VicChat Architecture

SENG 299 Milestone 2

June 16, 2017

Gregory O'Hagan
Subah Mehrotra
Pranay Rai
Sean Hoessmann

Table of Contents:

1. Introduction:

1.1. Purpose:

The purpose of this document is to describe the key architectural aspects of the VicChat application. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, and the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the developers and the evaluators of the system.

1.2. Scope:

The VicChat application will be a program that allows colleagues and co-workers to communicate within teams. The application will support multiple clients connected to a persistent server.

1.3. Definitions and Acronyms:

Term	Definition
Server	A persistent program that stores and redistributes all required information.
Client	A temporary program that accesses information from the server and displays it to the user.
User	A person using a client.

1.4. References:

1.4.1. IEEE Std 830-1998. *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Computer Society, 1998.

1.4.2. Caleb Shortt. *Project Outline Software Engineering 299.* University of Victoria, Summer, 2017

1.4.3. Caleb Shortt. *Project Milestone 2 Design Software Engineering 299.* University of Victoria, Summer, 2017

1.5. Overview:

The next chapter of this document, the Architectural Description section, describes our proposed design from four different architectural views. The third section of this document, the Requirements section, describes in detail how this design will fulfill all of the requirements listed in the SRS.

2. Architectural description:

2.1. Logical View

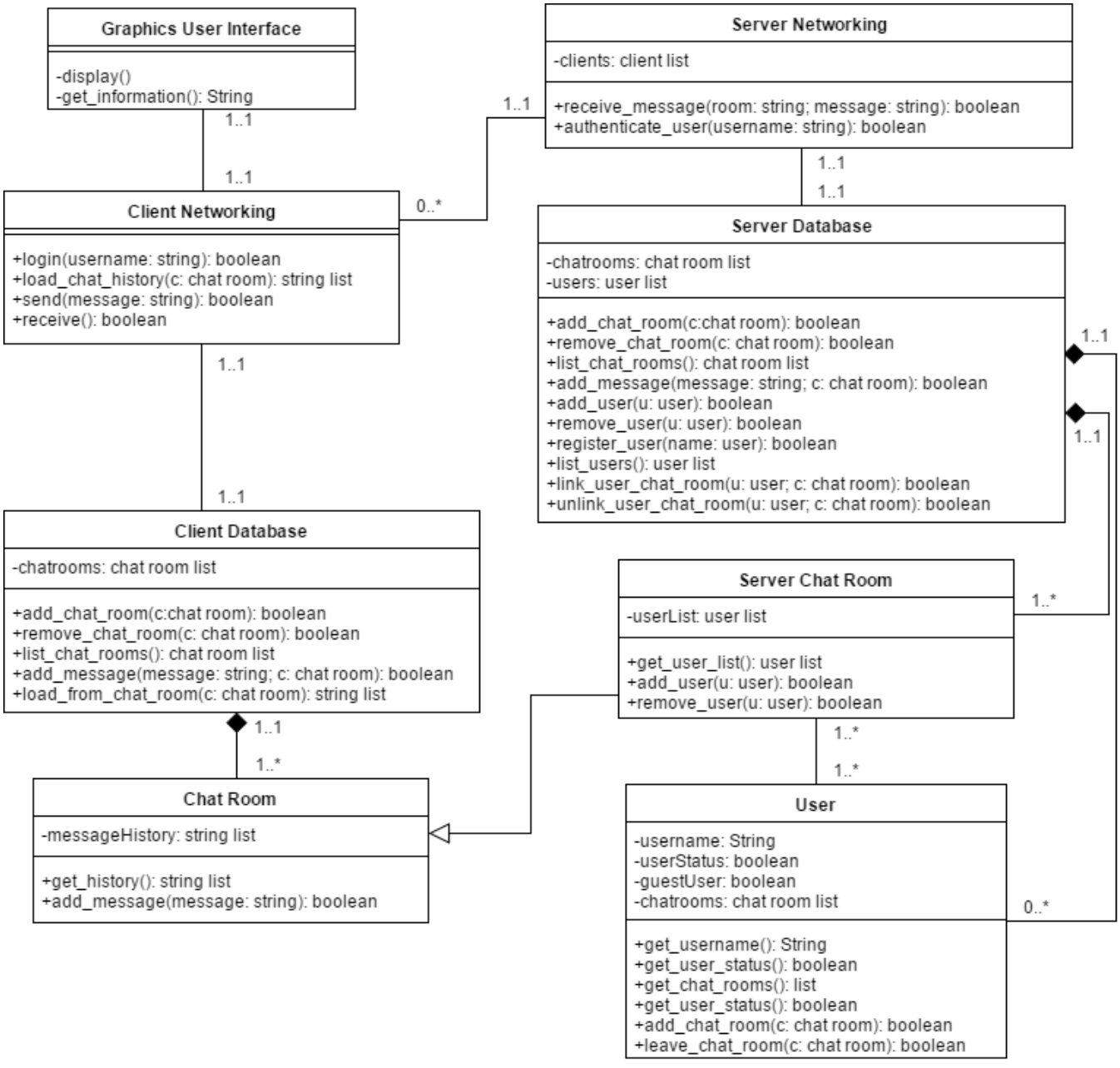

Figure 1: Class diagram for VicChat

All user interactions with the system will be performed through the graphics user interface module. This will interpret the input, and send it to the client networking module. If the client

already has the information required, it accesses it from the client database. If the information or command is not available on the client side, it sends it to the server networking module, which processes it, modifies the server database, and distributes the information to other clients as appropriate.

2.2. Process view

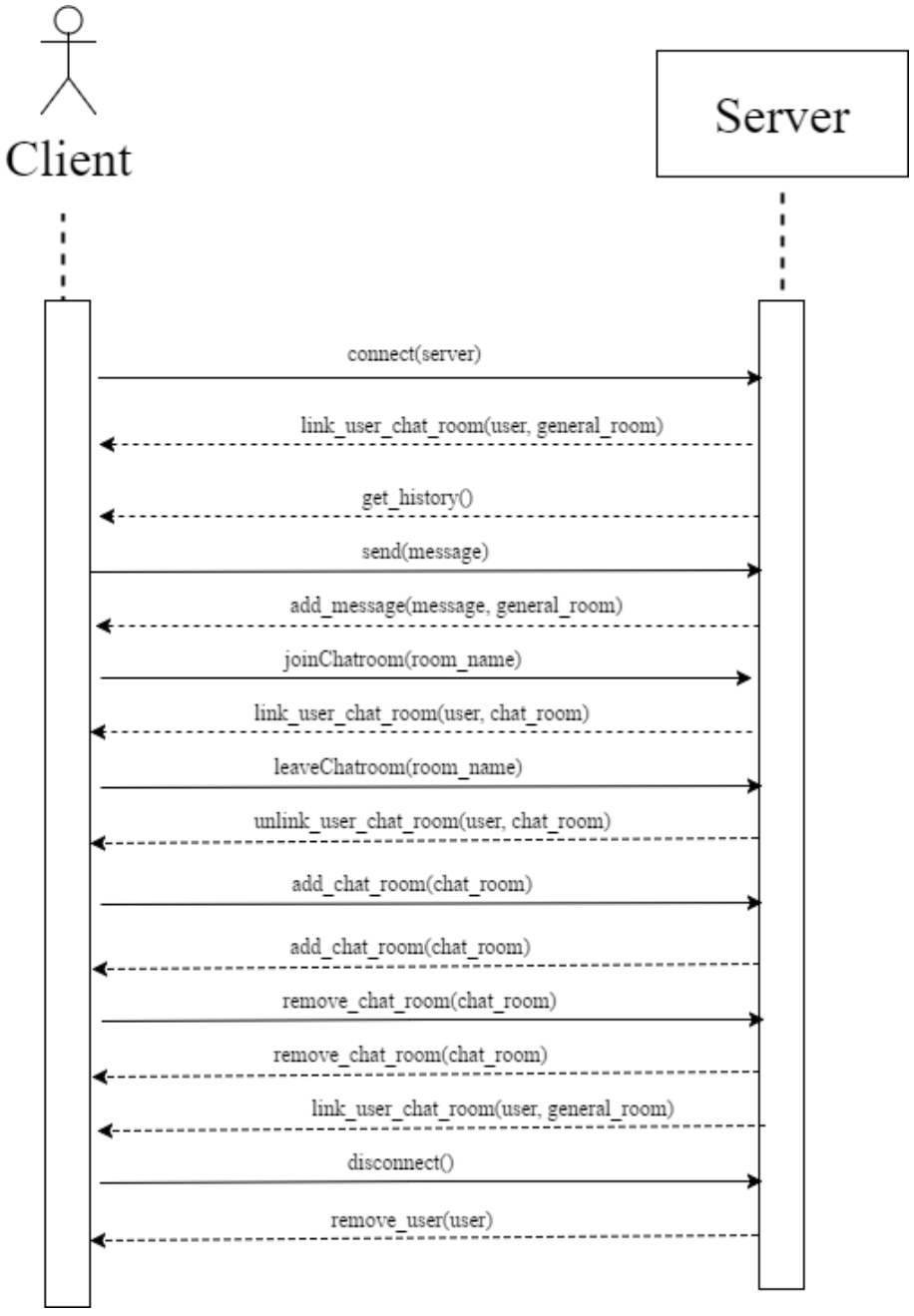

Figure 2: Sequence diagram for VicChat

Figure 2 shows the Sequence diagram of how is interaction performed between the client and server in a particular sequential order. It displays all the methods that will be used in the process.

2.3. Deployment view

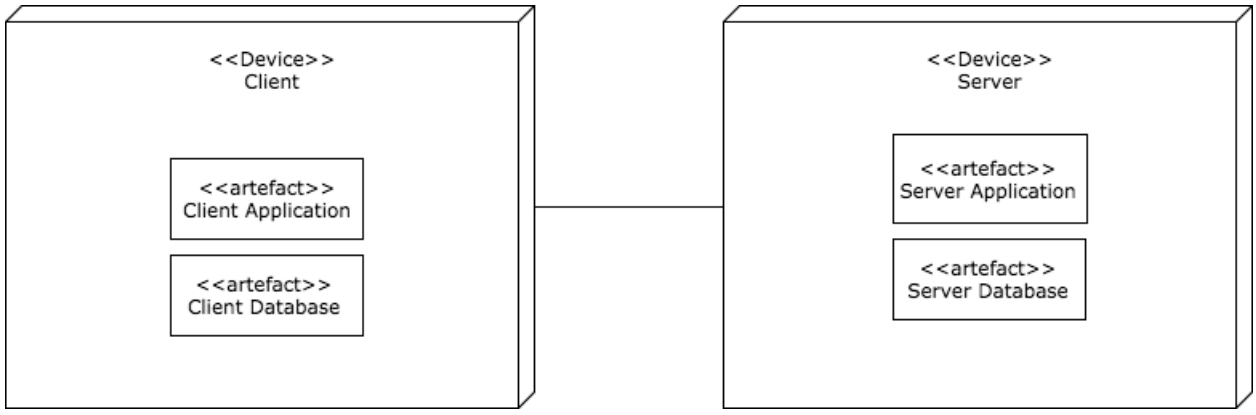

Figure 3: Deployment Diagram for VicChat

Figure 3 shows an overview of the 2 separate pieces of software that will be used for VicChat. There is a Server which is running an application alongside a database. Clients have their own application and database. The client will connect to the server and exchange information.

