Implementation Log

# 1.Set Up the Server

**Description**: Created the server-side code to handle incoming client connections. This involved setting up a socket, binding it to a specific IP and port, and listening for incoming connections.

**Problem: #1: Server Not Binding Properly**

**Description**: Encountered an issue where the server failed to bind to the specified IP and port.

**Solution**: Checked and corrected the specified IP and port. Ensured no conflicts with other processes using the same IP and port. Verified network connectivity.

# 2. Step: Implement Diffie-Hellman Key Exchange

**Description**: Integrated Diffie-Hellman key exchange for secure communication between the server and clients. This involved generating prime and generator values, exchanging public keys, and calculating shared secret keys.

**Problem**: #2: Inconsistent Key Exchange

**Description**: Discovered inconsistencies in the Diffie-Hellman key exchange process between the client and server.

**Solution**: Ensured that both the client and server followed the same Diffie-Hellman key exchange process. Verified the correctness of prime and generator values.

# 3. Step: Implement Menu and Game Logic

**Description**: Developed the menu system and game logic for the number guessing game. The server sends a menu to clients, allowing them to choose between playing the game or quitting.

**Problem**: #3: Incorrect Menu Display

**Description**: Clients were receiving an incorrect menu or not seeing the menu at all.

**Solution**: Verified the display\_menu function in the server, ensuring that the encryption key was correctly passed. Ensured the client properly handled encrypted menu options.

# 4. Step: Implement Number Guessing Game

**Description**: Added the number guessing game functionality. Clients can make guesses, and the server provides feedback based on the correctness of the guess.

**Problem**: #4: Threading Issues

**Description**: Encountered threading issues causing unexpected behaviour, such as data

**Solution**: Implemented proper thread synchronization using locks to ensure thread safety.

# 5. Step: Implement Encryption

**Description**: Integrated a simple encryption algorithm for secure communication. The Encryption class encrypts and decrypts messages exchanged between the client and server.

**Problem**: #5: Insecure Encryption Algorithm

**Description**: Used a basic and insecure encryption algorithm.

**Solution**: Considered using a more secure encryption algorithm, such as AES. Updated the Encryption class accordingly.

# 6. Step: Implement Problem Handling

**Description**: Added Problem: handling mechanisms to address unexpected situations, such as invalid user inputs, client disconnections, and server shutdowns.

**Problem**: #6: Invalid Input Handling

**Description**: The server or client crashed when receiving unexpected input.

**Solution**: Implemented robust input validation to handle unexpected user inputs gracefully. Included Problem messages to guide users.

**Problem**: #7: Unexpected Client Disconnect

**Description**: Clients disconnected unexpectedly without proper cleanup.

**Solution**: Implemented a mechanism to handle client disconnections gracefully. Closed sockets properly and removed client states from the dictionary.

# 7. Step: Implement Graceful Server Shutdown

**Description**: Implemented a mechanism for the server to shut down gracefully, closing sockets and releasing resources upon receiving a shutdown signal.

**Problem**: #8: Lack of Graceful Server Shutdown

**Description**: The server didn't shut down gracefully when interrupted.

**Solution**: Implemented a signal handler to catch keyboard interrupts (e.g., KeyboardInterrupt) and gracefully shut down the server.