PROJECT 1

October 5th, 2018

ECE/CS 5565

Network Architecture and Protocols

Ajit Sarkaar

9062-13662

Python 2.7

Sublime Text 3

macOS High Sierra 10.13.6

Implemention Discussion

The goal of this project is to implement a client server application. A client/server application consists of a client program that consumes services provided by a server program. The client requests services from the server by calling functions in the server application. Usually, the client program and the server program execute on different machines and possibly even on different platforms, and the client and server communicate through a communications layer which includes protocols or set of well defined rules.

For this project, both, the client object and the server object were implemented using python version 2.7 and tested by creating instances on the terminal. The code editor of choice was Sublime Text 3. The APIs used in the implementation were the OS API, SYS API and the Socket API available in python. Sending of text data and images across server and client was successfully implemented for this project. The client server application was built to share both text and image data.

Implementation of Server:

1. Get current working directory.
2. Ask the user to specify the IP address and port number on which the server should be set up.
3. Define the status responses which will be sent to clients.
4. Create a socket instance and bind it to a host and port number.
5. Start listening for incoming client requests.
6. Once a client is connected, process it’s request and find out what type of resource the client is requesting for. Also check for the correct format of the request before proceeding and send a bad request response if format is incorrect and close the connection.
7. Check if resource present on the server, otherwise send a resource not found response.
8. If resource is present, send a status OK response.
9. Divided the resource into chunks and send those chunks sequentially.
10. Close the connection with the client.
11. Serve the next client in queue or wait for incoming connections.

Implementation of Client:

1. Get current working directory.
2. Ask the user for the IP address and port number of the server.
3. Create a socket instance and start connection to the server.
4. Ask the user to specify what resource should be fetched from the server.
5. Create and send a request to the server to ask for the specified resource.
6. Receive the resource from the server if the server sends a OK response.
7. Ask the user to specify the name for the resource to be saved as.
8. Save the fetched resource locally on the client machine.

9. Close the connection if the server has to ended the connection or if the resource has been fetched and saved locally.

Testing Results

Test Procedure:

Combinations tested for the implemented server and client(+ = with):

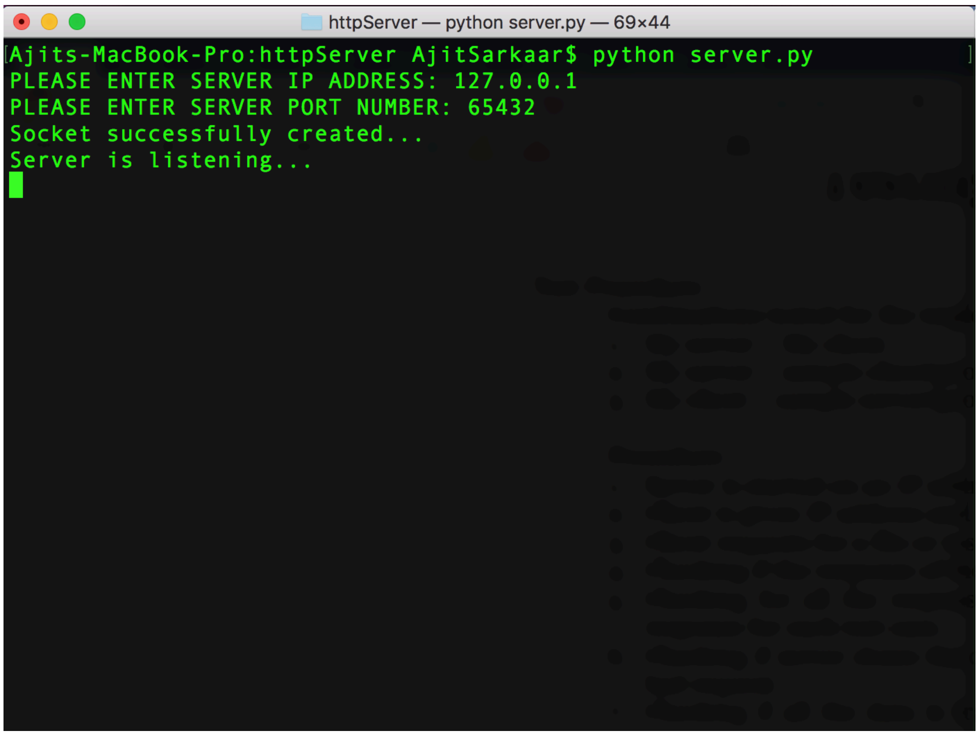
1. My server + My client.
2. My server + sample client(provided).
3. My client + sample server(provided).

