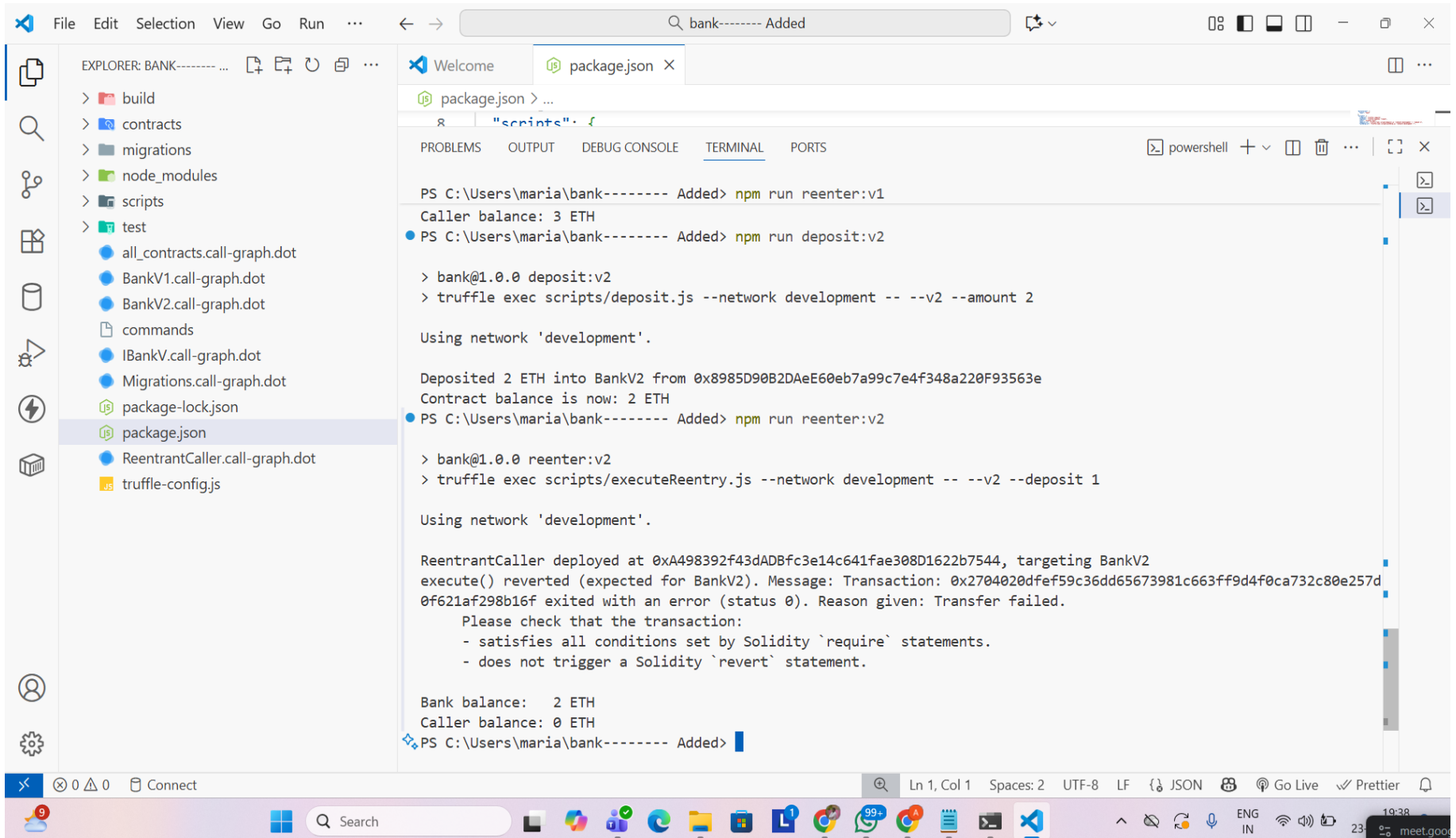


Output – Experiment 1



```
VS Code Explorer: BANK----- ...
> build
> contracts
> migrations
> node_modules
> scripts
> test
  all_contracts.call-graph.dot
  BankV1.call-graph.dot
  BankV2.call-graph.dot
  commands
  lBankV.call-graph.dot
  Migrations.call-graph.dot
  package-lock.json
  package.json
  ReentrantCaller.call-graph.dot
  truffle-config.js

Terminal:
PS C:\Users\maria\bank----- Added> npm run reenter:v1
Caller balance: 3 ETH
PS C:\Users\maria\bank----- Added> npm run deposit:v2
> bank@1.0.0 deposit:v2
> truffle exec scripts/deposit.js --network development -- --v2 --amount 2

Using network 'development'.

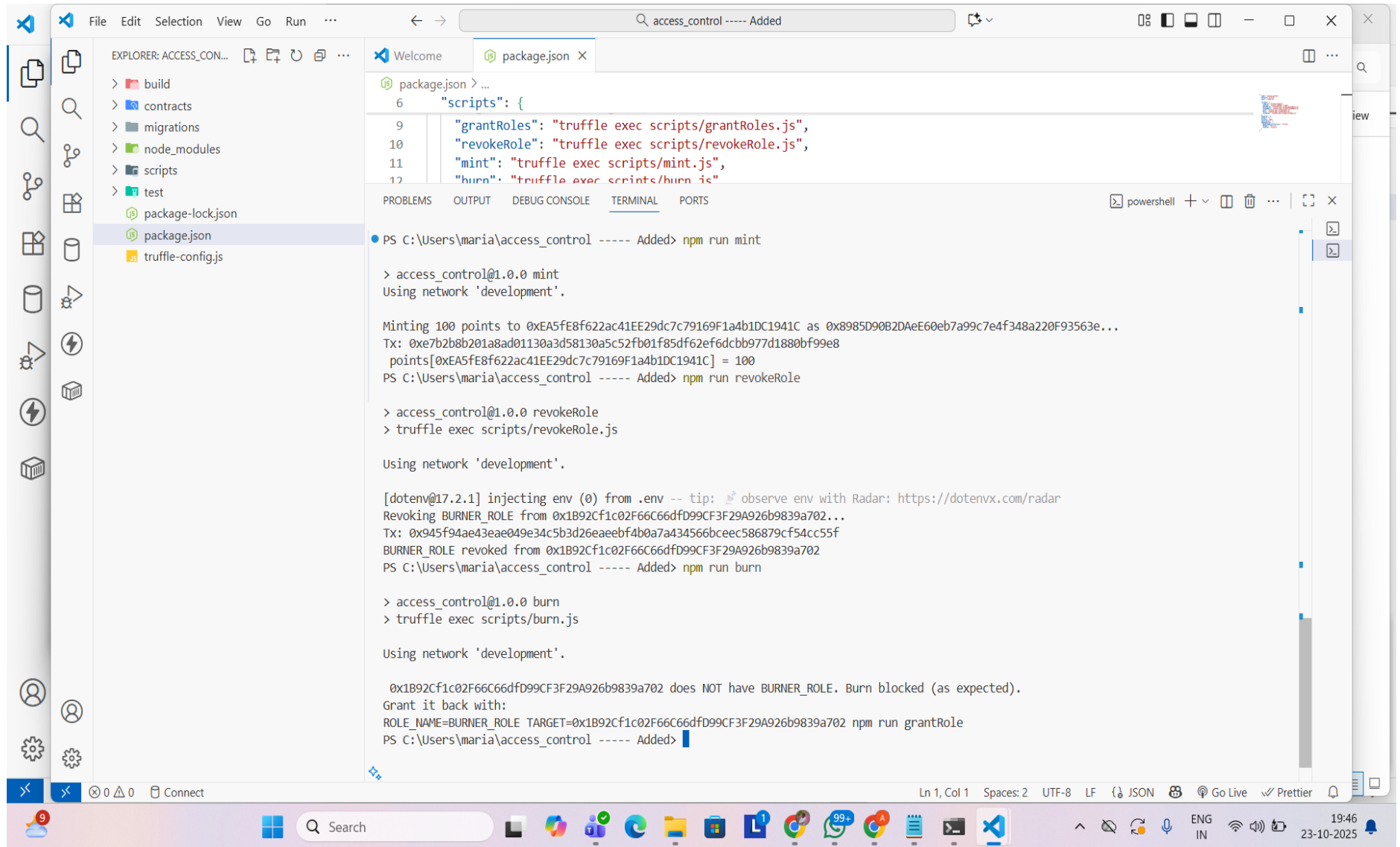
Deposited 2 ETH into BankV2 from 0x8985D90B2DAeE60eb7a99c7e4f348a220F93563e
Contract balance is now: 2 ETH
PS C:\Users\maria\bank----- Added> npm run reenter:v2
> bank@1.0.0 reenter:v2
> truffle exec scripts/executeReentry.js --network development -- --v2 --deposit 1

Using network 'development'.

ReentrantCaller deployed at 0xA498392f43dADBfc3e14c641fae308D1622b7544, targeting BankV2
execute() reverted (expected for BankV2). Message: Transaction: 0x2704020dfef59c36dd65673981c663ff9d4f0ca732c80e257d0f621af298b16f exited with an error (status 0). Reason given: Transfer failed.
Please check that the transaction:
- satisfies all conditions set by Solidity `require` statements.
- does not trigger a Solidity `revert` statement.

Bank balance: 2 ETH
Caller balance: 0 ETH
PS C:\Users\maria\bank----- Added>
```

