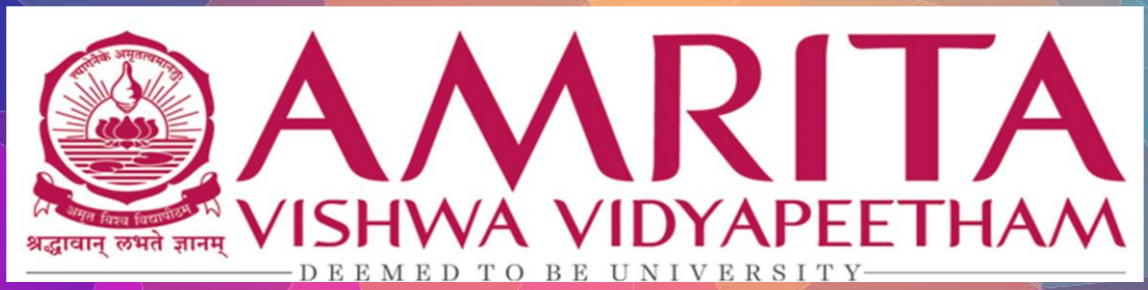
CLIENT SERVER APPLICATION USING HTTP

19CCE304 COMPUTER NETWORKS



Team Project -2

Team Number: 2

Members:

Sneha B - CB.EN.U4CCE19012

Dhanakarthik E - CB.EN.U4CCE19019

Harini V - CB.EN.U4CCE19025

Karthikeyan J - CB.EN.U4CCE19027

Authiselvi M - CB.EN.U4CCE19032

19CCE304 Computer Networks

TABLE OF CONTENTS

THEORY:

Networking: It is the process of connecting two more than two computers with the purpose to share data, provide technical support, and to communicate. Through internet we can connect different computer systems from different locations.

Web Page: A web page or webpage is a document, commonly written in HTML, that is viewed in an Internet browser. A web page can be accessed by entering a URL address into a browser's address bar. A web page may contain text, graphics, and hyperlinks to other web pages and files.

The Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) it transmits hypermedia documents (HTML). It was initially designed in order to communicate between web browsers and web servers. HTTP is a stateless protocol, meaning that the server does not keep any data (state) between two requests.

HTTP Protocol Properties

- ✓ It is an application layer protocol.
- ✓ Http protocol is considered to be a classical client- server with client request followed by waiting and then the connection is laid down.
- ✓ HTTP is **stateless protocol:** None of the data (or state) is not stored by the server.
- ✓ HTTP may be stateless but it is not session-less thus, allowing users to interact with pages coherently. Using headed extensions, HTTP Cookies are added to the workflow, hence allowing session to be created for HTTP request to share the same state.
- ✓ **Caching:** Cache can be achieved by instructing proxies and clients, to hold cache for a duration of time.
- ✓ **Relaxing the origin** This prevents snooping (privacy invasions). Only **same origin pages** are allowed to access all the information of a Web page. HTTP headers can relax strict separation on server side,