Procedure:

1. Server is started on an IP address and a port number.
2. Client is given IP address and port number to connect to the server.
3. Client requests for a file or resource by sending a request.
4. Checking if the response of the server is correct or not.
5. Try checking for each test case and see if server gives all three outputs for the appropriate test case. Check for all response types(200, 404, and 400).
6. Checking if server closes the connection in case of resource not present or bad request.
7. Checking if all the data is transferred from server to client without any errors.
8. Checking if different types of data such as text or images can be shared between server and client.
9. Testing both client and server being on different machines.

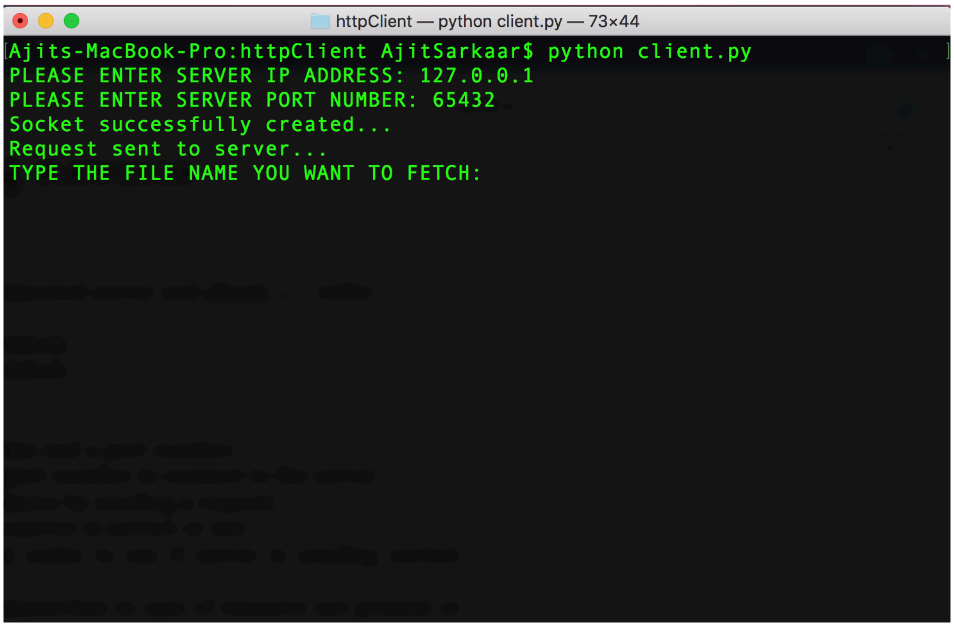
Results:

1. My Server with my client:



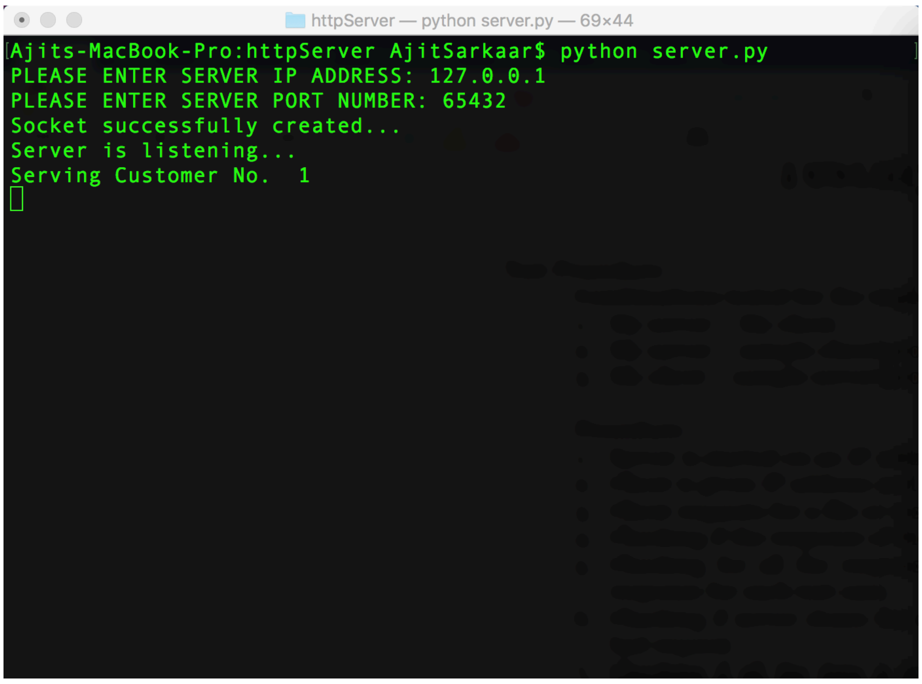
```
httpServer — python server.py — 69×44
Ajits-MacBook-Pro:httpServer AjitSarkaar$ python server.py
PLEASE ENTER SERVER IP ADDRESS: 127.0.0.1
PLEASE ENTER SERVER PORT NUMBER: 65432
Socket successfully created...
Server is listening...
```

