**File: Messenger.Java**

In this file we created new menus and changed the starting message. The new menus will allow a user to create an account or login to an existing one. The user can then manage their contacts, blocks, chats, messages, or delete their own account. The user can also log out to log into a different account. Here are the added functions:

public static void AddToBlock(Messenger esql, String user)

This function asks the user to input a name. The program determines if there exists a contact

that can be added to the block list, and then blocks them. The user cannot block themselves.

public static void RemoveFromBlock(Messenger esql, String user)

This function asks the user to input a name, it then removes that name from that users block

list if it exists in the list.

public static void AddToContact(Messenger esql, String user)

This function asks the user for a contact name, if it exists it will add that contact to the user's

contact list. The user cannot add themselves to thei contact list.

public static void RemoveFromContact(Messenger esql, String user)

This function removes a user from their contact list.

public static void ManageContactList(Messenger esql, String user)

This is a menu function. Determines if the user wants to add or remove a contact.

public static void ManageBlockList(Messenger esql, String user)

This is a menu function. Determines whether the user wants to add or remove a blocked user.

public static void ListContacts(Messenger esql, String user)

This function returns a list of all contacts in the user's contact list. It then runs the menu

function ManageContactList.

public static void BlockContacts(Messenger esql, String user)

This function returns a list of all blocked contacts in the user's blocked list. It then runs the menu

function ManageBlockList.

public static void PostMessage(Messenger esql, String user, String chatID)

This function takes in a chatID and the user enters a message. That message is created and added to

the chatID specified.

public static void EditSingleMessage(Messenger esql, String user, String chatID)

Given a chatID, the user enters a new string that will replace the old string.

public static void EditMessages(Messenger esql, String user, String chatID, int offset)

This is a menu function. The user is presented with 10 messages of a chat. The user is then

able to decide if they want to see the next or previous 10 messages. They can also choose to edit

a single message.

public static void DeleteChat(Messenger esql, String chatID)

User deletes a chat.

public static void DeleteSingleMessage(Messenger esql, String user, String chatID)

User can completely delete a message of their choice.

public static void DeleteMessages(Messenger esql, String user, String chatID, int offset)

This is a menu function. The user is presented 10 messages of a chat. The user is then

able to decide if they want to see the next or previous 10 messages of a chat. If the user is

the oiriginal sender of the chat, they can delete the entire chat or delete individual messages.

if the user is not the owner they can delete their own messages.

public static void ViewChatMessages(Messenger esql, String user, String chatID, int offset)

This is a menu function. This function first returns a list of the first 10 messages in a chat.

It then asks if the user wants to see the next or previous 10 messages. User can then decide if they

want to post a new message, delete a message, or edit an existing message.

public static void ViewChat(Messenger esql, String user)

This function asks the user to input a chat, that chat will then be listen in the ViewChatMessages

function.

public static void AddMembersToChat(Messenger esql, String user, String chatID)

Adds members to a chat.

public static void RemoveMembersFromChat(Messenger esql, String user, String chatID)

Removes membners from a chat.

public static void EditChat(Messenger esql, String user)

This is a menu function. This function will ask the user if they want to add or remove

users from a chat.

public static void ListChats(Messenger esql, String user)

This is a menu function. It will first list out all chats that include the user. It will

then prompt the user to see if they want to view a chat or edit. via ViewChat and EditChat

public static void UpdateStatus(Messenger esql, String user)

This function will update the user's status.

**File: create\_tables.sql**

In the create\_Tables.sql file we inserted some “ON DELETE CASCADE” to the foreign keys so that deleting chats and users would be more easily done.

**File: create\_indexes.sql**

In the create\_indexes.sql file we decided to create an index for all primary keys. We did this so we could access all tables faster. We then created an index for the sender\_login on Message and init\_sender on Chat because those two attributes are also searched over frequently.

**Problems Encountered**

**Starting of postgres** **-** Sometimes after an initdb attempting to start the postmaster would not work. The only way we found to fix was to try initdb until the data folder was properly created.

**Comparing strings -**  When doing some error checking we used “string1 == string 2”. We believe instead of returning a 1 for true it was returning a 0 to show the two strings were equal, and would only return 1 if the first string was greater than the second.

When using esql.executeAndReturn the returned value would be its maximum length. So when we tried to reutnr the login "Norma" we would get the String "Norma " because SQL returned all 50 characters. To solve this we removed all white spaces from names after they were returned to us, since name sin the database could not have white space to begin with.

When having the messages display only 10 at a time we had problems with the indexing and showing the current page. This was fixed with some tinkering with the numbers.