Spring 4-15-2020

CHAT APPLICATION WITH DISTRIBUTED SYSTEM

Shuyang Zhu

CHAT APPLICATION WITH DISTRIBUTED SYSTEM

by

Shuyang Zhu

Applied Project report submitted to the Faculty of the Graduate School of the

HARRISBURG UNIVERSITY OF SCIENCE AND TECHNOLOGY

in fulfillment of the requirements for GRAD 699 of the

Computer and Information Sciences program

Supervised by: Abrar Qureshi, Ph.D.

2020

Abstract

This project created a web application for users to instantly communicate with each other. An the most important part of building a good chat application is focus on the data flow on web. This project spends a good amount of research to find the most appropriate technology for deliver the data flow, which is REST api for account management, WebSocket for text chat and WebRTC for video and audio and find the result pleasant.

Keywords: chat application, web socket, web rtc

1. Introduction

1.1 Purpose

The purpose of this project is to design a chat application, also known as a instant messaging system. The main purpose of the software is to provide users with an instant messaging tool that has the ability to handle millions of users simultaneously when needed and can be easily done.

1.2 Scope

The application is designed as a web application. It provides a general architecture for chat applications, and anyone or organization can use it as the basis for providing instant messaging capabilities. The application is written in an object-oriented language called Java. This project uses Java version 8. Function flow, lambda, and NIO play a very important role in the implementation. The application is divided into two parts, server and client. The server is hosted on a local computer with Apache Tomcat and will be moved to Cloud AWS at the end of this project, because AWS can make the scaling process simple and cost-effective. But temporarily, the application will remain on the local computer. Clients do not need to install any software on their machine. Only network access is needed for communicating with each other.

1.3 Definitions, Acronyms, and Abbreviations

Definition/Acronyms	Description
JVM	Java Virtual Machine
OOP	Object Oriented Programming
API	Application Program Interface
TCP	Transmission Control Protocol
UDP	User Datagram Protocal
IM	Instant Messaging
IP	Internet Protocol
IO	Input output
NIO	Non-blocking IO
SSE	Server Side Event
webRTC	Web Real time Communication
REDIS	Remote Dictionary Server

Table 1: Abbreviations

1.4 Overview

This project provides the instant communication functionality between users. The users have the capability to do one to one communication, group communications and video/audio chat. And the users can receive messages instantly while online and still be able to see the messages sent while they are offline. The client-side application can be hosted on any machine with JVM. And communicate with server side with socket.

2. System Requirements

The following are the various types of requirements:

2.1 Functional Requirements

The following are the functional requirements of the system:

- The system shall provide the user the ability to create a new account.
- The system shall provide the user the ability to login with the username and password chosen at the time the account was created.
- The system shall deliver sender's message to receiver instantly if the receiver is online.
- The system shall deliver sender's message to receiver once the receiver is back online if the message was sent when receiver's offline.
- The system shall present the sender's message to all other users in the group if the message is sent to a group.
- The system shall allow multiple users chat in one single room.
- The system shall allow user to talk via video
- The system shall allow user to talk via audio

2.2 Non-Functional Requirements

The following are the non-functional requirements of the system:

- The system shall be a web-based application that can provide all the functions over the internet.
- The system shall deliver messages in the same order it gets sent out.
- The system shall guarantee the delivery. And shall notice sender if message is not delivered successfully.
- The system shall be scalable and robust.
- The system shall be cost efficiently. It shall not cost a lot if there is not many users.
- The system shall deliver the message relatively quickly.

2.3 Performance Requirements

The performance of the application is measured based on the time delay and package drop rate for each message and the response time for user's actions.

- The login and account information loading time should be less than 3 seconds.
- The time in between message sent and received should be less than 1 seconds when total online user's number < 10000, and less than 10 seconds with unlimited number of total online users.
- The dropped package rate for video/audio chat should be less than 1%

2.4 Database Requirements

MySQL is used to store all the information we want to keep, especially data that does not need to be changed frequently, such as user authentication and authorization information, such as username, password. There is also information such as user friend relationships. Connecting and reading the database is time consuming because the database needs to read and write from disk. Therefore, the project uses the in-memory database REDIS to use both storage and caching. REDIS provides storage for key-value pairs. It rarely reads and writes to the database, and most of the time it performs data operations in memory, which saves a lot of time and greatly reduces the response time of the application. Key-value pair structures can also speed up data lookups.

3. Technology discussion

The one of the most important targets of this project is the delivery information instantly. Thus picking the right technologies in communicating has became the main focus of this project.

3.1 OSI model

The Open Systems Interconnection model (Refer as OSI model below) is a conceptual model which describes how network systems communicate to each other. Which is the fundamental of all communicating technologies.

The top layer which is closest to the end user is the layer seven- Application layer. The application layer is the layer which end users actually interact with, such as Firefox, Internet Explorer, Safari, Chrome. In our project, this would be the high-level APIs, resources.

The next layer which is layer six – Presentation layer. The presentation layer normally refer to the layer where the operation system works on. This layer covers data formatting, encoding, encryption and compression.