Server is running and listening for connections.



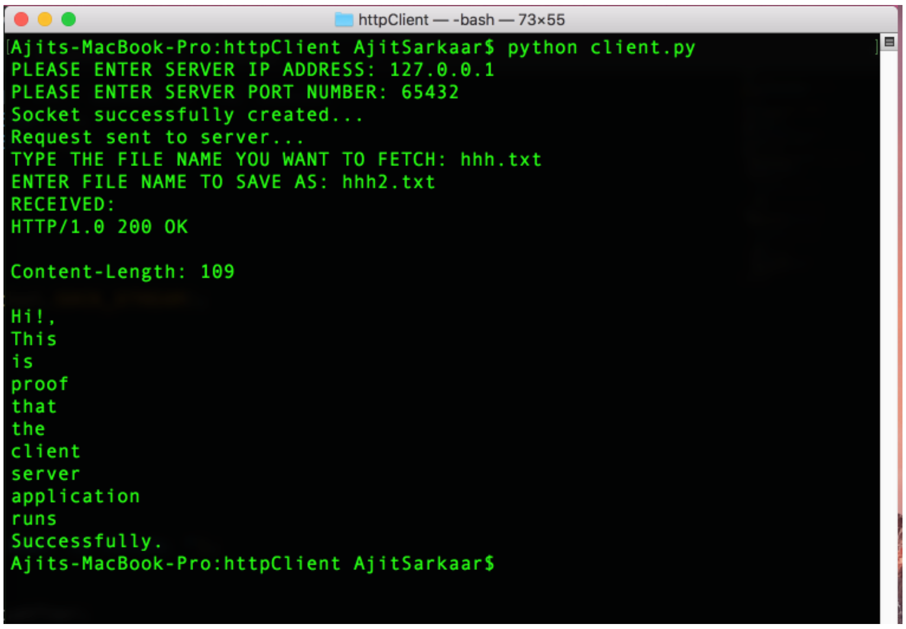
```
Ajits-MacBook-Pro:httpClient AjitSarkaar$ python client.py
PLEASE ENTER SERVER IP ADDRESS: 127.0.0.1
PLEASE ENTER SERVER PORT NUMBER: 65432
Socket successfully created...
Request sent to server...
TYPE THE FILE NAME YOU WANT TO FETCH:
```

Client connects to the server and asks the user for the file or resource to fetch.



```
Ajits-MacBook-Pro:httpServer AjitSarkaar$ python server.py
PLEASE ENTER SERVER IP ADDRESS: 127.0.0.1
PLEASE ENTER SERVER PORT NUMBER: 65432
Socket successfully created...
Server is listening...
Serving Customer No.  1
```

