Course Code	:	SWE312
Course Name	:	Distributed System
Lecturer	:	Dr Burra Venkata Durga Kumar
Academic Session	:	201904
Assessment Title	:	Project
Submission Due Date	:	22 July 2019

Prepared by :

Student ID	Student Name
SWE1609512	CHIN JIAN HWA
SWE1609638	SIEW QI XUAN
SWE1609011	HU GUANGYU

Date Received :

Feedback from Lecturer:

Mark:

Own Work Declaration

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.

Signature:

CHIN JIAN HWA SIEW QI XUAN HU GUANGYU

Date: 22 July 2019

Contents

CloudBond

1. Introduction

Messaging is proven to be the communication channel of choice for decades. The chat application is a feature or a program on the Internet to communicate directly among Internet users. The chat applications allow users to communicate with each other from a great distance. With the fast development of the Internet, an increasing number of people choose online chatting tools for communication, such as online chat application. Traditional real-time chatting software is usually a desktop application program, and specific client programs are needed during application. On the other hand, a browser-based, real-time chatting application does not require any additional client program.

In this project, a browser-based chatting application, CloudBond, is built to bring personal touch for online communication. A chatting application using HTML5, CSS, and Spring framework technologies is presented in this paper. CloudBond targets the market of corporate organization, as one of the features is adding users on the server-side, instead of on the client-side. This feature aims for organizations which assign an account for its agent, such as Telecommunication Company. Permitting “adding users” only to organization provide better security and integrity of the chatting application. The report is organized as follows. Section 2 describes the background study of the system. Section 3 presents the key technologies and design features of this system. Section 4 reports the limitations in this system and possible future work for them. Section 5 concludes this paper.

2. Background

CloudBond is a real-time,web-based chatting application. It features a private chat room for the users. The application targets organization or corporate which assigns the user account for its employees. The application is called CloudBond because it motivates bonding on the cloud.

▲ Figure 2.1: CloudBond

An open-source application framework, Spring Framework, is used in this project. The framework is an inversion of control container for the Java platform. Any Java application can use the core features of the framework, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform. Although the framework does not impose any specific programming model, it is becoming a replacement for the Enterprise JavaBeans (EJB) model.

Spring Boot is a convention-over-configuration solution of Spring Framework for creating stand-alone, production-grade Spring-based Applications that can be run directly over a network. It is preconfigured with the best configuration in the opinion of the Spring's team. Using Spring platform and third-party libraries allow a developer to start developing with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

Justification on choosing Spring Framework for the project:

- Ability to create a stand-alone Spring application
- Tomcat server is embedded directly, so it does not need to deploy WAR files
- It provides opinionated 'starter' Project Object Models (POMs) to simplify the Maven configuration
- Automatically configure Spring whenever possible
- No code generation and no requirement for XML configuration

WebSocket is a communication protocol that enables the CloudBond to establish a two-way communication channel between a server and a client. WebSocket works by first establishing a regular HTTP connection with the server and then upgrading it to a bidirectional WebSocket connection by sending an Upgrade header. WebSocket is supported in most modern web browsers. For browsers that do not support it, the Spring Framework has libraries that provide fallbacks to other techniques like comet and long-polling.

Distributed system is a network structure allows resources sharing (grid, for instance). It provides users with a concurrent and integrated coherent network. Load balancing is a useful technique to distribute various network traffic to a backend server pool. It is a widely used technique in distributed system, since some web application (YouTube, for instance) they always needed to undertake huge user's traffic from all over the world, and it is impossible to deploy this kind of web application on only one server. Load balancing technique solve the problem quite well, it redirects massive user's traffic to different back-end server, scaling to meet demand of responding to handling hundreds of thousands concurrent requests. A load balancer always has the following features: 1) Distributed request (user traffic) from different client and network load across multiple server efficiently. 2) Ensure the high availability. Server sometimes will encounter some unexpected situations (power failure, server crushes, for instance), in order to ensure user on the client side can always access to the server side, load balancer will check each back-end server and send requests only to servers that are online. 3) Load balancer provides the high flexibility to add or remove server on the clustering. If user's traffic becomes more massive and more servers are needed to handle those requests, this can be done in convenient way (register to load balancer). But we fail to achieve several aims in our project by implementing load balancing, such as failure of synchronizing database, didn't certificate the load balancer server, which will all be described into more detail in **Section 4**.

3. Design Features

In the system, the spring framework is used and there are three pages which are admin page, login page and chat room page. The user authentication function will be performed at the login page; the user management function will be performed at the admin page; the chat function will be performed at the chat room page.

3.1. Entity and Repository

For user authentication and user management, an entity and a repository class are needed in the spring framework to manage the user data. An entity class is annotated as *@Entity*, and has a special communication with database. The database can be specified in the *application.properties* but the codes for specifying the database have been commented out due to the database is not necessary in the system.

```
#spring.jpa.hibernate.ddl-auto=create
#spring.datasource.url=jdbc:mysql://localhost:3306/db_example
#spring.datasource.username=springuser
#spring.datasource.password=ThePassword

spring.mvc.view.prefix: /WEB-INF/jsp/
spring.mvc.view.suffix: .jsp
```

▲ Code 3.1.1: application.properties

An *User.java* entity class has been designed for the system. There are some attributes inside the class, include an integer, two strings and a boolean named id, userName, password and isOnline. A constructor is needed for creating a new user. The *toString()* method will print out the user's properties.

```
package com.example.websocketdemo.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class User {

   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
   private Integer id;
   private String userName;
   private String password;
   private boolean isOnline;

   public User() {
   }

   public User(String userName, String password) {
       this.userName = userName;
       this.password = password;
       this.isOnline = false;
   }

   @Override
   public String toString() {
       return String.format(
               "User[id=%d, userName='%s', password='%s']",
               id, userName, password);
   }

   public String toLogin() {
       return String.format( "%d", id );
   }

//Geters
   public Integer getId() {
       return id;
   }

   public String getUserName() {
       return userName;
   }

   public String getPassword() {
       return password;
   }

   public void online() {
       this.isOnline = true;
   }

   public void offline() {
       this.isOnline = false;
   }

   public boolean isOnline() {
       return isOnline;
   }
}
```

▲ Code 3.1.2: User.java

A *UserRepository.java* interface is created to work with user entity class. The repository will be auto implemented by Spring into a Bean named *userRepository*. *UserRepository.java* extends the CrudRepository interface and the type of entity and ID which are User and Integer are specified in the generic parameters on CrudRepository. Query methods can be simply declaring by their method signature in spring framework. The *findByUserName()* and *findAll()* methods are declared.

```
package com.example.websocketdemo.repository;

import org.springframework.data.repository.CrudRepository;

import com.example.websocketdemo.entity.User;

import java.util.List;

// This will be AUTO IMPLEMENTED by Spring into a Bean called userRepository
// CRUD refers Create, Read, Update, Delete

public interface UserRepository extends CrudRepository<User, Integer> {

   List<User> findByUserName(String userName);
   List<User> findAll();
}
```

▲ Code 3.1.3: UserRepository.java

3.2. User Authentication

After entity and repository have been created, the *LoginController.java* class can be created to perform the user authentication. The class is annotated with *@Controller* that means this class is a controller and it is the controller of *login.jsp* page. In this controller class, there are a *UserRepository* parameter which can be used to handle the user data and two methods which are *index()* and *logIn()*. The *index()* method is to redirect to admin page if the application is open in localhost. The *logIn() method* will check the username and password, if the username and password do not match, the failed message will be passed to login view by using *ModelAndView*.

```
package com.example.websocketdemo.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import com.example.websocketdemo.entity.User;
import com.example.websocketdemo.repository.UserRepository;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.net.UnknownHostException;
import java.util.List;

@Controller    // This means that this class is a Controller
@RequestMapping(path="/login") // This means URL's start with /login (after Application path)
public class LoginController {
   @Autowired // This means to get the bean called userRepository
   // Which is auto-generated by Spring, we will use it to handle the data
   private UserRepository userRepository;

   @GetMapping // Map ONLY GET Requests
   public ModelAndView index(HttpServletRequest request, HttpServletResponse response) throws IOException {
       ModelAndView modelAndView = new ModelAndView("/login");
       String clientAddr = request.getLocalAddr();
       String localAddr1 = "127.0.0.1";
       String localAddr2 = "0:0:0:0:0:0:0:1";
       System.out.println(clientAddr);
       if (clientAddr.equals(localAddr1)|| clientAddr.equals(localAddr2)){
           return new ModelAndView("redirect:/admin");
       }
       return modelAndView;
   }
```

```
    @PostMapping(path="/a") // Map ONLY GET Requests
    public ModelAndView logIn (@RequestParam String userName
            , @RequestParam String password) {
        // @ResponseBody means the returned String is the response, not a view name
        // @RequestParam means it is a parameter from the GET or POST request

        ModelAndView modelAndView = new ModelAndView("/login");

        List<User> userList = userRepository.findByUserName(userName);
        if(userList.size() == 1){
            if(userList.get(0).getPassword().equals(password))  {
                if(userList.get(0).isOnline() == false){
                    modelAndView.setViewName("/chatroom");
                    modelAndView.addObject("username", userName);
                    userList.get(0).online();
                } else {
                    modelAndView.addObject("message", "User is Online");
                }
            } else {
                modelAndView.addObject("message", "Wrong Password");
            }
        } else {
            modelAndView.addObject("message", "Wrong User");
        }

        System.out.println(modelAndView.getViewName());
        return modelAndView;
    }
}
```

▲ Code 3.2.1: LoginController.java

3.3. User Management

For user management, *AdminController.java* controller class has been created which is the controller of *admin.jsp* page. In this class, there are three methods which are *index(), deleteUser()* and *addUser()*. The *index()* method is used to get the users list for displaying at the *admin.jsp* page and to prevent normal users access to *admin.jsp* page. The *deleteUser()* method is user to remove a user from the user repository. The *addUser()* method is used to add a new user to the user repository.

```
package com.example.websocketdemo.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import com.example.websocketdemo.entity.User;
import com.example.websocketdemo.repository.UserRepository;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.net.Inet4Address;
import java.net.UnknownHostException;
import java.util.List;

@Controller    // This means that this class is a Controller
```

```java
@RequestMapping(path="/admin") // This means URL's start with /demo (after Application path)
public class AdminController {
   @Autowired // This means to get the bean called userRepository
   // Which is auto-generated by Spring, we will use it to handle the data
   private UserRepository userRepository;

   @GetMapping // Map ONLY GET Requests
   public ModelAndView index(HttpServletRequest request, HttpServletResponse response) throws IOException {
       ModelAndView modelAndView = new ModelAndView("/admin");
       String clientAddr = request.getLocalAddr();
       String localAddr1 = "127.0.0.1";
       String localAddr2 = "0:0:0:0:0:0:0:1";
       System.out.println(clientAddr);
       if (clientAddr.equals(localAddr1)|| clientAddr.equals(localAddr2)){
           List<User> userList = userRepository.findAll();
           modelAndView.addObject("userList", userList);
       } else {
//            return new ModelAndView("redirect:/login");
       }
       return modelAndView;
   }

   @PostMapping(path="/deleteUser") // Map ONLY POST Requests
   public ModelAndView deleteUser(@RequestParam String userName){
       ModelAndView modelAndView = new ModelAndView("/admin");

       List<User> userList = userRepository.findByUserName(userName);
       userRepository.delete(userList.get(0));

       userList = userRepository.findAll();
       modelAndView.addObject("userList", userList);
       return modelAndView;
   }

   @PostMapping(path="/add") // Map ONLY POST Requests
   public ModelAndView AddNewUser (@RequestParam String userName
           , @RequestParam String password) {
       // @ResponseBody means the returned String is the response, not a view name
       // @RequestParam means it is a parameter from the GET or POST request

       ModelAndView modelAndView = new ModelAndView("/admin");

       User newUser = new User(userName,password);

       List<User> userList = userRepository.findByUserName(userName);
       if(userList.size() == 0){
           userRepository.save(newUser);
       }

       userList = userRepository.findAll();
       modelAndView.addObject("userList", userList);
       return modelAndView;
   }
}
```

▲ Code 3.3.1: AdminController.java

3.4. Chat

In the chat function, WebSocket is used, it is a communication protocol that build a two-way communication channel between server and client. Therefore, the first step is to create a *config* package and create class called WebsocketConfig inside the package. To enable the WebSocket server, the annotation *@EnableWebSocketMessageBroker* and the implementation

WebSocketMessageBrokerConfigurer are needed in the class. In the *registerStompEndpoints()* method, websocket endpoint is registered so that clients can use it to connect to the server. The configureMessageBroker() method is to configure a message broker that will be used to route messages from one client to another.

```
package com.example.websocketdemo.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import org.springframework.web.socket.config.annotation.*;

@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

   @Override
   public void registerStompEndpoints(StompEndpointRegistry registry) {
       registry.addEndpoint("/ws").withSockJS();
   }

   @Override
   public void configureMessageBroker(MessageBrokerRegistry registry) {
       registry.setApplicationDestinationPrefixes("/app");
       registry.enableSimpleBroker("/topic");
   }
}
```

▲ Code 3.4.1: WebSocketConfig.java

A message payload in needed which can be exchanged between the clients and the server. Therefore, *ChatMessage.java* class has been created inside *model* package.

```
package com.example.websocketdemo.model;

public class ChatMessage {
   private MessageType type;
   private String content;
   private String sender;

   public enum MessageType {
       CHAT,
       JOIN,
       LEAVE
   }

   public MessageType getType() {
       return type;
   }

   public void setType(MessageType type) {
       this.type = type;
   }

   public String getContent() {
       return content;
   }

   public void setContent(String content) {
       this.content = content;
   }
```

```java
    public String getSender() {
        return sender;
    }

    public void setSender(String sender) {
        this.sender = sender;
    }
}
```

▲ Code 3.4.2: ChatMessage.java

After that, *ChatController.java* class is created inside the *controller* package. This controller class is the controller of *chatroom.jsp* page for receiving messages and broadcasting to others.

```java
package com.example.websocketdemo.controller;

import com.example.websocketdemo.model.ChatMessage;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.messaging.handler.annotation.Payload;
import org.springframework.messaging.handler.annotation.SendTo;
import org.springframework.messaging.simp.SimpMessageHeaderAccessor;
import org.springframework.stereotype.Controller;

@Controller
public class ChatController {

   @MessageMapping("/chat.sendMessage")
   @SendTo("/topic/public")
   public ChatMessage sendMessage(@Payload ChatMessage chatMessage) {
       return chatMessage;
   }

   @MessageMapping("/chat.addUser")
   @SendTo("/topic/public")
   public ChatMessage addUser(@Payload ChatMessage chatMessage,
                              SimpMessageHeaderAccessor headerAccessor) {
       // Add username in web socket session
       headerAccessor.getSessionAttributes().put("username", chatMessage.getSender());
       return chatMessage;
   }

}
```

▲ Code 3.4.3: ChatController.java

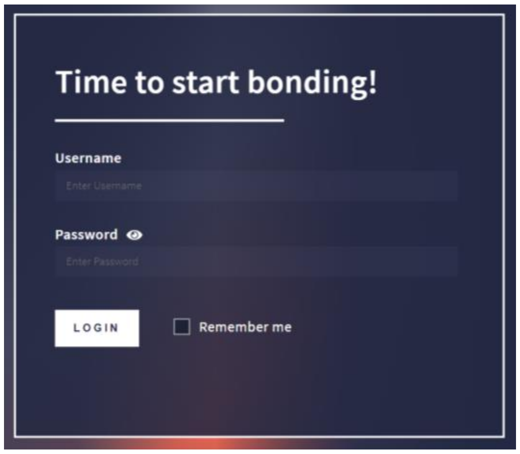
WebSocketEventListener.java class inside controller package is to listen for socket connect and disconnect event and broadcast the log messages.

```java
package com.example.websocketdemo.controller;

import com.example.websocketdemo.entity.User;
import com.example.websocketdemo.repository.UserRepository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.event.EventListener;
import org.springframework.messaging.simp.SimpMessageSendingOperations;
import org.springframework.messaging.simp.stomp.StompHeaderAccessor;
import org.springframework.stereotype.Component;
import org.springframework.web.socket.messaging.SessionConnectedEvent;
import org.springframework.web.socket.messaging.SessionDisconnectEvent;

import com.example.websocketdemo.model.ChatMessage;

import java.util.List;

@Component
public class WebSocketEventListener {

   private static final Logger logger = LoggerFactory.getLogger(WebSocketEventListener.class);


   @Autowired
   private SimpMessageSendingOperations messagingTemplate;
   @Autowired
   private UserRepository userRepository;

   @EventListener
   public void handleWebSocketConnectListener(SessionConnectedEvent event) {
       logger.info("Received a new web socket connection");
   }

   @EventListener
   public void handleWebSocketDisconnectListener(SessionDisconnectEvent event) {
       StompHeaderAccessor headerAccessor = StompHeaderAccessor.wrap(event.getMessage());

       String username = (String) headerAccessor.getSessionAttributes().get("username");
       if(username != null) {
           logger.info("User Disconnected : " + username);

           ChatMessage chatMessage = new ChatMessage();
           chatMessage.setType(ChatMessage.MessageType.LEAVE);
           chatMessage.setSender(username);

           List<User> userList = userRepository.findByUserName(username);
           if(userList.size() == 1){
               userList.get(0).offline();
           }

           messagingTemplate.convertAndSend("/topic/public", chatMessage);
       } else {
           System.out.println("Username is null.");
       }
   }
}
```

▲ Code 3.4.4: WebSocketEventListener.java

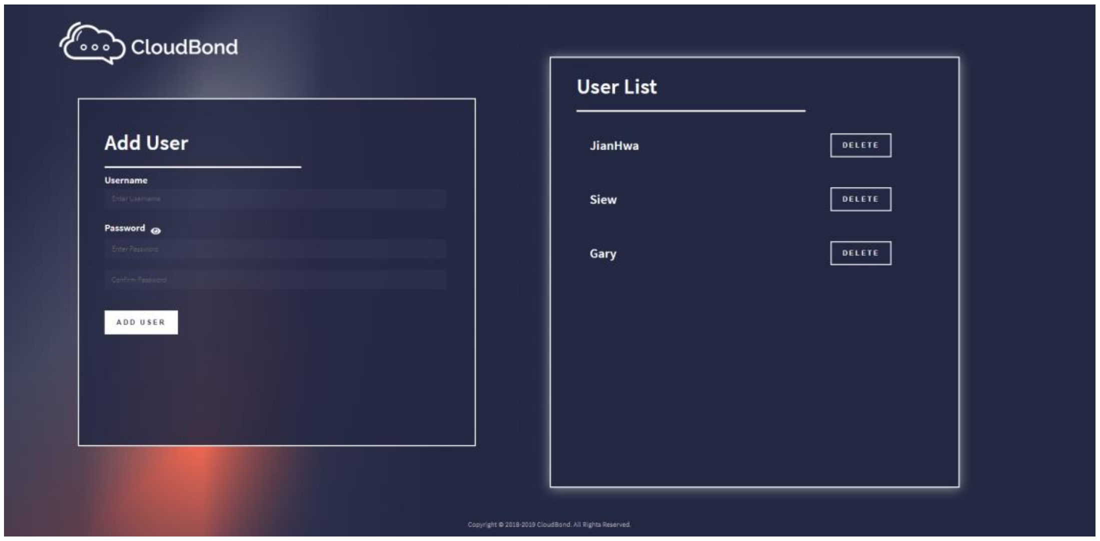
3.5. Front-end

There are three main pages which are *login.jsp*, *admin.jsp* and *chatroom.jsp.* Javascript named *main.js* is required for communicating to the WebSocket. Finally, *main.css* and *generic2.css* files for styling also has been built. All code of front-end pages was shown in appendix. In the *login.jsp* page, user was able to login with the username and password that were provided by admin.



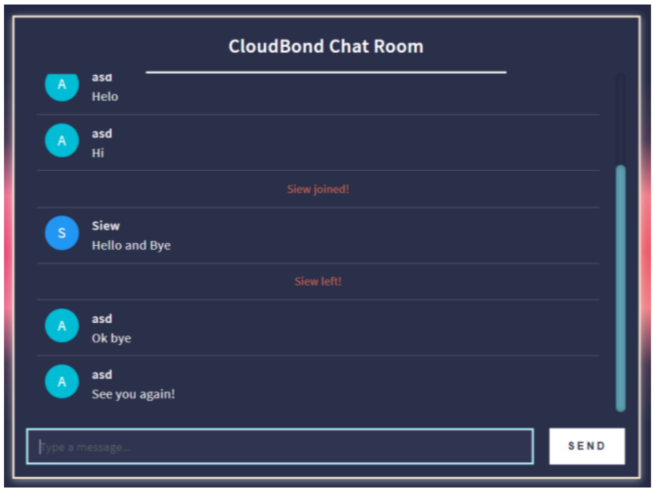
▲ Figure 3.5.1: Login Pop-up

In *admin.jsp* page, only admin can access into it. Admin can add and delete users in this page. All normal users was not able to access this page.



▲ Figure 3.5.2: Admin Page

In the chatroom page, all of the users who has been login successfully will enter the same chatroom. They can send messages and receive messages from other users.


CloudBond Chat Room

asd
Helo

asd
Hi

Siew joined!

Siew
Hello and Bye

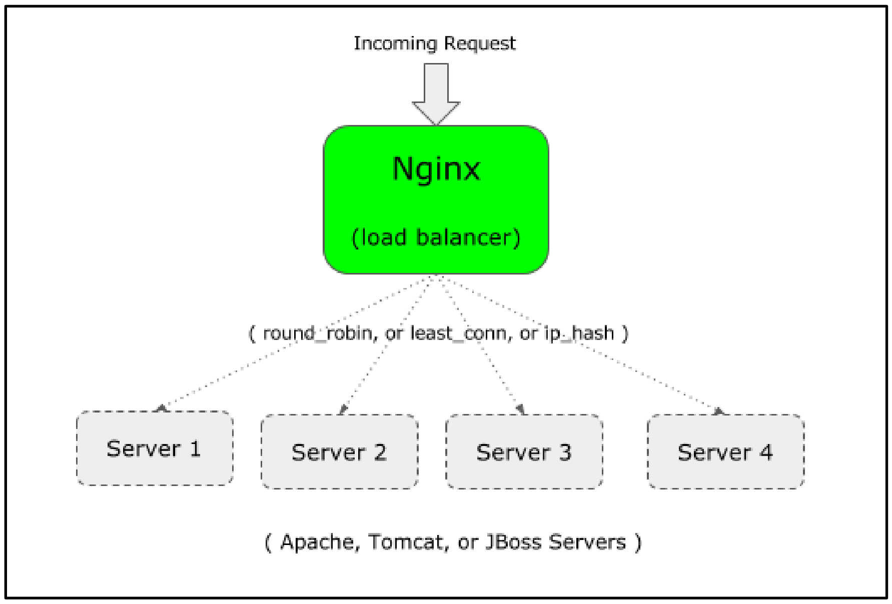
Siew left!

asd
Ok bye

asd
See you again!

▲ Figure 3.5.3: Chatroom Page

3.6. Distributed Architecture


Incoming Request
Nginx
(load balancer)
(round_robin, or least_conn, or ip_hash)
Server 1
Server 2
Server 3
Server 4
(Apache, Tomcat, or JBoss Servers)

▲ Figure 3.6.1: Nginx Proxy Server

The distributed architecture of our project is using load balancing. Nginx is a proxy server, which plays the role of load balancer, assuming this ip address is 192.123.145.10. Server 1, 2, 3 and 4 are the server that runs the web application, assuming their ip address is 23.128.112.0, 23.128.128.0, 23.128.144.0 and 23.128.176.0. Assume domain name of 192.123.145.10 (nginx server) is XMUM.com. When user access to XMUM.com, the nginx server will begin to redirect user to 23.128.112.0/23.128.128.0/23.128.144.0/23.128.176.0 by using round-robin technique (there are also have least connecting technique and ip hash checking technique, but the technique used in our project is round robin technique). Server 1, 2, 3 and 4 each of them has a weight value, the larger of weight value means the higher possibility to be the destination server after redirecting (by nginx).

