Deliverable II, Software Requirements Specification

Project: *Ventilate* (Peer-to-Peer Chat Program)

Group Members: Jacob Pebworth, Austin Hoppe, Elexa Evans,

Christopher Hines, Ryan Porterfield

Group Name: 4444-Chat_Group-5

Date: 07 October 2015

Contents

Introduction

Purpose

This document specifies all functional and non-functional requirements for Ventilate version 1.0.0. Ventilate is a peer-to-peer chat application similar to IRC but with the addition of private rooms. Ventilate communicates directly with other peers on the network, but uses a centralized server simply to store a list of peers which are currently online so that peers can find each other. This document defines functionality for both the client and the server application.

Document Conventions

There are some key terms used in this document that need to be defined.

Client – A client is a user who is running Ventilate on their local machine to talk to other users. Typically client is used to denote a user who is performing an action.

Server – The server is the central authority of Ventilate which keeps track of clients who are currently connected to the network so that new clients can be added to the network.

Peer – A peer is a client connected to the network who is not performing an action. In this document we will refer to a client getting information from, or sending information to peers. Peers are also clients on the network, but not the client completing the action being described.

DHT – A DHT is a distributed hash table which is a method to quickly store and retrieve information in a peer-to-peer network without every peer having to store all the information. See https://en.wikipedia.org/wiki/Distributed_hash_table for more information.

Intended Audience and Reading Suggestions

Project Scope

Ventilate is a peer-to-peer chat application. Ventilate uses chat rooms similar to IRC, but adds additional functionality. Ventilate uses distributed hash tables to store user accounts including email addresses, usernames, and passwords as well as chat history. Keeping information off a central server reduces the risk of getting hacked and losing all customer information. It also reduces the load on a central server during peak usage hours, instead distributing the load among all the online clients. The chat history not being in a central place also means that even if served a court order, the owners of Ventilate can't turn over chat history, so users privacy is nominally protected. This doesn't stop users from joining public rooms and reading back through all the stored history, up to a week, which Ventilate keeps as a feature for its users.

References

https://en.wikipedia.org/wiki/Distributed_hash_table

Description

Product Perspective

Ventilate is a new standalone chat application. This document defines the specification for the first production release of Ventilate.

Product Features

Ventilate allows users to create and delete accounts which have unique user names and are password protected. Ventilate allows users to change or reset passwords for their accounts. Ventilate also allows users to join chat rooms, create chat rooms, and leave chat rooms.

User Classes

Ventilate will potentially be used by many different types of users. Some classes of users who might use Ventilate are less technically skilled users who just want a simple, easy to use, well designed chat client that they can communicate with their friends on. Other groups who might use Ventilate are users who want the privacy benefits of a peer-to-peer decentralized client. The last class of people who might use Ventilate are technically skilled users who want the performance benefits of a chat application which is well designed and can continue to perform well even under heavy stress.

Operating Environment

Ventilate will run on a wide range of hardware and operating systems. Ventilate will run on modern version of Microsoft Windows, Apple Mac OS X and Linux-based systems, with Windows 7 and later, OS X 11.10 and later, and the Linux Kernel versions 4.0 and up being development targets. Our program is designed to be run on 32-bit and 64-bit x86 processors, and this is the architecture on which it is developed. Ventilate should not conflict with any major software that users may have installed, though some additional dependencies will likely be required should users with source code access wish to build it themselves.

Design and Implementation Constraints

The biggest constraint in designing and implementing Ventilate is time. Ventilate is a senior level group project for a semester long class. This means the developers have only 16 weeks to write specifications and implement functionality for Ventilate. The other major constraint is programming language. In order to save time and maximize the effectiveness of each group member the developers are restricted to a language that every member already knows, and the only language each developer has in common is C++.

User Documentation

Ventilate will ship with a simple user manual that describes how to do basic functions so that even users who aren't comfortable with technology can reliably use Ventilate.

Assumptions and Dependencies

Ventilate is built on the Qt 5 framework, which is an open source, full featured, cross platform framework for designing applications. The Qt libraries simplify network communication and provide a graphics library for designing easy to use Graphical User Interfaces.

