CS4962 Final Project

IRCme – IRC Client for iOS

Andrew Kuhnhausen, Colton Myers

Design Document

Spring 2011

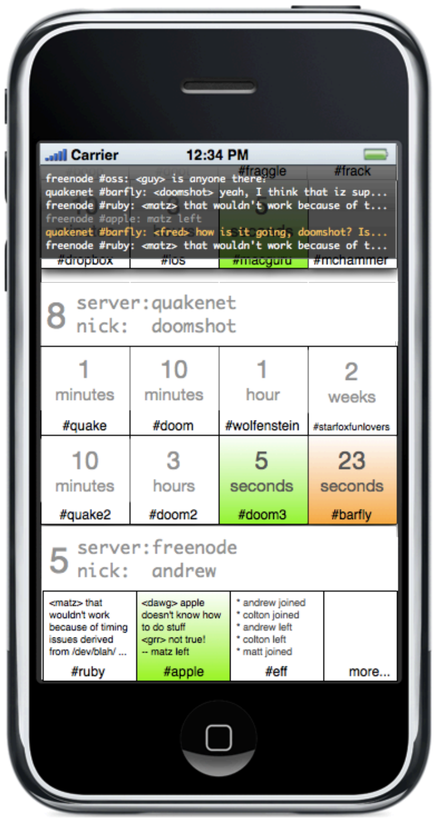
TEAM MEMBERS

- Colton Myers – u0502549
- Andrew Kuhnhausen – u0275126

ABSTRACT

Our application, IRCme, is meant to be a fully featured IRC client. There are several IRC clients on the App Store, however, there seem to be only 2 worthy of purchase, Colloquy ($1.99) and Limechat ($4.99). Both clients have a similar UI, mostly using tables. It is our aim to provide the user with a rich graphical UI that gives the user a more complete experience, while maintaining the same feature-set as those two clients, such as SSL, channel saving, auto-connect, and notifications/sounds. We also plan on making our app localization-compatible, by not using string literals, but rather using the language plists for strings.

Screens



```
freenode #oss: <guy> is anyone there?
quakenet #barfly: <doomshot> yeah, I think that iz sup...
freenode #ruby: <matz> that wouldn't work because of t...
freenode #apple: matz left
quakenet #barfly: <fred> how is it going, doomshot? Is...
freenode #ruby: <matz> that wouldn't work because of t...
```

8 server:quakenet
nick: doomshot

1 minutes #quake	10 minutes #doom	1 hour #wolfenstein	2 weeks #starfoxfunlovers
10 minutes #quake2	3 hours #doom2	5 seconds #doom3	23 seconds #barfly

5 server:freenode
nick: andrew

<matz> that wouldn't work because of timing issues derived from /dev/blah/ ... #ruby	<dawg> apple doesn't know how to do stuff <grr> not true! -- matz left #apple	* andrew joined * colton joined * andrew left * colton left * matt joined #eff	more...

Main Screen

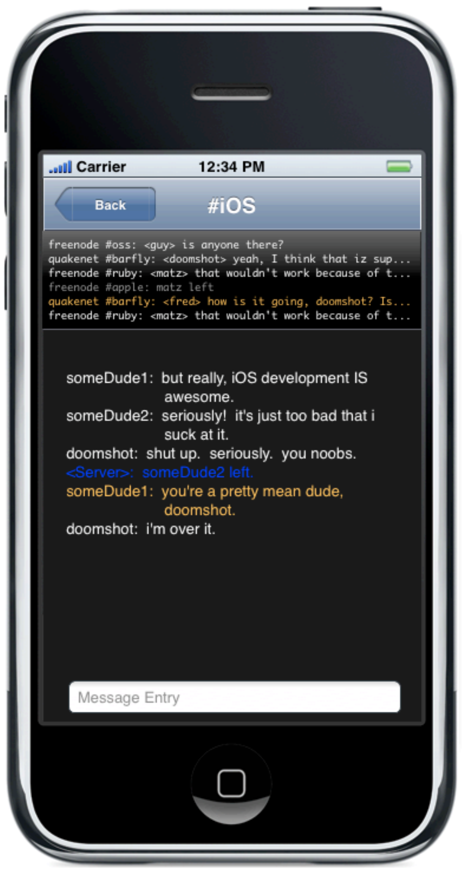
The main screen consists of the following major components:

- Notification view, which is persistent and always visible
- Vertically-scrollable servers view, which is a custom table group view, which represents both the server information, and contains the subviews representing the channels within the server
- Views for each channel within a given server, which will be arranged in a horizontally-scrollable grid, which will have dimensions 2 x (Number of channels / 2).

