CS 307 - Software Engineering
Design Document

Team 14

Members:

Rayten Arnold Rex, Cole Baughn, Vishal Gill, Zeyu Pan,
Andrew Sytsma, Michael Young

Table of Contents

1. Purpose

In the modern age, people don't communicate with other people around them; they are too busy on their smartphones and engaged with people far away. There is a lack of small talk and general meet and greet in this generation. Our application will allow people to talk with other people in their vicinity in order to make small talk, discuss an idea, and just for fun.

This design document thoroughly describes our functional requirements.

1. Chat Rooms
    a. All users will be able to send messages to the Chat Room they are in.
    b. All users will be able to view the messages sent by other users that are in the same Chat Room.
        i. The sending user's name will be displayed before their message.
        ii. The time the message was received will be displayed across from the user name. (if enabled in settings)
    c. All users will be able to block other users by clicking on that user's name in the Chat room.
        i. A box will ask if the user wants to block the other selected user.
        ii. Blocked users will be added to a blocked list and will no longer be displayed in the chat room
2. Messages
    a. Each message contains the username of the person who sent the message, the message sent, the time received, and a unique ID for the user who sent the message.
    b. Messages can contain text, images, and/or emoticons.
3. Chat Room List
    a. All users will be able to view a list of all chat rooms that they can connect to.
        i. All users will have access to a "global" public chat room.
            1. This chat room is joined by all users by default.
            2. There is no main focus to this chat room, so users can talk about anything without being off-topic.
        ii. Clicking will take the user to that chat room.
            1. The user can just enter a publc chat room.
            2. The user will be prompted for a password to a private chat room.
        iii. Users will be able to view whether rooms require passwords or not.
        iv. Global chat is always listed first
    b. All users will be able to create their own chat rooms
        i. A box will popup and ask for a room name and (optional) password.
        ii. Enter button is located in name box since password is optional.
        iii. All users will be able to create public chat rooms.
            1. These chat rooms are created by users.
            2. The focus of these chat rooms could be specific or open.
                a. For example, a chat room about the Purdue football game.
                b. For example, a chat room for people in a local coffee shop.
            3. These chat rooms will appear in the chat room list.
        iv. All users will be able to create private chat rooms.

            1. These chat rooms will be password protected.
            2. These chat rooms will appear in the chat room list denoted with a “lock” symbol.
        v.

4. Homepage
    a. Users will be able to change their in-app username.
        i. Users can select any username they wish.
        ii. There can be more than one user with the same username.
        iii. Users with the same name will have different colors.
    b. Users will be able to view a list of chat rooms via a button.
    c. Users will be able to access logs of previous chats via a button.
    d. Users will be able to view the settings page via a button.
5. Chat Logging
    a. Users will be able to view a list of logs of previous chats.
        i. The list will contain the name of the chat room and a timestamp of when the log was created.
        ii. Users will be able order the view of the logs by the name of the chat room or the time the log was created.
    b. Users will be able to view logs of previous chats.
        i. Users will be able to select a log to view by “clicking” it.
        ii. The logs will display similar to the original chat experience.
            1. It will display usernames.
            2. It will display messages.
            3. It will display timestamps.
    c. Old logs will be deleted automatically to ensure that they do not take up too much space on the user’s phone.
        i. There will be a constant maximum number of logs
        ii. If the logs take up more than a certain amount of space, then old logs will be deleted whether or not the maximum number of logs has been reached.
6. Settings
    a. Unblock Users: will allow users to unblock users they previously blocked.
        i. The blocked users will be listed by the name they had at the time of being blocked.
        ii. If two users have the same name, they will also have the time they were blocked and the room they were blocked in listed.
    b. Color: will allow the user to change the color scheme of their UI to another predetermined color scheme.
        i. Changes background and foreground.
        ii. There will be preset Light, Medium, and Dark options for the themes.
    c. Font: will allow users to change the font their chat is displayed in to another preset font.
    d. Style: will allow users to change the styling of their UI
    e. Timestamp Disp: will allow users to enable/disable the displaying of timestamps in their Chat Room View
7. Block Users

a. Users will be able to block other users.
b. Users will be able to unblock other users on the settings page.

8. Security
   a. The messages sent by users will be encrypted to ensure their messages cannot be viewed by a third party.
   b. Private chat room passwords will be encrypted to ensure that the passwords cannot be stolen easily.
   c. Phone information used to provide a unique ID for each user will be hashed when sent, so the information can no longer be retrieved, only compared.
9. Network
   a. Users will connect to each other using a peer to peer connection with the wireless radio. No internet connection is required.
   b. Chat rooms will be based on spacial distance.
      i. Two chat rooms with the same information, name and password, can exist as long as there are no two wireless radios in range of each other (approximately 105 feet).
      ii. Two chat rooms will merge into a single chat room if the information is identical and two wireless radios are within range.
      iii. To achieve both goals, messages will be dropped if a user receives a message that belongs to a room the user is not part of.

2. Design Outline

Our project will use the existing peer-to-peer (P2P) framework to create a mesh networking infrastructure, which is a decentralized communications model in which each party has the same communication capabilities and either party can initiate a communication session. Unlike the client-server model, in which the client makes a service request and the server fulfills that request, the mesh networking model allows each device to function both as a client and as a server.

Using iOS 7 multipeer connectivity for iPhone and Wifi Direct for Android devices, the range for peer-to-peer wifi communication is approximately 105 feet, but because of the mesh architecture behind Local Chat, the actual range of the network could be much larger. The application GUI will be the user's main interaction with our system. The user will open up the app and choose a chatroom to join. Once the user connects to a node (another device in the network) he/she will be able to chat with anyone else also connected to that node directly or indirectly.

