

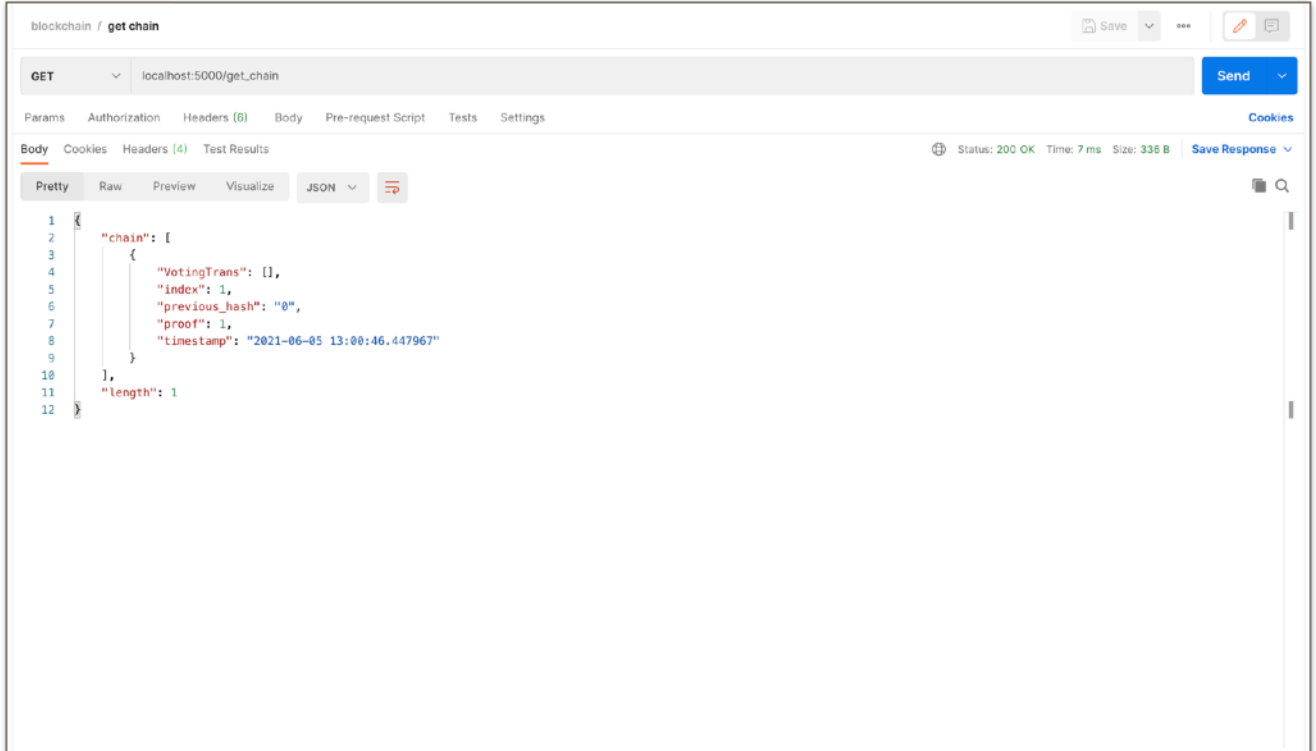
# **Academic Report Cover Page**

Sed et lacus quis enim mattis nonummy

Urna Semper  
Instructor's Name  
5 June 2021

# Working of Blockchain

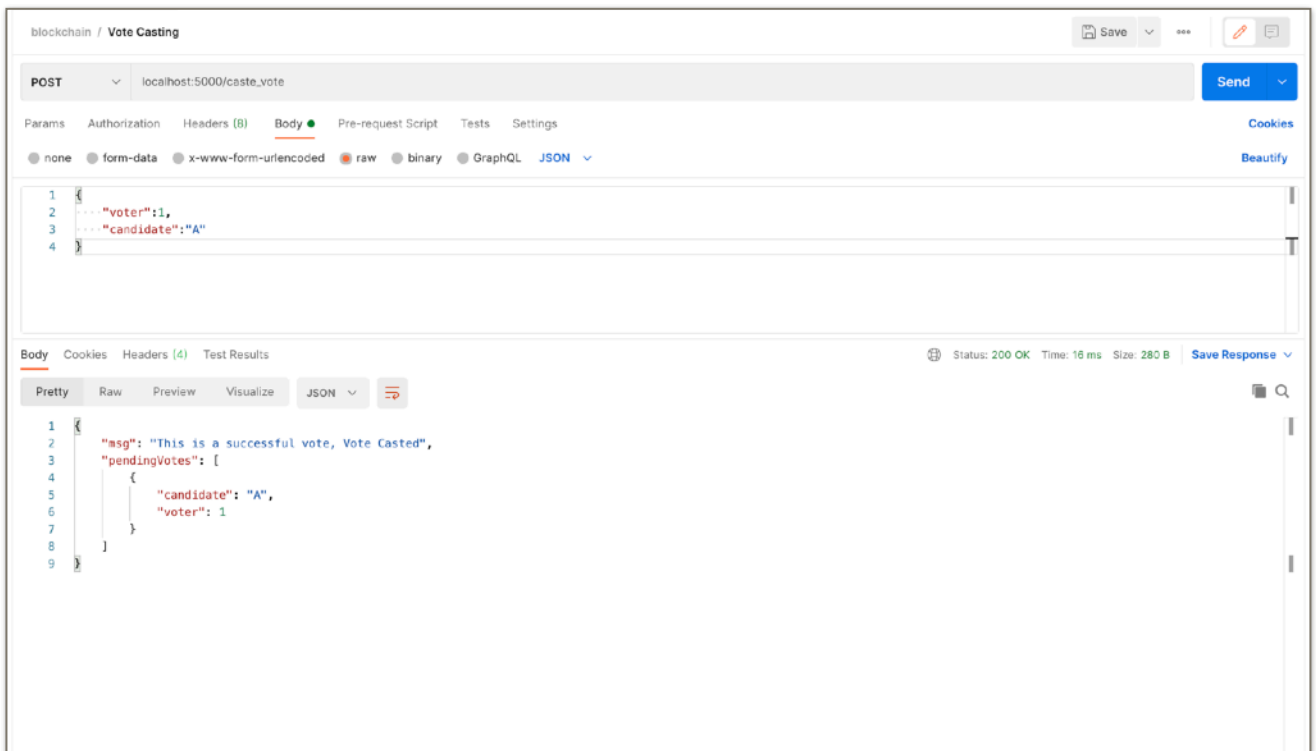
## 1. Fetching the Blockchain



The screenshot shows a REST client interface for a blockchain application. The request is a GET to `localhost:5000/get_chain`. The response is a JSON object with a status of 200 OK, a time of 7 ms, and a size of 336 B. The response body is displayed in a pretty-printed JSON format:

```
1 {
2   "chain": [
3     {
4       "VotingTrans": [],
5       "index": 1,
6       "previous_hash": "0",
7       "proof": 1,
8       "timestamp": "2021-06-05 13:00:46.447967"
9     }
10  ],
11  "length": 1
12 }
```

## 2. Casting a Vote



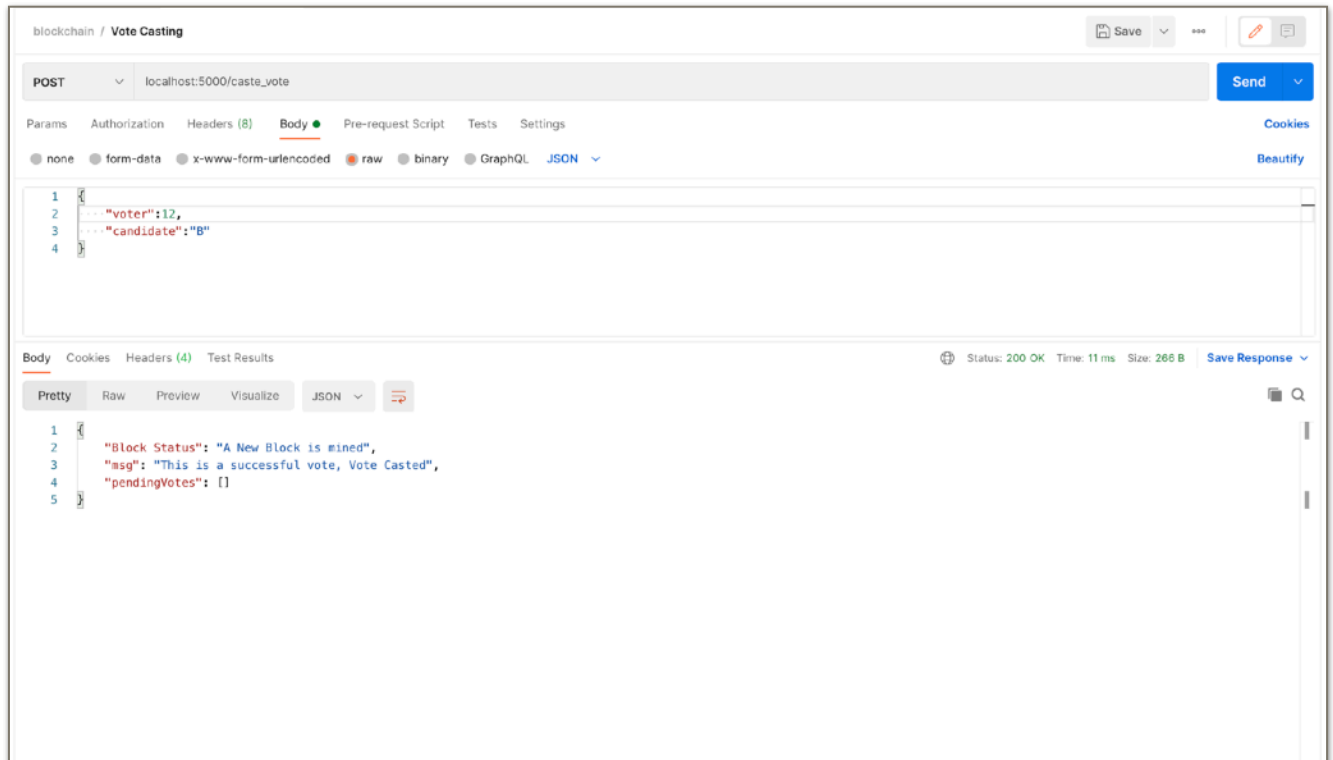
The screenshot shows a REST client interface for a blockchain application. The request is a POST to `localhost:5000/caste_vote`. The request body is a JSON object with the following data:

```
1 {
2   "voter": 1,
3   "candidate": "A"
4 }
```

The response is a JSON object with a status of 200 OK, a time of 16 ms, and a size of 280 B. The response body is displayed in a pretty-printed JSON format:

```
1 {
2   "msg": "This is a successful vote, Vote Casted",
3   "pendingVotes": [
4     {
5       "candidate": "A",
6       "voter": 1
7     }
8   ]
9 }
```

### 3. Mining a block ( After 4 Successful Votes)



blockchain / Vote Casting

POST localhost:5000/caste\_vote

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "voter": 12,
3   "candidate": "B"
4 }
```

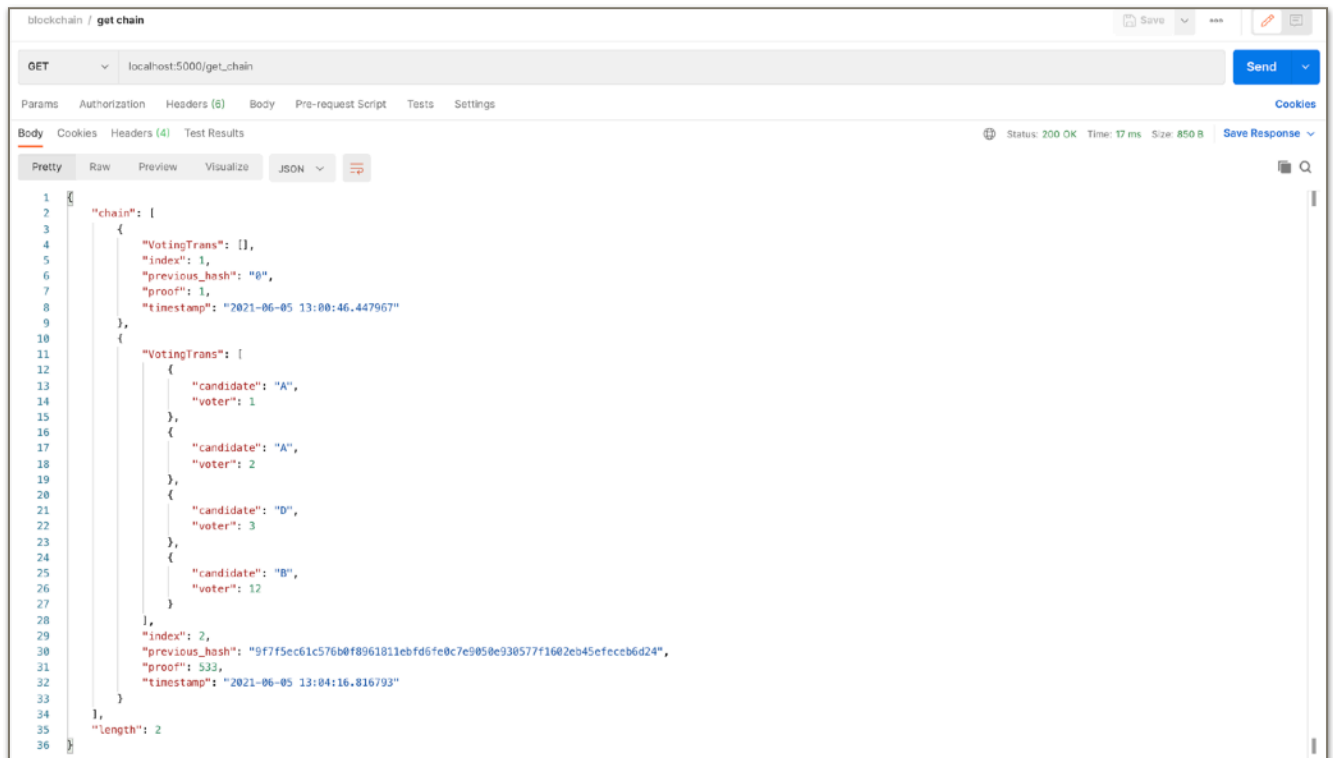
Body Cookies Headers (4) Test Results

Status: 200 OK Time: 11 ms Size: 266 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "Block Status": "A New Block is mined",
3   "msg": "This is a successful vote, Vote Casted",
4   "pendingVotes": []
5 }
```

### 4. Fetching the blockchain (After a new block is added)



blockchain / get chain

GET localhost:5000/get\_chain

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results

Status: 200 OK Time: 17 ms Size: 850 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "chain": [
3     {
4       "VotingTrans": [],
5       "index": 1,
6       "previous_hash": "0",
7       "proof": 1,
8       "timestamp": "2021-06-05 13:00:46.447967"
9     },
10    {
11      "VotingTrans": [
12        {
13          "candidate": "A",
14          "voter": 1
15        },
16        {
17          "candidate": "A",
18          "voter": 2
19        },
20        {
21          "candidate": "B",
22          "voter": 3
23        },
24        {
25          "candidate": "B",
26          "voter": 12
27        }
28      ],
29      "index": 2,
30      "previous_hash": "9f7f5ec61c576b0f8961811ebf6fe0c7e9050e938577f1602eb45efecb6d24",
31      "proof": 533,
32      "timestamp": "2021-06-05 13:04:16.816793"
33    }
34  ],
35  "length": 2
36 }
```

## 5. Blockchain (Added another Block)

blockchain / get chain

GETlocalhost:5000/get\_chain

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

BodyCookiesHeaders (4)Test Results

Status: 200 OKTime: 6 msSize: 1.34 KBSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "chain": [
3     {
4       "VotingTrans": [],
5       "index": 1,
6       "previous_hash": "0",
7       "proof": 1,
8       "timestamp": "2021-06-05 13:50:07.815750"
9     },
10    {
11      "VotingTrans": [ -
12      ],
13      "index": 2,
14      "previous_hash": "2c00389b4ad835211eb68f28e1069496795d9cba6f1250b57b36ce7c12e88c7c",
15      "proof": 533,
16      "timestamp": "2021-06-05 13:50:44.077591"
17    },
18    {
19      "VotingTrans": [ -
20      ],
21      "index": 3,
22      "previous_hash": "fb4101d4f77da8ab52b0e6acea6800509b9aaf8a4e5435353c8c229e87ee081",
23      "proof": 45293,
24      "timestamp": "2021-06-05 13:51:09.359319"
25    }
26  ],
27   "length": 3
28 }
```

## 6. Voting Results

blockchain / Results

GETlocalhost:5000/getResult

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

BodyCookiesHeaders (4)Test Results

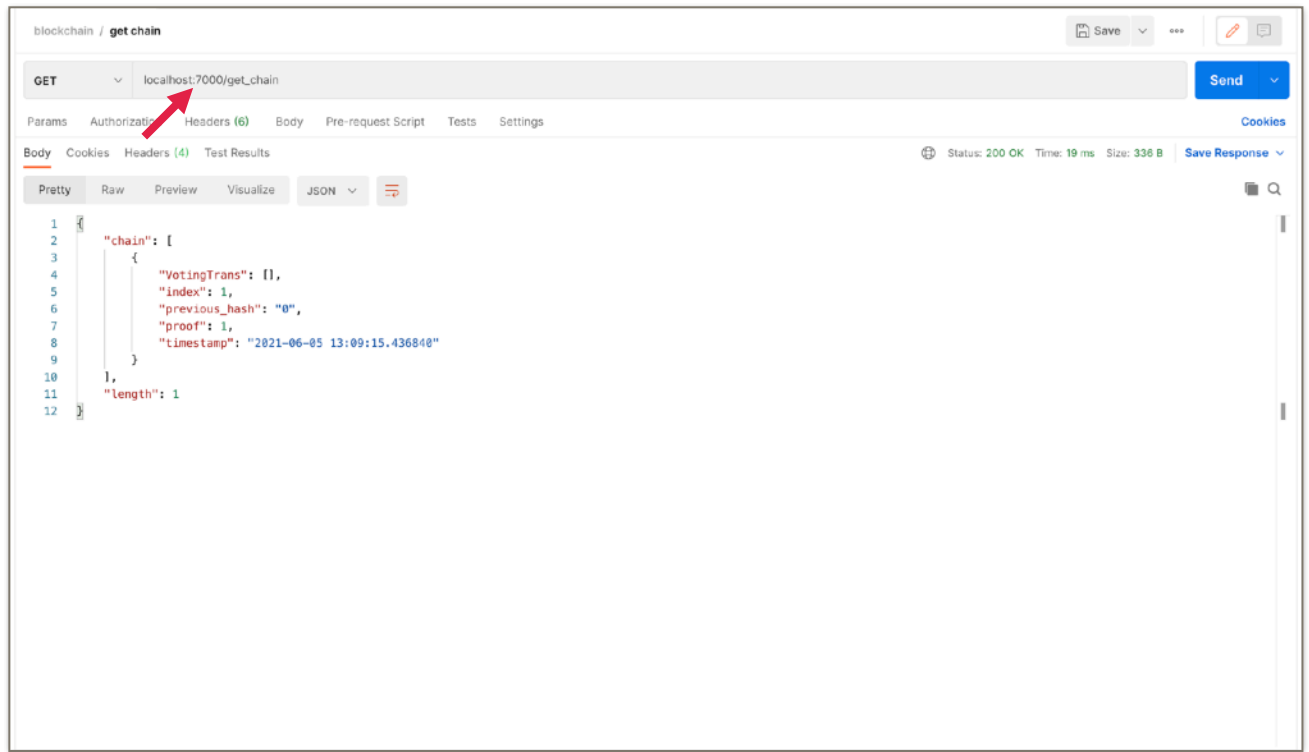
Status: 200 OKTime: 7 msSize: 191 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "A": 4,
3   "B": 2,
4   "C": 1,
5   "D": 1
6 }
```

# Decentralising the Blockchain

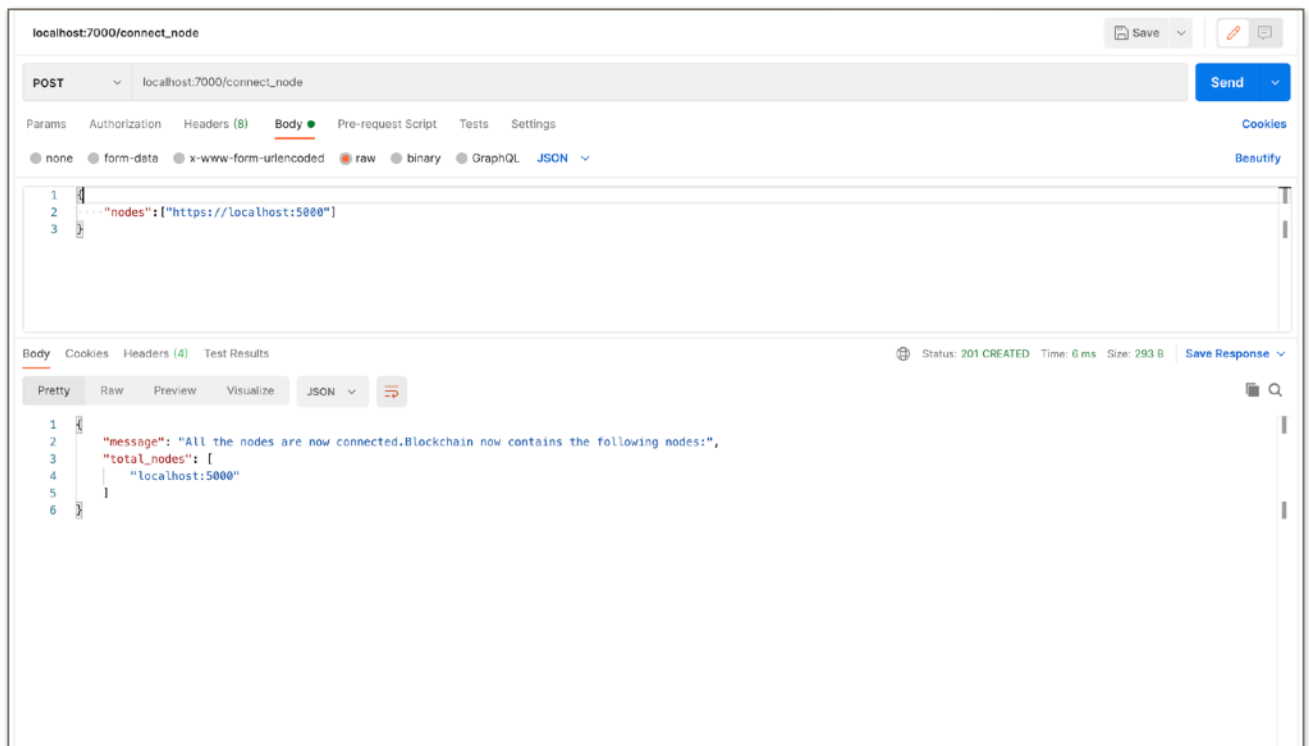
## 1. Second Node (With PORT:7000)



The screenshot shows a REST client interface for a GET request to `localhost:7000/get_chain`. The response status is 200 OK, with a time of 19 ms and a size of 336 B. The response body is a JSON object representing a blockchain chain with one block.

```
1 {
2   "chain": [
3     {
4       "VotingTrans": [],
5       "index": 1,
6       "previous_hash": "0",
7       "proof": 1,
8       "timestamp": "2021-06-05 13:09:15.436840"
9     }
10  ],
11  "length": 1
12 }
```

## 2. Adding PORT:5000 Node to the Network



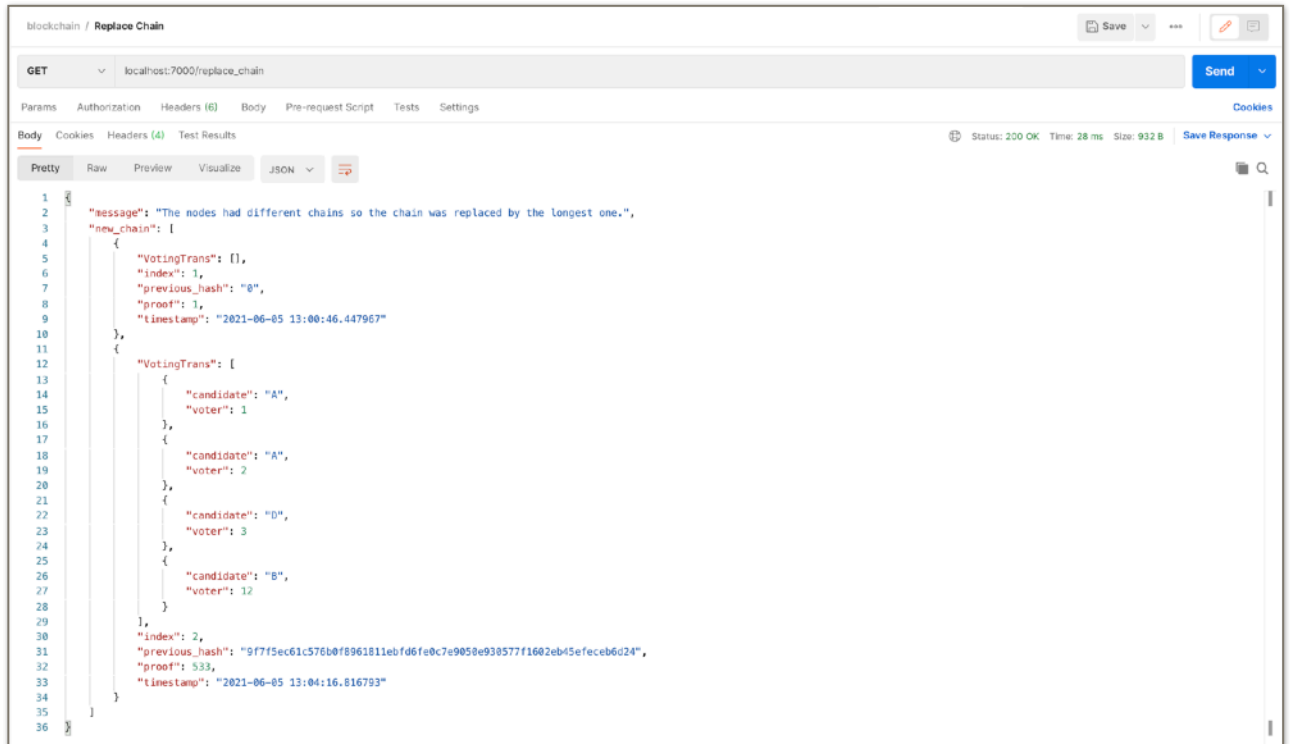
The screenshot shows a REST client interface for a POST request to `localhost:7000/connect_node`. The request body is a JSON object with a list of nodes. The response status is 201 CREATED, with a time of 6 ms and a size of 293 B. The response body is a JSON object with a message and a list of total nodes.

