

# **Blockchain Technologies**

## **Journal**

Submitted By

**Nagda Mihir Anupkumar**

Roll No: 31031523016

MSc CS – Part II

**Department Of Computer Science**

**Somaiya 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.

Name: Mihir Nagda  
Seat No: 31031523016

Practical 1A: Creating a simple blockchain to calculate sum of two number.

Code:

1. 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, '01/08/2024', 0, 0, '0');
  }

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

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

Name: Mihir Nagda  
Seat No: 31031523016

```
        nb.h = nb.calHash();
        this.chain.push(nb);
    }
}

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

## 2. test.js

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

let mb = new Blockchain();
console.log("Developed by Mihir Nagda - 31031523016");
console.log("First Transaction");

mb.addBlock(new Block(1, '01/08/2024', 23, 5));
console.log(JSON.stringify(mb, null, 3));
```

## Output:

```
PS C:\Users\admin\Desktop\Blockchain\Prac1A> node test.js
Developed by Mihir Nagda - 31031523016
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,
      "n2": 5,
      "sum": 28,
      "ph": "f0a3fc7bac56c13fb952d1f062337fe9d5c66953df566885bc504270d31f2aa8",
      "h": "2ed2dc7c51627b2e92b1b8c14722299ceeacde0b91f9af470a97bf304810c502"
    },
    {
      "i": 2,
      "t": "01/08/2024",
      "n1": 28,
      "n2": 65,
      "sum": 93,
      "ph": "2ed2dc7c51627b2e92b1b8c14722299ceeacde0b91f9af470a97bf304810c502",
      "h": "7e6c715a97d99e303b5b5ec761eed5c4c577a7e1ed947f2e80129203dd7d8d2d"
    }
  ]
}
```

Name: Mihir Nagda  
Seat No: 31031523016

Practical 1B: Creating a simple blockchain to calculate factorial of a number.

Code:

1. blockchain.js

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

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

Name: Mihir Nagda  
Seat No: 31031523016

```
        return new Block(0, '01/08/2024', 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;
```

## 2. test.js

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

let mb = new Blockchain();
console.log("Developed by Mihir Nagda - 31031523016")

mb.addBlock(new Block(1, '01/08/2024', 5))
mb.addBlock(new Block(2, '01/08/2024', 12))
console.log(JSON.stringify(mb, null, 3));
```

Name: Mihir Nagda  
Seat No: 31031523016

Output:

```
PS C:\Users\admin\Desktop\Blockchain\Prac1B> node test.js
Developed by Mihir Nagda - 31031523016
{
  "chain": [
    {
      "i": 0,
      "t": "01/08/2024",
      "n": 0,
      "fact": 1,
      "ph": 0,
      "h": "e019a5df4749fd98d7ebb9b957686271eec40fce8eb476a81d0ec5e3d1464dc0"
    },
    {
      "i": 1,
      "t": "01/08/2024",
      "n": 5,
      "fact": 120,
      "ph": "e019a5df4749fd98d7ebb9b957686271eec40fce8eb476a81d0ec5e3d1464dc0",
      "h": "11fab906dc85a559cd8dd9b6d964aeef99c43fc189cd8b383673e648fa8ba936"
    },
    {
      "i": 2,
      "t": "01/08/2024",
      "n": 12,
      "fact": 479001600,
      "ph": "11fab906dc85a559cd8dd9b6d964aeef99c43fc189cd8b383673e648fa8ba936",
      "h": "370575f3a3455dc985787e4f5692ee4b69350d601ef5ecf296e7831bb76eec5c"
    }
  ]
}
```



Name: Mihir Nagda  
Seat No: 31031523016

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

Code:

1. blockchain.js

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

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

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

    if (temp == 0) {
      return "0";
      // console.log("0")
    }
  }
}
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        else if (temp == 1) {
            return "Happy number";
            // console.log("HappyNumber")
        }

        else {
            return "Not a Happy Number"
            // console.log("Not a HappyNumber")
        }
    }

    calHash() {
        return c.createHash('sha256').update(this.i + this.t + this.happy +
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) {
        nb.ph = this.getCBlock().h;
        nb.h = nb.calHash();
        this.chain.push(nb);
    }
}
```

Name: Mihir Nagda  
Seat No: 31031523016

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

## 2. test.js

```
const { Block, Blockchain } = require('./blockchain');  
  
let mb = new Blockchain();  
console.log("Developed by Mihir Nagda - 31031523016")  
  
mb.addBlock(new Block(1, new Date(), 10));  
mb.addBlock(new Block(2, new Date(), 15));  
console.log(JSON.stringify(mb, null, 3));
```

## Output:

```
PS D:\Notes\PG\PG Sem 3\Practicals\Blockchain> node "d:\Notes\PG\PG Sem 3\Practicals\Blockchain\Prac2\test.js"  
Developed by Mihir Nagda - 31031523016  
{  
  "chain": [  
    {  
      "i": 0,  
      "t": "2024-08-08T16:20:26.264Z",  
      "n": 0,  
      "happy": "0",  
      "ph": "0",  
      "h": "3c2fafcd4b4f5f6f895b83c7826bfb10f1a08da7de386130ea42bd4cc8d41277"  
    },  
    {  
      "i": 1,  
      "t": "2024-08-08T16:20:26.278Z",  
      "n": 10,  
      "happy": "Happy number",  
      "ph": "3c2fafcd4b4f5f6f895b83c7826bfb10f1a08da7de386130ea42bd4cc8d41277",  
      "h": "ecae7ac05064334ae1861e12896ef89e597870c2e6f2408cfa2eb9b3d132184e"  
    },  
    {  
      "i": 2,  
      "t": "2024-08-08T16:20:26.278Z",  
      "n": 15,  
      "happy": "Not a Happy Number",  
      "ph": "ecae7ac05064334ae1861e12896ef89e597870c2e6f2408cfa2eb9b3d132184e",  
      "h": "2075b9e8c4d77314a19552b1daaab1c2da73a0776f1cee199287924e39087f"  
    }  
  ]  
}
```

Name: Mihir Nagda  
Seat No: 31031523016

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

Code:

#### 1. blockchain.js

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

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

  /*
  kap() {
    let n1 = this.n;
    let sq = n1 * n1;
    let s = 0;
    let c = 0;

    while (sq != 0) {
      // let r = sq % 10;
      c = c + 1;
      sq = Math.floor(sq / 10);
    }

    for (let i = 1; i < c; i++) {
      let sq1 = n1 * n1;
      s = 0;
      while (sq1 != 0) {
        let r = sq1 % Math.pow(10, i);
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        s = s + r;
        sq1 = Math.floor(sq1 / Math.pow(10, i));
    }

    if (this.n == s) {
        // return "Kaprekar Number";
        this.kaprekar = "Kaprekar Number";
        break;
    }
    else {
        // return "Not Kaprekar Number";
        this.kaprekar = "Not a Kaprekar Number";
    }
}

return this.kaprekar;
}
*/

kap() {
    let n = this.n;
    if (n < 1) {
        return false;
    }

    let square = n * n;
    let squareStr = square.toString();
    let len = squareStr.length;

    let left = parseInt(squareStr.substring(0, Math.floor(len / 2))) || 0;
    let right = parseInt(squareStr.substring(Math.floor(len / 2))) || 0;

    if (left + right === n) {
        return "Kaprekar Number";
    }
}
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        else {
            return "Not a Kaprekar Number";
        }
    }
}

calHash() {
    return c.createHash('sha256').update(this.i + this.t + this.kaprekar +
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) {
        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]
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        if (cb.h !== cb.calHash()) {  
            return false;  
        }  
  
        if (pb.h !== cb.ph) {  
            return false;  
        }  
    }  
  
    return true;  
}  
}  
  
module.exports.Block = Block;  
module.exports.Blockchain = Blockchain;
```

## 2. test.js

```
const { Block, Blockchain } = require('./blockchain');  
let mb = new Blockchain();  
console.log("Developed by Mihir Nagda - 31031523016")  
  
mb.addBlock(new Block(1, new Date(), 45));  
mb.addBlock(new Block(2, new Date(), 19));  
mb.addBlock(new Block(3, new Date(), 297));  
console.log("Chain Valid:", mb.validate());  
console.log(JSON.stringify(mb, null, 3));
```

Name: Mihir Nagda  
Seat No: 31031523016

Output:

```
PS D:\Notes\PG\PG Sem 3\Practicals\Blockchain> node "d:\Notes\PG\PG Sem 3\Practicals\Blockchain\Prac3\test.js"
Developed by Mihir Nagda - 31031523016
Chain Valid: true
{
  "chain": [
    {
      "i": 0,
      "t": "2024-08-17T14:11:11.001Z",
      "n": 0,
      "kaprekar": false,
      "ph": "0",
      "h": "5c36b85975922ea683ca8f829b66fcb4912a5ed4180136d06d6f2925a4cf987a"
    },
    {
      "i": 1,
      "t": "2024-08-17T14:11:11.010Z",
      "n": 45,
      "kaprekar": "Kaprekar Number",
      "ph": "5c36b85975922ea683ca8f829b66fcb4912a5ed4180136d06d6f2925a4cf987a",
      "h": "0f72574672d1ada67947864a2bc07478e4c6b58bce7d121b8d125bade0921af1"
    },
    {
      "i": 2,
      "t": "2024-08-17T14:11:11.010Z",
      "n": 19,
      "kaprekar": "Not a Kaprekar Number",
      "ph": "0f72574672d1ada67947864a2bc07478e4c6b58bce7d121b8d125bade0921af1",
      "h": "1b5d48396bf7ccfcd08192686d9cad7dc26231d3c041bf308ec9721038e488d9"
    },
    {
      "i": 3,
      "t": "2024-08-17T14:11:11.010Z",
      "n": 297,
      "kaprekar": "Kaprekar Number",
      "ph": "1b5d48396bf7ccfcd08192686d9cad7dc26231d3c041bf308ec9721038e488d9",
      "h": "2aa29e81ba7be2d2e122ceedc84bb8533be6a86c24a00c881ffbcf5ed663b303"
    }
  ]
}
```



Name: Mihir Nagda  
Seat No: 31031523016

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.

Code:

1. 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"
    }
  }
}
```

Name: Mihir Nagda  
Seat No: 31031523016

```
    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() {
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        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;
```

Name: Mihir Nagda  
Seat No: 31031523016

## 2. test.js

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

let mb = new Blockchain();
console.log("Developed by Mihir Nagda - 31031523016")