Figure 2.1

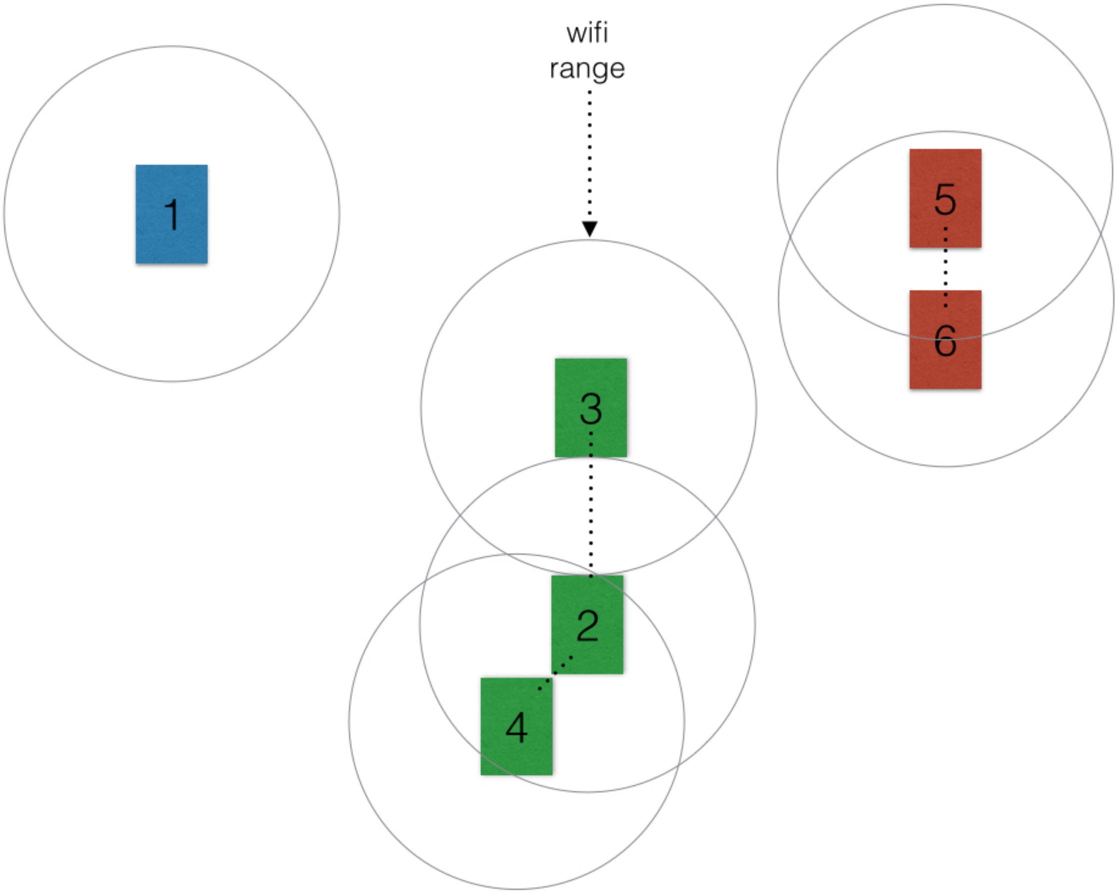


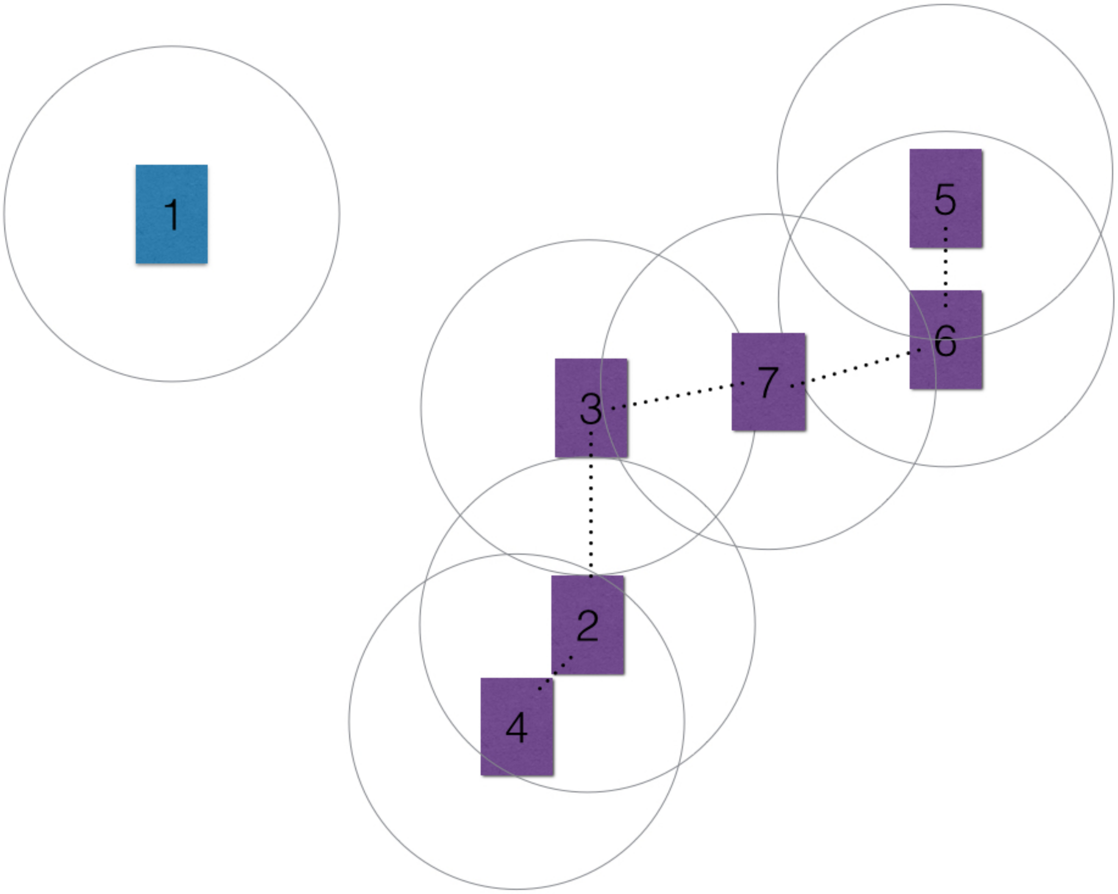
Figure 2.1 above shows 6 unique devices running the Local Chat application. There are three distinct groups based on the range of the devices. Blue group has no other devices except device #1. Green group has 3 connected devices. Finally, red group has 2.

For simplicity's sake, let us assume all devices are in the public chat room, global chat. Device 2 connects devices 3 and 4 together indirectly. Even though device 3 is out of the range of device 4's wifi signal. And in this way device's 2, 3, and 4 can all communicate amongst each other.

Now if a seventh device appears that is in range of device 3 and 6 the global chat will merge. Device's 5 and 6 will see four new devices added to their chat. Devices 2, 3, and 4 will see three new devices in their chat and device 7 will see five new devices.