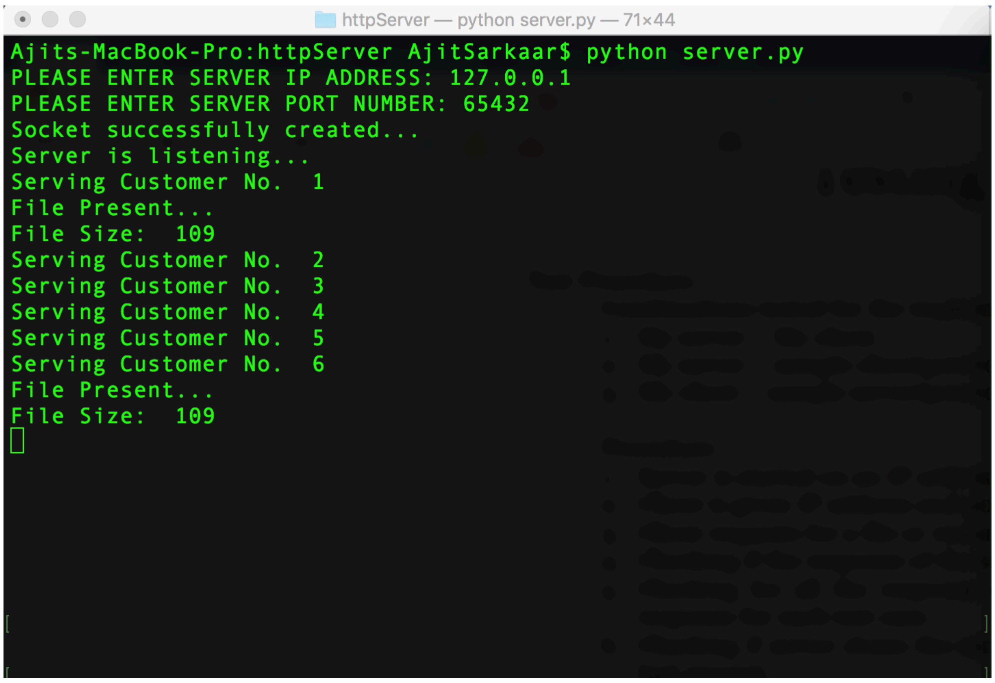
Server starts serving client 1



```
Ajits-MacBook-Pro:httpClient AjitSarkaar$ python client.py
PLEASE ENTER SERVER IP ADDRESS: 127.0.0.1
PLEASE ENTER SERVER PORT NUMBER: 65432
Socket successfully created...
Request sent to server...
TYPE THE FILE NAME YOU WANT TO FETCH: hhh.txt
ENTER FILE NAME TO SAVE AS: hhh2.txt
RECEIVED:
HTTP/1.0 200 OK

Content-Length: 109

Hi!,
This
is
proof
that
the
client
server
application
runs
Successfully.
Ajits-MacBook-Pro:httpClient AjitSarkaar$
```

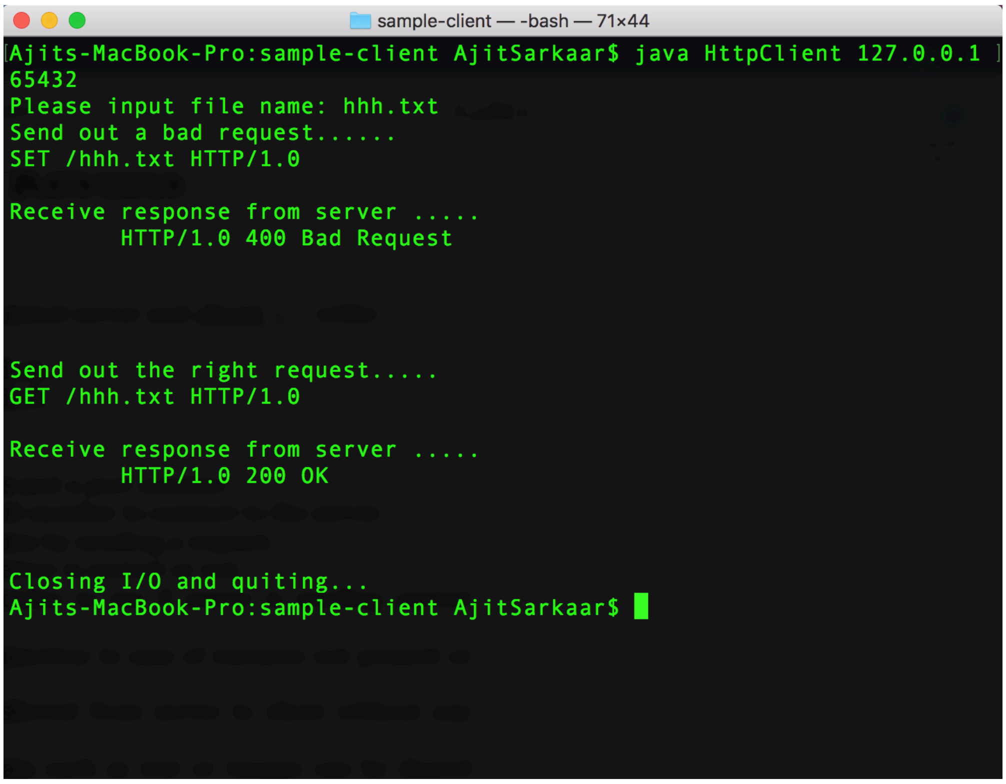
Client requests for a file and receives it. The client then saves the file locally.

2. My Server with sample client:



```
Ajits-MacBook-Pro:httpServer AjitSarkaar$ python server.py
PLEASE ENTER SERVER IP ADDRESS: 127.0.0.1
PLEASE ENTER SERVER PORT NUMBER: 65432
Socket successfully created...
Server is listening...
Serving Customer No.  1
File Present...
File Size:  109
Serving Customer No.  2
Serving Customer No.  3
Serving Customer No.  4
Serving Customer No.  5
Serving Customer No.  6
File Present...
File Size:  109
```

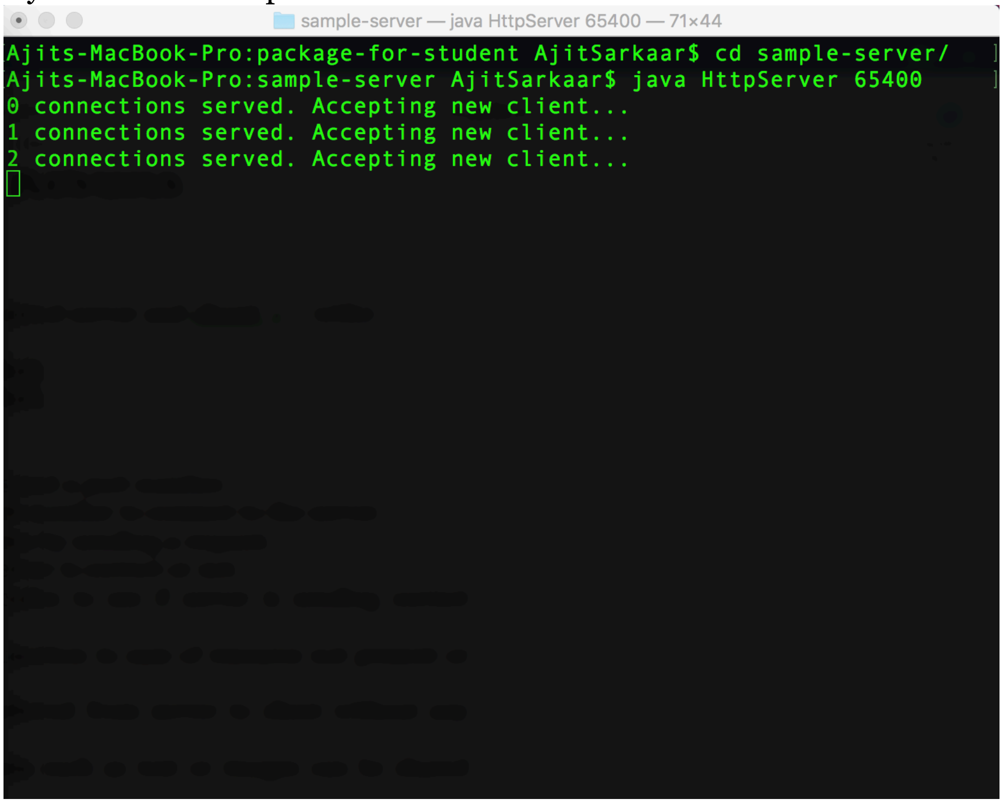
My server serving the sample client and sending the text file.



```
[Ajits-MacBook-Pro:sample-client AjitSarkaar$ java HttpClient 127.0.0.1 ]
65432
Please input file name: hhh.txt
Send out a bad request......
SET /hhh.txt HTTP/1.0

Receive response from server .....
        HTTP/1.0 400 Bad Request



Send out the right request.....
GET /hhh.txt HTTP/1.0

Receive response from server .....
        HTTP/1.0 200 OK



Closing I/O and quiting...
Ajits-MacBook-Pro:sample-client AjitSarkaar$ 
```

