*Low Level Design – MileStone 1(Submission 2)*

Detailed description of the Classes (fields and methods):

**Solution Files**

**Classes:**

**Program**

Methods:

* **void Main (String[] args)** : Main method. Calls Launch() method.
* **void Launch()** : Calls ChatRooms and GUIs Initiate methods to start main components of chat room.

**Presentation Layer**

**Classes:**

**GUI**

Methods:

* **void Initiate (string chatRoomStatusMessage):** Starts a user interactive menu(EntryMenu or Menu) determined by current user logged status (logged in or not).
* **void Menu():** Creates an interactive menu for logged users in which you can take all chat related actions. (calls ChatRoom class in order to perform all different operations).
* **boolean EntryMenu():** Creates an interactive menu for non-logged users in which you can take basic actions to enter the chat room . (calls ChatRoom class in order to perform all different operations).
* **string DisplayMessages (string messages):** Prints and displays a given string that represents user messages.
* **void ClearMessages():** Calls ChatRooms MessageFrame module method DeleteMessages() to delete all previous local stored messages. Shows notifications to user about the deletion process.
* **boolean Exit():** Shows notification to user about exiting the program.
* Calls ChatRooms loggedInUser method to perform Logout() and closes the program.
* **boolean Logout():** Shows user a notification aabout logging out of the program and calls ChatRoom Logout() method.
* **boolean Login(boolean logged):** Gets current user status (logged in\logged out) and shows notification according to it. If user is logged out, asks for user details to log in using ChatRooms Login() method.
* **boolean Register():** Shows notifications regarding registration process. Asks user details in order to perform registration using ChatRoom Register() method.
* **void WriteAndSend():** Asks for user input to be sent as a new message. Calls ChatRooms SendUserMessage() method.
* **void NewScreen():** A helper method which clears current screen and displays the programs title in the newly opened screen.
* **void MarkerMessage(int markedColor, string str):** A helper method that color text(string str) in marker notification color determined by the markedColor integer.
* **void ErrorMessage(int severity, string err):** A helper method that color text(string str) in error notification color determined by the severity integer.

**Business (logic) Layer**

**Classes:**

**ChatRoom**

Fields:

* **User loggedInUser** : the current user using the system.
* **MessageFrame messages** : A MessageFrame type holds and handles all message related occurrences.
* **EventLogger logger** : A logger responsible of logging and documenting all system important events.
* **String url** : Chat room IP\http address.
* **List<User> registeredUsers** : A list that holds all current registered users to the chat room.
* **UserHandler userHandler** : A handler responsible for all local system user related management.
* **CONT String GROUP\_NUM** : Authors constant group number.

Methods:

* **string Initiate()** : Initiates all ChatRoom Components, initializes all class fields and reads SystemFiles/Users.bin to get Registered users list from local system.
* **boolean Login(string username, string password)** : Validates user registration according to given user details (username and password) and logs in user to the system.
* **boolean Register(string username, string password, string groupID)** : Registers a new user with the following user details: username, password and group ID.

Checks if user with the same registration details already exist beforehand.

If user successfully registers writes updated Users list to SystemFiles/Users.bin.

* **User CheckRegisteredUser (string username, string password)** : Checks if a user with the given password and username is registered to the chat room using the registeredUsers list.
* **String CheckUserDetails (string username, string password)** : Checks user details (password and username) are valid according to following demands:
* A username can only contain only English ABC, digits and spaces.
* A valid username needs to be 2-15 characters only.
* A valid password needs to be 4-15 characters only and no spaces.
* **String SendUserMessage (string msgBody)**:Calls loggedInUser (current logged in user) and uses the SendMessage method to send a message with the content of string msgBody.
* **String RetrieveTenMessages()** : Calls MessageFrame messages and retrieves last 10 messages from server using the RetrieveTenMessages method in messages. Returns a string that indicates the number of messages retrieved.
* **void displayMessages(int number)** : Calls MessageFrame messages and gets a string representing last (int amount) messages using the messages DisplayMessages method.
* **void dispalyAllMessages()** : Calls MessageFrame messages and gets a string representing all current messages available in message cache using the messages DisplayAllMessages method.

**MessageFrame**

Fields:

* **Queue<Message> messages** : a stack type that contains all current messages saved in RAM.
* **MessageHnadler messageHnadler :** A handler used to manage all local message file integration.

Methods:

* **void NewUserMessage (Message message)** : Initiated when a user sends a message, after sending to server. Enqueues the message to the current message queue.

Writes updated message list to local SystemFiles.

* **int RetrieveTenMessages()** : Retrieves 10 last messages from server. Checks messages do not exist already and enquques them to local message list.

Writes updated message list to SystemFiles/Messages.bin.

* **string DisplayMessages(int amount)** : Gets last (int amount) messages from messages queue and creates and returns a string representing all the last messages.
* **string DisplayAllMessages()** : Gets all messages from messages queue and creates and returns a string representing all the messages.
* **boolean DeleteMessages() :** Resets local message list. Deletes and recreates SystemFiles/Messages.bin

**User : IComparable<User>**

Fields:

* + **string username** : User unique username chosen by the user.
  + **string password :** User password chosen by the user.
  + **boolean logged :** A status indicator to determine if the user is currently logged in or not.
  + **string groupID :** A group ID associated with the user.

Methods:

* **string sendMessage(string msgBody)** : Checks user message validity and sends the message to the server.
* **boolean Logout() :** Checks if the user is logged in, loggs user out. if the user is not logged in the method is will do nothing.
* **Int CompareTo (User user) :** Compares users by their usernames.

**Message : IEquatable<Message>**

Fields:

* + **string body** : the message body.
  + **DateTime time** : Message date and time.
  + **User user** : A user associated with the specific message.
  + **Guid guid :** A unique message identifier.

Methods:

* **boolean CheckMessageValidity(string msg)** : Checks the validity of message content according to the following demands :
* A valid message can only contain 1-150 characters.
* **string ToString()** : A ToString Method represents a message. Produces the following output:
* username says >> msgBody

Details :: Sent in: msgTime :: Group ID: userGroup ::

* **boolean Equals(Messge msg) :** Checks if to messages are equal by comparing their GUID. A method of the IEquatable<Message> interface.

**Persistent Layer**

**FileWriteRead<T>**

* + **void WriteToFile(String path, List<T> dataType)** : Write a list of objects to file in specified path.
  + **List<T> ReadFromFile(string path)** : Reads a file from given path and returns data read.
  + **boolean DeleteFiles(string path)** : Deletes file from given path.

**IHandler<T>** <<interface>>

* + **void Write(T list)** : implementation varies.
  + **T Read()** : implementation varies.

**MessageHandler : IHandler<List<Message>>**

Fields:

* + **FileWriteRead<Message> writeRead** : Uses FileWriteRead Type to write and read files.
  + **CONST string PATH** : A Constant path location on local system where the messages files is saved. (SystemFiles\Messages.bin).

Methods:

* + **void Write(List<User> users)** : Write a list of messages to local file path.
  + **List<User> Read()** : Reads from local file path a list of messages.
  + **boolean Delete()** : Deletes local messages file.

**UserHandler : IHandler<List<User>>**

Fields:

* + **FileWriteRead<User> writeRead** : Uses FileWriteRead Type to write and read files.
  + **CONST string PATH** : A Constant path location on local system where the registered users files is saved. (SystemFiles\Users.bin).

Methods:

* + **void Write(List<User> users)** : Write a list users to local file path.
  + **List<User> Read()** : Reads from local file path a list of users.

**EventLogger**Fields:

* + **ILog logger**: A logger implemented with log4net.
  + **StackFrame callStack** : A call stack to record logging file name, location and line number.

Methods:

* + **void Initiate()** : Starts callStack and loads logger xml configurations from log4net.config
  + **void log (int severity, string toLog)** : A helper method to log events to SystemFiles/Log/EventLog.log with a relevant callStack association.