System Features

The program Ventilate should have the following functions:

Create an account

Description and Priority

Users of Ventilate should be able to create persistent accounts with unique user names, protected by passwords.

The ability to create an account is **high** priority. All of Ventilate's features are based around users having unique accounts.

Stimulus/Response

On the main page of Ventilate users will have the option to log-in or create an account. Creating an account will be triggered by clicking the "Create new account" button on the main page of the application.

Functional Requirements

In order for users to create an account in Ventilate:

- Both the server and client applications need an account data-type (class) which stores the user's username, password, email address, and optionally cell phone number and service provider.
- Ventilate needs to store account information securely in a DHS (Distributed Hash Table) shared among the peers (clients).
- Clients need to be able to securely inform peers when a new account is created.
- When an account is created peers need to be able to add the account to the DHS.

Delete an account

Description and Priority

Users of Ventilate should be able to permanently delete an account so that the unique user name is no longer associated with them.

The ability to delete an account is **medium** priority. Deleting an account is not critical for any of Venilate's functions, but is a feature to protect the privacy of our users.

Stimulus/Response

Any user who has created an account on Ventilate should be able to delete their account at any time. This event will be triggered when a user selects the "Delete account" option from the menu bar.

Functional Requirements

The following requirements are in addition to the requirements for creating an account.

- Clients need to be able to securely inform peers when an account is deleted.
- When an account is deleted peers need to be able to remove the account from the DHS.

Login to an existing account

Description and Priority

Users of Ventilate should be able to login to an existing account if they have already created one.

The ability for users to login to accounts they have created is **high** priority.

Stimulus/Response

On the main page of Ventilate users will have the option to log-in or create an account. Logging in to an account will be triggered by clicking the “Login” button on the main page of the application after providing a username and password in the appropriate text fields, also found on the main page.

Functional Requirements

- The clients need to be able to retrieve account information from other peers to validate logins.
- Clients need a secure way to inform other peers when a new user has logged in.

Logout of an account

Description and Priority

Users of Ventilate should be able to logout of accounts they have logged in to so that they can login to other accounts, or login to the same account on a different device.

The ability for users to logout of accounts is **high** priority.

Stimulus/Response

Users of Ventilate who are logged in to the application should be able to logout at any time. The logout event is triggered by selecting the “Logout” option from the menu bar.

Functional Requirements

- Clients need to be able to securely inform peers that a user has logged out.
- When a user logs out, the network needs to rebalance itself without the user.

Change account password

Description and Priority

Users of Ventilate should be able to change the password associated with their account to increase security, if they know their current password.

The ability for users to change their account password is **high** priority.

Stimulus/Response

At any time, a user who is logged in to an account should be able to change their account password by providing their old password for verification and the new password. The change password event will be triggered by selecting the “Change password” option from the menu bar.

Functional Requirements

The following requirements are in addition to being able to validate account information as in the Login function, and store account information as in the Create Account function.

- Modify account information stored in the DHT

Reset account password

Description and Priority

Users of Ventilate should be able to securely reset their password in the event that they have forgotten their current password.

The ability for users to reset their account password is **high** priority.

Stimulus/Response

On the initial page of Ventilate, there will be a button labeled “Forgot password” next to the “Login” button. Clicking on the “Forgot password” button will trigger a reset password event.

Functional Requirements

The following requirements are in addition to all requirements for changing a password.

- Generate a temporary password for an account
- Send an email to the email address associated with a user’s account with the temporary password

Create a public chat room

Description and Priority

Users of Ventilate should be able to create a public chat room than any other user of Ventilate can view and join at any time.

The ability for users to create a public chat room is **high** priority.

Stimulus/Response

A user who is logged in to an account should be able to create a new public chat room at any time. The Create Public Room event is triggered by selecting the “Create public room” option under the “Chat rooms” tab in the menu.

Functional Requirements

- Clients need to be able to create new chat rooms
- Clients need to keep a list of public chat rooms
- Clients need to inform peers when a new public room is created
- Clients need to check if a public chat room with the same name exists before creating a new public chat room

Create a private chat room

Description and Priority

Users of Ventilate should be able to create private chat rooms that can only be accessed by people who have permission of the chat owner.

The ability for users to create a private chat room is **high** priority.

Stimulus/Response

A user who is logged in to an account should be able to create a new private chat room at any time. The Create Private Room event is triggered by selecting the “Create private room” option under the “Chat rooms” tab in the menu.

Functional Requirements

- Clients need to be able to create new chat rooms

Delete a public chat room

Description and Priority

The owner of a public chat room who no longer wishes for the chat room they created to exist should be able to delete and remove all users from the public chat room.A user who is logged in, and who has created a public chat room should be able to delete any chat room they have created, but only chat rooms they have created at any time.

The ability for users to delete public chat rooms is **low** priority.

Stimulus/Response

Deleting a chat room is triggered by selecting the “Delete chat room” option from the “Chat rooms” tab in the menu.

Functional Requirements

- Check if a user has permission to delete a chat room
- Clients need to be able to delete chat rooms
- Clients need to be able to notify peers that a public chat room has been deleted

Delete a private chat room

Description and Priority

The owner of a private chat room who no longer wishes for the chat room they created to exist should be able to delete and remove all users from the public chat room.A user who is logged in, and who has created a private chat room should be able to delete any chat room they have created, but only chat rooms they have created at any time.

The ability for users to delete private chat rooms is **low** priority.

Stimulus/Response

Deleting a chat room is triggered by selecting the “Delete chat room” option from the “Chat rooms” tab in the menu.

Functional Requirements

- Check if a user has permission to delete a chat room
- Clients need to be able to delete chat rooms
- Clients need to be able to notify only peers in the private chat room that it was deleted

Join a public chat room

Description and Priority

Every user of Ventilate should have full, unrestricted access to all public chat rooms, which they can join at will.

The ability to join public chat rooms is **high** priority.

Stimulus/Response

Joining a chat room is triggered by selecting the “Join chat room” option from the “Chat rooms” tab in the menu, and supplying Ventilate with a chat room name through a pop-up dialogue.

Functional Requirements

- Clients need to be able to get a list of all public chat rooms
- Clients need to inform peers when a user joins a public chat room

Add other users to private chat rooms

Description and Priority

Owners of a private chat room should be able to add users of their choice to the private chat room so groups of people can have private communication.

The ability to add users to private chat rooms is **high** priority.

Stimulus/Response

Adding a user to a private chat room is triggered by selecting the “Add to chat room” option from the “Chat rooms” tab in the menu, and supplying Ventilate with a username through a pop-up dialogue.

Functional Requirements

- Check if a user has permission to add people to a private chat room
- Inform a client that the user logged in to that client has been added to a private chat room
- Clients need to keep a private list of all private chats a user is in

Leave a chat room

Description and Priority

Any user of Ventilate should be able to leave both public and private chat rooms if they no longer want to be in the room for any reason.

The ability to leave chat rooms is **high** priority.

Stimulus/Response

Leaving a chat room is triggered by selecting the “Leave chat room” option from the “Chat rooms” tab in the menu.

Functional Requirements

- Client needs to stop listening for messages in chat rooms a user has left
- Client needs to inform peers when a user leaves a chat room

Send a message to a chat room

Description and Priority

Any user who joins a public room or is added to a private room should be able to post messages in the room that every other member of the room can see.

Stimulus/Response

A user sends a message to a chat room by typing a message into a text box below the chat view pane and pressing the send button.

Functional Requirements

- Clients need to propagate new messages through the network whether or not the user needs to see the message
- Clients need to display a message if the user is in the room that the message came from
- Clients need to ignore or not display the message if the user is not in the room the message came from

View chat history in a chat room

Description and Priority

Any user who joins a public room or was added to a private room should be able to scroll back and view at least a week of history, or until the chat was created, whichever is more recent.

Stimulus/Response

When a user scrolls further back in a chat room than they have history for, the application should fetch chat history up to the last week and display it to the user.

Functional Requirements

- Store chat history in a DHT
- Get history for a chat room from the DHT

External Interface Requirements

User Interfaces

The user interface for Ventilate will be composed of several unique windows which will alternate being active depending on the state of the application.

Software Interfaces

Ventilate will be built on the Qt5 framework.

