# VicChat Requirements and Project Plan

Gregory O'Hagan
Subah Mehrotra
Pranay Rai
Sean Hoessmann

## 1. Introduction:

### 1.1. Purpose:

The purpose of this document is to present a detailed description 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 1 Requirements and Planning Software Engineering 299.* University of Victoria, Summer, 2017


### 1.5. Overview:

The next chapter of this document, the Overall Description section, gives an overview of the functionality of the product. It describes the requirements informally and these requirements are elaborated more formally in the next section. The third section of this document, Specific Requirements, is aimed for developers and describes in technical detail the functionality of the product.

## 2. Overall Description:

### 2.1. Product Perspective:

VicChat is designed to operate independently. It will run in parallel with other applications on the same system.

### 2.2. Product Functions:

The VicChat clients will take input from the user and send it to the server as well as display information received from the server. The server will collect, store, and redistribute messages to the appropriate clients.

### 2.3. User Characteristics:

The VicChat application is designed to assist colleagues and co-workers in communicating on team projects. Most of our users will be technically skilled, but the application will be designed for users of all skill levels.

# 3. Specific requirements:

## 3.1. External interfaces:

3.1.1. All user interaction will be supported by the GUI. This includes connecting to chat rooms, sending messages, and displaying received messages.

3.1.2. Users will not require any additional interface to use this software.

## 3.2. Functions:

3.2.1. User starts client:

3.2.1.1. Client starts up the GUI and attempts to connect to the server.

3.2.1.1.1. If the client cannot connect to the server, it will display an error message. The user will be able to reattempt the connection.

3.2.1.2. Client requests the user's name.

3.2.1.2.1. If the username is not registered to the server, the client will join the "general" chat room. The GUI will ask the user if they want it to save their username.

3.2.1.2.2. If the username is registered, the client shall automatically join all chat rooms that the user is a part of.

3.2.1.2.3. Server shall prevent duplicate user names.

3.2.1.3. Client shall then update and display previous messages in all joined chat rooms.

3.2.2. User sends message:

3.2.2.1. User types message, then presses the "enter" key (or clicks on a "send" button).

3.2.2.2. Client sends information to server.

3.2.2.3. Server receives information, then sends it to any clients connected to the chat room that the message was sent to. The server shall also store the information for any clients that connect to that chat room.

3.2.2.3.1. If the server does not receive the information, the client shall attempt to resend the information. It shall also indicate to the user that the message was not sent.

3.2.2.4. Other client(s) receive the information and display them for the user.

3.2.3. User connects to another chat room:
- 3.2.3.1. User clicks a “join new chat room” button on the GUI.
- 3.2.3.2. Client then displays a list of all chat rooms available to join. User selects one from the list.
  - 3.2.3.2.1. User may also start typing the name of the desired chat room in a search bar at the top. This will actively filter the chat rooms displayed as the client types.
- 3.2.3.3. Server then updates client with past messages sent to the new chat room.

3.2.4. User creates new chat room:
- 3.2.4.1. User clicks a “create new chat room” button on the GUI.
- 3.2.4.2. Client prompts the user to enter a name.
  - 3.2.4.2.1. The client shall prevent the user from creating a chat room if the name is already taken by a different chat room.
- 3.2.4.3. The chat room is created, and the creator automatically joins the new chat room.

3.2.5. User disconnects from chat room:
- 3.2.5.1. The GUI shall clearly indicate a way of leaving the chat room.
- 3.2.5.2. The General chat room cannot be disconnected from. All other chat rooms shall have the option of leaving them.
- 3.2.5.3. If the last user of a chat room leaves, the room shall be deleted. The General chat room cannot be deleted this way.

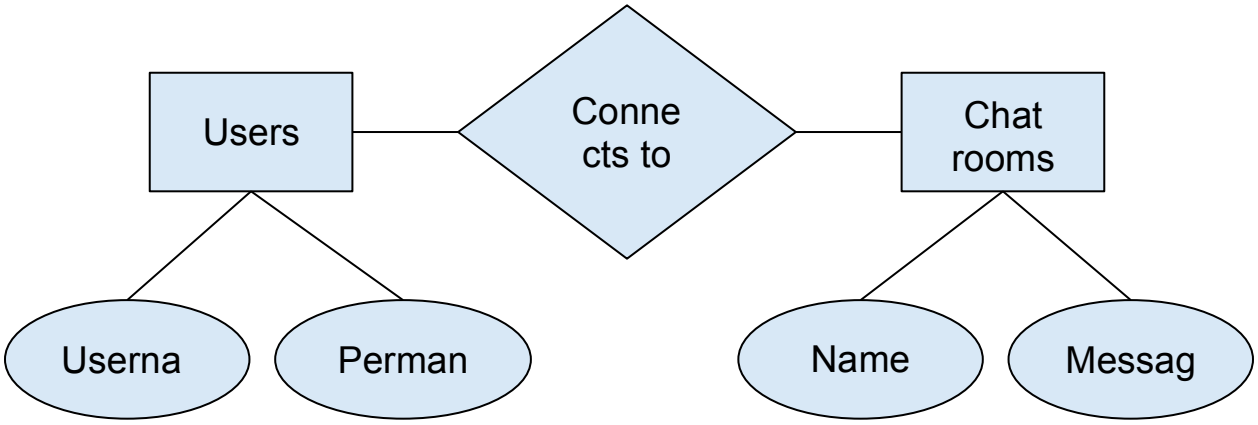
## 3.3. Performance Requirements:

3.3.1. Server shall send information to appropriate clients within 1 second of receiving the information.

3.3.2. Any other server requests (joining/leaving/creating a chat room, etc) shall be completed within 5 seconds of receiving the request.

3.3.3. Clients shall check that information was received by the server, and if it has not been, shall resend the information after 2 seconds.

## 3.4. Logical Database Requirements:




User can connect to multiple chat rooms and each chat room can have multiple users. The permanent user flag tracks whether the user is saved when the client disconnects.

## 3.5. Design Constraints:

- 3.5.1. All code must be implemented in Python.
- 3.5.2. Communication code must be written using Python requests, utlib2, or Python sockets.
- 3.5.3. The clients and server must run on computers in the lab.
- 3.5.4. The code will be managed using Git version control.

## 3.6. Software System Attributes:

- 3.6.1. Reliability:
  - 3.6.1.1. Client machines should have access to an internet connection with speed of at least 1 Mbps.
  - 3.6.1.2. Client to server ping should be at maximum 150ms.
- 3.6.2. Availability:
  - 3.6.2.1. Server shall reply to all client requests within 5 seconds, assuming the reliability requirements (3.6.1) are met.
- 3.6.3. Security:
  - 3.6.3.1. Server will check for input that will break the server.
- 3.6.4. Portability:

3.6.4.1. The client must run on Windows. As it will be completely Python based, it should be easy to port over to Linux and/or Mac OSX.

# 4. Appendices:

## 4.1. Appendix A: 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 |

Due to the set deliverable dates, we have selected the waterfall design approach. This will ensure that we can finalize our design documents on schedule.

## 4.2. Appendix B: Team Member Strengths and Assigned Tasks:

4.2.1. Gregory O'Hagan: strong analysis and debugging skills. Responsible for GUI development and coding support.

4.2.2. Subah Mehrotra: experience working with client/server architecture. Responsible for networking tasks and organizing team meetings.

4.2.3. Pranay Rai: co-op experience in software development. Responsible for documentation throughout the development process.

4.2.4. Sean Hoessmann: general programming knowledge. Responsible for server architecture and general programming.