2.4. Data view

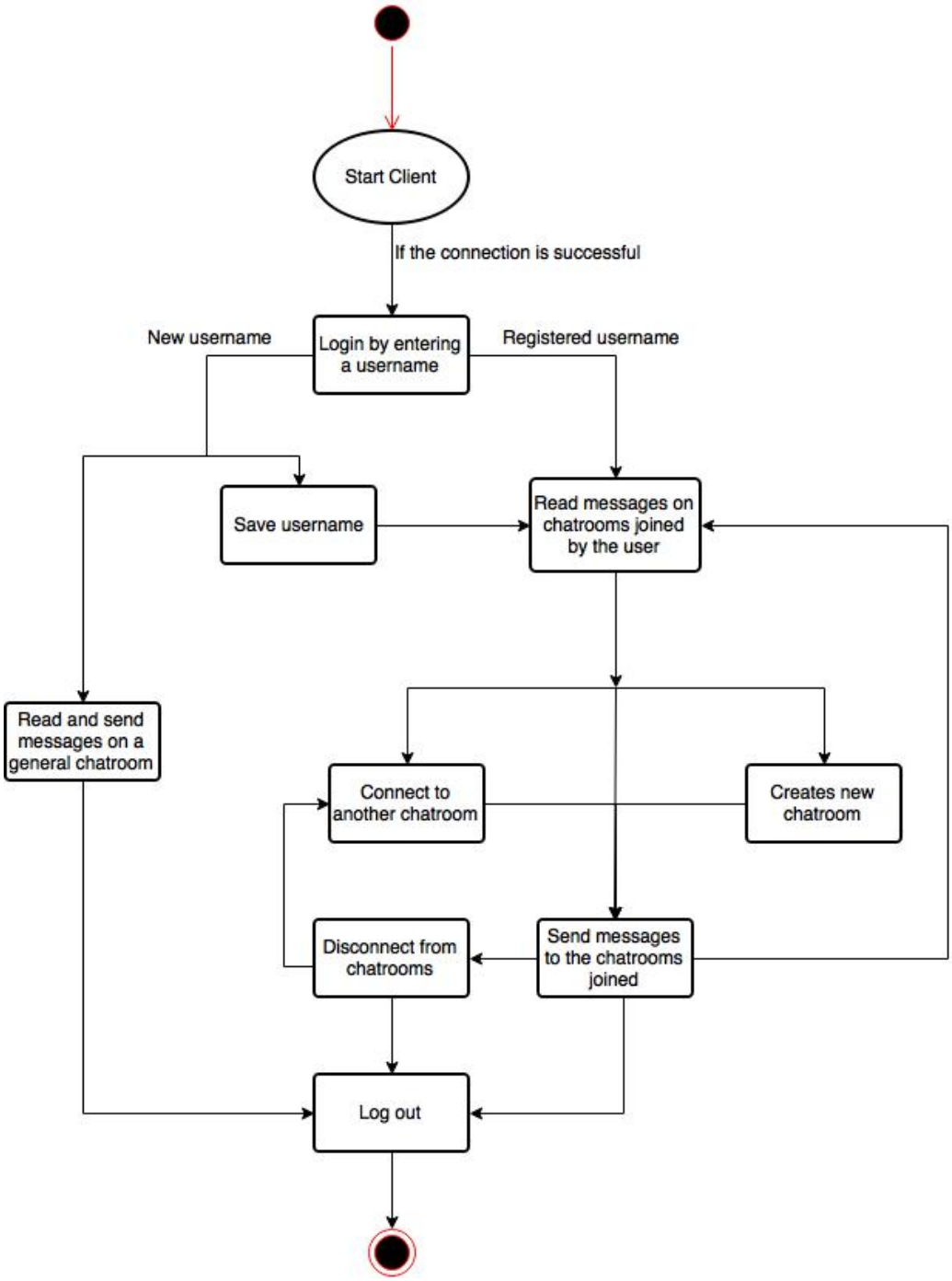

Figure 4: Activity diagram for VicChat

Figure 4 shows the graphical representation of workflows of stepwise activities and actions for VicChat. After successfully connecting the client to the server, user will be asked to login by entering a username. If it is a registered username, the client shall automatically join all chat rooms that the user is a part of. Otherwise the user will be directed to a general chat room. If the user choose to save the entered username, he/she then can use the features of the application

like disconnect from a chat room, connect to another chat room, create new chat rooms and read/send messages on the joined chat rooms. If the user decides not to save the entered username, he/she will only be able to read and send messages on the general chat room.

3. Requirements:

3.1. External interfaces:

The client only needs to interact with the display module (figure 1: class diagram) (req. 3.1.1 and 3.1.2)

3.2. Functions:

For a detailed breakdown of how each function is performed, see figure 2 (sequence diagram) and figure 4 (activity diagram).

3.2.1. Start client and login to server
3.2.2. User sends message
3.2.3. User connects to another chat room
3.2.4. User creates a new chat room
3.2.5. User disconnects from chat room

3.3. Performance requirements:

3.3.1. The lab machines used as servers will have the power required to process and distribute all requests within 1 second of receiving the information (req. 3.3.1 and 3.3.2).
3.3.2. All communication requests will generate a response once received. This allows the clients and server to ensure all networking requests are received, and deal with network errors appropriately (req. 3.3.3)

3.4. Database requirements:

As indicated on the class diagram (figure 1), each user will have a list of chat rooms they are connected to and each chat room will have a list of users connected to it. Note that this association is on the server side only: the client has a simplified chat room object.

3.5. Software System Attributes:

3.5.1. Reliability:
The lab machines used for testing will have an internet with a speed of at least 1Mbps (req. 3.6.1.1) and ping at most 150ms (req. 3.6.1.2).

3.5.2. Availability:
All server/client communication requests will send an answer once completed. This will allow us to monitor the server's availability (req. 3.6.2).

3.5.3. Security:
All client input to the server will be interpreted by the server networking class (figure 1: class diagram), which will check all inputs for malicious commands.

3.5.4. Portability:
The system will be implemented in Python (req. 3.5.1 and 3.5.2) and tested on the windows lab machines (req. 3.5.3). As such, it will run on Windows and will be easy to port to other operating systems that support Python (req. 3.6.4).

4. Appendices:

Appendix A: Design Process:

We developed this design by starting with the couple main requirements, then iteratively integrating each major requirement into our design. This ensured that we kept our initial modular vision, but still allowed us to make significant changes so that additional features are integrated well into the overall design.

Appendix B: Project Timeline:

Deliverable:	**Due date (2017):**
System design technical report	June 16
Design technical review report	July 5
Networking test and prototype GUI	July 14
Draft of code	July 21
Finalized code and in-class demo	July 28

Appendix C: Implementation Plan/Responsibilities:

Gregory O'Hagan	**Subah Mehrotra**	**Pranay Rai**	**Sean Hoessmann**
GUI development	Network architecture	Documentation throughout the project	Server architecture
Coding support	Organizing team meetings		General programming

Appendix D: Contributions:

Gregory O'Hagan	**Subah Mehrotra**	**Pranay Rai**	**Sean Hoessmann**
Logical view diagram	Data view diagram	Process view diagram	Deployment view diagram

All other work was completed as a team at meetings.