
```

```
// Define an array of numbers to test
const testNumbers = [45, 13, 297, 10];

// Iterate over each number, create a Block, and add it to the blockchain
testNumbers.forEach((num, index) => {
    const block = new Block(index + 1, new Date().toString(), num);
    blockchain.Add_Block(block);
    console.log(`Block ${index + 1} with number ${num} ${block.checkaps() ? 'is
a Kaprekar number.' : 'is not a Kaprekar number.'}`);
});

// Validate the entire blockchain
const isChainValid = blockchain.validate();
console.log(`\n Is the blockchain valid? ${isChainValid ? 'Yes' : 'No'}`);

// Optionally, display the blocks in the blockchain
console.log("\nBlockchain:");
blockchain.chain.forEach((block, index) => {
    console.log(`Block ${index}: `, block);
});

console.log('Name: Pratik Tiruwa, Roll no : 35, Date: 14-8-24');
```

Output :

```
PS C:\Users\Asus\Desktop\Blockchain Practical> node '.\Practical 3\test.js'
Block 1 with number 45
  is a Kaprekar number.
Block 2 with number 13
  is not a Kaprekar number.
Block 3 with number 297
  is a Kaprekar number.
Block 4 with number 10
  is not a Kaprekar number.

Is the blockchain valid? Yes

Blockchain:
Block 0: Block {
  i: 0,
  t: '2024-10-24T11:33:28.493Z',
  n1: 0,
  ph: '0',
  h: 'e4a9be934cbfb8033c3fb4a5f67a0d67d30cb913f158bbda8cb446d92f9a53ca'
}
Block 1: Block {
  i: 1,
  t: '2024-10-24T11:33:28.496Z',
  n1: 45,
  ph: 'e4a9be934cbfb8033c3fb4a5f67a0d67d30cb913f158bbda8cb446d92f9a53ca',
  h: '2ed034d770d6b592ef15569b6ef915aff019b6299cb7d74af7a3b80015dc2492'
}
Block 2: Block {
  i: 3,
  t: '2024-10-24T11:33:28.506Z',
  n1: 297,
  ph: '2ed034d770d6b592ef15569b6ef915aff019b6299cb7d74af7a3b80015dc2492',
  h: '5398eb45fe8c4a2457025d4e552d9c5f07079b7faa7db335af0a2f83152516cd'
}
Name:Pratik Tiruwa, Roll no: 31031523035
```

Practical 4 : Create a simple blockchain to store only automorphic number also secure your automorphic number by DES Algorithm and validate the block before adding it to the blockchain.

Blockchain.js

```
const c = require('crypto')

class Block {
  constructor(i, t, n, ph = '') {
    this.i = i;
    this.t = t;
    this.n = this.enc(n);
    this.automorphic = this.morph();
    this.ph = ph;
    this.h = this.calHash();
  }

  morph() {
    let n1 = this.dec(this.n);
    let squared = n1 * n1;

    let numStr = n1.toString();
    let squaredStr = squared.toString();

    let result = squaredStr.endsWith(numStr);

    if (result == true) {
      return "Automorphic Number"
    }
    else {
      return "Not an Automorphic Number"
    }
  }

  enc(text) {
    const key = c.scryptSync('password', 'salt', 32);
    const iv = c.randomBytes(16);
    const cipher = c.createCipheriv('aes-256-cbc', key, iv);
    const encrypted = cipher.update(text.toString(), 'utf8', 'hex') +
    cipher.final('hex');
    return iv.toString('hex') + ':' + encrypted;
  }

  dec(encryptedText) {
    const key = c.scryptSync('password', 'salt', 32);
    const [ivHex, encrypted] = encryptedText.split(':');
```

```
const iv = Buffer.from(ivHex, 'hex');
const decipher = c.createDecipheriv('aes-256-cbc', key, iv);
const decrypted = decipher.update(encrypted, 'hex', 'utf8') +
decipher.final('utf8');
return parseInt(decrypted, 10);
}

calHash() {
    return c.createHash('sha256').update(this.i + this.t + this.automorphic
+
this.ph).digest('hex');
}
}

class Blockchain {
    constructor() {
        this.chain = [this.createGBlock()];
    }

    createGBlock() {
        return new Block(0, new Date(), 0, '0');
    }

    getCBlock() {
        return this.chain[this.chain.length - 1];
    }

    addBlock(nb) {
        if (nb.morph() == "Automorphic Number") {
            nb.ph = this.getCBlock().h;
            nb.h = nb.calHash();
            this.chain.push(nb);
        }
    }

    validate() {
        for (let i = 1; i < this.chain.length; i++) {
            let cb = this.chain[i]
            let pb = this.chain[i - 1]

            if (cb.h != cb.calHash()) {
                return false;
            }

            if (pb.h != cb.ph) {
                return false;
            }
        }

        return true;
    }
}
```

```
}  
  
module.exports.Block = Block;  
module.exports.Blockchain = Blockchain;
```

test.js

```
const { Blockchain, Block } = require('./blockchain')  
  
let mb = new Blockchain();  
  
console.log("Developed by Pratik Tiruwa - 31031523035")  
  
mb.addBlock(new Block(1, new Date(), 5))  
mb.addBlock(new Block(2, new Date(), 7))  
mb.addBlock(new Block(3, new Date(), 6))  
  
console.log(mb.validate());  
console.log(JSON.stringify(mb, null, 3))
```

Output:

```
● PS C:\Users\Asus\Desktop\Blockchain Practical> node .\Practical_4\test.js  
Developed by Pratik Tiruwa - 31031523035  
true  
{  
  "chain": [  
    {  
      "i": 0,  
      "t": "2024-10-24T16:58:01.529Z",  
      "n": "8e21e440e2a14c64499d6a0af839cc19:8a2ae757d012ba7385c1954fa0fd441f",  
      "automorphic": "Automorphic Number",  
      "ph": "0",  
      "h": "b798b6e9253abe404b789c9bed04c5f750133757b3e45bf5616b3437772ff88b"  
    },  
    {  
      "i": 1,  
      "t": "2024-10-24T16:58:01.725Z",  
      "n": "e00a5225050d2d9f0ce951aef7a41584:ce7aa03ba39a5b7b511bc5363c2434c1",  
      "automorphic": "Automorphic Number",  
      "ph": "b798b6e9253abe404b789c9bed04c5f750133757b3e45bf5616b3437772ff88b",  
      "h": "3ad881beb51bd83b7818c8867cfec01c0a7fe993de37e9593ebcf2e342294161"  
    },  
    {  
      "i": 3,  
      "t": "2024-10-24T16:58:02.249Z",  
      "n": "1fa1bc0a65d8b85522d06fb2b34ef695:3753915cf3062ba39c6eaeaf401c6fa49",  
      "automorphic": "Automorphic Number",  
      "ph": "3ad881beb51bd83b7818c8867cfec01c0a7fe993de37e9593ebcf2e342294161",  
      "h": "aec8ddf478420c983b79e4c80cbe36af126a48f0e70c241d6b1d769b9fa18238"  
    }  
  ]  
}
```


Practical 5: Create a simple blockchain to record deposit and withdrawal transactions.

blockchain.js

```
const c = require('crypto')

class Block {
  constructor(i, t, op, m, b = 0, ph = '') {
    this.i = i;
    this.t = t;
    this.op = op;
    this.m = m;
    this.b = b;
    this.ph = ph;
    this.h = this.calHash();
  }

  calHash() {
    return c.createHash('sha256').update(this.i + this.t + this.op + this.b
+
this.ph).digest('hex');
  }
}

class Blockchain {
  constructor() {
    this.chain = [this.createGBlock()];
  }

  createGBlock() {
    return new Block(0, new Date(), 'D', 1000, 0, '0');
  }

  getCBlock() {
    return this.chain[this.chain.length - 1];
  }

  addBlock(nb) {
    nb.ph = this.getCBlock().h;

    if (nb.op == "D") {
      nb.b = this.getCBlock().b + nb.m;
    }
    else if (nb.op == "W") {
      if (this.getCBlock().b > nb.m) {
        nb.b = this.getCBlock().b - nb.m;
      }
      else {
        nb.b = this.getCBlock().b;
      }
    }
  }
}
```

```
    }  
  }  
  else {  
    console.log("Not a Valid Operation.")  
  }  
  
  nb.h = nb.calHash();  
  this.chain.push(nb);  
}  
  
validate() {  
  for (let i = 1; i < this.chain.length; i++) {  
    let cb = this.chain[i]  
    let pb = this.chain[i - 1]  
  
    if (cb.h !== cb.calHash()) {  
      return false;  
    }  
  
    if (pb.h !== cb.ph) {  
      return false;  
    }  
  }  
  
  return true;  
}  
}  
  
module.exports.Block = Block;  
module.exports.Blockchain = Blockchain;
```

test.js

```
const { resolve } = require('path');  
const { Blockchain, Block } = require('./Blockchain');  
  
let mb;  
console.log("Name : Pratik Tiruwa, Seat No : 31031523035");  
  
const readline = require('readline').createInterface({  
  input: process.stdin,  
  output: process.stdout  
})  
  
async function main() {  
  mb = new Blockchain();  
  let i = 1;  
  while (true) {  
    const DW = await new Promise(resolve => {
```

```
        readline.question('Type D - Deposit, W - Withdraw or Any Other  
Character - Exit: ', resolve);  
  
    });  
  
    if (DW.toUpperCase() !== 'D' && DW.toUpperCase() !== 'W') {  
        console.log('Existing...');  
        break;  
    }  
  
    const m = await new Promise(resolve => {  
        readline.question('Enter Amount: ', resolve);  
    });  
  
    mb.addBlock(new Block(i, new Date(), DW, parseInt(m)));  
    console.log(JSON.stringify(mb, null, 3));  
    i++;  
}  
  
readline.close();  
}  
  
main();
```

Output:

```
● PS C:\Users\Asus\Desktop\Blockchain Practical> node .\Practical_5\test.js  
Name : Pratik Tiruwa, Seat No : 31031523035  
Type D - Deposit, W - Withdraw or Any Other Character - Exit: D  
Enter Amount: 1500  
{  
  "chain": [  
    {  
      "i": 0,  
      "t": "2024-10-24T17:13:22.891Z",  
      "op": "D",  
      "m": 1000,  
      "b": 1000,  
      "ph": "0",  
      "h": "d8903280d9c73f036e36416f84c31205b5069fba42e222953d410538cfa9d77b"  
    },  
    {  
      "i": 1,  
      "t": "2024-10-24T17:13:28.162Z",  
      "op": "D",  
      "m": 1500,  
      "b": 2500,  
      "ph": "d8903280d9c73f036e36416f84c31205b5069fba42e222953d410538cfa9d77b",  
      "h": "f62364804f356b3f8ce7d5dbca2a185dd66278e9cdab4550f1f47e9f7ec222dc"  
    }  
  ]  
}
```

```
Type D - Deposit, W - Withdraw or Any Other Character - Exit: W
Enter Amount: 289
{
  "chain": [
    {
      "i": 0,
      "t": "2024-10-24T17:16:40.392Z",
      "op": "D",
      "m": 1000,
      "b": 1000,
      "ph": "0",
      "h": "f6ca8b969df432e9989f567d6b154106324c4fbb880bd7310b5787ac3ecd2295"
    },
    {
      "i": 1,
      "t": "2024-10-24T17:16:43.949Z",
      "op": "D",
      "m": 1500,
      "b": 2500,
      "ph": "f6ca8b969df432e9989f567d6b154106324c4fbb880bd7310b5787ac3ecd2295",
      "h": "fdedf4e090b8546ec6ed3f74a3cdc56ae5bffd2e0a2fefe6e388a16c5da03751"
    },
    {
      "i": 2,
      "t": "2024-10-24T17:16:50.183Z",
      "op": "W",
      "m": 289,
      "b": 2211,
      "ph": "fdedf4e090b8546ec6ed3f74a3cdc56ae5bffd2e0a2fefe6e388a16c5da03751",
      "h": "4c490d31b605d0d4a00e7a90ad9ace132c0f5c07d3e781bd962d1c85a5c438f1"
    }
  ]
}
Type D - Deposit, W - Withdraw or Any Other Character - Exit: 
```

Practical 6

A : Create a Smart Contract for addition of two numbers.

```
// SPDX-License-Identifier: MIT

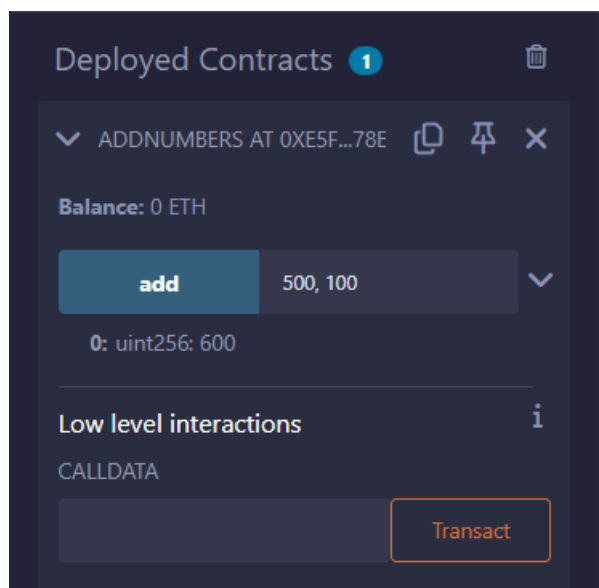
pragma solidity ^0.8.0;

contract addNumbers {

    function add(uint256 a, uint256 b) public pure returns (uint256) {

        return a + b;
    }
}
```

Output:



B. Write a simple auction contract when a susr bid on an item and the highest bidder wins.

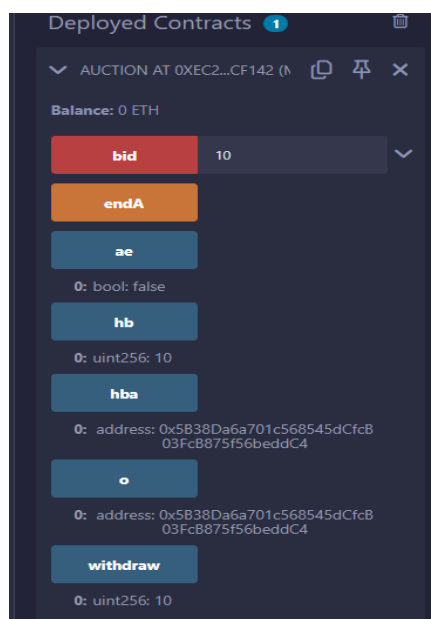
```
// SPDX-License-Identifer: MIT

pragma solidity ^0.8.0;

contract auction {
```

```
address public o;  
address public hba;  
uint public hb;  
bool public ae;  
  
constructor() {  
    o = msg.sender;  
    hb = 0;  
    ae = false;  
}  
  
function bid(uint256 a) public payable {  
    require(!ae, "Auction has ended!");  
    require(a > hb, "Bid must be greater than current value.");  
    hb = a;  
    hba = msg.sender;  
}  
  
function endA() public {  
    require(msg.sender == o, "Only owner can end");  
    ae = true;  
}  
  
function withdraw() public view returns (uint256) {  
    require(msg.sender == hba, "Only higehest bidder can withdraw");  
    // payable(hba).transfer(hb);  
    return hb;  
}  
}
```

Output:



Practical 7

A. Write a smart contract program to print factorial of a number.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract factorial {
    function fact(uint256 a) public pure returns (uint256) {
        uint256 ans = 1;
        for (uint256 i = 2; i <= a; i++) {
            ans = ans*i;
        }

        return ans;
    }
}
```

Output:


[illegible]

B. Write a program to create Smart contract to find nth term of Fibonacci series

```
//SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract fibonacci {

    function fib(uint256 a) public pure returns (uint256) {  infinite gas

        uint256 n1 = 0;

        uint256 n2 = 1;

        uint n3;

        for (uint256 i = 3; i <= a; i++) {

            n3 = n1 + n2;
            n1 = n2;

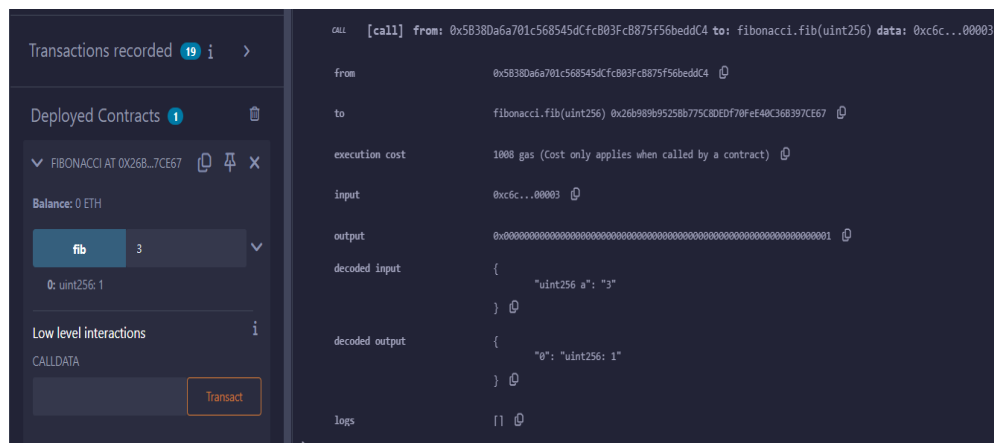
        }

        return n3;

    }

}
```

Output:



The screenshot displays a web interface for a deployed contract named "FIBONACCI AT 0x268...7CE67". The contract's balance is 0 ETH. A "fib" function is shown with an input of 3, resulting in an output of 1. The "Low level interactions" section shows a "Transact" button. The right panel displays the execution details for a call to the "fib" function, showing the input "3" and the output "1".


Field	Value
CALL	[call] from: 0x58380a6a701c568545dcfc803fc8875f56bedd4 to: fibonacci.fib(uint256) data: 0xc6c...00003
from	0x58380a6a701c568545dcfc803fc8875f56bedd4
to	fibonacci.fib(uint256) 0x26b989695258b775c80edf70fe40c368397ce67
execution cost	1808 gas (Cost only applies when called by a contract)
input	0xc6c...00003
output	0x0001
decoded input	{ "uint256 a": "3" }
decoded output	{ "0": "uint256: 1" }
logs	[]

C. Write a smart contract to check if the numebr is prime or not.


```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract primeno {

    function prime(uint256 a) public pure returns (bool) {  infinite gas

        for (uint256 i = 2; i < a; i++) {

            if (a % i == 0) {

                return false;

            }


        }

        return true;

    }

}
```

Output:



D. Write a smart contract to deposit and withdraw money

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract Bank{
    address public s;
    uint256 public bal;

    constructor(){
```

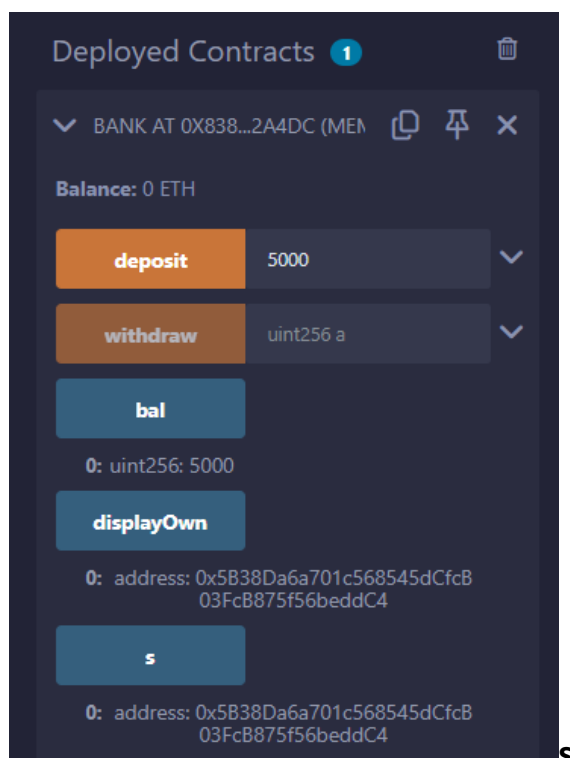
```
s = msg.sender;
bal = 0;
}

function deposit(uint256 a) public {
    bal += a;
    s = msg.sender;
}

function withdraw(uint256 a) public{
    if(bal >= a){
        bal -= a;
    }
    s = msg.sender;
}

function displayOwn() public view returns(address){
    return s;
}
}
```

Output:



Name : Pratik Tiruwa
Seat No : 31031523035

Deployed Contracts 1

▼ BANK AT 0X838...2A4DC (MEN

Balance: 0 ETH

deposit

5000

▼

withdraw

2000

▼

bal

0: uint256: 3000

displayOwn

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

s

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

Somaiya Vidyavihar University

Practical 8

A: Create a smart contract to calculate mean of n numbers.

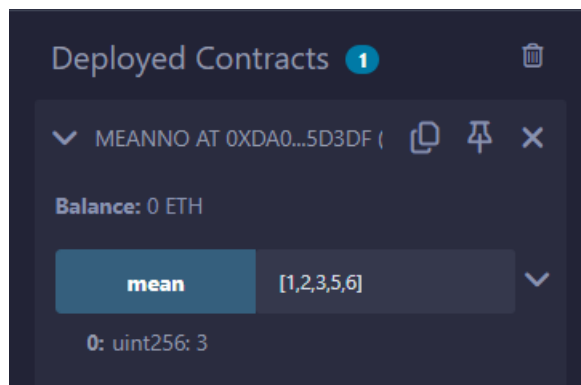
```
// SPDX-License-Identifier: MIT

contract meanNo {
    function mean(uint256[] memory arr) public pure returns(uint256){
        uint res = 0;

        for(uint i = 0; i < arr.length; i++){
            res = res + arr[i];
        }

        return res = res / arr.length;
    }
}
```

Output :



B. Create a smart contract to calculate median of n numbers.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract medianNo {

    function median(uint256[] memory arr) public pure returns (uint256) {
        uint256 n = arr.length;

        for (uint256 i=0; i < n - 1; i++) {
            for (uint256 j=0; j < n - i - 1; j++) {
```

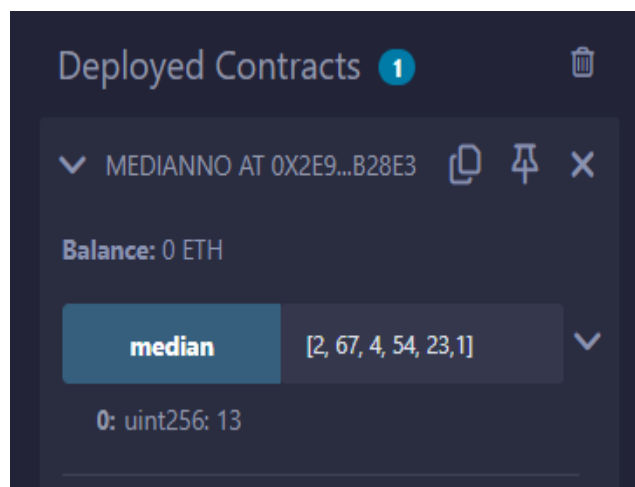
```
        if (arr[j] > arr[j + 1]) {
            (arr[j], arr[j + 1]) = (arr[j + 1], arr[j]);
        }
    }
}

uint256 med;

if(n % 2 == 0) {
    med = (arr[n / 2 - 1] + arr[n / 2]) / 2;
} else {
    med = arr[n / 2];
}

return med;
}
```

Output:



C. Create a student portal to register studnets and record their marks using their student ID. The portal should calculate the grades for three subjects and display the result.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract studentMarks {

    struct std{
        uint sid;
```

```
    string sname;
    address add;
    uint[3] marks;
    uint BT;
    uint ML;
    uint BD;
    string grade;
}

mapping (uint256 => std) public s1;

function Resgistration(uint i, string memory n) public {
    s1[i].sname = n;
    s1[i].add = msg.sender;
    s1[i].sid = i;
}

function SMarks(uint id, uint b, uint m, uint d) public {
    s1[id].BT = b;
    s1[id].ML = m;
    s1[id].BD = d;
    s1[id].grade = CSM(id);
}

function CSM(uint id) internal view returns (string memory) {
    std storage student = s1[id];
    uint256 totalMarks = student.BT + student.ML + student.BD;
    uint256 percentage = (totalMarks * 100) / 300;

    if (percentage >= 90) {
        return "A+";
    } else if (percentage >= 80) {
        return "A";
    } else if (percentage >= 70) {
        return "B+";
    } else if (percentage >= 60) {
        return "B";
    } else if (percentage >= 50) {
        return "C";
    } else if (percentage >= 40) {
        return "D";
    }
    else {
        return "F";
    }
}

function display (uint256 id) external view returns (std memory) {
    return s1[id];
}
```

```
}  
}
```

Output:

The screenshot displays a web application for a deployed contract named 'STUDENTMARKS' at address 0x4A9...E. The interface includes a sidebar with contract details and a main area showing transaction logs.

Contract Details:

- Contract: STUDENTMARKS AT 0x4A9...E
- Balance: 0 ETH
- Buttons: Registration, SMarks, display
- Inputs: uint256 i, string n (for Registration); uint256 id, uint256 b, uint256 s (for SMarks); 1 (for display)
- Output: 0: tuple(uint256,string,address,uint256[3],uint256,uint256,uint256,string): 1,Pratik,0x5B380Da6a701c56854dCfcB03FcB875f56beddC4,0.0,0.75,85,90,A
- Storage: s1: 1
- Output: 0: uint256: sid 1; 1: string: sname Pratik; 2: address: add 0x5B380Da6a701c56854dCfcB03FcB875f56beddC4; 3: uint256: BT 75; 4: uint256: ML 85; 5: uint256: BD 90; 6: string: grade A

Transaction Logs:

- creation of studentMarks pending...
- [vm] from: 0x5B3...eddC4 to: studentMarks.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x5e8...a524b
- transact to studentMarks.Registration pending ...
- [vm] from: 0x5B3...eddC4 to: studentMarks.Registration(uint256,string) 0x4a9...E31bf value: 0 wei data: 0xbb8...0000 logs: 0 hash: 0xd29...7ffa9
- transact to studentMarks.SMarks pending ...
- [vm] from: 0x5B3...eddC4 to: studentMarks.SMarks(uint256,uint256,uint256,uint256) 0x4a9...E31bf value: 0 wei data: 0xe65...0005a logs: 0 hash: 0x41e...bdb68
- call to studentMarks.display
- CALL [call] from: 0x5B380Da6a701c56854dCfcB03FcB875f56beddC4 to: studentMarks.display(uint256) data: 0xc2d...00001
- call to studentMarks.s1
- CALL [call] from: 0x5B380Da6a701c56854dCfcB03FcB875f56beddC4 to: studentMarks.s1(uint256) data: 0x448...00001

Practical 9 : Write a smart contract to create a voting application.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract voting {

    mapping(string => uint256) public c;
    mapping(address => bool) public voters;
    string[] public cn;

    constructor(string[] memory candN) {
        cn = candN;
    }

    function vote(string memory caNm) public {
        require(!voters[msg.sender], "Already Voting Done.");
        bool ce = false;

        for (uint256 i=0; i < cn.length; i++) {
            if (keccak256(bytes(caNm)) == keccak256(bytes(cn[i]))) {
                ce = true;
                break;
            }
        }

        require(ce, "Candidate does not exist.");
        c[caNm]++;
        voters[msg.sender] = true;
    }

    function getVoterC(string memory canM) public view returns (uint256) {
        return c[canM];
    }

    function getWinner() public view returns (string memory) {
        string memory winner;

        uint256 temp = 0;
        for (uint256 j=0; j < cn.length; j++) {
            if (getVoterC(cn[j]) > temp) {
                temp = getVoterC(cn[j]);
                winner = cn[j];
            }
        }

        return winner;
    }
}
```