The notification view will show the latest messages which have come in via any channel/server, and will show the channel/server from which the information comes, the username of the messenger, and will be highlighted in a user-defined color if a message "mentions" the user. The same notification view will also be shown when a user is messaging within a specific channel (this will be shown in the next image).

The server view contains different parts. The left-most part of the view will contain the number of channels to which the user is subscribed within that server. The main area of the view will contain the server's name (defined in server preferences) and the user's nickname for that server. In the current example, there is actually no way to enter preferences from the main view – this is a design issue we will handle once we have the framework in place, and will probably be a small button in the bottom-right-hand corner of the screen. Server preferences will be accessible via a long-press on the given server, or possibly the "info" icon (a blue circle with an "i") on each server view. Adding new servers will be handled by selecting an extra row at the bottom of the vertical scroll labeled so users obviously know it is the server adding mechanism.

Each channel view will be user-configurable to show a variety of information. The name of the channel will be noted at the bottom of each view. In the main part of each view, the user will define whether the contents of the channel (updated in real time) will be shown, an image for the channel, or the channel name in larger, decorative text, will be shown. It can also be configured to show the time since the last message in that channel. (A couple of these options are shown in the image) If a user is mentioned in a channel, that channel will be highlighted in a user-configurable color (orange, in the example). It will also be highlighted if a new message of any kind comes in, in a user-configurable color (green, in the example).



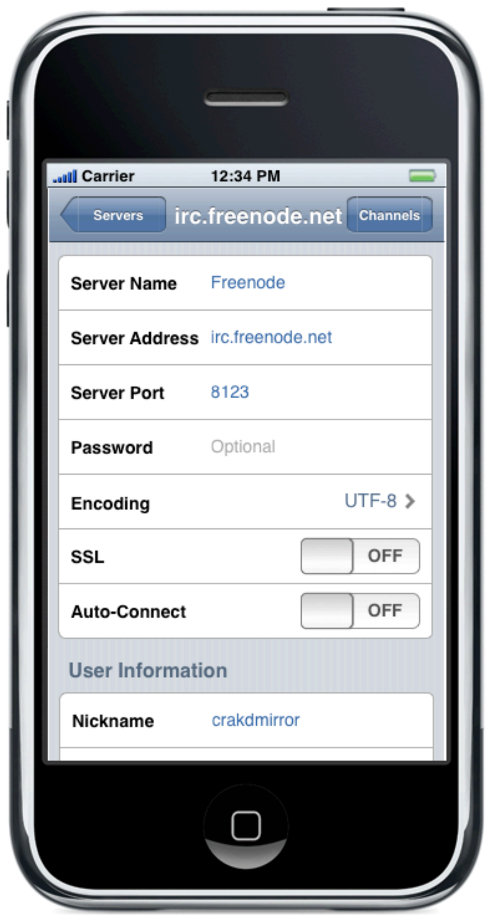
```
freenode #oss: <guy> is anyone there?
quakenet #barfly: <doomshot> yeah, I think that iz sup...
freenode #ruby: <matz> that wouldn't work because of t...
freenode #apple: matz left
quakenet #barfly: <fred> how is it going, doomshot? Is...
freenode #ruby: <matz> that wouldn't work because of t...
```

someDude1: but really, iOS development IS awesome.
someDude2: seriously! it's just too bad that i suck at it.
doomshot: shut up. seriously. you noobs.
<Server>: someDude2 left.
someDude1: you're a pretty mean dude, doomshot.
doomshot: i'm over it.

Chat Screen

This screen represents the interface which will allow a user to communicate on a channel. It will be reachable by single-tapping a given channel view from the main screen. It consists of the message-display view for the given channel, a message entry box which will summon the onscreen keyboard, and allow a user to type messages, the notification view mentioned previously, and a navigation view allowing return to the main server view.

Again, messages which "mention" the user will be highlighted in the user-configurable color mentioned previously (orange, in the example).



