

Visit at <https://hkchatroom.herokuapp.com/> and sign in with **test01** (username) & **123456** (password).

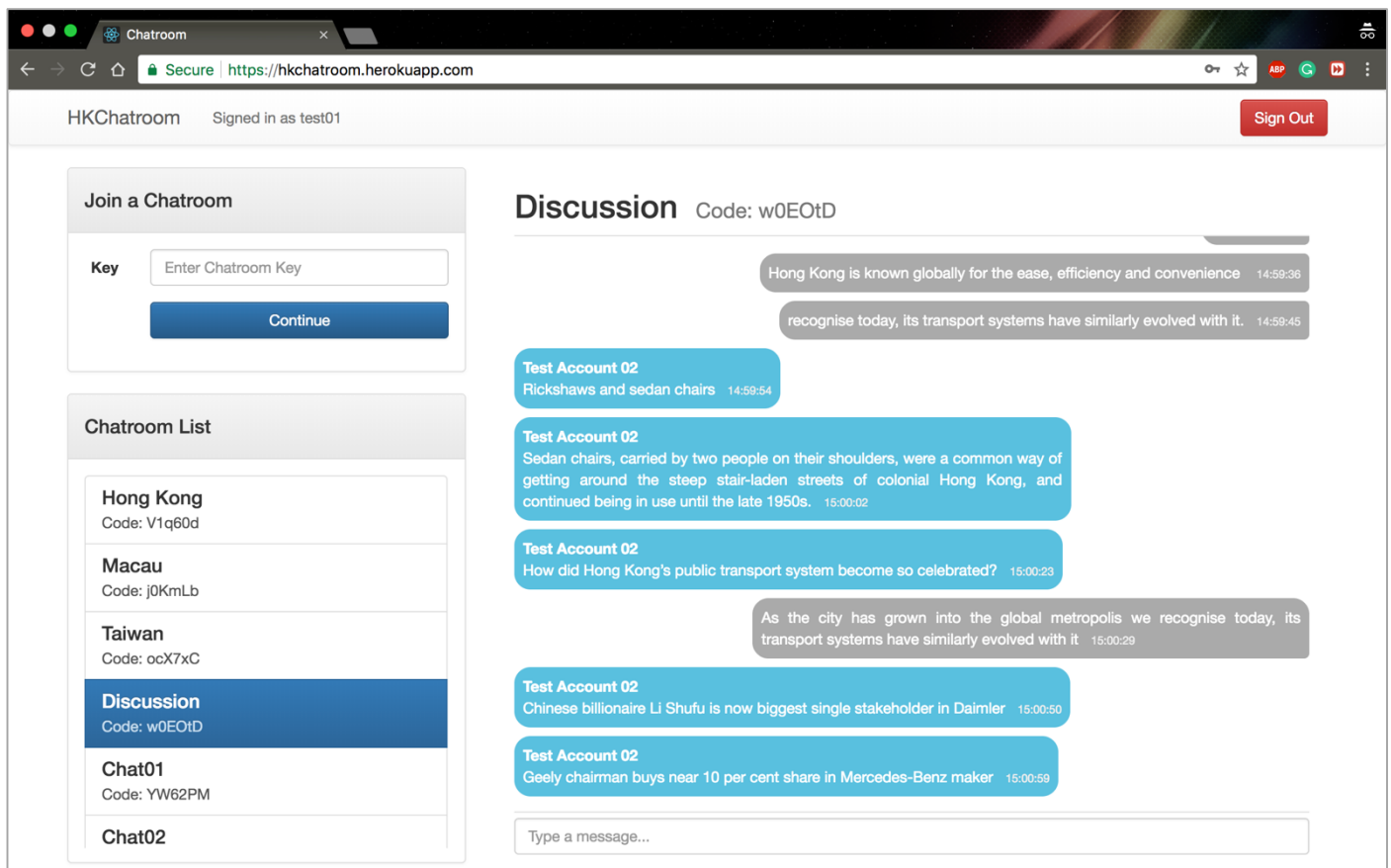
\*\*Please allow extra time for first click while awaiting start-up of heroku free server.

