Walter Matthews
COMP 2710
Alvin Lim
October 20, 2017

Lab 2 Design

Analysis

Use cases:

This program is designed to be a Social Network with a text-based interface. When the program starts, you must first log in. You are prompted for a user name. A new user will be created unless the user has logged in before, at that point you will log back in as that user. After this you are prompted with some options.

Broadcast allows the user to send multiple line of message to every single user that is in the social network.

Multicast allows the user to send a message to a particular group. Before writing the message you will choose what group you want to send to. The group must already exist.

Unicast allows the user to send a message directly to another user. The recipient must be a valid user in the social network.

Display Wall Page will display all messages that the user has sent.

Display Home Page will display all the messages that the user receives. Whether they are broadcast messages, messages to a group they are in or messages they directly received.

Create Group allows the user to create a new group that other users can join.

Join a Group allows the user to join an existing group in the social network,

Quit the Distributed TigerBook System – Program exits.

Classes

Menu

Handles menu choices. Creates menu for the main function to use.

User

Constructor for User. Maintains all information about the user.

Group

Constructor for Group. Maintains group members of each group.

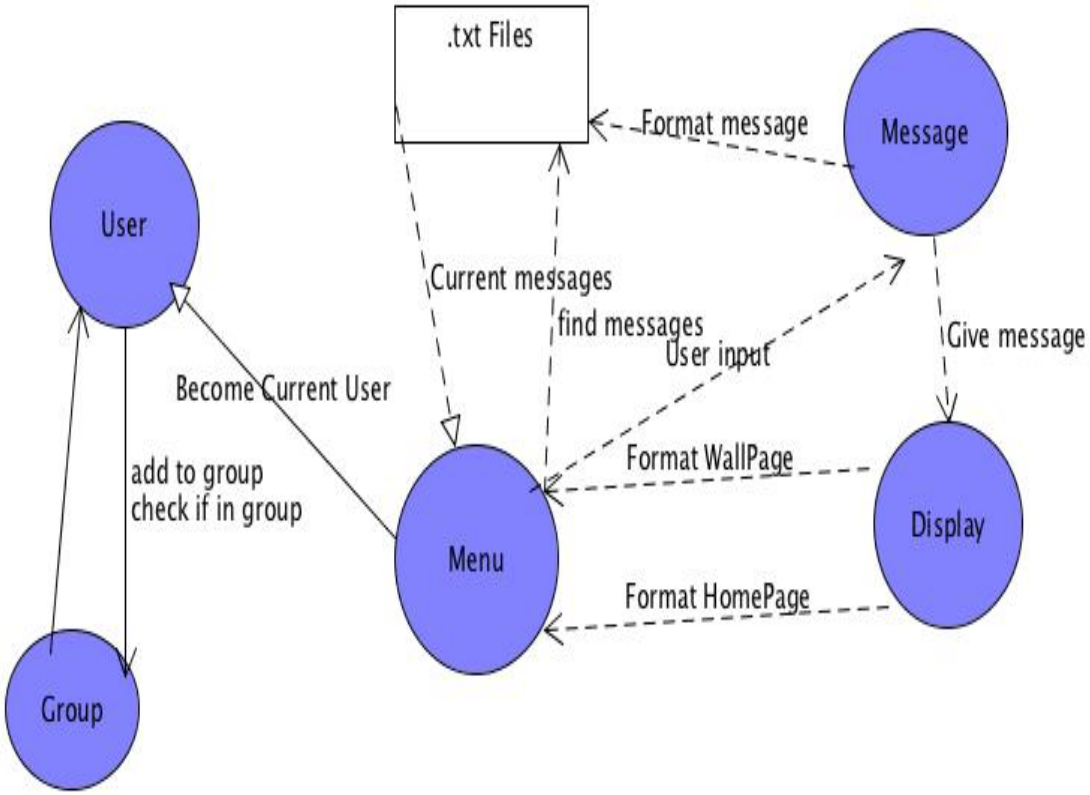
Display

Handles each unique header that are created for each menu choice. Also handles how wall pages and home pages are displayed.