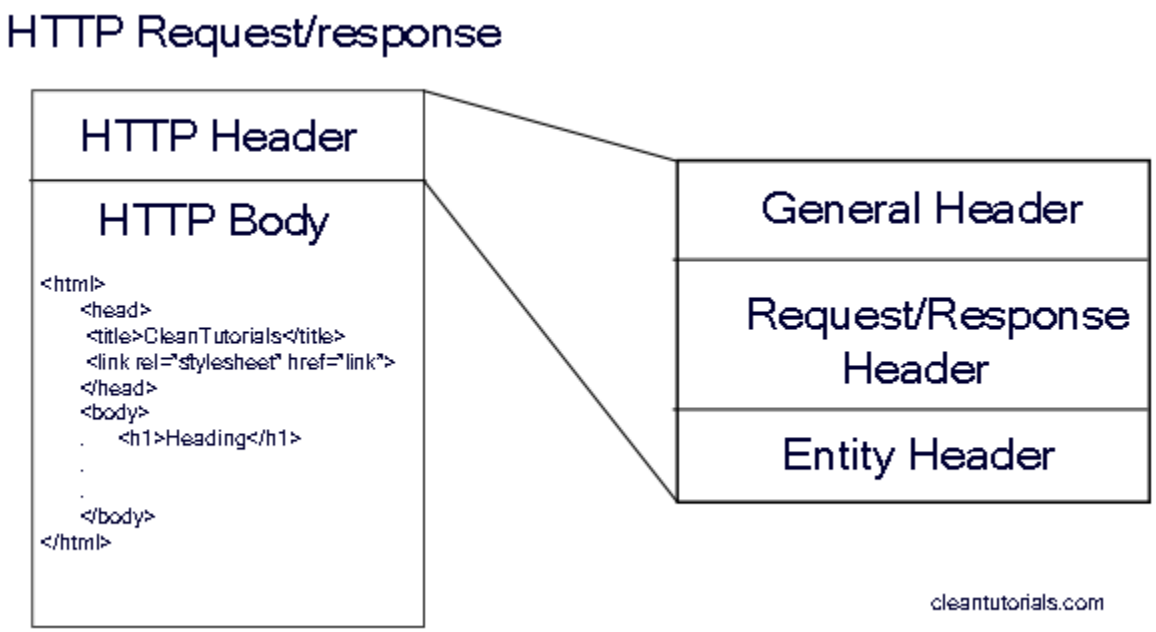
- ✓ Security is also considered while relaxing the origin of pages.
- ✓ **Authentication** of pages, so that only specific users can access them this is provided by HTTP.

HTTP header:

The header has fields to provide information regarding a request or response, or about the data in the message.

There are **four types** of HTTP message headers:

- **General-header:** The general fields have applicability for both request and response messages.
- **Client Request-header:** The Client Request-header has fielded the has applicability only for request messages.
- **Server Response-header:** The server Response-header has fields that have applicability only for response messages.
- **Entity-header:** The entity-header has fields that defines information (meta info) about entity-body or if body is present, about the type of request.



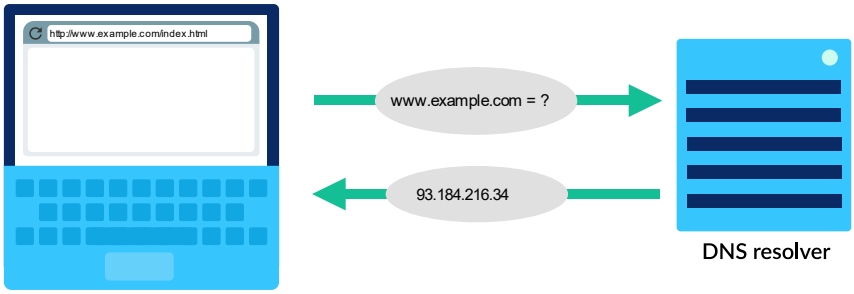
-

Transmitting Using HTTP

Step 1: Direct browser to URL

- While browsing the web, several types of computers (like laptops, desktops, and phones), as far as it has a **browser** application installed it can be used.
- Either a Uniform Resource Locator (URL) in the browser or follows a link from an already opened page is followed.

Step 2: Browser looks up IP



- Domain names map to IP addresses which gives the true location of the domain's computers. It is handled by the Domain Name System.

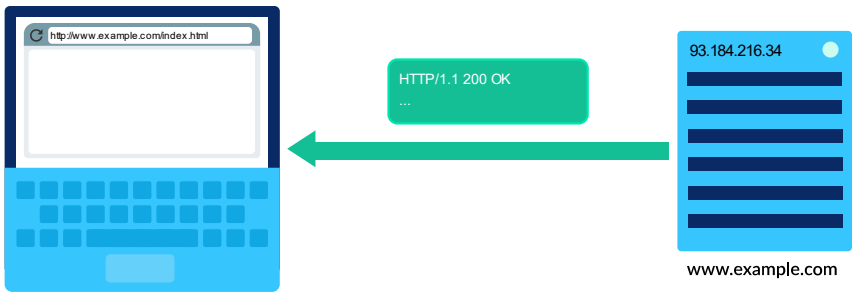
- The browser employs **DNS resolver** to map the domain to an IP address