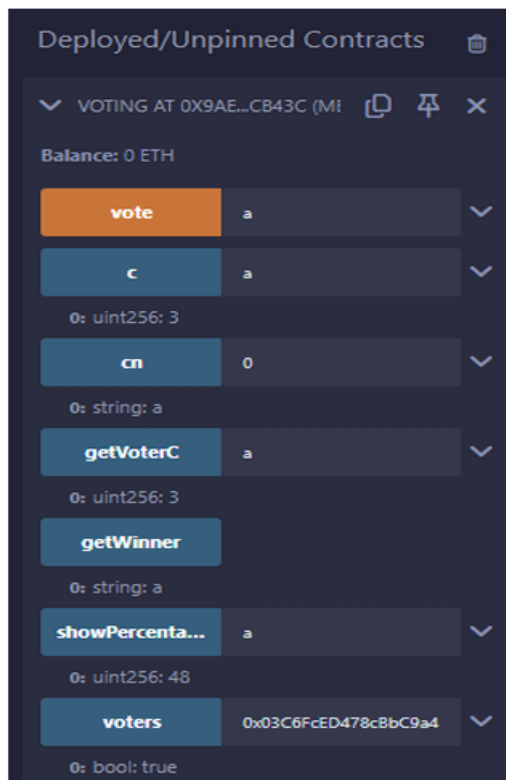


```
function showPercentage(string memory canM) public view returns (uint256) {
    uint256 total;

    for (uint256 i=0; i < cn.length; i++) {
        total = total + getVoterC(cn[i]);
    }

    uint256 per = getVoterC(canM) * (100 / total);
    return per;
}
```

Output:



Practical 10

A. Write a smart contract for Single Level Inheritance.

```
//SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract singer{
    string n;
    string[2] so;
    function setN(string memory a, string[2] memory arr) public {
        n = a;
        so = arr;
    }
}

contract song is singer{
    function getVal() public view returns (string memory, string[2] memory){
        return (n, so);
    }
}

contract test{
    song s = new song();
    function tInherit() public returns(string memory, string[2] memory){
        s.setN("Iqlipse Nova", ["Khwab", "Sajke"]);
        return s.getVal();
    }
}
```

Output:

Field	Value
execution cost	11769 gas
input	0x995...0000
output	0x
decoded input	{ "string a": "Iqlipse Nova", "string[2] arr": ["Khwab", "Sajke"] }
decoded output	{}

B. Write a smart contract for Multi-Level Inheritance.

```
//SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.8.0;

contract college {
    string internal cname;
    string internal pname;

    function setCollege(string memory cn, string memory pn) public {
        cname = cn;
        pname = pn;
    }
}

contract student is college {
    string internal sname;
    uint internal rollno;

    function setStudent(string memory sn, uint rn) public {
        sname = sn;
        rollno = rn;
    }
}

contract exam is student {
    uint8[5] marks;

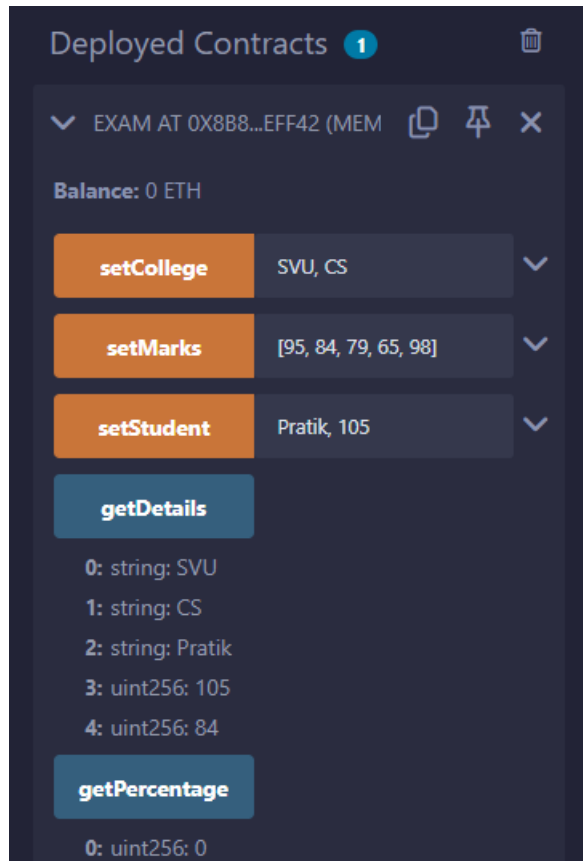
    function setMarks(uint8[5] memory m) public {
        marks = m;
    }

    function getPercentage() public view returns(uint) {

    }

    function getDetails() public view returns(string memory, string memory,
string
memory, uint, uint) {
        uint total = 0;
        for(uint i = 0; i < 5; i++) {
            total += marks[i];
        }
        uint per = total/5;
        return (cname, pname, sname, rollno, per);
    }
}
```

Output:



C. Write a smart contract for Multiple Inheritance.

```
//SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract employee{
    string n;
    uint mid;
    uint sal;
    function setE(string memory a, uint b, uint c) public {
        n = a;
        mid = b;
        sal = c;
    }
}

contract department{
    string dep;
    function setD(string memory a) public {
        dep = a;
    }
}
```

```
contract salary is employee, department{
    uint HRA;
    function calHRA() public returns(uint){
        if (sal >= 15000){
            HRA = 5000;
        }
        else if(sal >= 25000){
            HRA = 10000;
        }
        else{
            HRA = 2000;
        }
        return HRA;
    }

    function getVal() public view returns (string memory, uint, uint, string
memory, uint){
        return (n, mid, sal, dep, HRA);
    }
}

contract test{
    salary s = new salary();
    function tInherit() public returns(string memory, uint, uint, string memory,
uint){
        s.setE("Pratik Tiruwa", 101, 15000);
        s.setD("CS");
        s.calHRA();
        return s.getVal();
    }
}
```

Output:

SALARY AT 0X9DA...DEF66 (M)

Balance: 0 ETH

calHRA

setD CS

setE string a, uint256 b, uint2

getVal

0: string: Pratik Tiruwa
1: uint256: 101
2: uint256: 15000
3: string: CS
4: uint256: 0

output

0x00
00
00
697275776100
00

decoded input {}

decoded output {
 "0": "string: Pratik Tiruwa",
 "1": "uint256: 101",
 "2": "uint256: 15000",
 "3": "string: CS",
 "4": "uint256: 0"
}

logs []

raw logs []

D. Write a smart contract for Hierarchical Inheritance.

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract animal {
    uint legs;
    string color;

    function setA(uint a, string memory b) public {
        legs = a;
        color = b;
    }
}

contract dog is animal {
    string name;
    string species;

    function setVal(string memory a, string memory b) public {
        name = a;
        species = b;
    }

    function getVal() public view returns (uint, string memory, string memory, string memory) {
        return (legs, color, name, species);
    }
}

contract cat is animal {
    string name;
    string species;

    function setVal(string memory a, string memory b) public {
        name = a;
        species = b;
    }

    function getVal() public view returns (uint, string memory, string memory, string memory) {
        return (legs, color, name, species);
    }
}

contract test {
    dog d = new dog();
    cat c = new cat();
}
```

```
function dInherit() public returns (uint, string memory, string memory,
string memory) {
    d.setA(4, "Brown");
    d.setVal("Tom", "Labrador");
    return d.getVal();
}

function cInherit() public returns (uint, string memory, string memory,
string memory) {
    c.setA(4, "White");
    c.setVal("Goldie", "Indie");

    return c.getVal();
}
}
```

Output:

cInherit

Balance: 0 ETH

setA uint256 a, string b

setVal string a, string b

getVal

0: uint256: 4
1: string: White
2: string: Goldie
3: string: Indie

Low level interactions

CALLDATA

Transact

input

output

decoded input

decoded output

logs

raw logs

dInherit

Balance: 0 ETH

setA 4, "Brown"

setVal "Tom", "Labrador"

getVal

0: uint256: 4
1: string: Brown
2: string: Tom
3: string: Labrador

Low level interactions

CALLDATA

Transact

input

output

decoded input

decoded output

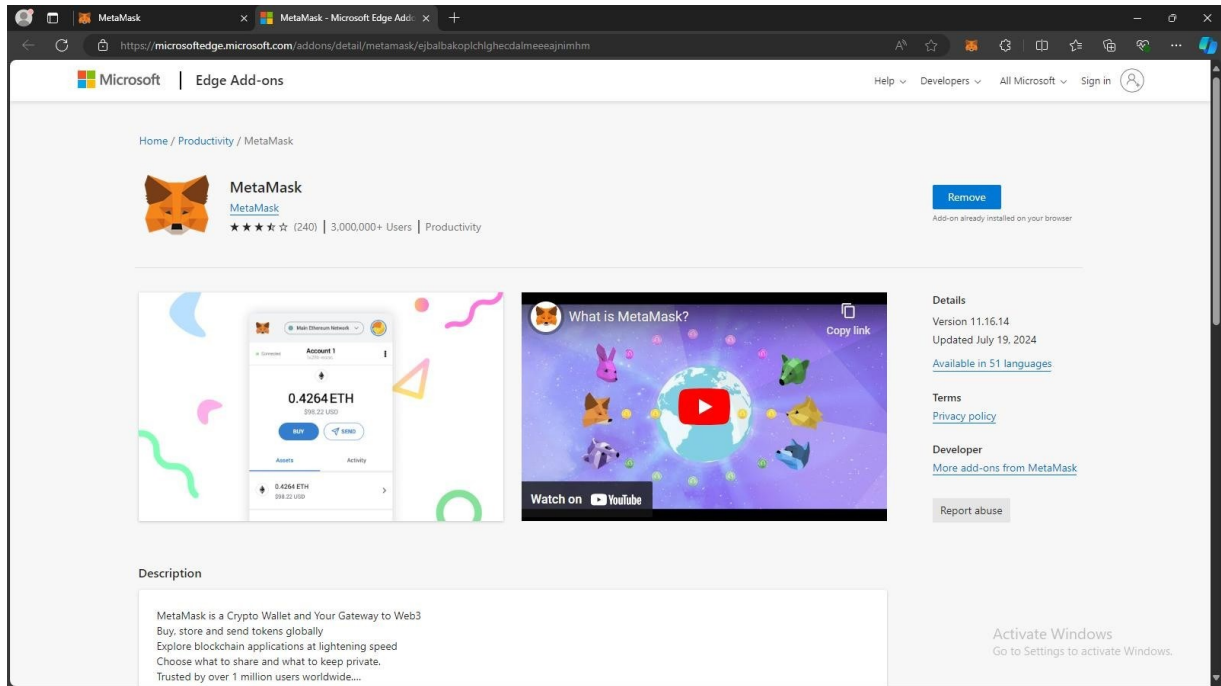
logs

raw logs

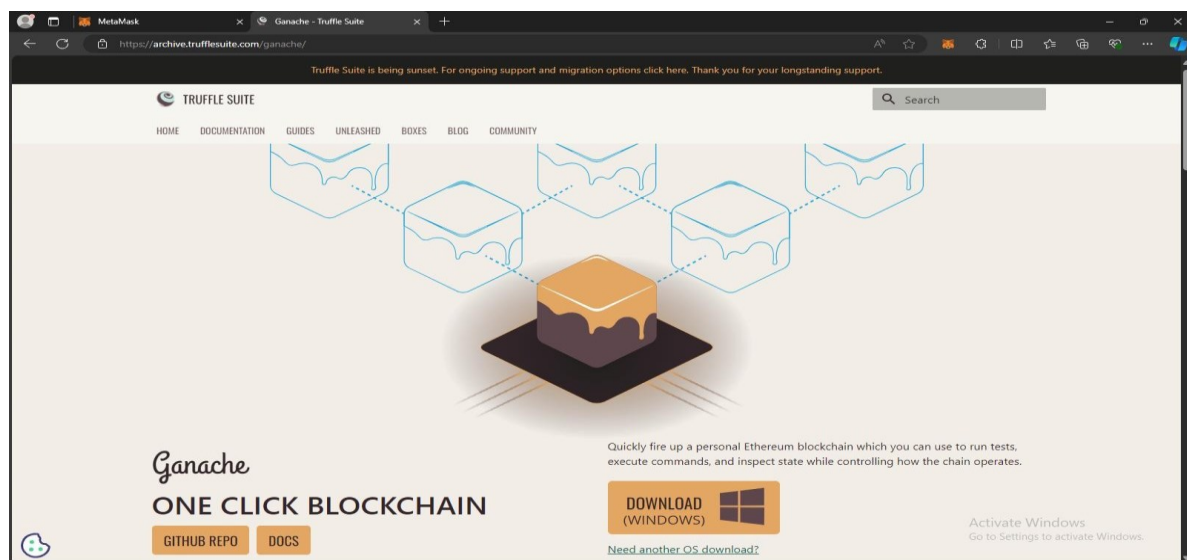
Practical 11: Creating a simple DApp for performing basic mathematical operations on two numbers.

Part 1: Setting up MetaMask and Ganache.

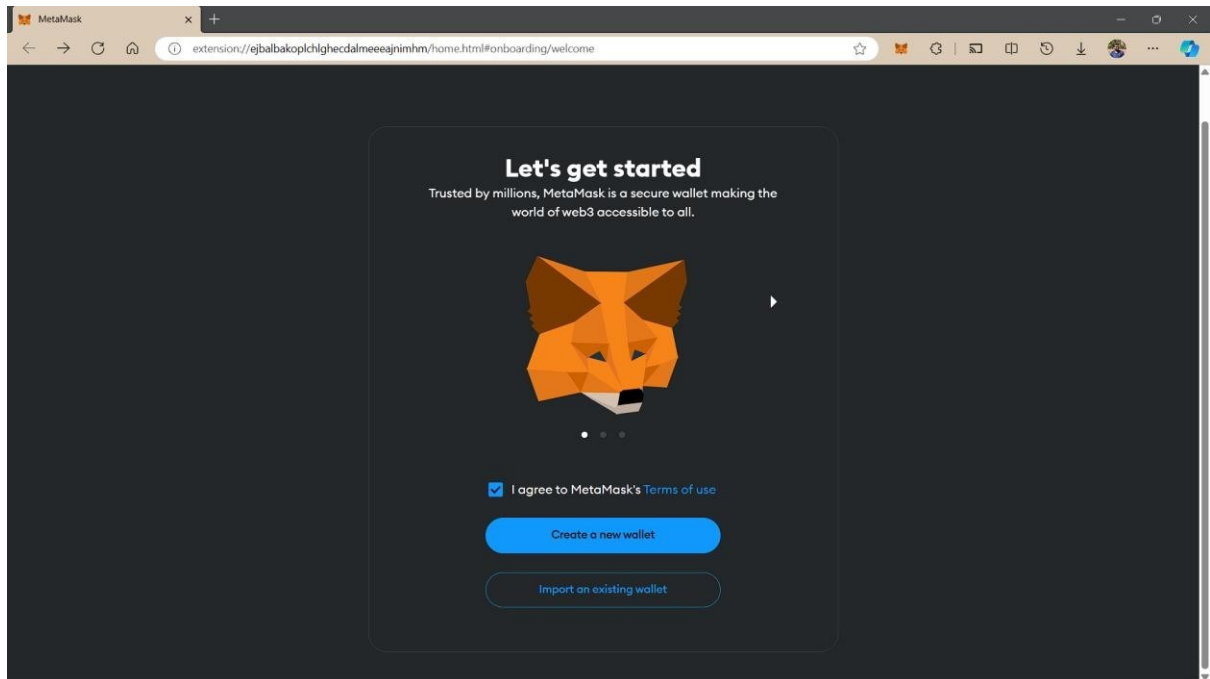
1. Install MetaMask Extension from [here](#).



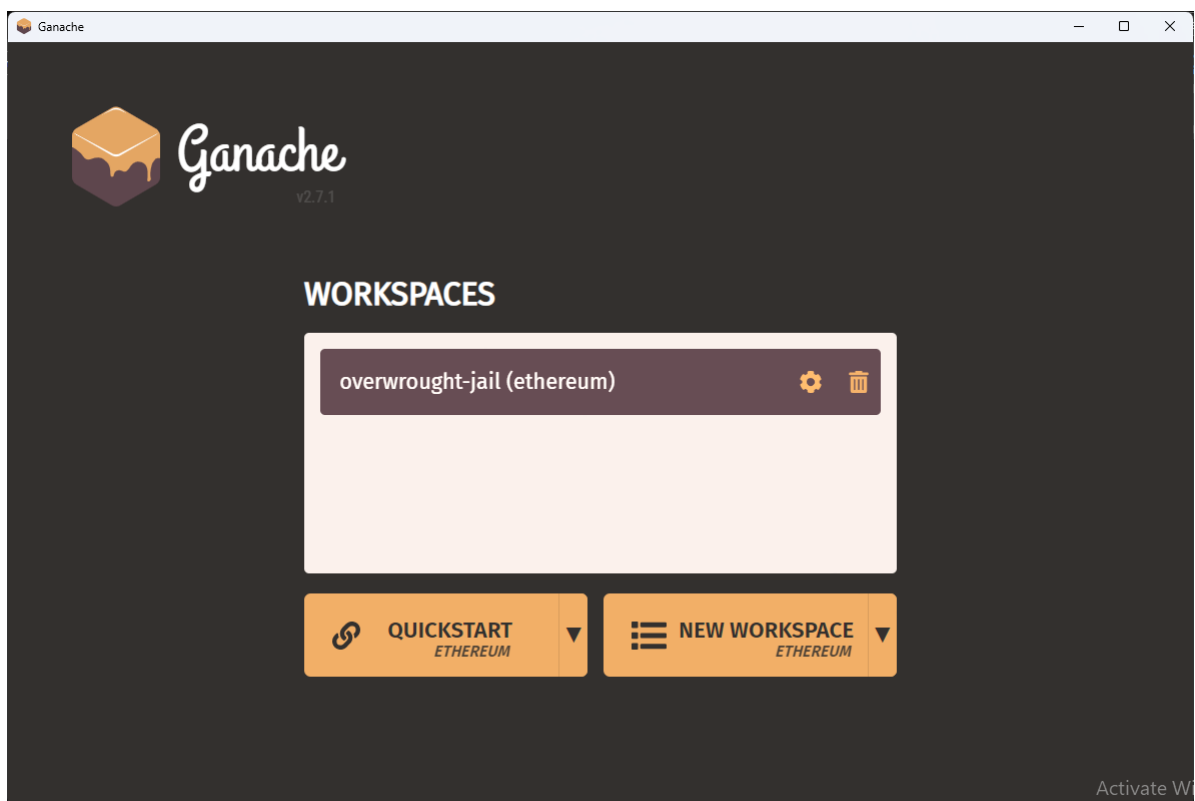
2. Install Ganache from [here](#).



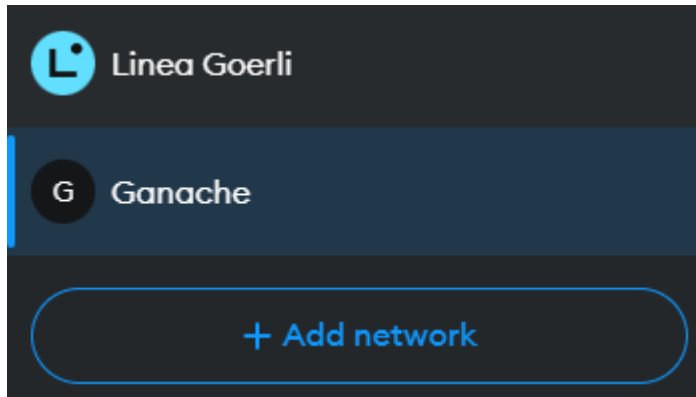
3. Create an Account on MetaMask or use an existing account.



4. Start a new workspace in Ganache.



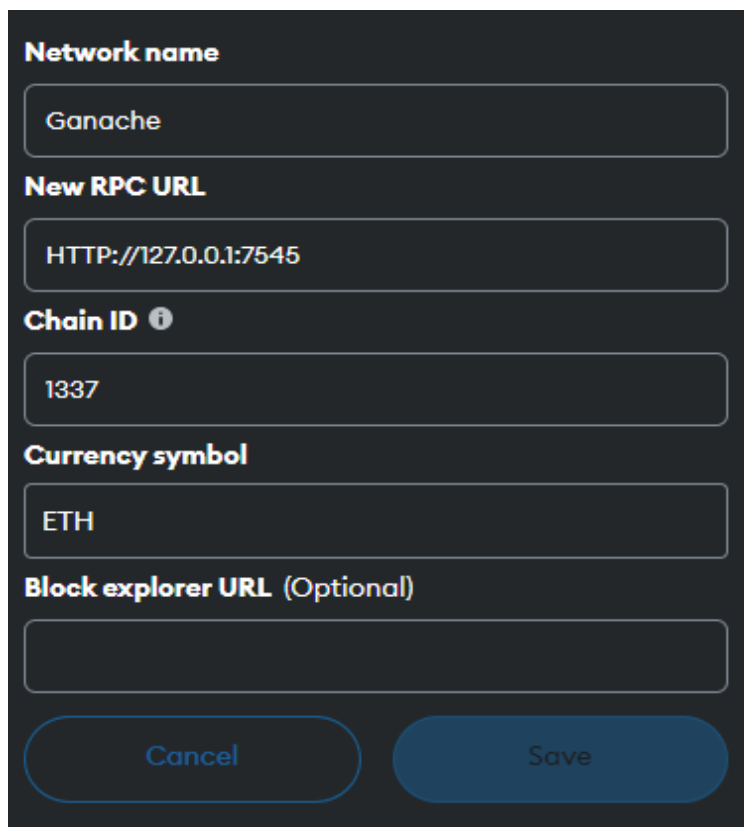
5. Go to MetaMask. Click on the Network Name → Add Network → Add Network Manually.



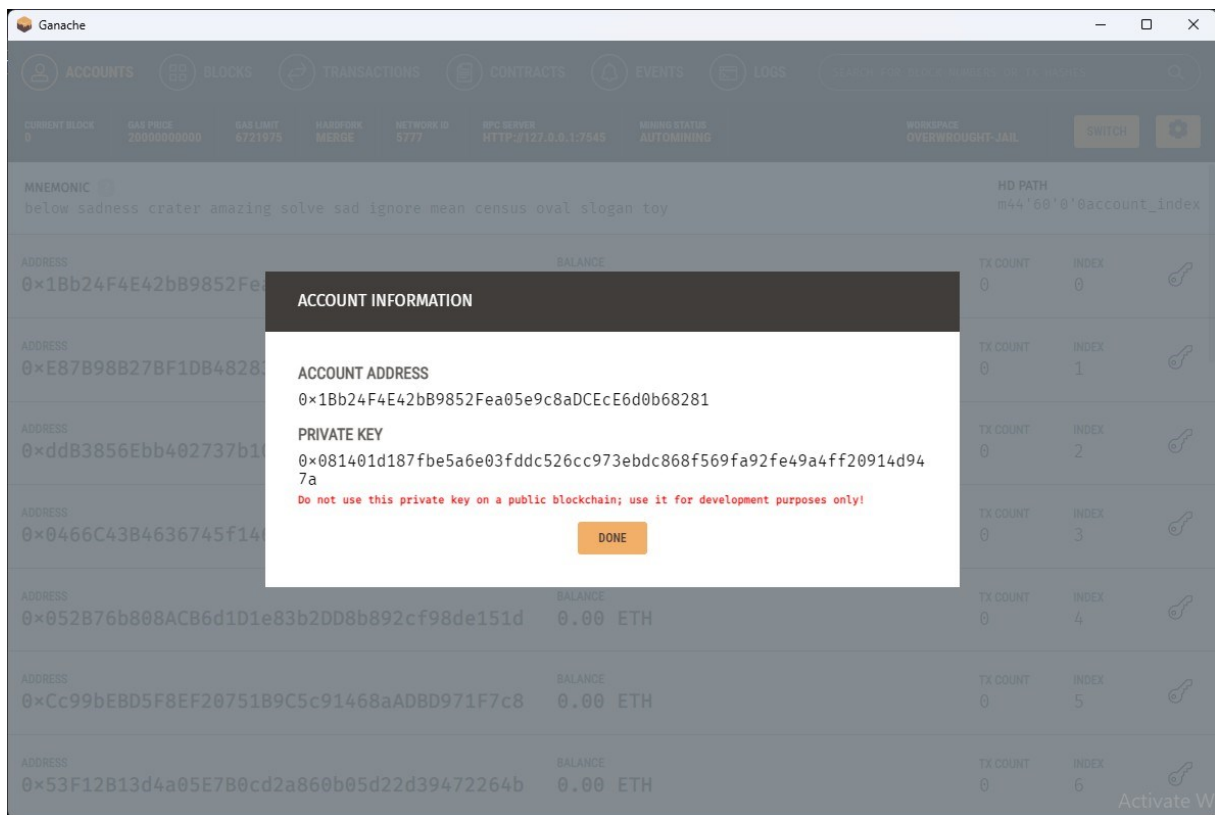
6. Give the Network a name of your choice.

7. Copy the RPC Server URL from Ganache GUI and paste it in the given box.
The default Chain ID is 1337.

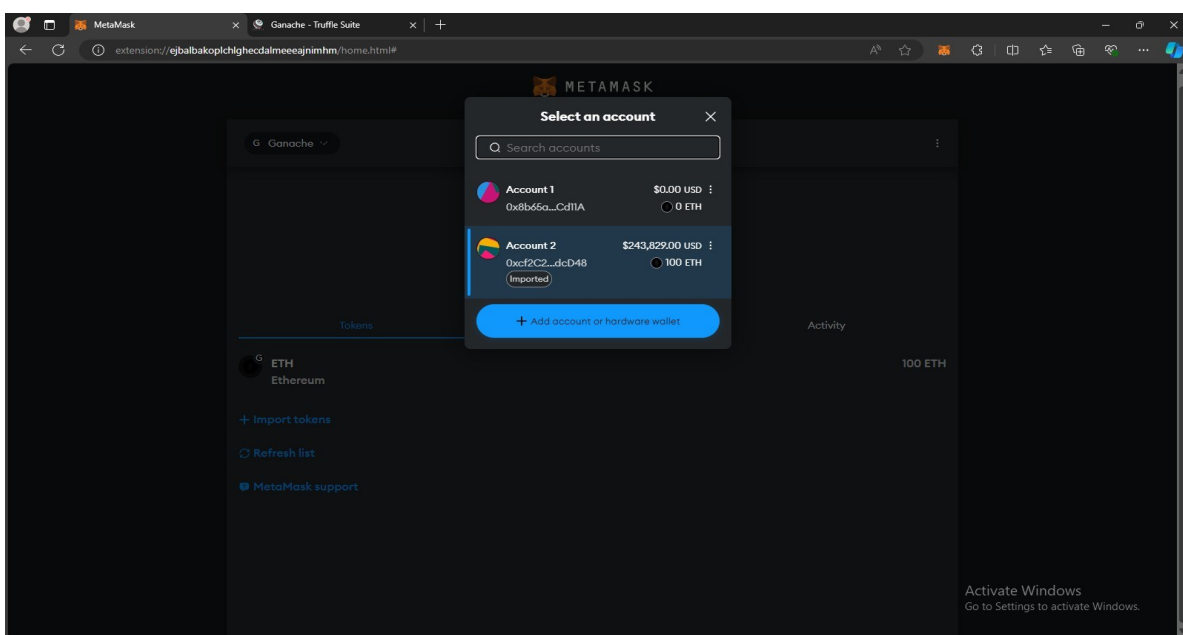
8. Give the currency symbol as ETH. Save the network.

A screenshot of the 'Add Network' form in the MetaMask application. The form is displayed on a dark background with white text. It contains five input fields: 'Network name' with the value 'Ganache', 'New RPC URL' with the value 'HTTP://127.0.0.1:7545', 'Chain ID' with the value '1337', 'Currency symbol' with the value 'ETH', and 'Block explorer URL (Optional)' which is currently empty. At the bottom of the form, there are two buttons: 'Cancel' and 'Save'. The 'Save' button is highlighted in a darker blue color.

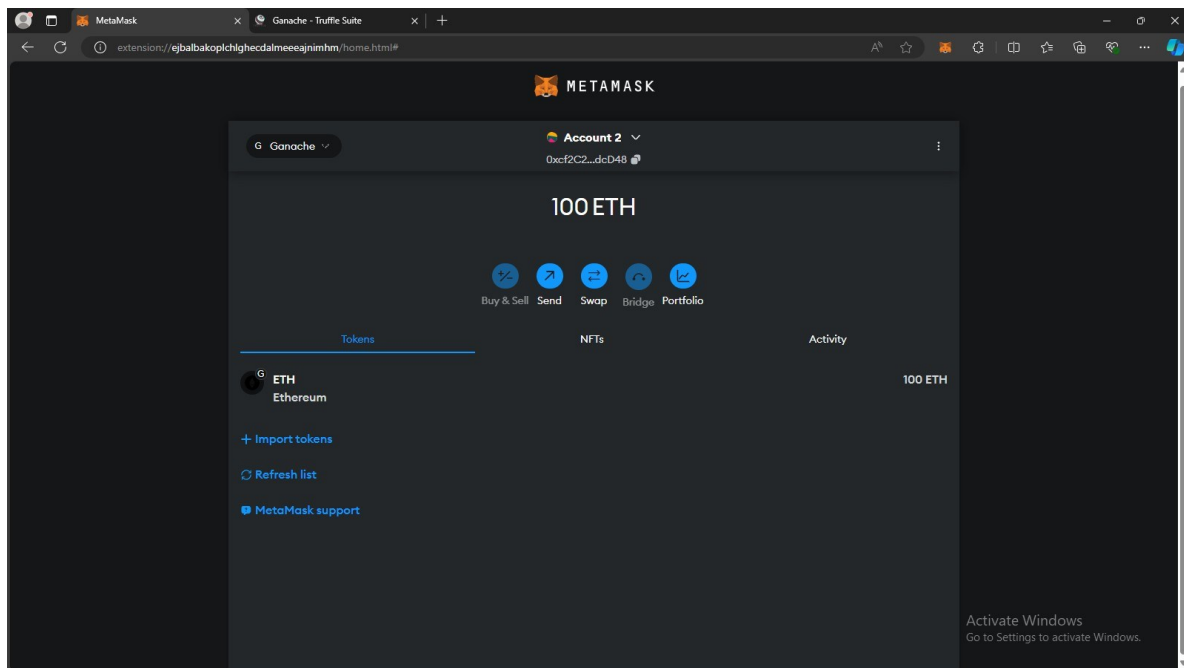
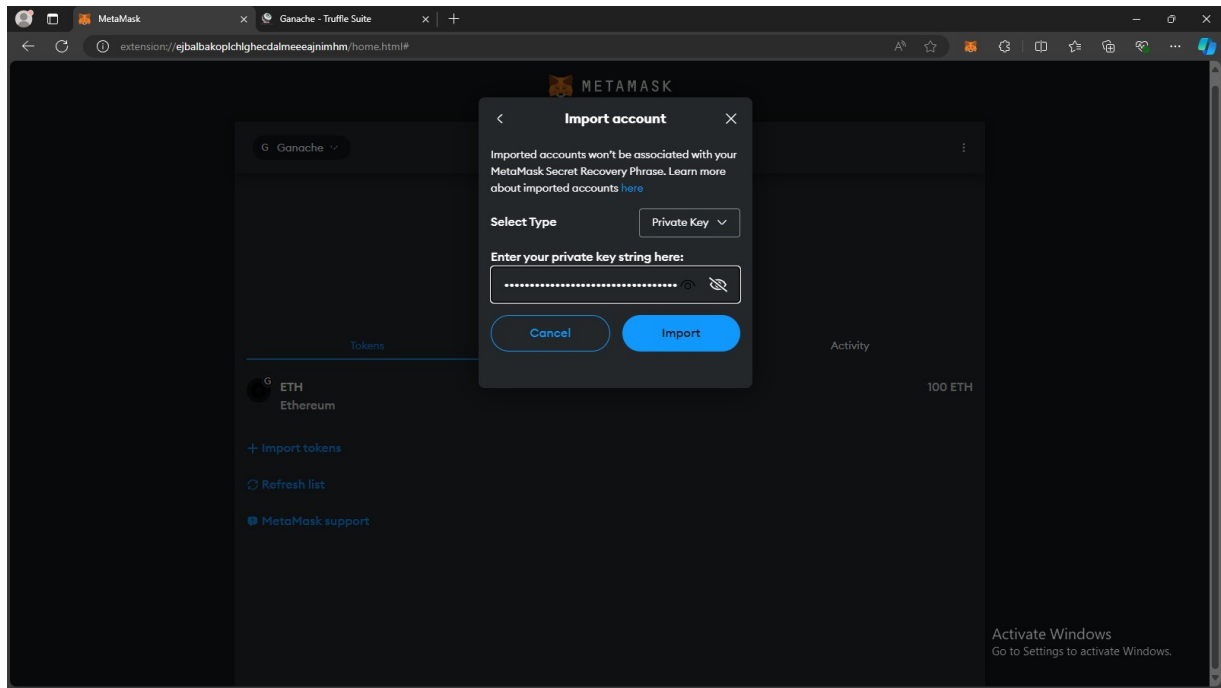
9. Go to Ganache and Click on the key symbol. Copy the private key.



10. Go to MetaMask and click on Account Name. Select Add Account or Hardware Wallet → Import Account and paste the private key in the given box.



Name : Pratik Tiruwa
Seat No : 31031523035



Part 2: Setting up VS Code.

Prerequisites:

a. Check if node and npm are installed with the following

commands `node -v` & `npm -v`

b. Install truffle, ganache & lite-server using the following command

`npm install -g truffle` / `npm install -g ganache` / `npm install lite-server --dev`

c. Ensure Ganache is running in the background

1. Open Terminal in VS Code and initialise Truffle with the following command

`truffle init`

```
PS D:\DApps> truffle init

Starting init...
=====

> Copying project files to D:\DApps

Init successful, sweet!

Try our scaffold commands to get started:
$ truffle create contract YourContractName # scaffold a contract
$ truffle create test YourTestName        # scaffold a test

http://trufflesuite.com/docs
```

2. Create a new contract in the contracts folder. And write a smart contract for adding two number.

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.19;

contract Addition {
    function add(uint256 a, uint256 b) public pure returns (uint256) {
        return a + b;
    }
}
```

3. Create a new folder frontend and make `index.html` and `app.js` files inside.

`index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>DApp-1</title>
</head>
<body>
  <h1>Blockchain Addition DApp</h1>
  <input type="number" id="num1" placeholder="Enter first number">
  <input type="number" id="num2" placeholder="Enter second number">
  <button onclick="addNumber()">Add Numbers</button>
  <h3>Result: <span id="result"></span></h3>

  <script
src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
    <script src="app.js"></script>
  </body>
</html>
```

`app.js`

```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;

window.addEventListener("load", async () => {
  if (window.ethereum) {
    web3 = new Web3(window.ethereum);
    await window.ethereum.enable();
  } else {
    console.log("MetaMask not detected. Please install MetaMask.");
  }
});
```

```
        contract = new web3.eth.Contract(contractABI, contractAddress);
    });

    async function addNumber() {
        const num1 = document.getElementById("num1").value;
        const num2 = document.getElementById("num2").value;
        const accounts = await web3.eth.getAccounts();
        console.log(num1);
        console.log(num2);
        contract.methods
            .add(num1, num2)
            .call({ from: accounts[0] })
            .then((result) => {
                console.log(result);
                document.getElementById("result").innerText = `${result}`;
            });
    }
}
```

4. Create `1_deploy.js` in the migrations folder.

```
const Addition = artifacts.require("Addition");

module.exports = async function (deployer) {
    await deployer.deploy(Addition);
    const instance = await Addition.deployed();
    console.log("Addition deployed at:", instance.address);
};
```

5. Create `test.js` in the test folder to verify the contracts before deploying it.

```
const Addition = artifacts.require("Addition");

contract("Addition", () => {
    it("should add two numbers correctly", async () => {
        const addition = await Addition.deployed();
        console.log("Contract Address: ", addition.address);
        const result = await addition.add(5, 3);
    });
});
```

```
    assert.equal(result.toNumber(), 8, "Addition of 5 and 3 should be 8");  
  });  
});
```

6. In the source directory create a new file `bs-config.json` and set the base directory as frontend.

```
{  
  "server": {  
    "baseDir": ["/frontend"]  
  }  
}
```

7. Make sure about the following things

a. In the `truffle-config.js` uncomment your network details. And ensure the `port` and `network_id` match with the `RPC Server` which can be found in Ganache GUI

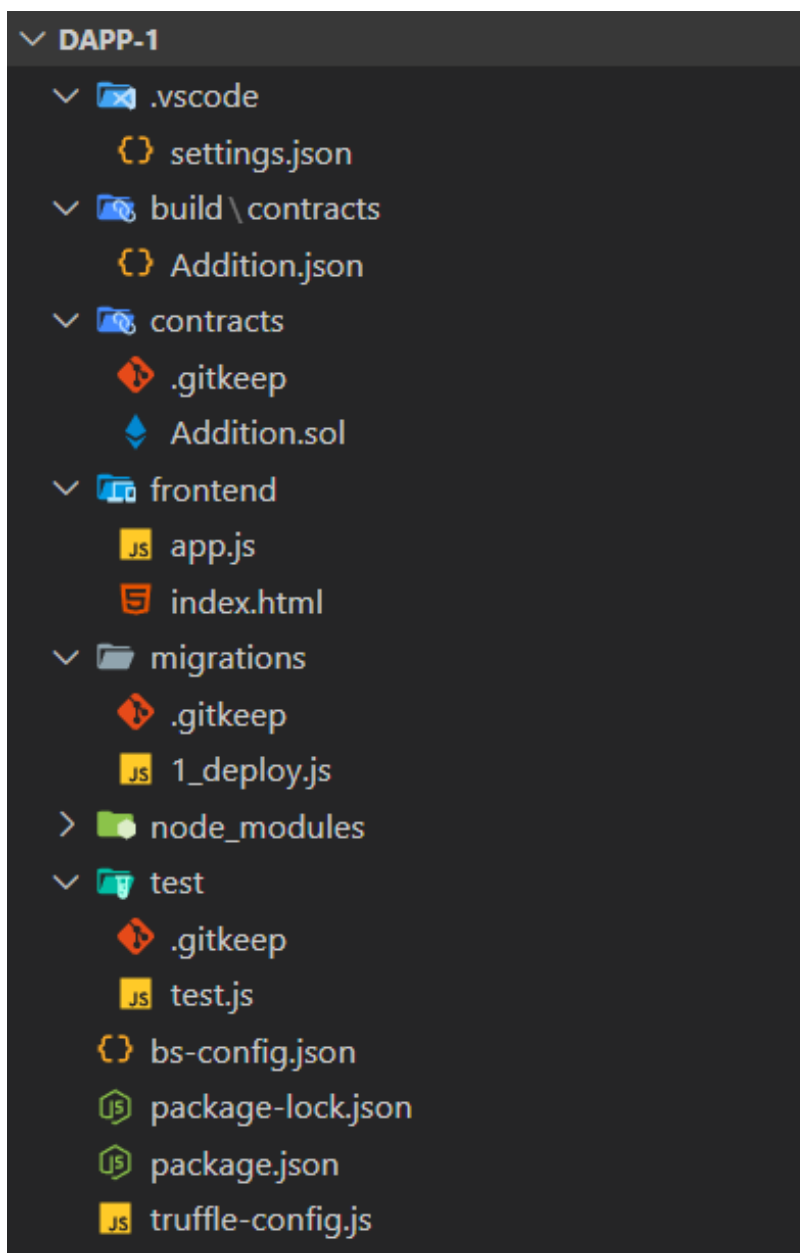
b. Ensure that solidity compiler version is set to 0.8.19 in the same file.

