Embedded Systems Labs

Lab#1 - Chat room

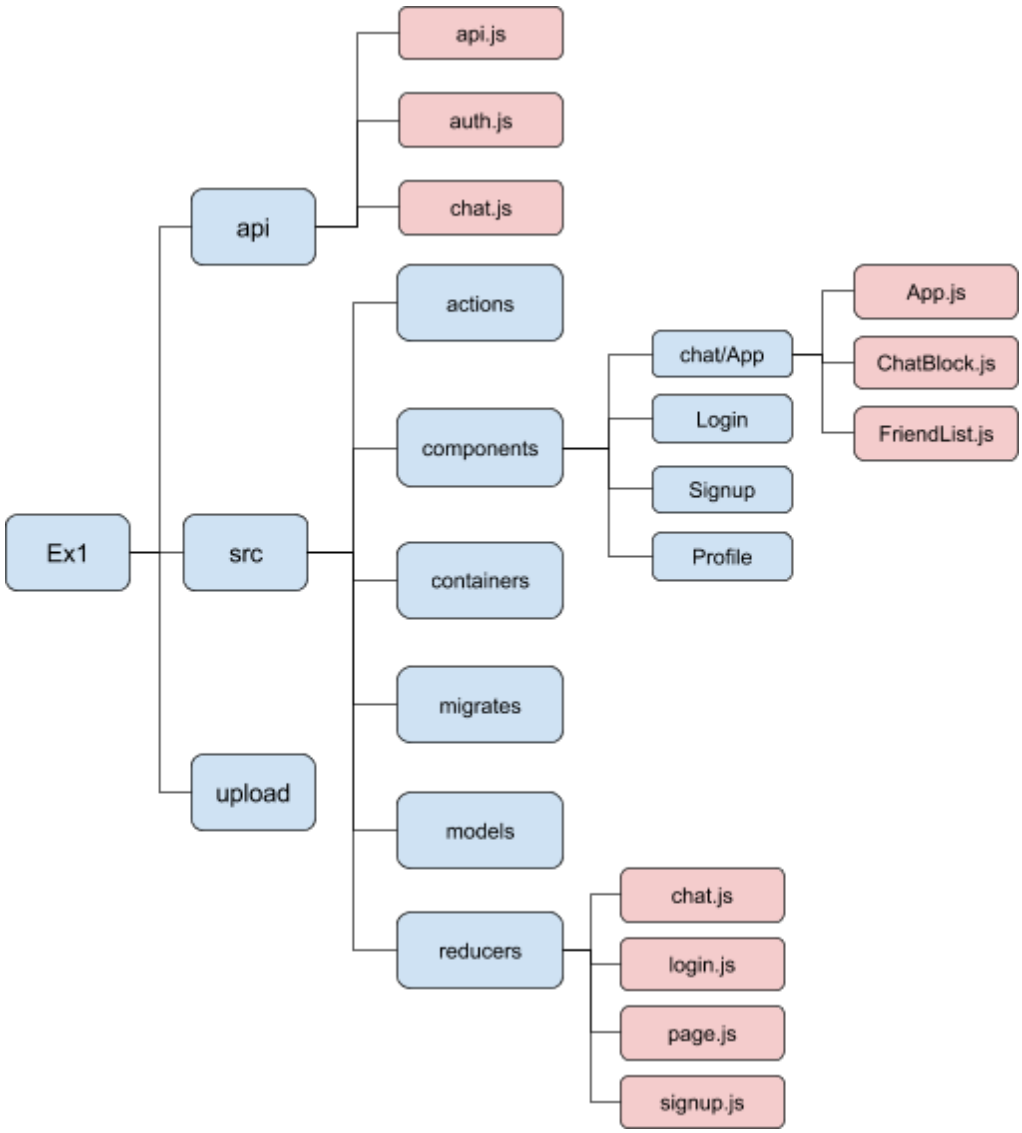
組員：陳昊 卓伯鴻 陳景由

1. Introduction

This project provide a web chat room and messenger like interface that we can first register an account and send messages to others, including broadcast messages to several people in a group. In addition to the basic functionality, we support sending photos and attachments to both individuals and groups.

2. Structure

First we describe our management of our source code, the main structure of our work is shown by the following flow chart. (Some details are not shown in the flow chart)



- api: we use react-express middleware to handle the request, use multer middleware to handle the upload files and use socket.io to handle real time communicate request.
    - auth.js: handle request sent from Login & Signup
      use express session to save the info. of online user.
    - chat.js: handle request sent from chat/
      we save the socket.id of online users, if there is a msg need to be sent to any online user, he will get it realtime.
- src:
    - actions: define all actions that can be done
    - components:
        - chat/App: (get props from containers/ChatContainer) decide which component should be sent
            - App.js: export all components.
            - ChatBlock.js: handle chatting history.
                - message(msg) array: msg[0] is the type of the msg, divided into four type, ‘text’, ‘image’, ‘audio’, ‘video’ , will be displayed differently.msg[1] us the content, if the msg[0] is ‘text’, msg[1] is real content, or it will be the path to the file. It will be refresh in any time, user can also leave messages to his friends.
                - is read: data bases will save the time you friend read your msg, but it only will be refresh when user reload the page.
                - time: the time the msg be store into the database is be stored, and show beside the msg.
            - FriendList: show friend list and the latest message. (User can add friend and choose who they want to chat with.)
                - the friend list will be sorted according to the time of latest message.
                - user can click different friend to switch the friend he want to chat with, in the meantime, user will sent a post request to tell server he had read the msgs, so when his friend open the chat, his friend will know the user had read his messages.
            - Config.js: has the drop selector
                - logout: sent a post request to the server, and destroy the cookie in api/auth.js
                - rename their group name:User can reset the name of groups, and front-end change accordingly.

Other user should reorganize the page to refresh the change.
        - add friend: User can invite his friend to jion the chat with another friend.
    - Login: (get props from containers/LogicContainer)
        - As we enter the page, website will sent a request to check if the user has been logged in. If not, user should log in, or the website will log in spontaneously.
    - Signup: (get props from containers/SignupContainer)
        - sent a JSON.stringify(object) to server, and server will check whether the name or email have been used. If not, it will create a new account for the user.
  - containers: combine proprs and components
  - migrates: build databases table by sequlizejs
  - models: build databases object model
  - reducers: handle actions sent from components
    - chat.js: handle actions sent from chat/App components.
    - login.js: handle actions sent from Login components.
    - page.js: handle actions sent from App components.
    - signup.js: handle actions sent from SingUp components.
- uploads: save the uploaded files

The main middleware used in this project to connect frontend and backend is “redux”, and a part of reactjs. Although if we use redux , we should store info. In only one ‘store’ and change store only by dispatch action, but we’re not very familiar with the asynchronous middleware lick redux-thunk in the beginning, so we sent request in react way instead of redux way.

3. Features

There are several special features in this project, we’ll introduce them briefly in the following.

- Account System: Allow user to register and identify the people to chat with, provide an user friendly interface to let us easily choose who we want to chat with and create group chat, further prevent annoying messages from strangers.

- I/O-socket System: Server can broadcast system messages to client, and can initiative notify the recipient that there’s a new incoming message without the recipient client sending a request to the server.
- Multi-media System: Allow user to upload images and some attachments and send to other user directly, also provided in broadcast group chat system.

- Database System: Use MySQL as database through Sequlizejs in this project. Store the chat history, including attachments and images uploaded by users. Provide an user friendly interface that allow user to select the desired history chat room and maintain the chat history that the most recently activating chat room is always on the top.
- No-privacy System: Show that the sent messages are read or unread by others.

Screenshots: