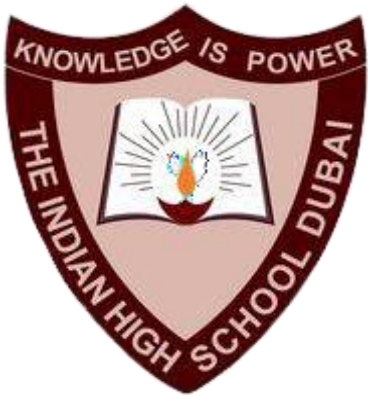
THE INDIAN HIGH SCHOOL - DUBAI



Computer Science Project
2025-26

Name: Abyaz Javid
Class: 12 Sci H
Roll no.:

[insert certificate here]

ACKNOWLEDGEMENT

I would like to take this opportunity to thank the Central Board of Secondary Education (CBSE) and The Indian High School-Dubai, for granting me the opportunity to deepen my knowledge in my favorite subject, Computer Science

I would also like to thank my teacher Mrs. Swapnil Verma for guiding me and sharing her wide variety of knowledge

INDEX

Introduction

Problem Definition

Cross platform copy-paste / clipboard application, using MySQL for persistent clipboard history and communication via a websocket hosted on a flask server.

Reason for Choosing the Topic

The project was chosen to simplify the objective of copy-paste from one device onto another, harnessing the cross-platform capabilities of the python programming language

Objectives

- Implement a server that facilitates basic text transportation between devices on the same network
- Develop a client to automatically discover the server via a UDP broadcast
- Ensure multi-client stability

Requirements

Hardware Requirements

- Computer running:
  - Windows 10+ / MacOS / Linux (most distros with a clipboard)
  - 2GB RAM
  - Network adapter with UDP & TCP/IP support
- LAN Networking
- At least two devices

Software Requirements

- Python 3.13+
- Required python libraries:
  - `websockets` (client websocket connection)
  - `flask` (server-side HTTP handling)
  - `flask-sock` (websocket endpoint in flask server)
  - `pyperclip` (for clipboard support)

Functions Used

Built-in Functions

- `print()` → display output to console
- `input()` → take user input
- `set()` → create empty set
- `exit()` and `sys.exit()` → terminates program
- `isinstance()` → type checking

User-Defined Functions

`client.py`:

- `listen(ws)` → asynchronously listens for incoming websocket messages and updates clipboard accordingly
- `send_input(ws)` → asynchronously takes user input from console and sends to websocket
- `main(uri)` → manages websocket connection and runs both send and receive loops

`server.py`:

- `getLocalIP()` → finds local LAN IP of the server
- `udpListener()` → listens for UDP broadcast discovery requests and replies with server "`IP:PORT/`"
- `websocket()` → handles websocket connections for all clients

`utils.py`:

- `log(msg, code)` → writes log to a file with a code ("OK", "INFO", "WARN", "FAIL")
- `closeLogFile ()` → closes the log file

Other Modules

`socket`:

- `socket.socket()` → creates network socket
- `sock.sendto()` → sends UDP packet
- `sock.recvfrom()` → receives UDP packet
- `sock.bind()` → binds socket to IP address and port
- `sock.setsockopt()` → sets socket options (broadcast and timeout)

`websockets`:

- `websockets.connect()` → establishes websocket connection

`asyncio`:

- `asyncio.run()` → runs the async event loop
- `asyncio.gather()` → runs multiple coroutines concurrently
- `asyncio.get_running_loop()` → gets the active event loop
- `loop.run_in_executor()` → executes blocking input function asynchronously

`pyperclip`:

- `pyperclip.copy()` → copies argument into clipboard

`threading`:

- `threading.Thread()` → runs a thread in the background

`flask`:

- `Flask()` → initializes flask application
- `Sock(app)` → initializes websocket integration
- `ws.recieve()` → receives websocket message
- `ws.send()` → sends websocket message

MySQL

All received data is backed up to a MySQL database running on localhost on the server for persistence (retention of data even in case of server shutting down).

```python
def init_db():
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password=tokens.sqlServerPassphrase,
            database="clipboard_db"
        )
        cursor = conn.cursor()
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS clipboard_history (
                id INT AUTO_INCREMENT PRIMARY KEY,
                content TEXT NOT NULL,
                timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            )
        """)
        conn.commit()
        return conn
    except mysql.connector.Error as err:
        print(f"MySQL Error: {err}")
        sys.exit(1)

db_conn = init_db()
db_cursor = db_conn.cursor()
```

```
try:
    db_cursor.execute("INSERT INTO clipboard_history (content) VALUES (%s)", (data,))
    db_conn.commit()
except mysql.connector.Error as e:
    print(f"MySQL insert error: {e}")
```

Code

utils.py:

```
from datetime import datetime, timedelta, timezone

sqlServerPassphrase = "..."

f = open(f"./logs/{datetime.now(timezone(timedelta(hours=4))).strftime('%d-%m-%Y--%H-%M-%S')}.txt", "w")

def log(msg: str, code: int = 0) -> None:
    label = {0: " OK ", 1: "INFO", 2: "WARN", 3: "FAIL"}
    formattedMessage = f"[{label[code]}] -
{datetime.now(timezone(timedelta(hours=4))).strftime('%H:%M:%S')} - {msg}"
    f.write(formattedMessage + "\n")
    f.flush()

def closeLogFile():
    f.close()
```

server.py:

```python
import sys, subprocess, socket, threading, utils

try:
    from flask import Flask
    from flask_sock import Sock
    import mysql.connector as msc
except ImportError as e:
    missing = getattr(e, "name", None) or str(e).split("'")[1]
    pkg = {"flask": "flask", "flask_sock": "flask-sock", "mysql": "mysql-connector-python"}.get(missing, missing)
    print(f"Missing module: {missing} (pip package: {pkg})")
    o = input(f"Install {pkg}? (y/n): ").strip().lower()
    if o == "y":
        subprocess.check_call([sys.executable, "-m", "pip", "install", pkg])
        print(f"{pkg} installed. Please rerun the script.")
    else:
        utils.closeLogFile()
        sys.exit(1)

db_conn = None
db_cursor = None

def init_db():
    global db_conn
    global db_cursor
    try:
        db_conn = msc.connect(
            host="127.0.0.1",
            user="clipuser",
            password=utils.sqlServerPassphrase,
            database="clipboard_db"
        )
        db_cursor = db_conn.cursor()
        db_cursor.execute("create database if not exists clipboard_db")
        db_cursor.execute("""
            CREATE TABLE IF NOT EXISTS clipboard_history (
                id INT AUTO_INCREMENT
                content TEXT NOT NULL,
                timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            )
        """)
        utils.log("Started server")
        db_conn.commit()
    except Exception as e:
        utils.log(e, 3)
        print(e)
    except msc.Error as err:
        utils.log(err, 3)
        print(f"MySQL Error: {err}")
        utils.closeLogFile()
        sys.exit(1)
```

```python
init_db()

app = Flask(__name__)
sock = Sock(app)

clients = set()

def getLocalIP() -> str:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        s.connect(('10.255.255.255', 1))
        IP = s.getsockname()[0]
    except Exception:
        IP = '127.0.0.1'
    finally:
        s.close()
    return IP

def udpListener():
    while True:
        try:
            sock_udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            sock_udp.bind(('', 6969))
            utils.log("Setup UDP listener")
            print("Listening for discovery")

            while True:
                data, addr = sock_udp.recvfrom(1024)
                print(f"[UDP] Received: {data} from {addr}")
                if data == "DISCOVER_clpx.services.homelab.ree".encode("utf-8"):
                    sock_udp.sendto(f"{getLocalIP()}:{6262}/".encode(), addr)
        except Exception as e:
            utils.log(e, 3)
            print(e)
            utils.closeLogFile()
            sys.exit(1)
```

```python
@sock.route('/ws')
def websocket(ws):
    clients.add(ws)
    utils.log(f"Client connected: {ws}")
    print(f"Client connected: {ws}")
    try:
        while True:
            data = ws.receive()
            if data is None:
                break

            print(f"Received: {data}")

            try:
                db_cursor.execute("INSERT INTO clipboard_history (content) VALUES (%s)", (data,))
                db_conn.commit()
            except msc.Error as e:
                utils.log(e, 3)
                print(f"MySQL error: {e}")

            dead_clients = []
            for client in clients:
                try:
                    client.send(data)
                    utils.log("Copy function performed")
                    print(f"Sent: {data.encode('utf-8')}")
                except:
                    dead_clients.append(client)

            for client in dead_clients:
                clients.discard(client)
    finally:
        clients.discard(ws)
        utils.log(f"Client disconnected: {ws}")
        print(f"Client disconnected: {ws}")


if __name__ == '__main__':
    try:
        threading.Thread(target=udpListener, daemon=True).start()
        app.run(host='0.0.0.0', port=6262)
    except KeyboardInterrupt:
        utils.log("Program interrupted by user", 3)
        print("Program interrupted by user")
        utils.closeLogFile()
        sys.exit(1)
```

client.py:

```
import sys, subprocess, asyncio, socket

try:
    import websockets
    import pyperclip
except ImportError as e:
    missing = e.name
    pkg = {"websockets": "websockets", "pyperclip": "pyperclip"}.get(missing, missing)
    print(f"Missing module: {missing} (pip package: {pkg})")
    o = input(f"Install {pkg}? (y/n): ").strip().lower()
    if o == "y":
        subprocess.check_call([sys.executable, "-m", "pip", "install", pkg])
        print(f"{pkg} installed. Please rerun the script.")
    else:
        sys.exit(1)


IP, PORT = None, None

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
sock.settimeout(5)

print("Locating CLPX server")
for i in range(3):
    print(f"Attempt ({i + 1}/3)...")
    try:
        sock.sendto("DISCOVER_clpx.services.homelab.ree".encode("utf-8"), ('255.255.255.255',
6969))  data, addr = sock.recvfrom(1024)
        print(f"Found server at {data.decode("utf-8")}")
        IP, PORT = data.decode("utf-8").rstrip("/").split(":")
        PORT = int(PORT)
        break
    except socket.timeout:
        print("No response, retrying...")


async def listen(ws):
    async for message in ws:
        print(f"\nReceived from server: {message}")
        pyperclip.copy(message)
```

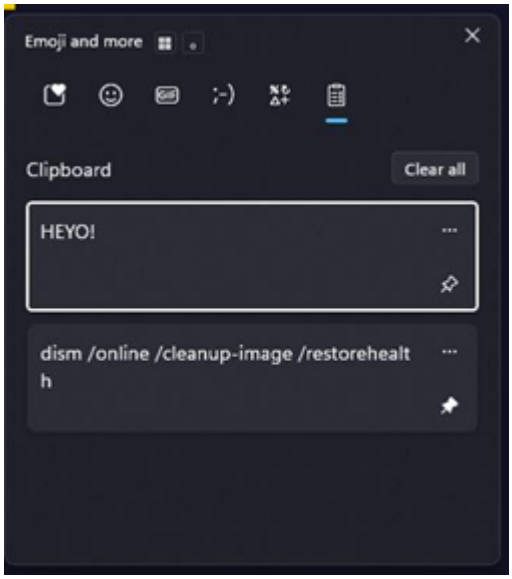
```
async def send_input(ws):
    loop = asyncio.get_running_loop()
    while True:
        user_input = await loop.run_in_executor(None, input, "\nEnter message:
")      message = str(user_input).strip()
        if message == "":
            continue
        await ws.send(message)
        print(f"\nSent to server: {message}")

async def main(uri: str):
    async with websockets.connect(uri) as ws:
        await asyncio.gather(
            listen(ws),
            send_input(ws)
        )


if __name__ == '__main__':
    try:
        if not IP or not PORT:
            print("Did not find a CLPX server.")
            sys.exit(1)
        asyncio.run(main(f"ws://{IP}:{PORT}/ws"))
    except KeyboardInterrupt:
        print("Program interrupted by user")
        sys.exit(1)
```

Output

```
PS D:\Programming\CLPX> python .\server.py
Listening for discovery
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:6262
 * Running on http://192.168.1.3:6262
Press CTRL+C to quit
[UDP] Received: b'DISCOVER_clpx.services.homelab.ree' from ('192.168.56.1', 50813)
Client connected: <simple_websocket.ws.Server object at 0x000001D3569F8D70>
Received: HEYO!
Sent: b'HEYO!'
Client disconnected: <simple_websocket.ws.Server object at 0x000001D3569F8D70>
192.168.1.3 - - [31/Aug/2025 11:12:26] "GET /ws HTTP/1.1" 200 -
```




```
PS D:\Programming\CLPX> python .\client.py
Locating CLPX server
Attempt (1/3)...
No response, retrying...
Attempt (2/3)...
Found server at 192.168.1.3:6262/

Enter message: HEYO!

Sent to server: HEYO!
```

Limitations

- Only works on a LAN, devices must be on a shared network
- All devices get a shared clipboard, no user groups
- Single centralized server setup (no redundancy or failover)

Bibliography

- Flask doc. → https://flask.palletsprojects.com/en/stable/
- MySQL doc. → https://dev.mysql.com/doc/connector-python/en/