Individual Assessment Coversheet

To be attached to the front of the assessment.

Campus: Tygervalley

Faculty: Information Technology

Module Code: ITPNA2-34

Group: 1

Lecturer's Name: Tawanda Dundhlu

Student Full Name: Valmy Kalombo

Student Number: EDUV4981741

Indicate	Yes	No
Plagiarism report attached		x

Declaration:

I declare that this assessment is my own original work except for source material explicitly acknowledged. I also declare that this assessment or any other of my original work related to it has not been previously, or is not being simultaneously, submitted for this or any other course. I am aware of the AI policy and acknowledge that I have not used any AI technology to generate or manipulate data, other than as permitted by the assessment instructions. I also declare that I am aware of the Institution's policy and regulations on honesty in academic work as set out in the Conditions of Enrolment, and of the disciplinary guidelines applicable to breaches of such policy and regulations.

Signature	Date
valmykalombo	18/11/2025

Lecturer's Comments:

Marks Awarded:	%

Signature	Date

TABLE OF CONTENTS

EDUV4981741

SECTION A

Introduction

This essay demonstrates the development of several Python network programming applications. The project focuses on real-world scenarios such as online ordering, secure transmission, and automated email systems. The main idea is to show both theoretical understanding and practical implementation using Python.

QUESTION 1: COMPUTER PART ORDERING SYSTEM

I wrote a simple client-server ordering application where two customers can order electronic components from a tech store. The server holds the information of four computer parts: Headsets, Cameras, Tablets, and Batteries, and the customer would place an order on the website, and then the server processes and stores the order.

Server Code:

```
import socket

def main():
    # Store available parts
    parts = {
        "1": "Headsets",
        "2": "Camera",
        "3": "Tablets",
        "4": "Batteries"
    }

    # Create socket
    server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    host = 'localhost'
    port = 5000
    server_socket.bind((host, port))
    server_socket.listen(2)  # up to two customers

    print(f"💻 Tech Store Server running on {host}:{port}")
    print("Waiting for customers to connect...")

    while True:
            conn, addr = server_socket.accept()
            print(f" Connection from: {addr}")

            # Send available parts
            menu = "\nAvailable parts:\n"
            for key, value in parts.items():
```

EDUV4981741

```
                menu += f"{key}. {value}\n"
            conn.send(menu.encode())

            # Receive order
            order = conn.recv(1024).decode()
            if order in parts:
                message = f"Order confirmed for:
{parts[order]}. Your item will be prepared for delivery!"
            else:
                message = "Invalid selection. Please choose
a valid part number."
            conn.send(message.encode())

            conn.close()
            print(f" Connection closed with {addr}\n")

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

Output:

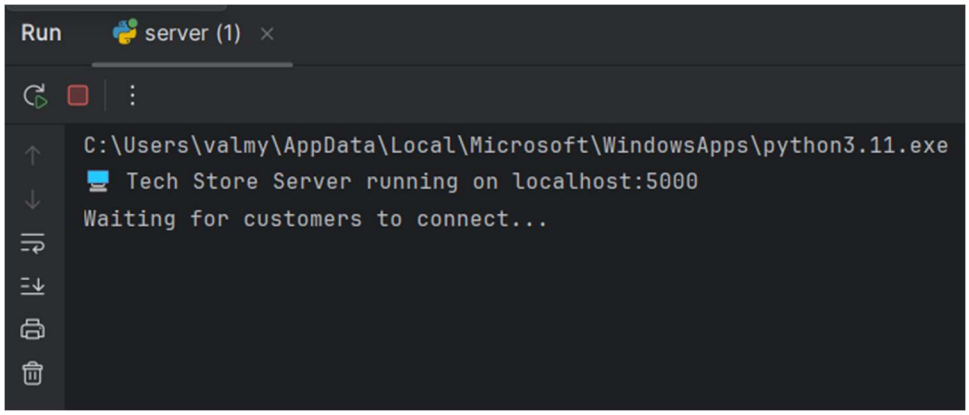

Figure 1: The Tech store server started.

EDUV4981741

Client Code:

```
import socket

def main():
    client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    host = 'localhost'
    port = 5000

    # Connect to server
    client_socket.connect((host, port))
    print("Connected to Tech Store Server!\n")

    # Receive the menu
    menu = client_socket.recv(1024).decode()
    print(menu)

    # Let user choose part
    order = input("Enter the number of the part you want to
order: ")
    client_socket.send(order.encode())

    # Receive confirmation
    message = client_socket.recv(1024).decode()
    print("\n" + message)

    client_socket.close()

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

Output:

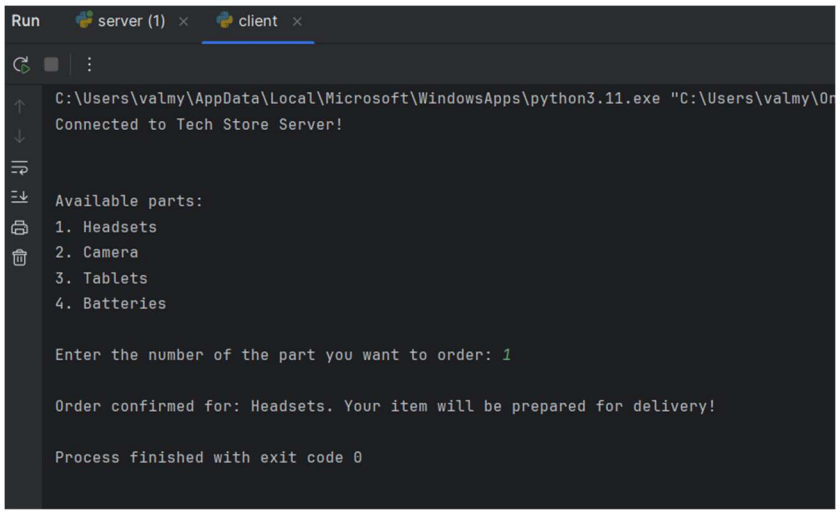

Figure 2:Client placing order.

QUESTION 2: ONLINE ORDERING WITH CRUD OVER HTTP

This code shows how CRUD(Create, Read, Update, Delete) are used over HTTP. The data is stored inside a JSON file and users can place orders, update and existing order, and delete an order.

Sever Code:

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import urllib.parse
import json
import os
import smtplib
from email.message import EmailMessage
from datetime import datetime

# === EMAIL RECEIPT FUNCTION ===
def send_order_receipt(customer_name, customer_email,
item_name):
    EMAIL_ADDRESS = os.getenv("EMAIL_USER")
    EMAIL_PASSWORD = os.getenv("EMAIL_PASS")

    msg = EmailMessage()
    msg["Subject"] = "📄 Order Confirmation - Tech Store"
    msg["From"] = EMAIL_ADDRESS
    msg["To"] = customer_email

    html_content = f"""
    <html>
    <body style="font-family: Arial; background-
color:#f7f7f7; padding:20px;">
      <div style="background:#fff; border-radius:10px;
padding:20px; max-width:600px; margin:auto;">
        <h2 style="color:#333;">☑️ Order Confirmed!</h2>
        <p>Hello <b>{customer_name}</b>,</p>
        <p>Thank you for ordering from <b>Tech
Store</b>.</p>
        <p>We have received your order for:
<b>{item_name}</b>.</p>
        <hr>
        <p>🧾 Order Date: {datetime.now().strftime("%Y-%m-
%d %H:%M:%S")}</p>
        <p>We'll prepare your order shortly.</p>
      </div>
    </body>
    </html>
    """

    msg.set_content("Your order has been confirmed.")
    msg.add_alternative(html_content, subtype='html')

    try:
        with smtplib.SMTP("smtp.gmail.com", 587) as smtp:
            smtp.starttls()
            smtp.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
```

```
            smtp.send_message(msg)
            print(f"📧 Email sent successfully to
{customer_email}")
    except Exception as e:
        print(f"❌ Error sending email: {e}")

# File to store orders
ORDERS_FILE = "orders.json"

# ☑ FIX 1: Properly load or initialize orders and move
save_orders() outside condition
orders = []
if os.path.exists(ORDERS_FILE):
    with open(ORDERS_FILE, "r") as f:
        orders = json.load(f)

def save_orders():
    """Save the current list of orders to a JSON file."""
    with open(ORDERS_FILE, "w") as f:
        json.dump(orders, f, indent=4)


class OrderHandler(BaseHTTPRequestHandler):

    def show_homepage(self):
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()

        html = """
        <!DOCTYPE html>
        <html lang="en">
        <head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-
width, initial-scale=1.0">
            <title>💻 Tech Store - Order Portal</title>
            <style>
    body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
        background: radial-gradient(circle at top left,
#0f2027, #203a43, #2c5364);
        color: #fff;
        text-align: center;
        padding: 40px;
        margin: 0;
    }
    h1 {
        color: #ffcc00;
        font-size: 2.5em;
        margin-bottom: 10px;
        text-shadow: 0 0 15px #ffcc00, 0 0 30px #ffcc00;
        animation: glow 2s ease-in-out infinite alternate;
    }
    @keyframes glow {
```

```
        from { text-shadow: 0 0 10px #ffcc00, 0 0 20px
#ffcc00; }
        to { text-shadow: 0 0 25px #ffd633, 0 0 50px
#ffd633; }
    }
    .container {
        background: rgba(255, 255, 255, 0.05);
        border-radius: 10px;
        padding: 30px;
        max-width: 500px;
        margin: 0 auto;
        box-shadow: 0 0 20px rgba(255, 255, 255, 0.1);
    }
    form {
        margin-top: 20px;
    }
    input[type="text"] {
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ffcc00;
    width: 80%;
    margin: 8px 0;
    background-color: rgba(255,255,255,0.1);
    color: #fff;
    box-shadow: 0 0 10px #ffcc00;
}

select {
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ffcc00;
    width: 84%;
    margin: 8px 0;
    background-color: rgba(255,255,255,0.1);
    color: #fff;
    box-shadow: none;
    text-shadow: none;
    outline: none;
    appearance: none;
}

select:focus {
    border-color: #ffd633;
    background-color: rgba(255,255,255,0.15);
    box-shadow: none;
}

option {
    background-color: #2c5364;
    color: white;
    text-shadow: none;
}

    }
    button {
        background-color: #ffcc00;
        color: #333;
```

```
        padding: 10px 20px;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        font-weight: bold;
        transition: 0.3s;
        box-shadow: 0 0 15px #ffcc00;
    }
    button:hover {
        background-color: #ffd633;
        transform: scale(1.05);
        box-shadow: 0 0 25px #ffd633, 0 0 40px #ffd633;
    }
    a {
        display: inline-block;
        margin-top: 20px;
        color: #ffcc00;
        text-decoration: none;
        font-weight: bold;
        text-shadow: 0 0 10px #ffcc00;
    }
    a:hover {
        color: #fff;
        text-shadow: 0 0 20px #fff;
    }
</style>
        </head>
        <body>
            <div class="container">
                <h1>🛒 Tech Store Online Ordering</h1>
                <p>Welcome! Please fill in your details to
order a computer part below.</p>

