Peter Moerman, Daniel Easterling, and Ryan O'Gorman

CPSC 330

Dr. K. Anewalt

26 March 2013

Writing Assignment 3

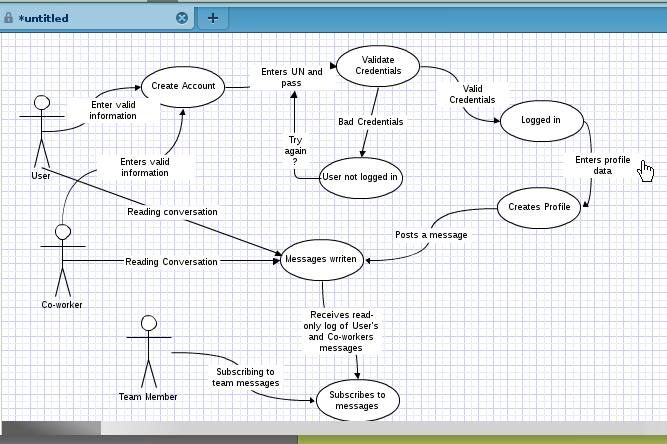
This documents purpose is to outline the functionality of the Bitter networking system. Our intended audience is the stake-holders of the company. Basically, anyone can be a stakeholder. This system is intended to offer a non-productive alternative to those using the Intranet of various computer labs, companies, coffeehouses, conventions, large LAN parties, or wherever an Intranet may be. Bitter allows users to chat asynchronously with other users of Bitter within the confines of the Intranet on which Bitter is installed. Users may set a profile and link to an e-mail if they wish, but this is not required. Messages posted by a user may be viewed by any other user that chooses to look at that first user's messages, unless they are marked private, in which case only subscribers of the user may view the messages.

To aid in communication, Bitter allows users to mark their messages with topics, so that one may browse messages by topic instead of by individual user. This way, even as users sit quietly, staring at the glowing portals that transport them to realms of pure logic and kittens, users may poll the ever-changing labgeist about important topics of the hour, such as why people are wrong to prefer Next Gen to Original, or who just farted.

From here onward there will first be an overall description of who Bitter is for and a more detailed description of how the product functions using a case write-up.

The client of Bitter is the system administrator on the up-and-up. The typical user of this system has watched the entirety of telnet Star Wars on a Friday night and recalls a glorious time of networks that were anti-social, rather than the tittering extroverts they are today.

Bitter will enable the user to create an account, and login if their credentials are correct. Given the user logged in correctly, create a profile, and the user can post messages to another user, and read messages in a conversation with another user. An external user to the conversation can opt to “subscribe” to the users conversation or message history; this means the external user receives a read-only log of the writer's messages.



Our system specification is outlined as follows:

System specification:

The Bitter system design is geared toward its goal - a beautiful melding

of design and functionality. The intention of the Bitter design team is to

exercise our craftsmanship by building upon the following rock-solid foundation

of a class list:

\*\*\*

UserInterface:

This class details the working of the Bitter user interface. By separating this functionality from the rest of the systems, the Bitter engineers can later implement a graphical user interface if it is deemed so necessary.

Action getAction()

void display(DisplayPiece dp)

\*\*\*

FileIO:

FileIO handles the persistence aspects of Bitter. It is typically triggered every five minutes or so, or by administrator fiat.

void save()

\*\*\*

Driver:

This class acts as the driver for the whole system. It handles interactions between the various input/output classes and the rest of the systems.

void main(String[] args)

void programLoop()

void act(Action)

\*\*\*

AccountTable:

AccountTable keeps track of usernames, passwords (hashed, of course!), and account status.

bool verify(String username, String password)

bool getStatus(String username)

\*\*\*

User:

User keeps track of settings for users, including profiles. It is mostly a data-tracking class, and its functionality consists largely of accessors and mutators.

\*\*\*

Profile:

Profile is another data-holding class that stores information about a specific user, such as e-mail, real name, etc.

\*\*\*

Query:

Query handles interactions between the driver and the message storing classes. Note that "filter" below can be a standard string, in which case it is considered a username, or may be affixed with a # (hashtag),

which indicates that one wishes to filter by topic.

MessageQueue getMessages(String filter)

\*\*\*

Time Stamp:

Keeps track of a specific time, typically used to see when a message was posted. Consists mostly of accessors and mutators

Message:

Creates a message that consists of the user who posted it, the actual text of the message, and time at which it was posted. Consists mostly of accusers and mutators.

\*\*\*

MessageHash:

MessageHash is a wrapper for the Standard Library HashMap class. Two main instances are planned: TopicHash and UserHash, which will associate messages with topics and users, respectively.

\*\*\*

MessageQueue:

MessageQueue is a wrapper for the Standard Library Queue interface. It holds

messages.

\*\*\*

The system requirements for Bitter are not heavy in the least. The most basic of networked systems, where each user has access to a keyboard and command line, should be able to run Bitter. The actual requirements are as follows:

1) A Java Runtime Environment (JRE), version 7 or later

2) A keyboard or keyboard alternative

3) Access to a command line, and permissions to run Bitter

4) An efficient and stable keyboard-chair interface