Group members: Zhenyu Yuan, Lu Zhang

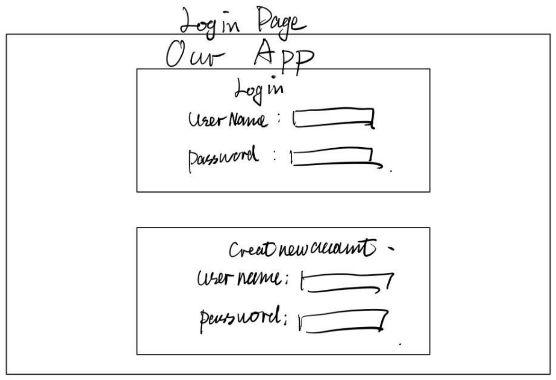
Overview:

Our group plans to make a Line or Wechat like messaging web application. This application will be a simple version of Line or Wechat, which will have a create account option, login and logout option, support single and group chat, link and picture share, user-customized background and font option, and we also plan to use ajax to make this application doesn't need to refresh the page when the user switches into different chat boxes.

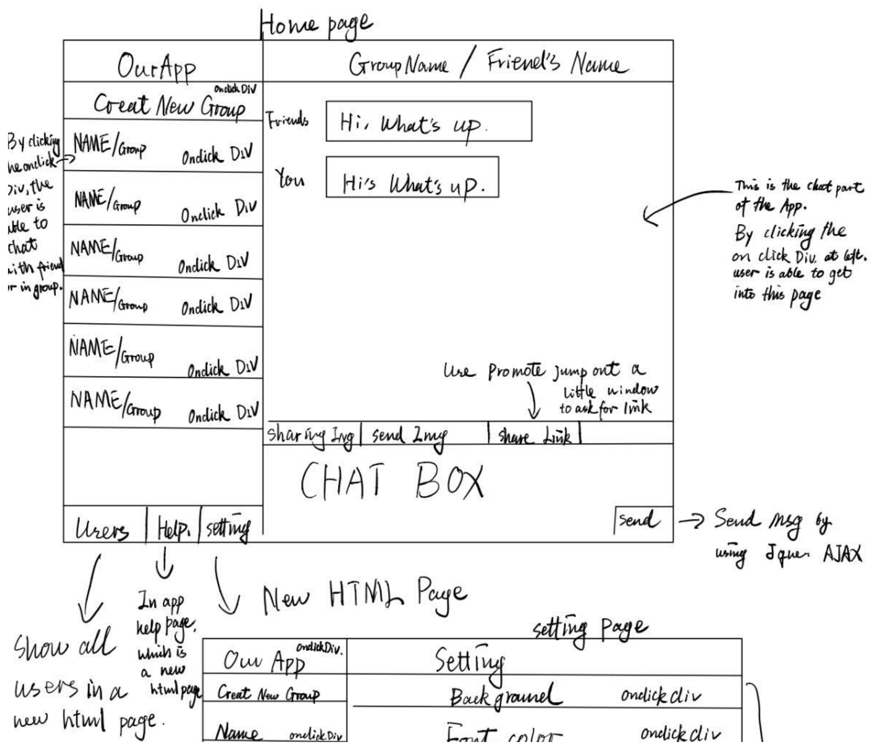
The database of this application is MongoDB, as the lecture video mentioned, we will also use hashing and salting to provide some basic security protection.

Frontend:

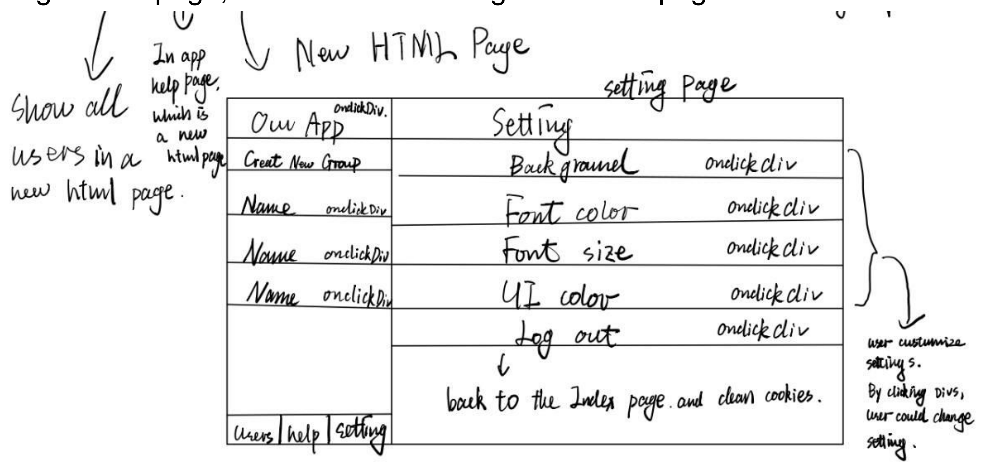
For the Frontend section, we are going to use 6 HTML pages to create the user interface. The first one is the login page, the user is able to create a new account and login on this page.



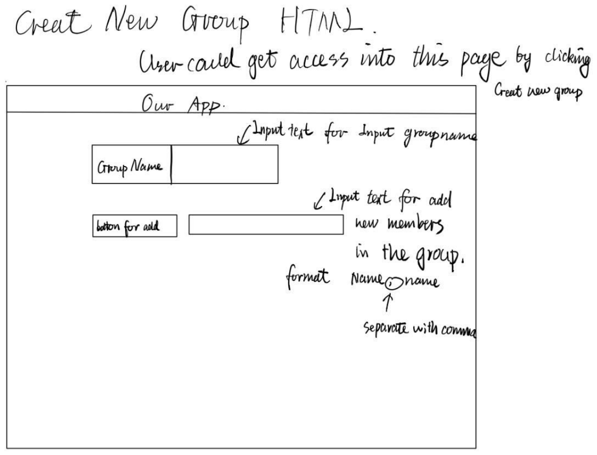
The second page is our main page, this main page is able to show chat boxes and friend list. The user is able to share links, imgs, and chat with other users and groups on this page.