4. Limitations

4.1. System & Implementation Limitations

The limitation of the project can be listed down into these several points:

- The web project is not deployed on various server, the project is deployed in local address (127.0.0.1) and various ports (for example, 127.0.0.1:8888). This due to the spring boot project after packaging to .jar file fails to run on server (tried it in the docker).
- It is difficult to synchronize the database (mysql, for instance) in a clustering structure. Server in the clustering should be able to synchronize all the data in the clustering structure, since the web application does not run on only one server. The project does not successfully synchronize all the database in different server.
- Security issues. SSL encryption is not used for the web application, and it still uses “Http” instead of using “Https” in the address. It is easily to be eavesdropped when user transmits their data on our online chat room.
- The online chat room project does not support file transmission feature yet. The file transmission feature involved database synchronization and transmission encryption, which are both not successfully being deployed on our project.
- The video call and voice call are both not supported yet. The video call feature involved handling the large traffic at the back-end server side. As the throughput is high, the function is not yet deployed.

4.2. Improvement & Future work

- More polished user interface
- File transmission
- Video call & Voice call

5. Conclusion

Our java web application (CloudBond) is a web-based online chat room, implemented by java spring boot framework. We also deploy a simple distributed system, which is load balance, on it. This web application has some advantages including fast responding, quick development, resource sharing and high scalability. The communication technique we used is Java Websocket, it can establish a communication channel between server and client.

The key design feature can be divided into three parts, which are admin page, user login page and chat room page. The structure we employ is MVC structure, which is a popular structure in web application development. MVC structure represents Model, View and Controller. View layer is constructed by .jsp, .css and .js file, which are directly interact with users. Controller layer is responsible for handling user request (such as input, click button). Model layer is for encapsulating the specific method and algorithm, it does not depend on any views or controllers. MVC structure works quite well on our web application (CloudBond).

However, our application still contains some drawbacks and limitations. Some features such as video chat and voice chat are not supported yet. With regard to the distributed deployment, our application also lack high security feature, which can encrypt the messages on the transmission channel. Although we use load balancing technique, clustering features are not completed. For instance, database synchronization also not successfully being employed, though it is a clustering system. There are still many space for us to develop this application further, and we are planning to add more good features on it in the future.

References

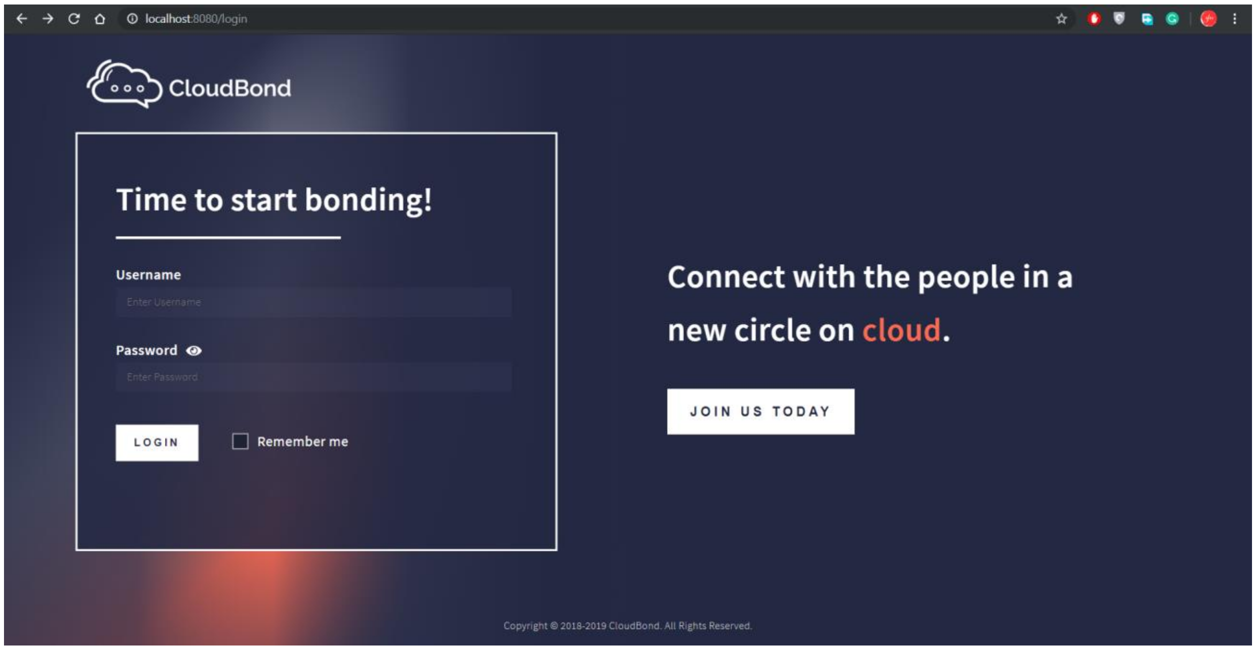
Hashavit, A., Tepper, N., Ronen, I., Leiba, L., & Cohen, A. (2018). Implicit User Modeling in Group Chat. *Adjunct Publication Of The 26Th Conference On User Modeling, Adaptation And Personalization - UMAP '18*. doi: 10.1145/3213586.3225236

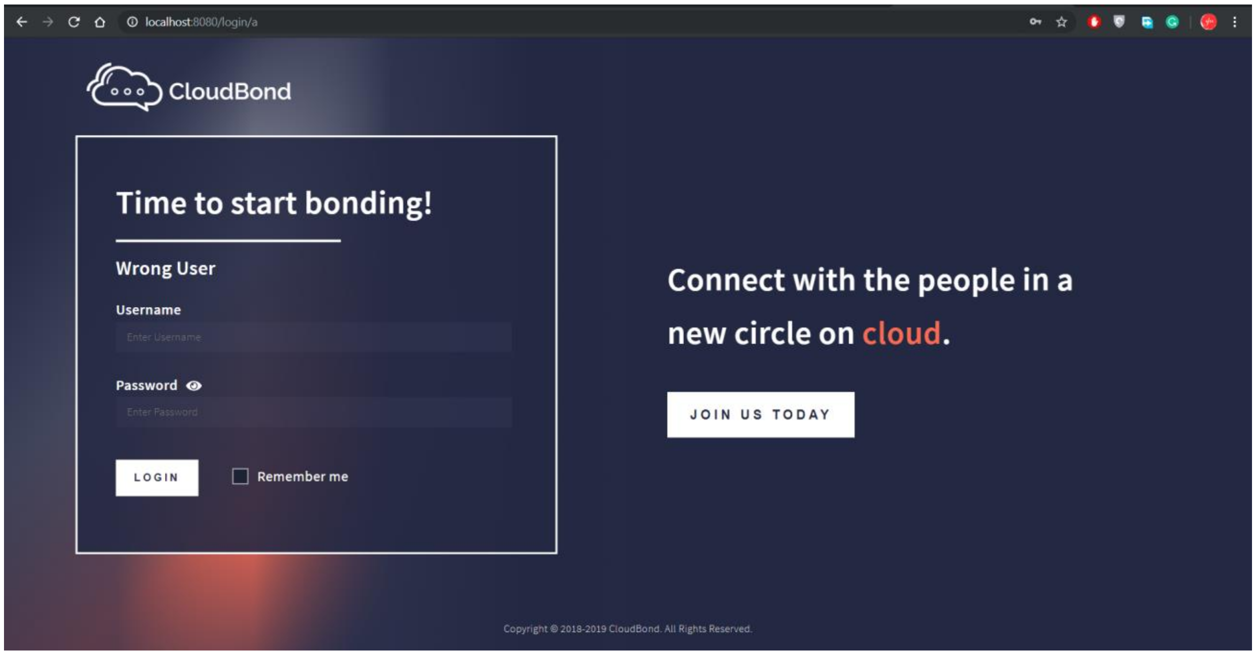
Henriyan, D., Devie Pratama Subiyanti, Fauzian, R., Anggraini, D., Vicky Ghani Aziz, M., & Ary Setijadi Prihatmanto. (2016). Design and implementation of web based real time chat interfacing server. *2016 6Th International Conference On System Engineering And Technology (ICSET)*. doi: 10.1109/icsengt.2016.7849628

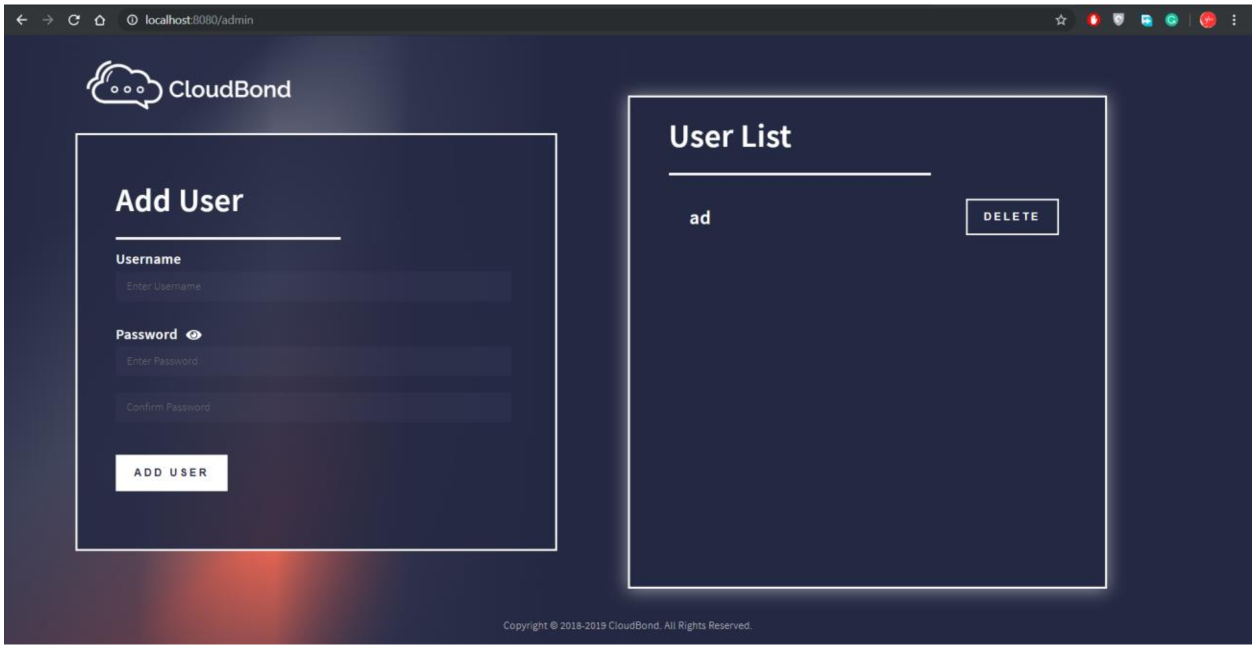
Yuzhuo, S., & Kun, H. (2013). Design and realization of chatting tool based on web. *2013 3Rd International Conference On Consumer Electronics, Communications And Networks*. doi: 10.1109/cecnet.2013.6703312