```
1 {
2   "nodes": ["https://localhost:5000"]
3 }
```

The response body is a JSON object with a message and a list of total nodes:

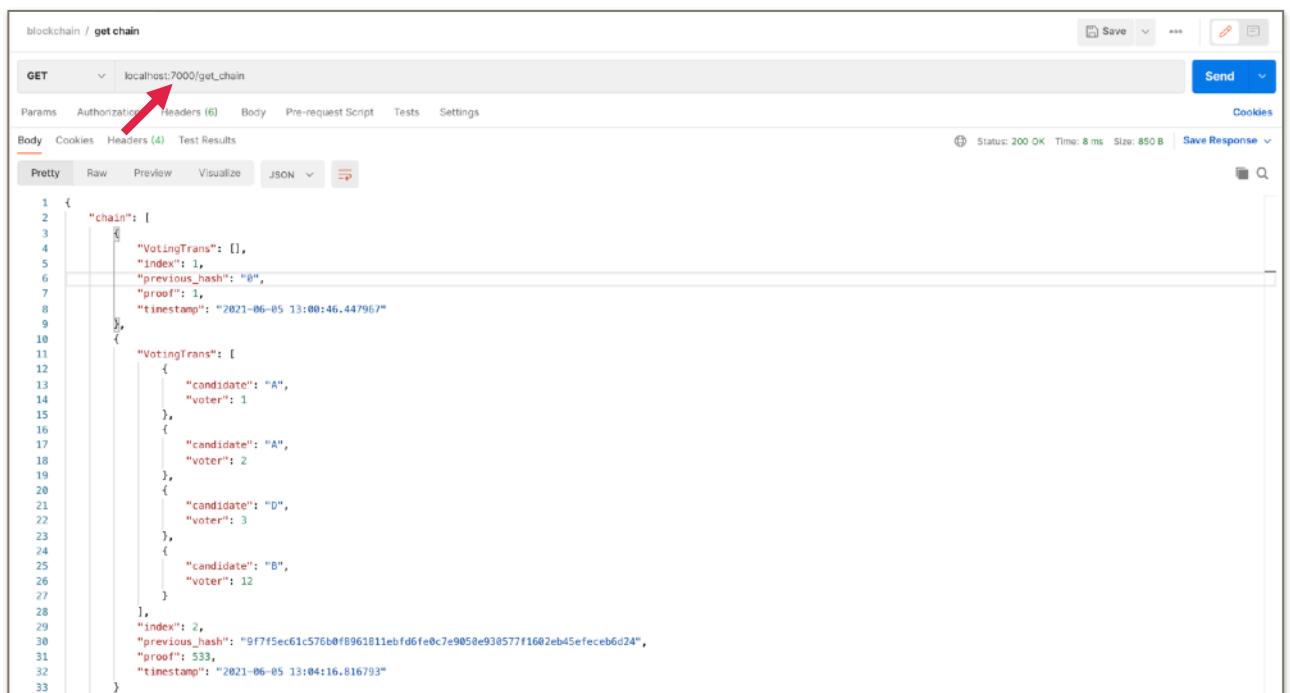
```
1 {
2   "message": "All the nodes are now connected. Blockchain now contains the following nodes:",
3   "total_nodes": [
4     "localhost:5000"
5   ]
6 }
```

### 3. Fetching the Longest Chain in the Network