# HKChatroom

*Online Chatroom Web Application with MERN Framework*

**V1.1 | Feb 25 2018**

**By Jeffrey Chung**

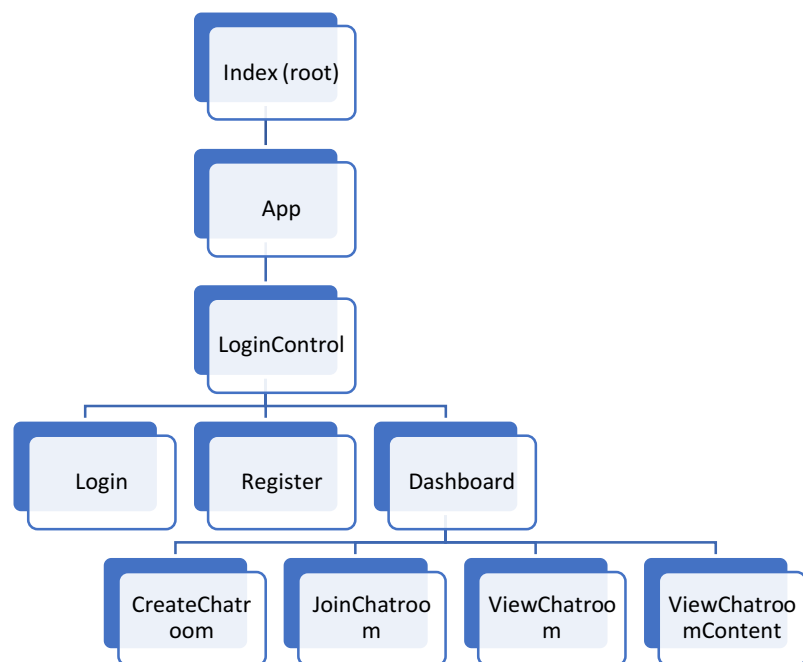


## Table of Contents

<b>HKChatroom</b> .....	<b>1</b>
<i>Online Chatroom Web Application with MERN Framework</i> .....	<b>1</b>
Front-end: ReactJS .....	<b>2</b>
Back-end: ExpressJS .....	<b>5</b>
Database: mLab MongoDB .....	<b>7</b>
User Interface Snapshots .....	<b>8</b>
Test Accounts .....	<b>10</b>
Remarks .....	<b>10</b>

## Front-end: ReactJS

- Hosted on: <https://hkchatroom.herokuapp.com/>
- The source code follows a hierarchical order:



- Source Code File Arrangement

Name	Location
<b>index.js</b>	/chat-client/src/index.js
<b>App.js</b>	/chat-client/src/App.js
<b>LoginControl.js</b>	/chat-client/src/Authentication/LoginControl.js
<b>Login.js</b>	/chat-client/src/Authentication/Login.js
<b>Register.js</b>	/chat-client/src/Authentication/Register.js
<b>Dashboard.js</b>	/chat-client/src/Authentication/Dashboard.js
<b>CreateChatroom.js</b>	/chat-client/src/Authentication/Chatroom/CreateChatroom.js
<b>JoinChatroom.js</b>	/chat-client/src/Authentication/Chatroom/JoinChatroom.js
<b>ViewChatroom.js</b>	/chat-client/src/Authentication/Chatroom/ViewChatroom.js
<b>ViewChatroomContent.js</b>	/chat-client/src/Authentication/Chatroom/ViewCharroomContent.js