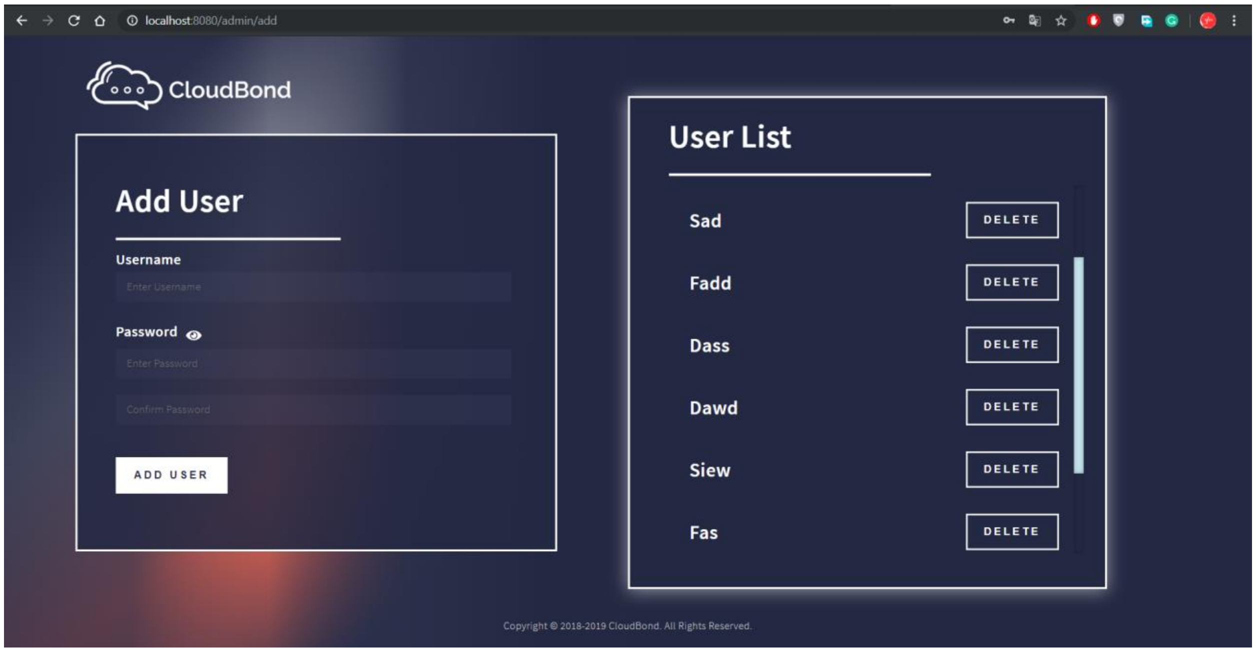
Appendix

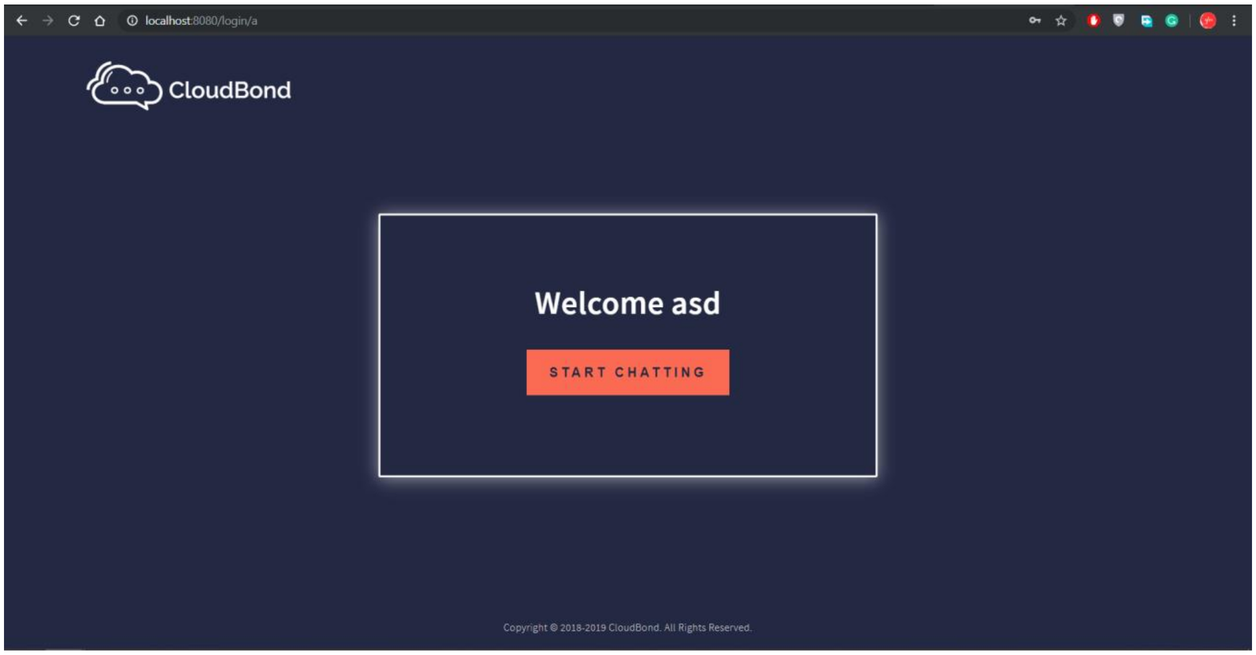
Screenshots of CloudBond

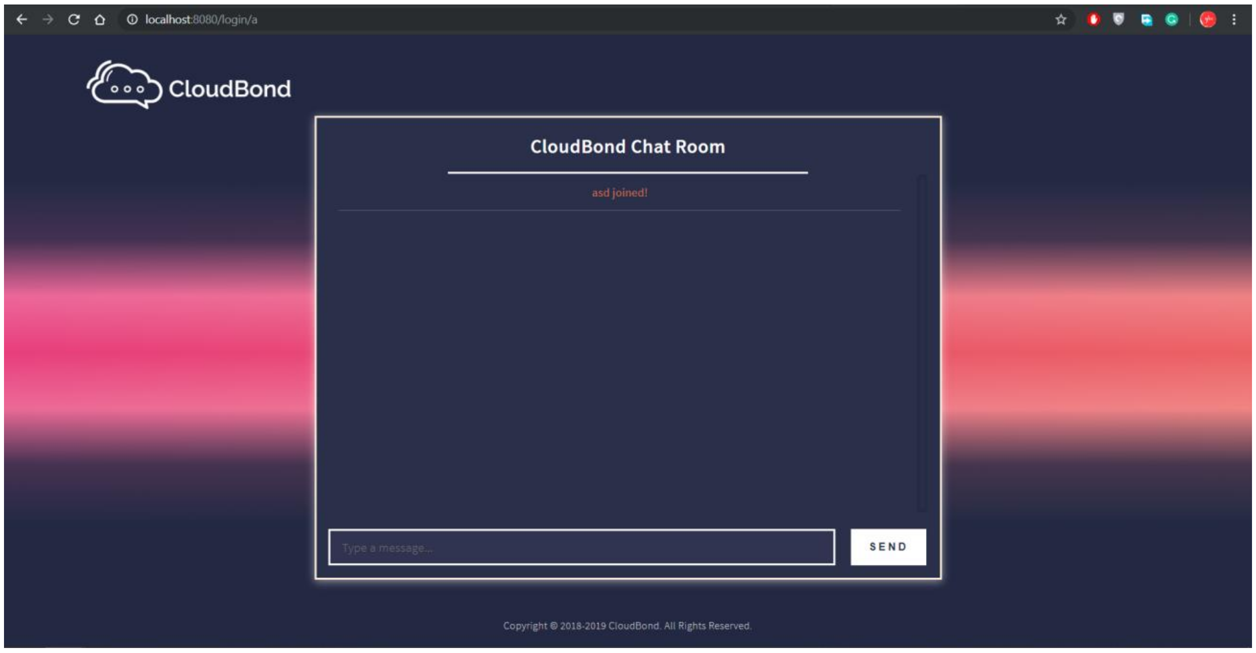


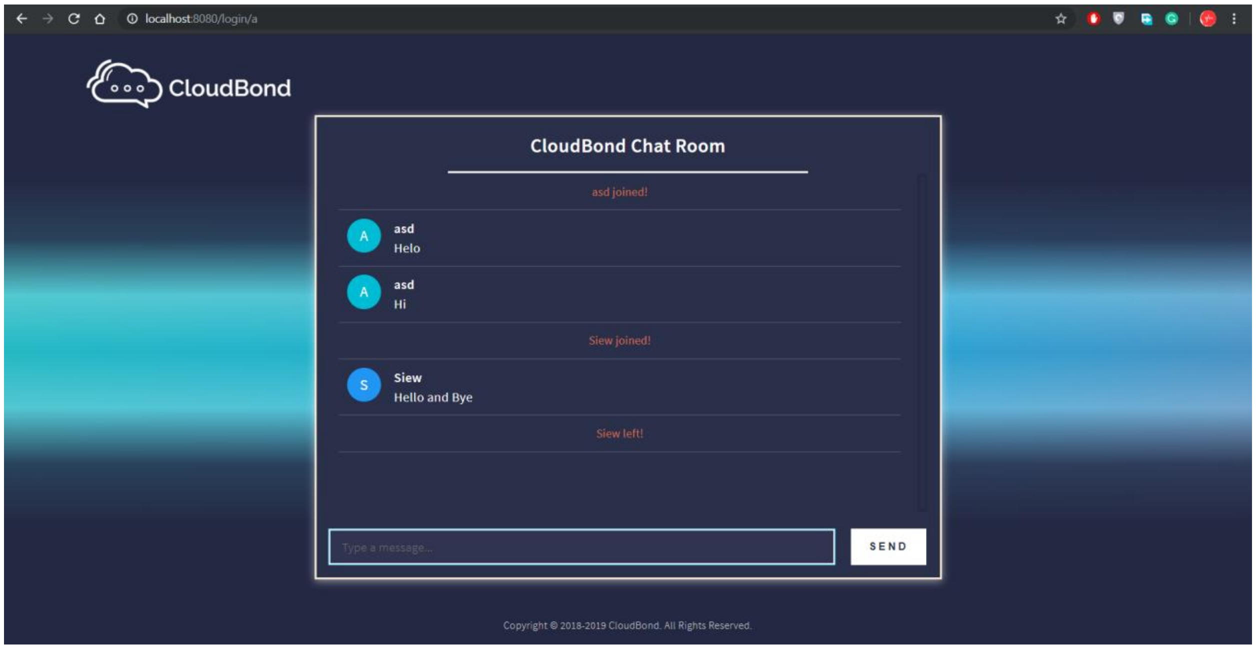



localhost:8080/admin
CloudBond
Add User
Username
Enter Username
Password
Enter Password
Confirm Password
ADD USER
User List
ad
DELETE
Copyright © 2018-2019 CloudBond. All Rights Reserved.


localhost:8080/admin/add
CloudBond
Add User
Username
Enter Username
Password
Enter Password
Confirm Password
ADD USER
User List
Sad
DELETE
Fadd
DELETE
Dass
DELETE
Dawd
DELETE
Siew
DELETE
Fas
DELETE
Copyright © 2018-2019 CloudBond. All Rights Reserved.


localhost:8080/login/a
CloudBond
Welcome asd
START CHATTING
Copyright © 2018-2019 CloudBond. All Rights Reserved.


localhost:8080/login/a
CloudBond
CloudBond Chat Room
asd joined!
Type a message...
SEND
Copyright © 2018-2019 CloudBond. All Rights Reserved.


CloudBond Chat Room

asd joined!

