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

February 23, 2018

SE310

Siewert

**SE420, Analysis and Design of Software Systems**

**Exercise #3 – Requirements, Specification, Analysis, and Architecture Level Design**

1. The application we chose is a Command Line Interface (CLI) Chat. The main objective of this application is to create a platform where users can connect to a server and chat with other users. Users can either chat one on one or partake in a group chat. This project includes many operations:
   1. 5 major function and feature 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. 5 performance requirements:
      1. The server application must be deployed on a Virtual Private Server that has at least 1GB of RAM, 1 vCPU, and 25 GB of disk space.
      2. 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.
      3. The client must be able to connect to the server within 10 seconds of startup. Otherwise, the client will abort.
      4. Upon client disconnection the client shall attempt to reconnect to the server within 10 seconds. Otherwise, the client will abort.
      5. The delay time between message sending and receiving in a room shall be the same across all clients regardless of how many clients are connected to a room.
   3. 5 error handling, recovery, or ease of use features:
      1. (Error Handling) Prevent usage of arbitrary characters
      2. (Recovery) When there’s a lost connection, there should be a time frame to reconnect
      3. (Error Handling) Check and recover from incorrect flag usage
      4. (Error Handling) Check for non-empty input when creating nickname
      5. (Error Handling) Check for non-empty input when sending a message
2. For mapping each requirement in part 1 to an acceptance test plan . . .
   1. 5 Major Tests for Function and Features
      1. Emit a ‘connect’ event channel call to admit a user into the server with a unique UUID
      2. Emit a ‘create room’ event channel call to request a room
      3. Emit a ‘private message’ event channel call to see if message sends
      4. Emit a ‘join room’ event channel and the user with switch to that room (if exists) or an error message will pop up
      5. Call a method called ‘block’ and ‘unblock’, to block or unblock another user.
   2. 5 Workload Tests
      1. Test the server by using the command ‘free -m’, ‘df –block-size=1MB’, and ‘nproc –all’
      2. There will be two active sessions, where one user will send a message, and another user will receive a message in real-time
      3. Have a stop watch and see if the user can connect within 10 seconds
      4. Have a stop watch and try to reconnect close to 10 seconds
      5. There will be more than two active sessions, where one user will send a message, and all users will receive that message in real-time
   3. 5 Tests to Generate Faults or to Force Failure:
      1. Try to send arbitrary characters
      2. Try to reconnect close to 10 seconds, determined by using a timestamp, after losing a connection
      3. Try using an unknown flag to see if the server catches it
      4. Try setting an empty string as a nickname
      5. Try sending an empty string to another user
3. 10 Suggestions to Improve Code Module:
   1. Make the program more OOP
   2. Design a web-based client interface
   3. Create unified style guide
   4. Continuously add meaningful comments
   5. Give variables meaningful names
   6. More rigorous error checking / fault tolerance
   7. Perform code coverage testing using LCOV
   8. Convert code to executable for distribution
   9. Upgrade server for higher performance
   10. Horizontally scale our current server for redundancy