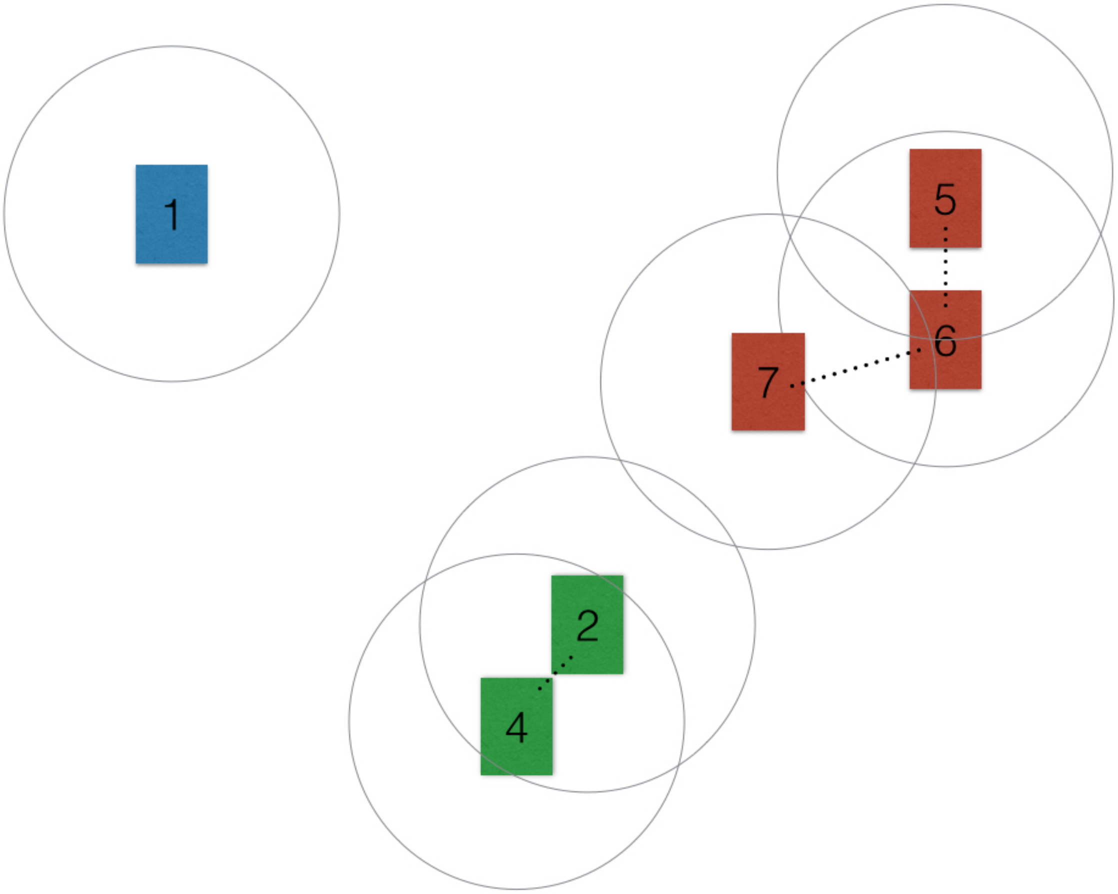
To take it a step further, if device 3's user closes the app, there is no possible way for devices 2-7 to connect to each other. Devices 2 and 4 and devices 5-7 will have their own respective groups.

Figure 2.2



In figure 2.2 above, once device 7 appears it becomes a bridge and allows the green group and red group to merge into one single big group.

Figure 2.3



In figure 2.3 above, with the removal of device 3, devices 2 and 4 are now part of another chat room, while 5, 6, and 7 have their own chat room.

3. Design Issues

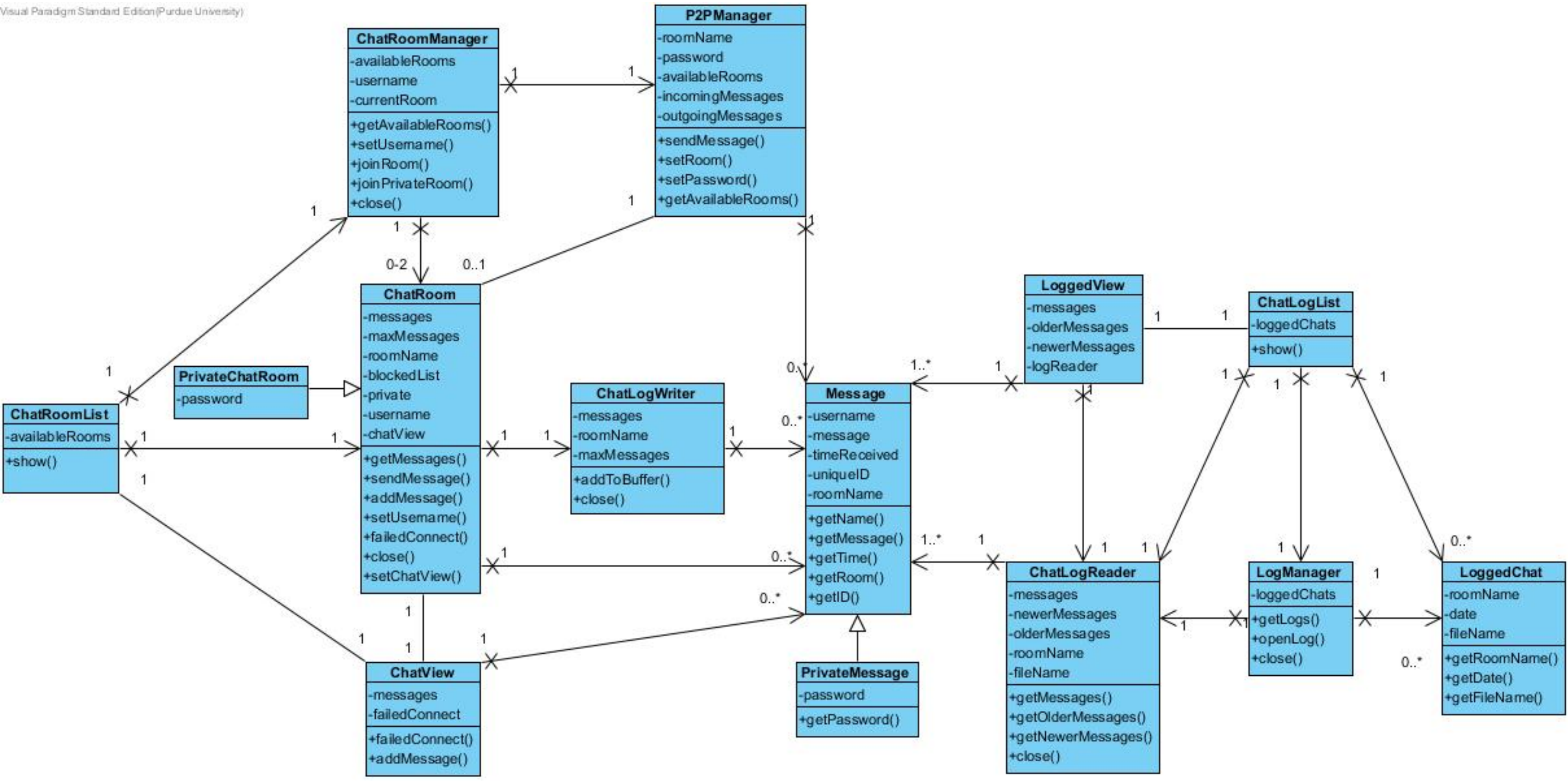
- **Design Issue #1:** iOS GUI Programming Language
  - Option #1: Swift
  - Option #2: Objective C
  - Decision: Option #1 was chosen because Swift is the native language for iOS development.
- **Design Issue #2:** Settings Saving
  - Option #1: Save at every setting change
  - Option #2: Save at application close
  - Decision: Option #2 was chosen because settings do not need to be saved every time an option is changed, just updated. By saving only on application close, less time is spent saving configurations.
- **Design Issue #3:** Background Programming Language
  - Option #1: Java for Android and C++ for iOS
  - Option #2: C++ for both platforms
  - Decision: Option #2 was chosen because we would not have to write a different version of the code for each platform.
- **Design Issue #4:** How Often Logs Should Be Saved
  - Option #1: Log one line of chat after 500 lines.
  - Option #2: Log chat using a buffer.
  - Decision: Option #2 was chosen because logging information line by line is inefficient. Logging individual lines at a time is slow due to storage access operations being generally slower compared to memory operations. By using a buffer, we use the memory and write a larger amount of data to storage versus several different storage operations.
- **Design Issue #5:** Networking
  - Option #1: Only use the wifi radio to communicate between users.
  - Option #2: Using mesh networking to connect users.
  - Decision: Option #2 was chosen because it allows users to communicate more easily. Also, it remedies the problem of having someone communicate with a user that other users cannot see due to the wifi radio's distance.
- **Design Issue #6:** Username Color
  - Option #1: Allow users to choose their color to be displayed in other user's chats
  - Option #2: Randomly assign username colors in chat.
  - Decision: Option #2 was chosen because all users could choose the same color defeating the purpose of having colored names in the first place. (they are for differentiating users)
- **Design Issue #7:** User ID
  - Option #1: Use google accounts to identify users.