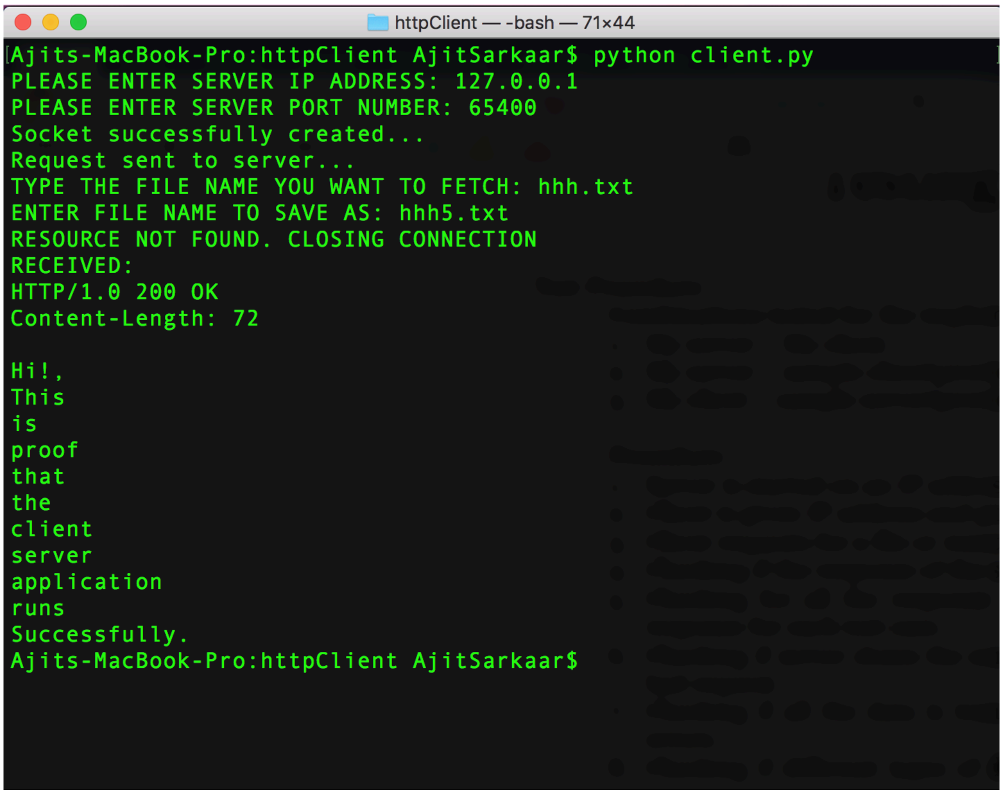
sample client requesting for a file and receiving the 200 response.

2. My client with sample server



```
Ajits-MacBook-Pro:package-for-student AjitSarkaar$ cd sample-server/   ]
Ajits-MacBook-Pro:sample-server AjitSarkaar$ java HttpServer 65400     ]
0 connections served. Accepting new client...
1 connections served. Accepting new client...
2 connections served. Accepting new client...
```

sample server serving requests of my client.