Step 3: Browser sends HTTP request

- After Ip identification of IP an **HTTP request** is sent

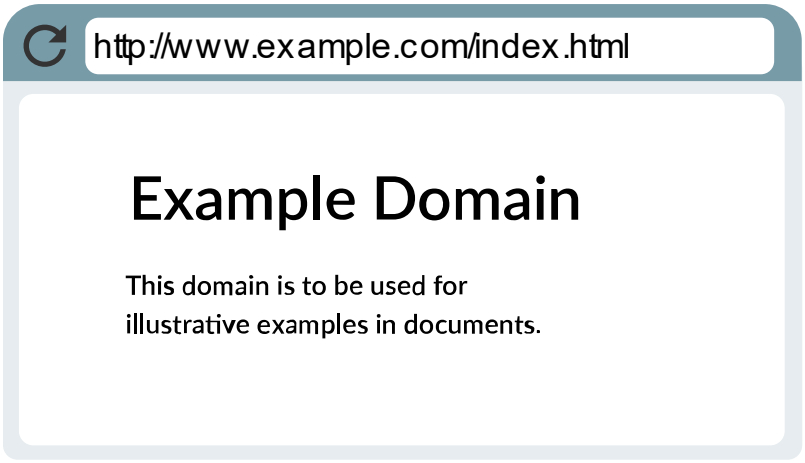
- An HTTP request can be as short as two lines of text: The first word is the HTTP verb: "GET". There are other verbs for other actions on the web, like submitting form data ("POST").

Step 4: Host sends back HTTP response



Once the host computer receives the HTTP request, it sends back a response with both the content and metadata about it.

Step 5: The browser renders the response



The browser has the necessary information it needs in order to request the document.

SOURCE CODE:

SERVER: This server sends the content of files (.html or .txt) stored in server to client on request.

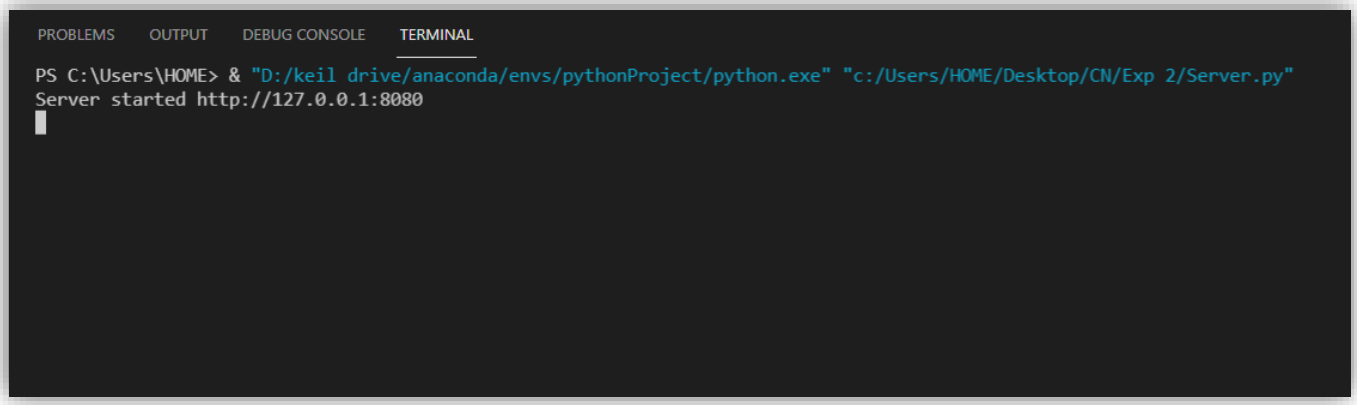
```
#HTTP SERVER
#Imports
from http.server import BaseHTTPRequestHandler, HTTPServer

#Create custom HTTPRequestHandler class
class WebServerHandler(BaseHTTPRequestHandler):

    #handle GET command
    def do_GET(self):
        rootdir = r'C:/Users/HOME/Desktop/CN/Exp 2/' #file location (give html files
location to send to client)
        try:
            if self.path.endswith(".html") or self.path.endswith(".txt"):
                f = open(rootdir + self.path) #open requested file

                self.send_response(200)  #send code 200 response

                self.send_header('Content-type', 'text/html')  #send header first
                self.end_headers()

                #send file content to client
                self.wfile.write(bytes("<html><head>THE GROUP-2</title></head>", "utf-8"))
                self.wfile.write(bytes("<p>Request: %s</p>" % self.path, "utf-8"))
                self.wfile.write(bytes("<body>", "utf-8"))
                self.wfile.write(bytes("<p>Content in Requested File:</p>", "utf-8"))
                self.wfile.write(bytes(f"<p>{f.read()}</p>", "utf-8"))
                self.wfile.write(bytes("</body></html>", "utf-8"))
                f.close()
                return

        # If file not Found
        except IOError:
            self.send_error(404, 'File Not Found: %s' % self.path)
def main():
    try:
        host = '127.0.0.1'
        port = 8080
        server = HTTPServer((host, port), WebServerHandler)
        print("Server started http://%s:%s" % (host, port))
        server.serve_forever()  # Starting the server
    except KeyboardInterrupt:
        print (" ^C entered, stopping web server....!") #To stop the server
        server.socket.close()

if __name__ == '__main__':
    main()
```