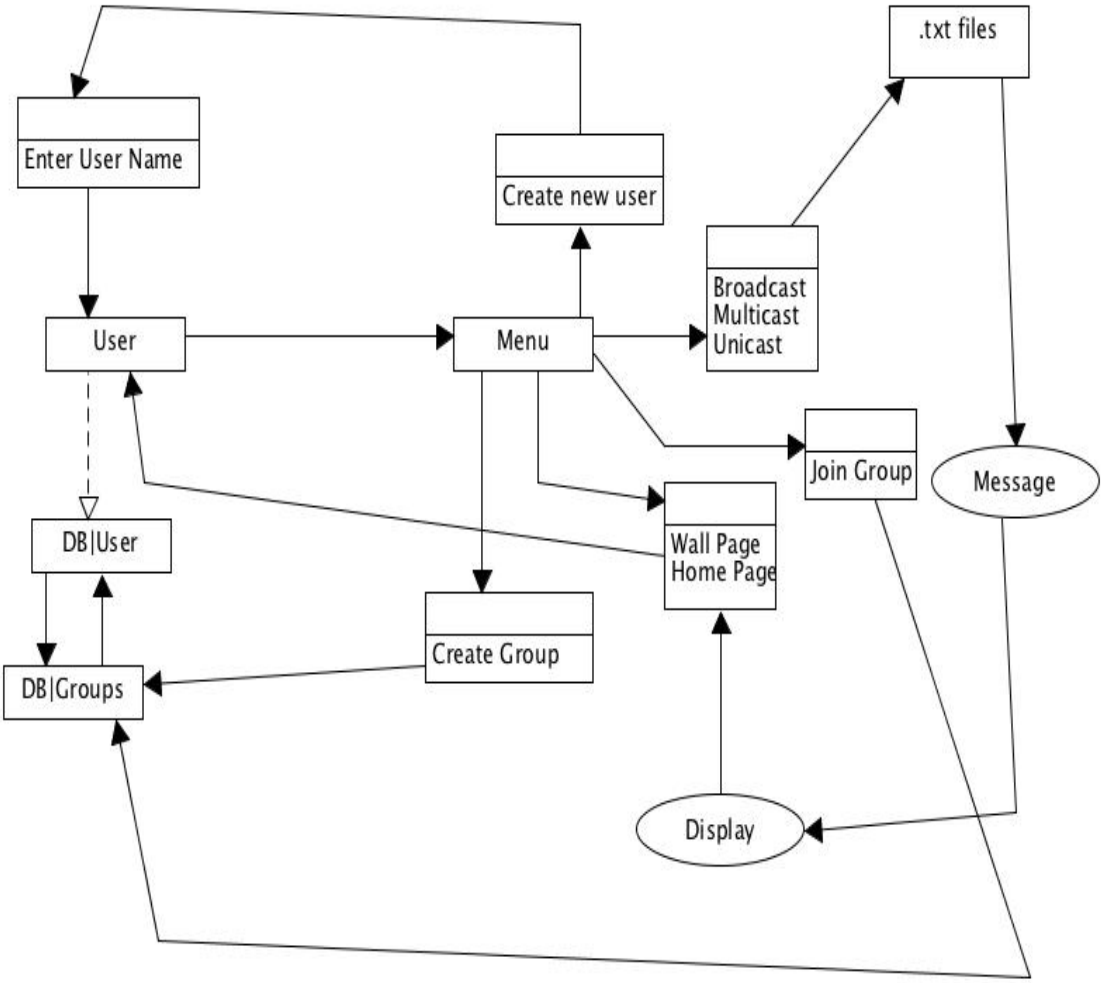
Message

Handles creating messages. Formats the messages correctly for .txt files and home/wall pages.

Class Diagram



Data Flow Diagram



Testing

System Testing

Enter Username:

- Enter new name -> Create new user
- Enter existing name -> Log back in as existing user
- Enter empty string -> re-enter

Broadcast:

- Enter message -> send to all groups
- Enter large message -> send to all groups
- Enter empty string -> try again

Multicast

- Enter non-existing group -> “Please enter an existing group”
- Enter existing group -> send to members of group
- Enter empty string -> try again

Unicast

- Enter non-existing user -> “Please choose an existing member”
- Enter existing member -> send to member
- Enter empty string -> try again

Wall

- Display Wall of user with sent messages
- Display wall of user who didn’t send messages

Home

- Display home of user with received messages
- Display home of user with no received messages

Create a group

- Create group with unused name -> Create new group
- Create group with used name -> Prompt to choose a different name.

Join a group

- Join existing group -> Add user to group
- Join nonexisting group -> Prompt to choose an existing group.

Unit Testing

User Class:

setName() -> produce error if string is empty or wrong data type

addToGroups() -> if name is already in group, don't add.

isInGroup() -> if group is found, return true.

If group isn't found, return false.

Group Class:

addMember() -> add User type to group.

If group doesn't exist produce error.

setName() -> set string argument to group name

getName() -> return group name string

Menu Class:

joinGroup() -> if group exists, add user to the vector

if it doesn't exist, produce an error.

createGroup() -> create new group object with name set from user input. if group exists, prompt message indicating this

startMenu() -> when called, program starts.

Display Class:

(unique)Header(): return header string.

displayWallPage() -> return string of all sent messages

displayHomePage() -> return string of all received messages.

Message Class:

addToFile(): add string object to text file. Formatted

insertMessage(): take in multiple lines of user input and store in a string

getNumberOfMessages(): return number of messages created.

getTime(): return “system time: