ITCS212 Web Programming

Project 2

By

Miss Wongchanok Devakula Na Ayudhya	6188050
Mr. Kittikorn Keeratikriengkrai	6188086
Miss Nattawipa Saetae	6188089
Mr. Kasadin Wisetprasit	6188102
Mr. Eijiro Amagasa	6288168

Faculty of Information and Communication Technology

Mahidol University

2019

What have you done for Web Accessibility?

- **Font size**
- **Can use in many device (Responsible)**

How is your webpage friendly?

- **Easy to use**
- **Have a large button**
- **Have low time to interactive (the amount of time it takes for the page to become fully interactive.**

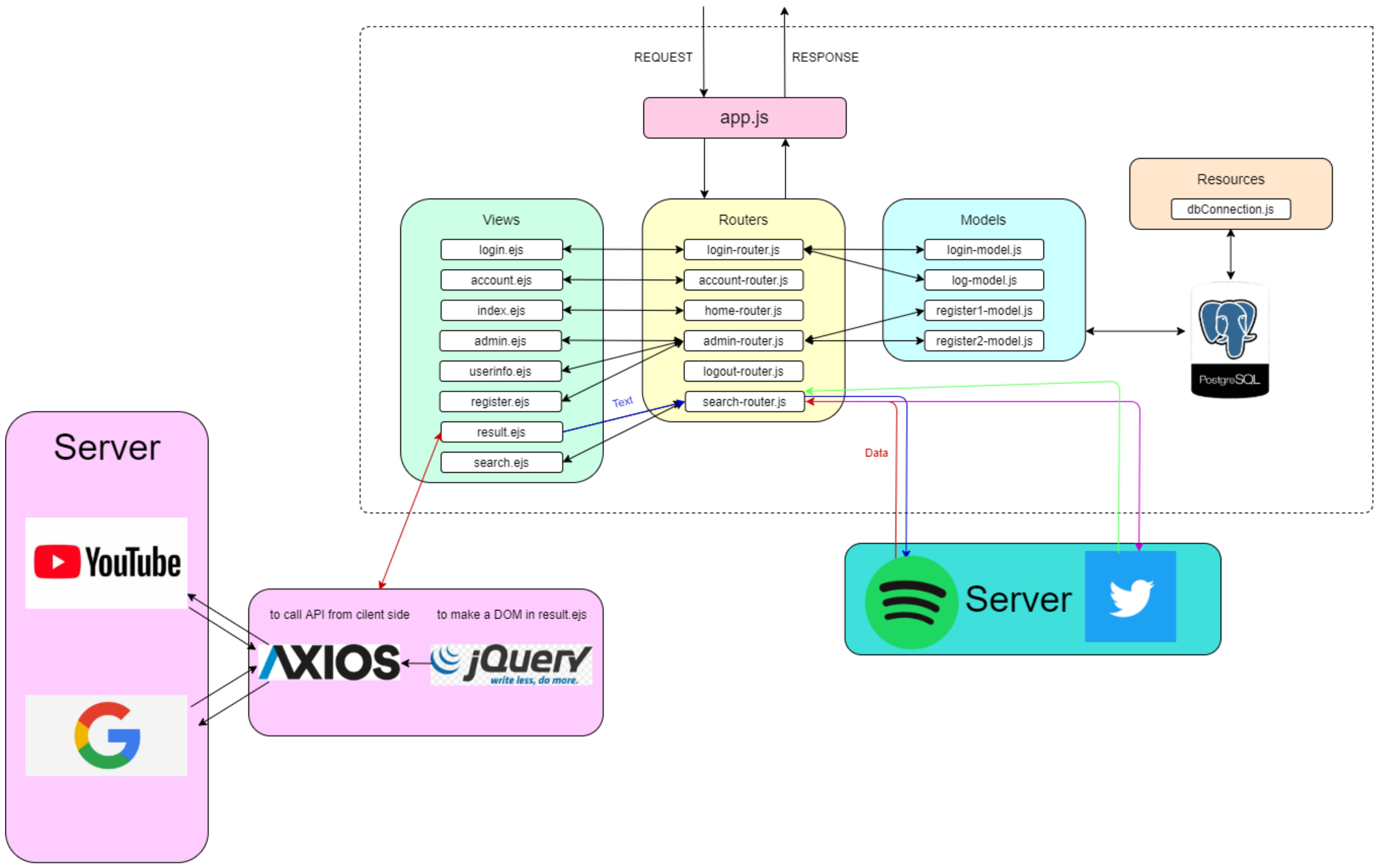
How can you interact with all web services?

User will send a request that is a name of video or movie to search then it will receive this one as json next we use API to search for Google, YouTube, Twitter, and Spotify and return in json. Lastly, we make that json result to be visible for human and make it more beautiful and easier to look.