Under Presentation layer is the Session layer. Session layer is the layer that deals with the communication creating a session between the two computers. For example, when the user visit a website. The computer of this user will create a session with the web server which the website is sitting on at the session layer. And it also includes authentication.

The layer 4 is called the Transport layer. The transport layer decides how much information should be sent at one time. Including how much data the user will send and how much data the server will send back. This layer deals with the transport of the data back and forth and responsible for massage segmentation, acknowledgement and resending the data if lost.

The layer three is the network layer. The network layer is the layer that routers operate on. The IP address is in the network layer.

The layer two is the data-link layer which is the layer where switching occurs. All the computers including server on a network need to be plugged into a switch so they can communicate with each other.

The last layer also the layer one is called physical layer, which is all the wiring that connects computers together, including patch cable.

We can consider all seven layers as seven steps the information needs to go through in order to be sent to the network. When we need to enhance the speed of messages/video travel. We can try to improve any of the seven layers.

3.2 TCP IP Model

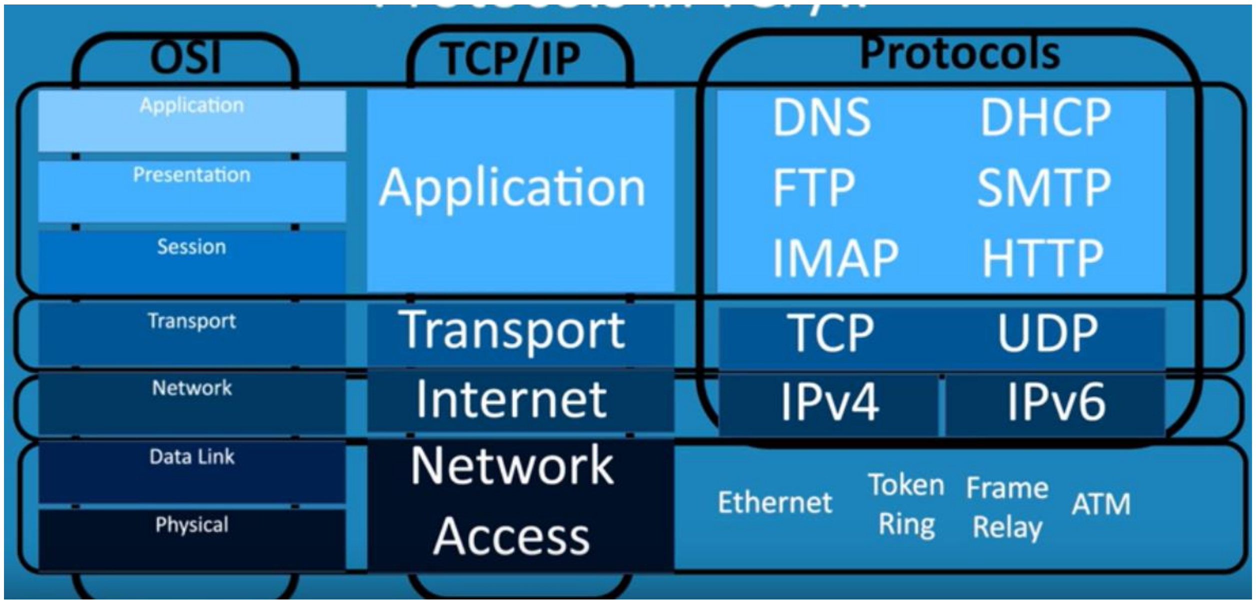
In order to make for easier communication, a series of standard protocol has been developed. Those protocols are grouped under the name TCP/IP. TCP/IP stands for Transmission Control Protocol and Internet Protocol. TCP/IP have 4 layers. Application layer, Transport layer, Internet layer and Network Access layer. Those 4 layers can be mapped to OSI model. The application layer in TCP/IP mapped to the first 3 layers in OSI model- application layer, presentation layer and session layer. The Transport layer in TCP/IP maps to transport layer in OSI. The Internet layer in TCP/IP maps to the Network layer in OSI. And the last 2 layers in OSI maps to the Network Access layer in TCP/IP.

In each layer of TCP/IP models, we have many different protocols. In the application layer, we have DNS, DHCP, FTP, SMTP, IMAP and HTTP.

In the transport layer, we have TCP and UDP. The TCP stands for Transmission Control Protocol. TCP gives reliable transmissions. If a packet is lost in transmission, the protocol will notice it has not arrived and request a resend. In contrast, UDP stands for User Datagram Protocol, is unreliable transmission. If a packet goes missing during transit, there is no resend.

And we have IPv4 and IPv6 in the Internet layer.

Most of enhancement technologies try to do is on the application layer. In this specific project, most network communication relies on HTTP.



3.3 Real-time Technologies

There are couple of matured technology been around for a long time in order to build a real-time web application, such as http, WebSocket and SSE.

