Due Date: 12/05/2018

Jie Chen([jic136@pitt.edu](mailto:jic136@pitt.edu))

Chia-Hsuan Hsieh(chh171@pitt.edu)

**Project Report**

**Project Name: Group Chat Room**

**Overview**

Our team implement a group chat room for multiple users. Users can login into their user profiles or register new accounts. When a new user visits this system, he/she should register a new account and use this account to sign in. After users log in, they can change their user nicknames and profile pictures. The system will record users’ personal information in database. The user can send text and picture to other users and check other users who are online.

**Assumptions**

1. Users can sign up a new account and sign in with their account.
2. Users must sign in if they want to join in the group chat.
3. Users can change their nicknames and profile pictures in their account.
4. Customers can check others whether they are online or not.

**Set up Environment**

**Prerequisites**

Before beginning, we make sure the development environment includes Node.js® and an npm package manager.

1. Node.js (required Node.js version 10.x.)

To check the version, run node -v in a terminal/console window.

1. npm package manager

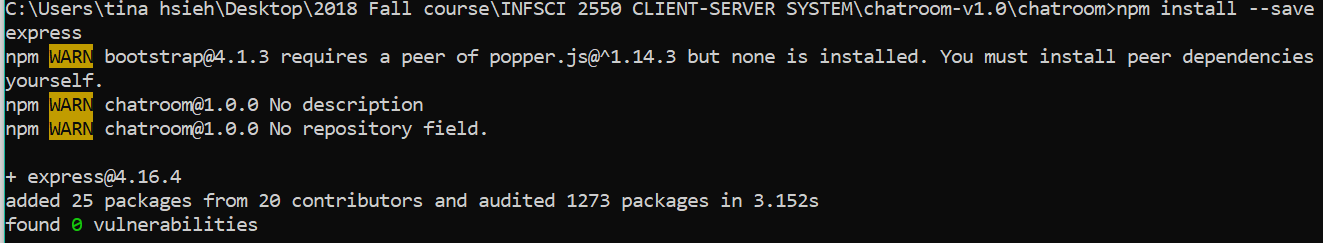
Our application depends on features and functionality provided by libraries that are available as npm packages. To download and install npm packages, we need an npm package manager.

This Quick Start uses the npm client command line interface, which is installed with Node.js by default. To check that you have the npm client installed, run npm -v in a terminal/console window.

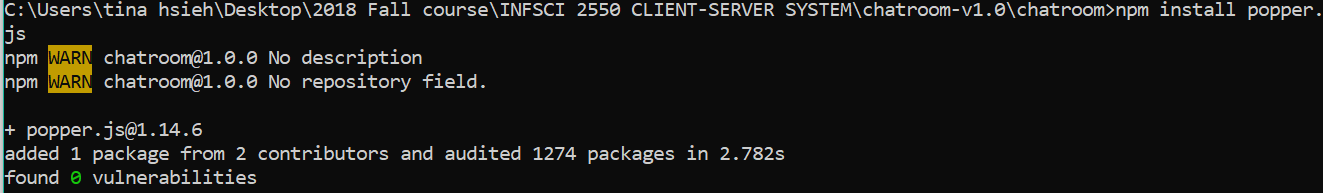
**Serve the application**

Go to the workspace directory and launch the application.

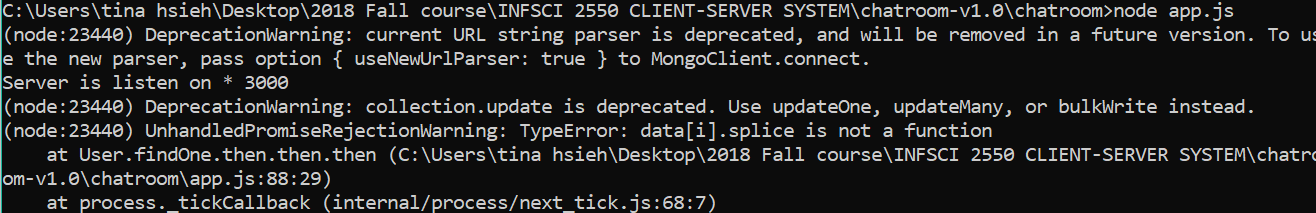
1. npm install --save express



1. npm install popper.js



1. node app.js



**Design and Connection**

In the front-end, we design three web pages for users on web browser:

In the login page, users can register, sign in and sign out. If the user does not input his/her username or password, our system will pop-up a warning page and show “Please input username!” / “Please input password！”.