Output – Experiment 2



```
package.json > ...
6  "scripts": {
9    "grantRoles": "truffle exec scripts/grantRoles.js",
10   "revokeRole": "truffle exec scripts/revokeRole.js",
11   "mint": "truffle exec scripts/mint.js",
12   "burn": "truffle exec scripts/burn.js"
}

PS C:\Users\maria\access_control ----- Added> npm run mint

> access_control@1.0.0 mint
Using network 'development'.

Minting 100 points to 0xEa5fE8f622ac41EE29dc7c79169F1a4b1DC1941C as 0x8985D90B2DAeE60eb7a99c7e4f348a220F93563e...
Tx: 0xe7b2b8b201a8ad01130a3d58130a5c52fb01f85df62ef6dcbb977d1880bf99e8
points[0xEa5fE8f622ac41EE29dc7c79169F1a4b1DC1941C] = 100
PS C:\Users\maria\access_control ----- Added> npm run revokeRole

> access_control@1.0.0 revokeRole
> truffle exec scripts/revokeRole.js

Using network 'development'.

[dotenv@17.2.1] injecting env (0) from .env -- tip: observe env with Radar: https://dotenvx.com/radar
Revoking BURNER_ROLE from 0x1B92Cf1c02F66C66dfD99CF3F29A926b9839a702...
Tx: 0x945f94ae43eae049e34c5b3d26eaeefb4b0a7a434566bceec586879cf54cc55f
BURNER_ROLE revoked from 0x1B92Cf1c02F66C66dfD99CF3F29A926b9839a702
PS C:\Users\maria\access_control ----- Added> npm run burn

> access_control@1.0.0 burn
> truffle exec scripts/burn.js

Using network 'development'.

0x1B92Cf1c02F66C66dfD99CF3F29A926b9839a702 does NOT have BURNER_ROLE. Burn blocked (as expected).
Grant it back with:
ROLE_NAME=BURNER_ROLE TARGET=0x1B92Cf1c02F66C66dfD99CF3F29A926b9839a702 npm run grantRole
PS C:\Users\maria\access_control ----- Added> 
```

Output – Experiment 3

The screenshot shows a Visual Studio Code editor with the following components:

- EXPLORER:** Displays the file structure of the project, including folders like `build`, `contracts`, `migrations`, `node_modules`, and `scripts`. The `scripts` folder is expanded, showing files like `checkBalance.js`, `eventListener.js`, `mint.js`, `queryEvents.js`, and `transfer.js`. The `package.json` file is selected.
- EDITOR:** Displays the content of `package.json`, showing the `scripts` section with the following configuration:

```
8 },
9   "scripts": {
10     "deploy": "truffle migrate --reset",
11     "mint": "truffle exec scripts/mint.js",
12     "transfer": "truffle exec scripts/transferTokens.js",
13     "listen": "truffle exec scripts/eventListener.js",
14     "test": "truffle test"
15   }
```
- TERMINAL:** Shows the output of the command `npm run listen`. The output indicates that the application is using the `development` network and attempting to connect to MongoDB. The connection is successful, and the application starts listening for events from block 2 to 2. A mint event is saved to MongoDB with the following details:

```
Mint event saved to MongoDB: {
  to: '0x8985D90B2DAe60eb7a99c7e4f348a220F93563e',
  amount: '100',
  transactionHash: '0xe71b0a3674ee174e578ead8fe15ae357288e2f3a1adcb51ba1d718ee7914410e'
}
```

The status bar at the bottom indicates the current line and column (Ln 11, Col 39), the number of spaces (2), the encoding (UTF-8), the line feed (LF), the JSON format, and the Prettier extension.

Output – Experiment 4

The screenshot displays the Visual Studio Code interface for a project named "Event-trigger". The Explorer sidebar on the left shows the project structure, including files like `artifacts`, `cache`, `contracts` (with `Lock.sol` and `ProductManager.sol`), `ignition`, `node_modules`, and `scripts` (containing `createProduct.js`, `eventListener.js`, `getBalance.js`, `getProductDetails.js`, `markedDelivered.js`, `purchaseProduct.js`, and `withdraw.js`). The `test` directory contains `EventTrigger.js` and `Lock.js`. The `package.json` file is open in the editor, showing the following scripts:

```
"scripts": {
  "--network localhost",
  "create-product": "hardhat run scripts/createProduct.js",
  "purchase-product": "hardhat run scripts/purchaseProduct.js",
  "mark-delivered": "hardhat run scripts/markDelivered.js",
  "withdraw": "hardhat run scripts/withdraw.js",
  "get-product": "hardhat run scripts/getProductDetails.js"
},
```

The terminal at the bottom shows the execution of the command `hardhat run .\scripts\eventListener.js`. The output indicates that the term 'hardhat' is not recognized as a commandlet, function, script file, or operable program. The error message is:

```
PS C:\Users\maria\Event-trigger> hardhat run .\scripts\eventListener.js
hardhat : The term 'hardhat' is not recognized as the name of a cmdlet, function, script file,
or operable program. Check the spelling of the name, or if a path was included, verify that the
path is correct and try again.
At line:1 char:1
+ hardhat run .\scripts\eventListener.js
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (hardhat:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

The terminal then shows the command `npm hardhat run .\scripts\eventListener.js` being executed. The output shows the product details:

```
[PRODUCT CREATED]
Product ID: 2
Name: kitbag
Price: 0.5 ETH
Block: 4
TX: 0xe5176fb67ef72ae31bec7bf402f82d03d5533579115e6480eacf0a3b9aec2ef1
```

The right sidebar shows the "CHAT" panel with the message "Build with agent mode" and a note that "AI responses may be inaccurate." It also includes a link to "Generate Agent Instructions to onboard AI onto your codebase."

Output – Experiment 5

The screenshot displays the Postman application interface. At the top, there is a navigation bar with 'Home', 'Workspaces', and 'Explore' options, along with a search bar and buttons for 'Sign In' and 'Create Account'. A blue banner below the navigation bar states: 'You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.'

The main interface is divided into three panels. The left panel, titled 'History', shows a list of recent requests, including a 'POST' request to 'http://localhost:3000/mint'. The middle panel displays the details of the selected request, showing the URL 'http://localhost:3000/mint' and the method 'POST'. The 'Body' tab is active, showing a JSON payload:

```
{  "to": "0x70997970C51812dc3A010C7d01b50e0d17dc79C8",  "amount": "250"}
```

. The right panel shows the response, which is a 200 OK status with a 508 ms response time and 422 B of data. The response body is displayed in the 'Pretty' view, showing a JSON object:

```
{  "success": true,  "message": "Minted 250 tokens to 0x70997970C51812dc3A010C7d01b50e0d17dc79C8",  "txHash": "0x3df228f6861470abc8f461bed0967547679ba7f1273dacea3f2f07f7e5fd4cab",  "blockNumber": 1}
```

At the bottom of the interface, there is a console panel showing the message 'Not connected to a Postman account'. The Windows taskbar is visible at the very bottom, showing the time as 23:28 on 23-10-2025.

Output – Experiment 6

```
client > package.json > ...
1 {
2   "name": "my-v0-project",
3   "version": "0.1.0",
4   "private": true,
5 }

PS C:\Users\maria\Downloads\Ipfs\server> node server.js
[+] Documents table is ready.
Database path: C:\Users\maria\Downloads\Ipfs\server\database\documents.db
Server running on http://localhost:3000/
Test the server at http://localhost:3000/
Documents API at http://localhost:3000/api/documents/
[+] Connected to SQLite database.
[+] Documents table is ready.
POST /api/documents/upload
[+] Upload request received
[+] File: Media.jpeg (190766 bytes)
[+] Adding to IPFS...
Uploading file to Pinata IPFS... Media.jpeg
File uploaded to Pinata successfully!
IPFS Hash (CID): Qmfwv6RkNgZSm4X5eaRA2nEEPFT3wKoY7L5HcyvtqpuP2U
Pin Size: 190780
[+] IPFS Result: {
  cid: 'Qmfwv6RkNgZSm4X5eaRA2nEEPFT3wKoY7L5HcyvtqpuP2U',
  pinSize: 190780,
  timestamp: '2025-10-23T17:51:35.256Z'
}
[+] Saving to database...
[+] Document saved to database with ID: 10
```

Output – Experiment 7

The screenshot shows a Visual Studio Code editor window with a project named "web3-library (1)". The Explorer sidebar on the left shows the project structure, including folders like "client", "server", "build", "contracts", "migrations", "node_modules", "scripts", and "test", and files like ".env", ".gitignore", ".secret", "package-lock.json", "package.json", and "truffle-config.js". The "package.json" file is open in the editor, showing the following "scripts" section:

```
"scripts": {
  "migrate:reset": "truffle migrate --reset",
  "migrate:ganache": "truffle migrate --network ganache --reset",
  "migrate:sepolia": "truffle migrate --network sepolia --reset",
  "test": "truffle test",
  "test:ganache": "truffle test --network ganache",
  "console": "truffle console",
  "console:ganache": "truffle console --network ganache",
  "console:sepolia": "truffle console --network sepolia",
  "add-books": "node scripts/addBooks.js",
  "add-books:local": "NETWORK=development node scripts/addBooks.js",
  "add-books:ganache": "NETWORK=ganache node scripts/addBooks.js",
  "add-books:sepolia": "NETWORK=sepolia node scripts/addBooks.js",
  "verify:sepolia": "truffle run verify Web3Library --network sepolia",
  "deploy:local": "npm run migrate:reset",
  "deploy:ganache": "npm run migrate:ganache && npm run add-books:ganache",
  "deploy:sepolia": "npm run migrate:sepolia && npm run add-books:sepolia"
}
```

The Terminal panel at the bottom shows the following commands and output:

```
PS C:\Users\maria\Downloads\web3-library (1)\server> npm run migrate:reset
=====
> Total deployments: 2
> Final cost: 0.005524918714439195 ETH

• PS C:\Users\maria\Downloads\web3-library (1)\server> npm run add-books

> web3-library-server@1.0.0 add-books
> node scripts/addBooks.js

Using wallet address: 0xcf5c32Fdc7d26e07807ac7b5542b889747f1a5EF
=== Web3 Library Book Addition Script ===

Adding books in bulk...
Estimated gas for bulk add: 608549
Bulk transaction hash: 0x10612cae862609793c140e0494f05d9b9535d6cf3e9c7864e05d451150906d6f
[✓] Bulk add successful! Gas used: 602599
PS C:\Users\maria\Downloads\web3-library (1)\server>
```

The Chat sidebar on the right shows a message from GitHub Copilot: "You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew." with a button to "Upgrade to GitHub Copilot Pro".

Output – Experiment 8

The screenshot displays the Postman API client interface. At the top, there's a navigation bar with 'Home', 'Workspaces', and 'Explore' options, a search bar, and buttons for 'Sign In' and 'Create Account'. A blue banner below the navigation bar states: 'You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.'

The main interface is divided into three panels. The left panel, titled 'History', shows a list of recent requests. The middle panel displays the details of the selected request, which is a POST request to 'http://localhost:3000/register'. The right panel shows the response of the request.

History Panel: A list of requests is shown, including several GET requests to 'http://localhost:3000/student/0xf39Fd6e51aad88' and POST requests to 'http://localhost:3000/register'. A red badge indicates '8 new items'.

Request Panel: The selected request is a POST request to 'http://localhost:3000/register'. The 'Send' button is visible. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', 'Settings', and 'Cookies'. The 'Query Params' section is currently active, showing a table with 'Key' and 'Value' columns.

Key	Value	Bulk Edit
Key	Value	

Response Panel: The response is displayed in the 'Body' tab, showing a JSON object with 'success' set to true and a 'txHash' value. The status is '200 OK' with a response time of '621 ms' and a size of '329 B'. The 'Save Response' button is visible.

```
{
  "success": true,
  "txHash": "0x943bb4374ae0a99746a3bfa0514e59ecc3c9ded24e36b4ed39a8f85ae4f8d339"
}
```

The bottom of the image shows the Windows taskbar with various application icons and the system clock indicating 20:12 on 23-10-2025.