```
module.exports = {  
  networks: {  
    development: {  
      host: "127.0.0.1",  
      port: 7545,  
      network_id: "5777",  
    },  
  },  
  
  // Configure your compilers  
  compilers: {  
    solc: {  
      version: "0.8.19"  
    }  
  }  
};
```

a. Ensure necessary dependencies are mentioned in the `package.json`.


```
{  
  "dependencies": {  
    "lite-server": "^2.6.1"  
  },  
  "scripts": {  
    "start": "lite-server"  
  }  
}
```

b. The final directory structure should look like this



Part 3: Running the DApp.

1. In a new terminal set directory to source and run `truffle compile` command.

```
PS D:\Notes\PG\PG Sem 3\Practicals\Blockchain\DApp-1> truffle compile

Compiling your contracts...
=====
> Compiling .\contracts\Addition.sol
> Compiling .\contracts\Addition.sol
> Artifacts written to D:\Notes\PG\PG Sem 3\Practicals\Blockchain\DApp-1\build\contracts
> Compiled successfully using:
   - solc: 0.8.19+commit.7dd6d404.Emscripten.clang
```

2. Go to build → contracts → Addition.json. Look for `abi` and Copy the complete array. Paste it in the `contractABI` constant inside `app.js`.

```
const contractABI = [{
  inputs: [
    {
      internalType: "uint256",
      name: "a",
      type: "uint256",
    },
    {
      internalType: "uint256",
      name: "b",
      type: "uint256",
    },
  ],
  name: "add",
  outputs: [
    {
      internalType: "uint256",
      name: "",
      type: "uint256",
    },
  ],
  stateMutability: "pure",
  type: "function",
},
];
```

3. Next run `truffle migrate`. Make note of the Contract Address displayed in the terminal.

```
Compiling your contracts...
=====
> Compiling .\contracts\Addition.sol
> Artifacts written to D:\Notes\PG\PG Sem 3\Practicals\Blockchain\DApp-1\build\contracts
> Compiled successfully using:
  - solc: 0.8.19+commit.7dd6d404.Emscripten.clang

Starting migrations...
=====
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)

1 deploy.js
=====

Replacing 'Addition'
-----
> transaction hash: 0x791e3ab88d05b3012242b865bbd1105d39a645bcffaa0857f2c58ba562c22073
> Blocks: 0        Seconds: 0
> contract address: 0x9FC99312430F79737561207e8d26a2B92D3f280B
> block number:     24
> block timestamp:   1728397885
> account:          0xfFcFc0BF73A62a7A6b197fAB896164944A297B35
> balance:          99.99234039414607952
> gas used:         146899 (0x23dd3)
> gas price:        2.545027085 gwei
> value sent:       0 ETH
> total cost:       0.000373861933759415 ETH

Addition deployed at: 0x9FC99312430F79737561207e8d26a2B92D3f280B
> Saving artifacts
-----
> Total cost:       0.000373861933759415 ETH

Summary
=====
> Total deployments: 1
> Final cost:       0.000373861933759415 ETH
```

4. Copy the contract address and paste it in the `contractAddress` constant in the `app.js` file.

```
const contractAddress = "0xf3539bF7942055a8944EB7048A15450c05b1815A";
```

5. Run `truffle test` to ensure our contract is correct.

```
Using network 'development'.

Compiling your contracts...
=====
> Compiling .\contracts\Addition.sol
> Artifacts written to C:\Users\Admin\AppData\Local\Temp\test--19664-8nJHijHoKW2f
> Compiled successfully using:
  - solc: 0.8.19+commit.7dd6d404.Emscripten.clang
Addition deployed at: 0xb8D72a1caE30c216B0dE94f99f82b219A6f25A3c

Contract: Addition
Contract Address: 0xb8D72a1caE30c216B0dE94f99f82b219A6f25A3c
  ✓ should add two numbers correctly

1 passing (66ms)
```

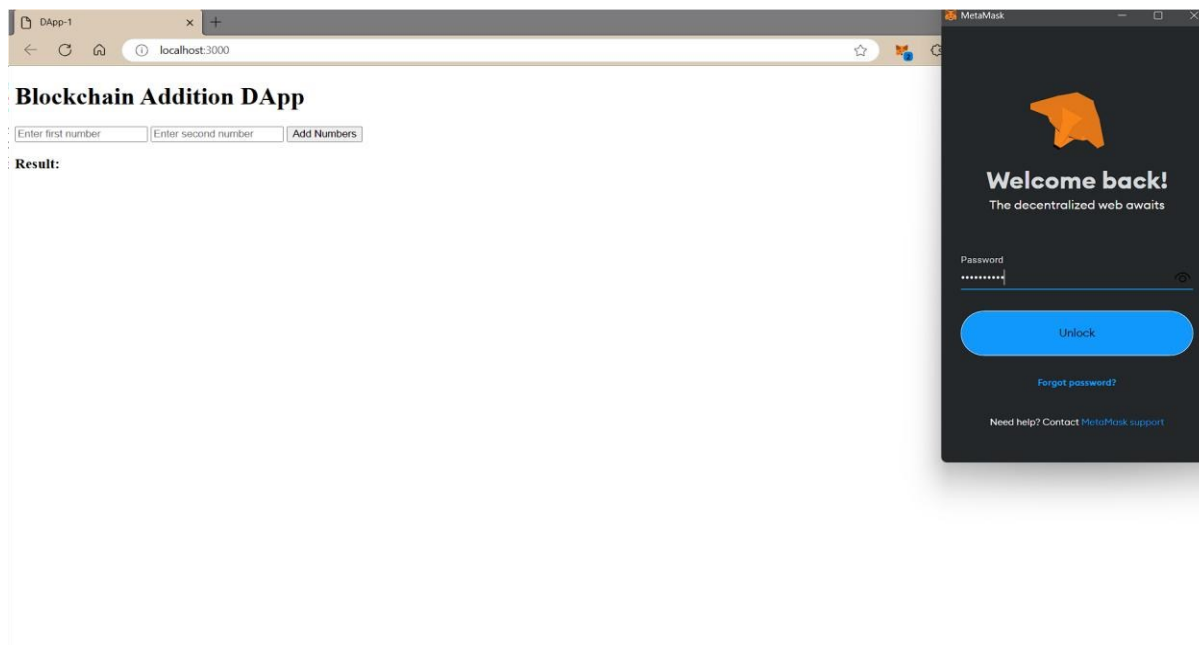
6. Run `npm start` if everything is correct.

```
PS D:\Notes\PG\PG Sem 3\Practicals\Blockchain\DApp-1> npm start

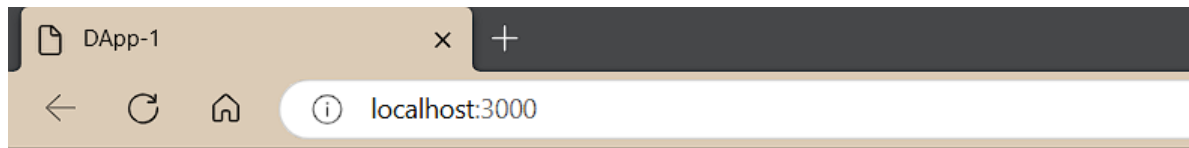
> start
> lite-server

** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.html,css,js' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: [ './frontend' ],
    middleware: [ [Function (anonymous)], [Function (anonymous)] ]
  }
}
[Browsersync] Access URLs:
-----
    Local: http://localhost:3000
  External: http://192.168.0.224:3000
-----
    UI: http://localhost:3001
  UI External: http://localhost:3001
-----
[Browsersync] Serving files from: ./frontend
[Browsersync] Watching files...
24.10.08 20:12:58 200 GET /index.html
24.10.08 20:12:58 200 GET /app.js
24.10.08 20:12:58 404 GET /favicon.ico
```

7. Sign in to MetaMask and grant the required access.



8. Give the input and click on Add Numbers. The result should be displayed.



Blockchain Addition DApp

45 98 Add Numbers

Result: 143

Part 4: Modify the DApp to integrate subtraction, multiplication & division operations.

1. Make changes in the smart contract (Operations.sol → I have renamed thefile)

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.19;

contract Operations {
    function add(uint256 a, uint256 b) public pure returns (uint256) {
        return a + b;
    }

    function sub(uint256 a, uint256 b) public pure returns (uint256) {
        return a - b;
    }

    function mul(uint256 a, uint256 b) public pure returns (uint256) {
        return a * b;
    }

    function div(uint256 a, uint256 b) public pure returns (uint256) {
        return a / b;
    }
}
```

2. Modify index.html to accommodate other buttons and onClick functions.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>DApp-1</title>
  </head>
  <body>
    <h1>Blockchain Addition DApp</h1>
```

```
Number 1:
<input type="number" id="num1" placeholder="Enter first number" />
<br /><br />
Number 2:
<input type="number" id="num2" placeholder="Enter second number" />
<br />
<h2>Choose Operation:</h2>
<button onclick="addNumber()">Add</button>
<button onclick="subNumber()">Sub</button>
<button onclick="mulNumber()">Mul</button>
<button onclick="divNumber()">Div</button>
<h2>Result:</h2>
<h3>Addition: <span id="resultA"></span></h3>
<h3>Subtraction: <span id="resultS"></span></h3>
<h3>Multiplication: <span id="resultM"></span></h3>
<h3>Division: <span id="resultD"></span></h3>
<script
src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
<script src="app.js"></script>
</body>
</html>
```

3. Similarly modify app.js.

```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;

window.addEventListener("load", async () => {
  if (window.ethereum) {
    web3 = new Web3(window.ethereum);
    await window.ethereum.enable();
  } else {
    console.log("MetaMask not detected. Please install MetaMask.");
  }
})
```

```
    contract = new web3.eth.Contract(contractABI, contractAddress);
  });

  async function addNumber() {
    const num1 = document.getElementById("num1").value;
    const num2 = document.getElementById("num2").value;
    const accounts = await web3.eth.getAccounts();
    console.log(num1);
    console.log(num2);
    contract.methods
      .add(num1, num2)
      .call({ from: accounts[0] })
      .then((result) => {
        console.log(result);
        document.getElementById("resultA").innerText = `${result}`;
      });
  }

  async function subNumber() {
    const num1 = document.getElementById("num1").value;
    const num2 = document.getElementById("num2").value;
    const accounts = await web3.eth.getAccounts();
    console.log(num1);
    console.log(num2);
    contract.methods
      .sub(num1, num2)
      .call({ from: accounts[0] })
      .then((result) => {
        console.log(result);
        document.getElementById("resultS").innerText = `${result}`;
      });
  }

  async function mulNumber() {
    const num1 = document.getElementById("num1").value; const num2 =
    document.getElementById("num2").value; const accounts = await web3.eth.getAccounts();
```



```
    console.log(num2);
    contract.methods
      .mul(num1, num2)
      .call({ from: accounts[0] })
      .then((result) => {
        console.log(result);
        document.getElementById("resultM").innerText = `${result}`;
      });
  }

  async function divNumber() {
    const num1 = document.getElementById("num1").value;
    const num2 = document.getElementById("num2").value;
    const accounts = await web3.eth.getAccounts();
    console.log(num1);
    console.log(num2);
    contract.methods
      .div(num1, num2)
      .call({ from: accounts[0] })
      .then((result) => {
        console.log(result);
        document.getElementById("resultD").innerText = `${result}`;
      });
  }
}
```

4. Modify 1_deploy.js (If you haven't renamed leave it as is)

```
const Operations = artifacts.require("Operations");

module.exports = async function (deployer) {
  await deployer.deploy(Operations);
  const instance = await Operations.deployed();
  console.log("Operations deployed at:", instance.address);
};
```

5. Update test.js to include different test cases.

