

Activity 1 — Mini-Blockchain

Name: Amish Pandya

Student ID: 16369192

[blockchain.js](#) code

```
// blockchain.js
// Mini blockchain with Proof-of-Work (difficulty=3 by default) and
validation.
// Run: node blockchain.js [difficulty]
// Example: node blockchain.js 3

'use strict';
const crypto = require('crypto');

// ---- helpers -----
/** Stable JSON stringify (sorts object keys for deterministic hashing)
 */
function stableStringify(value) {
  if (value === null || typeof value !== 'object') return
JSON.stringify(value);
  if (Array.isArray(value)) return
`${value.map(stableStringify).join(',')}`;
  const keys = Object.keys(value).sort();
  return `${keys.map(k =>
JSON.stringify(k)+':'+stableStringify(value[k])).join(',')}`;
}

/** SHA-256 hex */
function sha256(s) {
  return crypto.createHash('sha256').update(s).digest('hex');
}

// ---- core classes -----
class Block {
  /**
   * @param {number} index
   * @param {string} timestamp - ISO string is recommended (e.g., new
Date().toISOString())
   * @param {any} data - transaction payload (object/array/primitive)
   * @param {string} previousHash
   */
  constructor(index, timestamp, data, previousHash = '') {
```

```

    this.index = index;
    this.timestamp = timestamp;
    this.data = data;
    this.previousHash = previousHash;
    this.nonce = 0;
    this.hash = this.calculateHash();
}

/** Compute SHA-256 over the block's contents */
calculateHash() {
    const payload =
        String(this.index) +
        this.timestamp +
        stableStringify(this.data) +
        this.previousHash +
        String(this.nonce);
    return sha256(payload);
}

/** Proof-of-Work: find a hash starting with N leading zeros */
mineBlock(difficulty) {
    const target = '0'.repeat(difficulty);
    const start = Date.now();
    while (this.hash.substring(0, difficulty) !== target) {
        this.nonce++;
        this.hash = this.calculateHash();
    }
    const ms = Date.now() - start;
    console.log(`✅ Block mined (idx=${this.index},
nonce=${this.nonce}, ${ms}ms): ${this.hash}`);
}

class Blockchain {
    constructor(difficulty = 3) {
        this.chain = [this.createGenesisBlock()];
        this.difficulty = difficulty;
    }

    createGenesisBlock() {
        return new Block(0, new Date().toISOString(), 'Genesis Block',
'0');
    }
}

```

```

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

/** Add a new block: link prev hash, mine, then append */
addBlock(newBlock) {
  newBlock.previousHash = this.getLatestBlock().hash;
  newBlock.mineBlock(this.difficulty);
  this.chain.push(newBlock);
}

/** Verify integrity: hash consistency + correct previousHash links
 */
isChainValid() {
  for (let i = 1; i < this.chain.length; i++) {
    const current = this.chain[i];
    const previous = this.chain[i - 1];

    // recompute from current contents
    if (current.hash !== current.calculateHash()) return false;

    // ensure the link matches the actual previous hash
    if (current.previousHash !== previous.hash) return false;
  }
  return true;
}
}

// ---- demo / walkthrough -----
function main() {
  const cliDiff = Number(process.argv[2]);
  const difficulty = Number.isFinite(cliDiff) && cliDiff > 0 ? cliDiff
: 3;

  console.log(`\n🚀 Starting mini blockchain
(difficulty=${difficulty})\n`);
  const demoCoin = new Blockchain(difficulty);

  console.log('🔨 Mining block #1 ...');
  demoCoin.addBlock(
    new Block(1, new Date().toISOString(), { from: 'Alice', to: 'Bob',
amount: 50 })

```

```

);

console.log('🔨 Mining block #2 ...');
demoCoin.addBlock(
    new Block(2, new Date().toISOString(), { from: 'Charlie', to:
'Dana', amount: 75 })
);

console.log('🔨 Mining block #3 ...');
demoCoin.addBlock(
    new Block(3, new Date().toISOString(), [
        { from: 'Eve', to: 'Frank', amount: 20 },
        { from: 'Gina', to: 'Hank', amount: 10 },
    ])
);

// Show the chain
console.log('\n📄 Full chain:');
console.log(JSON.stringify(demoCoin, null, 2));

// Validate (should be true)
console.log('\n🔍 Is chain valid?', demoCoin.isChainValid());

// Tamper test
console.log('\n⚠️ Tampering with block #1 data ...');
demoCoin.chain[1].data.amount = 9999;

// Validate again (should be false)
console.log('🔒 Is chain valid after tamper?',
demoCoin.isChainValid());
}

main();