                <form action="/create" method="POST">
    <input type="text" name="name" placeholder="Your Name"
required><br>
    <input type="email" name="email" placeholder="Your
Email" required><br>
                    <select name="item" required>
                        <option value="">Select a Computer
Part</option>
                        <option>Headsets</option>
                        <option>Camera</option>
                        <option>Tablets</option>
                        <option>Batteries</option>
                    </select><br>
                    <button type="submit">Place
Order</button>
                </form>
                <a href="/orders">📦 View All Orders</a>
            </div>
        </body>
        </html>
        """
        self.wfile.write(html.encode())

    def show_orders(self):
```

```
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()

        html = """
        <!DOCTYPE html>
        <html lang="en">
        <head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-
width, initial-scale=1.0">
            <title>All Orders</title>
            <style>
    body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
        background: linear-gradient(135deg, #141E30,
#243B55);
        color: #fff;
        text-align: center;
        padding: 40px;
    }
    h1 {
        color: #ffcc00;
        font-size: 2.2em;
        text-shadow: 0 0 15px #ffcc00, 0 0 25px #ffd633;
        animation: pulse 1.8s infinite alternate;
    }
    @keyframes pulse {
        from { text-shadow: 0 0 15px #ffcc00; }
        to { text-shadow: 0 0 35px #ffd633; }
    }
    table {
        width: 80%;
        margin: 30px auto;
        border-collapse: collapse;
        background: rgba(255, 255, 255, 0.05);
        border-radius: 10px;
        box-shadow: 0 0 25px rgba(255,255,255,0.2);
    }
    th, td {
        border-bottom: 1px solid #ffcc00;
        padding: 12px;
        text-align: center;
        font-size: 1.1em;
    }
    tr:hover {
        background-color: rgba(255,255,255,0.1);
        box-shadow: 0 0 15px #ffd633;
    }
    a {
        color: #ffcc00;
        text-decoration: none;
        font-weight: bold;
        text-shadow: 0 0 10px #ffcc00;
    }
    a:hover {
```

```
        color: #fff;
        text-shadow: 0 0 20px #fff;
    }
</style>
        </head>
        <body>
            <h1>📋 Order Records</h1>
            <table>
                <tr><th>Customer Name</th><th>Item
Ordered</th></tr>
        """

        for i, order in enumerate(orders):
            if isinstance(order, dict) and 'name' in order
and 'item' in order:
                html +=
f"<tr><td>{order['name']}</td><td>{order['item']}</td>"
                html += f"<td><a href='/update?id={i}'>✏️
Edit</a> | <a href='/delete?id={i}'>❌
Delete</a></td></tr>"

        html += """
            </table>
            <a href="/">🔙 Back to Order Page</a>
        </body>
        </html>
        """
        self.wfile.write(html.encode())

    def show_update_form(self, order_id):
        order = orders[int(order_id)]
        self.send_response(200)
        self.send_header("Content-type", "text/html;
charset=utf-8")  # ☑️ Fix encoding
        self.end_headers()

        html = f"""
        <!DOCTYPE html>
        <html lang="en">
        <head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-
width, initial-scale=1.0">
            <title>✏️ Update Order</title>
            <style>
                body {{
                    font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif;
                    background: radial-gradient(circle at
top right, #0f2027, #203a43, #2c5364);
                    color: #fff;
                    text-align: center;
                    padding: 50px;
                    margin: 0;
                }}
                .container {{
```

```
                    background: rgba(255, 255, 255, 0.08);
                    border-radius: 12px;
                    padding: 40px;
                    max-width: 500px;
                    margin: 0 auto;
                    box-shadow: 0 0 25px
rgba(255,255,255,0.15);
                }}
                h1 {{
                    color: #ffcc00;
                    font-size: 2em;
                    margin-bottom: 15px;
                    text-shadow: 0 0 15px #ffcc00, 0 0 30px
#ffd633;
                    animation: glow 1.8s infinite
alternate;
                }}
                @keyframes glow {{
                    from {{ text-shadow: 0 0 10px #ffcc00;
}}
                    to {{ text-shadow: 0 0 25px #ffd633, 0
0 50px #ffd633; }}
                }}
                input[type="text"] {{
                    padding: 12px;
                    border-radius: 5px;
                    border: 1px solid #ffcc00;
                    width: 80%;
                    margin: 10px 0;
                    background-color:
rgba(255,255,255,0.1);
                    color: #fff;
                    font-size: 1em;
                    box-shadow: 0 0 10px #ffcc00; /* glow
only for input box */
                }}
                select {{
                    padding: 12px;
                    border-radius: 5px;
                    border: 1px solid #ffcc00;
                    width: 84%;
                    margin: 10px 0;
                    background-color:
rgba(255,255,255,0.1);
                    color: #fff;
                    box-shadow: none; /* ☑ no glow */
                    text-shadow: none; /* ☑ remove glow on
text */
                    outline: none;
                    appearance: none; /* modern dropdown
arrow */
                }}
                select:focus {{
                    border-color: #ffd633;
                    background-color:
rgba(255,255,255,0.15);
                    box-shadow: none; /* ☑ no glow on
```

```
focus */
                }}
                option {{
                    background-color: #2c5364;
                    color: white;
                    text-shadow: none; /* ☑ no text glow
*/
                }}
                button {{
                    background-color: #ffcc00;
                    color: #333;
                    padding: 10px 25px;
                    border: none;
                    border-radius: 8px;
                    cursor: pointer;
                    font-weight: bold;
                    transition: 0.3s;
                    box-shadow: 0 0 15px #ffcc00;
                }}
                button:hover {{
                    background-color: #ffd633;
                    transform: scale(1.05);
                    box-shadow: 0 0 25px #ffd633, 0 0 40px
#ffd633;
                }}
                a {{
                    display: inline-block;
                    margin-top: 20px;
                    color: #ffcc00;
                    text-decoration: none;
                    font-weight: bold;
                    text-shadow: 0 0 10px #ffcc00;
                }}
                a:hover {{
                    color: #fff;
                    text-shadow: 0 0 20px #fff;
                }}
            </style>
        </head>
        <body>
            <div class="container">
                <h1>✏ Update Order</h1>
                <form action="/update?id={order_id}"
method="POST">
                    <input type="text" name="name"
value="{order['name']}" required><br>
                    <select name="item" required>
                        <option {'selected' if
order['item'] == 'Headsets' else ''}>Headsets</option>
                        <option {'selected' if
order['item'] == 'Camera' else ''}>Camera</option>
                        <option {'selected' if
order['item'] == 'Tablets' else ''}>Tablets</option>
                        <option {'selected' if
order['item'] == 'Batteries' else ''}>Batteries</option>
                    </select><br><br>
                    <button type="submit">Update
```