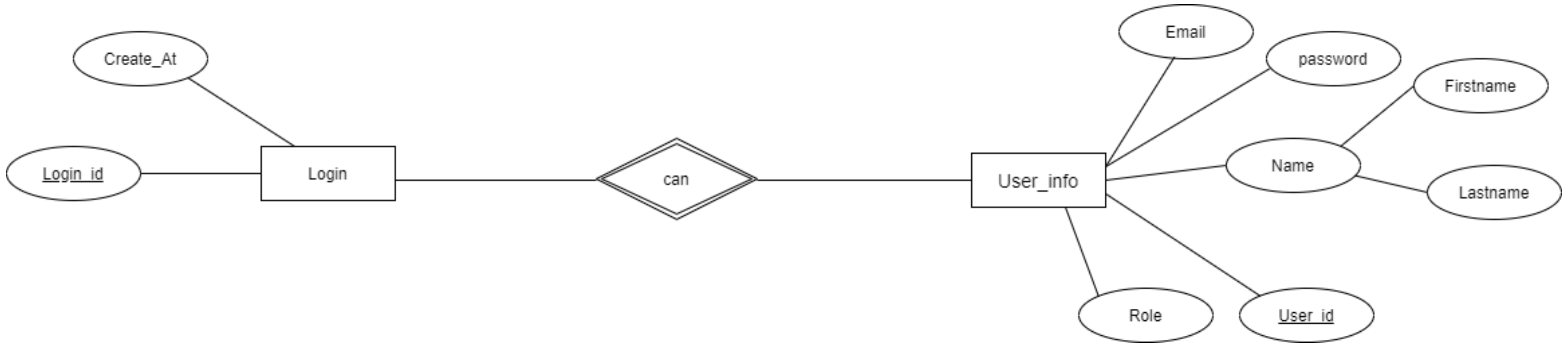
About the user login information, our team uses local database for now. All of SQL is written by PostgreSQL. In this web service user information which is containing username, role, email, and password is stored and that information would be referred every login request and the user want to see the account information. Moreover, when the user login to the system the time stamp will be shown up.

※the data base is running on Japan standard time, so there is time difference between Thailand and Japan.

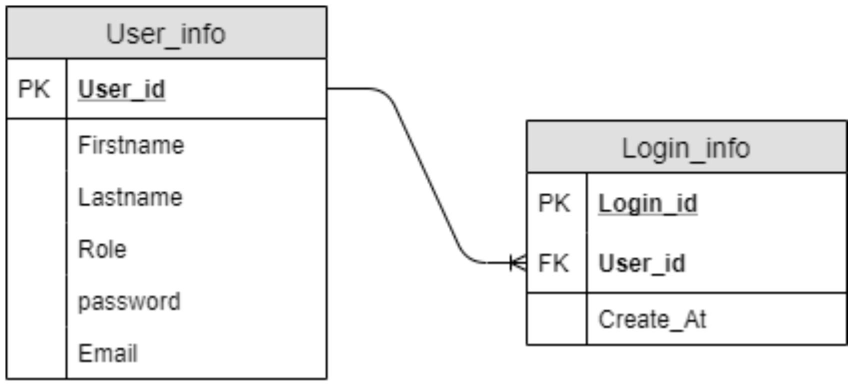
The architecture of your system.



ER-Diagram



Chen Notation



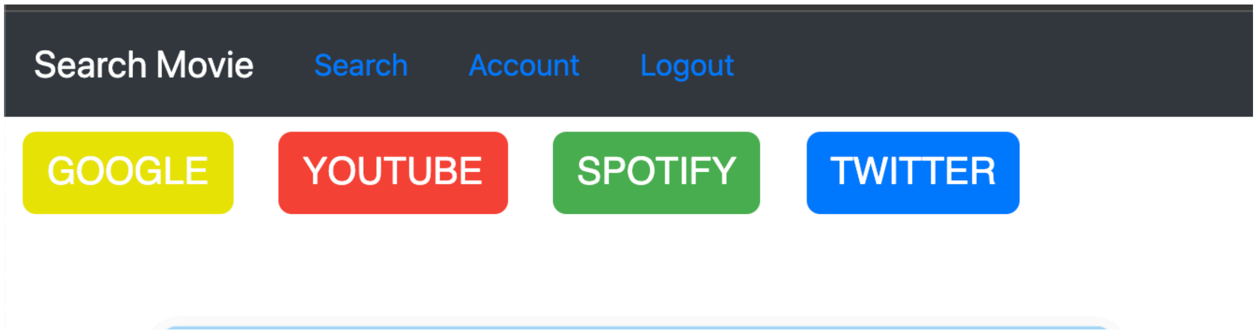
Crow’s Foot

The structure of your code

In our project folder, we have roughly **models** which contains the execute code on the database, **node_modules** which contains the information of npm modules our web service will use, **resources** which contains the file to connect database, **views** which contains the ejs files, and app.js which is the express router.