```

Output Screenshot

```
function main() {
  // ...
  // Tamper test
  console.log('\n⚠ Tampering with block #1 data ...');
  demoCoin.chain[1].data.amount = 9999;
  // ...
  // Validate again (should be false)
  console.log('🔒 Is chain valid after tamper?', demoCoin.ischainValid());
  // ...
}
```

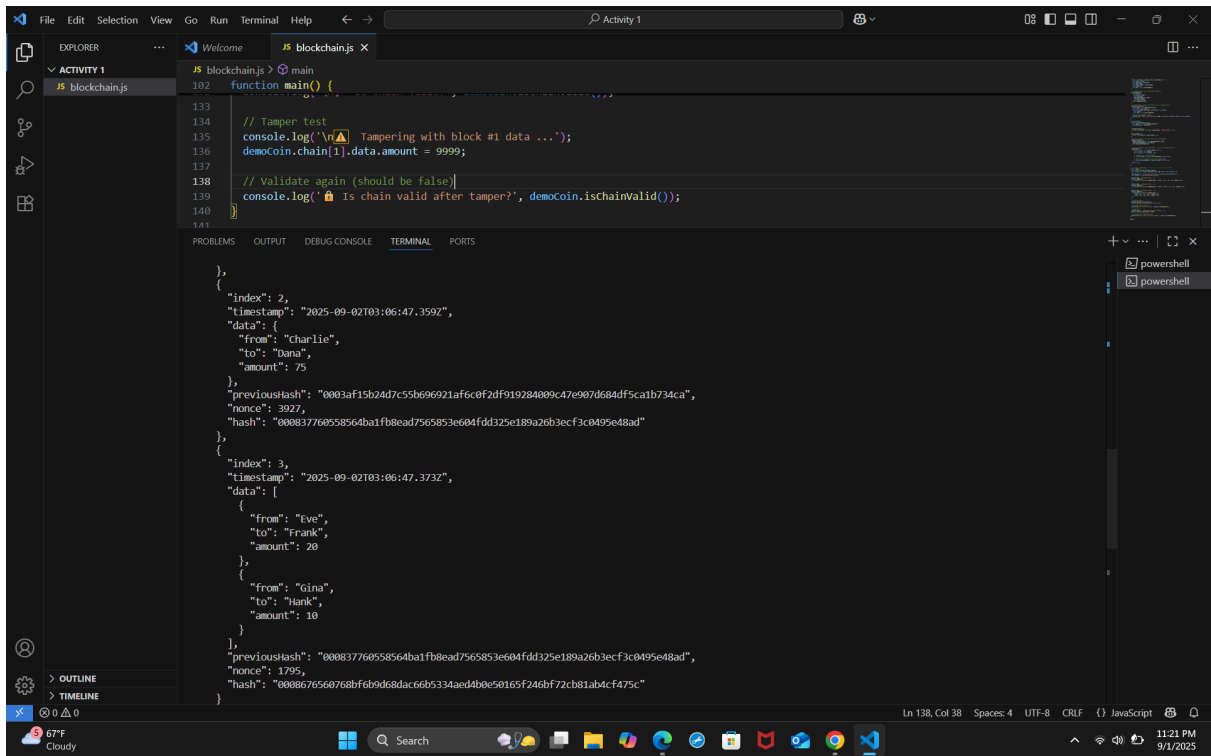
```
PS C:\Users\amish\Desktop\VMKC\Academic\Fall 2025\blockchain\Activity 1> npm -v
10.9.3
PS C:\Users\amish\Desktop\VMKC\Academic\Fall 2025\blockchain\Activity 1> node blockchain.js

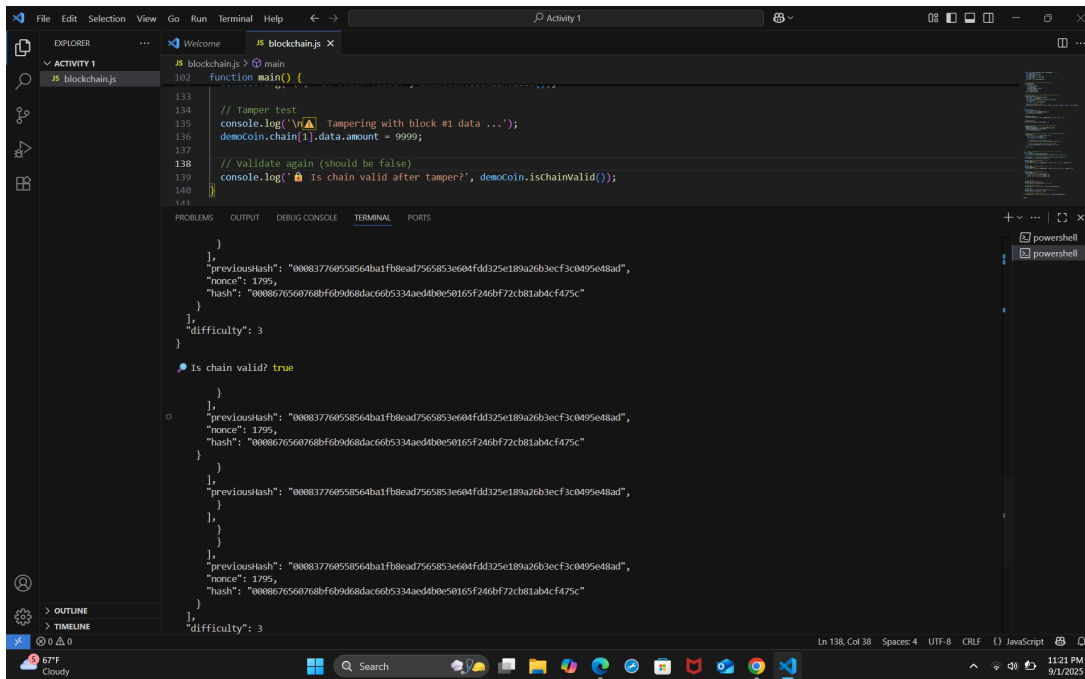
Starting mini blockchain (difficulty=3)

Mining block #1 ...
Block mined (idx=1, nonce=4671, 62ms): 0003af15b24d7c55b696921afec0f2df919284009c47e907d684df5ca1b734ca
Mining block #2 ...
Block mined (idx=2, nonce=3927, 14ms): 000837760558564ba1fb8ead7565853e604fdd325e189a26b3ecf3c0495e48ad
Mining block #3 ...
Block mined (idx=3, nonce=1795, 15ms): 0008676560768bf6b9d68dac66b5334aed4b0e90165f246bf72cb81ab4cf475c

Full chain:
{
  "chain": [
    {
      "index": 0,
      "timestamp": "2025-09-02T03:06:47.292Z",
      "data": "Genesis Block",
      "previousHash": "0",
      "nonce": 0,
      "hash": "a04e4894586e6d72686336a9e59564e1f85b734ff897489c61a3713b0e8442fc"
    },
    {
      "index": 1,
      "timestamp": "2025-09-02T03:06:47.295Z",
      "data": {
        "from": "Alice",
        "to": "Bob",
        "amount": 50
      },
      "previousHash": "a04e4894586e6d72686336a9e59564e1f85b734ff897489c61a3713b0e8442fc",
      "nonce": 4671,
      "hash": "0003af15b24d7c55b696921afec0f2df919284009c47e907d684df5ca1b734ca"
    }
  ]
}
```

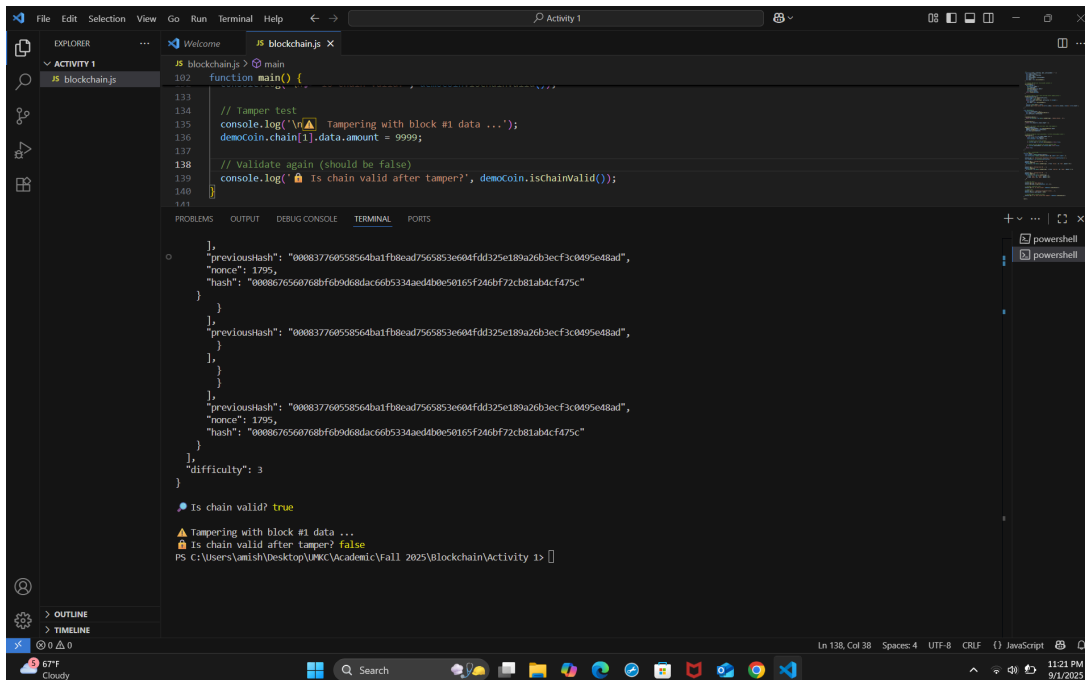
```
Full chain:
{
  "chain": [
    {
      "index": 0,
      "timestamp": "2025-09-02T03:06:47.292Z",
      "data": "Genesis Block",
      "previousHash": "0",
      "nonce": 0,
      "hash": "a04e4894586e6d72686336a9e59564e1f85b734ff897489c61a3713b0e8442fc"
    },
    {
      "index": 1,
      "timestamp": "2025-09-02T03:06:47.295Z",
      "data": {
        "from": "Alice",
        "to": "Bob",
        "amount": 50
      },
      "previousHash": "a04e4894586e6d72686336a9e59564e1f85b734ff897489c61a3713b0e8442fc",
      "nonce": 4671,
      "hash": "0003af15b24d7c55b696921afec0f2df919284009c47e907d684df5ca1b734ca"
    },
    {
      "index": 2,
      "timestamp": "2025-09-02T03:06:47.359Z",
      "data": {
        "from": "Charlie",
        "to": "Dana",
        "amount": 75
      },
      "previousHash": "0003af15b24d7c55b696921afec0f2df919284009c47e907d684df5ca1b734ca",
      "nonce": 3927,
      "hash": "000837760558564ba1fb8ead7565853e604fdd325e189a26b3ecf3c0495e48ad"
    }
  ]
}
```





```
function main() {  
  // Tamper test  
  console.log("\n⚠ Tampering with block #1 data ...");  
  demoCoin.chain[1].data.amount = 9999;  
  // Validate again (should be false)  
  console.log("🔒 Is chain valid after tamper?", demoCoin.isChainValid());  
}
```

Is chain valid? true



```
function main() {  
  // Tamper test  
  console.log("\n⚠ Tampering with block #1 data ...");  
  demoCoin.chain[1].data.amount = 9999;  
  // Validate again (should be false)  
  console.log("🔒 Is chain valid after tamper?", demoCoin.isChainValid());  
}
```

Is chain valid? true

⚠ Tampering with block #1 data ...

🔒 Is chain valid after tamper? false

PS C:\Users\amish\Desktop\UPKC\Academic\Fall 2025\Blockchain\Activity 1>