Low Level Design – MileStone 2(Submission 3)

Detailed description of the Classes (fields and methods):

Solution Files

Classes:

Program

Methods:

- **void Program_Startup(object sender, StartupEventArgs e)** : Initiates program main components.. Calls Launch() method.
- **void Launch()** : Launches Chat-Room logical layer and GUI components.

Presentation Layer

Classes:

GUI

Fields:

- **static string _currntRegisteredUsersAmount** : A string describing the current registered users amount.
- **public static bool _soundStatus** : Indicating the state of the sound in the program.
- **private static SoundPlayer _player** : A Sound player that plays system sounds.
- **private static bool _userClose** : Defines if the program was user closed or closed internally.

Methods:

- **void Initiate(string chatRoomStatusMessage)** : Initiates the GUI components according to the current components state.
- **void PlaySound(String soundPath)** : Plays a sound from the sound path given if sound state is on.
- **void ColorTextToRichTextBox(RichTextBox box, FlowDocument flowDoc, string text, string hexColor, bool bold, bool underline)** : A method that allows to color components in a rich text boxes in the GUI.
- **void PerformClick(Button btn)** : Performs a programmatically initiated click on a certain button.
- **bool IsNumeric(string input)** : Checks if input values given is a numeric value
- **void WindowClose(Window toClose)** : Performs a programmatically initiated window close.

LoginWindow

Fields:

- **ChatRoomWindow _openChatRoomWindow** : Current related instance of a chat room window window.
- **String _currentUsername** : Current username entered in the username text box
- **String _currentPassword** : Current password entered in the password text box
- **String _groupID** : Current group ID entered in the groupID text box

Methods:

- **bool Login(bool logged)** : Tries to perform a login by calling chat room login method with the entered user details.
- **bool Register(string statusMessage)** : Tries to perform a user register by calling chat room register method with the entered user details.
- **void OpenChatWindow(string statusMessage)** : Opens a new chat room window and shows it.
- **void Clear_LoginRegister_TextBoxes(bool clearGroupTextBox)** : Cleares all user detailes entered in the user details text boxes.

ChatRoomWindow

Fields:

- **static LoginWindow _openLoginWindow** : Current related instance of a login window.
- **FlowDocument _shownMessagesDocument** : A document containing all current displayed mesasges.
- **Timer _messageLoadTimer** : A Timer that refreshes retrieves message by defined time interval.
- **DispatcherTimer _clearNotificationsTimer** : A Dispatcher Timer that cleares notification text box by defines time interval.
- **Double _chatBoxScrollbarVerticalOffset** : Current messages text box scroll bar vertical offset.
- **String _sortMethod** : A String representing the current active sorting method.
- **bool _ascendingOrder** : Defines wether the messges are displayed in ascending order or descending order.
- **bool _hasFilters** : Defines wether messages are currently filtered by any parameters.
- **bool _chatBubbleView** : Determines whether the current message view type is chat bubble or textual.
- **String[] _filterBy** : An Array of message filtering parameters.
- **String _currentUserMessageText** : Current end-user text inserted in the user message text box.
- **int _textSizeOffset** : A indentation value to the chat message font size.
- **WindowState _windowState** : Indicated the window size state.
- **int _dispalyedMessages** : Indicates how many messages are currently showing in the chat box.
- **const int MESSAGES_RETRIEVE_TIME_INTERVAL = 2** : A constant time in seconds to retrieve new messages in.
- **const int NOTIFICATION_DELETE_TIME_INTERVAL = 7** : A constant time in seconds to retrieve new messages in.
- **const int DEFAULT_FONT_SIZE_OFFSET = 6** : A constant time in seconds to retrieve new messages in.

Methods:

- **void RetrieveMessages()** : Calls the chat room to retrieve messages from server and calls the display messaages method to display the retrieved messages in chat box.
- **void DisplayMessages(int newMessages)** : Calls the load messages method with the number of new messages retrieved from server and an indicator if filters are enabled currently.
- **void LoadMessages(int newMessges, bool filtered)** : Calls the chat room methods to display messages according to the filtering state. Adds each message retrieved from the chat room display messages methods to the chat box Updates the current message counter.
- **void AddMessageToChatBox(bool userMessage, String[] message, Paragraph paragraph)** : Adds a message to a flow document by the correct layout and design (current user message or other user messages). Designs each message accordingly (Textual or chat bubble style).
- **void StartMessageTimer()** : Starts a timer that retrieves messages from server in a given constant time interval.
- **void MessageTimer_Tick()** : Called every constant time interval by the message timer to retrieve new messages.
- **void Start_Timers()** : Initiates both program timers, messages retrieval timer and notification deletion timer.
- **void UpdateMessageCounter()** : Updates and restarts the current dispalyed chat box messags.
- **void UpdateUserCounter()** : Updates the user counter label
- **bool CheckChatScrollBarScrolledToEnd()** : Checks if the chat box scroll bar is currently scrolled to the end of the chat box or not.
- **void ManageChatBoxScroller()** : Manages the chat box scroller behavior.Keeps scroller to the end of chat box if it is aready there.Otherwise scrolles to the last saves postition of to scroller.The goal is to not interupt user scrolling while viewing older messages.
- **void SetScrollbarTo_NewestMessages()** : Sets the scroll bar to the end of the chat box hence to the end of the check box. Saves current scrolled location.
- **void StopMessageTimer()** : Stops the messages retrieval timer.