    - Option #2: Use phone information to identify users.
    - Decision: Option #2 was chosen because it allows for cross-platform communication down the road since iOS does not use google accounts for its users.
- **Design Issue #8:** Log Display
    - Option #1: Display logs as the raw text.
    - Option #2: Display logs as formatted text.
    - Decision: Option #2 was chosen because it allows the messages in the logs to be laid out in a manner that is easier for the users to read.
- **Design Issue #9:** Blocking Users
    - Option #1: Block users based off of their unique user ID.
    - Option #2: Block users based off of their username.
    - Decision: Option #1 was chosen because issues could arise if there were multiple users with the same username. The blocker would unintentionally block all users with that specific username even if they only wanted to block a single user.
- **Design Issue #10:** Timestamps
    - Option #1: Always have timestamps shown for messages in chat.
    - Option #2: All users to enable or disable timestamps through settings.
    - Decision: Option #2 was chosen because showing or not showing timestamps is not a hard feature to implement, but it allows users to further customize their chat experience.
- **Design Issue #11:** In-Chat Message Storage
    - Option #1: Store all messages in the background during chat.
    - Option #2: Store all messages in the user interface during chat.
    - Decision: Option #2 was chosen because, although storing messages in the background allows more separation of the parts, it would slow down the user interface unnecessarily to ask for the messages every time the display is changed.
- **Design Issue #12:** Logged Chat Message Storage
    - Option #1: Store the entire log in the background.
    - Option #2: Store the entire log in the user interface.
    - Option #3: Store current messages being viewed and buffered chunks in the user interface and ask the background for further chunks when needed.
    - Decision: Option #3 was chosen because it prevents reading from the log file excessively and only keeps relevant messages in RAM when moving through the chat.
- **Design issue #13:** Parallel Background
    - Option #1: Have the user interface and the background run at startup.
    - Option #2: Have the user interface create the background when needed.

- Decision: Option #2 was chosen because it makes it easier for the background to be linked to and converse with the user interface.

4. Design Details

Figure 4.1



Class diagram for the application.

Description of classes and models:

- ChatRoomList
  - ChatRoomList displays the list of available rooms.
  - ChatRoomList creates the ChatRoomManager to start network communication and get the list of available rooms.
  - ChatRoomList gets a newly created ChatRoom and gives it to ChatView.
  - ChatRoomList creates ChatView when the user joins a chat room, and ChatRoomList gives ChatView its ChatRoom.
- ChatView
  - ChatView manages the chat room page that the user sees.
  - ChatView calls ChatRoom's send message when the user wants to send a message.
  - ChatView gets message details out of its messages, using the Message class, when it displays the messages to the user.

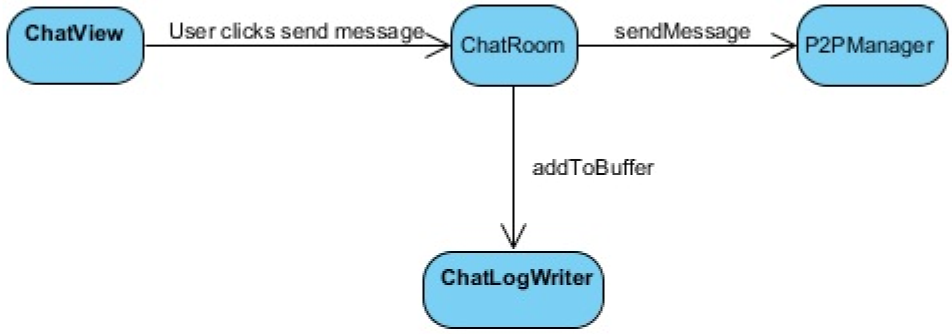
    - ChatView gets its ChatRoom from ChatRoomList.
- ChatRoom
    - ChatRoom handles the communication between the user interface, the background, and the networking.
    - ChatRoom calls P2PManager's sendMessage to send a message.
    - P2PManager calls ChatRoom's addMessage when it receives a message for the chat room, and P2PManager calls ChatRoom's failedConnect to indicate a connection failure.
    - ChatRoom is managed by ChatRoomManager.
        - ChatRoomManager keeps track of the active ChatRoom.
        - ChatRoomManager joins and leaves chat rooms, creating the ChatRoom and calling ChatRoom's close.
        - ChatRoomManager connects ChatRoom to P2PManager.
    - ChatRoom adds messages to the ChatLogWriter using addToBuffer, and ChatRoom tells ChatLogWriter to write all the messages into the log with close.
    - ChatRoom sends messages received from ChatView through sendMessage, and ChatRoom adds messages to ChatView's queue and tells ChatView about connection errors through addMessage and failedConnect.
    - ChatRoom creates, sends, and stores Messages.
    - ChatRoom is sent to ChatView via ChatRoomList.
    - ChatRoom is the super class of PrivateChatRoom.
- PrivateChatRoom
    - PrivateChatRoom saves the room's password and uses a separate encryption/decryption of messages.
    - PrivateChatRoom is a sub class of ChatRoom.
- ChatRoomManager
    - ChatRoomManager creates the P2PManager and manages the active ChatRoom.
    - ChatRoomList uses ChatRoomManager to list available chat rooms with getAvailablChatRooms.
    - ChatRoomManager creates the P2PManager and links it to ChatRoom when it's created.
    - ChatRoomManager creates each ChatRoom, linking them to the P2PManager, and closes them when the user changes rooms or leaves.
- P2PManager
    - P2PManager manages the Peer-to-Peer network and sends and receives messages for the user.
    - ChatRoomManager creates P2PManager and links to each ChatRoom.
    - P2PManager calls ChatRoom's failedConnect to indicate a connection failure.
    - P2PManager receives a message from ChatRoom and sends it.