3.3.1 HTTP

HTTP follows the standard client-server model. It has header and body. Client will send a http request to the server. Those requests must have headers. The header includes some structure information, such as the type of body, example json text or xml. And it can also include the action the client wants to do, such as put/delete/get. It can also include any type of customized information specific to one application. And the request may or may not have the source information in the body.

After the server successfully receive the http request, it will go through a series of actions and return back some result which also in a header-body format. Normally the http will return immediately after the server finish the action. While in this project, after the client send one message to the server, server don't have anything to return since the message is actually send to another client the server is just the deliver guy. The server need to wait for the other client to response in order to have something return the this client.

Thus we have an special type of http called Long-polling http. Polling means the client send request to the server regularly. Long-polling means each of the regular client request will wait for a while for the server to get a response.

3.3.2 WebSocket

WebSocket is a computer communications protocol, providing a stabled two way channel over a TCP connection. It's located at layer 7 as same as HTTP. But it's different than HTTP, it work over HTTP port 80 and 443 and use upgraded HTTP header to be a Websocket header.

The client will send a websocket request to server, then a communication channel will be established so that the server can send message to the client later without waiting another request from client, which is usually how http works. This works very well for the scenario that server and client send messages to each other very frequently.

3.3.3 WebRTC

WebRTC, the name derived from the abbreviation of Web Real-Time Communication, is a real-time communication technology that allows network applications or sites to establish peer-to-peer (Peer-Peer- to-Peer) connection to realize the transmission of video stream and / or audio stream or any other data, and support web browser for real-time voice conversation or video conversation. These standards included in WebRTC make it possible for users to create peer-to-peer data sharing and conference calls without installing any plug-ins or third-party software. It is a technology acquired by Google in May 2010 for $ 68.2 million in the acquisition of Global IP Solutions, which has technologies such as codec and echo cancellation. The project is a fusion of the GIPS project and the libjingle project. The GIPS part mainly provides media processing functions. The libjingle project part mainly provides the functions of the P2P transmission part. In May 2011, the source code of the project was opened, and industry standards were established with the relevant institutions IETF and W3C to form the existing WebRTC project. It was widely supported and applied in the industry and became the next-generation video call standard

3. Software Specification

This application is written in Java, thus there is not much requirement for clients' machine. The following specification is the software and hardware we used while developing the application.

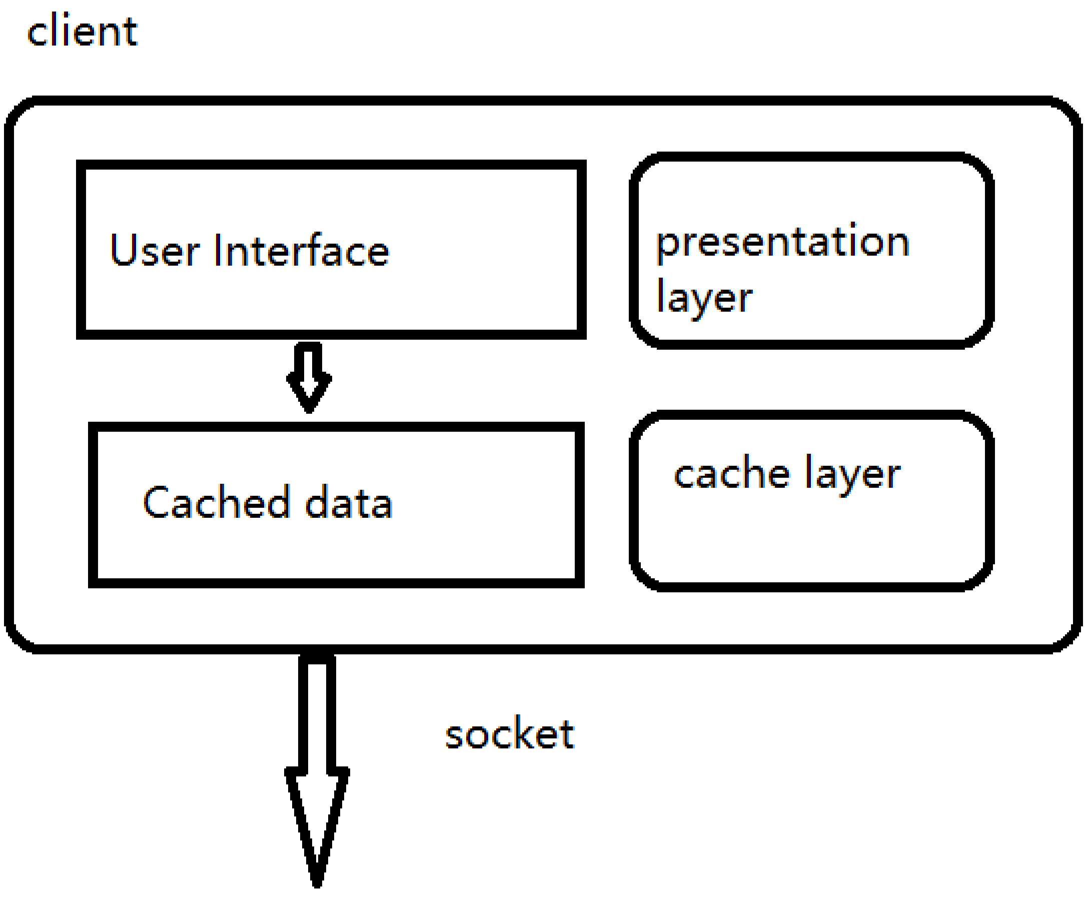
Software	
Operating System	Windows 10
Programming Language	Java 8
IDE	IntelliJ