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 D:\Notes\PG\PG Sem 3\Practicals\Blockchain> node "d:\Notes\PG\PG Sem 3\Practicals\Blockchain\Prac4\test.js"
Developed by Mihir Nagda - 31031523016
true
{
  "chain": [
    {
      "i": 0,
      "t": "2024-08-21T15:59:42.217Z",
      "n": "311440db0f0284db5278a9c56212c34b:3592a7cd2d7c6a4c5f9227a3347c3017",
      "automorphic": "Automorphic Number",
      "ph": "0",
      "h": "c8e4131b53a9a0206067b021d757629d437b44d121c662ee3484b17431600441"
    },
    {
      "i": 1,
      "t": "2024-08-21T15:59:42.360Z",
      "n": "1152951b71147c7d8c2ee56f88e74de6:4b03a71c52cf5d30a83488ec645b7fe6",
      "automorphic": "Automorphic Number",
      "ph": "c8e4131b53a9a0206067b021d757629d437b44d121c662ee3484b17431600441",
      "h": "c817d2a4b6a9e89dfb165d25d000e1a899bfaa1ec3b4066e9d35896c5ce0896c"
    },
    {
      "i": 3,
      "t": "2024-08-21T15:59:42.742Z",
      "n": "0e29754b5844da194fc0858c48892c6a:38e6282cf6d9ce00fa11b55c3ba3c8dd",
      "automorphic": "Automorphic Number",
      "ph": "c817d2a4b6a9e89dfb165d25d000e1a899bfaa1ec3b4066e9d35896c5ce0896c",
      "h": "c4de5ef7bbb9956fb557efad2bd93484d65823093312c3e13d798f02595c30d5"
    }
  ]
}
```

Name: Mihir Nagda  
Seat No: 31031523016

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

Code:

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];
  }
}
```

Name: Mihir Nagda  
Seat No: 31031523016

```
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;
}
```

Name: Mihir Nagda  
Seat No: 31031523016

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

test.js:

```
const { Blockchain, Block } = require('./blockchain')  
  
let mb;  
console.log("Developed by Mihir Nagda - 31031523016")  
  
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  
Charcter - Exit: ', resolve);  
    });  
  
    if (DW.toUpperCase() !== 'D' && DW.toUpperCase() !== 'W') {  
      console.log('Exiting...');  
      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++;  
  }  
}
```

Name: Mihir Nagda  
Seat No: 31031523016

```
}  
  
    readline.close();  
}  
  
main();
```

## Output:

```
PS D:\Notes\PG\PG Sem 3\Practicals\Blockchain> node "d:\Notes\PG\PG Sem 3\Practicals\Blockchain\Prac5\test.js"  
Developed by Mihir Nagda - 31031523016  
Type D - Deposit, W - Withdraw or Any Other Charcter - Exit: D  
Enter Amount: 1500  
{  
  "chain": [  
    {  
      "i": 0,  
      "t": "2024-08-22T16:19:26.041Z",  
      "op": "D",  
      "m": 1000,  
      "b": 1000,  
      "ph": "0",  
      "h": "0e82c2cd8572f98b4b9d7440ef48cb185aeabebc262e8525946fe12a8b582b51"  
    },  
    {  
      "i": 1,  
      "t": "2024-08-22T16:19:37.673Z",  
      "op": "D",  
      "m": 1500,  
      "b": 2500,  
      "ph": "0e82c2cd8572f98b4b9d7440ef48cb185aeabebc262e8525946fe12a8b582b51",  
      "h": "32213e7be2602b1086224e93c228f7523f3c6f05c67d1d724990e3fdbf58df42"  
    }  
  ]  
}  
Type D - Deposit, W - Withdraw or Any Other Charcter - Exit: W  
Enter Amount: 289
```


```
Type D - Deposit, W - Withdraw or Any Other Charcter - Exit: W  
Enter Amount: 289  
{  
  "chain": [  
    {  
      "i": 0,  
      "t": "2024-08-22T16:19:26.041Z",  
      "op": "D",  
      "m": 1000,  
      "b": 1000,  
      "ph": "0",  
      "h": "0e82c2cd8572f98b4b9d7440ef48cb185aeabebc262e8525946fe12a8b582b51"  
    },  
    {  
      "i": 1,  
      "t": "2024-08-22T16:19:37.673Z",  
      "op": "D",  
      "m": 1500,  
      "b": 2500,  
      "ph": "0e82c2cd8572f98b4b9d7440ef48cb185aeabebc262e8525946fe12a8b582b51",  
      "h": "32213e7be2602b1086224e93c228f7523f3c6f05c67d1d724990e3fdbf58df42"  
    },  
    {  
      "i": 2,  
      "t": "2024-08-22T16:19:45.614Z",  
      "op": "W",  
      "m": 289,  
      "b": 2211,  
      "ph": "32213e7be2602b1086224e93c228f7523f3c6f05c67d1d724990e3fdbf58df42",  
      "h": "786528ab7ce417bade4358042386a7c17572585301e4c5f9699406f2974c1510"  
    }  
  ]  
}  
Type D - Deposit, W - Withdraw or Any Other Charcter - Exit: E  
Exiting...
```



Name: Mihir Nagda  
Seat No: 31031523016

Practical 6A: Create a smart contract for addition of two numbers.

Code:

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.0;
4
5 contract AddNumbers {
6     function add(uint256 a, uint b) public pure returns (uint256) {  infinite gas
7         return a + b;
8     }
9 }
```

Output:



Name: Mihir Nagda  
Seat No: 31031523016

Practical 6B: Write a simple auction contract where a user can bid on an item and the highest bidder wins.

Code:

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.0;
4
5 contract auction {
6     address public o;
7     address public hba;
8     uint public hb;
9     bool public ae;
10
11     constructor(){ 434165 gas 380200 gas
12         o = msg.sender;
13         hb = 0;
14         ae = false;
15     }
16
17     function bid(uint256 a) public payable { 51064 gas
18         require(!ae, "Auction has ended!");
19         require(a > hb, "Bid must be greater than current value.");
20         hb = a;
21         hba = msg.sender;
22     }
23
24     function endA() public { 26600 gas
25         require(msg.sender == o, "Only owner can end.");
26         ae = true;
27     }
28
29     function withdraw() public view returns (uint256){ 4608 gas
30         require(msg.sender == hba, "Only highest bidder can withdraw.");
31         // payable(hba).transfer(hb);
32         return hb;
33     }
34 }
```

Output:

Deployed/Unpinned Contracts

▼ AUCTION AT 0XB27...07C2C {

Balance: 0 ETH

**bid** 5 ▼

**endA**

**ae**  
0: bool: false

**hb**  
0: uint256: 5

**hba**  
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

**o**  
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

**withdraw**  
0: uint256: 5

Name: Mihir Nagda  
Seat No: 3103152016

Practical 7A: Write a smart contract to display factorial of a number.

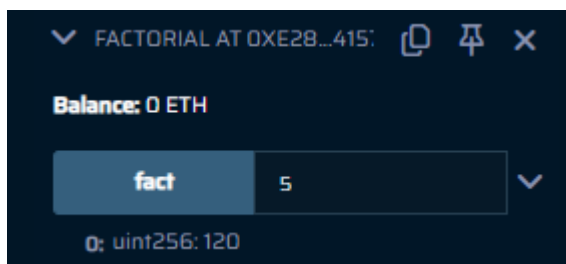
Code:

```
//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:



Name: Mihir Nagda  
Seat No: 3103152016

Practical 7B: Write a smart contract to display  $n^{\text{th}}$  term of Fibonacci series.

Code:

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

pragma solidity ^0.8.0;

contract fibo{
    function fib(uint256 a) public pure returns(uint256){
        uint256 n1 = 0;
        uint256 n2 = 1;
        uint256 n3;

        for (uint256 i = 3; i <= a; i++){
            n3 = n1 + n2;
            n1 = n2;
            n2 = n3;
        }
        return n3;
    }
}
```

Output:



Name: Mihir Nagda  
Seat No: 3103152016

Practical 7C: Write a smart contract to check if the number is prime or not.

Code:

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

pragma solidity ^0.8.0;

contract prime{
    function primee(uint256 a) public pure returns(bool){
        for (uint256 i = 2; i < a; i++){
            if(a % i == 0){
                return false;
            }
        }
        return true;
    }
}
```

Output:



Name: Mihir Nagda  
Seat No: 3103152016

Practical 7D: Write a smart contract to deposit and withdraw money.

Code:

```
//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;
    }
}
```

Name: Mihir Nagda  
Seat No: 3103152016

Output:

▼

BANK AT 0X93F...C96CC (M)

×

Balance: 0 ETH

deposit

5000

▼

withdraw

uint256 a

▼

bal

0: uint256: 5000

displayOwn

0: address: 0x5B38Da6a701c568545d  
CfcB03FcB875f56beddC4

s

0: address: 0x5B38Da6a701c568545d  
CfcB03FcB875f56beddC4

▼

BANK AT 0X93F...C96CC (M)

×

Balance: 0 ETH

deposit

5000

▼

withdraw

3746

▼

bal

0: uint256: 1254

displayOwn

0: address: 0x5B38Da6a701c568545d  
CfcB03FcB875f56beddC4

s

0: address: 0x5B38Da6a701c568545d  
CfcB03FcB875f56beddC4

Name: Mihir Nagda  
Seat No: 31031523016

Practical 8A: Create a smart contract to calculate mean of n numbers.

Code:

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

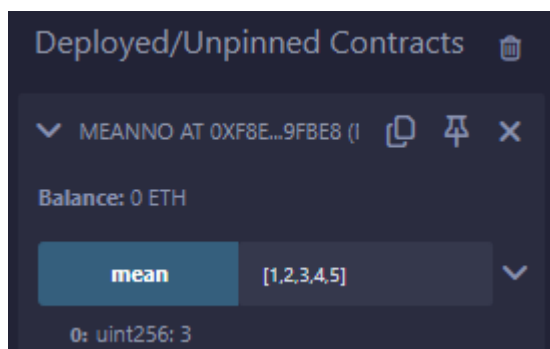
pragma solidity ^0.8.0;

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:





Name: Mihir Nagda  
Seat No: 31031523016

Practical 8B: Create a smart contract to calculate median of n numbers.

Code:

```
// 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:



Name: Mihir Nagda  
Seat No: 31031523016

Practical 8C: 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.