  - When P2PManager receives a message, it sends the message to ChatRoom.\
  - P2PManager gets message details from Message.
- ChatLogWriter
  - ChatLogWriter writes message to logs.
  - ChatRoom calls ChatLogWriters's addToBuffer every time a message is sent or received.
  - ChatLogWriter gets message details from Message.
- Message
  - Message contains all information needed for a message including: username, message, time, chat room, and user ID.
  - P2PManager gets message details form Message.
  - LoggedView gets message details from Message.
  - ChatLogReader gets message details from Message and creates Messages.
  - ChatView gets message details from Message.
  - ChatRoom gets message details from Message and creates Messages.
  - ChatLogWriter gets message details from Message.
  - Message is a super class of PrivateMessage, which contains the same details, but it also contains a password and differing encryption/decryption.
- PrivateMessage
  - PrivateMessage saves the room's password and uses a separate encryption/decryption of messages.
  - PrivateMessage is a sub class of Message.
- ChatLogReader
  - ChatLogReader opens the log file and turns its contents into messages for LoggedView to display.
  - LoggedView gets the messages from ChatLogReader through getMessages, getOlderMessages, and getNewerMessages.
  - ChatLogList takes a newly created ChatLogReader and gives it to LoggedView for access.
  - LogManager creates ChatLogReader.
  - ChatLogReader creates Messages from the text of the log file.
- LoggedView
  - LoggedView displays messages from a chat log.
  - LoggedView gets all the messages in the log from ChatLogReader using getMessages, getOlderMessages, and getNewerMessages.
  - LoggedView gets the ChatLogReader from ChatLogList.
  - LoggedView gets the details to display from Message.
- LoggedChat

- LoggedChat is a class made to help with sorting logs based off of time or room name.
- ChatLogList contains a list of LoggedChats to display for the user.
- LogManager creates the list of LoggedChats from the files in the log directory.

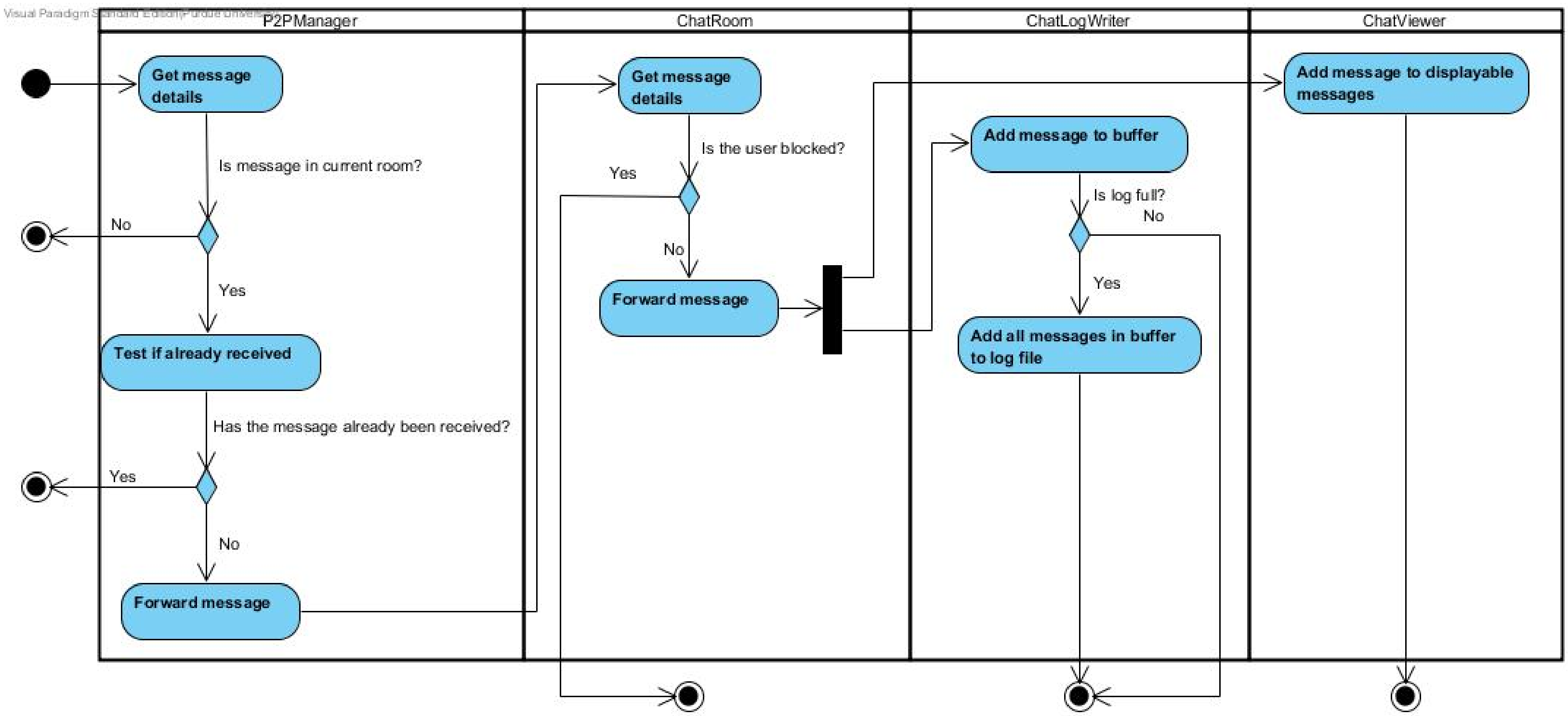
- ChatLogList
    - ChatLogList contains a list of LoggedChats that it displays in a sorted order defined by the user.
    - ChatLogList gets the list of LoggedChats from LogManager using getLogs.
    - ChatLogList contains a list of LoggedChats, which it sorts using LoggedChat's getName and getDate methods.
    - ChatLogList gets a newly created LogReader from LogManager to give to LoggedView.
    - ChatLogList gives LoggedView the LogReader it obtained from LogManager.
- LogManager
    - LogManager creates a list of LoggedChats from the files in the log directory and shares the list or creates a LogReader for LoggedView.
    - LogManager provides a list of LoggedChats and creates a LogReader for ChatLogList.
    - LogManager creates a list of LoggedChats from the files in the log directory.
    - LogManager creates a Log Reader for ChatLogList.