```
Order</button>
                </form>
                <a href="/orders">⬅ Back to Orders</a>
            </div>
        </body>
        </html>
        """
        self.wfile.write(html.encode("utf-8"))

    def delete_order(self, order_id):
        if 0 <= int(order_id) < len(orders):
            del orders[int(order_id)]
            save_orders()

        self.send_response(302)
        self.send_header("Location", "/orders")
        self.end_headers()

    def do_GET(self):
        parsed = urllib.parse.urlparse(self.path)
        path = parsed.path
        query = urllib.parse.parse_qs(parsed.query)

        if path == "/favicon.ico":
            self.send_response(200)
            self.end_headers()
        elif path == "/":
            self.show_homepage()
        elif path == "/orders":
            self.show_orders()
        elif path == "/update" and "id" in query:
            self.show_update_form(query["id"][0])
        elif path == "/delete" and "id" in query:
            self.delete_order(query["id"][0])
        else:
            self.send_error(404, "Page Not Found")

    def do_POST(self):
        # ☑ FIX 2 + 3: Correct indentation and add
save_orders()
        if self.path == "/create":
            content_length = int(self.headers['Content-
Length'])
            post_data = self.rfile.read(content_length)
            data =
urllib.parse.parse_qs(post_data.decode())

            name = data.get('name', [''])[0]
            item = data.get('item', [''])[0]
            email = data.get('email', [''])[0]

            orders.append({'name': name, 'email': email,
'item': item})

            # Send the email receipt
            send_order_receipt(name, email, item)
```

```
            self.send_response(200)
            self.send_header("Content-type", "text/html;
charset=utf-8")
            self.end_headers()

            confirmation_html = f"""
            <!DOCTYPE html>
            <html lang="en">
            <head>
                <meta charset="UTF-8">
                <meta name="viewport"
content="width=device-width, initial-scale=1.0">
                <title>Order Confirmed</title>
                <style>
                    body {{
                        font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif;
                        background: radial-gradient(circle
at top right, #0f2027, #203a43, #2c5364);
                        color: #fff;
                        text-align: center;
                        margin: 0;
                        padding: 50px;
                    }}
                    .container {{
                        background: rgba(255, 255, 255,
0.08);
                        border-radius: 12px;
                        padding: 40px;
                        max-width: 600px;
                        margin: 0 auto;
                        box-shadow: 0 0 20px
rgba(0,0,0,0.2);
                        backdrop-filter: blur(8px);
                    }}
                    h1 {{
                        color: #ffcc00;
                        font-size: 2em;
                        margin-bottom: 15px;
                        text-shadow: 0 0 15px #ffcc00, 0 0
30px #ffd633;
                        animation: glow 1.8s infinite
alternate;
                    }}
                    @keyframes glow {{
                        from {{ text-shadow: 0 0 10px
#ffcc00; }}
                        to {{ text-shadow: 0 0 25px
#ffd633, 0 0 50px #ffd633; }}
                    }}
                    p {{
                        font-size: 1.1em;
                        color: #e0e0e0;
                    }}
                    .button {{
                        display: inline-block;
                        background-color: #ffcc00;
```

```
                        color: #333;
                        padding: 12px 25px;
                        border-radius: 8px;
                        font-weight: bold;
                        text-decoration: none;
                        margin: 15px;
                        transition: all 0.3s ease;
                        box-shadow: 0 0 15px #ffcc00;
                    }}
                    .button:hover {{
                        background-color: #ffd633;
                        transform: scale(1.05);
                        box-shadow: 0 0 25px #ffd633, 0 0
40px #ffd633;
                    }}
                </style>
            </head>
            <body>
                <div class="container">
                    <h1>☑ Order Confirmed!</h1>
                    <p>Thank you, <b>{name}</b>. You have
successfully ordered <b>{item}</b>.</p>
                    <a href="/" class="button">🔙 Back to
Homepage</a>
                    <a href="/orders" class="button">📦
View All Orders</a>
                </div>
            </body>
            </html>
            """
            self.wfile.write(confirmation_html.encode("utf-
8"))

        elif self.path.startswith("/update"):
            parsed = urllib.parse.urlparse(self.path)
            query = urllib.parse.parse_qs(parsed.query)
            order_id = int(query["id"][0])
            content_length = int(self.headers['Content-
Length'])
            post_data = self.rfile.read(content_length)
            data =
urllib.parse.parse_qs(post_data.decode())

            orders[order_id]['name'] = data.get('name',
[''])[0]
            orders[order_id]['item'] = data.get('item',
[''])[0]
            save_orders()

            self.send_response(302)
            self.send_header("Location", "/orders")
            self.end_headers()


def run():
    host = "localhost"
    port = 8080
```

```
    server = HTTPServer((host, port), OrderHandler)
    print(f" Server started at http://{host}:{port}")
    server.serve_forever()