Code:

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

pragma solidity ^0.8.0;

contract studentC{
    struct std{
        uint sid;
        string name;
        address addr;
        uint BCT;
        uint ML;
        uint BDA;
        uint percent;
        string grade;
    }

    mapping(uint256 => std) public student;

    function register(uint i, string memory n, uint b, uint m, uint a) public {
        student[i].name = n;
        student[i].addr = msg.sender;
        student[i].sid = i;
        student[i].BCT = b;
        student[i].ML = m;
        student[i].BDA = a;

        uint p = (b + m + a) / 3;
        student[i].percent = p;

        if(p >= 80){
            student[i].grade = "O";
        }
        else if(p >= 60 && p < 80){
            student[i].grade = "A";
        }
    }
}
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        else if(p >= 40 && p < 60){  
            student[i].grade = "B";  
        }  
        else{  
            student[i].grade = "F";  
        }  
    }  
}
```

Output:

The screenshot shows a web application interface with a dark theme. At the top, there's a header bar with a dropdown menu showing 'STUDENTC AT 0X5E1...4EFF5' and icons for copy, pin, and close. Below the header, the balance is shown as 'Balance: 0 ETH'. There are two main buttons: 'register' (orange) and 'student' (blue). The 'register' button is currently selected, showing a dropdown menu with the text '1, Rupin, 95, 56, 83'. The 'student' button is also visible, showing a dropdown menu with the text '1'. Below these buttons, there is a list of student records. The first record is expanded, showing the following details:

- 0: uint256: sid 1
- 1: string: name Rupin
- 2: address: addr 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
- 3: uint256: BCT 95
- 4: uint256: ML 56
- 5: uint256: BDA 83
- 6: uint256: percent 78
- 7: string: grade A

Name: Mihir Nagda  
Seat No: 31031523016

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

Code:

```
//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;
```

Name: Mihir Nagda  
Seat No: 31031523016

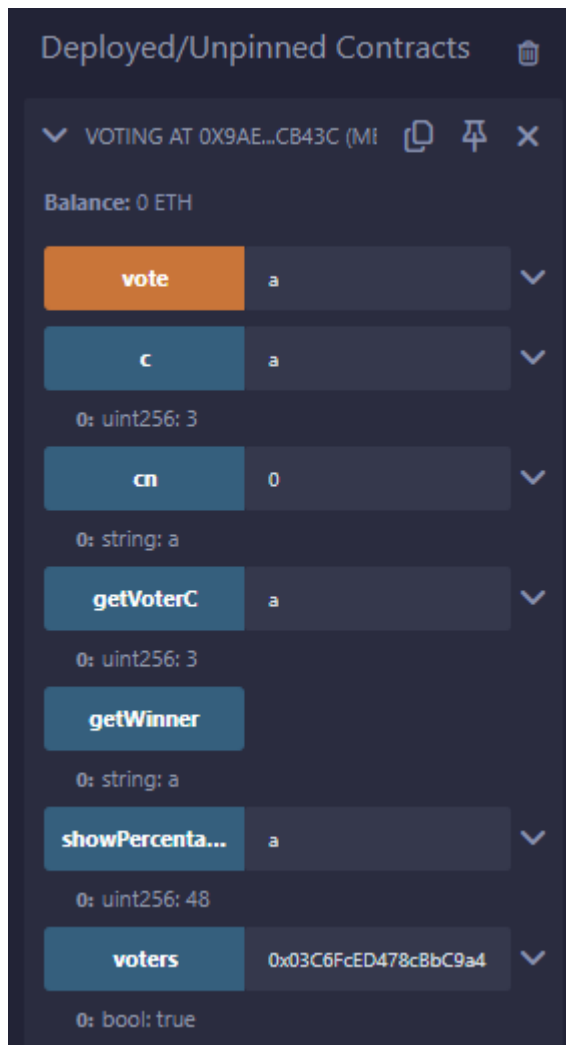
```
uint256 temp = 0;
for (uint256 j = 0; j < cn.length; j++) {
    if (getVoterC(cn[j]) > temp) {
        temp = getVoterC(cn[j]);
        winner = cn[j];
    }
}
return winner;
}

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;
}
}
```

Name: Mihir Nagda  
Seat No: 31031523016

Output:



Name: Mihir Nagda  
Seat No: 31031523016

Practical 10A: Write a smart contract for Single Level Inheritance.

Code:

```
//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:

```
decoded output      {
                    "0": "string: Iqlipse Nova",
                    "1": "string[2]: Khwab,Sajke"
                    }  
```

Name: Mihir Nagda  
Seat No: 31031523016

Practical 10B: Write a smart contract for Multi-Level Inheritance.

Code:

```
//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);
    }
}
```



Name: Mihir Nagda  
Seat No: 31031523016

Output:

Deployed/Unpinned Contracts

EXAM AT 0X540...C7569 (MEM)

Balance: 0 ETH

setCollege

SVU, CS

setMarks

[95, 84, 79, 65, 98]

setStudent

Mihir, 101

getDetails

0: string: SVU

1: string: CS

2: string: Mihir

3: uint256: 101

4: uint256: 84

Name: Mihir Nagda  
Seat No: 31031523016

Practical 10C: Write a smart contract for Multiple Inheritance.

Code:

```
//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();
}
```

Name: Mihir Nagda  
Seat No: 31031523016

```
function tInherit() public returns(string memory, uint, uint, string memory,
uint){
    s.setE("Mihir Nagda", 101, 15000);
    s.setD("CS");
    s.calHRA();
    return s.getVal();
}
}
```

Output:

decoded output

```
{
  "0": "string: Mihir Nagda",
  "1": "uint256: 101",
  "2": "uint256: 15000",
  "3": "string: CS",
  "4": "uint256: 5000"
}
```

Name: Mihir Nagda  
Seat No: 31031523016

Practical 10D: Write a smart contract for Hierarchical Inheritance.

Code:

```
//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();
}
```

Name: Mihir Nagda  
Seat No: 31031523016

```
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:

```
decoded output {
    "0": "uint256: 4",
    "1": "string: White",
    "2": "string: Goldie",
    "3": "string: Indie"
}
```

dInherit:

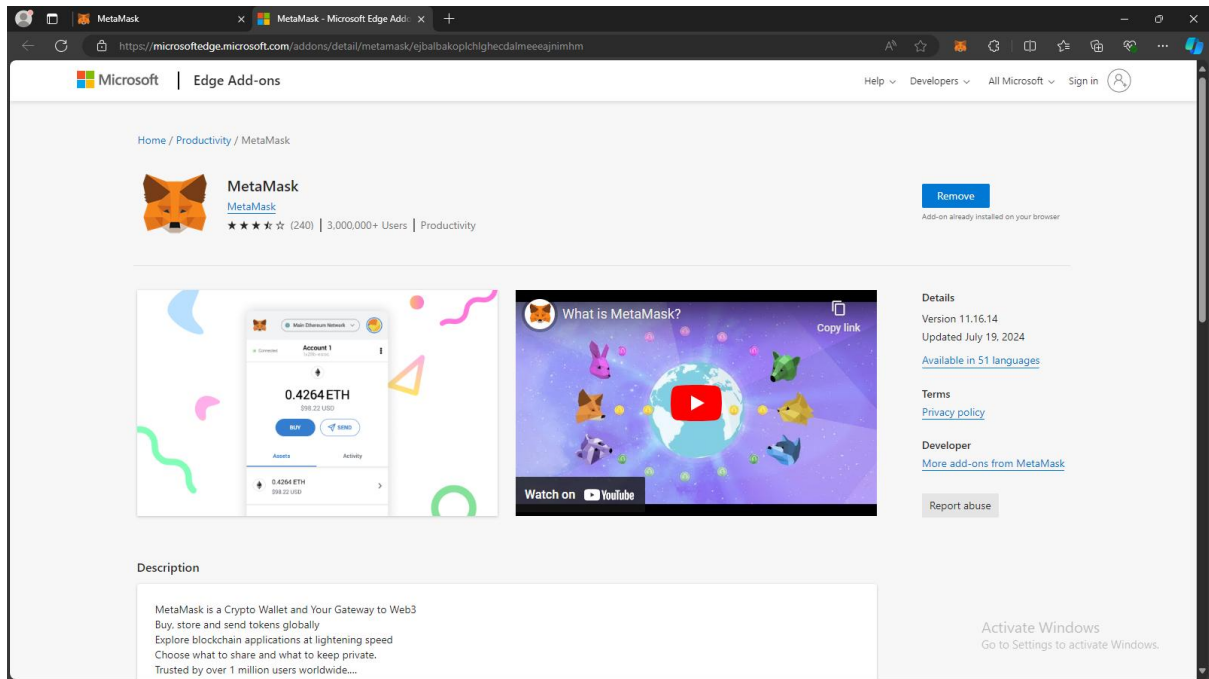
```
decoded output {
    "0": "uint256: 4",
    "1": "string: Brown",
    "2": "string: Tom",
    "3": "string: Labrador"
}
```

Name: Mihir Nagda  
Seat No: 31031523016

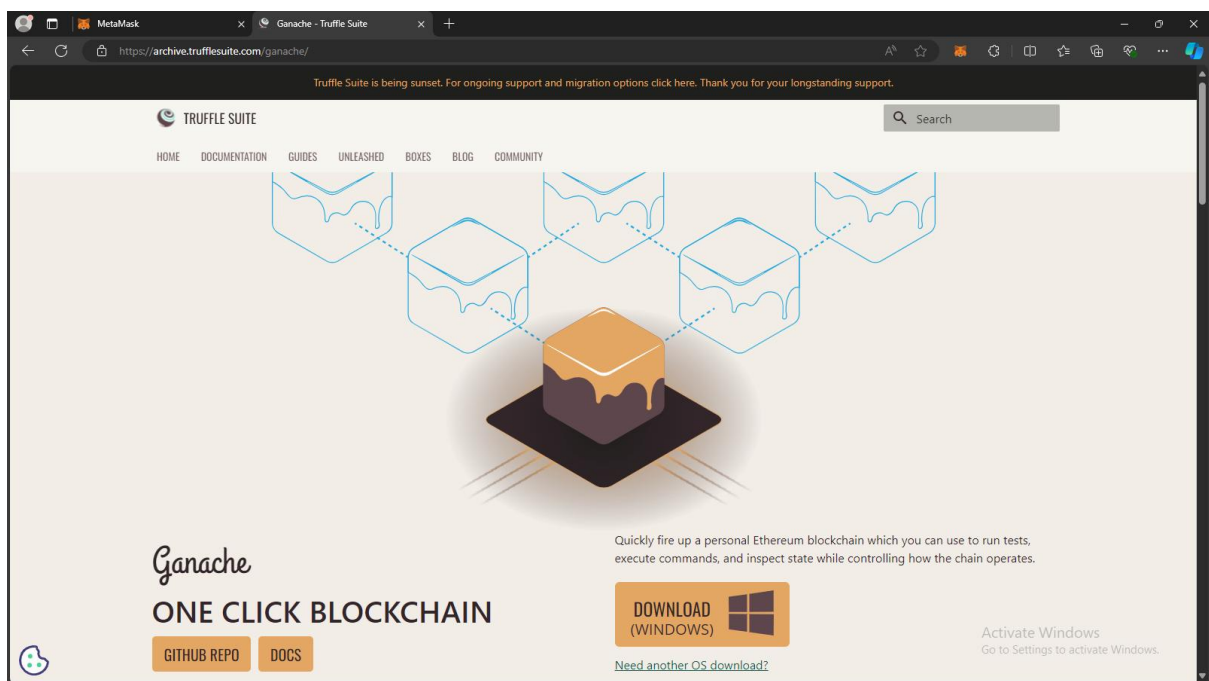
## Practical 11: Creating a simple DApp for performing basic mathematical operations on two numbers.

### Part 1: Setting up MetaMask and Ganache.

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

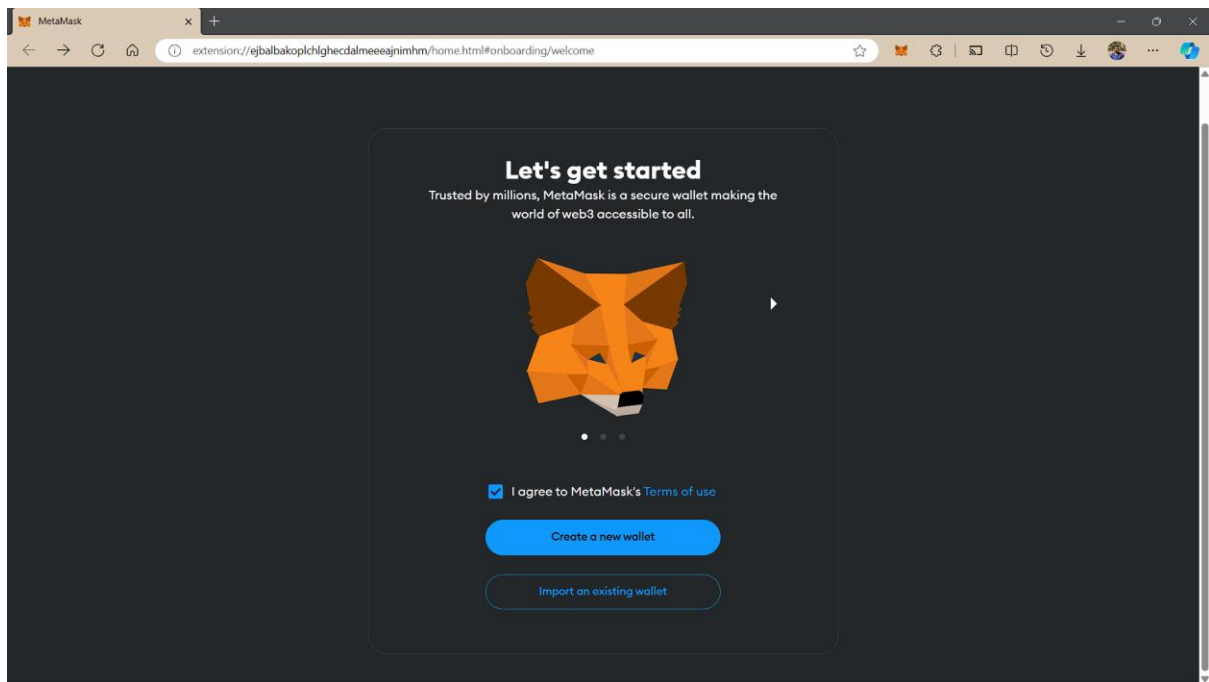


#### 2. Install Ganache from [here](#).

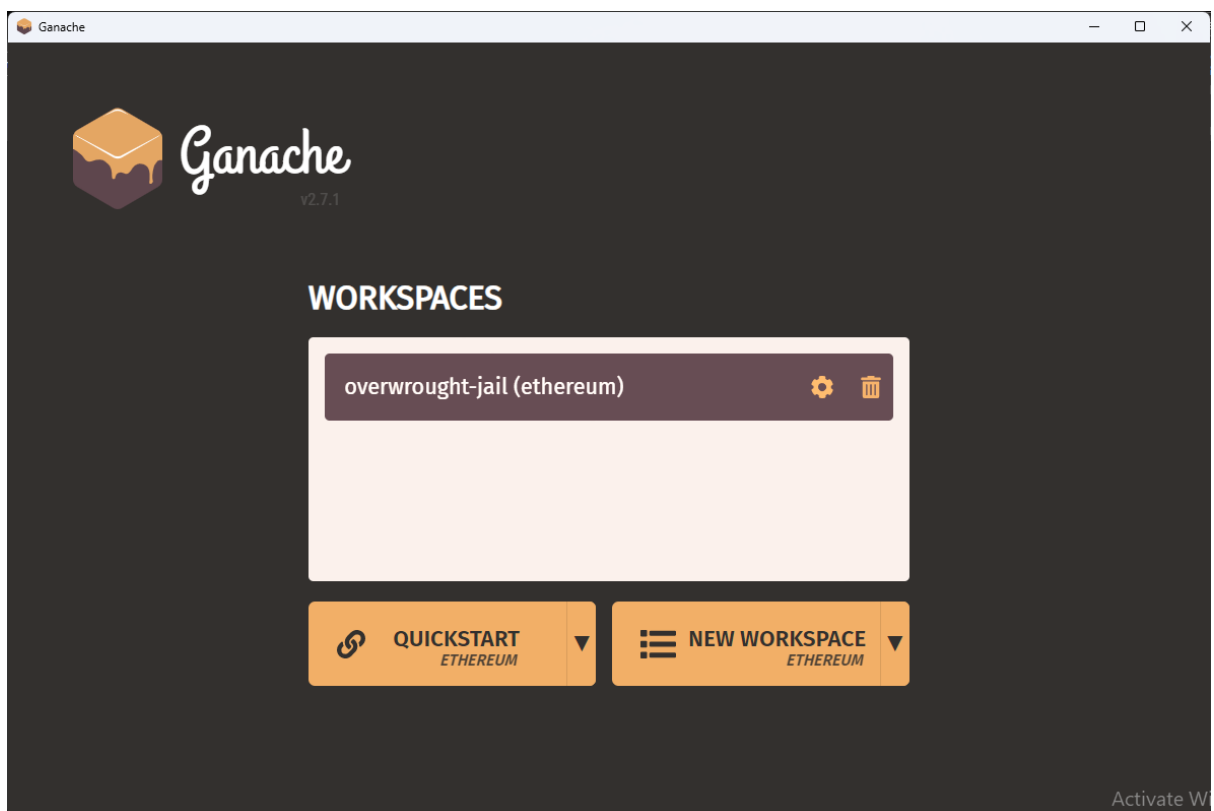


Name: Mihir Nagda  
Seat No: 31031523016

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

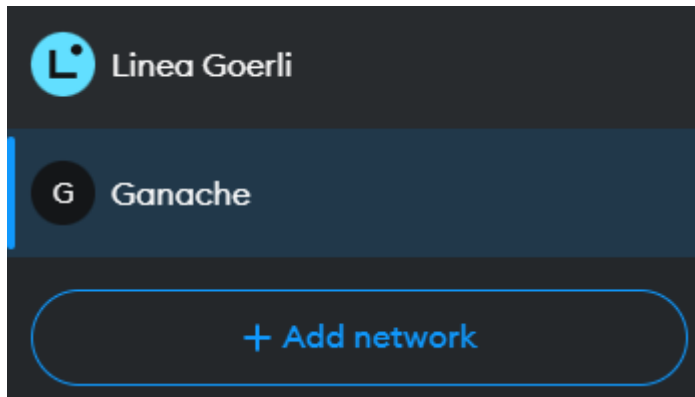


4. Start a new workspace in Ganache.



Name: Mihir Nagda  
Seat No: 31031523016

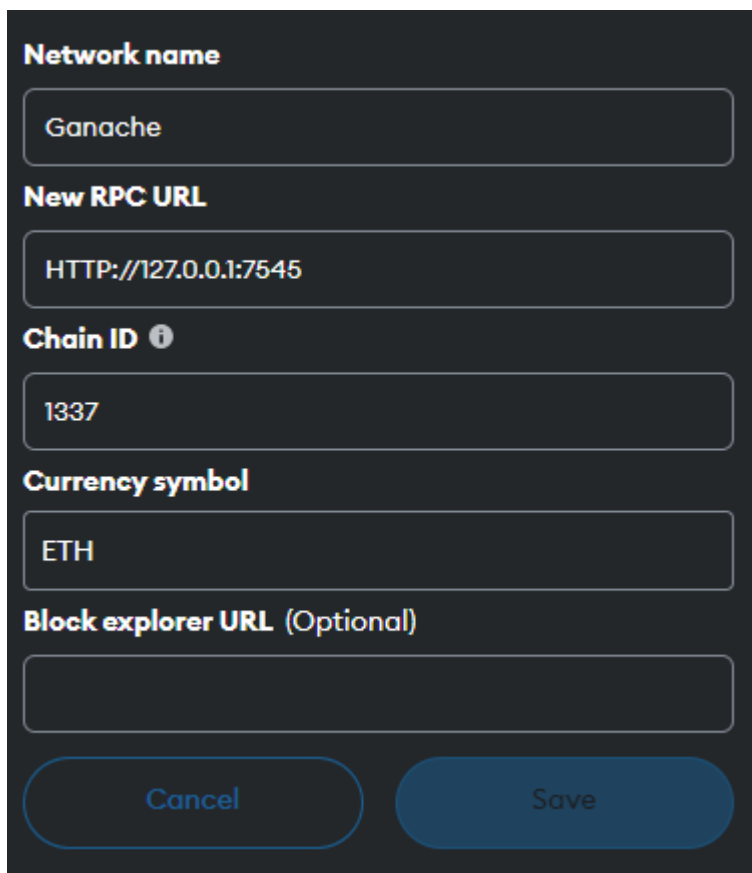
5. Go to MetaMask. Click on the Network Name → Add Network → Add Network Manually.



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.