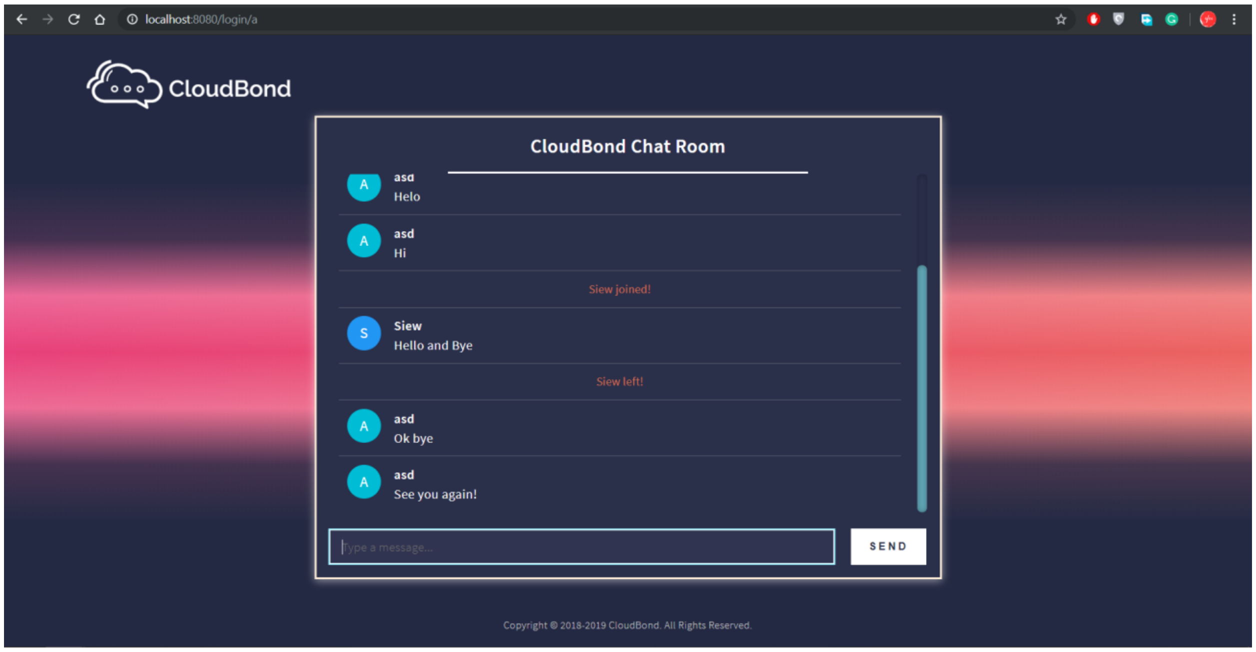
asd
Helo

asd
Hi

Siew joined!

Siew
Hello and Bye

Siew left!


CloudBond Chat Room

asd
Helo

asd
Hi

Siew joined!

Siew
Hello and Bye

Siew left!

asd
Ok bye

asd
See you again!

admin.jsp

```
<%@ page import="org.springframework.web.servlet.ModelAndView" %>
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <title>CloudBond - Login</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" type="text/css" media="screen" href="/css/generic2.css" />
   <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600" rel="stylesheet">
</head>
<body>


<section id="logGrid">
   <div class="logLeft">
       <div class="logBlur"></div>

       <div  id="signUp" class="logWrap">

       </div>

       <div id="signIn" class="logWrap2">

           <form class="animate" action="/login/a" method="post" id="LoginServlet">

               <div class="logPop in">
                   <h1>Time to start bonding!</h1>
                   <h2> ${message} </h2>
                   <br>
                   <label><b>Username</b></label>
                   <input type="text" placeholder="Enter Username" name="userName" required>
                   <br>
                   <label><b>Password</b></label>
                   <button type="button" onclick = "showPass()" class="see">
                       <svg xmlns="http://www.w3.org/2000/svg" aria-hidden="true" data-prefix="fas" data-icon="eye"
class="svg-eye" role="img" viewBox="0 0 576 512"><path fill="currentColor" d="M569.354 231.631C512.969 135.949 407.81 72 288
72 168.14 72 63.004 135.994 6.646 231.631a47.999 47.999 0 0 0 0 48.739C63.031 376.051 168.19 440 288 440c119.86 0 224.996-
63.994 281.354-159.631a47.997 47.997 0 0 0 0-48.738zM288 392c-75.162 0-136-60.827-136-136 0-75.162 60.826-136 136-136 75.162
0 136 60.826 136 136 0 75.162-60.826 136-136 136zm104-136c0 57.438-46.562 104-104 104s-104-46.562-104-104c0-17.708 4.431-
34.379 12.236-48.9731-.001.032c0 23.651 19.173 42.823 42.824 42.823s42.824-19.173 42.824-42.823c0-23.651-19.173-42.824-
42.824-42.8241-.032.001C253.621 156.431 270.292 152 288 152c57.438 0 104 46.562 104 104z"/></svg>
                   </button>
                   <input id="pass3" type="password" placeholder="Enter Password" name="password" required>
                   <br><br>
                   <button class="white" type="submit">Login</button>
                   <label>
                       <input type="checkbox" name="remember"> Remember me
                   </label>
               </div>
           </form>
       </div>
   </div>

   <div class="logRight">
       <h1>Connect with the people in a new circle on <span id="red">
         cloud</span>.
       </h1><br><br>
       <div class="logBox">
           <button onclick="showPop()" class="white2">
               Join us today
           </button>
       </div>
   </div>
</section>

<header id="main-header" class="log">
```

```
   <div class="logHeader2">
       <img src="/DSLOGO.png" alt="" class="CBlogo">
   </div>
</header>

<footer id="logFooter">
   <div class="footerWrapper">
       <div class="footer-social-list">
           <p>Copyright &copy; 2018-2019 CloudBond. All Rights Reserved.</p>
       </div>
   </div>
</footer>
<script src="/js/main.js"></script>
</body>
</html>
```

admin.jsp

```
<%@ page import="java.util.List" %>
<%@ page import="com.example.websocketdemo.entity.User" %>
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <title>CloudBond - Server Interface</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" type="text/css" media="screen" href="/css/main.css">
   <link rel="stylesheet" type="text/css" media="screen" href="/css/generic2.css" />
   <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600" rel="stylesheet">
</head>
<body>


<section id="logGrid">
   <div class="logLeft">
       <div class="logBlur"></div>

       <div class="logWrap2">
           <form class="animate" method="post" action="/admin/add" id="RegisterServlet">

               <div class="logPop in">
                   <h1>Add User</h1>
                   <label><b>Username</b></label>

                   <input type="text" placeholder="Enter Username" name="userName" required>
                   <br>
                   <label><b>Password</b></label>
                   <button type="button" onclick = "showPass()" class="see">
                       <svg xmlns="http://www.w3.org/2000/svg" aria-hidden="true" data-prefix="fas" data-icon="eye"
class="svg-eye" role="img" viewBox="0 0 576 512"><path fill="currentColor" d="M569.354 231.631C512.969 135.949 407.81 72 288
72 168.14 72 63.004 135.994 6.646 231.631a47.999 47.999 0 0 0 0 48.739C63.031 376.051 168.19 440 288 440c119.86 0 224.996-
63.994 281.354-159.631a47.997 47.997 0 0 0 0-48.738zM288 392c-75.162 0-136-60.827-136-136 0-75.162 60.826-136 136-136 75.162
0 136 60.826 136 136 0 75.162-60.826 136-136 136zm104-136c0 57.438-46.562 104-104 104s-104-46.562-104-104c0-17.708 4.431-
34.379 12.236-48.9731-.001.032c0 23.651 19.173 42.823 42.824 42.823s42.824-19.173 42.824-42.823c0-23.651-19.173-42.824-
42.824-42.8241-.032.001C253.621 156.431 270.292 152 288 152c57.438 0 104 46.562 104 104z"/></svg>
                   </button>

                   <input id="Pass" type="password" placeholder="Enter Password" name="password" required
oninput="checkUp(this)">
                   <br>
                   <input id="pass2" type="password" placeholder="Confirm Password" name="password2" required
oninput="checkDown(this)">
                   <span class ="red" id="invalid"></span>
                   <br><br>
                   <button class="white" type="submit">Add user</button>
                   <br><br>
               </div>
           </form>
       </div>

       <div id="signIn" class="logWrap">
       </div>
   </div>
   </div>

   <div class="logRight2">
       <h1>User List</h1>

       <table id="uList" class="style-2">
           <%
       List<User> userList = (List<User>)request.getAttribute("userList");
               System.out.println("UserList: " +userList.toString());
       for(User currentUser : userList){
           String username = currentUser.getUserName();
```

```
            %>
            <form method="post" action="/admin/deleteUser" >
                <tr style="height:10vh">
                    <td>
                        <div style="width: 22vw; font-weight: bold; font-size: 1.5em; padding-left: 1em;" >
                            <%=username%><input name="userName" value="<%=username%>" style="display:none">
                        </div>
                    </td>
                    <td><button type="submit">Delete</button></td>
                </tr>
            </form>
            <%
        }
            %>
        </table>
    </div>
</section>

<header id="main-header" class="log">
    <div class="logHeader2">
        <img src="/DSLOGO.png" alt="" class="CBlogo">
    </div>
</header>

<footer id="logFooter">
    <div class="footerWrapper">
        <div class="footer-social-list">
            <p>Copyright &copy; 2018-2019 CloudBond. All Rights Reserved.</p>
        </div>
    </div>
</footer>
<script src="/js/main.js"></script>
</body>
</html>
```

chatroom.jsp

```
<!DOCTYPE html>
<html>
<head>
   <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0">
   <title>CloudBond - Chat</title>
   <link rel="stylesheet" href="/css/main.css" />
   <link rel="stylesheet" type="text/css" media="screen" href="/css/generic2.css" />
   <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600" rel="stylesheet">
</head>
<body>
<noscript>
   <h2>Sorry! Your browser doesn't support Javascript</h2>
</noscript>
<header>
   <div class="logHeader2">
       <img src="/DSLOGO.png" alt="" class="CBlogo">
   </div>
</header>

<div id="username-page">
   <div class="username-page-container">
       <h1 class="title">Welcome ${username}</h1>
       <form id="usernameForm" name="usernameForm">
           <div class="form-group">
               <input type="text" id="name" style="display:none" value="${username}" autocomplete="off" class="form-control"
/>
           </div>
           <div class="form-group">
               <button type="submit" class="red username-submit">Start Chatting</button>
           </div>
       </form>
   </div>
</div>

<div id="chat-page" class="hidden">
   <div class="bgA">
       <div class="bgABlur">

       </div>
   </div>
   <div class="chat-container">
       <div class="chat-header">
           <h2>CloudBond Chat Room</h2>
       </div>
       <div class="connecting">
           Connecting...
       </div>
       <ul id="messageArea" style="background:inherit !important" class="style-2">

       </ul>
       <form id="messageForm" name="messageForm">
           <div class="form-group">
               <div class="input-group clearfix">
                   <input type="text" id="message" placeholder="Type a message..." autocomplete="off" class="form-control"/>
                   <button type="submit" class="white">Send</button>
               </div>
           </div>
       </form>
   </div>
</div>
<footer id="logFooter">
   <div class="footerWrapper">
       <div class="footer-social-list">
           <p>Copyright &copy; 2018-2019 CloudBond. All Rights Reserved.</p>
       </div>
   </div>
</footer>
```

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.4/sockjs.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/stomp.js/2.3.3/stomp.min.js"></script>
<script src="/js/main.js"></script>
</body>
</html>
```

main.js

```
'use strict';

var usernamePage = document.querySelector('#username-page');
var chatPage = document.querySelector('#chat-page');
var usernameForm = document.querySelector('#usernameForm');
var messageForm = document.querySelector('#messageForm');
var messageInput = document.querySelector('#message');
var messageArea = document.querySelector('#messageArea');
var connectingElement = document.querySelector('.connecting');

var stompClient = null;
var username = null;

var colors = [
   '#2196F3', '#32c787', '#00BCD4', '#ff5652',
   '#ffc107', '#ff85af', '#FF9800', '#39bbb0'
];

function connect(event) {
   username = document.querySelector('#name').value.trim();

   if(username) {
       usernamePage.classList.add('hidden');
       chatPage.classList.remove('hidden');

       var socket = new SockJS('/ws');
       stompClient = Stomp.over(socket);

       stompClient.connect({}, onConnected, onError);
   }
   event.preventDefault();
}


function onConnected() {
   // Subscribe to the Public Topic
   stompClient.subscribe('/topic/public', onMessageReceived);

   // Tell your username to the server
   stompClient.send("/app/chat.addUser",
       {},
       JSON.stringify({sender: username, type: 'JOIN'})
   )

   connectingElement.classList.add('hidden');
}


function onError(error) {
   connectingElement.textContent = 'Could not connect to WebSocket server. Please refresh this page to try again!';
   connectingElement.style.color = 'red';
}


function sendMessage(event) {
   var messageContent = messageInput.value.trim();
   if(messageContent && stompClient) {
       var chatMessage = {
           sender: username,
           content: messageInput.value,
           type: 'CHAT'
       };
       stompClient.send("/app/chat.sendMessage", {}, JSON.stringify(chatMessage));
       messageInput.value = '';
   }
   event.preventDefault();
}
```