```
const Operations = artifacts.require("Operations");

contract("Operations", () => {
  let operationsInstance;

  before(async () => {
    operationsInstance = await Operations.deployed();
  });

  it("should add two numbers correctly", async () => {
    console.log("Contract Address: ", operationsInstance.address);
    const resultA = await operationsInstance.add(5, 2);
    assert.equal(resultA.toNumber(), 7, "Addition of 5 and 2 should be 7");
  });

  it("should subtract two numbers correctly", async () => {
    console.log("Contract Address: ", operationsInstance.address);
    const resultS = await operationsInstance.sub(5, 2);
    assert.equal(resultS.toNumber(), 3, "Subtraction of 5 and 2 should be 3");
  });

  it("should multiply two numbers correctly", async () => {
    console.log("Contract Address: ", operationsInstance.address);
    const resultM = await operationsInstance.mul(5, 2);
    assert.equal(resultM.toNumber(), 10, "Multiplication of 5 and 2 should be 10");
  });

  it("should divide two numbers correctly", async () => {
    console.log("Contract Address: ", operationsInstance.address);
    const resultD = await operationsInstance.div(5, 5);
    assert.equal(resultD.toNumber(), 1, "Division of 5 and 5 should be 1");
  });
});
```

6. Run the DApp following the same steps in Part 3.

DApp-1

×

+

←

↺

🏠

ⓘ

localhost:3000

Blockchain Operations DApp

Number 1:

Number 2:

Choose Operation:

Add

Sub

Mul

Div

Result:

Addition: 522

Subtraction: 350

Multiplication: 37496

Division: 5

Practical 12

A. Create a DApp to calculate factorial of a number.

1. In a new terminal run `truffle init`
2. Create a new contract to calculate Factorial.

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.19;

contract factorial {
    function fact(uint n) public pure returns (uint) {
        if (n == 0) {
            return 1;
        } else {
            uint result = 1;
            for (uint i = 1; i <= n; i++) {
                result *= i;
            }
            return result;
        }
    }
}
```

- 3.. Make a new folder frontend and create two files, `index.html` & `app.js`.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>DApp-2</title>
  </head>
  <body>
    <h1>Blockchain Factorial DApp</h1>
    Number:
```

```
    <input type="number" id="num" placeholder="Enter Number" />
    <br /><br />
    <h2>Calculate Factorial:</h2>
    <button onclick="facto()">Calculate</button>
    <h2>Result: <span id="result"></span></h2>
    <script
src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
    <script src="app.js"></script>
  </body>
</html>
```

app.js (get contractABI & contractAddress after compilation and migration respectively)

```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;

window.addEventListener("load", async () => {
  if (window.ethereum) {
    web3 = new Web3(window.ethereum);
    await window.ethereum.enable();
  } else {
    console.log("MetaMask not detected. Please install MetaMask.");
  }

  contract = new web3.eth.Contract(contractABI, contractAddress);
});

async function facto() {
  const num = document.getElementById("num").value;
  const accounts = await web3.eth.getAccounts();
  console.log(num);
  contract.methods
```

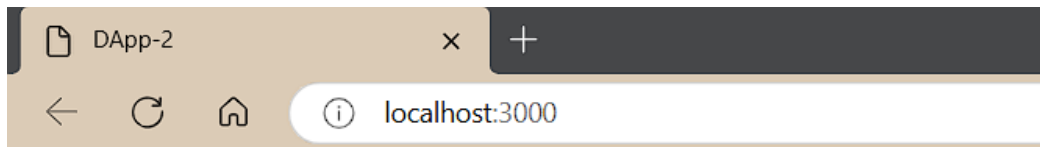
```
.fact(num)
.call({ from: accounts[0] })
.then((result) => {
  console.log(result);
  document.getElementById("result").innerText = `${result}`;
}));
}
```

4. Create 1_deploy.js in migrations folder.

```
const factorial = artifacts.require("factorial");

module.exports = async function (deployer) {
  await deployer.deploy(factorial);
  const instance = await factorial.deployed();
  console.log("Operations deployed at:", instance.address);
};
```

5. Run the DApp by `npm start`. Connect wallet and test.



Blockchain Factorial DApp

Number:

Calculate Factorial:

Result: 120

B. Create a DApp to implement transactions between two accounts.

1. index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>DApp-3</title>
  </head>
  <body>
    <h1>Blockchain Transactions DApp</h1>
    <h2>Send Ether:</h2>
    <input type="text" id="toAddr" placeholder="To Address" />
    <input type="number" id="amount" placeholder="Amount" />
    <button onclick="send()">Send</button>
    <h2>Check Balance:</h2>
    <button onclick="checkBalance()">Check Balance</button>
    <p>Your Balance is: <span id="bal"></span></p>
    <script
src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
    <script src="app.js"></script>
    </body>
</html>
```

2. app.js

```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;

window.addEventListener("load", async () => {
  if (window.ethereum) {
    web3 = new Web3(window.ethereum);
```



```
    await window.ethereum.enable();
  } else {
    console.log("MetaMask not detected. Please install MetaMask.");
  }

  contract = new web3.eth.Contract(contractABI, contractAddress);
});

async function send() {
  const accounts = await web3.eth.getAccounts();
  const amount = web3.utils.toWei(document.getElementById('amount').value, 'ether');
  const toAddress = document.getElementById('toAddr').value;
  const sender = accounts[0];

  console.log("Sender: ", accounts[0]);
  console.log("Receiver: ", toAddress);
  console.log("Amount: ", amount);

  if (amount <= 0) {
    alert("Amount must be greater than 0");
    return;
  }
  else if (toAddress == "") {
    alert("Please enter receiver address");
    return;
  }
  else {
    contract.methods.transfers(toAddress).send({
      from: sender,
      value: amount
    }).on('transactionHash', (hash) => {
      console.log('Transaction Hash:', hash);
    }).on('receipt', (receipt) => {
      console.log('Transaction Receipt:', receipt);
    }).on('error', (error) => {
```

```
        console.error('Error:', error);
    });
}
};

async function checkBalance() {
    const accounts = await web3.eth.getAccounts();
    const balance = await web3.eth.getBalance(accounts[0]);
    const balanceInEther = web3.utils.fromWei(balance, 'ether');
    document.getElementById("bal").innerText = `${balanceInEther}`;
}
```

3. transactions.sol

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.19;

contract transactions {
    event Transfer(address indexed from, address indexed to, uint256 value);

    function transfers(address payable _to) public payable {
        require(msg.value > 0, "Send some ether");
        _to.transfer(msg.value);
        emit Transfer(msg.sender, _to, msg.value);
    }

    receive() external payable {
        emit Transfer(msg.sender, address(this), msg.value);
    }
}
```

1. 1_deploy.js

```
const transaction = artifacts.require("transactions");

module.exports = async function (deployer) {
```

```
  await deployer.deploy(transaction);
  const instance = await transaction.deployed();
  console.log("Contract deployed at:", instance.address);
};
```

1. bs-config.json

```
{
  "server": {
    "baseDir": ["../frontend"]
  }
}
```

2. Output:

ADDRESS	BALANCE
0x8d4e6F53AeEc3698af5ba4b3012257CfECEed938	89.00 ETH
0x9273a924CCCD7eBb553FB588790423C9a832E00d	106.00 ETH

C. Create a DApp to implement elections.

1. index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>DApp-4</title>
</head>

<body>
  <h1>Blockchain Voting DApp</h1>
  <h2>Select Candidate to Vote</h2>
  <button onclick="vote('Can1')">Candidate 1</button>
  <button onclick="vote('Can2')">Candidate 2</button>
  <button onclick="vote('Can3')">Candidate 3</button>
  <br><br>
  <h2>Check Results:</h2>
  <button onclick="checkResult()">Check Result</button>
  <p>The Winner Is: <span id="result"></span></p>

  <script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
  <script src="app.js"></script>
</body>

</html>
```

2. app.js

```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;
```

```
window.addEventListener("load", async () => {
  if (window.ethereum) {
    web3 = new Web3(window.ethereum);
    await window.ethereum.enable();
  } else {
    console.log("MetaMask not detected. Please install MetaMask.");
  }

  contract = new web3.eth.Contract(contractABI, contractAddress);
});

async function vote(can) {
  var canM = can;
  const accounts = await web3.eth.getAccounts();
  const voter = accounts[0];

  contract.methods.vote(canM).send({
    from: voter
  });
};

async function checkResult() {
  const accounts = await web3.eth.getAccounts();

  contract.methods.getWinner()
    .call({ from: accounts[0] })
    .then((winner) => {
      document.getElementById("result").innerText = `${winner}`;
    });
}s
```

3. voting.sol

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.19;

contract voting {
    mapping(string => uint256) public c;
    mapping(address => bool) public voters;
    string[] public cn;

    constructor() {
        cn = ["Can1", "Can2", "Can3"];
    }

    function vote(string memory caNm) public {
        require(!voters[msg.sender], "Already Voting Done.");
        bool ce = false;
        for (uint256 i = 0; i < cn.length; i++) {
            if (keccak256(bytes(caNm)) == keccak256(bytes(cn[i]))) {
                ce = true;
                break;
            }
        }
        require(ce, "Candidate does not exist.");
        c[caNm]++;
        voters[msg.sender] = true;
    }

    function getVoterC(string memory canM) public view returns (uint256) {
        return c[canM];
    }

    function getWinner() public view returns (string memory) {
        string memory winner;

        uint256 temp = 0;
```

```
    for (uint256 j = 0; j < cn.length; j++) {
        if (getVoterC(cn[j]) > temp) {
            temp = getVoterC(cn[j]);
            winner = cn[j];
        }
    }
    return winner;
}

function showPercentage(string memory canM) public view returns (uint256) {
    uint256 total;
    for (uint256 i = 0; i < cn.length; i++) {
        total = total + getVoterC(cn[i]);
    }

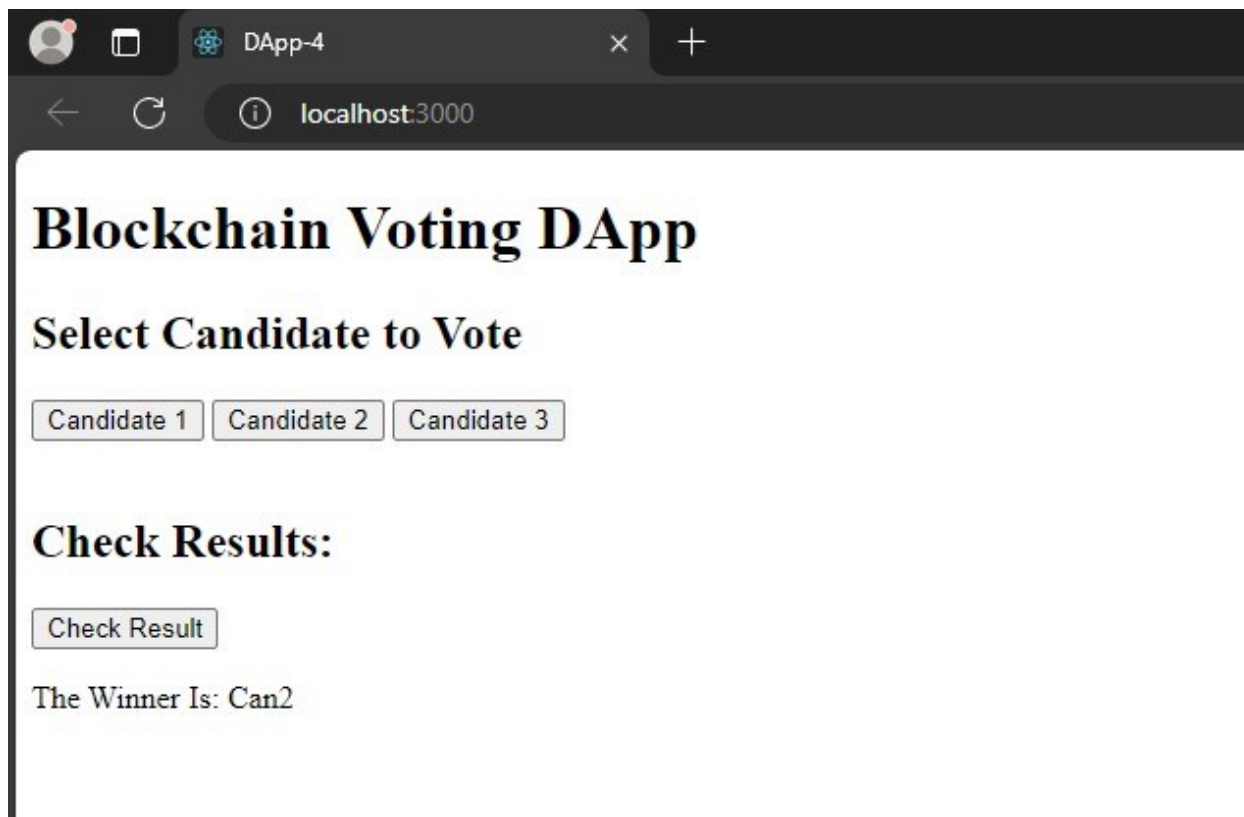
    uint256 per = getVoterC(canM) * (100 / total);
    return per;
}
}
```

4. 1_deploy.js

```
const vote = artifacts.require("voting");

module.exports = async function (deployer) {
    await deployer.deploy(vote);
    const instance = await vote.deployed();
    console.log("Contract deployed at:", instance.address);
};
```

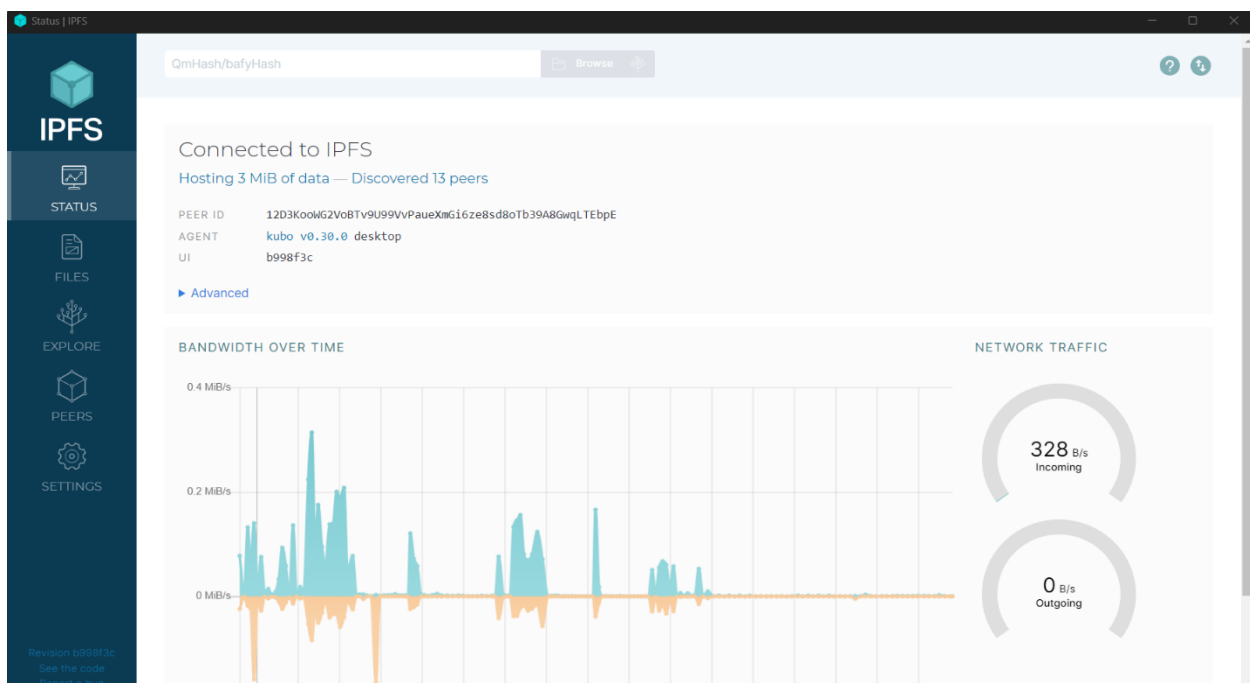

5. Output:



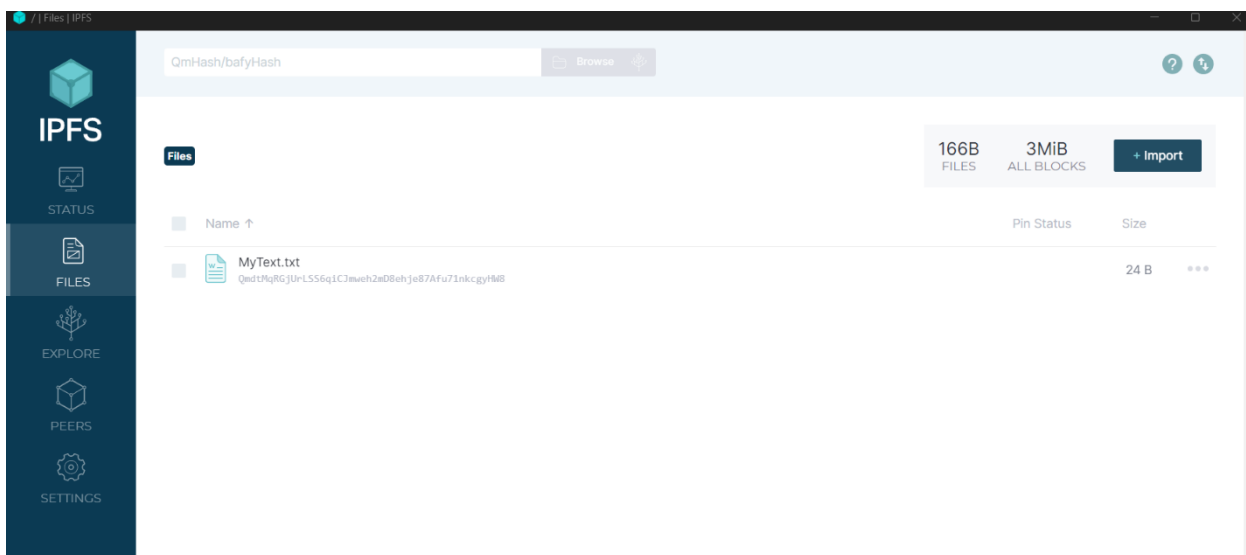
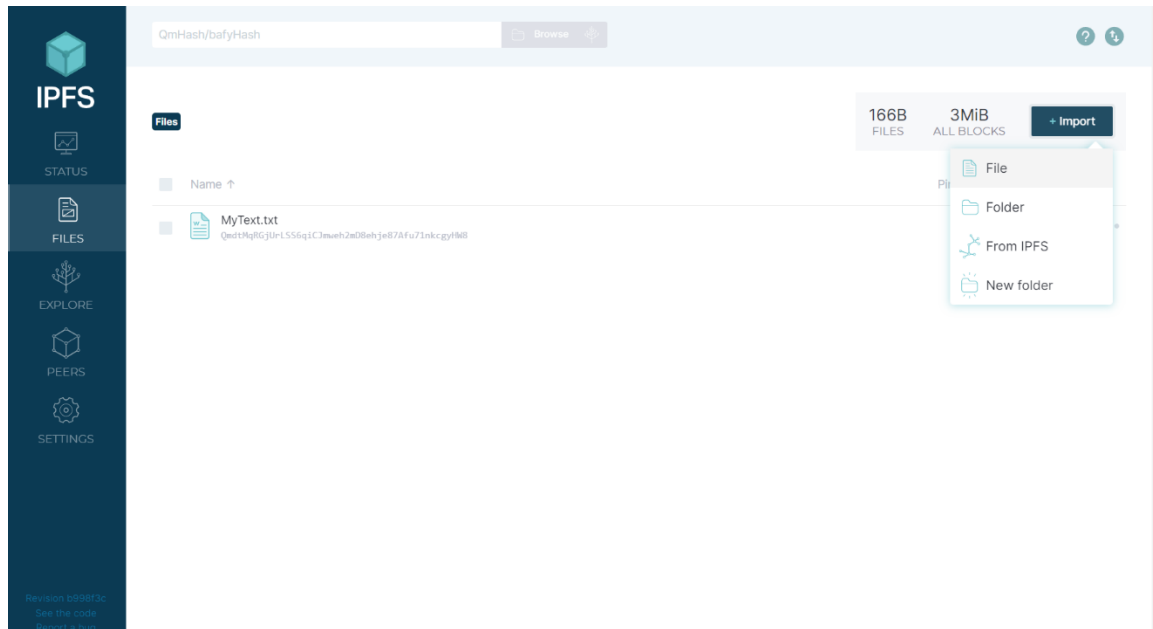
Practical 13 : Stroing and Retrieving files using IPFS

Step 1 : Download and Install IPFS Desktop from [here](#).

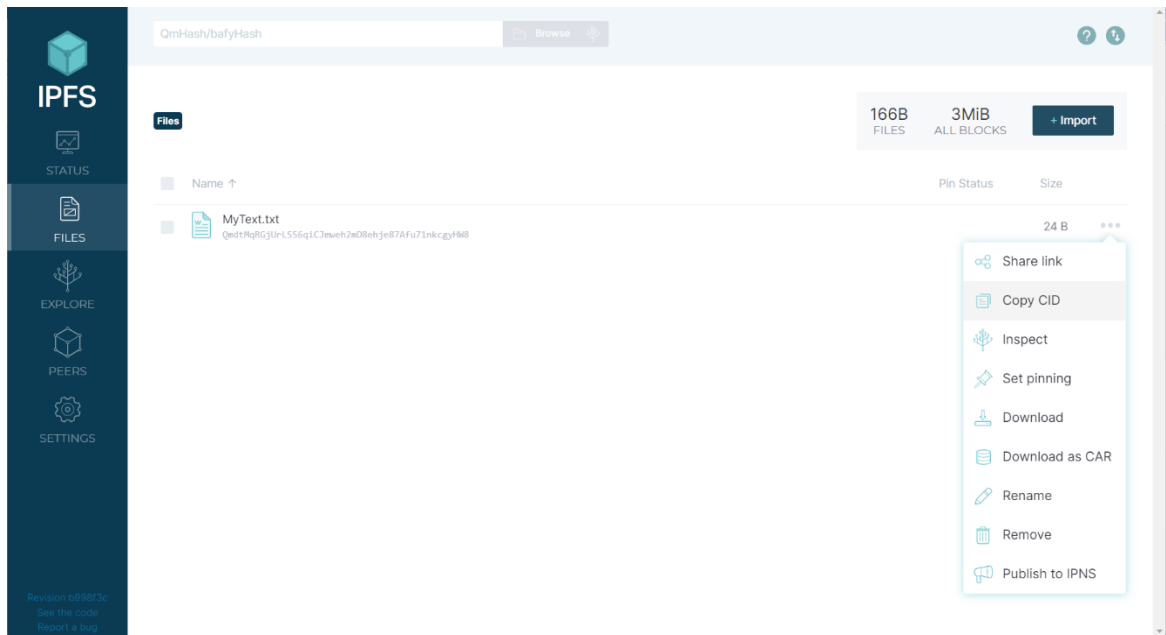
Contributors		
lidel		
▼ Assets 16		
ipfs-desktop-0.38.0-linux-amd64.deb	115 MB	Sep 13
ipfs-desktop-0.38.0-linux-x64.freebsd	116 MB	Sep 13
ipfs-desktop-0.38.0-linux-x64.tar.xz	113 MB	Sep 13
ipfs-desktop-0.38.0-linux-x86_64.AppImage	157 MB	Sep 13
ipfs-desktop-0.38.0-linux-x86_64.rpm	114 MB	Sep 13
ipfs-desktop-0.38.0-mac.dmg	248 MB	Sep 13
ipfs-desktop-0.38.0-mac.dmg.blockmap	266 KB	Sep 13
ipfs-desktop-0.38.0-squirrel.zip	239 MB	Sep 13
ipfs-desktop-0.38.0-squirrel.zip.blockmap	254 KB	Sep 13
IPFS-Desktop-Setup-0.38.0.exe	114 MB	Sep 13
IPFS-Desktop-Setup-0.38.0.exe.blockmap	123 KB	Sep 13
latest-linux.yml	738 Bytes	Sep 13
latest-mac.yml	520 Bytes	Sep 13
latest.yml	356 Bytes	Sep 13
Source code (zip)		Sep 13
Source code (tar.gz)		Sep 13



Step 2 : Click on files and import a sample file.

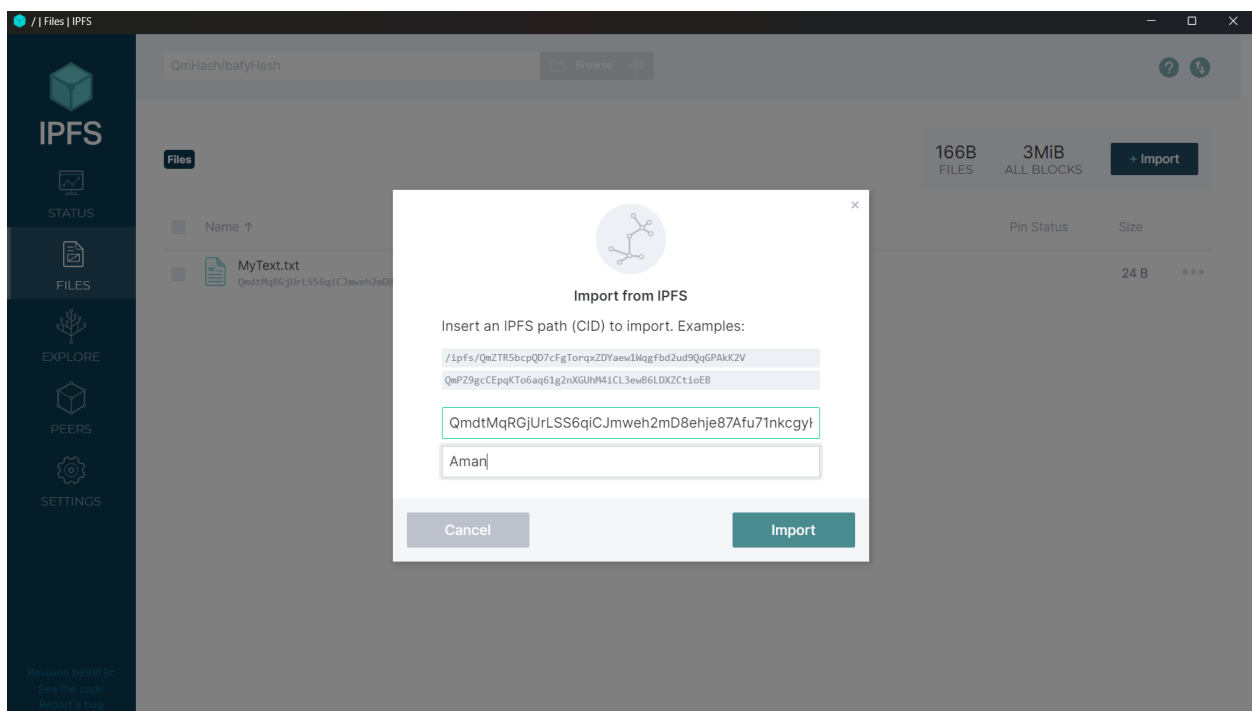


Step 3: Click on 3 dots and copy CID



Step 4: Share the CID to someone else to open the shared file.

Step 5: Click on import → Import from IPFS



Step 6: The imported file will be visible.