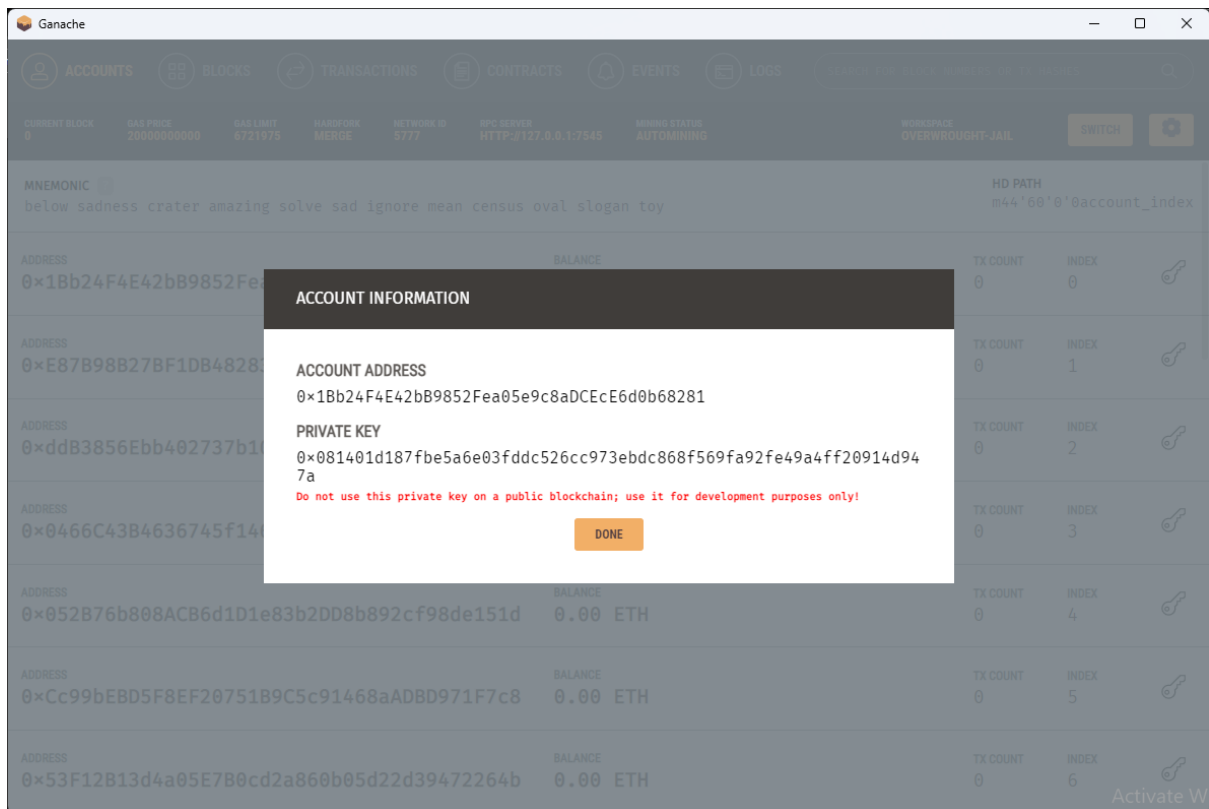
8. Give the currency symbol as ETH. Save the network.

A screenshot of the 'Add Network Manually' form in the MetaMask application. The form is set against a dark background with light-colored text and input fields. It includes the following sections and inputs: 'Network name' with the value 'Ganache'; 'New RPC URL' with the value 'HTTP://127.0.0.1:7545'; 'Chain ID' with a small information icon and the value '1337'; 'Currency symbol' with the value 'ETH'; and 'Block explorer URL (Optional)' which is currently empty. At the bottom of the form are two rounded buttons: 'Cancel' and 'Save'.

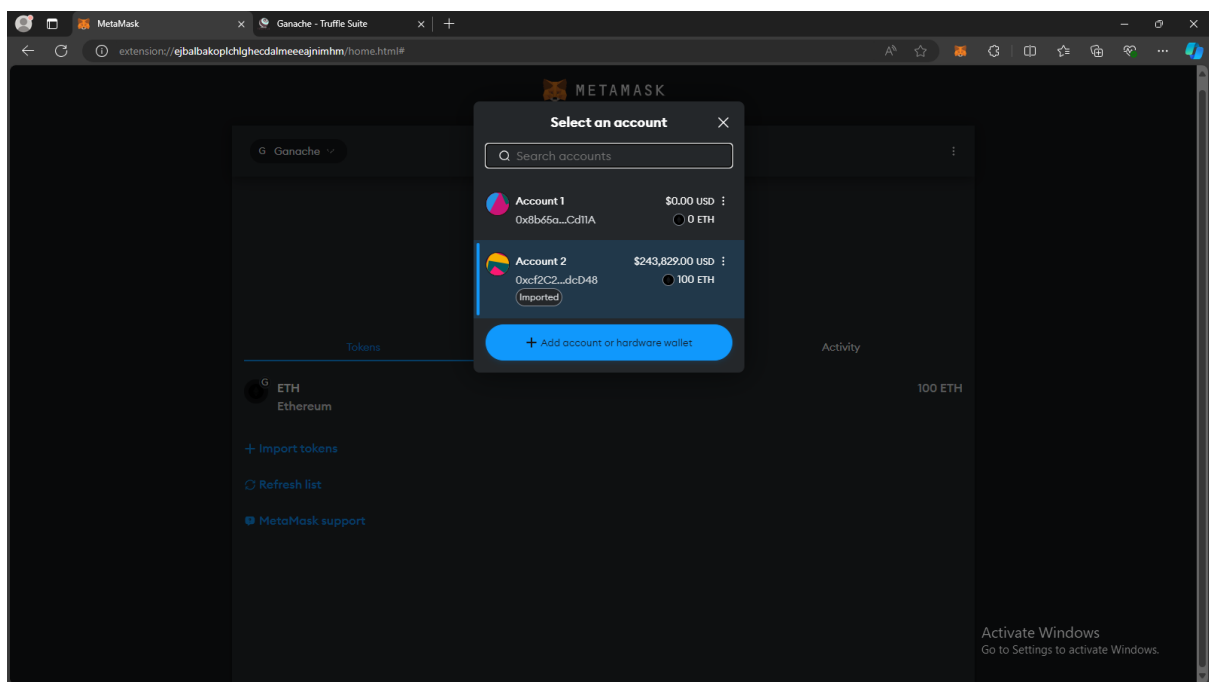


Name: Mihir Nagda  
Seat No: 31031523016

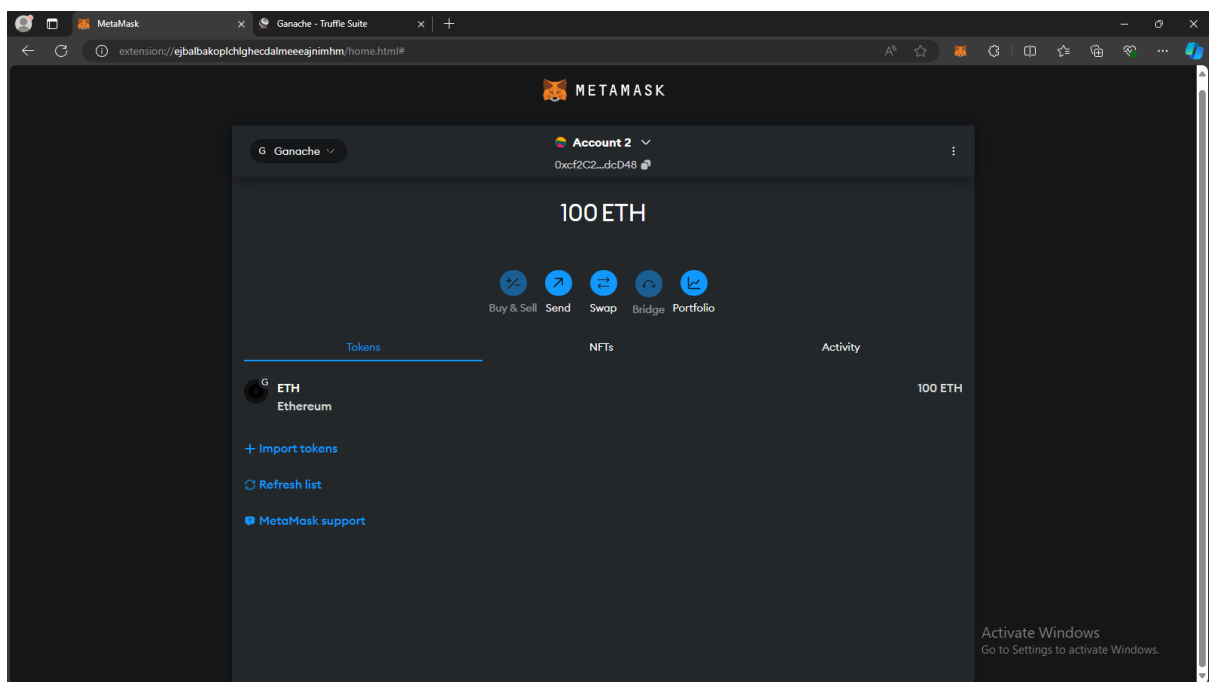
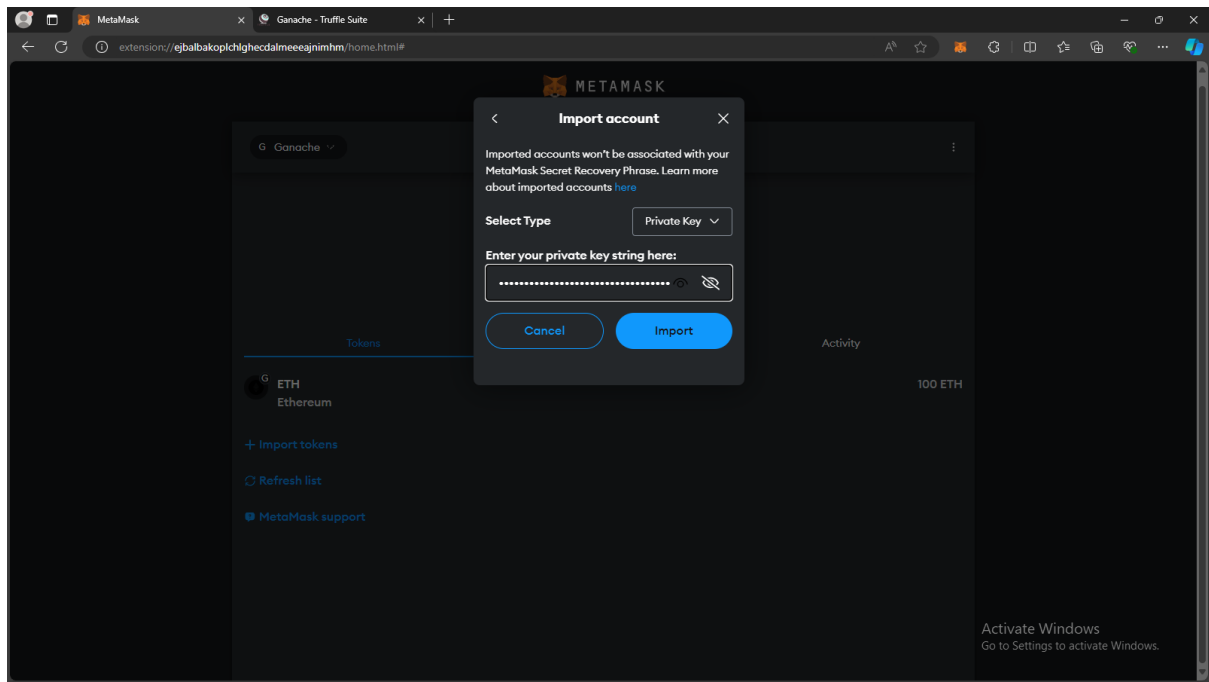
9. Go to Ganache and Click on the key symbol. Copy the private key.



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: Mihir Nagda  
Seat No: 31031523016



Name: Mihir Nagda  
Seat No: 31031523016

## 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;
    }
}
```

Name: Mihir Nagda  
Seat No: 31031523016

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.");
  }
})
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        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);
```

Name: Mihir Nagda  
Seat No: 31031523016

```
    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.

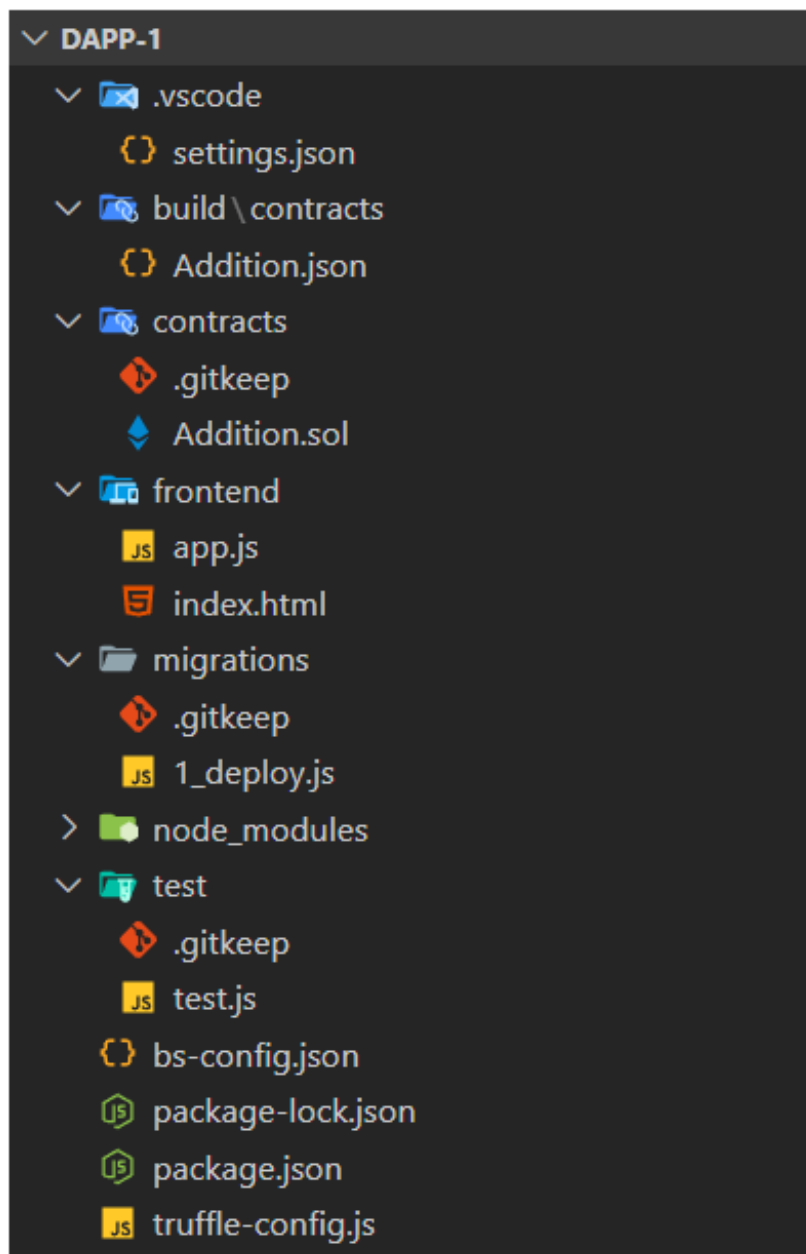
```
module.exports = {  
  networks: {  
    development: {  
      host: "127.0.0.1",  
      port: 7545,  
      network_id: "5777",  
    },  
  },  
  
  // Configure your compilers  
  compilers: {  
    solc: {  
      version: "0.8.19"  
    }  
  }  
};
```

Name: Mihir Nagda  
Seat No: 31031523016

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

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

d. The final directory structure should look like this



Name: Mihir Nagda  
Seat No: 31031523016

### 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",
},
];
```



Name: Mihir Nagda  
Seat No: 31031523016

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--19664-8nJHijHokWzf
> 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)
```

Name: Mihir Nagda  
Seat No: 31031523016

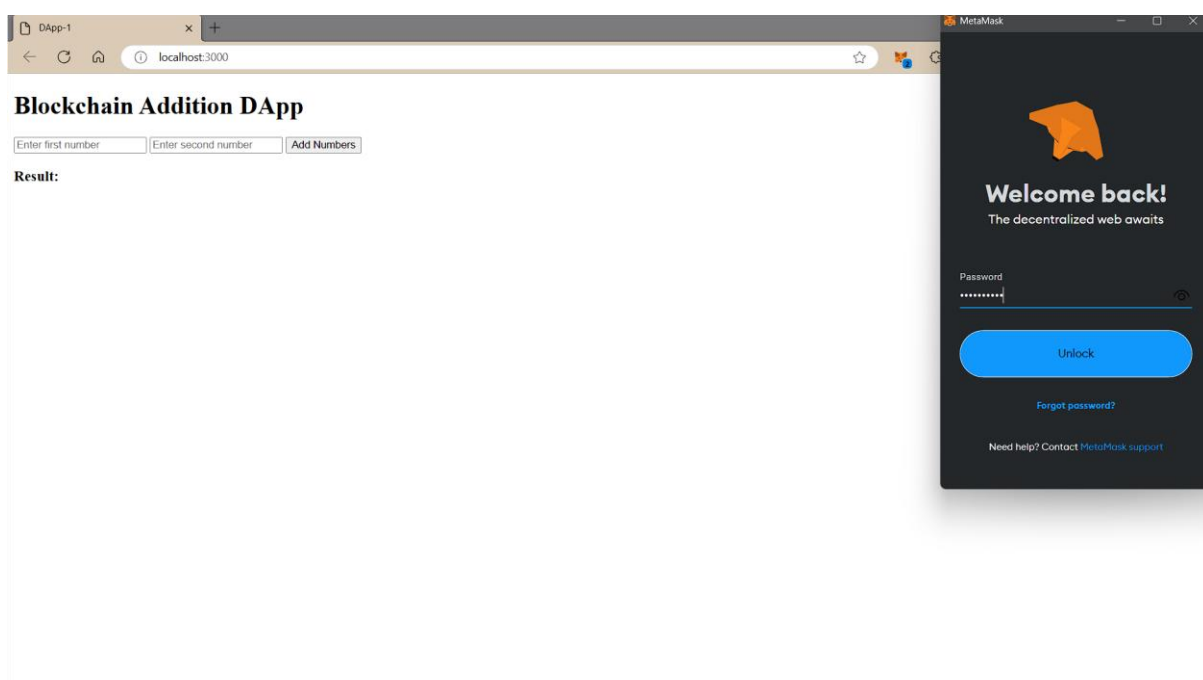
6. Run `npm start` if everything is correct.

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

> start
> lite-server

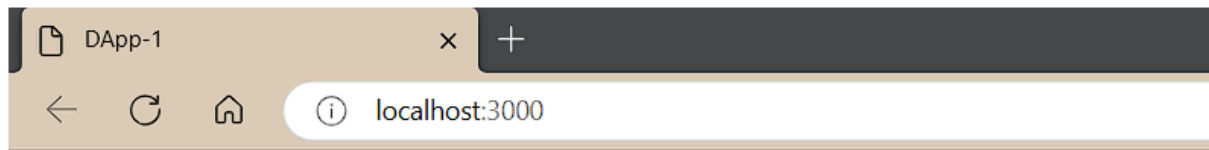
** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.html,htm,css,js' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: [ './frontend' ],
    middleware: [ [Function (anonymous)], [Function (anonymous)] ]
  }
}
[Browsersync] Access URLs:
-----
    Local: http://localhost:3000
  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.