- Source Code File Functionalities

Name	Function
<b>index.js</b>	<ul style="list-style-type: none"><li>• Set page title</li><li>• Map <b>App.js</b> to root of webpage</li></ul>
<b>App.js</b>	<ul style="list-style-type: none"><li>• Include Bootstrap 3.3.7 online source code reference for CSS</li><li>• Include <b>LoginControl.js</b> for this class</li></ul>
<b>LoginControl.js</b>	<ul style="list-style-type: none"><li>• Manage authentication status of end user and fetch respective pages with the following <b>states</b> and <b>methods</b>:<ul style="list-style-type: none"><li>○ States<ul style="list-style-type: none"><li>▪ username</li><li>▪ displayName</li><li>▪ chatroomList</li><li>▪ isLoggedIn</li><li>▪ isToRegister</li></ul></li><li>○ Methods</li></ul></li></ul>

Name	Function
	<ul style="list-style-type: none"> <li>▪ setLogin(isLoggedIn, username, displayName, chatroomList)</li> <li>▪ getUsername()</li> <li>▪ getDisplayName()</li> <li>▪ getChatroomList()</li> <li>▪ appendChatroomList(chatroomName, chatroomKey)</li> <li>▪ login()</li> <li>▪ logout()</li> <li>▪ register()</li> <li>• Include <b>Login.js, Register.js &amp; Dashboard.js</b> for this class</li> <li>• Display the top navigation bar</li> </ul>
Login.js	<ul style="list-style-type: none"> <li>• Implements login interface with <b>states, methods</b> and <b>inherited methods</b>: <ul style="list-style-type: none"> <li>○ States <ul style="list-style-type: none"> <li>▪ usernameReg</li> <li>▪ displayNameReg</li> <li>▪ passwordReg</li> <li>▪ confirmPasswordReg</li> <li>▪ error</li> </ul> </li> <li>○ Methods <ul style="list-style-type: none"> <li>▪ clearForm()</li> </ul> </li> <li>○ Inherited Methods <ul style="list-style-type: none"> <li>▪ setLogin()</li> </ul> </li> </ul> </li> </ul>
Register.js	<ul style="list-style-type: none"> <li>• Implements Registration interface with <b>states</b> and <b>inherited methods</b>: <ul style="list-style-type: none"> <li>○ States <ul style="list-style-type: none"> <li>▪ username</li> <li>▪ password</li> <li>▪ error</li> </ul> </li> <li>○ Inherited Methods <ul style="list-style-type: none"> <li>▪ login()</li> <li>▪ setLogin()</li> </ul> </li> </ul> </li> </ul>
Dashboard.js	<ul style="list-style-type: none"> <li>• Implements Dashboard interface (i.e. page after signed in) with <b>states, methods</b> and <b>inherited methods</b>: <ul style="list-style-type: none"> <li>○ States <ul style="list-style-type: none"> <li>▪ isDashboard</li> <li>▪ activeChatroomId</li> <li>▪ activeChatroomName</li> <li>▪ activeStringKey</li> <li>▪ activeLastMsgId</li> <li>▪ activeChatroomMsgPack</li> </ul> </li> <li>○ Methods <ul style="list-style-type: none"> <li>▪ setViews(isDashboard)</li> <li>▪ setActiveChatroom(chatroomId, chatroomName, stringKey, lastMsgId)</li> <li>▪ getActiveChatroomId()</li> <li>▪ getActiveChatroomName()</li> <li>▪ getActiveStringKey()</li> <li>▪ setActiveLastMsgId(lastMsgId)</li> </ul> </li> </ul> </li> </ul>

