HTTP Server

● Graded

Group

AVANISH R SAMALA
AKSHAJ KAMMARI
View or edit group

Total Points

120 / 120 pts

Autograder Score

120.0 / 120.0

Passed Tests

Checking Server for correctness (120/120)

Autograder Results

Checking Server for correctness (120/120)

Submitted Files

server.py

Download

```python
import socket
import json
import random
import datetime
import hashlib
import sys

#Function to handle a POST request for user login:
def post(requestheaders, accounts, session):
    username = requestheaders.get("username")
    password = requestheaders.get("password")

    if not username or not password:
        log("LOGIN FAILED")
        return "501 Not Implemented", "LOGIN FAILED"

    if username in accounts and checkpassword(password, accounts[username]):
        sessionID = hex(random.getrandbits(64))
        session[sessionID] = {"username": username, "timestamp": datetime.datetime.now()}
        log(f"LOGIN SUCCESSFUL: {username} : {password}")
        return "200 OK", f"Set-Cookie: {'sessionID'}={sessionID}\nLogged in!"

    log(f"LOGIN FAILED: {username} : {password}")
    return "501 Not Implemented", ""

def checkpassword(plaintext, data):
    storedpassword, salt = data
    inputpassword = hashlib.sha256((plaintext + salt).encode()).hexdigest()
    return inputpassword == storedpassword

#Function to handle a GET request for file downloads:
def get(requestheaders, session, root_directory):
    cookies = requestheaders.get("Cookie", "").split("; ")
    sessionID = None

    for cookie in cookies:
        name, value = cookie.split("=")
        if name == 'sessionID':
            sessionID = value

    requesttarget = requestheaders.get("Request-Target", "/")

    if not sessionID or sessionID not in session:
        log(f"COOKIE INVALID: {requesttarget}")
        return "401 Unauthorized", ""

```

```
    current = session[sessionID]
    username = current["username"]
    timestamp = current["timestamp"]
    requesttarget = requestheaders.get("Request-Target", "/")

    if (datetime.datetime.now() - timestamp).seconds > SESSION_TIMEOUT:
        log(f"SESSION EXPIRED: {username} : {requesttarget}")
        return "401 Unauthorized", ""

    session[sessionID]["timestamp"] = datetime.datetime.now()

    requesttarget = requestheaders.get("Request-Target", "/")
    filepath = f"{root_directory}{username}{requesttarget}"

    try:
        with open(filepath, "r") as file:
            filecontents = file.read()
            log(f"GET SUCCEEDED: {username} : {requesttarget}")
            return "200 OK", filecontents
    except FileNotFoundError:
        log(f"GET FAILED: {username} : {requesttarget}")
        return "404 Not Found", ""

def log(message):
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
    print(f"SERVER LOG: {timestamp} {message}")

#Function to start the server
def start(ip, port, accounts_file, session_timeout, root_directory):
    serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    serversocket.bind((ip, port))
    serversocket.listen()

    with open(accounts_file, "r") as file:
        accountdata = json.load(file)

    accounts = accountdata

    session = {}

    while True:
        clientsocket, clientaddress = serversocket.accept()
        requestclient = clientsocket.recv(4096).decode("utf-8")

        if not requestclient:
            continue

        headers, body = requestclient.split("\r\n" + "\r\n", 1)
        requestheaders = ""
```

```
96
97          headerlines = headers.split("\r\n")
98          firstline = headerlines.pop(0)
99          method, requesttarget, _ = firstline.split(" ")
100         headerdict = {"Method": method, "Request-Target": requesttarget}
101
102         for line in headerlines:
103             key, value = line.split(": ")
104             headerdict[key] = value
105
106         requestheaders = headerdict
107
108         method = requestheaders.get("Method")
109         if method == "POST" and requestheaders.get("Request-Target") == "/":
110             status, response = post(requestheaders, accounts, session)
111
112         elif method == "GET":
113             status, response = get(requestheaders, session, root_directory)
114
115         else:
116             status, response = "501 Not Implemented", ""
117
118         responseheaders = f"HTTP/1.0 {status}\n\n"
119         clientsocket.sendall((responseheaders + response).encode("utf-8"))
120         clientsocket.close()
121
122 if __name__ == "__main__":
123     if len(sys.argv) != 6:
124         sys.exit(1)
125
126     IP = sys.argv[1]
127     PORT = int(sys.argv[2])
128     ACCOUNTS_FILE = sys.argv[3]
129     SESSION_TIMEOUT = int(sys.argv[4])
130     ROOT_DIRECTORY = sys.argv[5]
131
132     start(IP, PORT, ACCOUNTS_FILE, SESSION_TIMEOUT, ROOT_DIRECTORY)
```