```
function onMessageReceived(payload) {
   var message = JSON.parse(payload.body);

   var messageElement = document.createElement('li');

   if(message.type === 'JOIN') {
       messageElement.classList.add('event-message');
       message.content = message.sender + ' joined!';
   } else if (message.type === 'LEAVE') {
       messageElement.classList.add('event-message');
       message.content = message.sender + ' left!';
   } else {
       messageElement.classList.add('chat-message');

       var avatarElement = document.createElement('i');
       var avatarText = document.createTextNode(message.sender[0]);
       avatarElement.appendChild(avatarText);
       avatarElement.style['background-color'] = getAvatarColor(message.sender);

       messageElement.appendChild(avatarElement);

       var usernameElement = document.createElement('span');
       var usernameText = document.createTextNode(message.sender);
       usernameElement.appendChild(usernameText);
       messageElement.appendChild(usernameElement);
   }

   var textElement = document.createElement('p');
   var messageText = document.createTextNode(message.content);
   textElement.appendChild(messageText);

   messageElement.appendChild(textElement);

   messageArea.appendChild(messageElement);
   messageArea.scrollTop = messageArea.scrollHeight;
}


function getAvatarColor(messageSender) {
   var hash = 0;
   for (var i = 0; i < messageSender.length; i++) {
       hash = 31 * hash + messageSender.charCodeAt(i);
   }
   var index = Math.abs(hash % colors.length);
   return colors[index];
}

function test(){
   var node = document.createElement('li');
   node.classList.add('event-message');
   var textnode = document.createTextNode("I left!");
   node.appendChild(textnode);
   document.getElementById("messageArea").appendChild(node);
}
usernameForm.addEventListener('submit', connect, true)
messageForm.addEventListener('submit', sendMessage, true)

// show homepage nav bar
var i = 0;

function showNav(){

   if(i % 2 == 0){
       document.getElementById("hidden").style.visibility = "visible";
   }
   else{
       document.getElementById("hidden").style.visibility = "hidden";
   }
   i++;
}

var i = 0;

function showNav2(){

   if(i % 2 == 0){
       document.getElementById("hide").classList.remove("hidden");
   }
   else{
       document.getElementById("hide").classList.add("hidden");
   }

   i++;
}

function showPop(){
   var logIn = document.getElementById("signIn");
   var sigUp = document.getElementById("signUp");
   var logTrue = 0;

   if (logTrue != 1) {
       sigUp.style.display='none';
       logIn.style.display='grid';
       logTrue = 1;
   }
   else {
       logIn.style.display='none';
       logTrue = 0;
   }
}



function showPop2(){
   var sigUp = document.getElementById("signUp");

   if (sigUp.style.display === "none") {
       sigUp.style.display = "grid";
   } else {
       sigUp.style.display = "none";
   }
}


function showPass(){
   var x = document.getElementById("Pass");
   var y = document.getElementById("pass2");
   var z = document.getElementById("pass3");

   if (x.type === "password") {
       x.type = "text";
   } else {
       x.type = "password";
   }
   if (y.type === "password") {
       y.type = "text";
   } else {
       y.type = "password";
   }
   if (z.type === "password") {
       z.type = "text";
   } else {
       z.type = "password";
   }
}
```

```
function checkUp(input) {
    if (input.value != document.getElementById('pass2').value) {
        document.getElementById("invalid").innerHTML = "Password does not match";
    } else {
        document.getElementById("invalid").innerHTML = "";
    }
}

function checkDown(input) {
    if (input.value != document.getElementById('Pass').value) {
        document.getElementById("invalid").innerHTML = "Password does not match";
    } else {
        document.getElementById("invalid").innerHTML = "";
    }
}

window.onscroll = function() {posFix()};

var header = document.getElementById("main-header");
var sticky = header.offsetTop;

function posFix() {
    if (window.pageYOffset > sticky) {
        header.classList.add("sticky");
        header.style.boxShadow = "0 2px 10px rgba(0,0,0,.5)";
    } else {
        header.classList.remove("sticky");
        if(header.classList.contains("alt") == true || header.classList.contains("log") == true){
            header.style.boxShadow = "none";
        }else{
            header.style.boxShadow = "0 0 0.25em 0 rgba(0, 0, 0, 0.15)";
        }
    }
}
```

main.css

```
html,body {
   height: 100%;
   overflow: hidden;
}

.clearfix:after {
   display: block;
   content: "";
   clear: both;
}

.hidden {
   display: none;
}

.form-control {
   width: 100%;
   min-height: 38px;
   font-size: 15px;
   border: 1px solid #c8c8c8;
}

.form-group {
   margin-bottom: 15px;
}

input {
   padding-left: 10px;
   outline: none;
}

h1 {
   font-size: 1.7em;
}

a {
   color: #128ff2;
}

button {
   box-shadow: none;
   border: 1px solid transparent;
   font-size: 14px;
   outline: none;
   line-height: 100%;
   white-space: nowrap;
   vertical-align: middle;
   padding: 0.6rem 1rem;
   border-radius: 2px;
   transition: all 0.2s ease-in-out;
   cursor: pointer;
   min-height: 38px;
}

button.default {
   background-color: #e8e8e8;
   color: #333;
   box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.12);
}

button.primary {
   background-color: #128ff2;
   box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.12);
   color: #fff;
}

#username-page {
   text-align: center;
}

.username-page-container {
   box-shadow: 0 2px 20px rgba(255, 255, 255, 0.5);
   border: 2px #FFF solid;
   border-radius: 2px;
   width: 100%;
   max-width: 500px;
   display: inline-block;
   margin-top: 42px;
   vertical-align: middle;
   position: relative;
   padding: 35px 55px 35px;
   min-height: 250px;
   position: absolute;
   top: 50%;
   left: 0;
   right: 0;
   margin: 0 auto;
   margin-top: -160px;
}

.username-page-container .username-submit {
   margin-top: 10px;
}


#chat-page {
   position: relative;
   height: 100%;
}

.chat-container {
   background: #2a2f4a !important;
   max-width: 50vw;
   margin-left: auto;
   margin-right: auto;
   box-shadow: 0 2px 10px rgba(255, 255, 255, 0.5);
   border:2px antiquewhite solid;
   height: 75vh;
   max-height: 600px;
   position: relative;
}

#chat-page ul {
   list-style-type: none;
   margin: 0;
   margin-left: 0.5em !important;
   margin-right: 1em !important;
   overflow: auto;
   overflow-y: scroll;
   padding: 0 20px 0px 20px;
   height: calc(100% - 150px);
}

#chat-page #messageForm {
   padding-top: 20px;
   padding-left:15px;
   padding-right: 15px;
}

#chat-page ul li {
   line-height: 1.5rem;
   padding: 10px 20px;
   margin: 0;
   border-bottom: 1px solid rgba(180,180,180,0.3);
}

#chat-page ul li p {
   margin: 0;
}

#chat-page .event-message {
   text-align: center;
   clear: both;
}

#chat-page .event-message p {
   color: rgba(225, 120, 83, 0.91);
   font-size: 14px;
   word-wrap: break-word;
}

#chat-page .chat-message {
   padding-left: 68px;
   position: relative;
}

#chat-page .chat-message i {
   position: absolute;
   width: 42px;
   height: 42px;
   overflow: hidden;
   left: 10px;
   display: inline-block;
   vertical-align: middle;
   font-size: 18px;
   line-height: 42px;
   color: #fff;
   text-align: center;
   border-radius: 50%;
   font-style: normal;
   text-transform: uppercase;
}

#chat-page .chat-message span {
   color: #f9f9f9;
   font-weight: 600;
}

#chat-page .chat-message p {
   color: #fefefe;
}

#messageForm .input-group input {
   float: left;
   max-width: 80%;
   border: 2px white solid;
   margin-right: 2.5vh;
}

#messageForm .input-group button {
   float: left;
}

.chat-header {
   text-align: center;
   padding: 15px;
}

.chat-header h2 {
   margin: 0;
   font-size: 1.5em !important;
}

.chat-header h2:after{
   content: '';
   background-color: #fefefe;
   display: block;
   height: 3px;
   margin-top: 0.5em !important;
   margin-bottom: -0.6em !important;
   margin: 0 auto;
   width: 60%;
}
.connecting {
   padding-top: 5px;
   text-align: center;
   color: #777;
   position: absolute;
   top: 65px;
   width: 100%;
}


@media screen and (max-width: 730px) {

   .chat-container {
       margin-left: 10px;
       margin-right: 10px;
       margin-top: 10px;
   }
}

@media screen and (max-width: 480px) {
   .chat-container {
       height: calc(100% - 30px);
   }

   .username-page-container {
       width: auto;
       margin-left: 15px;
       margin-right: 15px;
       padding: 25px;
   }

   #chat-page ul {
       list-style-type: none;
       margin: 0;
       background: #2a2f4a !important;
       overflow: auto;
       overflow-y: scroll;
       padding: 0 20px 0px 20px;
       height: calc(100% - 150px);
   }

   #messageForm .input-group button {
       width: 65px;
   }

   #messageForm .input-group input {
       width: calc(100% - 70px);
   }

   .chat-header {
       padding: 10px;
   }

   .connecting {
       top: 60px;
   }

   .chat-header h2 {
       font-size: 1.5em !important;
   }

   .chat-header h2:after{
       content: '';
       background-color: #fefefe;
```

```
        display: block;
        height: 3px;
        margin-top: 0.5em !important;
        margin-bottom: -0.6em !important;
        margin: 0 auto;
        width: 60%;
    }
}

.style-2::-webkit-scrollbar-track
{
    -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.3);
    background-color: inherit;
}

.style-2::-webkit-scrollbar
{
    width: 10px;
    background-color: inherit;
}

.style-2::-webkit-scrollbar-thumb
{
    -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,.3);
    background-color: rgba(214, 250, 255, 0.91);
}

.title{
    padding-top:1em;
}

.bgA{
    position: absolute;
    top: 11%;
    left: 0;
    width: 100vw;
    height: 55vh;
    background: linear-gradient(-45deg, #EE7752, #E73C7E, #23A6D5, #23D5AB);
    background-size: 400% 400%;
    -webkit-animation: Gradient 15s ease infinite;
    -moz-animation: Gradient 15s ease infinite;
    animation: Gradient 15s ease infinite;
}
@-webkit-keyframes Gradient {
    0% {
        background-position: 0% 50%
    }
    50% {
        background-position: 100% 50%
    }
    100% {
        background-position: 0% 50%
    }
}

@-moz-keyframes Gradient {
    0% {
        background-position: 0% 50%
    }
    50% {
        background-position: 100% 50%
    }
    100% {
        background-position: 0% 50%
    }
}

@keyframes Gradient {
    0% {
        background-position: 0% 50%
    }
    50% {
        background-position: 100% 50%
    }
    100% {
        background-position: 0% 50%
    }
}

.bgABlur{
    display: block;
    width: 100%;
    height: 100%;
    background-image: linear-gradient(#252a44,#2a2f4adb, #ffffff38,#ffffff00,#ffffff38,#2a2f4adb,#252a44);
}
```

generic2.css