```
1 {
2   "message": "The nodes had different chains so the chain was replaced by the longest one.",
3   "new_chain": [
4     {
5       "VotingTrans": [],
6       "index": 1,
7       "previous_hash": "0",
8       "proof": 1,
9       "timestamp": "2021-06-05 13:00:46.447967"
10    },
11    {
12      "VotingTrans": [
13        {
14          "candidate": "A",
15          "voter": 1
16        },
17        {
18          "candidate": "A",
19          "voter": 2
20        },
21        {
22          "candidate": "D",
23          "voter": 3
24        },
25        {
26          "candidate": "B",
27          "voter": 12
28        }
29      ],
30      "index": 2,
31      "previous_hash": "9f7f5ec61c576b0f8961811ebfd6fe0c7e9058e930577f1602eb45efec6b6d24",
32      "proof": 533,
33      "timestamp": "2021-06-05 13:04:16.816793"
34    }
35  ]
36 }
```

### 4. Getting on the Blockchain on PORT:7000 node



```
1 {
2   "chain": [
3     {
4       "VotingTrans": [],
5       "index": 1,
6       "previous_hash": "0",
7       "proof": 1,
8       "timestamp": "2021-06-05 13:00:46.447967"
9     },
10    {
11      "VotingTrans": [
12        {
13          "candidate": "A",
14          "voter": 1
15        },
16        {
17          "candidate": "A",
18          "voter": 2
19        },
20        {
21          "candidate": "D",
22          "voter": 3
23        },
24        {
25          "candidate": "B",
26          "voter": 12
27        }
28      ],
29      "index": 2,
30      "previous_hash": "9f7f5ec61c576b0f8961811ebfd6fe0c7e9058e930577f1602eb45efec6b6d24",
31      "proof": 533,
32      "timestamp": "2021-06-05 13:04:16.816793"
33    }
34  ]
35 }
```

# Checking Validity

## 1. Checking the Validity

The screenshot shows a REST client interface for a 'blockchain' project. The request is a GET to 'localhost:5000/is\_valid'. The response is a 200 OK with a status of 200 OK, time of 5 ms, and size of 199 B. The response body is displayed in 'Pretty' format as a JSON object: `{ "message": "All good. The Blockchain is valid." }`.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1 {
2   "message": "All good. The Blockchain is valid."
3 }
```

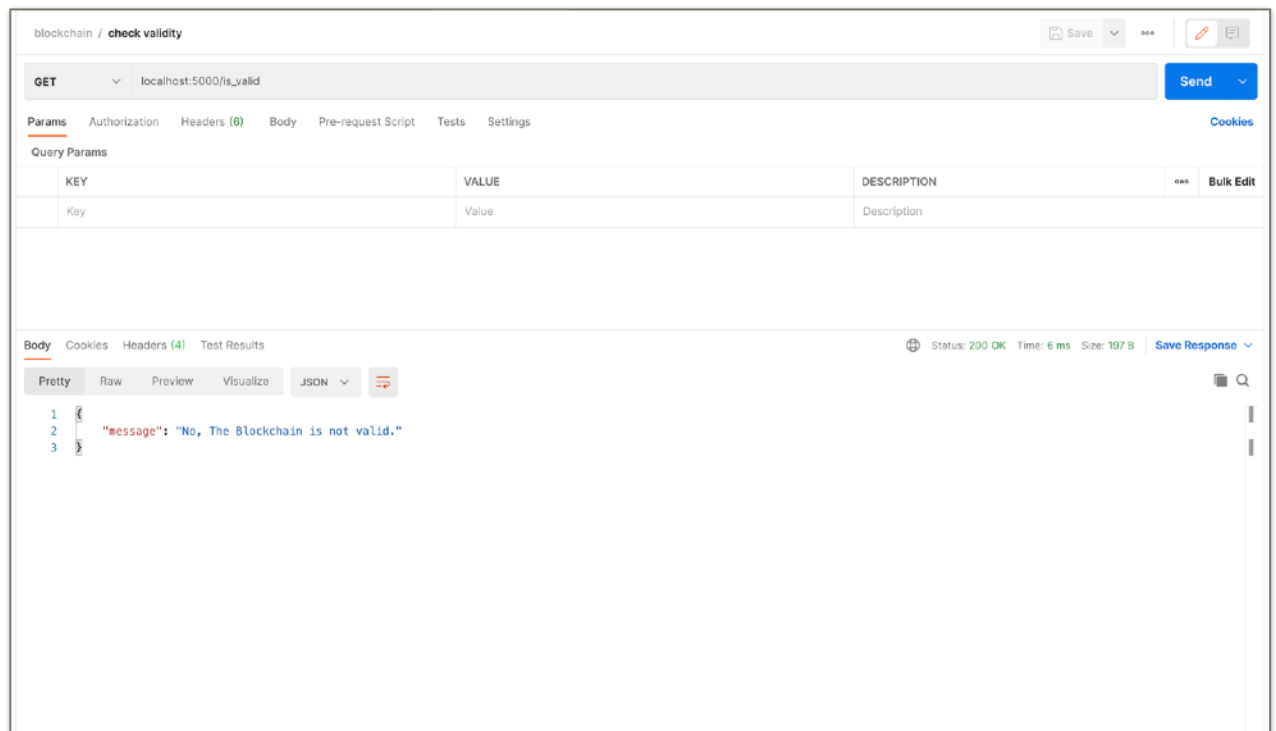
## 2. Modifying a data in the Block

The screenshot shows a REST client interface for a 'blockchain' project. The request is a GET to 'localhost:5000/change\_data'. The response is a 200 OK with a status of 200 OK, time of 5 ms, and size of 199 B. The response body is displayed in 'Pretty' format as a JSON object. A red arrow points to the 'candidate' field in the 'VotingTrans' array, which has been changed from 'A' to 'B'.

```
1 {
2   "block": {
3     "VotingTrans": [
4       {
5         "candidate": "A",
6         "voter": 1
7       },
8       {
9         "candidate": "B",
10        "voter": 2
11      },
12      {
13        "candidate": "D",
14        "voter": 3
15      },
16      {
17        "candidate": "B",
18        "voter": 12
19      }
20    ],
21    "index": 2,
22    "previous_hash": "9f7f5ec61c576b0f8961811ebfd6fe0c7e9050e930577f1602eb45efcecb6d24",
23    "proof": 533,
24    "timestamp": "2021-06-05 13:04:16.816793"
25  },
26   "msg": "Block Data Modified"
27 }
```

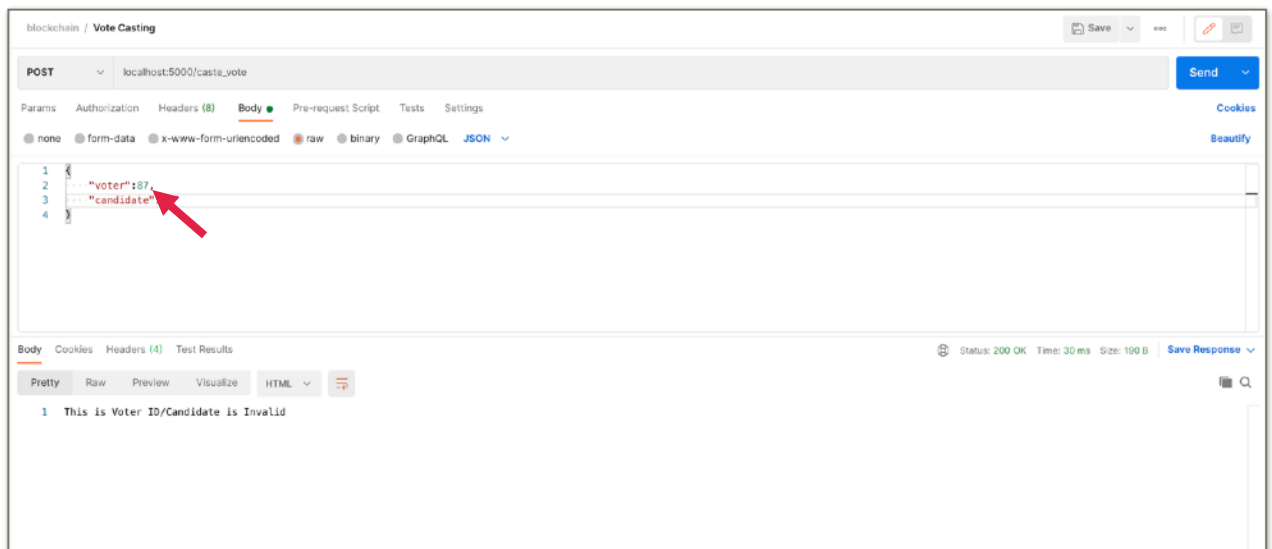
Earlier it is 'A', now changed to 'B' check above images.

### 3. Checking the Blockchain Validity after changing the data



## Input Validations

### 1. Invalid Voter ID





## 2. Invalid Candidate Name

The screenshot shows a REST client interface for a 'Vote Casting' endpoint. The request is a POST to `localhost:5000/caste_vote`. The body is a JSON object with `"voter": 4` and `"candidate": "Z"`. A red arrow points to the value `"Z"`. The response is displayed in the 'Body' tab, showing the message: `1 This is Voter ID/Candidate is Invalid`.

```
POST localhost:5000/caste_vote
```

```
{  "voter": 4,  "candidate": "Z"}
```

```
1 This is Voter ID/Candidate is Invalid
```

## 3. No Required Input

The screenshot shows a REST client interface for a 'Vote Casting' endpoint. The request is a POST to `localhost:5000/caste_vote`. The body is a JSON object with only `"voter": 4`. A red arrow points to the value `4`. The response is displayed in the 'Body' tab, showing the message: `1 Some required data for the transaction are missing`. The status bar at the bottom indicates `Status: 400 BAD REQUEST`.

```
POST localhost:5000/caste_vote
```

```
{  "voter": 4}
```

```
1 Some required data for the transaction are missing
```

Status: 400 BAD REQUEST Time: 5 ms Size: 212 B Save Response