In the register page, the user must input the username, the password, and retype the password. If the user does not input his/her username, our system will pop-up a warning page and show “Please input username!”; if the user does not input the wrong type of password or retype the wrong password, our system will pop-up a warning page and show “the password should be 6 - 12 digits or alpha！” or “password does not match”.

In the home page, users can change their profile nicknames and pictures on the sidebar, and see other users who are online. Users can chat with others on the center column, and the user can send pictures and texts. We use bootstrap CSS and some pictures to build the web outline.

In the back-end, we use Node.js and Express framework to build a server, use MongoDB which is a NoSQL database to store the data and use Socket.io to send and receive message:

Express is a fast, unopinionated and minimalist web framework for Nodej.js, by using Express developers can build a node.js server very fast.

MongoDB is a NoSQL database which provide many benefits for our application than relational database like MySQL. Using MongoDB is convenient, we do not need to design our table in database, because we store data as document in MongoDB. In our project, we use mongoose which is an elegant object data model (ODM) for MongoDB. Mongoose provides a straightforward, schema-based solution to model our data.

Socket.io is a library for real-time web application. It provides real-time and bi-dimensional communication between client and server.

First, we build login and signup part. In this part, our server receive data from front-end that contains username and password. Since each username should be unique, if a new user wants to use the username which already been taken, server will return an error message to front-end that tell the user that this username has been used, otherwise we store the username and password in our database. Login have the same logic with signup.

Then, we build our chat part which is the core part in out project. This part contains three small model – show online user, edit user information and chat.

**Show online user**

If users are in online status, we show them at “Online User” window. If an offline user become online, we broadcast the information “a new user online” to all of online users in this chatroom.

**Chat**

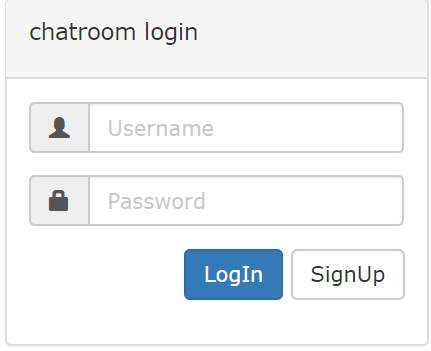
Users can send text messages and photos to chatroom. In the back-end, if the server receives a message, it will broadcast this message to all of users in the chatroom. Sending text messages are easy, but sending photo messages are different. When user decide to send a photo, we use FileReader which provided by Javascript to read photo as a base64 file, then send it to server.

**Edit user information**

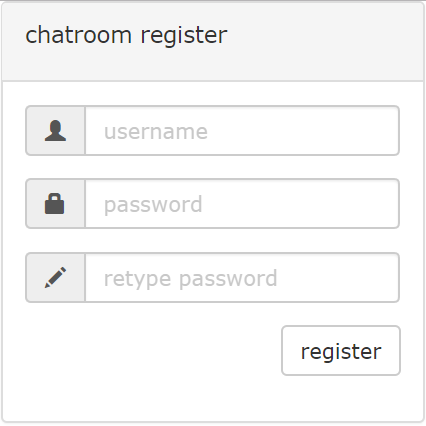
Users can change their personal information like photo. The method that user can change their photo is like sending a photo to chatroom. What we do just store this photo in our database which can display when user login.