```
/* Global */
*{
 margin: 0;
 padding: 0;
 border: 0;}

body{
 letter-spacing: 0.025em;
 vertical-align: baseline;
 font-size: 100%;
 font-family: 'Source Sans Pro', sans-serif;
 background-color: #242943;
 color: #ffffff;
}

ol, ul {
 list-style: none;
}

a{
 text-decoration: none;
 color: inherit;
}

a:hover {
 border-bottom-color: transparent;
 color: #9bf1ff;
 cursor: pointer;
}

a:active{
 color: #53e3fb;
}

img{
 width: 100%;
 height: auto;
 max-height: 55vh;
 object-fit: cover;
 display: block;
}

h1, h2, h3, h4, h5, h6 {
 font-weight: 600;
 line-height: 1.65;
 font-size: 2.5em;
}

input[type="text"], input[type="password"], input[type="email"], input[type="tel"], input[type="search"], input[type="url"],
select {
 height: 2.75em;
}

input[type="text"], input[type="password"], input[type="email"], input[type="tel"], input[type="search"], input[type="url"],
select, textarea {
 -moz-appearance: none;
 -webkit-appearance: none;
 -ms-appearance: none;
 appearance: none;
 background: rgba(212, 212, 255, 0.035);
 border: none;
 border-radius: 0;
 color: inherit;
 display: block;
 outline: 0;
 padding: 0 1em;
 text-decoration: none;
 width: 100%;
}

input[type="checkbox"] {
 -webkit-appearance: none;
 appearance: none;
 height:1.5em;
 width:1.5em;
 margin-bottom: -0.325em;
 margin-right: 0.5em;
 cursor:pointer;
 position:relative;
 background-color: #1e2235;
 border: #fefefe90 1.5px solid !important;
}

input[type="checkbox"]:checked,
input[type="radio"]:checked{
 background: #53e3fb;
 border-color: #9bf1ff;
 color: #242943;
}

input[type="checkbox"]:focus,
input[type="radio"]:focus{
 box-shadow: 0 0 0 1px #9bf1ff;
 border: none !important;
}

input[type="file"]{
 background-color: white;
 color: #242943;
 text-transform: uppercase !important;
 font-size: 1em;
 font-weight: 600;
 letter-spacing: 0.1em;
}
input, textarea{
 font-family: 'Source Sans Pro', sans-serif;
 font-weight: 300;
 letter-spacing: 0.025em;
 line-height: 1.65;
}

input:focus,textarea:focus{
 outline: #9bf1ff 2px solid !important;
}

.wrapper textarea{
 width:97%;
 font-size: 1em;
 padding-top: 1.5em;
}

button {
 -moz-appearance: none;
 -webkit-appearance: none;
 -ms-appearance: none;
 appearance: none;
 -moz-transition: background-color 0.2s ease-in-out, box-shadow 0.2s ease-in-out, color 0.2s ease-in-out;
 -webkit-transition: background-color 0.2s ease-in-out, box-shadow 0.2s ease-in-out, color 0.2s ease-in-out;
 -ms-transition: background-color 0.2s ease-in-out, box-shadow 0.2s ease-in-out, color 0.2s ease-in-out;
 transition: background-color 0.2s ease-in-out, box-shadow 0.2s ease-in-out, color 0.2s ease-in-out;
 background-color: transparent;
 border: 0;
 border-radius: 0;
 box-shadow: inset 0 0 0 2px #ffffff;
 color: #ffffff;
 cursor: pointer;
 display: inline-block;
 font-size: 0.8em;
 font-weight: 600;
 height: 3.5em;
 letter-spacing: 0.25em;
 line-height: 3.5em;
 padding: 0 1.75em;
 text-align: center;
 text-decoration: none;
 text-transform: uppercase;
 white-space: nowrap;
}

button.white{
 background-color: #ffffff;
 box-shadow: none;
 color: #242943 !important;
}

button.white2{
 background-color: #ffffff;
 box-shadow: none;
 color: #242943 !important;
 font-size: 1em;
}

button.red{
 background-color: #FB6B53;
 box-shadow: none;
 color: #242943 !important;
 font-size: 1em;
}

button:hover{
 color: #9bf1ff;
 box-shadow: inset 0 0 0 2px #9bf1ff;
}

button:focus {outline:0;}

button:active{
 color: #53e3fb;
 box-shadow: inset 0 0 0 2px #53e3fb;
}

button.white:hover{
 background-color: #9bf1ff;
}

button.white2:hover{
 background-color: #9bf1ff;
}

button.red:hover{
 background-color: #9bf1ff;
}

button.white:active{
 color: #53e3fb;
}

.wrapper{
 padding: 4em 0 2em 0;
 margin: 0 auto;
 min-width: 75vw;
 max-width: 65em;
 width: calc(100% - 6em);
 overflow: hidden;
}

.errorWrapper{
 width:100%;
 height: 85vh;
 margin: 0 auto;
 overflow: hidden;
 display: grid;
 grid-template-rows: auto minmax(2em,auto) auto auto;
 text-align: center;
 justify-content: center;
 align-content: center;
}

.errorWrapper h2{
 font-size: 2em;
}

.hidden{
 display: none;
}

/* SVG */
.svg-menu{
 width: 1.5em;
 overflow: visible;
}

.svg-login{
 width: 1.6em;
 overflow: visible;
}

.logSide .svg-login{
 width: 2.5em;
 overflow: visible;
}

.svg-up{
 width: 1em;
 padding-top: 5px;
 overflow: visible;
}

.svg-fb{
 width: 1.5em;
 overflow: visible;
}

.svg-tw{
 width: 1.5em;
 overflow: visible;
}

.svg-yt{
 width: 1.5em;
 overflow: visible;
}

.svg-li{
 width: 1.5em;
 overflow: visible;
}

.svg-insta{
 width: 1.5em;
 overflow: visible;
}

.svg-error{
 width: 10em;
 overflow: visible;
}
```

```
.svg-eye, .svg-noEye{
 width: 1.5em;
 overflow: visible;
 margin-bottom: -0.3em;
 margin-left: 0.5em;
}

.animate {
 -webkit-animation: animatezoom 0.6s;
 animation: animatezoom 0.6s
}

@-webkit-keyframes animatezoom {
 from {-webkit-transform: scale(0)}
 to {-webkit-transform: scale(1)}
}

@keyframes animatezoom {
 from {transform: scale(0)}
 to {transform: scale(1)}
}

/* Header */
.headerWrapper{
 width:95%;
 margin:auto;
 display: grid;
 grid-template-columns: 1fr 1fr;
}

#main-header{
 background-color: #2a2f4a;
 cursor: default;
 font-weight: 600;
 height: 3.25em;
 letter-spacing: 0.25em;
 line-height: 3.25em;
 text-transform: uppercase;
 width: 100%;
 display:block;
}

#main-header.alt{
 background-color: transparent;
 box-shadow: none;
 cursor: default;
 font-weight: 600;
 height: 3.25em;
 letter-spacing: 0.25em;
 line-height: 3.25em;
 text-transform: uppercase;
 width: 100%;
 position:absolute;
}

button#hide{
 outline: none;
 box-shadow: none;
 border: none;
}
header nav{
 display: grid;
 grid-template-columns: 3fr 1fr;
 justify-content: end;
}

#header nav a {
 display: block;
 font-size: 0.8em;
 height: inherit;
 line-height: inherit;
 position: relative;
}

header ul{
 display: grid;
 grid-template-columns: 1fr 1fr 1fr 1fr;
 text-align: center;
 font-weight:500;
}

header li:nth-child(1){
 text-align: right;
}

.menu{
 display:grid;
 justify-content: end;
 align-content: center;
}

.menu:hover{
 color: #9bf1ff;
}

.menu:active{
 color: #53e3fb;
}

.menu button{
 padding-top: 5px;
 background: none;
 color: inherit;
 cursor: pointer;
 outline: inherit;
 text-align: inherit;
 text-indent: inherit;
}

.login a{
 display:grid;
 grid-template-columns: 1fr 1fr;
 text-align: right;
 align-items: center;
 justify-items: end;
}

.sticky{
 position: fixed !important;
 background-color: #242943 !important;
 top: 0;
 width: 100%;
 -webkit-transition: box-shadow 200ms ease-in-out;
 transition: box-shadow 200ms ease-in-out;
 transition-property: box-shadow;
 /* box-shadow: 0 2px 4px -1px rgba(1,1,1,0.06),  0 4px 5px 0 rgba(0,0,0,0.06),  0 1px 10px 0 rgba(0,0,0,0.08); */
}

/* Header */
#main-header-side{
 display: grid;
 grid-template-columns: 1fr 1fr;
 grid-column-gap: 10px;
}

#main-header-side .login a{
 display:grid;
 justify-content: end;
 align-content: center;
 padding-top: 12px
}

/* Header Logo */
#main-header .logo{
 -moz-transition: background-color 0.2s ease-in-out, color 0.2s ease-in-out;
 -webkit-transition: background-color 0.2s ease-in-out, color 0.2s ease-in-out;
 -ms-transition: background-color 0.2s ease-in-out, color 0.2s ease-in-out;
 transition: background-color 0.2s ease-in-out, color 0.2s ease-in-out;
}

#main-header .logo strong {
 border: currentColor 2px solid;
 border-right-color: transparent;
 color: #ffffff;
 display: inline-block;
 line-height: 1.65em;
 margin-right: -0.25em;
 padding: 0 0.125em 0 0.375em;
 font-size: 1em;
}

#main-header .logo:hover strong{
 color: #9bf1ff;
}

#main-header .logo span{
 color: #FB6B53;
 font-weight: 600;
}

#main-header .logo:hover span {
 color: #9bf1ff;
}

#main-header .logo:active span{
 color: #53e3fb;
}

#showcase h1{
 font-size: 3em;
}
#showcaseImg{
 opacity: 0.3;
 max-width: 100%;
 min-height: 100vh;
 margin: auto;
 object-fit: cover;
}

#showcase .wrapper{
 width:75%;
 margin: auto;
 padding-top: 10em;
}

#showcase .inner{
 width:50%;
 display: grid;
 grid-template-columns: 1fr 1fr;
}

#showcase h1{
 padding-top: 0.5em;
}

/* CSS Logo */
#showLogo{
 display: grid;
 grid-template-rows: repeat(5,1fr);
 grid-template-columns: repeat(3,1fr);
 width: 100px;
 font-family: 'Source Sans Pro', sans-serif;
 text-transform: uppercase;
 letter-spacing: 0.25em;
 overflow: visible;
}

#logoBorder{
 border: #ffffff 5px solid;
 grid-row: 1/end;
 grid-column: 1/3;
}

#logoBody{
 background-color: #242943;
 color: #FB6B53;
 grid-row: 3/5;
 grid-column: 3/4;
 text-align: right;
 font-size: 1.5em;
}

#d{
 font-size: 1.5em;
 grid-row: 2/4;
 grid-column: 1/2;
 text-align: right;
 width:30px;
}

#i{
 font-size: 1.5em;
 grid-row: 3/5;
 grid-column: 2/3;
 background-color: #242943;
 text-align: center;
 letter-spacing: 0.25em;
 width:25px;
}

#red{
 color: #FB6B53;
}

.absolute{
 position: absolute;
 display: grid;
 height: 100%;
}

/* Title */
h1.major{
 width: max-content;
 display: block;
}

.minor{
 margin-bottom: 2em;
 color: #cdcdcf;
 font-size: 90%;
}

.minor #line{
 margin-left: 10px;
 margin-right: 10px;
}

h1.major > :first-child {
 margin-bottom: 0;
 width: calc(100% + 0.5em);
}

h1.major > :first-child:after{
 content: '';
 background-color: #fefefe;
 display: block;
 height: 3px;
 margin: 0.325em 0 0.3em 0;
 width: 100%;
}

/* Footer */
.empty{
 background-color: rgba(212, 212, 255, 0.1);
 display: block;
 height:1px;
}

.footerWrapper{
 padding: 3em 0 2em 0;
 margin: 0 auto;
 width: 95%;
 overflow: hidden;
 text-align: center;
}

.footerWrapper.alt{
 bottom: 0;
 left: 0;
 right:0;
 position: fixed !important;
}

.footer-nav-list{
 display:grid;
 grid-template-columns: repeat(6,1fr);
 min-width: 600px;
 max-width: 50%;
 margin:auto;
 text-align: center;
 align-items: center;
 margin-bottom: 0.5rem;
 font-weight: 600;
}

.footer-nav-list.alt{
 grid-template-columns: repeat(5,1fr) !important;
}

.footer-nav-list2{
 display:grid;
 grid-template-columns: repeat(5,1fr);
 min-width: 400px;
 max-width: 50%;
 left: 0;
 right: 0;
 margin: auto;
 text-align: center;
 align-items: center;
 padding: 0.5rem 0;
 margin-bottom: 1rem;
 font-weight: 300;
 font-size: 0.9em;
 position: absolute;
 background: #242943fc;
}

