Joshua Van Deren, Jacob Lai, Grant Savage, Joy Tan

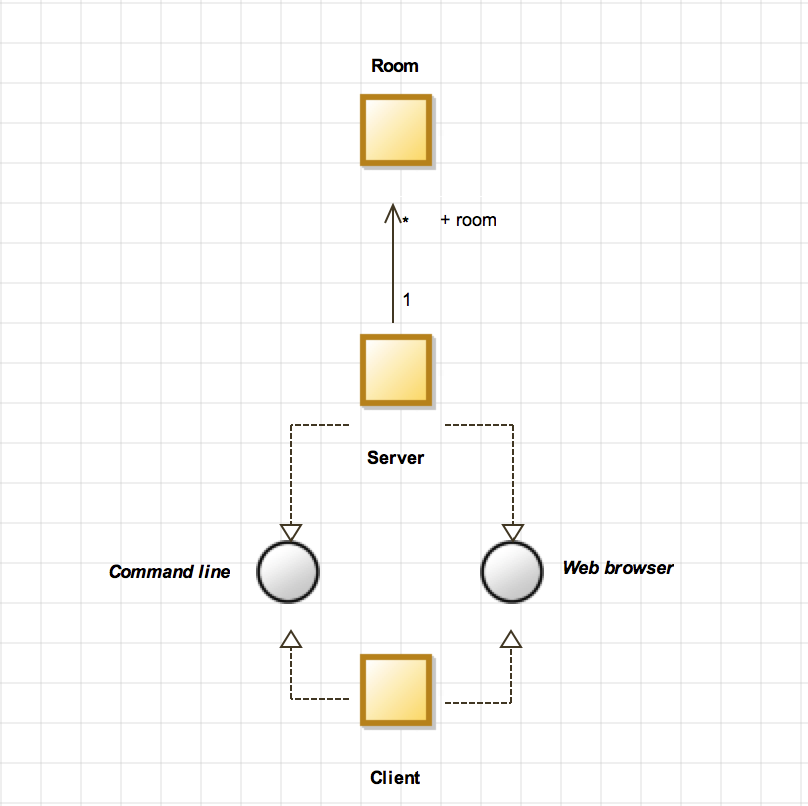
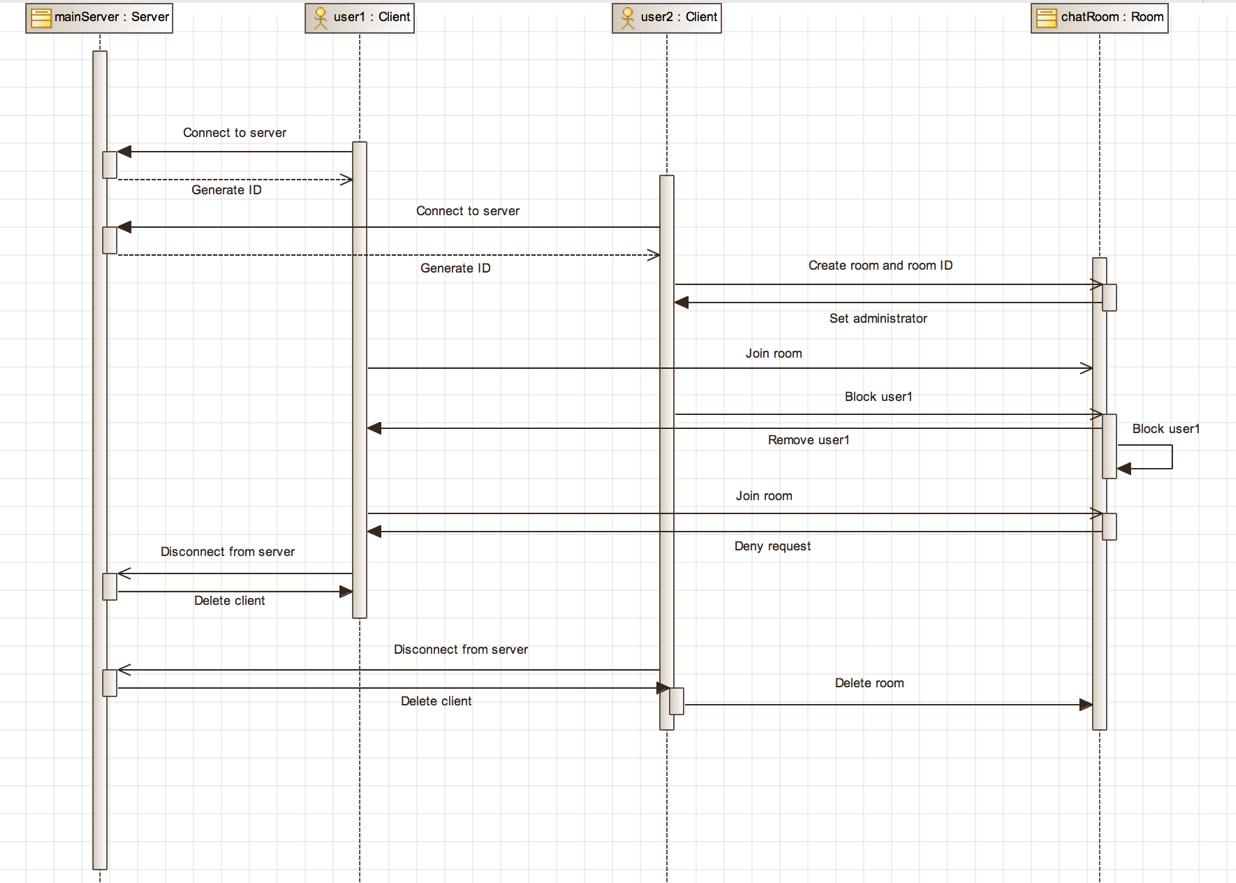
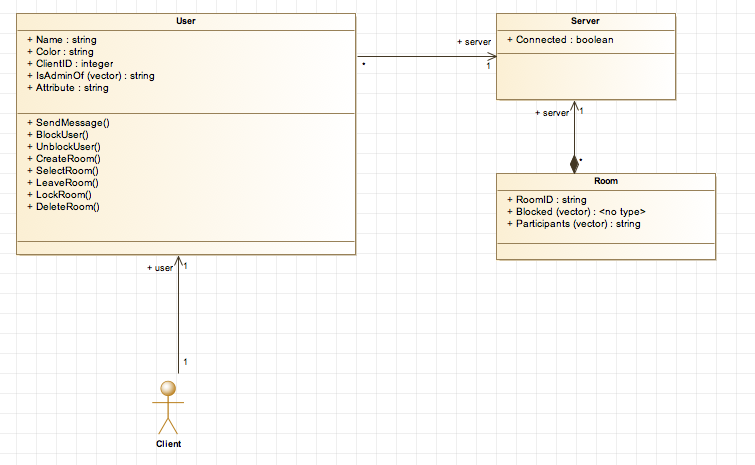
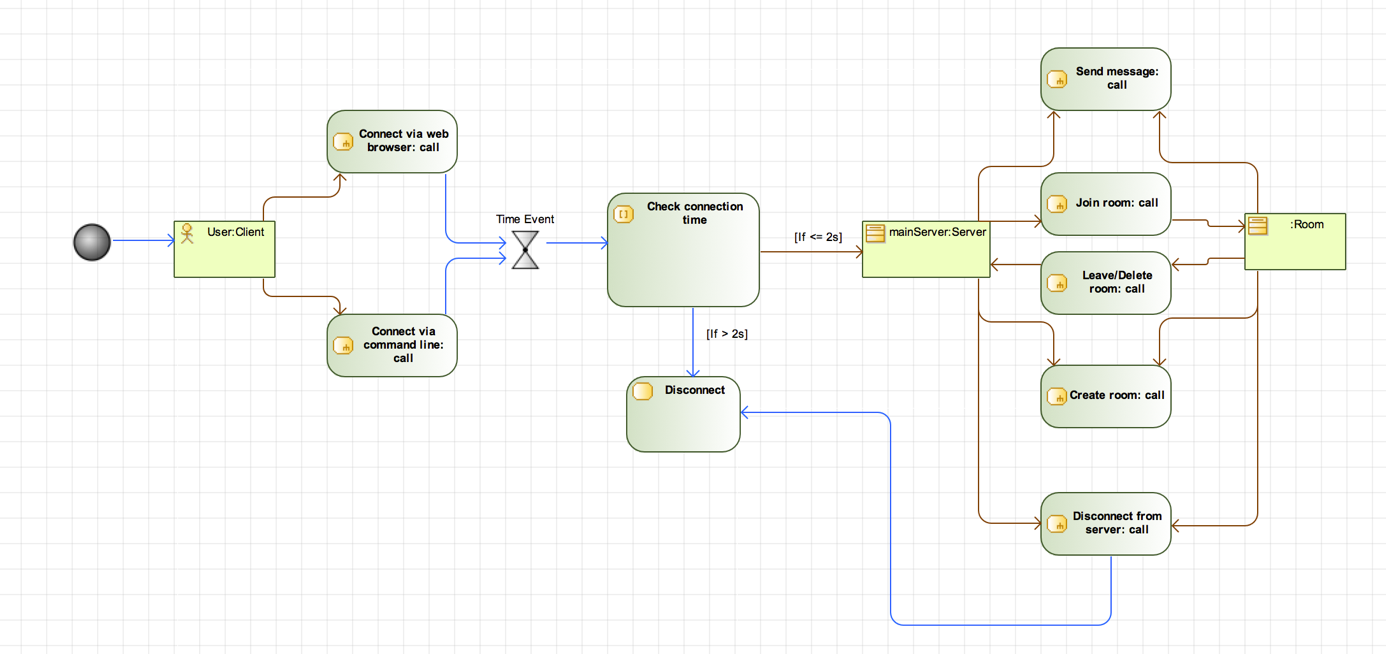
March 9, 2018

SE310

Siewert

**SE310, Analysis and Design of Software Systems**

**Exercise #4 – Group Work Requirements and Analysis of Proposed Project**

1. Seeing as communication is essential to collaboration and socialization, our command line interface chat (or cli\_chat) aims to address this. With students like us, using a command line chat interface would help open the communication bridge from any internet-connected device. This application shall be developed in Javascript, since a VHLL helps for rapid prototyping. All that is required for us to make this happen is a server to house the communication and clients to interact with each other. We also intend to utilize Socket.io to help with basic message passing and socket logic.  
     
   The server will be able to create and open sockets for other clients to connect to. An architectural pattern that will be utilized for the server-side code is a singleton since there will only be one server that should ever run for a single session. Once a client is connected, they will receive a unique identifier to distinguish them from other clients. Upon entry, a client will have the option to message other clients. We will make the ability to contact another user via nicknames in the future. Once a message is sent, the other user should receive it in a reasonable time.  
     
   A room would be useful to have, as there could be subgroups of students who need to collaborate on a topic in private. Therefore, a client will also have the ability to create a private from, from which any user can join. When a client creates a room, that client shall become admin of that room. Admins of a room will have the ability to both delete a room and block other clients from entering a room. A room is deleted once it’s admin is no longer on the server.  
     
     
     
     
     
     
     
     
     
     
     
     
   Block Diagram:  
     
     
   **Figure 1: Block Diagram**
2. Requirements for our application
   1. 5 capability requirements
      1. Must generate an ID made of a string of random characters and numbers for every user that connects to the server
      2. Every user must have the ability to create a room
      3. Every user must have the ability to send a message
      4. A user can only be in one chat room or in none at any given moment
      5. Any user has the ability to block another user or to unblock those they have blocked
   2. 3 minimum performance requirements:
      1. The time it takes between the sending of a message and the receiving of a message between two users must be as close to real-time (1/1) as possible.
      2. The client must be able to connect to the server within 2 seconds of startup. Otherwise, the client will abort.
      3. Upon client disconnection the client shall attempt to reconnect to the server within 2 seconds. Otherwise, the client will abort.
   3. 3 system requirements:
      1. Must run on at least 1GB of RAM
      2. Must run using at least 1 vCPU
      3. Must have at least 25 GB of disk space
3. 1 behavioral model (Interaction Sequence Diagram):  
     
     
   **Figure 2: Interaction Sequence Diagram (How User is Blocked)**  
     
   1 structural model (Class Diagram):  
     
     
   **Figure 3: Class Diagram**  
   1 supporting model where structure relates to behavior (Activity Diagram):  
     
     
   **Figure 4: Activity Diagram**
4. Use case to requirements tracing  
     
   R1: Send Message  
   R2: Join Room  
   R3: Leave Room  
   R4: Colored Name  
   R5: Create Nickname  
     
   U1: Peer-to-Peer Chat  
   U2: Peer-to-Room Chat  
   U3: Customize Nickname  
   U4: Block User from Room

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requirements | Priority Weights | U1 | U2 | U3 | U4 |
| R1 | 4 | X | X |  |  |
| R2 | 3 |  | X |  |  |
| R3 | 3 |  | X |  |  |
| R4 | 5 | X | X | X | X |
| R5 | 1 |  |  | X |  |
| R6 | 1 |  |  | X |  |
| Score | | 9 | 15 | 7 | 5 |

Therefore, we will work on U2 first, then U1 second, etc.  
  
Roles and responsibilities this sprint:

* 1. Joshua Van Deren
     1. Updated UML diagrams
     2. Generated SE310 assignment 4 report
     3. Created Proposal
  2. Joy Tan
     1. Added 3 new features with tests for each
  3. Grant Savage
     1. Created 5 test cases for client & server
     2. Generated SE420 assignment 4 report
     3. Developed use case to requirements tracing
  4. Jacob Lai
     1. Added 3 new features with tests for each
  5. Everyone
     1. Further developed requirements