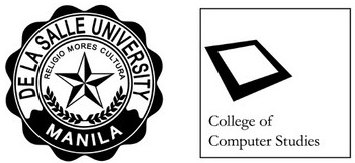
**De La Salle University**

**College of Computer Studies**

**CSNETWK Machine Project – File Exchange System**



|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Dy, Sealtiel B. | Section: | S13 |
| Name: | Lasala, Kyle Carlo C. | Section: | S13 |
| Name: | Manlises, Maria Monica | Section: | S13 |

# Machine Project – File Exchange System

**Overview**

The program is a simple web server made using Python that can handle one HTTP at any one moment. This server is tasked with receiving and interpreting the HTTP request, fetching the requested file from its file system, creating an HTTP response with header lines followed by the requested file, and then sending this response directly to the client. If the requested file is not found, the server should send a "404 Not Found" message to the client.

**Program**

A screenshot of a computer

Description automatically generated

Figure Import Socket and Sys Module

Figure 1 shows the importing of necessary modules such as the socket module and sys module. The socket module is used to create sockets and the sys module is used to be able to terminate the program.

A screen shot of a computer program

Description automatically generated

Figure 2 Creation and Preparation of Server Socket

Figure 2 shows the creation of the server socket. The socket was then bound to port 6969 in preparation for listening to incoming connection requests. The server was bound to the address **‘ ’** so that it will listen to all available network interfaces in the machine such as ‘localhost’ or the IP address of the machine. Furthermore the ‘1’ argument in the .listen() function means that there is a maximum of 1 queued connection.

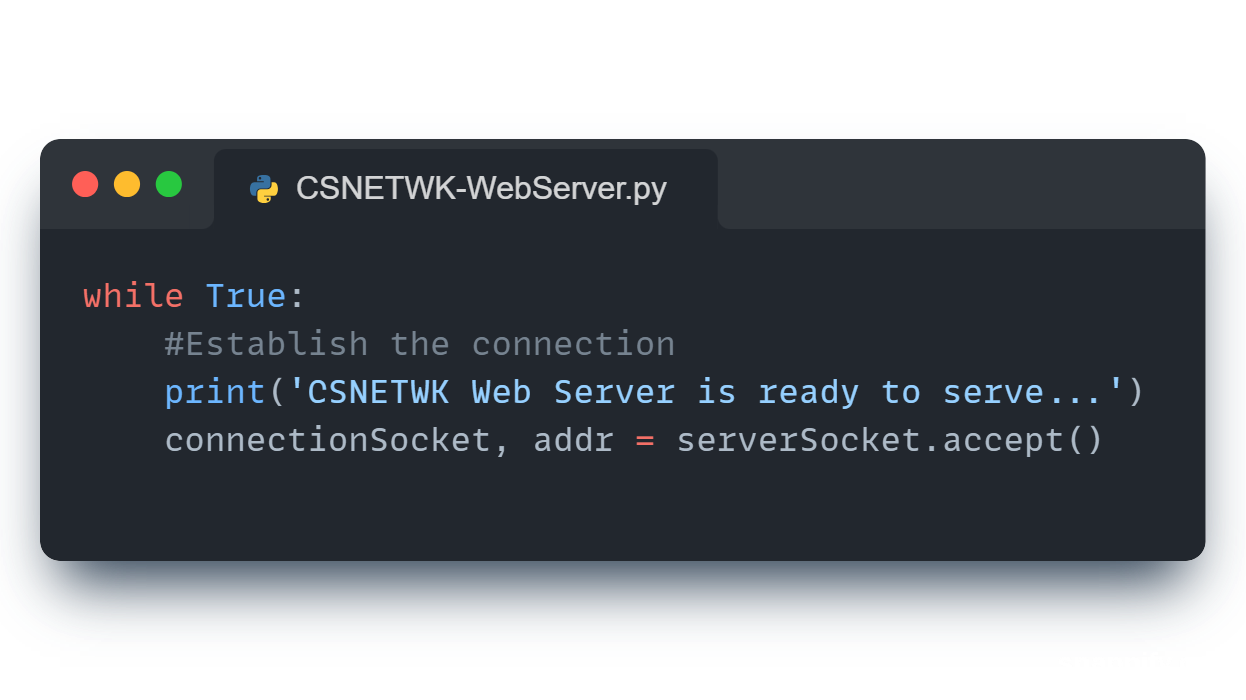


Figure 3 Waiting for Connection Requests

Figure 3 shows that the web server is waiting for connection requests in order to accept it.