.footer-nav-list2 li{
 border-left: #cdcdcf 2px solid;
```

```
}

.footer-nav-list2 li:first-child{
 border-left-color: transparent;
}

.footer-nav-list .write{
 color: #242943;
 background: #fff;
 padding: 0.1em .3em;
}

footer li:nth-child(5):hover{
 color: #9bf1ff;
 cursor: pointer;
}

footer li:nth-child(5):active{
 color: #53e3fb;
}

.footer-nav-list .write a:hover{
 color: #242943;
}

.footer-nav-list .write:hover {
 background:#9bf1ff;
}

.footer-nav-list .write:active {
 background:#53e3fb;
}

.footer-social-list{
 display: flex;
 align-items: center;
 justify-content: center;
 padding-top: 10px;
}

.footer-social-list a{
 padding-right: 10px
}

footer p{
 padding-bottom: 5px;
 font-size:0.8em;
 color: #cdcdcf;
 font-weight: 300;
}

section#game .wrapper{
 display: grid;
 grid-template-columns:1fr 1fr 1fr;
 grid-template-rows: 1fr 1fr 1fr 1fr;
 grid-gap:1em;
}

section#game div > div{
 border: #e8491d 2px solid;
 grid-row:2/5;
}

section#game h1{
 grid-column:1/4;
 grid-row:1;
}

section#game h2{
 font-size: 2em !important;
```

```
}

section#game button{
 display: inline;
}

/* Article */
article p{
 margin: 0 0 2em 0;
 display:block;
 font-weight: 300;
}

picture>img{
 margin-bottom: 2em;
}

/* Login Page */
.log{
 background-color: transparent !important;
 top:0;
 left: 0;
 right:0;
 position: absolute !important;
}

.logInput{
 border: 2px rgba(255, 255, 255, 0.8) solid !important;
}

.logInput:hover{
 border: 2px #9bf1ff solid !important;
}

.logInput:active{
 border: 2px #53e3fb solid !important;
}

.logHeader{
 top: 0;
 left: 3%;
 right:0;
 width: 95%;
 height: 10vh;
 margin: 0 auto;
 display: grid;
 grid-template-columns: 1fr 1fr;
 vertical-align: middle;
 align-content: center;
 align-items: center;
 position: fixed;
}

.logHeader .logo{
 font-size: 1.25em;
}

.logSide{
 padding-top: 0.5em;
 display:grid;
 max-width: 35vw;
 min-width: 30vw;
 grid-template-columns: auto auto auto;
 grid-column-gap: 10px;
 align-items: center;
 justify-items: center;
}

.logSide .b{
 padding-top: 1em;
 display: grid;
 grid-template-rows: auto minmax(auto,1em);
 text-align: left;
}
.logSide .b a{
 font-weight: 300;
 font-size: 0.5em;
 letter-spacing: 0.1em;
 text-transform: none;
 margin-top: -1em;
}

.logSide .b a:hover{
 text-decoration: underline;
}


.logSide .c{
 padding-top: 0.75em;
}

.logSide .see{
 box-shadow:none !important;
 padding: 0;
}

#logGrid{
 display: grid;
 grid-template-columns: 1fr 1fr;
 height: 100vh;
 align-items: center;
}

.logLeft{
 height:100vh;
 background-image: linear-gradient(to bottom right,#242943 , #2a2f4a,#ffffff90, #FB6B53,#2a2f4a,#242943 );
 background-size: 115% 100%;
 -webkit-animation: Gradient 10s ease infinite;
 -moz-animation: Gradient 15s ease infinite;
 animation: Gradient 10s ease infinite;
}

.logBlur{
 height:100vh;
 background-image: linear-gradient(to left,#252a44,#242943, #2a2f4af0, #FB6B5300,#2a2f4ac0,#242943d0 );
 background-size: 110% 90%;
 background-position: center right;
 -webkit-animation: Gradient 20s ease infinite;
 -moz-animation: Gradient 10s ease infinite;
 animation: Gradient 20s ease infinite;
}

@-webkit-keyframes Gradient {
 0% {
   background-position: 0% 50%
 }
 50% {
   background-position: 100% 50%
 }
 100% {
   background-position: 0% 50%
 }
}

@-moz-keyframes Gradient {
 0% {
   background-position: 0% 50%
 }
 50% {
   background-position: 100% 50%
 }
 100% {
   background-position: 0% 50%
 }
}

@keyframes Gradient {
 0% {
   background-position: 0% 50%
 }
 50% {
   background-position: 100% 50%
 }
 100% {
   background-position: 0% 50%
 }
}

.logRight{
 padding-left: 3em;
 display:grid;
 width:35vw;
}

.logRight2{
 border: 2px #ffffff solid;
 box-shadow: 0 2px 20px rgba(255, 255, 255, 0.5);
 padding-left: 3em;
 display:block;
 width:35vw;
 height: 80vh !important;
}

.logRight h2,.logPop h2{
 font-size: 1.5em;
}

.logRight .white{
 margin-right: 0.5em;
}

#logFooter{
 bottom: 0;
 left: 0;
 right:0;
 position: absolute;
 padding-bottom: 0 !important;
 margin-bottom: -1em;
 overflow: hidden;
}

.logWrap{
 display: none;
 top: 0;
 width: 50vw;
 height: 100vh;
 position: fixed;
 text-align: left;
 align-items: center;
 justify-content: center;
}

.logWrap2{
 display: grid;
 top: 0;
 width: 50vw;
 height: 100vh;
 position: fixed;
 text-align: left;
 align-items: center;
 justify-content: center;
}

.logPop{
 display: block;
 padding:3em;
 padding-right: 5em;
 width:30vw;
 height: 63vh;
 background-color: #242943e0;
 border: 2px #ffffff solid;
}

.logPop.in{
 height: 55vh !important;
}

.logPop .white{
 margin-right: 3em;
}

.logPop h1:after,.submitWrapper h1::after,.logRight2 h1::after{
 content: '';
 background-color: #fefefe;
 display: block;
 height: 3px;
 margin: 0.325em 0 0.3em 0;
 width: 60%;
}

.logPop label{
 display:inline-block;
 font-size: 1.1em;
 margin-bottom: 0.25em;
}

button.see{
 box-shadow:none !important;
 height: 1em;
 width: 1em;
 padding: 0;
}

.newShowcase{
 width:100%;
 height: 50vh;
 display:grid;
 grid-template-columns: repeat(4,1fr);
 grid-template-rows: 1fr 1fr;
 grid-gap: 1em;
 margin-bottom: 4em;
}

.newShowcase>div{
 border: white 2px solid;
 height: fit-content;
}

.newOne{
 grid-column: 1/3;
 grid-row: 1/3;
 max-width: 100%;
 overflow: hidden;
 align-self: center;
}

.newOne:hover,.newTwo:hover{
 position: relative;
 overflow: hidden;
}
```

```
.newOne:hover>div,.newTwo:hover>div{
 display: block;
 overflow: hidden;
}

.newsTitle{
 display: none;
 position: absolute;
 top:0;
 left:0;
 right: 0;
 width: 100%;
 height: 100%;
 background-color: #242943d0;
 color: white;
 padding-top: 40%;
 font-size: 0.8em;
 text-align: center;
 overflow: hidden;
}

.newsubTitle{
 display: none;
 position: absolute;
 top:0;
 left:0;
 right: 0;
 width: 100%;
 height: 100%;
 background-color: #242943d0;
 color: white;
 padding-top: 40%;
 font-size: 0.6em;
 text-align: center;
 overflow: hidden;
}

.news{
 display: grid;
 grid-template-columns: 3fr 7fr;
 grid-template-rows: 1fr 1fr auto;
 width: 100%;
 min-height: 40vh;
 grid-column-gap: 1em;
 margin-bottom: 2em;
}

.newButton{
 width:30%;
}

.news .image{
 grid-row: 1/end;
 border: white 2px solid;
 overflow: hidden;
 width:30vw;
}

.news .image:hover{
 position: relative;
}

.news img{
 min-width: 100%;
 width: auto;
 max-width: 35vw;
 height: auto;
 object-fit: cover;
}
```

```
.submitWrapper h1{
 min-width:40vw;
 max-width: 55%;
 margin-bottom: 2em;
}

.submitWrapper h2{
 font-size: 1.5em;
}

.submitWrapper{
 margin-bottom: 4em;
}

/* Leaderboard */
#leaderboard{
 margin: 0 auto;
}

.col:nth-child(even){
 background:#242943 !important;
}

.col:nth-child(odd){
 background:#353a54 !important;
}

#ldr{
 margin-bottom: 1em;
}

.col{
 display: grid;
 grid-template-columns: 1fr 4fr 1fr 2fr;
}

.col > div{
 border:white 1px solid;
 padding-left: 1em;
}

.col.main{
 height: 2.5em;
 vertical-align: middle;
 font-weight: 600;
 font-size: 1.05em;
 text-transform: uppercase;
}
.col.main > div{
 padding-top: 0.5em;
 /* border: white 2px solid; */
 border-left: #242943 1px solid;
 background: white;
 color: #242943;
}

/* chat room Page */
.gameList{
 margin: 0 auto;
 display: grid;
 width: 95%;
 height: 100%;
 grid-template-columns: repeat(3,1fr);
 grid-template-rows: repeat(3,1fr);
 grid-gap: 1em;
}

.gameList h2{
 font-size: 1.25em;
 text-align: center;
}

.gameImg{
 width: 100%;
 height: 250px;
 overflow: hidden;
 border: white 2px solid;
}

/* Generic Game */
.gameC{
 justify-self: end;
 padding-right: 4em;
}

#gameCenter{
 display: grid;
 grid-template: 1fr;
 justify-items: center;
 margin: auto;
 width:45vw;
 height: 70vh;
 margin-top: 8em;
 border: white 2px solid;
}

#gameFit{
 width: 100%;
 height: 100%;
}

.wordWrapper{
 display: grid;
 grid-template: 1fr;
 justify-content: center;
 align-self:center;
 align-items:center;
 align-content: center;
 margin:0 auto;
 width:30vw;
 text-align: center;
}

.gameTitle{
 display: block;
 margin-top: 2em;
 height: 2.5em;
 margin-bottom: -10em !important;
 padding-bottom: -10em !important;
 font-weight: 600;
 font-size: 2.5em;
}

.wordWrapper h3{
 font-size: 1.25em !important;
}
.wordWrapper h4{
 font-size: 1.5em;
 height: 1.5em;
 display: block;
}
.typeGame{
 width: 30vw !important;
}

/* Snake */
.SCORE{
 color:white;
 font-family: 'Source Sans Pro', sans-serif;
 font-weight: 600;
 font-size: 2em;
}

canvas {
 border: 2px solid white;
}

#button{
 display: none;
}

#visible{
 display: block;
}

/* Profile */
#profile h1{
 font-size: 2em;
}

#profile h2{
 font-size: 1.5em;
}

#profile p{
 font-weight: 300;
 font-size: 1em;
 margin-bottom: 2em;
}

.profileWrapper{
 width: 30%;
 margin: auto;
 border: white 1px solid;
 text-align: center !important;
}

.profileWrapper input{
 width: 50%;
 margin: 0 auto;
}

.profileWrapper input:last-child{
 margin-bottom: 3em;
}


@media screen and (max-width: 736px){
 h1 {
   font-size: 2em;
 }

 h2{
   font-size: 1em !important;
 }
}

.dropdown {
 position: relative;
 display: inline-block;
}

.dropdown-content {
 display: none;
 position: absolute;
 background-color: #f9f9f9;
 min-width: 160px;
 padding: 12px 16px;
 z-index: 1;
}

.dropdown:hover .dropdown-content {
 display: block;
}

/* For CloudBond */
.CBlogo{
 padding-top: 1.5em;
 padding-left: 6em;
 height: 10vh;
 width: 35vh;
}

.logRight2 h1{
 margin-top: 2vh;
}

#uList{
 display: block;
 height: 60vh;
 width:95%;
 overflow: auto;
}
```