Internet Technology Lab Report – Assignment 1 & 2

TITLE

Name – Sourav Dutta
Roll – 001610501076
Class – BCSE 4th year
Group – A3
Assignment Numbers – 1 and 2

Assignment 1

Implement a TCP-based key-value store. The server implements the key-value store and clients make use of it. The server must accept clients' connections and serve their requests for 'get' and 'put' key value pairs. All key-value pairs should be stored by the server only in memory. Keys and values are strings.

The client accepts a variable no of command line arguments where the first argument is the server hostname followed by port no. It should be followed by any sequence of "get <key>" and/or "put <key> <value>".

./client 192.168.124.5 5555 put city Kolkata put country India get country get city get Institute
India
Kolkata
<blank>

The server should be running on a TCP port. The server should support multiple clients and maintain their key-value stores separately.

Implement authorization so that only few clients having the role "manager" can access other's key-value stores. A user is assigned the "guest" role by default. The server can upgrade a "guest" user to a "manager" user.

Assignment 2

Implement a key-value store using Websocket. The server implements the key-value store and clients make use of it. The server must accept clients' connections and serve their requests for 'get' and 'put' key value pairs. All key-value pairs should be stored by the server only in memory. Keys and values are strings as in Assignment 1.

Implement authorization so that only few clients having the role "manager" can access other's key-value stores. A user is assigned the "guest" role by default. The server can upgrade a "guest" user to a "manager" user.

CODE SNIPPET

- **Assignment 1**

Server program (server.py): [Language: Python]

```
import socket
import time

class Server():
      def __init__(self):
            self.serverhost = '127.0.0.1'
            self.serverport = 8080
            self.max_client = 100
            self.auth_dict = {}
            self.key_store = {}

      def init_authorization(self):
            for i in range(1,self.max_client+1):
                  self.auth_dict[i] = "guest"
                  self.key_store[i] = {}

      def process_query(self):

            while True:
                  res = input('Want to close the server (y/n)? ')
                  if res == 'y':
                        return

                  while True:
                        res = input('Want to change the authorization to any
client (y/n)? ')
                        if res == 'y':
                              client_no, auth = input('Enter client number
and role (manager/guest): ').split()
                              if auth == 'manager' or auth == 'guest':
                                    self.auth_dict[int(client_no)] = auth
                                    print('Client-'+client_no+' is now a
\''+auth+'\' !!')
                        elif res == 'n':
                              break
                        else:
                              print('Invalid response! Please enter again')
                  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                  sock.bind((self.serverhost,self.serverport))
                  sock.listen(1)
                  conn = sock.accept()

                  query = []
                  response = []
                  data = conn[0].recv(1024).decode()
                  while data:
                        if data == 'end':
                              break
                        query.append(data)
                        data = conn[0].recv(1024).decode()
```

```
                qlen = len(query)
                client_no = int(query[1])
                print('-'*70)
                print('\t\tConnected with client '+query[1])
                print('-'*70)
                #for i in range(4,qlen):
                i = 4
                while i < qlen:
                     if query[i] == 'get':
                          if i+1 < qlen:
                               if query[i+1] in
self.key_store[client_no]:

     response.append(self.key_store[client_no][query[i+1]])
                                    print('GET REQUEST: Sending the
Value of \''+query[i+1]+'\' as
\''+self.key_store[client_no][query[i+1]]+'\' to client-'+str(client_no))
                               else:
                                    if self.auth_dict[client_no] ==
'manager':
                                         print(query[i+1],'could not be
found in self key-value store')
                                         print('Since the client is a
manager, Finding it in other\'s key-value store')
                                         found = 0
                                         for j in
range(1,self.max_client+1):
                                              if query[i+1] in
self.key_store[j]:
                                                   found = 1

     response.append(self.key_store[j][query[i+1]])
                                                   print('GET REQUEST:
Sending the Value of \''+query[i+1]+'\' as
\''+self.key_store[j][query[i+1]]+'\' to client-'+str(client_no))
                                                   break
                                         if found == 0:

     response.append("<blank>")
                                              print('GET REQUEST:
Sending the Value of \''+query[i+1]+'\' as \''+'<blank>'+'\' to client-
'+str(client_no))
                                    else:
                                         response.append("<blank>")
                                         print('GET REQUEST: Sending
the Value of \''+query[i+1]+'\' as \''+'<blank>'+'\' to client-
'+str(client_no))
                               i = i+2
                          else:
                               print('INVALID GET REQUEST!')
                               break
                     elif query[i] == 'put':
                          if i+2 < qlen:
                               key = query[i+1]
                               value = query[i+2]
                               i = i+3
                               while i < qlen:
```

```
                                        if query[i] == 'put' or query[i] ==
'get':
                                             break
                                        value += " "+query[i]
                                        i = i+1
                                   self.key_store[client_no][key] = value
                                   print('PUT REQUEST: Value of \''+key+'\'
as \''+value+'\' added to client-'+str(client_no)+'\'s key-value store')
                              else:
                                   print('INVALID PUT REQUEST!')
                                   break
                         else:
                              print('INVALID REQUEST FOUND!')
                    response.append("end")

                    for x in response:
                         time.sleep(.0005)
                         conn[0].send(x.encode())
                    conn[0].close()
                    print('-'*70)
                    print(end='\n\n')

if __name__ == '__main__':
     s = Server()
     s.init_authorization()
     s.process_query()
```

Client program (client.py): [Language: Python]

```
import socket
import sys
import time

class Client():
     def __init__(self):
          self.serverhost = '127.0.0.1'
          self.serverport = 8080

     def process_query(self, args):
          if args[2] != self.serverhost or int(args[3]) !=
self.serverport:
               print('ERROR: Invalid server host or port number
entered!')
               return
          sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
          sock.connect((args[2],int(args[3])))
          for x in args:
               time.sleep(.0005)
               sock.send(x.encode())
          sock.send("end".encode())
          data = sock.recv(1024).decode()
          while data:
               if data == 'end':
                    break
               print(data)
               data = sock.recv(1024).decode()
```

```
            sock.close()

if __name__ == '__main__':
      c = Client()
      num_args = len(sys.argv)
      if num_args < 4:
            print("ERROR: Invalid arguments provided!")
            exit()
      client_no = int(sys.argv[1])
      if client_no < 1 or client_no > 100:
            print("ERROR: Invalid client number provided!")
            exit()
      c.process_query(sys.argv)
```

- **Assignment 2**

Server program (server.js): [Language: Javascript, Framework used: Node.js]

```
var server = require('ws').Server;
var s = new server({ port : 8080 });
var dict = {};
var auth = {};
s.on('connection', function(ws){
      ws.on('message', function(message) {
            message = JSON.parse(message);

            if(message.type == "clientId") {
                  ws.clientId = message.data;
                  if(auth[ws.clientId] == "guest" || auth[ws.clientId] ==
"manager") {
                        console.log("ERROR: DUPLICATE CLIENT ID!");
                        ws.clientId = "duplicateuser";
                        ws.send(JSON.stringify({
                              type: "errormsg",
                              data: "DUPLICATE-CLIENT"
                        }));
                        ws.close();
                        return;
                  }
                  auth[ws.clientId] = "guest";
                  dict[ws.clientId] = {};
                  console.log("Client with clientId "+ws.clientId+"
connected!");
            }
            else if(message.type == "PUT") {
                  console.log("PUT REQUEST:");
                  dict[ws.clientId][message.key] = message.value;
                  console.log("The value of '"+message.key+"' as
'"+message.value+"' is inserted in "+ws.clientId+"'s key-store");
            }
            else if(message.type == "GET") {
                  console.log("GET REQUEST:");
                  if(auth[ws.clientId] == "guest") {
                        var val = dict[ws.clientId][message.key];
                        if(typeof val == "undefined") {
                              ws.send(JSON.stringify({
```

```
                                    type: "message",
                                    data: "blank"
                               }));
                               console.log(message.key+" could not be found
in "+ws.clientId+"'s key-store!");
                               console.log("Sending the value of
'"+message.key+"' as "+"<blank>");
                          } else {
                               ws.send(JSON.stringify({
                                    type: "message",
                                    data: val
                               }));
                               console.log("Sending the value of
'"+message.key+"' as '"+val+"' from "+ws.clientId+"'s key-store");
                          }
                     }
                     else if(auth[ws.clientId] == "manager") {
                          console.log(message.key+" could not be found in
"+ws.clientId+"'s key-store!");
                          console.log("Since the client is a manager,
Searching for the value in other client's key-stores");
                          var val = dict[ws.clientId][message.key];
                          if(typeof val == "undefined") {
                               var found = 0;
                               s.clients.forEach(function(client){
                                    var val =
dict[client.clientId][message.key];
                                    if(found == 0 && typeof val !=
"undefined") {
                                         found = 1;
                                         ws.send(JSON.stringify({
                                              type: "message",
                                              data: val
                                         }));
                                         console.log("Sending the value of
'"+message.key+"' as '"+val+"' from "+client.clientId+"'s key-store");
                                    }
                               });