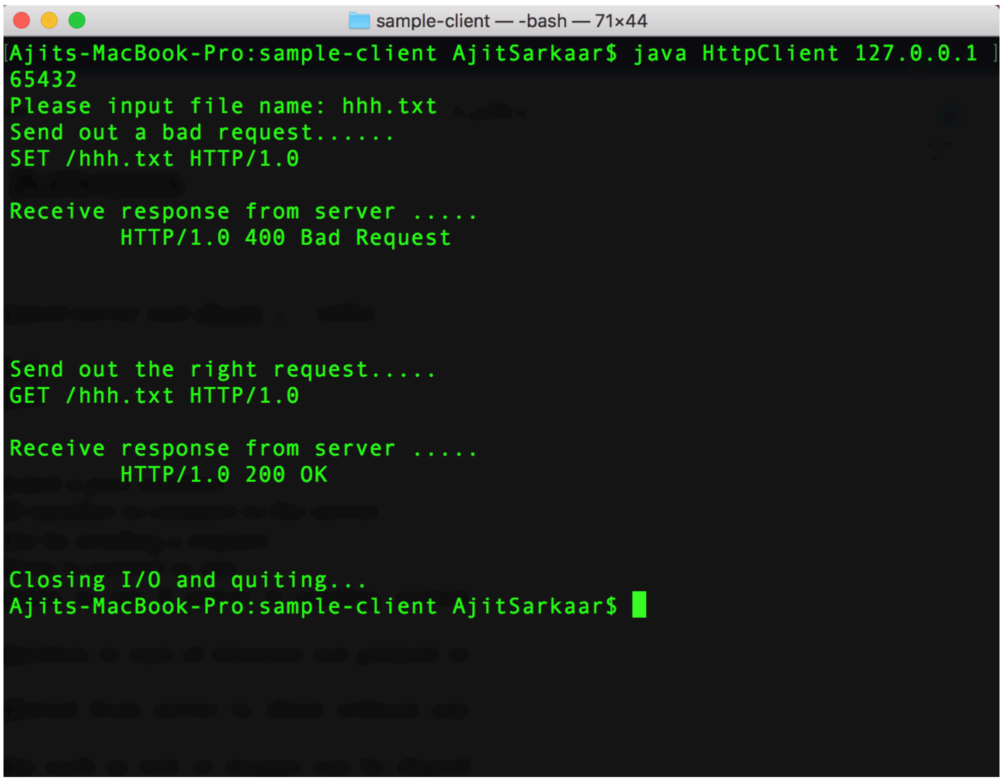


```
[Ajits-MacBook-Pro:httpClient AjitSarkaar$ python client.py
PLEASE ENTER SERVER IP ADDRESS: 127.0.0.1
PLEASE ENTER SERVER PORT NUMBER: 65400
Socket successfully created...
Request sent to server...
TYPE THE FILE NAME YOU WANT TO FETCH: hhh.txt
ENTER FILE NAME TO SAVE AS: hhh5.txt
RESOURCE NOT FOUND. CLOSING CONNECTION
RECEIVED:
HTTP/1.0 200 OK
Content-Length: 72

Hi!,
This
is
proof
that
the
client
server
application
runs
Successfully.
Ajits-MacBook-Pro:httpClient AjitSarkaar$
```

[redacted]

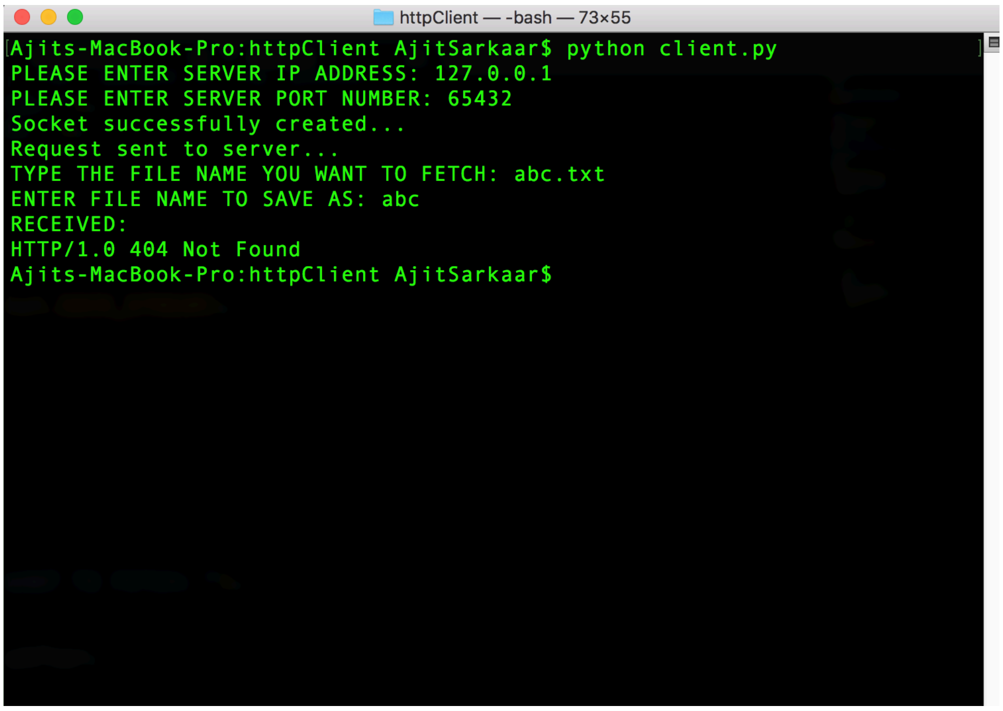
My client receiving files from the samle server.