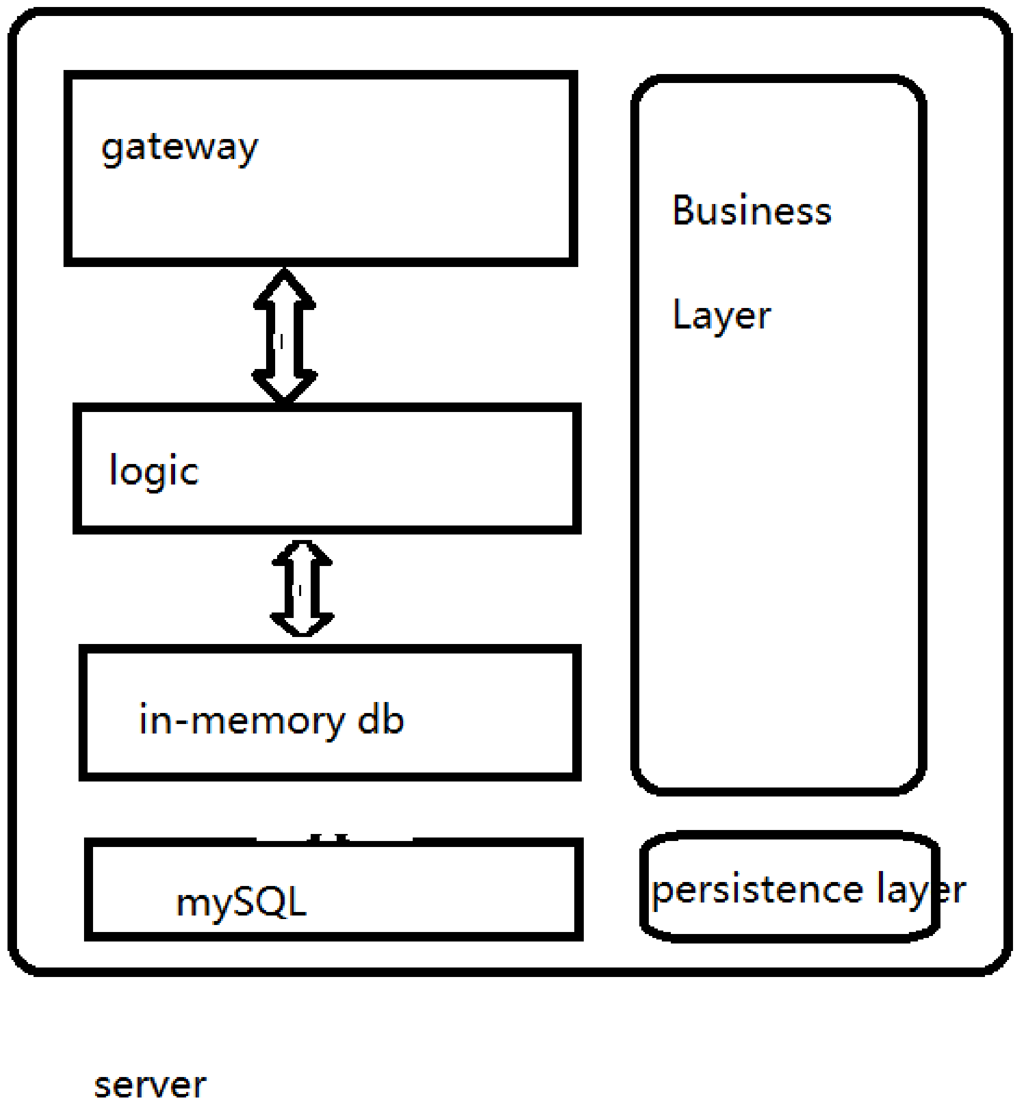
Frontend Technology	Angular
Database	MySQL, REDIS
Application Server	Apache Tomcat
Hardware	
Architecture	64 Bit windows architecture
Processor	Intel i3
RAM	5GB per server instance

4. System Design

Below is the detail of architecture design of the chat application.

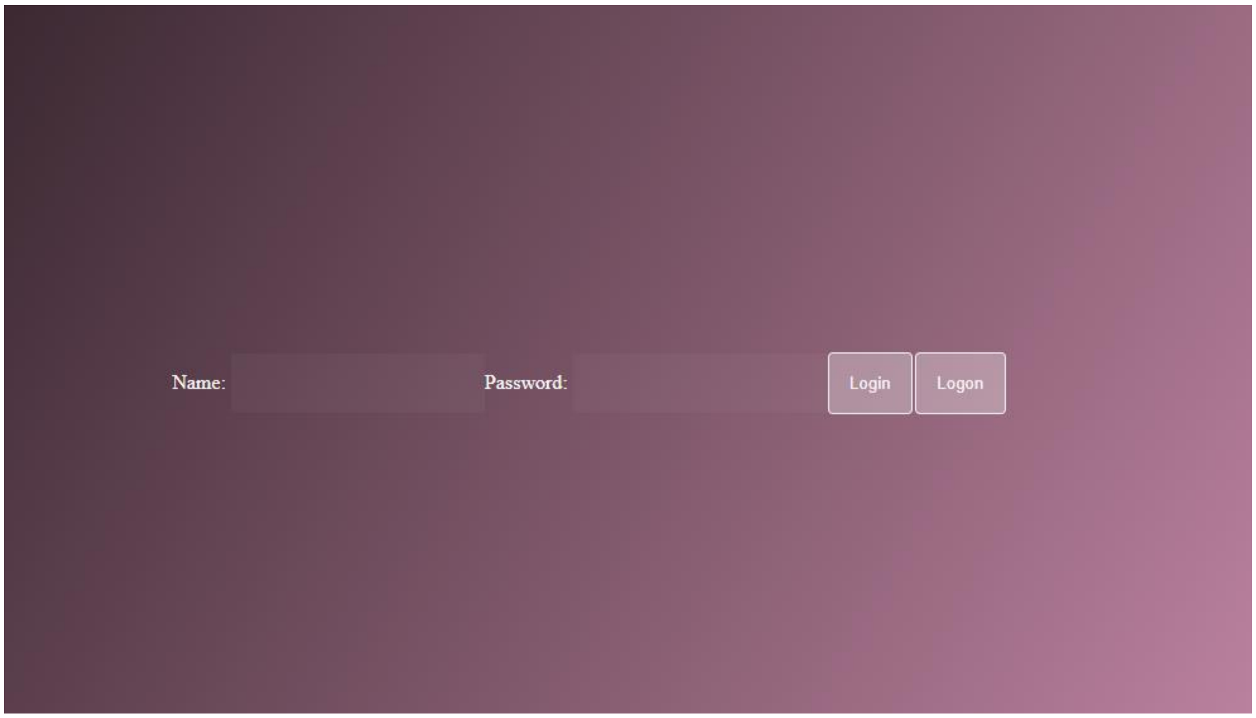
4.1 Architecture


client
User Interface
presentation layer
Cached data
cache layer
socket

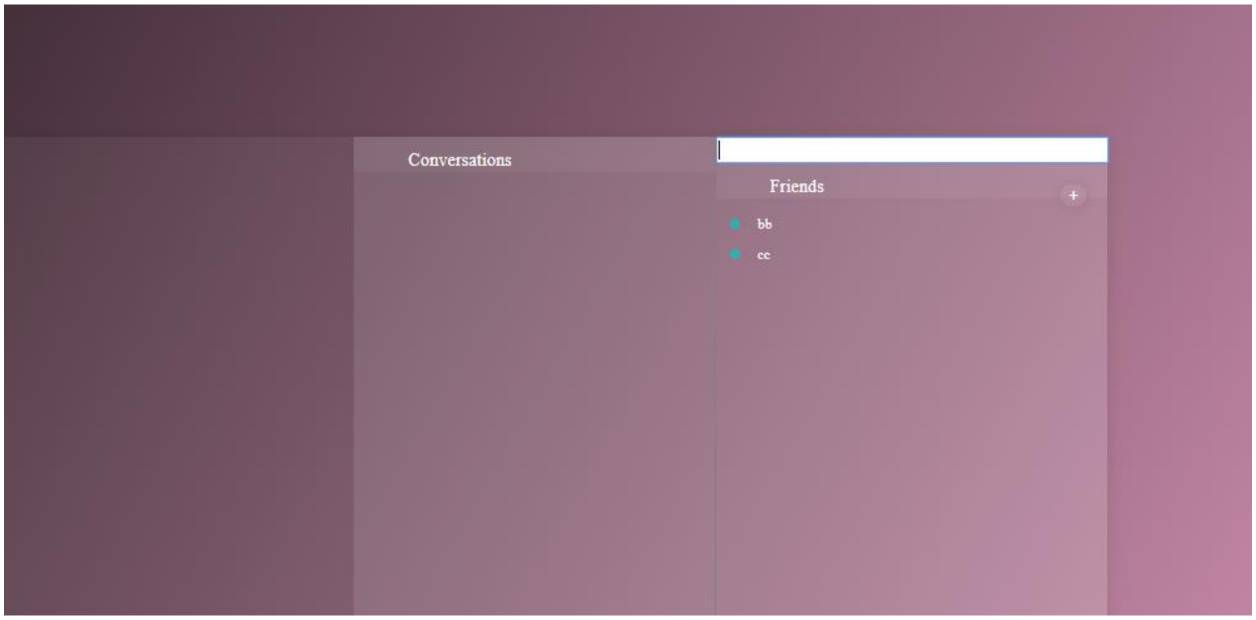


The application architecture is designed with 3 main layers, presentation layer, business layer and persistence layer. And based on the client-server nature of this application, presentation layer is located at client side, and for better performance, there is also a cache layer at client side. And the business layer and persistence layer is located at server side. And the detail of each layer is designed as below:

Presentation Layer: The presentation layer is basically the user interface in this application. It's written with Angular. Angular is a TypeScript-based open-source web application framework. At the end, it's based on javascript. And in this application the UI is designed as below:

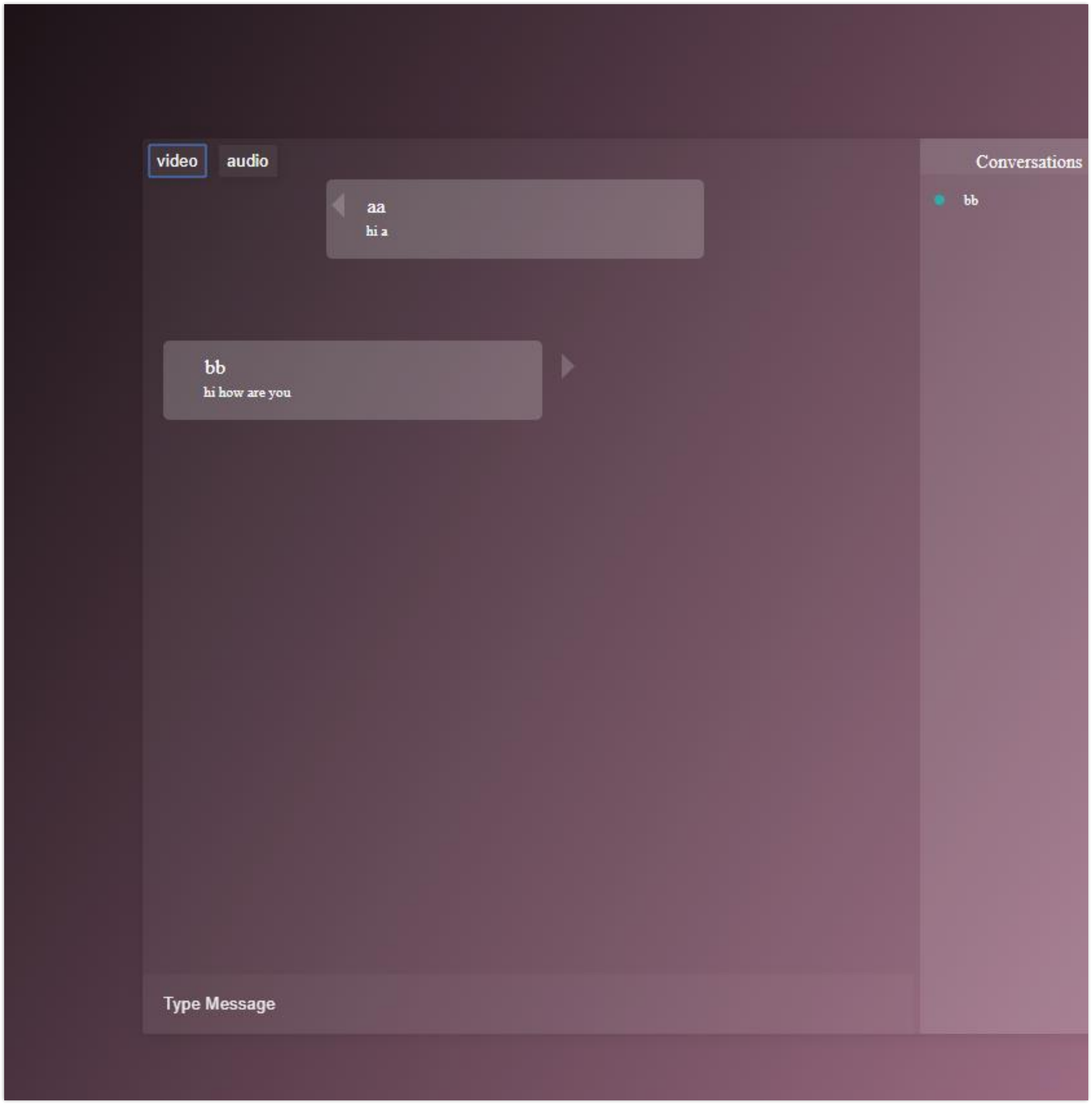


Landing page. Provide a form for user to put in user name and password to either log in or log on.



Once logged in, there will be 3 panels. First one shows all the conversations. Second one show all the friend list. The last one is the chatting page

In the conversations list, user can click any conversation to enter the conversation page.

