Phase 1: Requirement Analysis

1. **Understand Requirements**:
    - Determine the purpose of the socket program.
    - Determine client-server model or peer-to-peer model (chat program)
        1. Client-server model (default assignment)
            1. When a client sends a message to the server, the server simply replies with the fixed message saying “this is what you sent to me” along with the received message.
        2. Peer-to-peer model (**extra credit assignment: 2 more points when completely successfully**)
            1. A N-way chat program where connected clients can exchange messages. When a client sends a message to the server, the server will relay the message to other connected clients in real-time except for the sender (the client who sent the message). You may need to show the connected client as a list.
    - Identify the features for each model. You may need to write a user manual for this.
    - Select the programming language (e.g., Python with Tkinter/PyQt, Java with Swing/JavaFX).
    - Determine the network protocol (TCP or UDP). You will need to choose either TCP socket or UDP socket). The decision should be made before the design as a group.
2. **Define Scope and Goals**:
    - List functional and non-functional requirements.
        1. Depending on the model you chose to do, the functional features will vary.
            1. Base model’s features are described in the posted homework description.

2. N-way chat will have a list of connected client and message from the other client (not the server message). There will be no message coming from the server. Server program will have similar GUI but there will be one more button – Stop the server.

2. Non-functional features include easy-of-useness (no prior knowledge required for use), reliability, and maintainability (easy to install and fix errors). You may need to specify error handling, security, and performance expectations.

- Outline basic GUI elements (buttons, text areas, chat window, etc.).

3. List of task assignment for each member must be completed and included in here.

Phase 2: Design

1. **Network Design**:
   - **Client-Server Architecture**: Plan the roles of the client and server (e.g., multi-threaded server for handling multiple clients).
   - **Socket Communication**: Determine the communication protocol (TCP for reliable, ordered delivery or UDP for faster but unreliable transmission).
   - **Connection Management**: Establish connection initiation, maintenance, and termination processes.
2. **GUI Design**:
   - **Wireframe Design**: Sketch the layout for the GUI (use tools like Figma or Adobe XD).
   - **Component Design**: Decide on UI elements like buttons, text boxes, labels, and message displays.
   - **Event Handling**: Plan how the GUI interacts with the socket (e.g., send button triggers socket write, incoming message displays on chat area).
3. List of task assignment for each member must be completed and included in here.

Phase 3: Setup and Prototyping

1. **Basic Socket Setup**:
    - Write a simple server that can accept connections from clients.
    - Write a basic client that can connect to the server.
    - Test sending and receiving messages between client and server (console-based testing at this stage is ok).
2. **Basic GUI Setup**:
    - Build a basic static GUI with minimal interaction.
    - Integrate GUI with backend logic to send user input from the GUI to the socket layer.
3. **Prototype Testing**:
    - Test GUI interaction with basic socket communication.
    - Ensure messages sent from the GUI are delivered to the server and that responses appear in the GUI.
    - Begin testing on multiple clients (e.g., local machines or over LAN).
4. List of task assignment for each member must be completed and included in here.

Phase 4: Core Development

1. **Core Socket Functionality**:
    - Implement core socket functionalities like establishing a connection, handling multiple clients (extra credit project only), and sending/receiving messages.
2. **GUI Enhancement**:
    - Enhance the GUI with better design, responsive elements, and dynamic content updates (real-time message display).

- Implement additional features like:
  - Input validation (e.g., IP address, port numbers).
  - Connection status indicators.
  - Scrollable chat area.

Phase 5: Testing and Debugging

1. **Unit Testing**:
   - Test socket communication (connections, message exchange, disconnections).
   - Test all GUI components for functionality (buttons, forms, displays).
   - Test interaction between GUI and socket components.
2. **Integration Testing**:
   - Run tests with multiple clients to ensure scalability and concurrency handling (extra credit only).
   - Test the entire system (e.g., message exchanges).
3. Wireshark analysis must be performed and explained

List of task assignment for each member must be completed and included in here.

** All other deliverables, reference group homework description.