                               if(found == 0) {
                                    ws.send(JSON.stringify({
                                         type: "message",
                                         data: "blank"
                                    }));
                                    console.log("Sending the value of
'"+message.key+"' as "+"<blank>");
                               }

                          } else {
                               ws.send(JSON.stringify({
                                    type: "message",
                                    data: val
                               }));
                               console.log("Sending the value of
'"+message.key+"' as '"+val+"' from "+ws.clientId+"'s key-store");
                          }
                     }

               }
```

```
            else if(message.type == "UPGRADE") {
                  if(auth[ws.clientId] == "manager") {
                        console.log(ws.clientId+" is already a manager!");
                  } else {
                        auth[ws.clientId] = "manager";
                        console.log(ws.clientId+" is now a manager!");
                  }
            }
            else if(message.type == "DOWNGRADE") {
                  if(auth[ws.clientId] == "guest") {
                        console.log(ws.clientId+" is already a guest!");
                  } else {
                        auth[ws.clientId] = "guest";
                        console.log(ws.clientId+" is now a guest!");
                  }
            }

      });

      ws.on('close', function() {
            auth[ws.clientId] = "";
            dict[ws.clientId] = {};
      });
});
```

Client program (client.html): [Language: HTML, Javascript]

```
<!DOCTYPE html>
<html lang='en'>
<head>
      <meta charset="UTF-8">
      <title>WebSocket client</title>
</head>
<body>

      <input type="text" placeholder="Enter key here" id="putKeyText">
      <input type="text" placeholder="Enter value here" id="putValueText">
      <button id="putButton">SEND PUT REQUEST</button>
      <br>
      <br>

      <input type="text" placeholder="Enter key here" id="getKeyText">
      <button id="getButton">SEND GET REQUEST</button>
      <br>
      <br>

      <button id="upgradeButton">UPGRADE TO MANAGER</button>
      <button id="downgradeButton">DOWNGRADE TO GUEST</button>
      <br>
      <br>

      <div id="log"> </div>

      <script>
            var clientId = prompt('What is your client id?');

            var sock = new WebSocket("ws://localhost:8080");
```

```
            sock.onopen = function() {
                  sock.send(JSON.stringify({
                        type: "clientId",
                        data: clientId
                  }));
            }

            var log = document.getElementById('log');

            sock.onmessage = function(event) {
                  var json = JSON.parse(event.data);
                  if(json.type == "errormsg" && json.data == "DUPLICATE-
CLIENT") {
                        alert("ERROR: DUPLICATE CLIENT ID! Please try again
with a different id");
                        return;
                  }
                  log.innerHTML += json.data+"<br>";
            }

            document.getElementById("putButton").onclick = function() {
                  var keyText =
document.getElementById('putKeyText').value;
                  if(keyText == "") {
                        alert("Please enter key while sending PUT request");
                        return;
                  }
                  var valueText =
document.getElementById('putValueText').value;
                  if(valueText == "") {
                        alert("Please enter value while sending PUT
request");
                        return;
                  }
                  sock.send(JSON.stringify({
                        type: "PUT",
                        key: keyText,
                        value: valueText
                  }));
            };

            document.getElementById("getButton").onclick = function() {
                  var keyText =
document.getElementById('getKeyText').value;
                  if(keyText == "") {
                        alert("Please enter key while sending PUT request");
                        return;
                  }
                  sock.send(JSON.stringify({
                        type: "GET",
                        key: keyText
                  }));
            };

            document.getElementById("upgradeButton").onclick = function()
{
                  sock.send(JSON.stringify({
```

```
                        type: "UPGRADE"
                  }));
            };

            document.getElementById("downgradeButton").onclick =
function() {
                  sock.send(JSON.stringify({
                        type: "DOWNGRADE"
                  }));
            };
      </script>

</body>
</html>
```

OUTPUT

- **Assignment 1**

Server program (server.py):

```
C:\Users\SOURAV\Desktop\it-lab\ass1>python server.py
Want to close the server (y/n)? n
Want to change the authorization to any client (y/n)? n
-----------------------------------------------------------------
        Connected with client 1
-----------------------------------------------------------------
PUT REQUEST: Value of 'city' as 'kolkata' added to client-1's key-value store
PUT REQUEST: Value of 'country' as 'india' added to client-1's key-value store
GET REQUEST: Sending the Value of 'country' as 'india' to client-1
GET REQUEST: Sending the Value of 'city' as 'kolkata' to client-1
GET REQUEST: Sending the Value of 'institute' as '<blank>' to client-1
-----------------------------------------------------------------


Want to close the server (y/n)? n
Want to change the authorization to any client (y/n)? n
-----------------------------------------------------------------
        Connected with client 2
-----------------------------------------------------------------
PUT REQUEST: Value of 'state' as 'west bengal' added to client-2's key-value store
PUT REQUEST: Value of 'institute' as 'jadavpur university' added to client-2's key-value store
GET REQUEST: Sending the Value of 'state' as 'west bengal' to client-2
GET REQUEST: Sending the Value of 'institute' as 'jadavpur university' to client-2
-----------------------------------------------------------------


Want to close the server (y/n)? n
Want to change the authorization to any client (y/n)? y
Enter client number and role (manager/guest): 1 manager
Client-1 is now a 'manager' !!
Want to change the authorization to any client (y/n)? n
-----------------------------------------------------------------
        Connected with client 2
```

GET REQUEST: Sending the Value of 'institute' as 'jadavpur university' to client-2

Want to close the server (y/n)? y

Client programs (client.py):

[Executing client 1]

C:\Users\SOURAV\Desktop\it-lab\ass1>python client.py 1 127.0.0.1 8080 put city kolkata put country india get country get city get institute
india
kolkata
<blank>

[Executing client 2]

C:\Users\SOURAV\Desktop\it-lab\ass1>python client.py 2 127.0.0.1 8080 put state west bengal put institute jadavpur university get state get institute
west bengal
jadavpur university

[Executing client 1]

C:\Users\SOURAV\Desktop\it-lab\ass1>python client.py 2 127.0.0.1 8080 get institute
jadavpur university

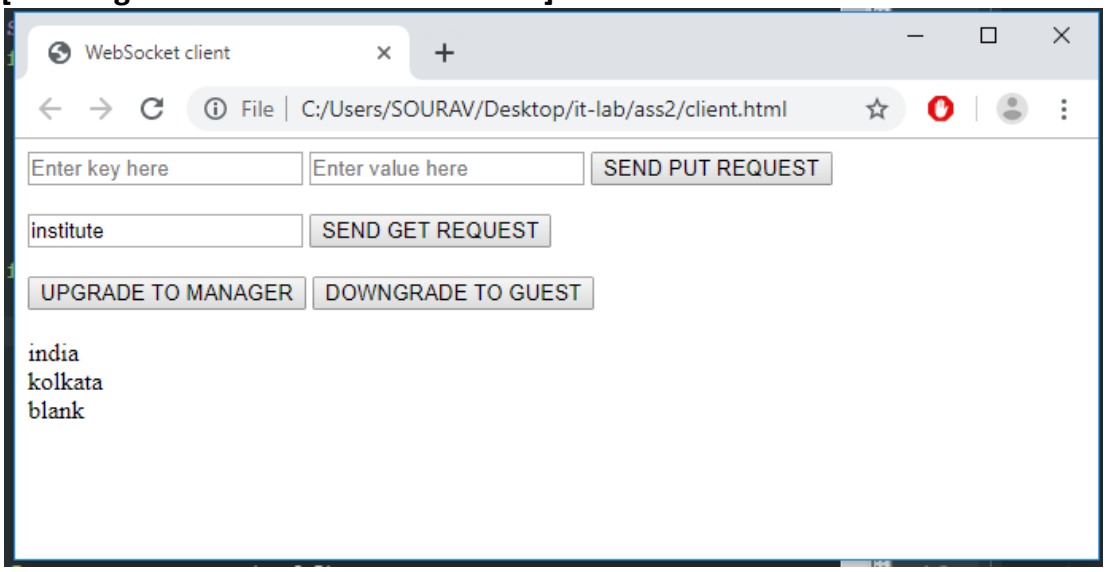
- **Assignment 2**

Server program (server.js):

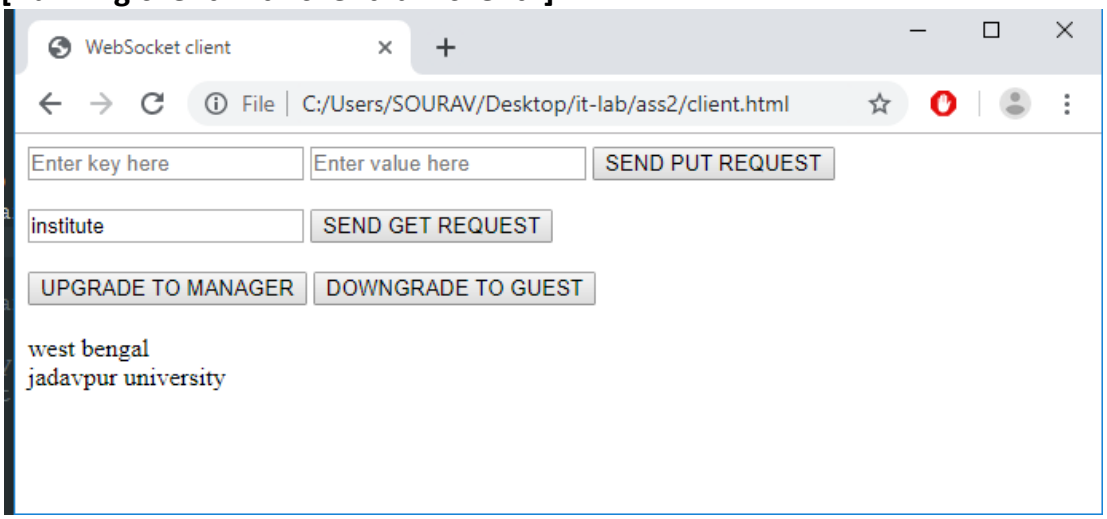
C:\Users\SOURAV\Desktop\it-lab\ass2>node server.js
Client with clientId client1 connected!
Client with clientId client2 connected!
PUT REQUEST:
The value of 'country' as 'india' is inserted in client1's key-store
PUT REQUEST:
The value of 'city' as 'kolkata' is inserted in client1's key-store
GET REQUEST:
Sending the value of 'country' as 'india' from client1's key-store
GET REQUEST:
Sending the value of 'city' as 'kolkata' from client1's key-store
GET REQUEST:
institute could not be found in client1's key-store!
Sending the value of 'institute' as <blank>
PUT REQUEST:
The value of 'institute' as 'jadavpur university' is inserted in client2's key-store
PUT REQUEST:
The value of 'state' as 'west bengal' is inserted in client2's key-store
GET REQUEST:
Sending the value of 'state' as 'west bengal' from client2's key-store
GET REQUEST:
Sending the value of 'institute' as 'jadavpur university' from client2's key-store
client1 is now a manager!
GET REQUEST:
institute could not be found in client1's key-store!
Since the client is a manager, Searching for the value in other client's key-stores
Sending the value of 'institute' as 'jadavpur university' from client2's key-store

Client programs (client.html):

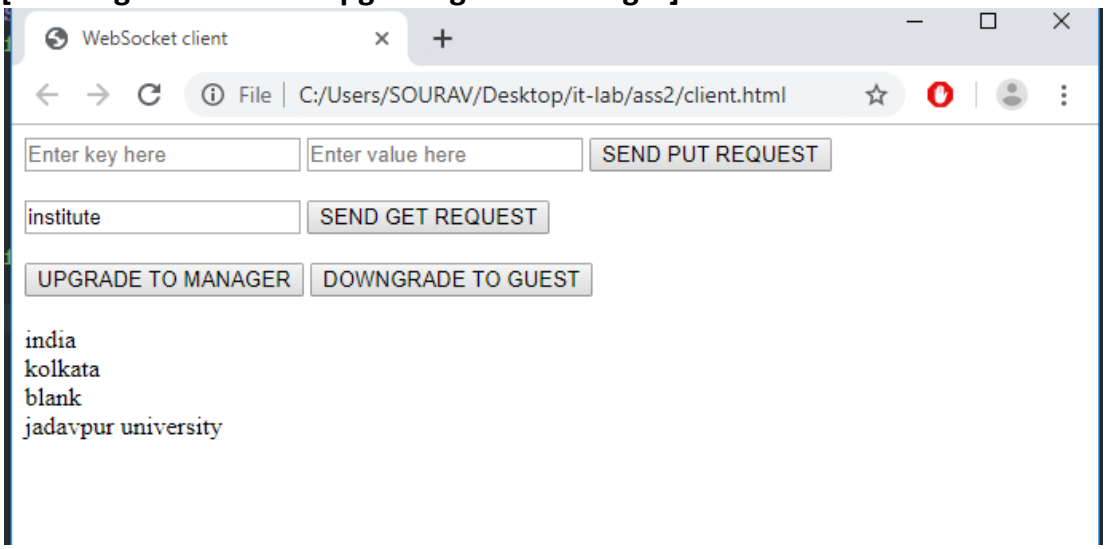
[Running client with client id – client1]



[Running client with client id – client2]



[Running client1 after upgrading it as manager]



COMPARATIVE ANALYSIS

WebSockets represent a long awaited evolution in client/server web technology. They allow a long-held single TCP socket connection to be established between the client and server which allows for bi-directional, full duplex, messages to be instantly distributed with little overhead resulting in a very low latency connection.

Both the WebSocket API and the well as native WebSocket support in browsers such as Google Chrome, Firefox, Opera and a prototype Silverlight to JavaScript bridge implementation for Internet Explorer, there are now WebSocket library implementations in Objective-C, .NET, Ruby, Java, node.js, ActionScript and many other languages.

Why WebSockets are better:
WebSockets represent a standard for bi-directional realtime communication between servers and clients. Firstly in web browsers, but ultimately between any server and any client. The standards first approach means that as developers we can finally create functionality that works consistently across multiple platforms. Connection limitations are no longer a problem since WebSockets represent a single TCP socket connection. Cross domain communication has been considered from day one and is dealt with within the connection handshake. This means that services such as Pusher can easily use them when offering a massively scalable realtime platform that can be used by any website, web, desktop or mobile application. That is why WebSockets perform better when "**manager**" role is assigned to it, than regular TCP sockets. Whereas when "**guest**" role is assigned to a client, both performs equally well, but in WebSockets, we do not need to take care of multiprogramming or threading of the client processes.

WebSockets typically run from browsers connecting to Application Server over a protocol similar to HTTP that runs over TCP/IP. So they are primarily for Web Applications that require a permanent connection to its server. On the other hand, plain sockets are more powerful and generic. They run over TCP/IP but they are not restricted to browsers or HTTP protocol. They could be used to implement any kind of communication.