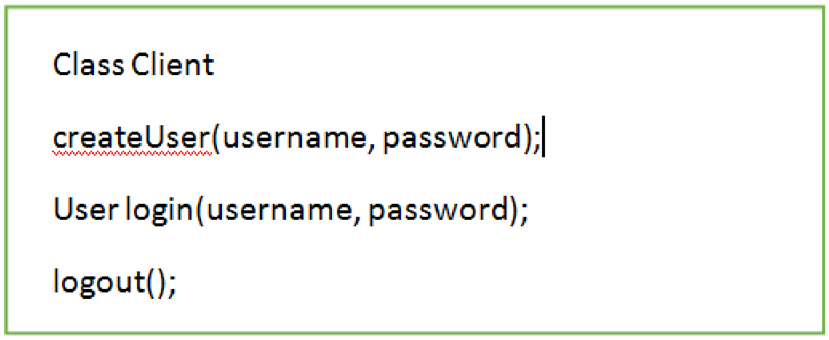


aa
hi a

bb
hi how are you

Cache Layer: we store the chat history in the cache layer in user’s machine unless user choose to upload it to server(to be implemented).

5. Application OOP Design

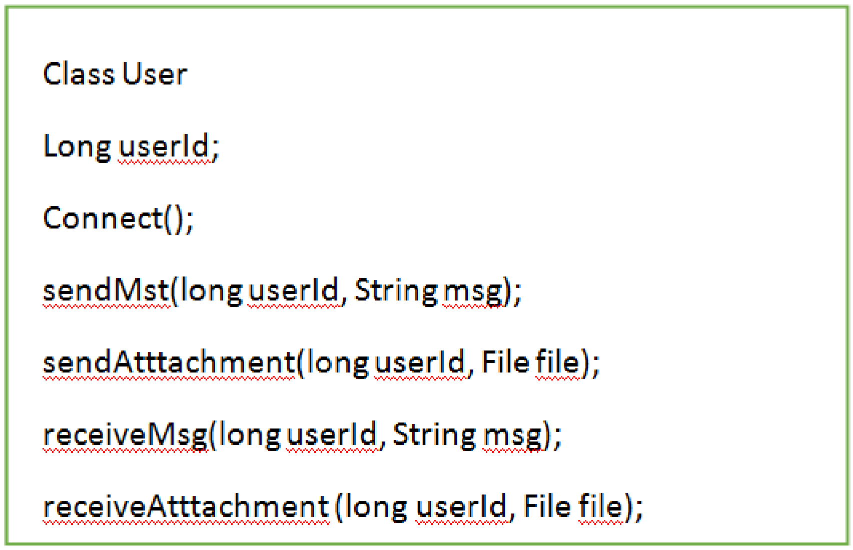


```
Class Client

createUser(username, password);

User login(username, password);

logout();
```



```
Class User

Long userId;

Connect();

sendMst(long userId, String msg);

sendAtttachment(long userId, File file);

receiveMsg(long userId, String msg);

receiveAtttachment (long userId, File file);
```

```
class Server

start();

connect(long userId);

sendMsg(long userId, String msg);

sendAttachment(long userId, File file);

boolean lookforUser(long userId);

storeToDb(long sender, long receiver, String msg);

storeToDb(long sender, long receiver, File file);

List<Object> fetchFromDb(long userId);
```

6. Database Designer

User

User id	Unique identifier
User Name	User choice
User password	Encrypted password

User_activity

User id	Unique identifier
Last active time	The last time

Conversation

sender	Sender’s user id
receiver	Receiver’s user id
timestamp	The time sender send the msg

Send timestamp	The time stamp receiver receives the msg

Friendship

User 1	Unique identifier for user1
User 2	Unique identifier for user2

7. Project Structure and Code

ServerSide

```
-ms-account-management
    -src.main.java.com.chat.msaccountmanagement
        -Controller.java
        -MsAccountManagementApplication.java
        -model
            -FriendShip.java
            -User.java
        -repository
            -UserRepository.java
            -UserDbRepository.java
        -service
            -AccountService.java
    -pom.xml
-ms-chat-application
    -src.main.java
        -MsChatApplicationMain.java
        -model
            -ChatMessage.java
        -controller
            -ChatController.java
            -WebSocketEventListener.java
        -config
            -WebSocketConfig.java
            -pom.xml

  ClientSide

  -app
      -app.component.css
      -app.component.html
      -app.component.ts
      -app.module.ts
      -chatList
              -chatlist.component.css
              -chatlist.component.html
              -chatlist.component.ts
      -chatPage
              - chat.component.css
              - chat.component.html
              - chat.component.ts
      -confirmation-dialog
              -confirmationDialog.component.css
              - confirmationDialog.component.html
              - confirmationDialog.component.ts
      -friendList
              -friendlist.component.css
              - friendlist.component.html
              - friendlist.component.ts
      -landingPage
              - landingPage.component.css
              - landingPage.component.html
              - landingPage.component.ts
      - mainPage
              - mainPage.component.css
              - mainPage.component.html
              - mainPage.component.ts
      -models
              -account.interface.ts
              -message.interface.ts
      -service
              -app.service.ts
      -store
              -app.reducer.ts
              -app.state.ts
```

```
-actions
        -account.actions.ts
        -messages.actions.ts
-effects
        -account.effects.ts
        -messages.effects.ts
-reducers
        -account.reducers.ts
        -messages.reducers.ts
-selectors
        -account.selectors.ts
        -messages.selectors.ts
```

8. Test Cases

Test Case Number:	TC1
Revision:	Rev. 1
Author:	Shuyang Zhu
Date Conducted:	4/10/2020
Test Conductor:	Shuyang Zhu
Customer Representative:	NA
Description:	create new account
Pre- Test Setup:	NA

User Action	Expected Result	P/F	Comment
User enter name/password, click create button	A pop up "created"	P	
User use name/password, click login	The friends and conversations view displayed	P	

Test Case Number:	TC2
Revision:	Rev. 1
Author:	Shuyang Zhu
Date Conducted:	4/10/2020
Test Conductor:	Shuyang Zhu
Customer Representative:	NA
Description:	sending message when both are online
Pre- Test Setup:	create and login to users

User Action	Expected Result	P/F	Comment
User a login	Conversation and friends view displayed	P	
User a create conversation with b	A new conversation created and displayed in conversation view	P	
User b login	Conversation and friends view displayed	P	
User a send 'hi' to user b	User b receive notification. Conversation view updated	P	
User b click conversation	Conversation page display with message from user a	P	

Test Case Number:	TC3
Revision:	Rev. 1
Author:	Shuyang Zhu
Date Conducted:	4/10/2020
Test Conductor:	Shuyang Zhu
Customer Representative:	NA
Description:	sending message when one is offline
Pre- Test Setup:	create 2 users, login with only one user

User Action	Expected Result	P/F	Comment
User a login	Conversation and friends view displayed	P	
User a open conversation with b	Chat page with User b displayed	P	
User a send ‘hi’ to user b	Stored to conversation table	p	
User b login	A notification pop up, Conversation view updated	p	
User b click conversation	Conversation page display with message from user a	P	

Test Case Number:	TC4
Revision:	Rev. 1
Author:	Shuyang Zhu
Date Conducted:	4/10/2020
Test Conductor:	Shuyang Zhu
Customer Representative:	NA
Description:	video chat
Pre- Test Setup:	create user account

User Action	Expected Result	P/F	Comment
User A login	Conversation and friends view displayed	P	
User A open conversation with b	Chat page with User b displayed	P	
User A click video button	Pop up appear on B's screen	P	
User B accept the video chat request	Screen show up with A	P	

Test Case Number:	TC5
Revision:	Rev. 1
Author:	Shuyang Zhu
Date Conducted:	4/10/2020
Test Conductor:	Shuyang Zhu
Customer Representative :	NA
Description:	audio chat
Pre- Test Setup:	create user account

User Action	Expected Result	P/F	Comment
User A login	Conversation and friends view displayed	P	
User A open conversation with b	Chat page with User b displayed	P	
User A click audio button	Pop up appear on B's screen	P	
User B accept the video chat request	A's sound appear	P	

9. Ethical and Social Effect

The technology of web services and communication infrastructure is growing rapidly now days. Chat messaging apps, such as WhatsUp, Skype, yahoo messenger, Blackberry messenger (BBM), have become a very important part of people's lives.

People can’t always be around with the ones they love because of many reasons, a good instant message application come to the rescue. The application implemented in this project can provide users the ability to instantly communicate with other. And communication means not only texting but also video/audio. Seeing and hearing the ones you love would be the moment of joy this application wants to bring.

10. Bibliography

[1] Malhotra, A., Sharma, V., Gandhi, P., & Purohit, N. (2010). UDP based chat application. *2010 2nd International Conference on Computer Engineering and Technology, 6*, V6-374-V6-377.

[2] Bamane, A., Bhoyar, P., Dugar, A., & Antony, L. (2012). Enhanced Chat Application.

[3] Singh, A., & Haahr, M. (2006). A Peer-to-Peer Reference Architecture. *2006 1st International Conference on Communication Systems Software & Middleware*, 1-10.

[4] Aberer, K., Alima, L.O., Ghodsi, A., Girdzijauskas, S., Haridi, S., & Hauswirth, M. (2005). The essence of P2P: a reference architecture for overlay networks. *Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, 11-20.

[5] Junio, O. M., & Chavez, E. P. (2018). Development of Offline Chat Application: Framework for Resilient Disaster Management. *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, Chongqing, China, 2018, pp. 510-514.

[6] Seguin, K. (2014). The little redis book. Retrieved from *https://github.com/karlseguin/the-little-redis-book*

[7] Chen, H., Wen, J., & Yang, C. (2014). A Secure End-to-End Mobile Chat Scheme. *2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications*, 472-477.

[8] Bremler-Barr, A., Dekel, O., Goldschmidt, R., & Levy, H. (2011). Controlling P2P Applications via Address Harvesting: The Skype Story. *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, Shanghai, 2011, pp. 1579-1586.