Name	Function
	<ul style="list-style-type: none"> <li>▪ <code>getActiveLastMsgId()</code></li> <li>▪ <code>setActiveChatroomMsgPack(msgPack)</code></li> <li>▪ <code>getActiveChatroomMsgPack()</code></li> <li>○ Inherited Methods <ul style="list-style-type: none"> <li>▪ <code>getUsername()</code></li> <li>▪ <code>getDisplayName()</code></li> <li>▪ <code>getChatroomList()</code></li> <li>▪ <code>appendChatroomList()</code></li> </ul> </li> <li>• Include <b>CreateChatroom.js, JoinChatroom.js, ViewChatroom.js &amp; ViewChatroomContent.js</b> for this class</li> <li>• Manage application views of end user and fetch respective chatroom/dashboard pages</li> </ul>
<b>CreateChatroom.js</b>	<ul style="list-style-type: none"> <li>• Display on <b>Right Div of Dashboard Page</b></li> <li>• Implements Chatroom Creation interface (i.e. input form) with <b>states</b> and <b>inherited methods</b>: <ul style="list-style-type: none"> <li>○ States <ul style="list-style-type: none"> <li>▪ <code>chatroomName</code></li> <li>▪ <code>error</code></li> </ul> </li> <li>○ Inherited Methods <ul style="list-style-type: none"> <li>▪ <code>setViews ()</code></li> <li>▪ <code>setActiveChatroom ()</code></li> <li>▪ <code>appendChatroomList ()</code></li> </ul> </li> </ul> </li> </ul>
<b>JoinChatroom.js</b>	<ul style="list-style-type: none"> <li>• Display on <b>Left Div of Dashboard &amp; Chatroom Page</b></li> <li>• Implements Joining Chatroom by Code and Accessing Chatroom List with <b>states, methods</b> and <b>inherited methods</b>: <ul style="list-style-type: none"> <li>○ States <ul style="list-style-type: none"> <li>▪ <code>chatroomKeyerror</code></li> </ul> </li> <li>○ Methods <ul style="list-style-type: none"> <li>▪ <code>setViewChatroom(key, name)</code></li> </ul> </li> <li>○ Inherited Methods <ul style="list-style-type: none"> <li>▪ <code>setViews ()</code></li> <li>▪ <code>setActiveChatroom ()</code></li> <li>▪ <code>getActiveChatroomID()</code></li> <li>▪ <code>getActiveChatroomName()</code></li> <li>▪ <code>getActiveStringKey ()</code></li> </ul> </li> </ul> </li> </ul>
<b>ViewChatroom.js</b>	<ul style="list-style-type: none"> <li>• Display on <b>Upper Right Div of Chatroom Page</b></li> <li>• Implements Chatroom Name &amp; Code Display with <b>states</b> and <b>inherited methods</b>: <ul style="list-style-type: none"> <li>○ States <ul style="list-style-type: none"> <li>▪ <code>activeChatroomID</code></li> <li>▪ <code>activeChatroomName</code></li> <li>▪ <code>activeStringKey</code></li> </ul> </li> <li>○ Inherited Methods <ul style="list-style-type: none"> <li>▪ <code>setViews ()</code></li> <li>▪ <code>setActiveChatroom ()</code></li> <li>▪ <code>getActiveChatroomID()</code></li> <li>▪ <code>getActiveChatroomName()</code></li> <li>▪ <code>getActiveStringKey ()</code></li> </ul> </li> </ul> </li> </ul>
<b>ViewChatroomContent.js</b>	<ul style="list-style-type: none"> <li>• Display on <b>Lower Right Div of Chatroom Page</b></li> </ul>

Name	Function
	<ul style="list-style-type: none"> <li>Implements Chatroom Name &amp; Code Display with <b>states, methods and inherited methods</b>:             <ul style="list-style-type: none"> <li>States                 <ul style="list-style-type: none"> <li>activeChatroomID</li> <li>activeChatroomName</li> <li>activeStringKey</li> </ul> </li> <li>Methods                 <ul style="list-style-type: none"> <li>clearActiveChatroomTypeBox()</li> <li>checkMsg()</li> <li>refreshMsgPack()</li> </ul> </li> <li>Inherited Methods                 <ul style="list-style-type: none"> <li>getActiveChatroomMsgPack ()</li> <li>getActiveLastMsgId()</li> <li>getActiveStringKey ()</li> <li>getActiveChatroomID()</li> <li>setActiveChatroom ()</li> <li>setActiveChatroomMsgPack ()</li> <li>setActiveLastMsgId()</li> </ul> </li> </ul> </li> </ul>