**System Implementation**

**User login:**

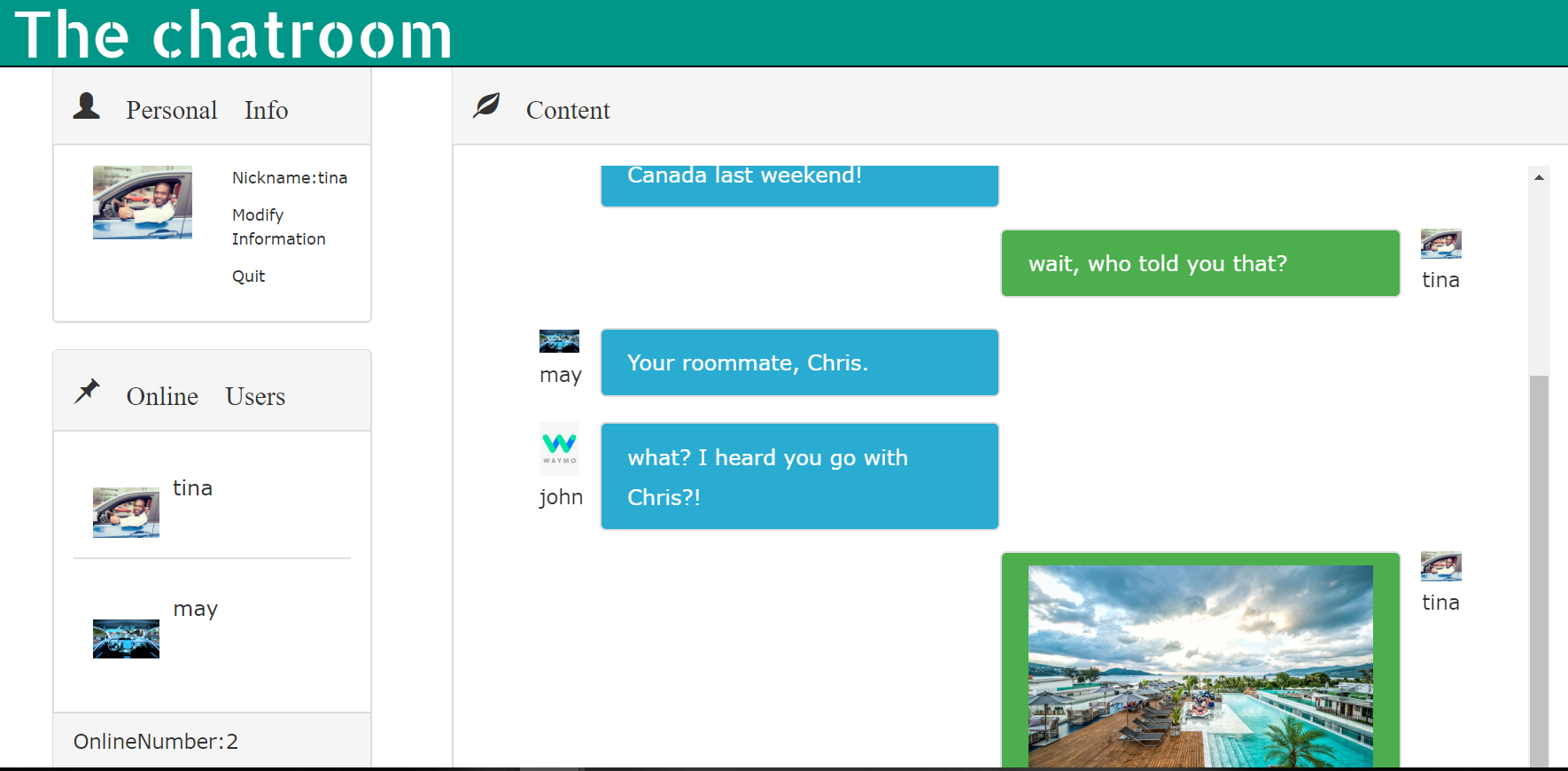


**Register:**

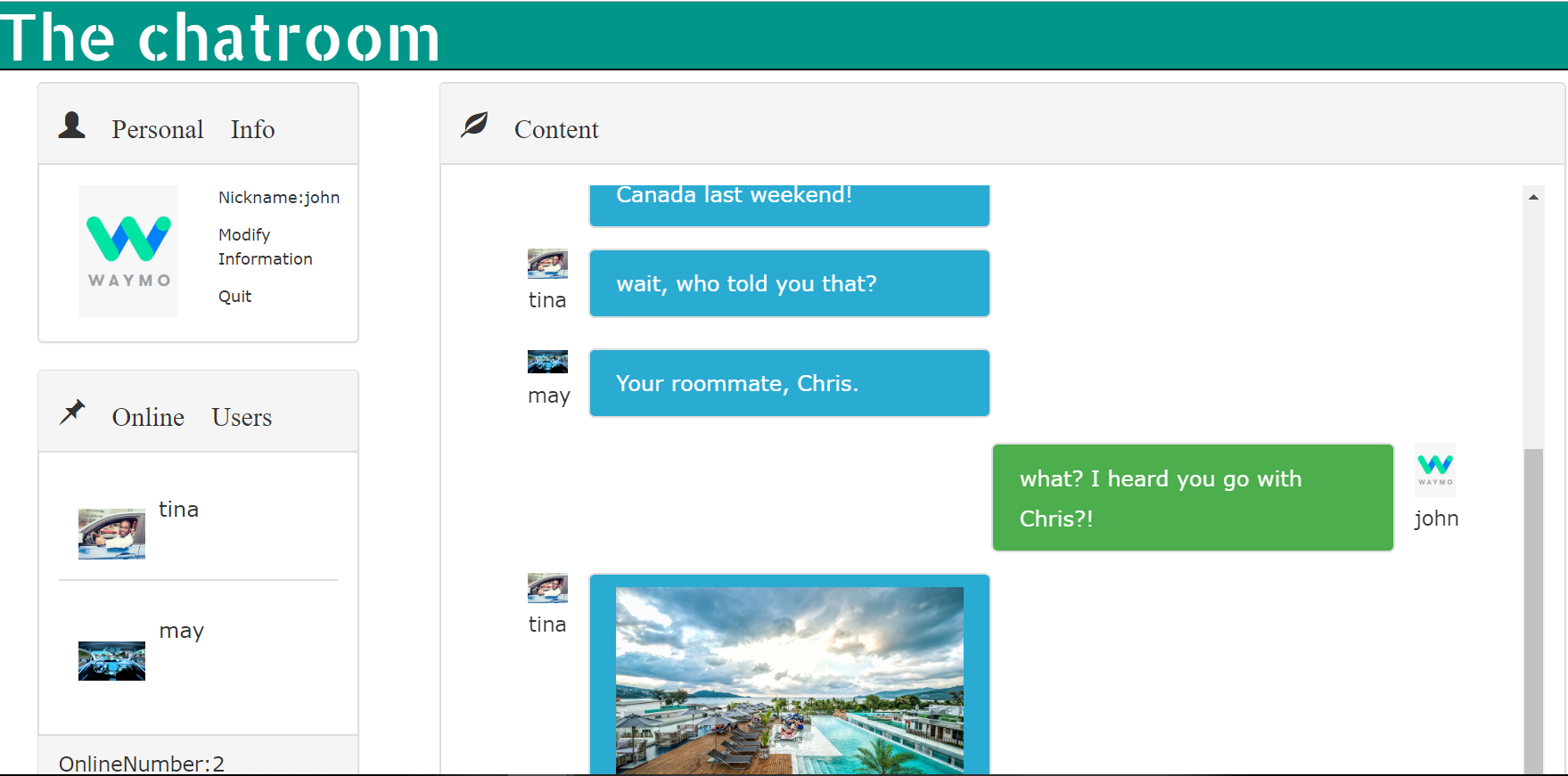


**User home page:**

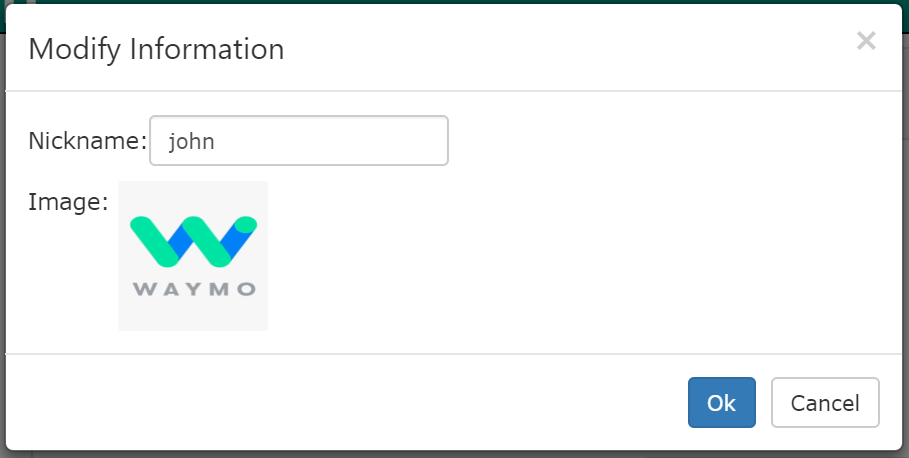
**User 1:**



**User 2:**



**Modify personal information:**



**Limitations and Future Improvements**

**Limitations:**

1. Users cannot modify their password.
2. Our systems do not support users to chat one by one.

**Improvements:**

1. Users can sign up and sign in by using their social media.
2. Users can modify their password.
3. User can chat with others individually.