Home page
OurApp
Creat New Group
onclick DIV
NAME/Group
Onclick DIV
By clicking the onclick div, the user is able to chat with friend or in group.
Group Name / Friend's Name
Friends
Hi, What's up.
You
Hi's What's up.
This is the chat part of the App. By clicking the on click Div at left, user is able to get into this page
Use Promote jump out a little window to ask for link
Sharing Img
send Img
share Link
CHAT BOX
Send
Send msg by using Jquer AJAX
Users
Help.
setting
Show all users in a new html page.
In app help page, which is a new html page
New HTML Page
setting Page
Our App
onclickDiv.
Creat New Group
Name
onclickDiv
Setting
Background
onclick div
Font color
onclick div

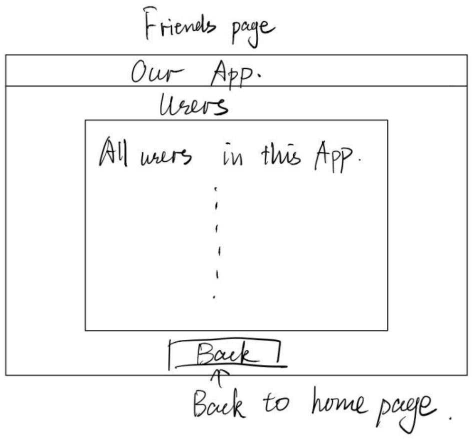
The third page is the setting page, the user is able to change font size, color, and background setting on this page, also the user can log out on this page.



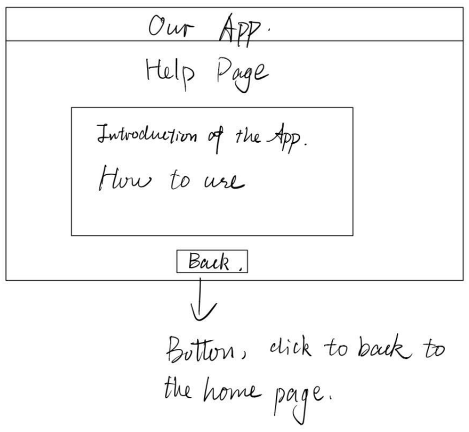
The fourth page is to create a new group page, the user can create a new group on this page by entering the group name and group members. This page can also add new members to an existing group.


Creat New Group HTML.
User could get access into this page by clicking Creat new group
Our App.
Input text for Input groupname
Group Name
Input text for add new members in the group.
botton for add
format Name, name
separate with comma

The sixth page is the page that shows all users.



And the last page is a help page, which will contain information about this app and how to use it.



Backend:

We will use a group of modules and APIs to implement the server and database(see below for details).

> Log in/out: This app will require users to login via cookies, salt, and hash. Authentication function before login to the home page

>modules: ‘express’, ‘cookie-parser’, ‘mongoose’, ‘body-parser’, ‘crypto’, ‘jimp’

> API:

‘/add/user’ (POST) add a new user to the app
‘/add/message’ (POST) send a new message the server, the server will store the message in the database
‘/get/message/:groupid’ (GET) return all messages in the group
‘/get/users’ (GET) return all users in this app
‘/get/groups’(GET) get all groups associated with the user
‘/create/group’ (POST) create a new chat group
‘/enter/group/:groupId’ (GET) show all messages in the groups
‘/settings’ (POST) the user can change its setting (like font, background color)
‘/logout’ (GET) clear the cookies

> Database:

```
var UserSchema = new Schema({
   username: String,
   salt: String,
   hash: String,
   // other user's setting like font, background color
});

var MsgSchema = new Schema({
   message: String,
   group: {type: Schema.Types.ObjectId, ref: 'Group'},
   from: {type: Schema.Types.ObjectId, ref: 'User'},
   to: {type: Schema.Types.ObjectId, ref: 'User'}
});


var GroupSchema = new Schema({
   members: [String],
   message: [{type: Schema.Types.ObjectId, ref: 'Msg'}]
});
```

Timeline:
Milestone

1. Index.html ---- including creating account, login and logout via cookie, salt and hash. And implement the API ('/get/users') and authentication function (about 1 hour)
2. Home.html ---- client-side, including some main buttons ('users', 'help'), we will divide the home page into several different modules (group list, chat window, and chatbox), and implement these modules step by step. (about 2 hours)
3. Implement 'create a group' (about 10 hours)
    a. Server-side: API '/create/group'
    b. Server-side: API '/get/group'
    c. Client-side: onclick div 'create a group'
    d. '/create.html'
    e. Implement 'group list', using API '/get/group'
    f. Switch chat windows between different groups/users.
    g. Click the back button to return to the home page.
4. Implement 'send message' (basic) (about 6 hours)
    a. Server-side: '/add/message'
    b. Client-side: call setinterval() to refresh chat window every second
    c. Client-side: chat box, send button call onclick () to send string message.
5. Implement 'send message' (advanced) (about 8 hours)
    a. Send images and share links.
6. Implement 'settings' button (about 5 hours)
    a. '/settings.html' (change font, font color, background color, ui color and provide a logout option).
    b. Click the back button to return to the home page.
7. Implement 'users'/'help' button (witch jumps to a new HTML page and shows information of all users/ the information of the application and how to use this application)(about 1 hour)
    a. Click the back button to return to the home page.