Name: Mihir Nagda  
Seat No: 31031523016

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



## Blockchain Addition DApp

<input type="text" value="45"/>	<input type="text" value="98"/>	<input type="button" value="Add Numbers"/>
---------------------------------	---------------------------------	--

**Result: 143**

Name: Mihir Nagda  
Seat No: 31031523016

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

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

```
// 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>
```

Name: Mihir Nagda  
Seat No: 31031523016

```
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.");
  }
})
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        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(num1);
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        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);
};
```

Name: Mihir Nagda  
Seat No: 31031523016

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");
  });
});
```



Name: Mihir Nagda  
Seat No: 31031523016

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

DApp-1

×

+

←

↻

🏠

i

localhost:3000

# Blockchain Operations DApp

Number 1:

Number 2:

## Choose Operation:

Add

Sub

Mul

Div

## Result:

**Addition: 522**

**Subtraction: 350**

**Multiplication: 37496**

**Division: 5**

Name: Mihir Nagda  
Seat No: 31031523016

Practical 12A: 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;
        }
    }
}
```

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

`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-2</title>
  </head>
  <body>
    <h1>Blockchain Factorial DApp</h1>
    Number:
```

Name: Mihir Nagda  
Seat No: 31031523016

```
<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
```

Name: Mihir Nagda  
Seat No: 31031523016

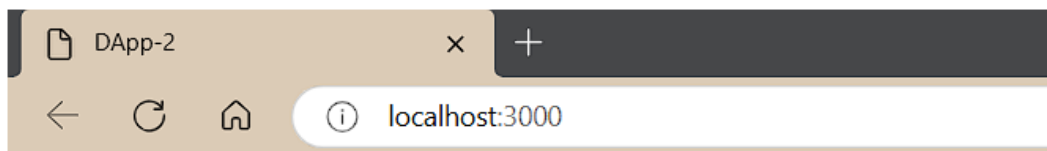
```
.fact(num)
.call({ from: accounts[0] })
.then((result) => {
  console.log(result);
  document.getElementById("result").innerText = `${result}`;
});
}
```

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**

Name: Mihir Nagda  
Seat No: 31031523016

Practical 12B: 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);
```

Name: Mihir Nagda  
Seat No: 31031523016

```
    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) => {
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        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);
    }
}
```

### 4. 1\_deploy.js

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

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

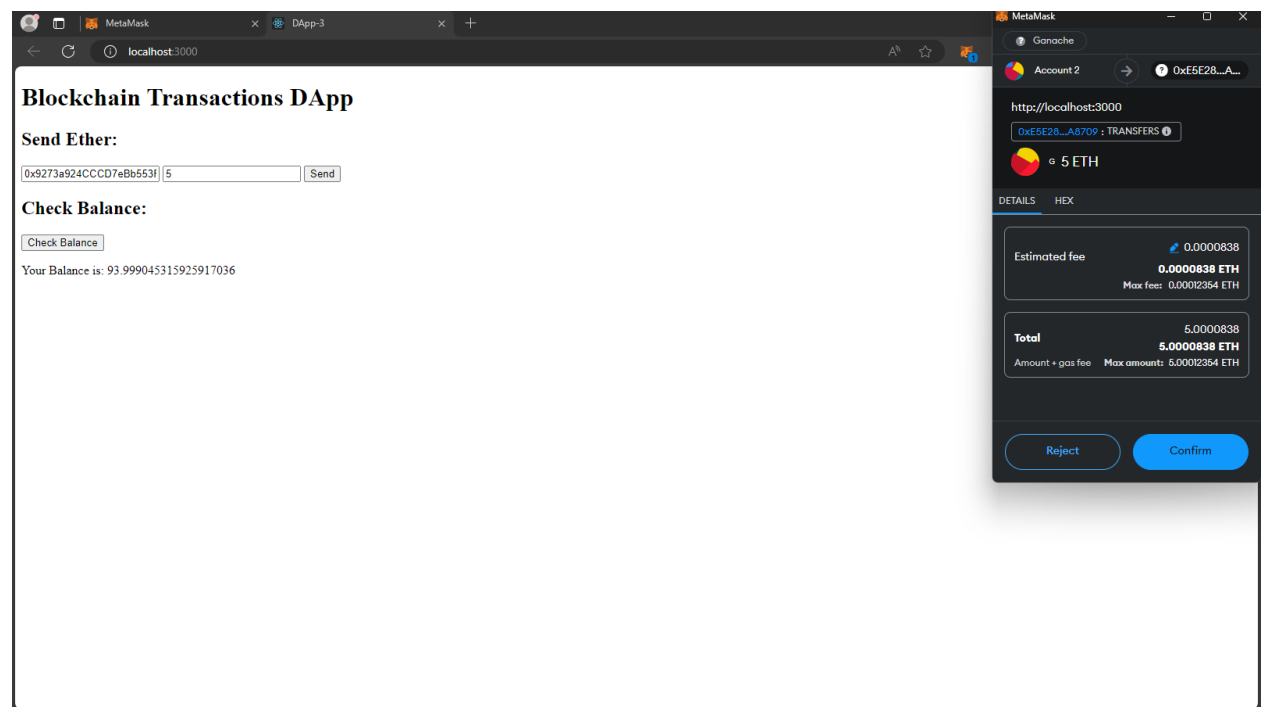
Name: Mihir Nagda  
Seat No: 31031523016

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

## 5. bs-config.json

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

## 6. Output:



ADDRESS	BALANCE
0x8d4e6F53AeEc3698af5ba4b3012257CfECEed938	89.00 ETH
ADDRESS	BALANCE
0x9273a924CCCD7eBb553FB588790423C9a832E00d	106.00 ETH



Name: Mihir Nagda  
Seat No: 31031523016

Practical 12C: 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;
```

Name: Mihir Nagda  
Seat No: 31031523016

```
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}`;
    });
}
```

Name: Mihir Nagda  
Seat No: 31031523016

### 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;
```

Name: Mihir Nagda  
Seat No: 31031523016

```
        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;
    }
}
```

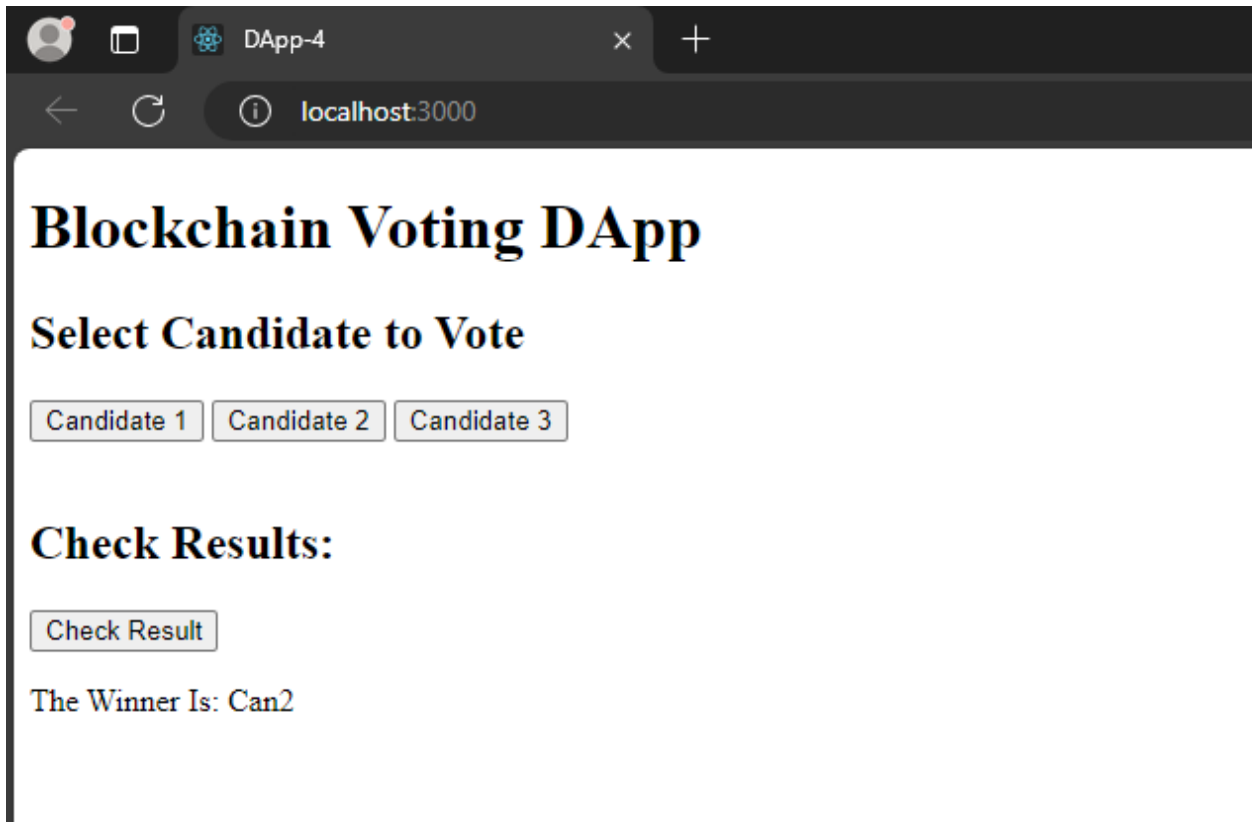
#### 4. 1\_deploy.js

```
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);
};
```