if __name__ == "__main__":
    run()
```

Output:

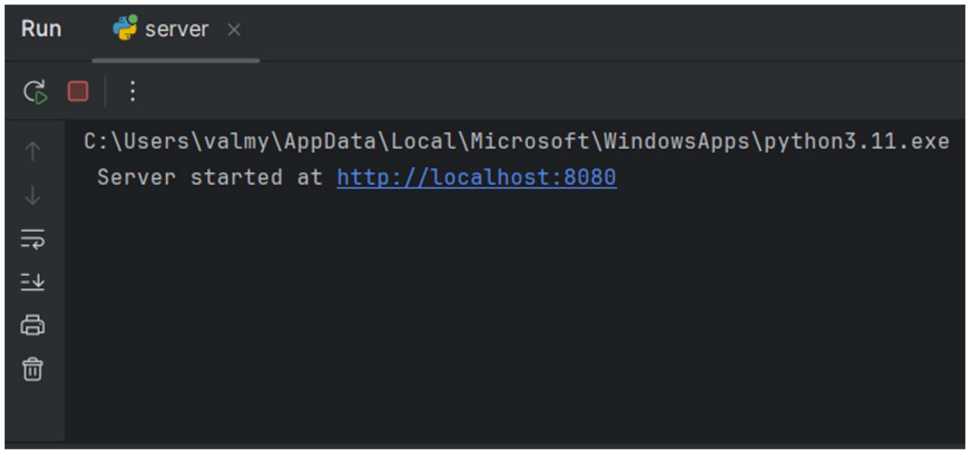

Figure 3:Server running localhost:8080.

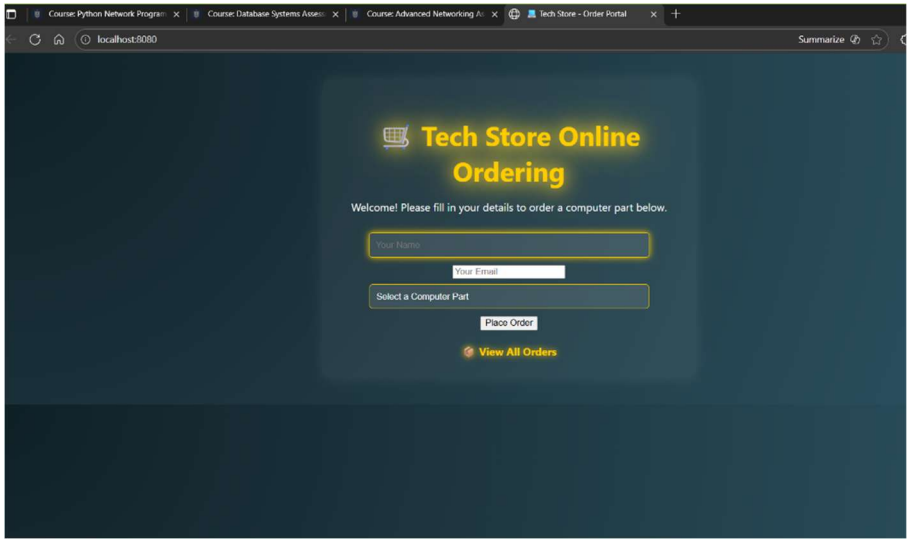

Figure 4:Running website for Tech Store.

server.py orders.json

```
[
    {
        "customer": "Valmy",
        "item": "Laptop"
    },
    {
        "name": "Noah",
        "item": "Batteries"
    },
    {
        "name": "John",
        "email": "example@gmail.com",
        "item": "Camera"
    }
]
```

Figure 5:JSON file reports for orders.

QUESTION 3: SECURITY IMPLEMENTATION (TLS + AES)

3.1)

This code displays the use of TLS and AES where it secures the connection channel while AES encrypts data inside the channel, providing double-layer protection. I also had to create a certs folder with the following inside: ca.crt; ca.key; ca.srl; client.crt; client.csr; client.key; server.crt; server.csr; and the server.key certificates to ensure a secure connection to the server from the client using TLS and AES encryption.

Server Code:

```
import socket, ssl, json
from cryptography.hazmat.primitives.ciphers.aead import
AESGCM
import os

HOST = '127.0.0.1'
PORT = 8443

# Application symmetric key (32 bytes for AES-256).
# In real systems you'd derive per-session keys (e.g., from
a KDF or handshake).
# For this educational demo we generate a static key and
print it so client can use it.
APP_KEY =
b'\xb5\x08\x80\x8e3\x88\xa1\x11>\xffQ\n\xf3\x80\x9d32*\xc4h
\x03\t$tr\x05\xa5\xf3\xf3h\x809'
print("Application AES key (base16) — give this to client
for demo:", APP_KEY.hex())

def build_ssl_context():
    ctx =
ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
    ctx.load_cert_chain(certfile='certs/server.crt',
keyfile='certs/server.key')
    ctx.load_verify_locations(cafile='certs/ca.crt')
    ctx.verify_mode = ssl.CERT_REQUIRED  # require client
certificate
    return ctx

def decrypt_and_process(encrypted_blob: bytes):
    """
    Encrypted blob format (JSON bytes encoded):
    {
      "nonce": "<hex>",
      "ciphertext": "<hex>",
      "aad": "<optional hex or empty>"
    }
    """
    payload = json.loads(encrypted_blob.decode('utf-8'))
    nonce = bytes.fromhex(payload['nonce'])
```

```
    ct = bytes.fromhex(payload['ciphertext'])
    aad = bytes.fromhex(payload.get('aad', '')) if
payload.get('aad') else None

    aesgcm = AESGCM(APP_KEY)
    if aad:
        plaintext = aesgcm.decrypt(nonce, ct, aad)
    else:
        plaintext = aesgcm.decrypt(nonce, ct, None)
    return plaintext.decode('utf-8')

def main():
    context = build_ssl_context()
    bindsock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM, 0)
    bindsock.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
    bindsock.bind((HOST, PORT))
    bindsock.listen(5)
    print(f"[SERVER] Listening on {HOST}:{PORT} (TLS, mTLS
required)")

    while True:
        newsock, addr = bindsock.accept()
        try:
            with context.wrap_socket(newsock,
server_side=True) as ssock:
                print(f"[SERVER] TLS connection from
{addr}")
                # read size first (4 bytes network order)
                size_bytes = ssock.recv(4)
                if len(size_bytes) < 4:
                    print("[SERVER] invalid size header")
                    continue
                size = int.from_bytes(size_bytes, 'big')
                data = b''
                while len(data) < size:
                    chunk = ssock.recv(min(4096, size -
len(data)))
                    if not chunk:
                        break
                    data += chunk
                if not data:
                    print("[SERVER] no payload")
                    continue

                # decrypt application payload
                try:
                    text = decrypt_and_process(data)
                    print("[SERVER] Received (decrypted):",
text)
                    # example response (echo)
                    response_plain = f"Server received:
{text}"
                    # encrypt response
                    aesgcm = AESGCM(APP_KEY)
                    resp_nonce = os.urandom(12)
```

```
                    resp_ct = aesgcm.encrypt(resp_nonce,
response_plain.encode('utf-8'), None)
                    resp_payload = json.dumps({
                        "nonce": resp_nonce.hex(),
                        "ciphertext": resp_ct.hex()
                    }).encode('utf-8')
                    # send size + payload

ssock.sendall(len(resp_payload).to_bytes(4, 'big') +
resp_payload)
                except Exception as e:
                    print("[SERVER] Decrypt/Process
error:", e)
                    ssock.sendall(b'ERR')
        except ssl.SSLError as e:
            print("[SERVER] SSL error:", e)
        except Exception as e:
            print("[SERVER] Connection error:", e)
        finally:
            try:
                newsock.close()
            except:
                pass

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

Output:

Figure 6:TLS server started.

