Milestone 1

Requirements Document

General

In this milestone, you are requested to implement a CLI desktop client for a chatroom. Your client will connect and communicate with a server that we have deployed for you.

Your client will retrieve messages from the server, display and write messages, and handle users login and registration.

Your client will have three layers constructed by you – presentation layer (CLI), logic layer, and persistent layer. In addition we will supply a communication layer for your client in order to communicate with the server.

Goals

The goals of this milestone are:

1. Practice N-tier architecture
2. Practice team work & version control
3. Practice C#
4. Experience right use of OOP with C#
5. Understand High Level Design and write a Low Level Design (HLD & LLD).

Business and Integrity Rules

1. User is identified by:
    a. A group ID, which is set in the registration google sheet (first column).
    b. A nickname, which must be unique in a group.
2. A message received by the server has the following attributes (besides the message itself):
    a. A unique identifier (called GUID, global unique identifier).
    b. The time that it was received by the server.
    c. The user information.
    d. The message body.

Functional Requirements

1. A client command-line interface (CLI) that will support the following operations:
    a. Registration.
    b. Login/Logout.
    c. Retrieve last 10 messages from server.
    d. Display last 20 retrieved messages (without retrieving new ones from the server), sorted by the message timestamp.
    e. 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.
    f. Write (and send) a new message (max. Length 150 characters).
    g. Exit (logout first).
2. Messages displayed on screen should include all message details received from the server.

Non-functional Requirements

1. Persistency:
    a. In this assignment, persistent data is stored in local files.
    b. The following data should be persistent:
        i. Users data.
        ii. Received messages (including messages returned by calling Communication.Instance.Send(), see below.
    c. Persistent data should be restored once the CLI client starts.
2. Logging:
   You must maintain a log which will track all errors in the system. Note that "error" does not necessarily mean an exception (see next item) that is "thrown" or "raised" by your runtime environment, but any situation which counts as invalid in our domain. Some guidelines:
    a. Tag entries with their severity / priority
    b. Provide enough information to understand what went wrong, and where
    c. Avoid storing entire stack traces directly in the log (they are more verbose than useful)
    d. Use a shelf Logger. Some examples: log4net, Observer logger, Composition Logger, end there are many more.
3. Exception Handling is the process of responding to the occurrence, during computation, of exceptions – anomalous or exceptional conditions requiring special processing – often changing the normal flow of program execution. Your program is expected to operate even when error occurs:
    a. Handle any malformed input.
    b. Handle logic errors (e.g., login for non-existing user, etc.).

Supplied Material

Communication layer for the chat client

You are supplied with a client side communication module which is capable of getting the last 10 messages from the server, and sending a message. Be aware that the server sets the message's GUID and message's DateTime. The communication class is designed using a singleton pattern and you will get it by typing ClassName.Instance.

The server can be reached at: http://ise172.ise.bgu.ac.il:80 (only in BGU network).

The structure of the communication module:

```
namespace MilestoneClient.CommunicationLayer;
Sealed class Communication {
     public CommunicationoMessage Send(string url,
     CommunicationoMessage msg);
     public List<CommunicationoMessage> GetTenMessages(string
     url);
}
```

In order to use the communication layer you need to install the following Nuget: newtonsoft.json:
The communication model is available for downloading from the course's website.

Server side

Although you will not use it directly, some information regarding the server is brought here:

1. The server is capable of receiving new messages and storing them. The server generates a unique ID for each message.
2. The server is capable of sending the user the last ten messages.

The server address is ise172.ise.bgu.ac.il. You can open it's URL in your web browser to view the latest messages. The server is available only from within the BGU network. You can get it's source code from Github and run it locally, but when presenting, you will use the server in BGU, make sure your code runs with it.

Do not flood the server: users that will run more than 20 queries/requests in 10 seconds will be blocked and their grade will be reduced!

You will need to send the server a valid group id in all communications.

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.
- Pay attention to "Magic numbers".

Design instructions

- Before starting your design (and of course writing your code), consider the following:
    - Are there any operations that a user can/should do?
    - Are there any operations that a message can/should do?
    - Where do each method belongs? (login & logout/write & retrieve messages, etc.)
    - Do you need any additional class for
    - Can you think of new requirements that the client (the course staff) will probably request? Can you make the design and the code versatile enough to support such requests (of course without too much work on your side because we might not ask for it)?
- N-Tier structure: pay attention to right use of the N-tier model. For example, a common mistakes is accessing the "Data" layer directly from the "Presentation" layer.
- OOP principles: remember OOP principles and use them (Encapsulation, Abstraction, Interface, and Inheritance).
- Having a persistent layer does not mean that the data objects are also stored there. In fact, these data object can have some logic and method (e.g., the user object). Moreover, our data should be stored in the RAM for fast retrieval. The persistent layer should be called only upon system startup (for restoring the persistent data) and upon data update (e.g., registration of a new user).

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.

Design Deadline - 18.03 23:59

A software design precedes the coding phase, and so it will be here. You need to prepare for your application a high level design document and a low level design document (one page each). The documents should be pushed to the git repository by this deadline. We supply a basic HLD, you are expected to update if needed.

Milestone Deadline - 08.04 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.