Figure 4.2



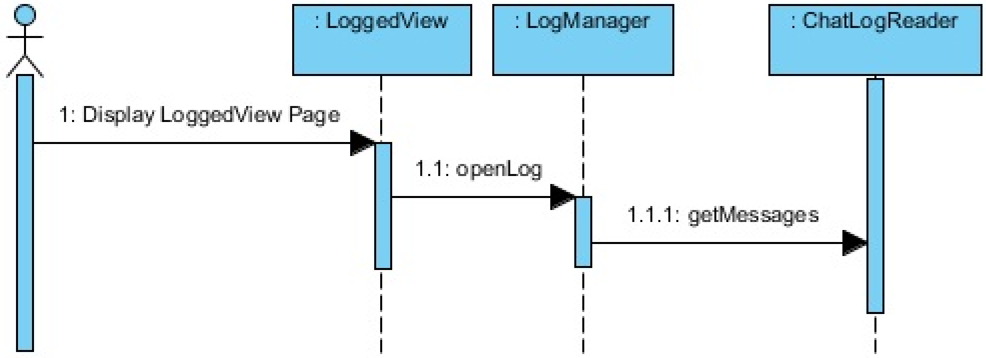
Activity diagram of when a user sends a message.

Figure 4.3



Activity diagram of when a user receives a message.

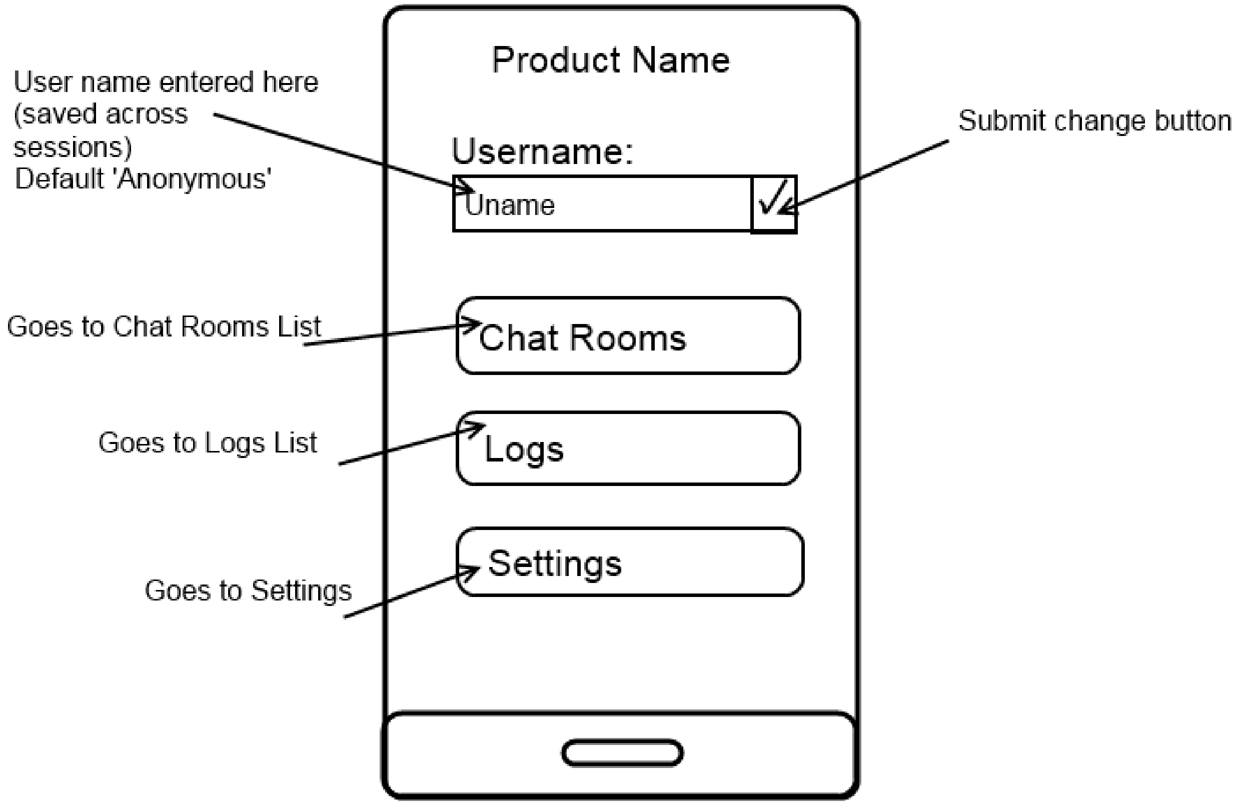
Figure 4.4



Activity diagram of when a user reads a log.

UI Mockups

Home Page



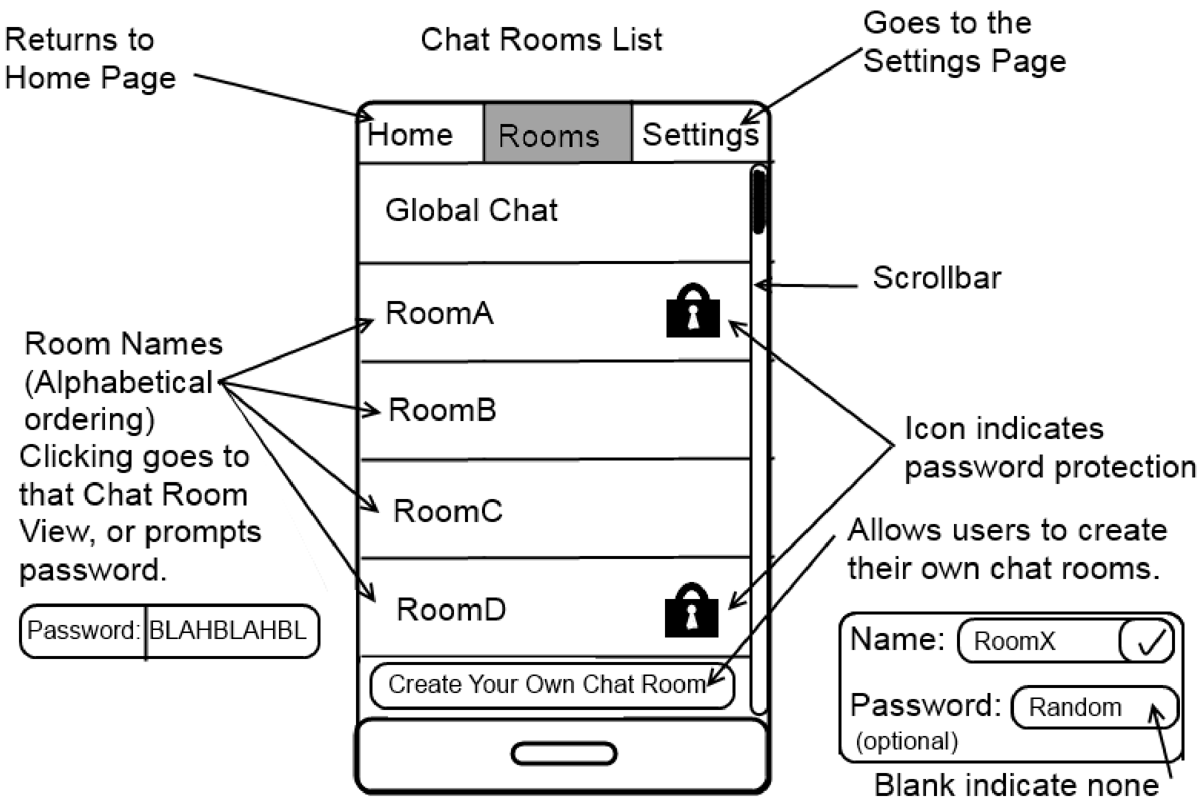
User name entered here (saved across sessions)
Default 'Anonymous'

Submit change button

Goes to Chat Rooms List

Goes to Logs List

Goes to Settings



Chat Rooms List

Returns to Home Page

Goes to the Settings Page

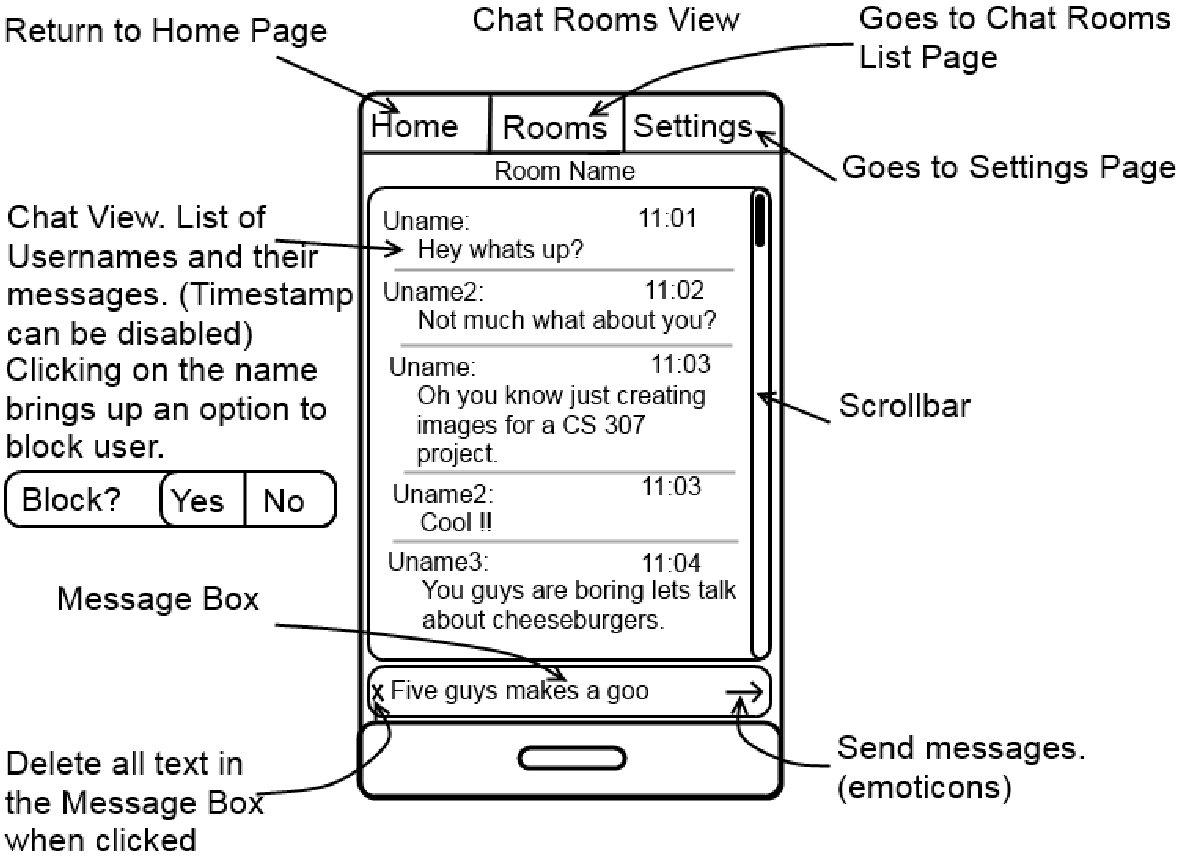
Scrollbar

Room Names (Alphabetical ordering) Clicking goes to that Chat Room View, or prompts password.

Icon indicates password protection

Allows users to create their own chat rooms.

Blank indicate none



Chat Rooms View

Return to Home Page

Goes to Chat Rooms List Page

Goes to Settings Page

Chat View. List of Usernames and their messages. (Timestamp can be disabled) Clicking on the name brings up an option to block user.

Scrollbar

Message Box

Delete all text in the Message Box when clicked

Send messages. (emoticons)

Uname: 11:01
Hey whats up?