Client Code:

```
import socket, ssl, json, os
from cryptography.hazmat.primitives.ciphers.aead import
AESGCM

HOST = '127.0.0.1'
PORT = 8443

# Paste the application key printed by the server (hex)
APP_KEY_HEX =
"b508808e3388a1113eff510af3809d33322ac468030924747205a5f3f3
688039"
APP_KEY = bytes.fromhex(APP_KEY_HEX)
```

```
def build_ssl_context():
    ctx =
ssl.create_default_context(ssl.Purpose.SERVER_AUTH,
cafile='certs/ca.crt')
    ctx.load_cert_chain(certfile='certs/client.crt',
keyfile='certs/client.key')
    # optional: require host verification if CN matches
'localhost'
    ctx.check_hostname = False
    return ctx

def encrypt_message(plaintext: str):
    aesgcm = AESGCM(APP_KEY)
    nonce = os.urandom(12)
    ct = aesgcm.encrypt(nonce, plaintext.encode('utf-8'),
None)
    return json.dumps({
        "nonce": nonce.hex(),
        "ciphertext": ct.hex()
    }).encode('utf-8')

def decrypt_response(data: bytes):
    payload = json.loads(data.decode('utf-8'))
    nonce = bytes.fromhex(payload['nonce'])
    ct = bytes.fromhex(payload['ciphertext'])
    aesgcm = AESGCM(APP_KEY)
    pt = aesgcm.decrypt(nonce, ct, None)
    return pt.decode('utf-8')

def main():
    context = build_ssl_context()
    raw = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    with context.wrap_socket(raw,
server_hostname='localhost') as ssock:
        ssock.connect((HOST, PORT))
        print("[CLIENT] TLS connected to server (mTLS)")

        # Example application message
        msg = "Order: Camera x1 for user Alice"
        payload = encrypt_message(msg)
        # send length header + payload
        ssock.sendall(len(payload).to_bytes(4, 'big') +
payload)
        # receive response length header
        size_bytes = ssock.recv(4)
        size = int.from_bytes(size_bytes, 'big')
        data = b''
        while len(data) < size:
            chunk = ssock.recv(min(4096, size - len(data)))
            if not chunk:
                break
            data += chunk
        if data:
            resp = decrypt_response(data)
            print("[CLIENT] Decrypted response:", resp)
        else:
```

```
            print("[CLIENT] No response or error")

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

Output:

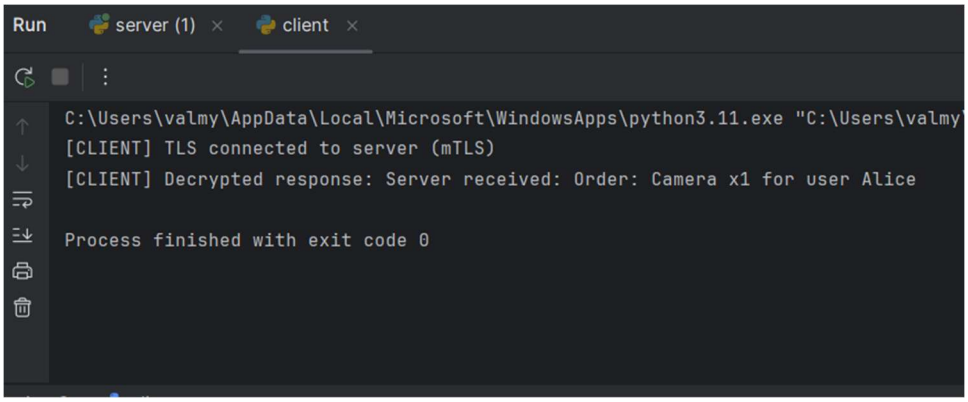

Figure 7:Client connected securely.

QUESTION 4: EMAIL AUTOMATION SYSTEM

4.1)

With this code, whenever a customer places an order, the system sends a confirmation email using Python's smtplib. I used my Gmail account and a generated password to authenticate safely.

Email Receipt Code:

```
import os
import smtplib
from email.message import EmailMessage
from datetime import datetime

# 🧠 Step 1: Load email credentials from environment
variables
EMAIL_ADDRESS = os.getenv("EMAIL_USER")
EMAIL_PASSWORD = os.getenv("EMAIL_PASS")

# 📄 Step 2: Define a function to send the order
confirmation email
def send_order_receipt(customer_name, customer_email,
item_name):
    # Create the email message
    msg = EmailMessage()
    msg["Subject"] = "📄 Order Confirmation - Tech Store"
    msg["From"] = EMAIL_ADDRESS
    msg["To"] = customer_email

    # Email body (HTML styled for a nice look)
    html_content = f"""
    <html>
    <body style="font-family: Arial, sans-serif;
background-color: #f2f2f2; padding: 20px;">
        <div style="background-color: white; border-radius:
10px; padding: 20px; max-width: 600px; margin: auto;">
            <h2 style="color: #333;">☑️ Order
Confirmed!</h2>
            <p>Hello <strong>{customer_name}</strong>,</p>
            <p>Thank you for shopping with <b>Tech
Store</b>.</p>
            <p>We have successfully received your order
for:</p>
            <p style="font-size: 1.2em; color:
#008080;"><strong>{item_name}</strong></p>
            <hr>
            <p>🧮 Order Date: {datetime.now().strftime("%Y-
%m-%d %H:%M:%S")}</p>
            <p>We will process your order shortly and send
you another email once it’s ready for delivery.</p>
```

```
            <p>Kind regards,<br><strong>The Tech Store
Team</strong></p>
        </div>
    </body>
    </html>
    """

    # Set email content
    msg.set_content("Your order has been confirmed.")
    msg.add_alternative(html_content, subtype='html')

    # Step 3: Connect to Gmail's secure SMTP server
    try:
        with smtplib.SMTP("smtp.gmail.com", 587) as smtp:
            smtp.starttls()  # Encrypt connection
            smtp.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
            smtp.send_message(msg)
            print(f"📧 Receipt sent successfully to
{customer_email}")
    except Exception as e:
        print(f"❌ Error sending email: {e}")

# 🧩 Step 4: Example usage
if __name__ == "__main__":
    print("=== Tech Store Email System ===")
    name = input("Enter customer name: ")
    email = input("Enter customer email: ")
    item = input("Enter ordered item: ")

    send_order_receipt(name, email, item)
```

Output:

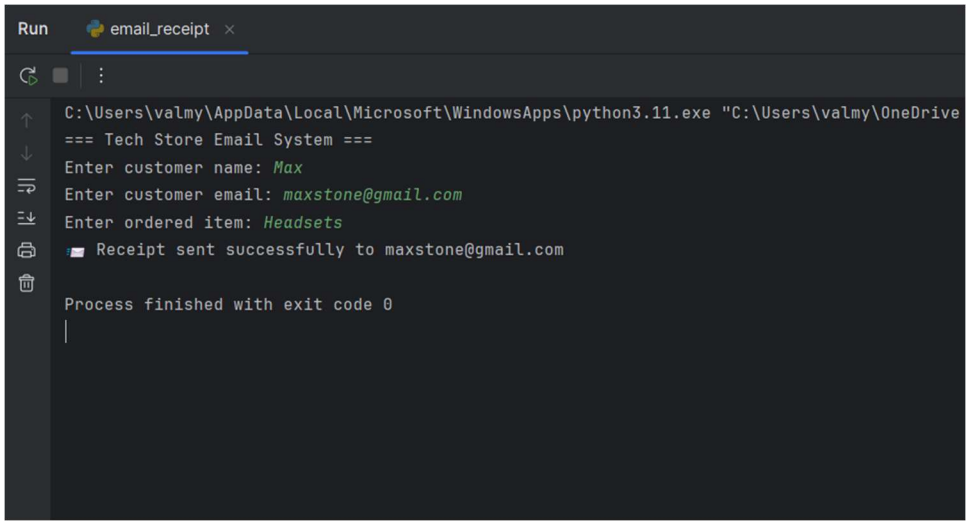

Figure 8: Requesting email receipt.

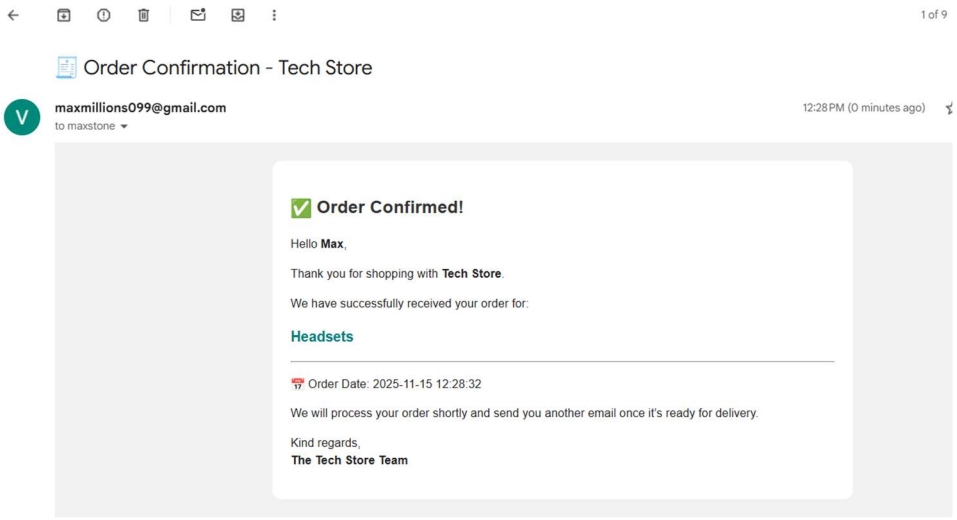

Figure 9: Confirmation of email receipt

4.2)

Here, the system reads all the orders from the JSON file and sends a summary at the end of the month to the recipient's email.

Monthly Summary Code:

```
import os
import smtplib
import json
from datetime import datetime
from email.message import EmailMessage

# === 1. Load credentials securely from environment
variables ===
EMAIL_ADDRESS = os.getenv("EMAIL_USER")
EMAIL_PASSWORD = os.getenv("EMAIL_PASS")

# === 2. Load all orders from orders.json ===
ORDERS_FILE = r"C:\Users\valmy\OneDrive - Eduvos\Year 2
Courses\Python Network Programming\Project 2\Question
2\orders.json"


def load_orders():
    if not os.path.exists(ORDERS_FILE):
        print("❌ No orders file found.")
        return []
    with open(ORDERS_FILE, "r") as f:
        try:
            return json.load(f)
        except json.JSONDecodeError:
            return []


# === 3. Format a summary report ===
def generate_summary(orders):
    if not orders:
        return "<p>No orders were placed this month.</p>"

    html = """
    <h3>🧮 Monthly Purchase Summary</h3>
    <table border='1' cellspacing='0' cellpadding='6'
style='border-collapse:collapse;'>
      <tr style='background-color:#f2f2f2;'>
        <th>Customer Name</th>
        <th>Item Ordered</th>
      </tr>
    """
    for order in orders:
        name = order.get("name", "Unknown")
        item = order.get("item", "N/A")
        html += f"<tr><td>{name}</td><td>{item}</td></tr>"
    html += "</table>"
    return html
```

EDUV4981741