Server Name	Freenode
Server Address	irc.freenode.net
Server Port	8123
Password	Optional
Encoding	UTF-8
SSL	OFF
Auto-Connect	OFF

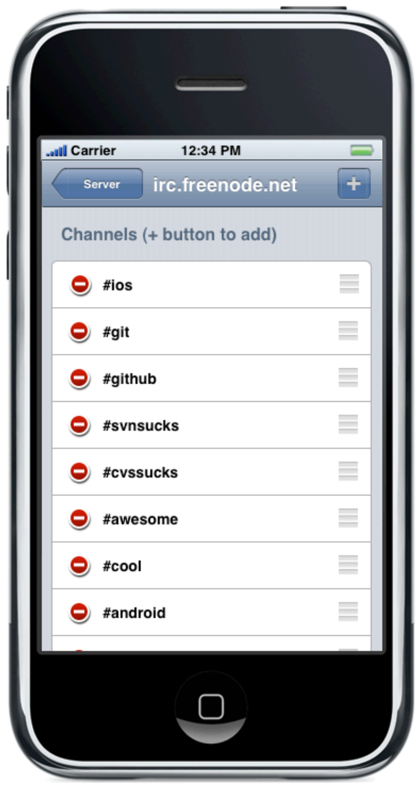
User Information

Nickname	crakdmirror

Server Preferences Screen

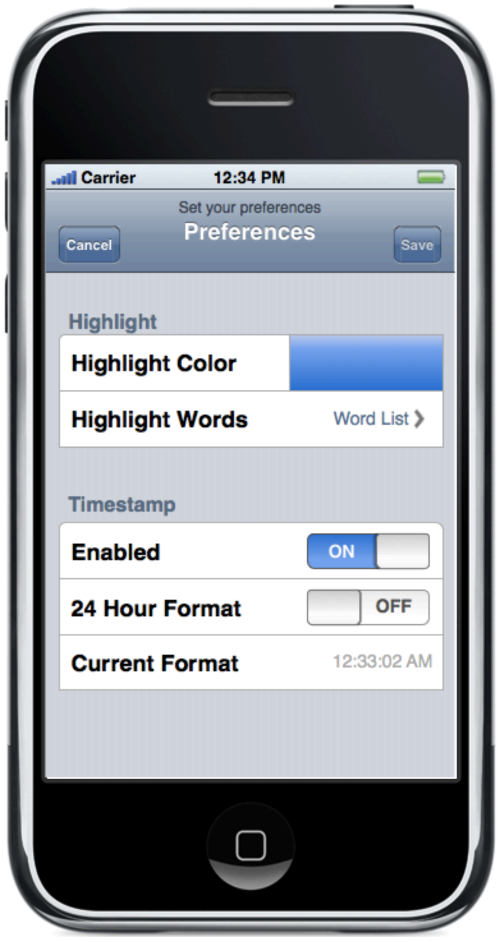
This screen will be used both for editing an existing server, as well as adding a new server. It provides functionality for defining the basic information for a server (URI, port, password (optional), etc), as well as additional options, such as auto-connect, SSL support, the user's nickname, and possibly some display options for the server. (Number of rows of channels, etc – the customization options are potentially endless)

The navigation bar at the top of the view allows access to the channel list, as well as the ability to return to the main servers view.



Channel List Screen

This screen allows the user to view/modify the channel list for the given server. A navigation bar at the top provides both functionality to return to the server preferences screen, as well as add a new channel to the server. The remainder of the view is an editable tableview, which lists all currently-subscribed channels, and allows channels to be rearranged and deleted.



Set your preferences
Preferences

Highlight

Highlight Color	🟦
Highlight Words	Word List

Timestamp

Enabled	ON
24 Hour Format	OFF
Current Format	12:33:02 AM

Global Preferences Screen

The global preferences screen allows easy access to a series of preferences which apply to the application as a whole. Some of the available preferences are shown in the example above, such as 24 vs 12 hour time format, highlight colors, etc. Additional options may include settings related to server communication and the like. (Update frequency, etc) Some of this will depend on how we implement the network access parts of the project. The view provides the ability to save or cancel changes to the user's settings.

ADDITIONAL FEATURES

The following are a list of some additional features which we can implement at a later date, or if we have time during the final project period:

- Controls to access neighboring channels within a server while in the messaging view
- Long-tap support for server and channel views to delete/reorder
- Additional languages (specifically Brazilian Portuguese, as Andrew speaks that language – but other languages as we can)
- iPad support
    - Obviously a completely different platform, and a lot of modification would have to take place to make it a proper, native iPad app, but it would go a long way in making our app more flexible and desirable
- Ability to follow links within the app without leaving the app for the browser app (We've heard that there is a way to add browser functionality right within our app, and it's supposedly quite straightforward)
- Custom keyboard with IRC-specific shortcuts
- Implement IRSSI IRC shortcut commands
- Voice support (app speaks messages as they come in)
- Ability to use the notification system (push notifications via iOS)
- Custom "away" messages
- …Additional functionality as we think of it, or users suggest it (assuming we actually market the app)