**Code -**

import hashlib

import datetime

import json

from flask import Flask, jsonify

class Blockchain:

def \_\_init\_\_(self):

self.chain = []

self.create\_block(proof=1, previous\_hash='0')

def create\_block(self, proof, previous\_hash):

block = {

"index": len(self.chain) + 1,

"timestamp": str(datetime.datetime.now()),

"proof": proof,

"previous\_hash": previous\_hash

}

self.chain.append(block)

return block

def get\_previous\_block(self):

return self.chain[-1]

def proof\_of\_work(self, previous\_proof):

new\_proof = 1

check\_proof = False

while check\_proof == False:

new\_hash = hashlib.sha256(str(new\_proof\*\*2 - previous\_proof\*\*2).encode()).hexdigest()

if new\_hash[:4] == "0000":

check\_proof = True

else:

new\_proof += 1

return new\_proof

def hash(self, block):

encoded\_block = json.dumps(block, sort\_keys=True).encode()

return hashlib.sha256(encoded\_block).hexdigest()

def is\_chain\_valid(self, chain):

previous\_block = chain[0]

block\_index = 1

while block\_index < len(chain):

block = chain[block\_index]

if block['previous\_hash'] != self.hash(previous\_block):

return False

previous\_proof = previous\_block['proof']

proof = block['proof']

hash\_operation = hashlib.sha256(str(proof\*\*2 - previous\_proof\*\*2).encode()).hexdigest()

if hash\_operation[:4] != '0000':

return False

previous\_block = block

block\_index += 1

return True

app = Flask(\_\_name\_\_)

blockchain = Blockchain()

@app.route('/mine\_block', methods=['GET'])

def mine\_block():

previous\_block = blockchain.get\_previous\_block()

previous\_proof = previous\_block['proof']

proof = blockchain.proof\_of\_work(previous\_proof)

previous\_hash = blockchain.hash(previous\_block)

block = blockchain.create\_block(proof, previous\_hash)

response = {'message': 'A block is MINED',

'index': block['index'],

'timestamp': block['timestamp'],

'proof': block['proof'],

'previous\_hash': block['previous\_hash']}

return jsonify(response), 200

@app.route('/get\_chain', methods=['GET'])

def display\_chain():

response = {'chain': blockchain.chain,

'length': len(blockchain.chain)}

return jsonify(response), 200

@app.route('/valid', methods=['GET'])

def valid():

valid = blockchain.is\_chain\_valid(blockchain.chain)

if valid:

response = {'message': 'The Blockchain is valid.'}

else:

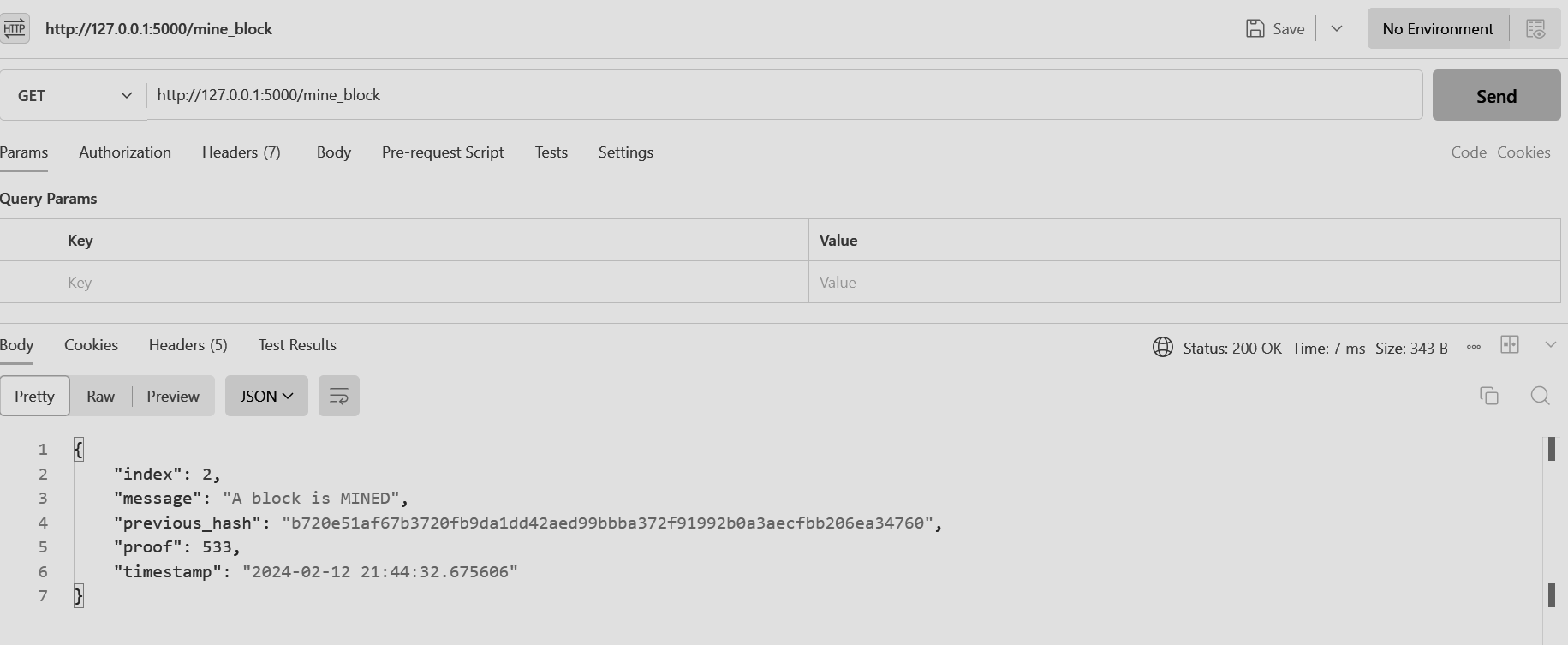
response = {'message': 'The Blockchain is not valid.'}

return jsonify(response), 200

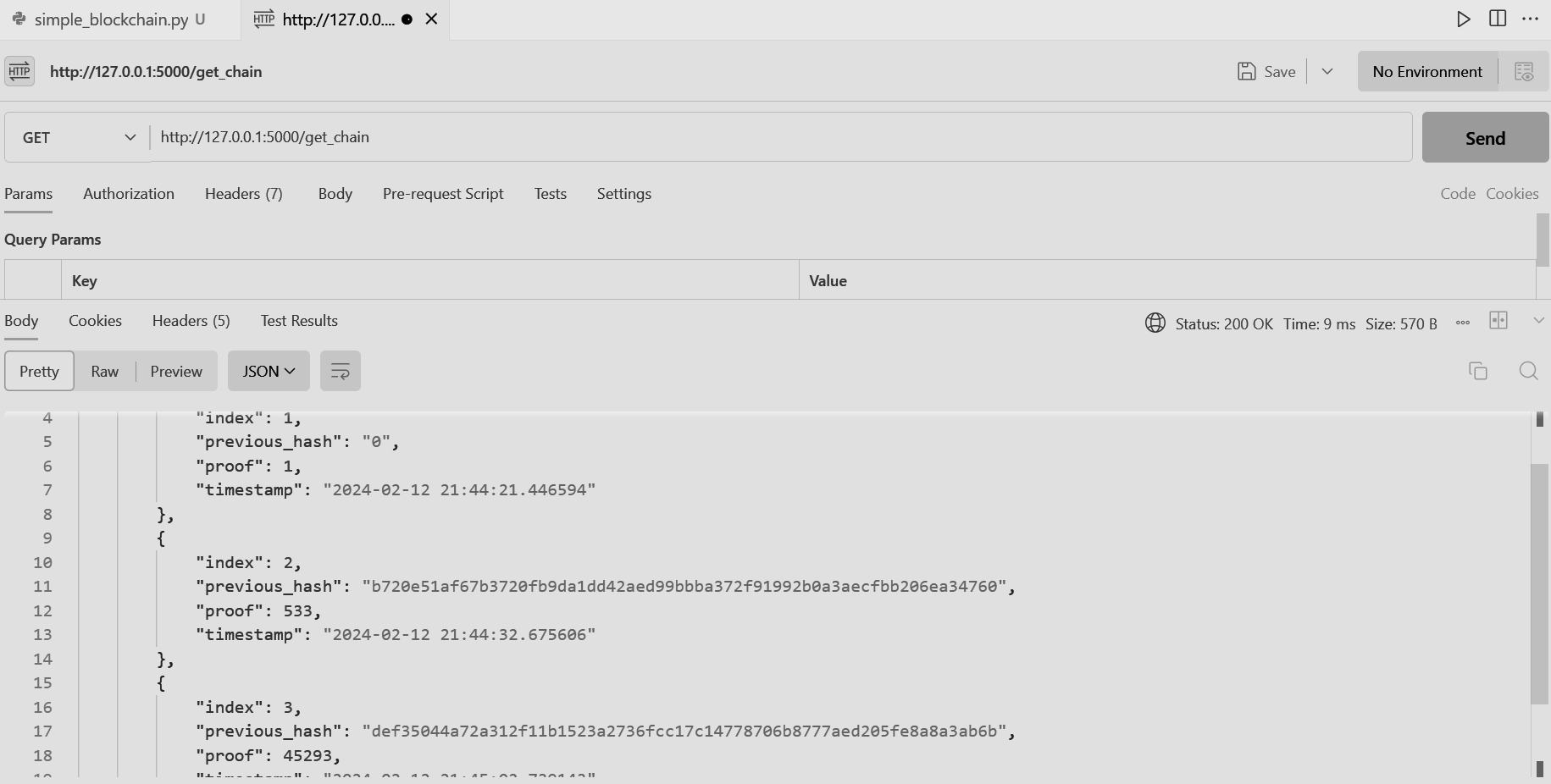
app.run(host='127.0.0.1', port=5000)

**Output-**

1. Mine Block.



1. Print Chain.



1. Valid Chain.