Name: Mihir Nagda  
Seat No: 31031523016
























5. Output:



Name: Mihir Nagda  
Seat No: 31031523016

## Practical 13: Storing and Retrieving files using IPFS.

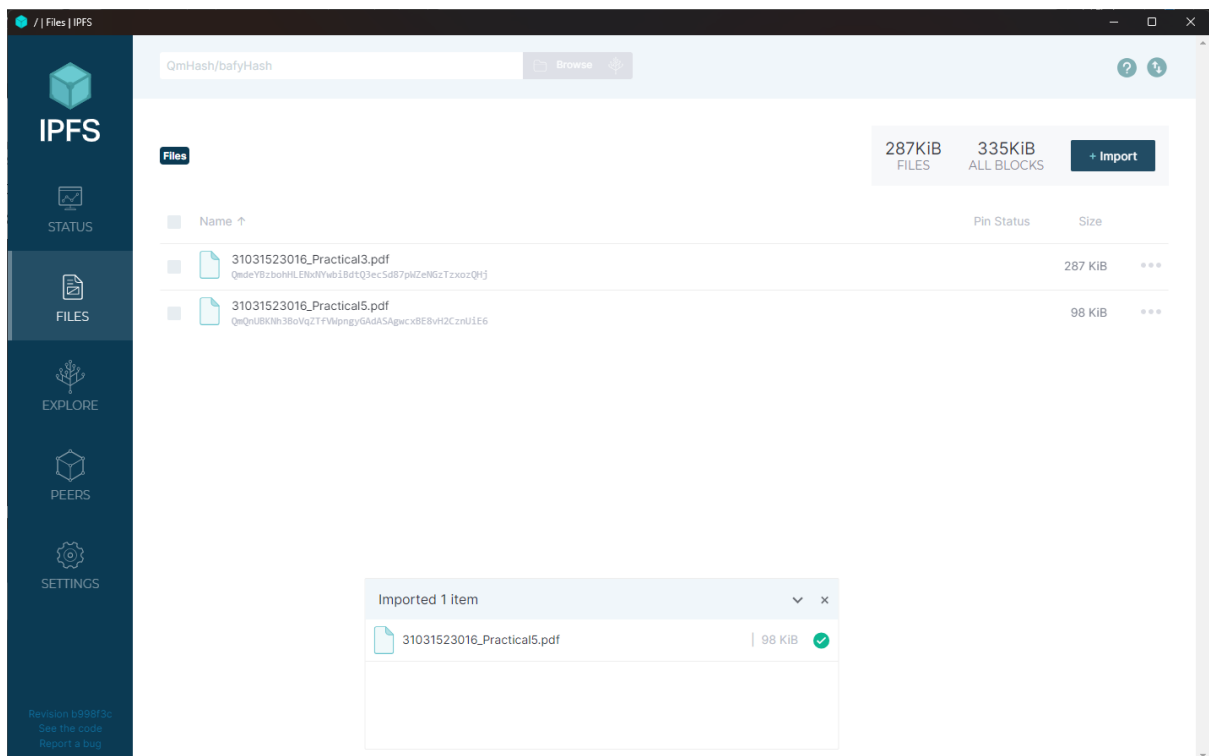
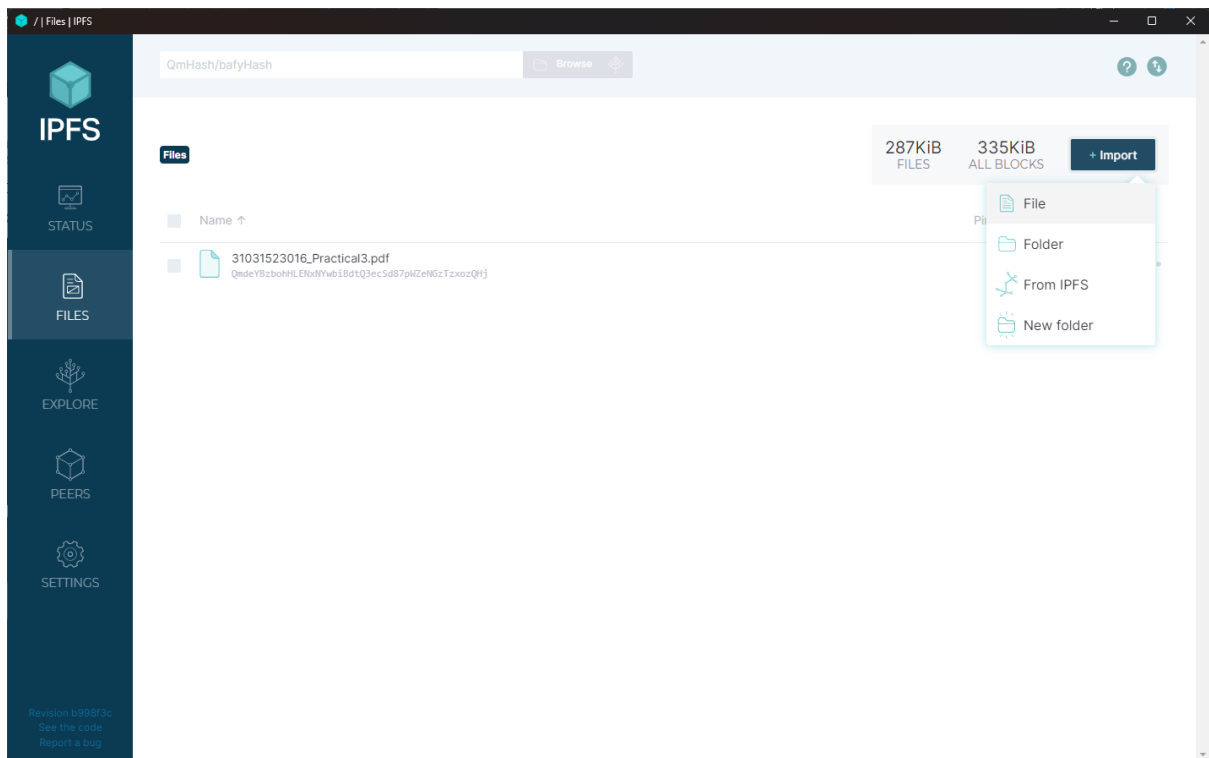
Step 1: Download and Install IPFS Desktop from [here](#).

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
 4  2  3  3  3  2 5 people reacted		



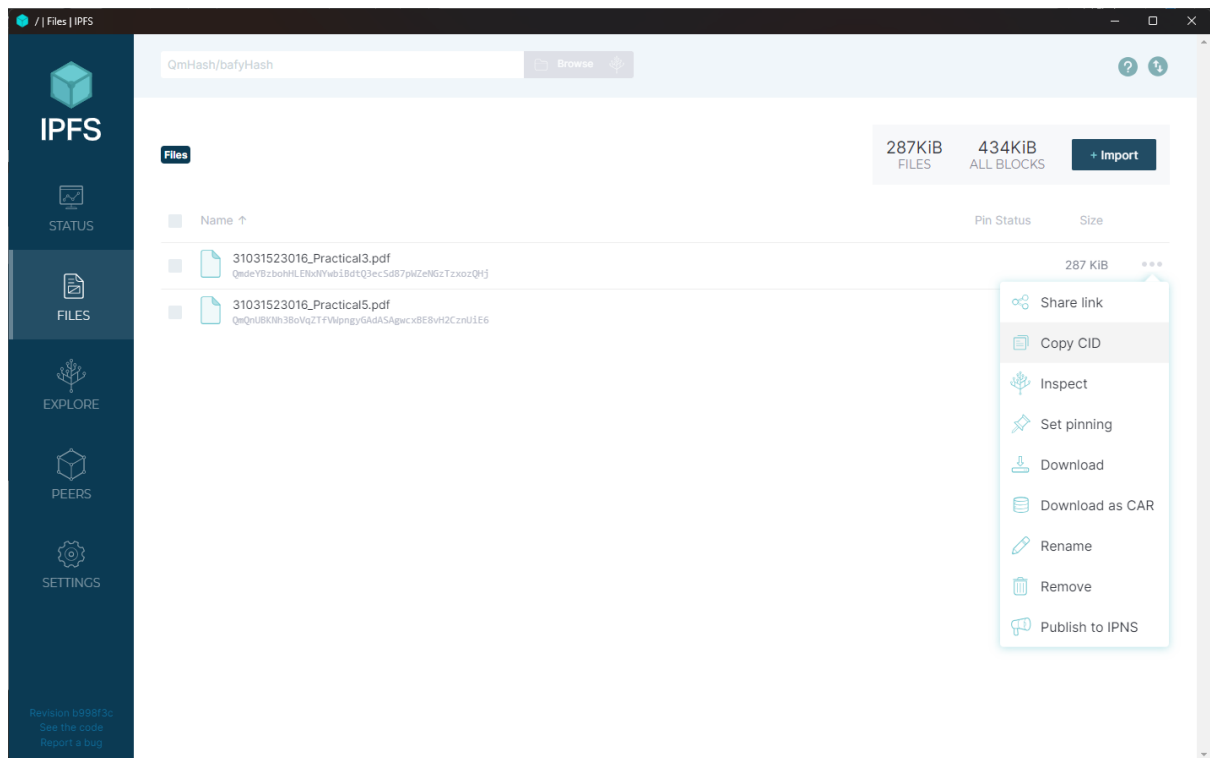
Name: Mihir Nagda  
Seat No: 31031523016

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



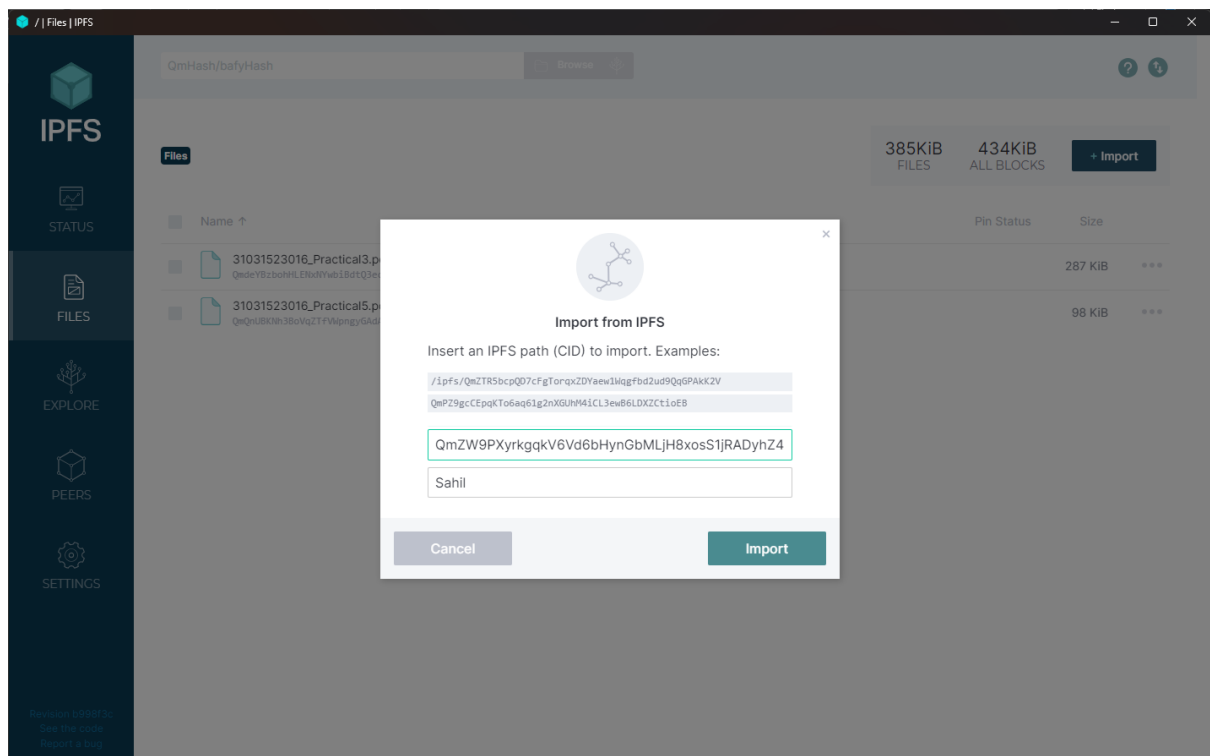
Name: Mihir Nagda  
Seat No: 31031523016

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





Name: Mihir Nagda  
Seat No: 31031523016

Step 6: The imported file will be visible.