Communications Interfaces

Ventilate will talk to the server and other peers over the TCP/IP protocol.

Other Nonfunctional Requirements

Security Requirements

User account information is stored in a DHT, meaning that each client on the network has a portion of the total user database. In order to protect user information, the information stored in the DHT needs to be encrypted and passwords need to be hashed.

Users of Ventilate also have an expectation of privacy, and every message has to be propagated to every client on the network in case that client is listening for that message. As such messages need to be transmitted securely and with encryption so users can't snoop and listen for messages not intended for them.

Software Quality Requirements

Users of Ventilate come from many different levels of tech literacy which makes it important that Ventilate is a high quality product. People who are not comfortable with technology need to be able to reliably use Ventilate, and Ventilate needs to uphold standards of usability so that these users feel comfortable with the application.

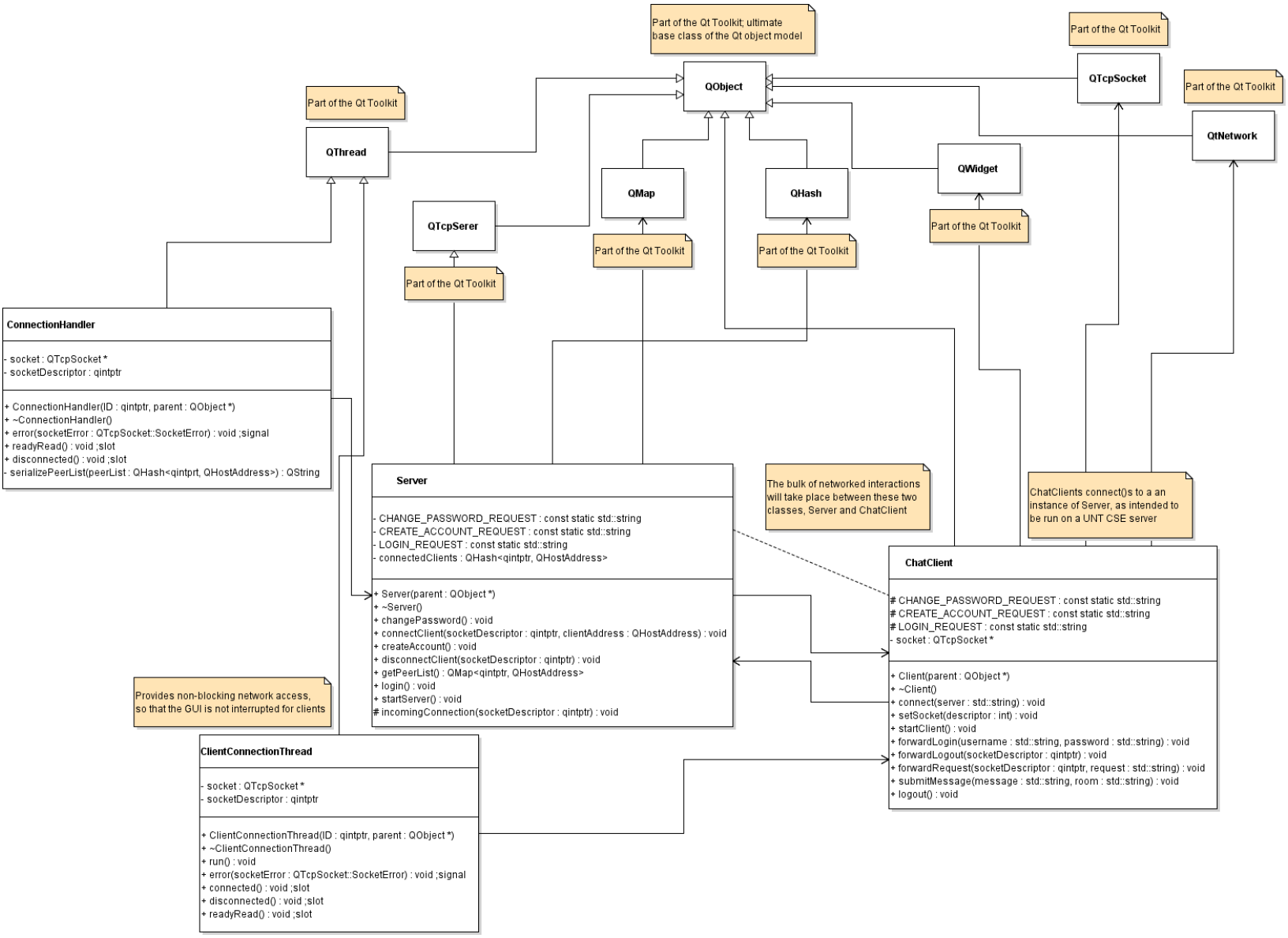
Architecture

The architecture of Ventilate is Client-Server, where each client acts as both a client and a server. Implementing both the client and server in a single application solves the problem of shared data models and inefficient data exchange.

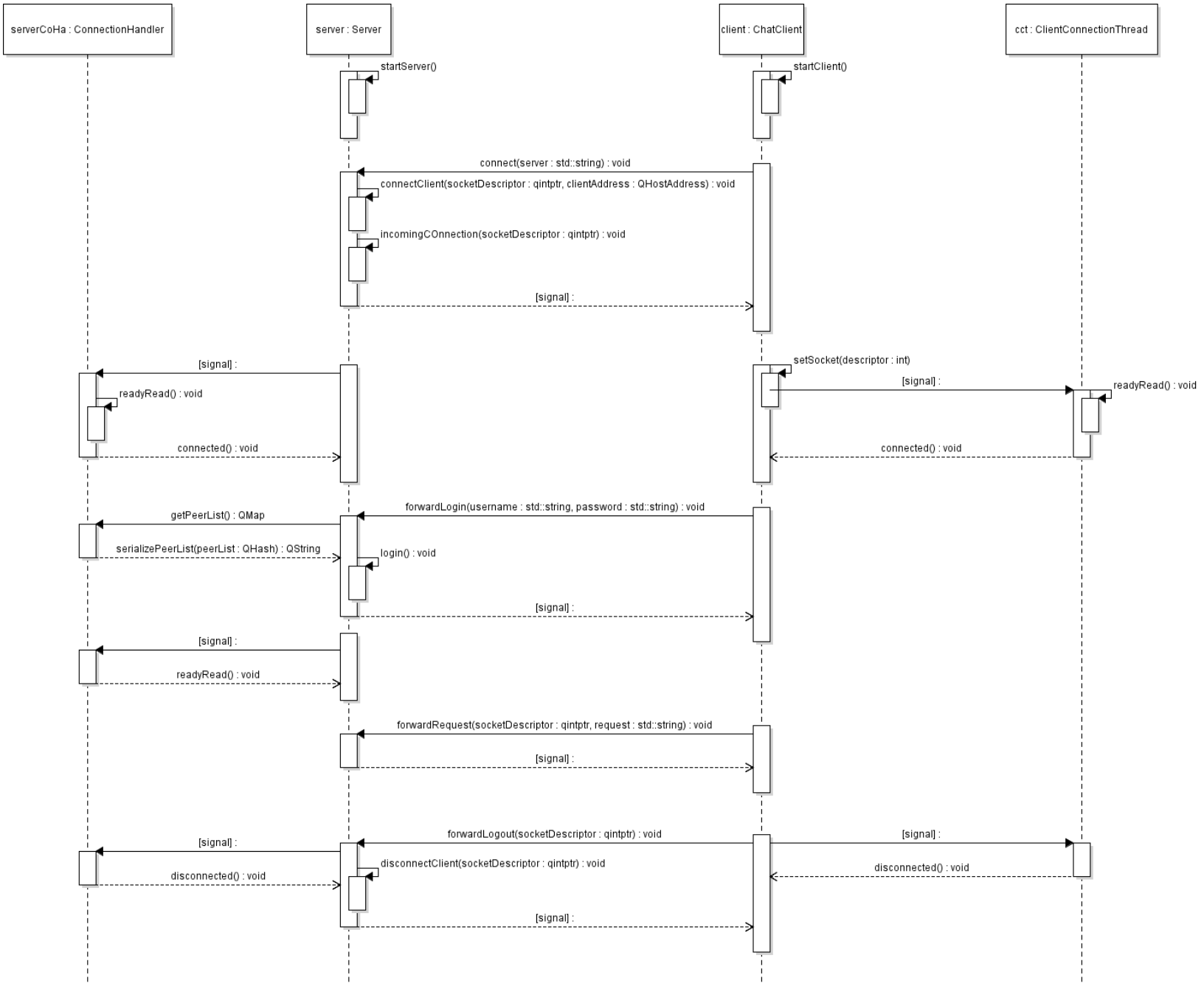
UML Diagrams

(Continued on separate pages)

