Spirent-pep-logo

**2018 International Internship Program**

**2nd round screening process**

**Project Design Documentation**

**Dongjin Park**

# Table of Contents

[Table of Contents 2](#_Toc512289448)

[I. Introduction 3](#_Toc512289449)

[I.1 Purpose 3](#_Toc512289450)

[I.2 Tasks 3](#_Toc512289451)

[II. Software Design 4](#_Toc512289452)

[II.1 Overview 4](#_Toc512289453)

[II.2 Flow Charts and Diagrams 5](#_Toc512289454)

[II.3 Class Relationships 11](#_Toc512289455)

[II.4 Database Schema 12](#_Toc512289456)

[III. Functionality 13](#_Toc512289457)

[IV. References 14](#_Toc512289458)

# Introduction

This project is to develop a chat-messaging system. It is based on server and client using multithreading, sockets, and database. There is one chat room, and clients(users) can chat in the chat room. Server and client are written in Java(jdk10) and database is written in MySQL(Centos5). My development environment is Windows7 64bit, and I use Intellij as a develop tool.

## I.1 Purpose

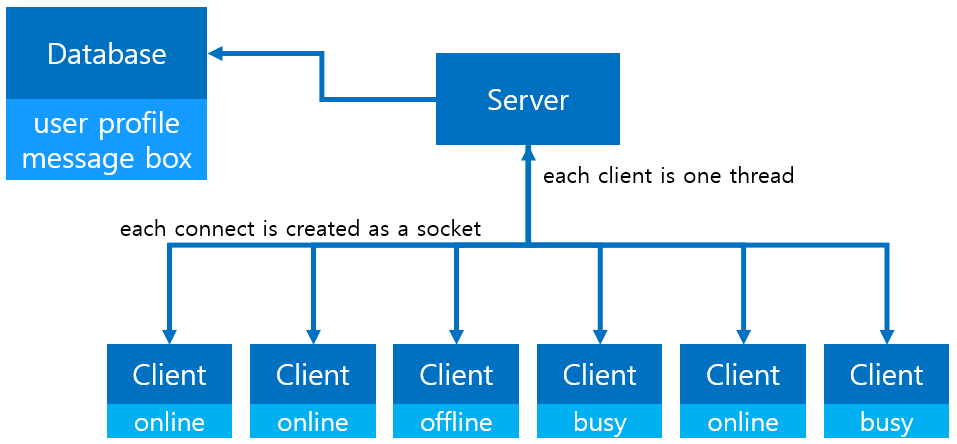
The purpose of this document is to describe the design of Chat Messaging System and how it works.

## I.2 Tasks

* implement server which spawns one thread per client(multithreading) and users TCP socket to communicate with client and also manages all message flow
* implement database which is store user profile and chat history
* implement client which is able to chat with another client and sign in/up