#### Back-end: ExpressJS

- Hosted on: <https://chatroom-hk.herokuapp.com/api/>
- Routes are coded in **/chatroom-backend/routes/api.js**
- Route Functionalities

Name	Route	Attributes	Function
<b>/load</b>	GET	-	<ul style="list-style-type: none"> <li>Determine if end user has signed in by session variables             <ul style="list-style-type: none"> <li>If yes, make query to database to obtain username, chatroom list and display name of the user</li> <li>If no, fetch error message</li> </ul> </li> </ul>
<b>/login</b>	POST	username; password	<ul style="list-style-type: none"> <li>Determine if end user possess correct login credentials by making query to database             <ul style="list-style-type: none"> <li>If yes, set session variables <b>userId, displayName &amp; username</b> for the user, and fetch JSON response composing of username, chatroom list and display name of the user</li> <li>If no, fetch error message</li> </ul> </li> </ul>
<b>/logout</b>	GET	-	<ul style="list-style-type: none"> <li>Unset session variables <b>userId, displayName &amp; username</b> for the user, and fetch null JSON response</li> </ul>
<b>/register</b>	POST	username; password; displayName	<ul style="list-style-type: none"> <li>Determine if the provided username is already taken by making query to user database</li> </ul>

Name	Route	Attributes	Function
			<ul style="list-style-type: none"> <li>○ If taken, fetch JSON response with corresponding error message</li> <li>○ If not taken, make query to create a new user in the database with the supplied information, and send JSON response with username, chatroom list and display name of the user <ul style="list-style-type: none"> <li>▪ If unsuccessful, fetch error message</li> </ul> </li> </ul>
<b>/createchatroom</b>	POST	name; session.username	<ul style="list-style-type: none"> <li>• Generate a new 6-digit Chatroom Code by giving a random string that consists of lower case, uppercase letters and numbers</li> <li>• Determine if the newly generated is already taken by making query to user database <ul style="list-style-type: none"> <li>○ If taken, repeat the generation process</li> <li>○ If not taken, make query to create a new chatroom in the database with the supplied information, and send JSON response with string key, chatroom name and chatroom id for the user</li> </ul> </li> <li>• If unsuccessful, fetch error message</li> </ul>
<b>/appendChatroomToList</b>	POST	chatroomName; chatroomKey; session.username	<ul style="list-style-type: none"> <li>• “Upsert” the provided chatroom information (i.e. Chatroom Name &amp; Key) to the corresponding user identified by username <ul style="list-style-type: none"> <li>○ If successful, send JSON response with result: success for the user</li> <li>○ If unsuccessful, fetch error message</li> </ul> </li> </ul>
<b>/getchat/:key</b>	GET	params.key; session.username	<ul style="list-style-type: none"> <li>• Load chatroom information and all messages of that chatroom to the user by making query to user, chatroom and messages database <ul style="list-style-type: none"> <li>○ If successful, send JSON response with chatroomName, chatroomID, stringKey, largestMsgId and msgPack</li> </ul> </li> </ul>

Name	Route	Attributes	Function
			<p>(i.e. an array containing message id, isSelf toggle, isDateAppeared toggle, sender id, sender name, content, date and time for each message)</p> <ul style="list-style-type: none"> <li>○ If unsuccessful, fetch error message</li> </ul>
<b>/checkNewMsg/:key</b>	GET	params.key; session.username	<ul style="list-style-type: none"> <li>• Provide a quick and low network usage API to check if end user has the most updated messages for the chatroom current viewing by making query to chatroom and messages database               <ul style="list-style-type: none"> <li>○ If successful, send JSON response with a MsgId indicating the ID of the most recently created message at the corresponding chatroom</li> <li>○ If unsuccessful, fetch error message</li> </ul> </li> </ul>
<b>/postmessage/:chatroomid</b>	POST	content; date; time; params.chatroomid; session.userId session.username; session.displayName	<ul style="list-style-type: none"> <li>• Insert the provided message contents (i.e. content, date, time) with the corresponding chatroom information (indicated by params) and user information (indicated by session variables)               <ul style="list-style-type: none"> <li>○ If successful, send JSON response with message ID of the newly inserted message</li> <li>○ If unsuccessful, fetch error message</li> </ul> </li> </ul>

