Milestone 2

General

After the success of the first milestone, you are now requested to further expand the functionality of your chat-room client.

After using your chat room client for several months your clients complain that interactions with your client are too cumbersome (difficult). To improve this, they ask you to build a pretty graphical user interface (GUI) for it. This interface should support all of the existing functionality (with some changes, see below). Additionally, the client will automatically pull new messages, update the view, and support filtering and sorting the messages. You and your team are tasked to upgrade your current chat-room client and generate a new version that supports these new demands.

Goals:

1. GUI.
2. Practice team work & version control.
3. Practice N-tier architecture.
4. Practice NUnit.
5. Practice C#.
6. Experience right use of OOP with C#.

Functional Requirements

The GUI will have at least the following two windows:

1. Login / registration.
2. The chat-room window, which will include:
    a. The received messages display.
    b. Buttons for filtering and sorting the messages.
    c. An option to send new messages.

The required GUI functionalities are partly taken from milestone 1:

1. Registration.
2. Login/Logout.
3. ~~Retrieve last 10 messages from server.~~ (check out the new requirements).
4. ~~Display last 20 retrieved messages (without retrieving new ones from the server), sorted by the message timestamp~~. (check out the new requirements).
5. ~~Display all retrieved messages (without retrieving new ones from the server) written by a certain user (identified by a username and a group id), sorted by the message timestamp.~~ (check out the new requirements).

6. Messages displayed on screen should include all message details received from the server.
7. Write (and send) a new message (max. Length 150 characters).
8. Exit (logout first).

New requirements:

1. While there is a signed-in client, every 2 seconds (see supplied materials for help):
    a. Retrieve last 10 messages from the server.
    b. Refresh the received messages view with the new data.
2. Add an option to the GUI for filtering the messages by:
    a. The group <g_id>.
    b. The user <g_id, nickname>.
3. Add an option to the GUI for sorting the messages ascending and descending by the:
    a. Message timestamp.
    b. Nickname.
    c. g_id, nickname, and timestamp (all ascending or all descending). Example at the appendix.

Non-functional Requirements

1. All the non-functional requirements from Milestone 1.
2. Implement at least 7 Unit Tests.
    a. Instruction and examples are given in lecture 4 slides.
    b. Test classes, methods names, and variables must follow the name conventions , as explain in lecture 4.
3. Usability:
    - This GUI must be comfortable for both experienced and inexperienced users.
    - Buttons in the interface should contain only short text descriptions (no more than two words) or a descriptive icon (preferably both).
    - Text fields and text boxes should be large enough to comfortably accommodate user input, but not so large that they are unwieldy.
    - Data input by users should be validated, and input of invalid characters should be blocked (e.g. empty messages, nicknames, etc.).
    - For additional guidelines, read this page on Good User Interface Design Tips.

Supplied Material

Timers in C#

The .NET Timer class allows for scheduling a method call in a specific time. A timer may run once or repeatedly depending on the configuration. Using timers you can make your application execute some method periodically without the need for occupying the main program context (GUI remain active).

Using the "AutoReset" property of the timer you can determine whether the timer repeats itself. If this property is set to "false" then the "Enabled" property of the timer will be set to "false" after the first execution and it must be manually set to "true" in order to re-enable the timer.

For example: A timer set to 2 seconds with the "AutoReset" property set to "true" can call a predefined method "printTime" that prints the time to the console. The result will be a console print of the time every two seconds.

A code example and more information can be found here: MSDN.Timer

Note: Using more than one timer at a time can cause synchronization issues, we will not learn about synchronization so it is best that you just make sure you don't have more than one timer running at any given time. You can always cancel running timers in case you want to activate new ones.

Additional tip is not to use shared variable with the timer, meaning that you should not use variables that are being used outside the timer function.

General Guidelines

- Version Control: You are expected to work as a team and use source control (GitHub specifically). More info in the "Project information" document.
- Document your code thoroughly.

Design instructions

- In this assignment, most of your work will concentrate on the GUI. Yet, there are some required changes to the business layer. Please follow the design instructions from Milestone 1.
- While you are not required to make the application a beautiful work of art, you must observe the usability non-functional requirement defined in this milestone.

Submission Dates, Deliverables and Grading

General Submission Notes

1. You need to work with git. All team members must commit.
2. You need to make sure your git repository is private; the name of the repository as requested above; and course staff are added as collaborators to the repository.
3. Your group is registered, and a link to your git repo is filled.

4. Every hour late in the submission will cause a reduction of 5 points from your grade.

Milestone Deadline - 12.05 23:59

1. Your client should be in the master branch.
2. Your code should be documented.
3. Your design documents (HLD & LLD) should match and reflect your actual program design. **Please update them and re-push them to the repository** before the milestone deadline.
4. You need to tag your latest git version as "milestone_<#>". This is how we will determine that you have finished your code and documentation on time.

Milestone Presentation

The client functionality will be examined during one of the lab sessions, while the rest (e.g., design documents, documentation, etc.) will be examined offline.

All team members must arrive at least 10 minutes before your time slot (a schedule will be published) and present your work. The following documents should be ready and **opened in different tabs** on a computer at your time slot:

a. Your GIT repo webpage.
b. Your project on visual studio, connected to the server.
c. Your design documents.

Appendix:

How to sort more than one field:

Example: if we want to sort by group_id, user_name and timestamp in descending order (remember that you need to support only descending or ascending for all three at a time).

The raw data is:

- 1, ben, 12:00
- 2, benny, 12:00
- 1, ben, 12:01
- 2, asaf, 12:00
- 1, tania, 12:00

The sorted data will be:

- 1, tania, 12:00

- 1, ben, 12:01
- 1, ben, 12:00
- 2, benny, 12:00
- 2, asaf, 12:00

Explanation: first the group_id field appears sorted over the whole list. For each group_id the user_names are ordered, and last, for each user_name the timestamps are ordered.