A screen shot of a computer screen

Description automatically generated

Figure 4 Receive Request and Read File

Figure 4 shows that the server decodes the received request to extract the requested filename and then reads the file. If the file does not exist, it will go to the except block.

A screenshot of a computer

Description automatically generated

Figure 5 Respond with One HTTP Header Line

Figure 5 shows that the server responds by sending one HTTP header line into the socket. It will respond with “HTTP/2 200 OK” which means that the request is successfully processed (status 200). The HTTP/2 is the first major revised HTTP network protocol after HTTP/1.1. Note that “\r\n\r\n” is the format for end of reader.

A screen shot of a computer program

Description automatically generated

Figure 6 Sending the Requested File

Figure 6 shows the server sending the contents of the requested file to the client. The “for loop” is used to send each character from the variable containing the file’s contents. The “\r\n” is then added to mark as the end of reader. After sending the end of reader marker, the connection is closed.

A screenshot of a computer program

Description automatically generated

Figure 7 Except Block

Figure 7 shows the except block corresponding to the try block. The except block works if the requested file does not exist. It responds with “HTTP/2 404 Not Found” which means that the page cannot be found (status 404) with HTTP/2 version. Note that “\r\n\r\n” is the format for end of reader. A 404-error page made using html code is also sent to the client. After sending the error page, the connection is closed.

A screenshot of a computer

Description automatically generated

Figure 8 Termination

Figure 8 shows the break statement after the try-except block to be able to exit the loop after processing one HTTP request, whether it was successful or not. Afterwards, the server socket is closed and the program is terminated. Note that the break statement is placed based on the understanding that the web server should only process one HTTP request. Should the server be capable of processing multiple requests, this line should be removed.

# Screenshots (Top to Bottom | Start to Finish)

# import socket module

from socket import \*

import sys # In order to closed,ate the program

serverSocket = socket(AF\_INET, SOCK\_STREAM)

#Prepare a sever socket

serverPort = 6969

serverSocket.bind(('', serverPort))

serverSocket.listen(1)

while True:

    #Establish the connection

    print('CSNETWK Web Server is ready to serve...')

    connectionSocket, addr = serverSocket.accept()

    try:

        message = connectionSocket.recv(1024).decode()

        filename = message.split()[1]

        f = open(filename[1:])

        outputdata = f.read()

        #Send one HTTP header line into socket

        response = "HTTP/2 200 OK\r\n\r\n"

        connectionSocket.send(response.encode())

        #Send the content of the requested file to the client

        for i in range(0, len(outputdata)):

            connectionSocket.send(outputdata[i].encode())

        connectionSocket.send("\r\n".encode())

        connectionSocket.close()

    except IOError:

        #Send response message for file not found

        response = 'HTTP/2 404 Not Found\r\n\r\n'

        not\_found\_html = """

            <html>

                <head></head>

                <body>

                    <h1>404 Not Found</h1>

                </body>

            </html>

        """

        connectionSocket.send(response.encode())

        connectionSocket.send(not\_found\_html.encode())

        #Close client socket

        connectionSocket.close()

    break

serverSocket.close()

sys.exit()  #Terminate the program after sending the corresponding data

Figure 9 Full Python Code for the Web Server Application

A screenshot of a computer

Description automatically generated

Figure 10 Running of the Web Server Application

A screenshot of a computer

Description automatically generated

Figure 11 Successfully Processed HTTP Request for the CSNETWK.html File

A screenshot of a computer

Description automatically generated

Figure 12 Error Shown for HTTP Request for the Non-Existent CCICOMP.html File