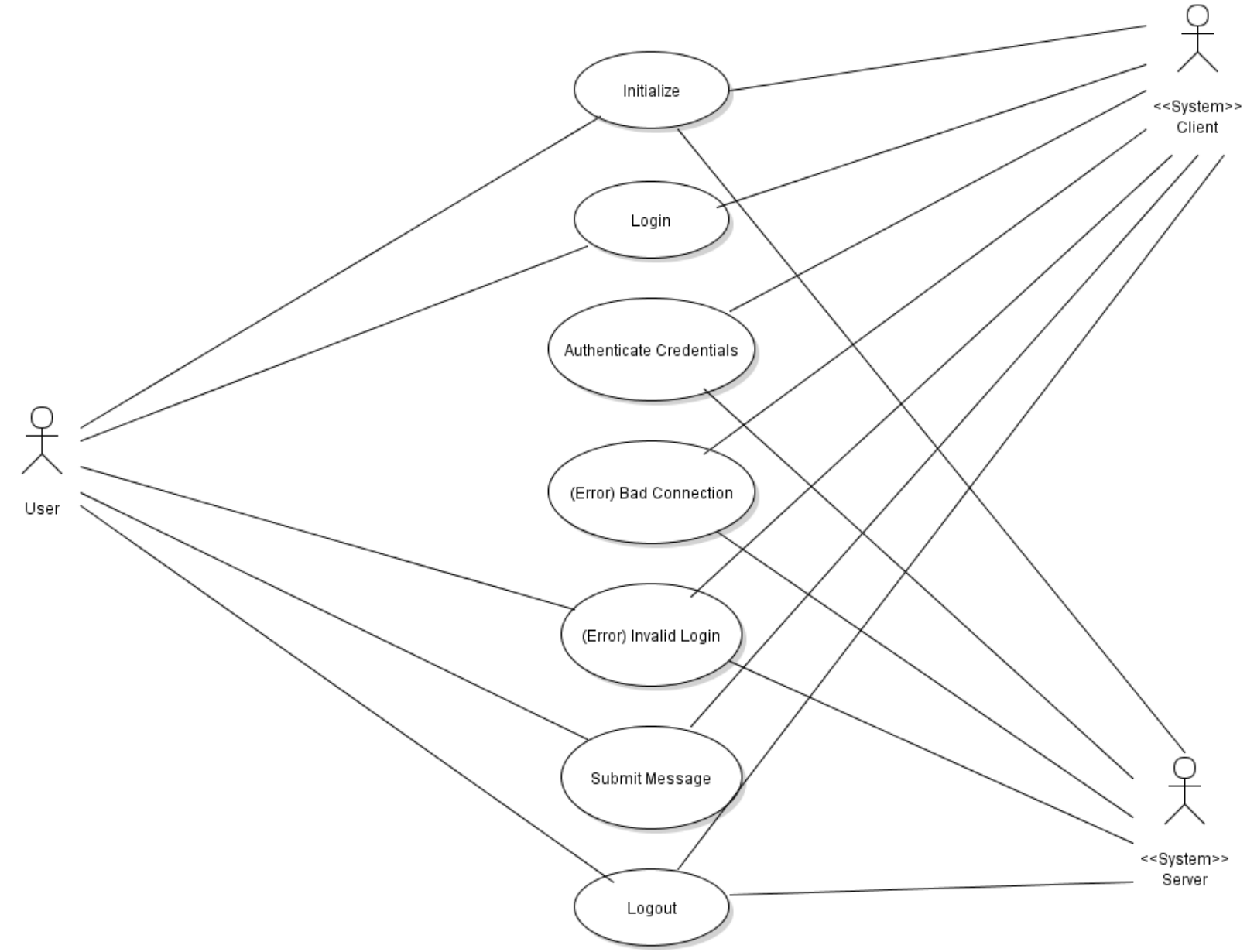
Class Diagram



Sequence Diagram



Use Case Diagram



Test Plan

Ventilate is written on the Qt5 framework which provides its own unit testing framework. Each data type in Ventilate, and every function for which it is practical will have its own unit test. This will allow the developers to ensure that every function and data type performs as expected. In addition to unit tests the developers will write a QA test and individually test the program for bugs or unexpected functionality. The combination of these two tests will make QA and bug testing easy, and help the developers to quickly identify, diagnose, and fix any problems which may arise.

Updated Risk Management

In the initial risk management plan for Ventilate we prepared to lose a member of our team. We have a team member who hasn't contributed to the project at all, but who also hasn't dropped the class. We have had to use the safety measures we put in place of cutting back on project scope in order to compensate for the loss of a member.

One risk that was not considered in our initial assessment was that all of our group members might be procrastinators, which is a problem our group has run into. Having a procrastinator on a team is not detrimental as long as they complete their share of the work, however having every member procrastinate means that activity prerequisites get completed late and activity completion gets pushed back even later. In order to compensate for this some members have started completing work earlier than they normally would, and we have increased our communication so that we can work on the project simultaneously to complete activities on time. This has introduced some risk of conflicting changes to the same files/pieces of documentation, which we are mitigating by more strictly assigning scheduled responsibilities.

Updated Project Plan

The updated project plan reflects a more peer-to-peer based chat, relying on the server for fewer functions. This reflects our refined scope by limiting server interaction, which simplifies the networking requirements. The updated project plan has additionally excised all the optional features to reflect some ongoing issues with individual participation.

Server

Milestone	**Days to Completion**
Connect to Clients	2
Store List of Connected Peers	1
Send List of Connected Peers	1

Client

Milestone	Days to Completion
Connect to Server	1
Store Accounts in DHT	3
Create new Account	1
Update Account information	1
Authenticate Users	1
Connect to Peers	3
Distribute List of Public Chat Rooms	1
Distribute New Messages to Peers	1
Qt GUI	7

Meeting Minutes

First Group Meeting

- Discusses multiple project options
- Decided on peer-to-peer chat application for project
- Discussed project features
    - Global administrator
    - Default skeleton chat room every user is added to on log-in
    - Discussed local area network vs internet communication
    - Discussed central peer authority vs local, private peer authorities. Decided on central authority
- IRC was brought up as being an influence of the project

Customer Interaction

- Asked about loading list of all public chat rooms piece by piece, in small sections to avoid flooding network, getting a huge list of rooms.
- Asked if all online users would be visible (privacy concerns). Only users in same rooms as you are visible.
- Asked about sending files. Not planned.
- Asked about flagging certain words, watching for “illicit (not vulgar)” speech. Not planned.
- Asked about multiple moderators/administrators per room - could be implemented
- Asked how the chat history/room information would be stored. Storage is distributed among peers. Each peer stores part of the history, client stores the full history up to a pre-defined expiration date or maximum size.
- On log-in, local machine is checked for history before getting missing history. Any deficits are served first by other peers, and ultimately by the server if the appropriate logs are not available.
- Thorough debate about persistency of chat rooms

Second group Meeting

- Discussed GUI layout and design
- Discussed UML diagrams and project layout
- Discussed possibility of resetting user password via SMS

Progress Report

Development of the project is proceeding according to our schedule. The networking specification for the project has been defined in the file “Networking Spec.txt” and all of the data types have been defined per the UML diagrams provided above.

Work on the server is slightly ahead of schedule. The server can accept incoming connections and keeps a list of peers currently connected to the network. The server can also remove a peer from the list when it disconnects or goes offline. The server can’t yet send a list of online peers to clients when they connect however.

Client development is slightly behind the server, but still on schedule. Standardizing interactions with the server is still ongoing. The initial GUI design is mostly completed for the client, but the underlying code still needs to be merged into the non-blocking connection handler threads, and the available slots/signals updated as necessary.