#### Database: mLab MongoDB

- Hosted on: `mongodb://<dbuser>:<dbpassword>@ds233238.mlab.com:33238/chat101`
- Free Sandbox account
- 0.5GB storage space

## User Interface Snapshots

- Sign In Page

A screenshot of a web browser showing the Sign In page for HKChatroom. The browser's address bar displays "Secure | https://hkchatroom.herokuapp.com". The page header includes "HKChatroom" and "Welcome!" on the left, and a "Register" button on the right. The main heading is "Sign In to Your Account". Below this, there are two input fields: "Username" and "Password". A blue "Sign In" button is positioned below the password field.

HKChatroom Welcome!

Register

### Sign In to Your Account

Username

Password

Sign In

- Registration Page

A screenshot of a web browser showing the Registration page for HKChatroom. The browser's address bar displays "Secure | https://hkchatroom.herokuapp.com". The page header includes "HKChatroom" and "Welcome!" on the left, and a "Sign In" button on the right. The main heading is "Register for an Account and Start Chatting". Below this, there are four input fields: "Username", "Display Name", "Password", and "Re-enter Password" (labeled "Confirm Password"). A blue "Register" button is positioned below the last input field.

HKChatroom Welcome!

Sign In

### Register for an Account and Start Chatting

Username

Display Name

Password

Re-enter Password

Register



## • Dashboard Page

HKChatroom Signed in as test01 [Sign Out](#)

### Join a Chatroom

Key  [Continue](#)

### Chatroom List

- Hong Kong**  
Code: V1q60d
- Macau**  
Code: j0KmLb
- Taiwan**  
Code: ocX7xC
- Discussion**  
Code: w0EOtD
- Chat01**  
Code: YW62PM
- Chat02**

## Welcome, Test Account 01! Select a Chatroom to start chatting.

### Create a Chatroom

Name  [Start a new Chatroom!](#)

## • Chatroom Page

HKChatroom Signed in as test01 [Sign Out](#)

### Join a Chatroom

Key  [Continue](#)

### Chatroom List

- Hong Kong**  
Code: V1q60d
- Macau**  
Code: j0KmLb
- Taiwan**  
Code: ocX7xC
- Discussion**  
Code: w0EOtD
- Chat01**  
Code: YW62PM
- Chat02**

## Macau Code: j0KmLb

Sun Feb 25 2018

Hello 15:05:40

**User 01**  
Number of use cases should be approximately equal to number of major processes described in the business requirements. You should thus have between 6-10 use cases as per current business requirements. Thus either you have not put everything in your business requirement, or more likely, you are over estimating. 15:06:19

We have in-app purchases, corporate campaigns and general advertising. And we are going to implement them orderly. 15:06:35

**User 01**  
\* A series of analysis has been conducted. We have made references to SnapChat and askFM as benchmark. 15:06:44

**User 01**  
\* Benefits are from sales 15:06:59

For Full-Screen advertisements, we believe at that point users get used to the app and visit regularly. 15:07:06

Type a message...

## Test Accounts

- Several accounts have been set up with the deployed application to facilitate views

No.	Username	Password	Remarks
1	test01	123456	Multiple Chatrooms, Chatroom named "Discussion"
2	test02	123456	Chatroom named "Discussion"
3	test03	123456	Clean account

## Remarks

- HKChatroom is deployed on free accounts of Heroku and mLab
- In case of any behavioural abnormalities, please contact administrator at [jeffreylchung@gmail.com](mailto:jeffreylchung@gmail.com) for assistance