Rene Reyna

Dr Habib M. Ammari, Ph.D., Ph.D.

CSCE 3335

February 12, 2025

Project 1: Problem Analysis, Problem Solving, and Programming –

Socket Programming Using TCP/IP

**Problem Statement**

This project involves developing a client-server program using TCP/IP sockets. The client sends a sequence of letters (‘T’, ‘A’, ‘M’) ending with a ‘#’ character to the server. The server sorts the sequence and returns the result. The program features both a graphical user interface (GUI) and command-line functionality.

**Proposed approach/solution**

The program was structured with three main components: the GUI, the networking module, and automated unit tests. The GUI, created with Tkinter, allows the user to start the client and server, input sequences, send data, run tests, and exit the program. It also displays real-time responses from the client and server.

The networking module manages the client and server communication using sockets. Once initialized, the server listens for connections, processes sequences from the client, and returns sorted results. Subprocesses are used to handle the client and server executions, ensuring smooth communication between the two.

Automated testing was implemented using Pytest to verify client-server functionality and sorting accuracy. Tests can be run directly from the GUI, and their results are displayed immediately

**Algorithm**

The sorting process used is based on the Dutch National Flag algorithm. This algorithm efficiently arranges letters into the sorted order by using three segments to divide the letters, representing ‘T’, ‘A’, and ‘M’. By maintaining three pointers: low, mid, and high, the algorithm arranges all ‘A’s first, followed by ‘M’s, and finally ‘T’s. The algorithm uses linear traversal, swapping elements in place, which provides both speed and efficiency. This algorithm was chosen for simplicity and performance.

Additionally, subprocesses are integrated within the GUI to manage client and server operations simultaneously without blocking the interface. Threading is employed to capture and display server responses in real time. This combination of algorithm and subprocesses was essential for ensuring the program worked reliably.

**Tests**

The program was tested with various inputs, such as 'TAM#', 'MAT#', and 'ATM#', with the expected output being the sorted sequence of letters. Pytest was used to automate these tests, ensuring consistent results and error detection.

**Commented Code**A screen shot of a computer program

AI-generated content may be incorrect.

gui\_main.py

**A computer screen shot of a program code

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.**

Main.py

networking.py

**A screenshot of a computer program

AI-generated content may be incorrect.**

algorithm.py

**Tests (input given by the user and output generated by your program execution)**A computer screen shot of a program code

AI-generated content may be incorrect.

Unit Tests

A screen shot of a computer program

AI-generated content may be incorrect.

Unit Test Results

A screenshot of a computer program

AI-generated content may be incorrect.*A screenshot of a computer

AI-generated content may be incorrect.*

A screenshot of a computer

AI-generated content may be incorrect.

**Lessons Learned**

Through this project, I gained valuable experience with socket programming and GUI development using Tkinter. I also deepened my understanding of algorithmic problem-solving, particularly through implementing the Dutch National Flag algorithm. Additionally, I learned the importance of proper subprocess management for ensuring proper and efficient program execution.