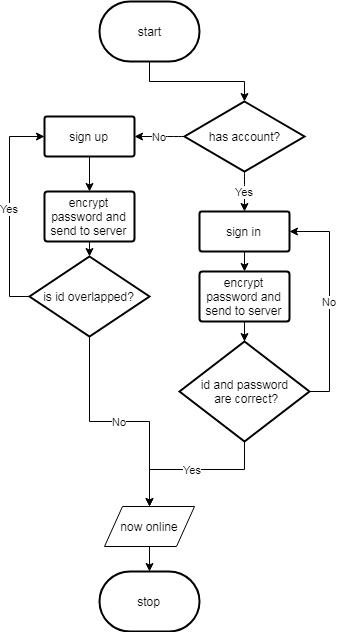
# Software Design

## II.1 Overview

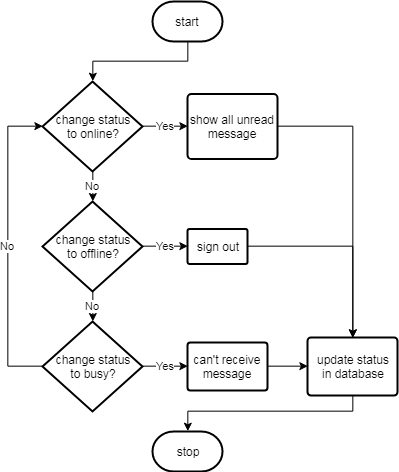


- Figure 1. overview -

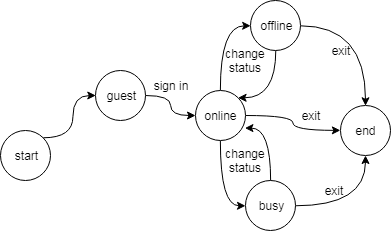
## II.2 Flow Charts and Diagrams



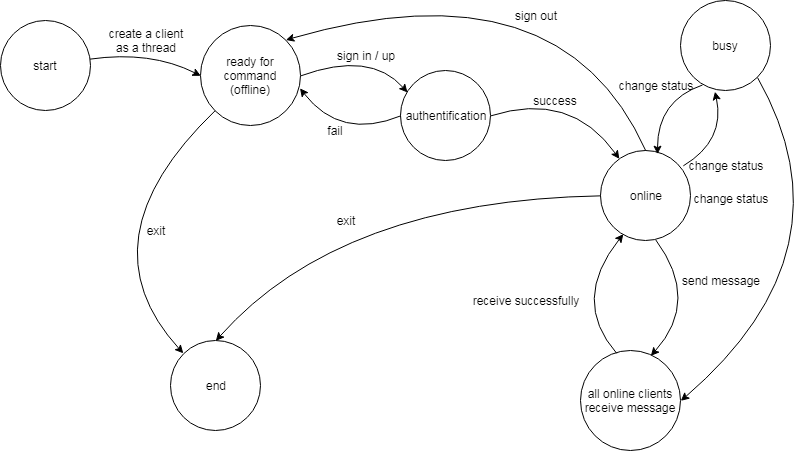
- Figure 2. sign in/up flow chart -



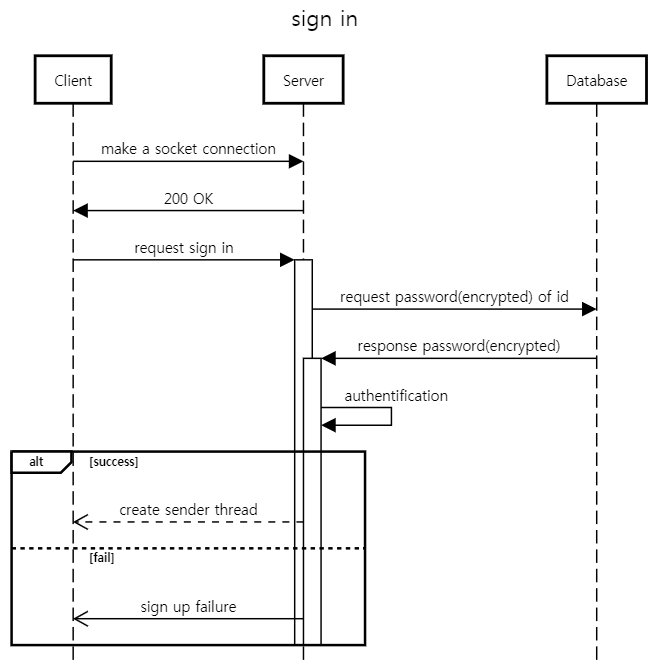
- Figure 3. change status flow chart -



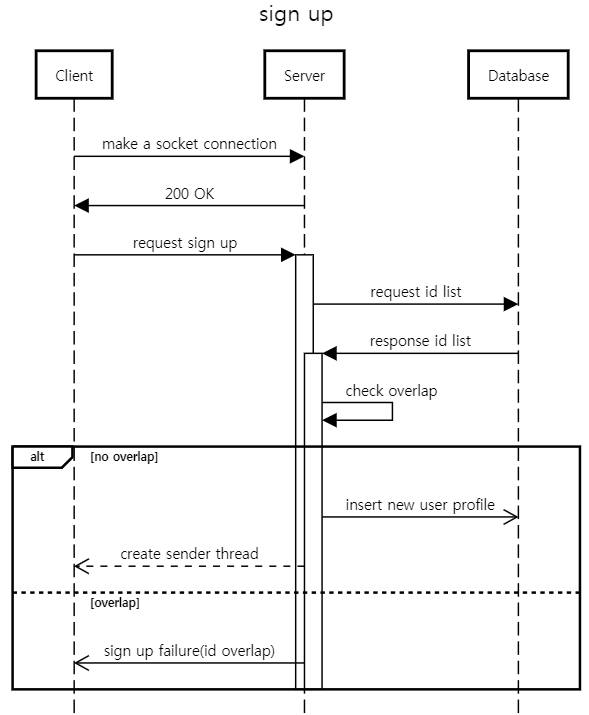
- Figure 4. state transition diagram(simple) -



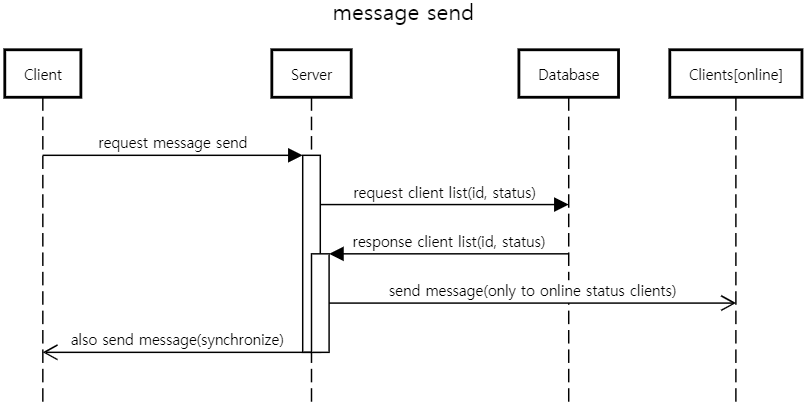
- Figure 5. state transition diagram -



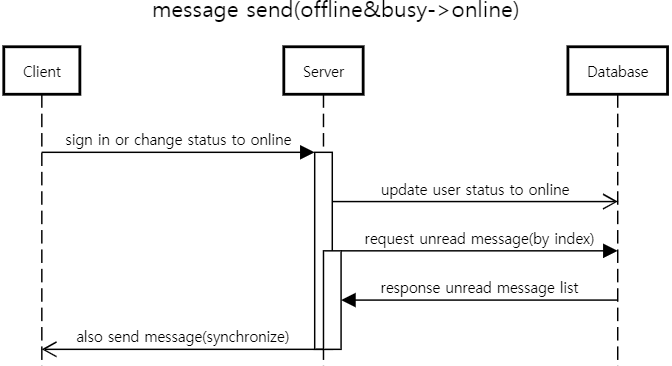
- Figure 6. sign in sequence diagram -



- Figure 6. sign up sequence diagram -

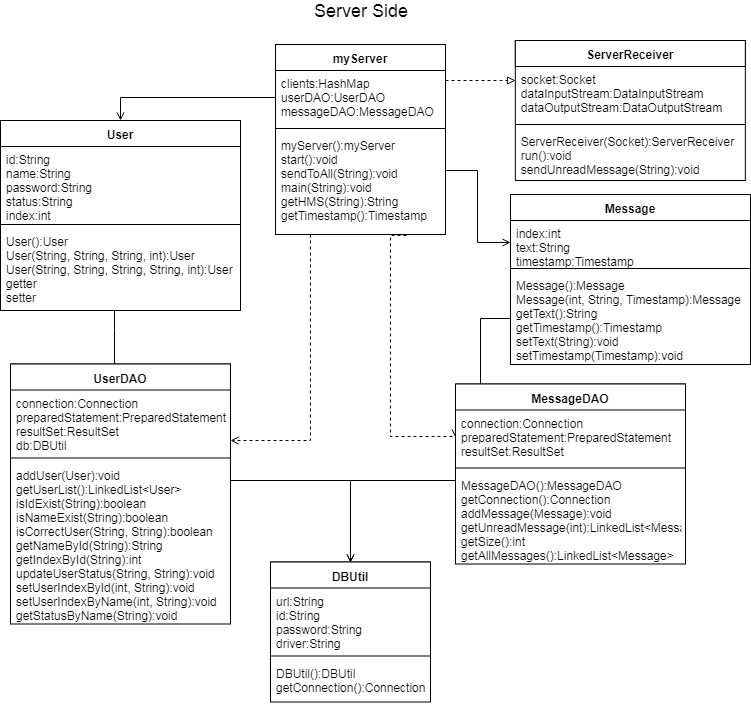


- Figure 7. message send sequence diagram -

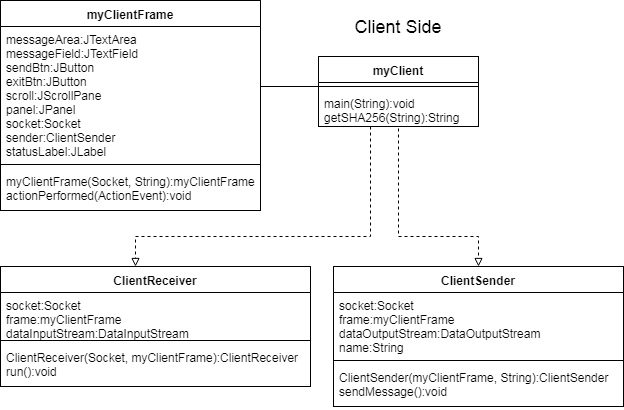


- Figure 8. message send(offline&busy->online) -

## II.3 Class Relationships

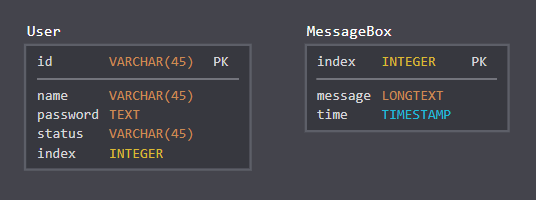


- Figure 9. Server Class Diagram -



- Figure 10. Client Class Diagram -

## II.4 Database Schema



- Figure 11. Database Schema -

In User, id is primary key. So user’s id must not be overlapped. Also there is name which is used in chat room. User password is stored as encrypted plain text(by SHA-256) for information security issue. User status is also in database. If user changes status, database must be updated. The last one, index indicates how much this user has read the message.

MessageBox has all messages which clients send. All messages have unique index(in order) and it is a yardstick of read or unread. And also all messages have to include the time stamp when it is sent. There is a NOTICE message that is sent by server, and it is not stored in the database.

# Functionality

1. Server
   1. manage all clients as a thread(multithreading)
   2. use TCP/IP socket connection with client
   3. use database when receives request from client
   4. check id and password when client requests sign in
   5. check if id is overlapped when client requests sign up
   6. close connection when client requests exit
   7. manage all clients and messages
      1. when receive message, send to all users and insert into database
      2. read command(/command) and send specific message(unread message, userlist)
      3. all messages include timestamp from server
      4. use DAO(Data Access Object) to access database
      5. check client status(connected, disconnected) in real-time
      6. check number of client in real-time
2. Client
   1. sign in
      1. if you have an account, you can sign in immediately in console UI.
      2. enter your id and password.
      3. if password is incorrect, you need to enter the correct password again.
   2. sign up
      1. if you don't have an account, you can sign up in console UI.
      2. all you need to enter is name, id, password. Your password will be encrypted(SHA-256) and stored in database.
      3. if same id or name is already exist, you should use another id or name(notice which one is overlapped).
   3. send message(press enter or click “send button”)
   4. special commands(/online, /offline, /busy, /userlist, /retrieve, /exit)
      1. /online : your status will be changed to "online". Also you will receive messages that you can't reveive before when your status was "busy" or "offline".
      2. /offline : your status will be changed to "offline". Now you can't send and receive messages, but you're still in the chat room.
      3. /busy : your status will be changed to "busy". You can send message but can't receive messages.
      4. /userlist : you can see the whole user's name and status.
      5. /retrieve : you can retrieve all message history in database(MessageBox). I tried to make this function to retrieve 10~20 messages each time, but it needs another field in database. So I implement it to retriece all message.
      6. /exit : the program will be closed. You can exit the program without clicking the "exit" button.
   5. status
      1. you can check your current status on the top of window.
      2. in online status, user can send message and receive message.
      3. in offline status, user can’t send or receive message but can change the status. It is different from exit the program.
      4. in busy status, user can send message but can’t receive message.
   6. receive all unread message when status is changed to online.
   7. load message history.
   8. exit program(enter /exit or click “exit” button).

# References

The Standard Formula of Java (NamGoongSung, 2016)