CLIENT: Requesting contents of file from Server

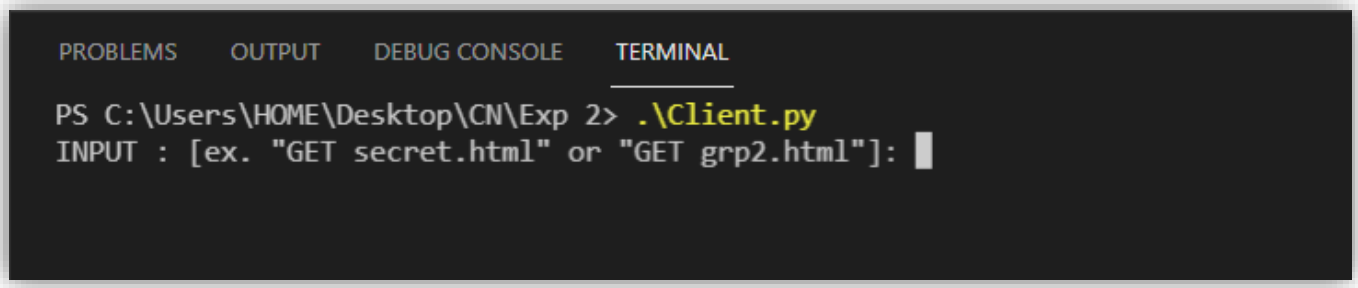
```
#HTTP Client

#Imports
import http.client

#get http server ip
http_server = "127.0.0.1"

#create a connection
conn = http.client.HTTPConnection(http_server,8080)



while True:
    cmd = input('INPUT : [ex. "GET secret.html" or "GET grp2.html"]: ')
    cmd = cmd.split()

    if cmd[0] == 'end': #type "end" to end client side connection
        break

    #request command to server
    conn.request(cmd[0], cmd[1])

    #get response from server
    rsp = conn.getresponse()

    #print server response and data
    print(rsp.status, rsp.reason)
    data_received = rsp.read()
    print(data_received)

conn.close()
```

Screenshot of Inputs and Output:

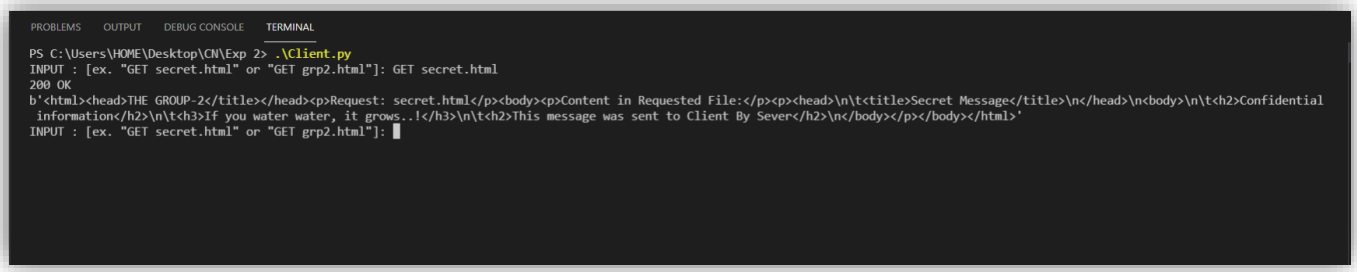
1. Starting the Server:



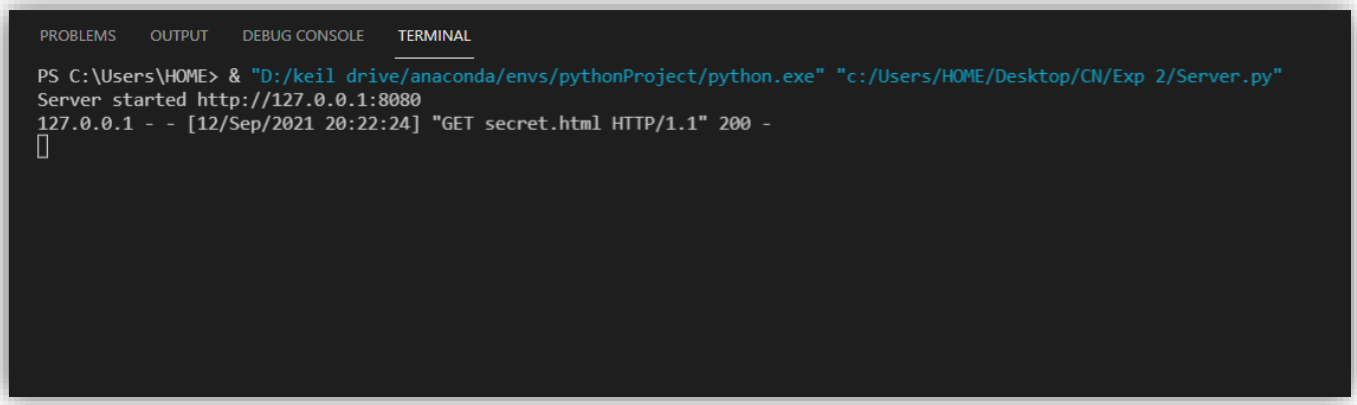
2. Starting the Client:



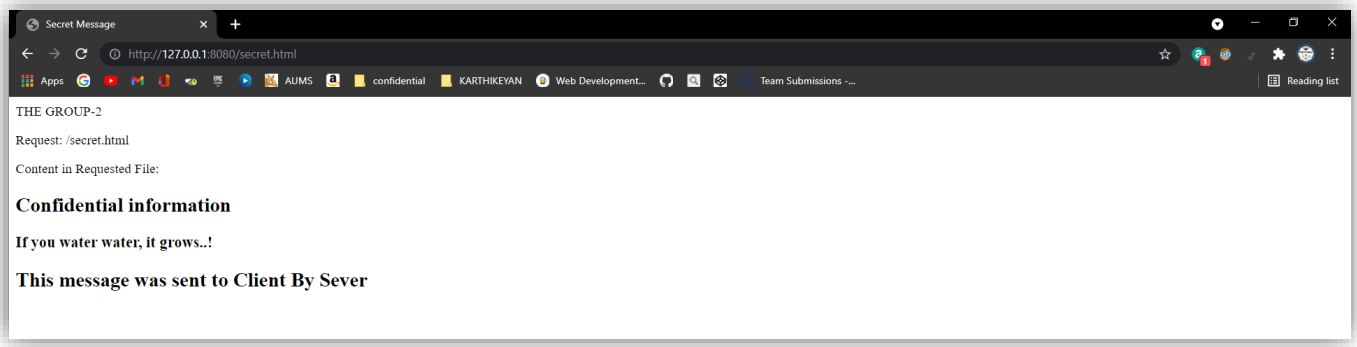
3. Requesting contents of file in server from client:



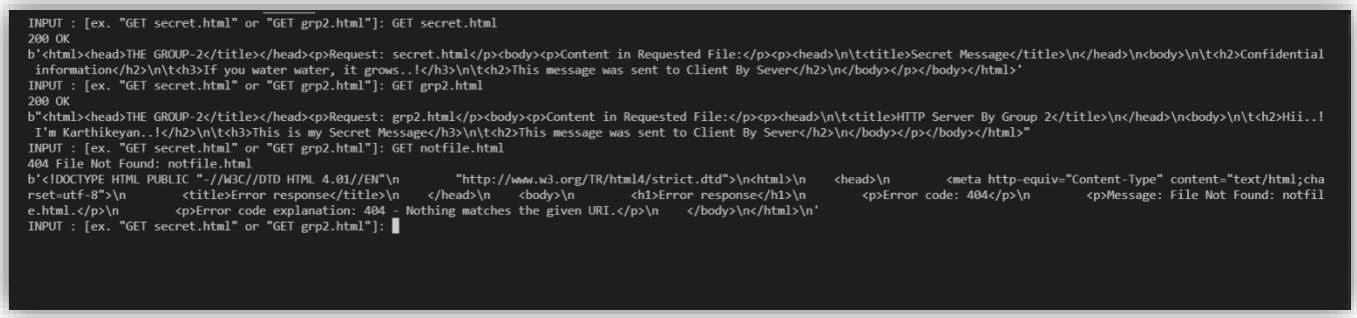
4. On Server Side:



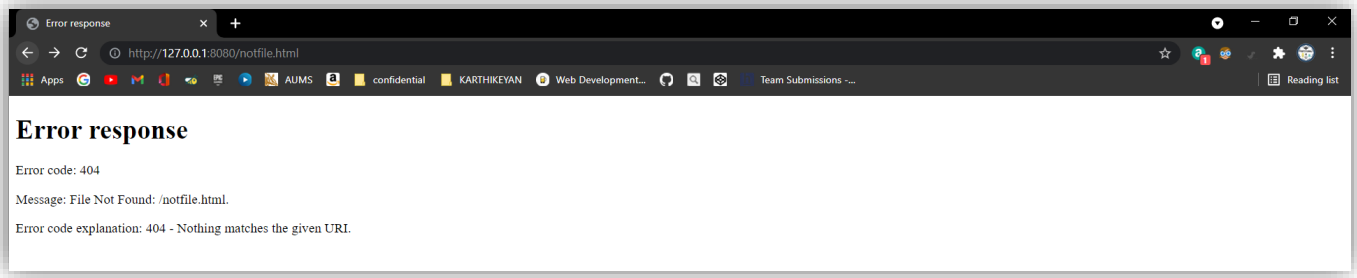
5. On Web:



6. On Request of file which is not available in server (Client):



7. On Request of file which is not available in server (Web Server):



Conclusion:

The web Server has been successfully created using Python. When client requests content of file in the server, the Server sends the data to client. We can navigate to the webpage using URL with the server Address. When we paste in the browser, the created webpage is loaded and contents in the web server (created on the local host) is displayed. Upon client requests we get the HTML response.

References:

Books:

Foundations of Python 3 Network Programming, Second Edition by John Goerzen, Brandon Rhodes.

Foundations of Python network Programming by Beaulne, Alexandre Goerzen, John Membrey, Peter Rhodes, Brandom 2014.