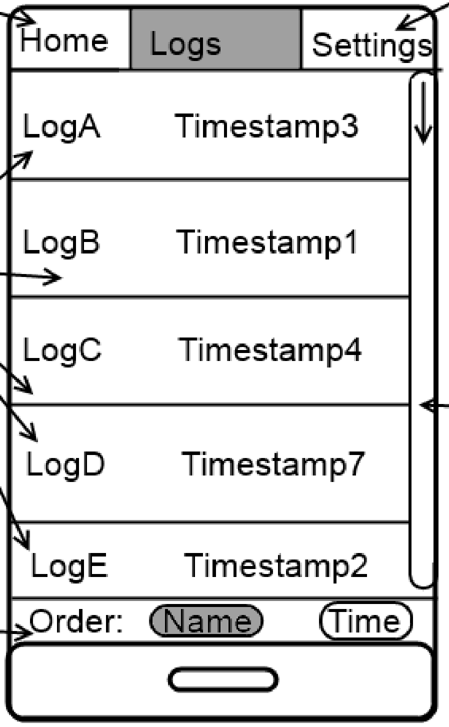
Uname2: 11:02
Not much what about you?

Uname: 11:03
Oh you know just creating images for a CS 307 project.

Uname2: 11:03
Cool !!

Uname3: 11:04
You guys are boring lets talk about cheeseburgers.

Logs List



Return to the Home Page

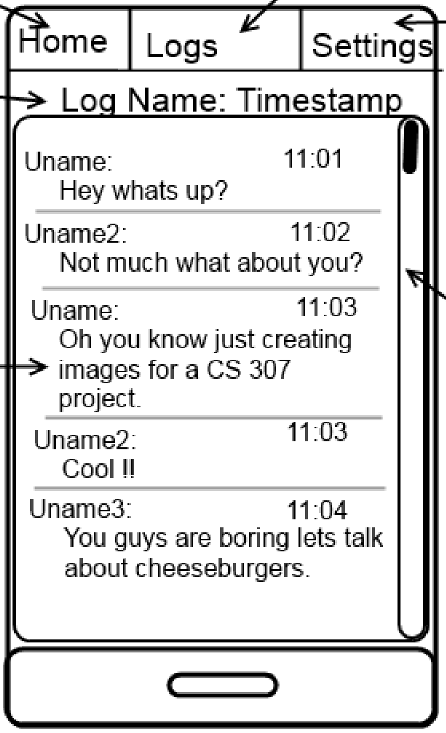
Goes to the Settings Page

Chat room names. Clicking goes to that Log View

Scrollbar

Orders the list based on selection

Log View



Uname: 11:01
Hey whats up?

Uname2: 11:02
Not much what about you?

Uname: 11:03
Oh you know just creating images for a CS 307 project.

Uname2: 11:03
Cool !!

Uname3: 11:04
You guys are boring lets talk about cheeseburgers.

Return to the Home Page

Goes to Logs List.

Goes to the Settings Page

Title (Log followed by timestamp)

Scrollbar

Log View. Same format as Chat View. (Timestamp can be disabled)

Settings Page

Return to the Home Page

Returns to prior page

Goes to the Unblock List

Settings. Can be changed through drop boxes.

Scrollbar

<u>All Settings</u>

-Blocked List

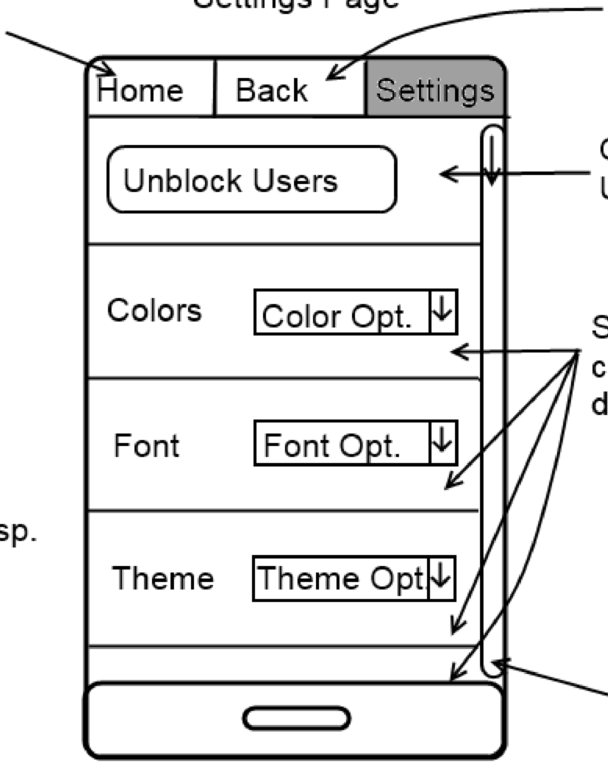
-Colors

-Font

-Theme

-Style

-Timestamp Disp.



Unblock List

Return to the Home Page

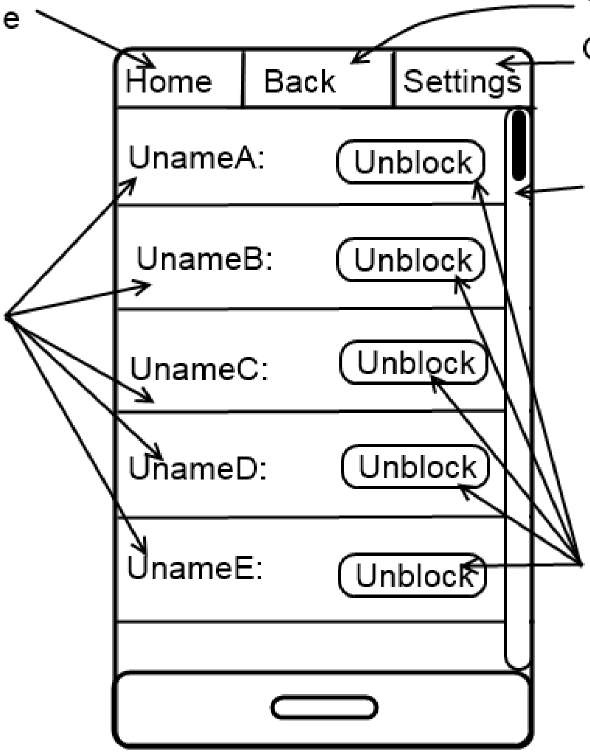
Goes to previous page

Go to Settings Page

Scrollbar

Blocked Users (alphabetical)

Unblock button (unblocks user)


Home
Back
Settings
UnameA:
Unblock
UnameB:
Unblock
UnameC:
Unblock
UnameD:
Unblock
UnameE:
Unblock