Faculty of Engineering & Technology

Department of Computer Science and Engineering

(Artificial Intelligence & Machine Learning)

Jain Global Campus, Kanakapura Taluk - 562112
Ramanagara District, Karnataka, India

2024-2025

(8th Semester)

A report on

Assignment Work

"File Sharing Application with Python GUI"

Submitted in fulfilment for the project work

Bachelor of Technology

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Machine Learning)

Submitted by

Divyanshu Sharma (21BTRCL039)

Under the guidance of

S. Sivaram

Trainer

PhD. In Cybersecurity

Introduction

The File Transfer System is a desktop-based application developed using Python's Tkinter library for the graphical user interface and the socket library for network communication. This project enables seamless file sharing between a sender and a receiver over a local network.

Objectives

- To create a simple and user-friendly file transfer system.
- To implement client-server communication using sockets.
- To provide a graphical interface for file selection and transfer.
- To ensure a smooth user experience with features like reset and error handling.

Technologies Used

- Programming Language: Python
- GUI Library: Tkinter
- Networking Library: Socket
- Multithreading: Used to handle multiple client connections efficiently.

System Architecture

- The system consists of two main components:
- Sender Application: Selects and sends files.
- Receiver Application: Receives and saves files.
- Both applications communicate over a TCP/IP socket connection.

Appendix

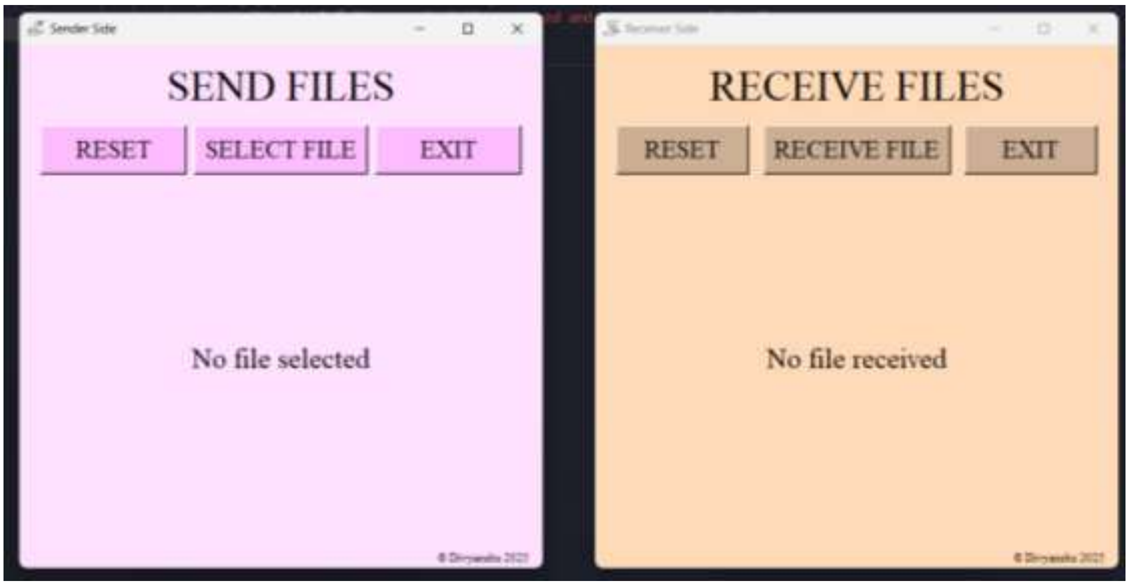
Sender.py:

```
def start_server(file_path):

    global server_socket

    if server_socket is None:

        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        server_socket.bind((SERVER_HOST, SERVER_PORT))

        server_socket.listen(1)
        print(f"[*] Server listening on {SERVER_HOST}:{SERVER_PORT}")
    while True:
        try:
            if server_socket.fileno() == -1:
                print("[!] Server socket is closed. Re-initializing...")
                server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                server_socket.bind((SERVER_HOST, SERVER_PORT))
                server_socket.listen(1)
            client_socket, client_address = server_socket.accept()
            print(f"[*] Accepted connection from {client_address[0]}:{client_address[1]}")
            threading.Thread(target=handle_client, args=(client_socket, file_path)).start()
        except OSError as e:
            print(f"[!] Socket error: {e}")
            messagebox.showerror("Socket Error", f"An error occurred with the socket: {e}")
def handle_client(client_socket, file_path):
    try:
        with open(file_path, 'rb') as file:
            file_data = file.read()
            client_socket.sendall(file_data)
            print("[*] File sent successfully.")
            messagebox.showinfo("Success", "File sent successfully.")
    except FileNotFoundError:
        print("[!] File not found.")
        client_socket.sendall(b'File not found')
    finally:
        client_socket.close()
```

Receiver.py:

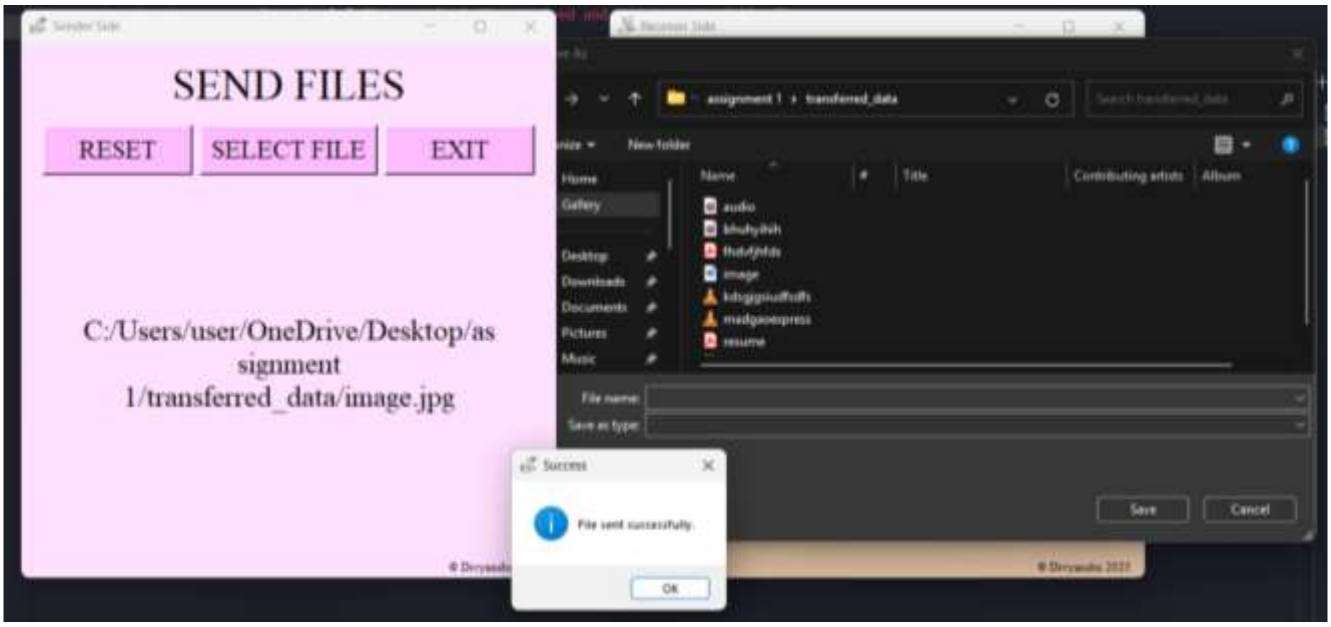
```
def receive_file():
    global client_socket
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        client_socket.connect((SERVER_HOST, SERVER_PORT))
        file_name = filedialog.asksaveasfilename(defaultextension=".*")
        if file_name:
            with open(file_name, 'wb') as file:
                while True:
                    file_data = client_socket.recv(BUFFER_SIZE)
                    if not file_data:
                        break
                    file.write(file_data)
            file_label.config(text=f"File saved as: {file_name}")
            print("[*] File received and saved.")
            messagebox.showinfo("Success", "File received and saved successfully.")
        else:
            print("[!] No file name provided.")
    except Exception as e:
        print(f"[!] Error: {e}")
        messagebox.showerror("Error", f"An error occurred: {e}")
    finally:
        client_socket.close()
        client_socket = None
```

Screenshots



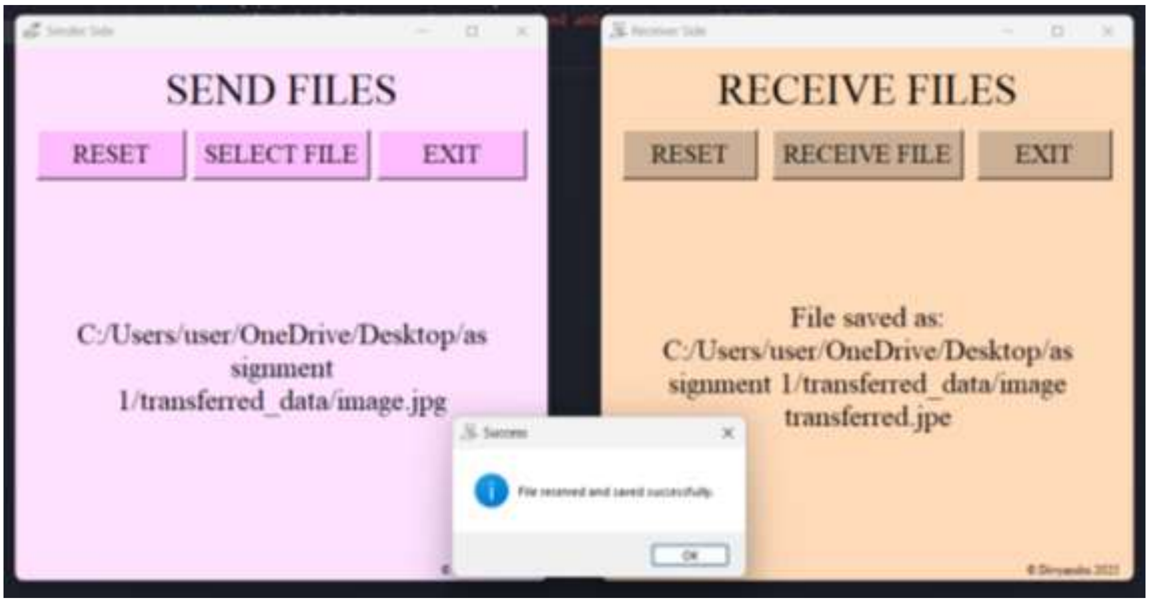
No file selected

No file received



C:/Users/user/OneDrive/Desktop/as
signment
1/transferred_data/image.jpg

File sent successfully.



C:/Users/user/OneDrive/Desktop/as
signment
1/transferred_data/image.jpg

File saved as:
C:/Users/user/OneDrive/Desktop/as
signment 1/transferred_data/image
transferred.jpe

File received and saved successfully.

Conclusion

This project successfully demonstrates a basic yet effective file transfer system using Python, Tkinter, and sockets. The application provides a user-friendly interface for selecting, sending, and receiving files over a local network. With its robust error handling and simple implementation, this system serves as a fundamental model for more advanced file-sharing applications.