3. Checking for all responses of my server using the sample client:



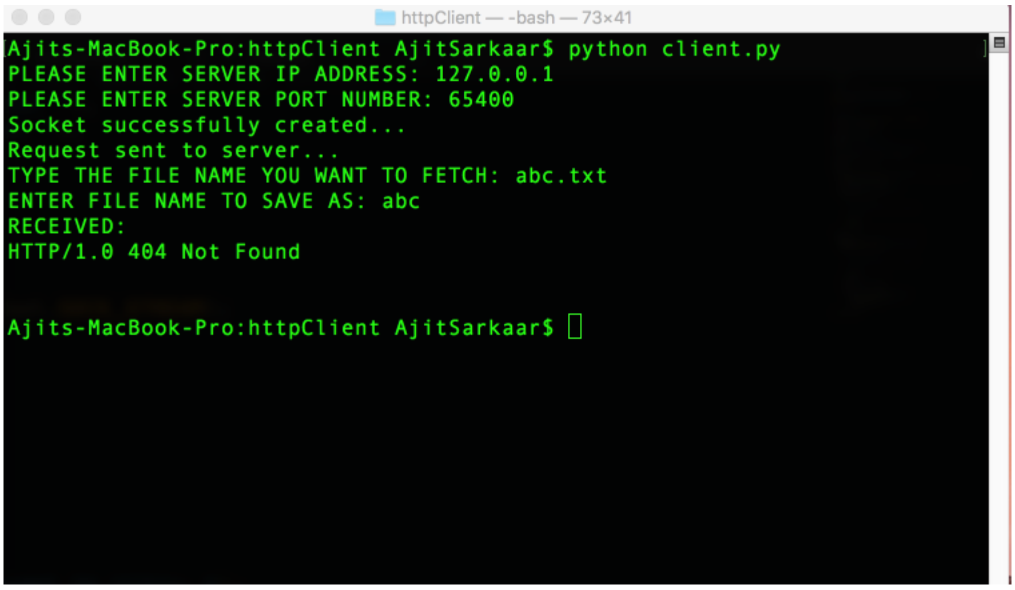
```
[Ajits-MacBook-Pro:sample-client AjitSarkaar$ java HttpClient 127.0.0.1
65432
Please input file name: hhh.txt
Send out a bad request......
SET /hhh.txt HTTP/1.0

Receive response from server .....
        HTTP/1.0 400 Bad Request


[redacted]


Send out the right request.....
GET /hhh.txt HTTP/1.0

Receive response from server .....
        HTTP/1.0 200 OK
[redacted]


Closing I/O and quiting...
Ajits-MacBook-Pro:sample-client AjitSarkaar$ █

[redacted]
```

My server returning 400 and 200 response codes.



```
Ajits-MacBook-Pro:httpClient AjitSarkaar$ python client.py
PLEASE ENTER SERVER IP ADDRESS: 127.0.0.1
PLEASE ENTER SERVER PORT NUMBER: 65432
Socket successfully created...
Request sent to server...
TYPE THE FILE NAME YOU WANT TO FETCH: abc.txt
ENTER FILE NAME TO SAVE AS: abc
RECEIVED:
HTTP/1.0 404 Not Found
Ajits-MacBook-Pro:httpClient AjitSarkaar$
```

My server returning 404 not found response.



```
Ajits-MacBook-Pro:httpClient AjitSarkaar$ python client.py
PLEASE ENTER SERVER IP ADDRESS: 127.0.0.1
PLEASE ENTER SERVER PORT NUMBER: 65400
Socket successfully created...
Request sent to server...
TYPE THE FILE NAME YOU WANT TO FETCH: abc.txt
ENTER FILE NAME TO SAVE AS: abc
RECEIVED:
HTTP/1.0 404 Not Found


Ajits-MacBook-Pro:httpClient AjitSarkaar$
```

My client receiving 404 not found from sample server.

Results Summary:

As the test results show, the server successfully parses the request and sends the appropriate response. The server also sends the text or the image data to the client successfully.

The client is able to generate a request and connect to the server to ask for a resource. The client also saves the received files locally successfully.