

# WPF Laboratory Part

## Group Chat

Your task is to develop a simple Group Chat client using WPF.

## Scoring

### 1. Layout (1 point)

- Application window should include:
  - A menu bar at the top.
  - A list view with chat bubbles in the center.
  - An input field with a Send button at the bottom.
- Window requirements:
  - Minimum size: 420x240
  - Initial size: 800x480

### 2. Menu Bar (1 point)

- Includes two top-level menus: File and Help.
- File menu options:
  - Connect — Enabled only when not connected. Changes client state to "connected".
  - Disconnect — Enabled only when connected. Changes client state to "disconnected".
  - Exit — Closes the application.
- Help menu option:
  - About — Displays a message box with information about the application.

### 3. Messages Area (4 points)

Contains a scrollable list of messages, including:

- Chat Bubbles (3 points):
  - Appearance:
    - \* Rounded DarkOrange rectangle.
    - \* White bold username at the top.
    - \* White, read-only, selectable message content in the middle.
    - \* LightGray timestamp at the bottom-right.
    - \* Message bubbles must alternate their horizontal alignment:
      - Even-indexed messages aligned to the left.
      - Odd-indexed messages aligned to the right.
  - Timestamp display rules:
    - \* "Now" if message sent < 1 minute ago.
    - \* "Xm ago" if sent < 15 minutes ago.
    - \* "HH:mm" format if sent < 1 day ago.
    - \* "dd/MM/YYYY" format if sent > 1 day ago.
  - Timestamp tooltip shows exact date and time of sending.
  - Timestamp updates dynamically as time passes.
- System Messages (1 point):
  - Displayed as DarkGray centered labels with informational text.
  - Changing state to "connected" or "disconnected" generates a "Connected"/"Disconnected" system message.

### 4. Input Area (2 points)

- Contains:

- A text input field:
  - \* Text wraps automatically.
  - \* Scrollbars are disabled.
  - \* Height adjusts dynamically based on the amount of text entered.
- A "Send" button positioned next to the input.
- Message sending behavior:
  - Pressing Enter sends the message.
  - Pressing Shift + Enter inserts a new line in the input.
  - Sent messages:
    - \* Appear in the Messages Area.
    - \* Are sent alternately by two different users
    - \* Clear the input field after sending.

## Hints

- To update the timestamp text periodically, consider using a `DispatcherTimer` for timed updates on the UI thread.
- To handle the Enter key behavior in the input area:
  - Set the `AcceptsReturn` property to `true` on the input control to allow multiline input.
  - Subscribe to the `PreviewKeyDown` event to differentiate between `Enter` (send message) and `Shift + Enter` (insert newline).
- To apply different data templates for user and system messages, use a `DataTemplateSelector`.