```
# === 4. Send the summary email ===
def send_monthly_summary():
    orders = load_orders()
    html_summary = generate_summary(orders)
    current_month = datetime.now().strftime("%B %Y")

    msg = EmailMessage()
    msg["Subject"] = f"📄 Tech Store Monthly Summary -
{current_month}"
    msg["From"] = EMAIL_ADDRESS
    msg["To"] = EMAIL_ADDRESS  # send to yourself or
manager

    body = f"""
    <html>
    <body style="font-family:Arial, sans-serif;">
      <h2>Tech Store Monthly Sales Report</h2>
      <p>Dear Manager,</p>
      <p>Here is the summary of all orders for
<b>{current_month}</b>.</p>
      {html_summary}
      <p style="margin-top:20px;">Kind regards,<br><b>Tech
Store System</b></p>
    </body>
    </html>
    """
    msg.set_content("Monthly sales summary attached.")
    msg.add_alternative(body, subtype='html')

    try:
        with smtplib.SMTP("smtp.gmail.com", 587) as smtp:
            smtp.starttls()
            smtp.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
            smtp.send_message(msg)
            print(f"📧 Monthly summary sent successfully to
{EMAIL_ADDRESS}")
    except Exception as e:
        print(f"❌ Failed to send summary: {e}")


# === 5. Run the function manually or via scheduler ===
if __name__ == "__main__":
    send_monthly_summary()
```

Output:

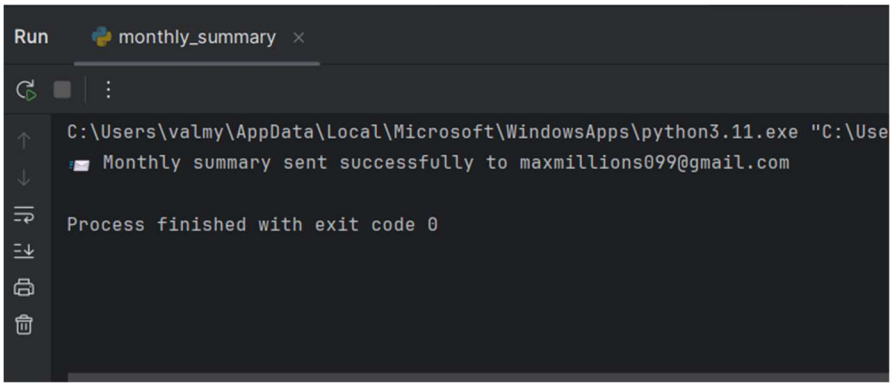

Figure 10: Sending monthly summary to email.

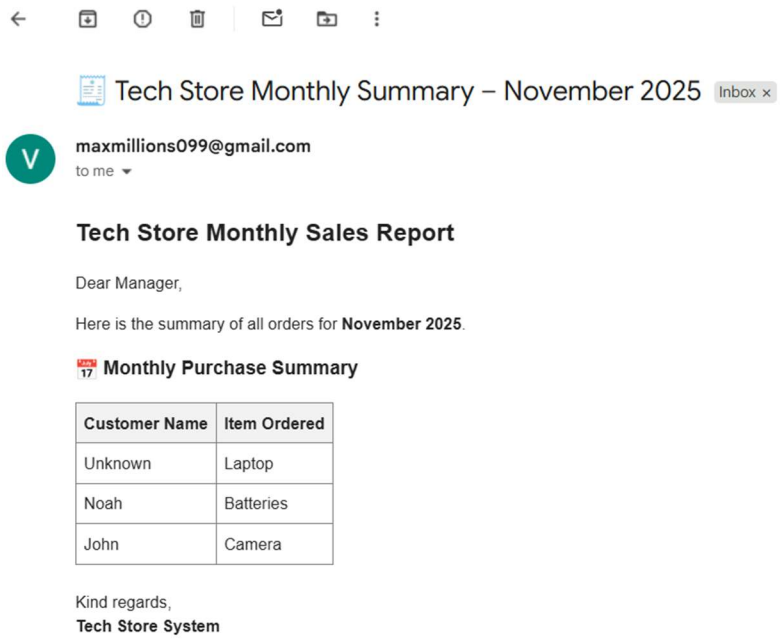

Figure 11: Confirmation of monthly receipt sent,

EDUV4981741

Conclusion

This project has helped me understand Python network programming by exposing me to real-world scenarios such as handling HTTP communication, processing user requests, building CRUD web applications, implementing encryption, and automating email communication. Adding to the theory and practical development improved my knowledge of client–server systems and Python's libraries.

BIBLIOGRAPHY

Ali. (2020). How to make a TLS connection using python? *stack overflow*. Retrieved from https://stackoverflow.com/questions/63226614/how-to-make-a-tls-connection-using-python

Crocker, D. H. (1982, August 13). *Standard for the Format of ARPA Internet Text Messages.* Retrieved from Dept. of Electrical Engineering University of Delaware, Newark, DE: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.rfc-editor.org/rfc/pdfrfc/rfc822.txt.pdf

Dierks, T., & Rescorla, E. (2008, August). *The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246.* Retrieved from Network Working Group: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.rfc-editor.org/rfc/pdfrfc/rfc5246.txt.pdf

Klensin, J. (2008, October). *Simple Mail Transfer Protocol. RFC 5321.* Retrieved from Network Working Group: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.rfc-editor.org/rfc/pdfrfc/rfc5321.txt.pdf

Postel, J. (1980, August 28). *User Datagram Protocol. RFC 768.* Retrieved from IEEE: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.rfc-editor.org/rfc/pdfrfc/rfc768.txt.pdf

Python Software Foundation. (2025). Python 3.11 Documentation. *Python Software Foundation*. Retrieved November 1, 2025, from https://docs.python.org/3/

Python Software Foundation. (2025). smtplib — SMTP protocol client. *Python Software Foundation*. Retrieved from https://docs.python.org/3/library/smtplib.html

Rescorla, E. (2018, August). *The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446.* Retrieved from Internet Engineering Task Force (IETF): chrome-

extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.rfc-editor.org/rfc/pdfrfc/rfc8446.txt.pdf

Rhodes, B., & Goerzen, J. (2004). *Foundations of Python Network Programming.* Apress.

W, W. (2016). Breaking Down the REST Dissertation, Part 1: Overview. *Medium*. Retrieved from https://medium.com/vulk-coop/breaking-down-the-rest-dissertation-part-1-overview-b8ea62dae347