- **void OpenLoginWindow(string statusMessage)** : Opens a new login window and shows it.
- **void CheckServerCommunication(string statusMessage)** : Calls the chat room method to check if currently gor server communication. Stops messages retrievel timer if no communication and shows relevant controls to try and re-establish connection.
- **void ServerCommunicationError_Message()** : Adds a error message to notification box indicating no server communication established.

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.
- **bool _serverCommunicationStatus**: Indicates whether the chat room is connected to server

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 CheckUserRegistered (string username, string password)** : Checks if a user with the given password and username is registered to the chat room using the registeredUsers list.
- **bool isRegisteredUsername(string username)** : Check input username is already taken by other user.
- **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(bool checkRetrieval)** : 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.
- **List<String[]> DisplayAllMessages(String sortMethod, bool ascending)** : Calls message frame and gets a string representing all existing messages by the curreny sorrting method and direction.
- **List<String[]> DisplayFiltered_Messages(String[] filterBy, String sortMethod, bool ascending)** : Calls message frame and gets a string representing all mesges corresponding the filters, sorting method and sorting direction given.
- **string CheckServerCommunication()** : Checks the connection to the server by trying to retrive messages.

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.
- **List<Message> SortingMethods(String sortMethod, bool ascending)** : Sorts all the stores messages Queue by the given parameters.
- **List<String[]> DisplayFiltered_Messages(String[] filterBy, String sortMethod, bool ascending)**: Gets all current messages and creates a list of String arrays indicating the seperated message parts. Adds only the messages corresponding to the filters given.
- **List<String[]> DisplayAllMessages(String sortMethod, bool ascending)**: Gets all current messages and creates a list of String arrays indicating the seperated message parts.
- **boolean DeleteMessages() :** Resets local message list. Deletes and recreates SystemFiles/Messages.bin

User : IComparable<User>, IEquatable<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.
- **bool 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.
- **bool Equals(User u)** : Checks if a user given as paramter is the same user as this current user. Users are same if they have same username, password and group ID.

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.
- **const int MAX_LENGTH = 150** : Maximum length of the message body.

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.

Unit Tests

Detailed description of the Unit Tests:

Business (logic) Layer Tests:

ChatRoomTests

- **void CheckRegister_UserExists_ReturnException()** : Tests the main user registration method .Test user exists in system after registration.Expects a message indicating the user registration failed.
- **void CheckUserRegistered_UserRegistered_ReturnUser()** : Test the method that check if a user is already registered in system. Registers a user and checks existence.

MessageFrameTests

- **void CheckNewMessageSaved_MessageSaved_ReturnsMessage()** : Tests addition of new messages to the message frame messages list. Assert messages added are equal to the messages on the messages list.
- **void CheckSortMessagesByUsername_MessagesSorted_ReturnsSortedList()** : Tests a message list sort by username after being shuffled. Asserts that the messages are sorted as a pre-defined sorted list.
- **void CheckSortMessagesByAllCriteria_MessagesSorted_ReturnsSortedList()** : Tests a message list sort by group ID then by username and then by timestamp after being shuffled. Asserts that the messages are sorted as a pre-defined sorted list.
- **void CheckMessageFiltering_MessagesFiltered_ReturnsFilteredList()** : Tests a message list filtering by username and group ID. Asserts that the messages are filtered as a pre-defined sorted list.

MessageTests

- **void CheckMessageValidation_MessegeValidityCheck_ReturnsFalse()** : Test message validation does not allow illegal inputs. Illegal inputs checked: Message content exceeds max message length, empty string message.

UserTests

- **void CheckSendMessage_MessegeValidityCheck_ReturnsException()** : Check user send message method.Test tries to send an illegal user message.
- **void Checklogged_loggedAfterRegister_Returntrue()** : Tests user is logged automatically after registration.

Persistent Layer Tests:

FileWriteReadTests

- **void CheckFileWriteRead_WriteRead_ReturnTrue()** : Tests File writes and reads correctly from file.
- **void CheckFileWriteRead_Delete_ReturnTrue()** : Tests file deletion.