Usage of general User

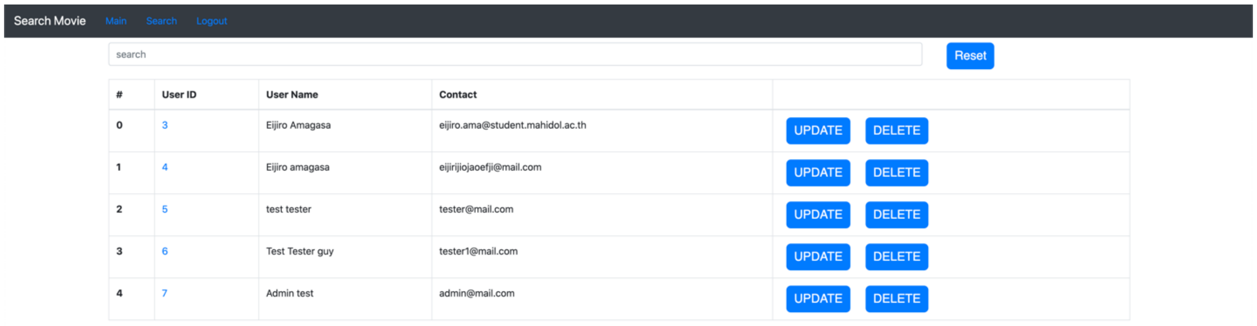
In the top directory, index.ejs will be displayed which is routed by "home-router.js", and general user will proceed to login page which is routed by "login-router.js". After filling the information, general users will go to search page which is routed by "search-router.js" and user will type any query which should be the name of movie. After user press search button, result page will be displayed. In this page, user can change the information within the data from "google custom search", "youtube", "twitter", and "spotify". This changing function is controlled by "Jquery.js" which is stored in public folder in the source. In this jquery file, it will call youtube data API and google custom search API as client side. Moreover, in "seach-router.js" which will get the query from a user will call twitter search API and spotify API as server side, and the response from those API will be passed to "Jquery.js". This is the normal usage for the general user.



※ **Result page will show the four buttons at the top to change the displaying data.**

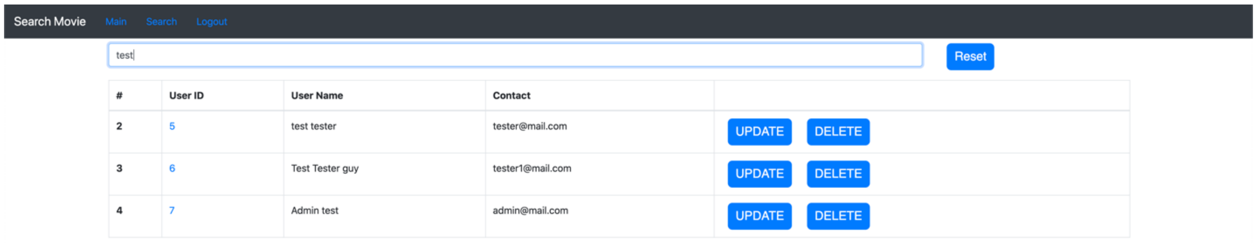
Usage of administrator(admin) User

In the login page, an admin user will check the checkbox to specify that the user would try to login as admin account. If the request is accepted, admin page which is generated by "admin.ejs" will be displayed. In this page, the admin can do actions within register new account and search account. When the user go to register new account the admin will fill the form powered by “register.ejs”, and if the email address is not duplicate within the existing data on the database and if the information is following the regulation, the new account will be stored in the database. On the other hand, the admin can search users on the page constructed by “userinfo.ejs”.



#	User ID	User Name	Contact	
0	3	Eijiro Amagasa	eijiro.ama@student.mahidol.ac.th	UPDATE DELETE
1	4	Eijiro amagasa	eijirijiojaoefji@mail.com	UPDATE DELETE
2	5	test tester	tester@mail.com	UPDATE DELETE
3	6	Test Tester guy	tester1@mail.com	UPDATE DELETE
4	7	Admin test	admin@mail.com	UPDATE DELETE

The admin will see the page like above, and the admin will be able to search by any data by typing information on the text box on the top.



#	User ID	User Name	Contact	
2	5	test tester	tester@mail.com	UPDATE DELETE
3	6	Test Tester guy	tester1@mail.com	UPDATE DELETE
4	7	Admin test	admin@mail.com	UPDATE DELETE

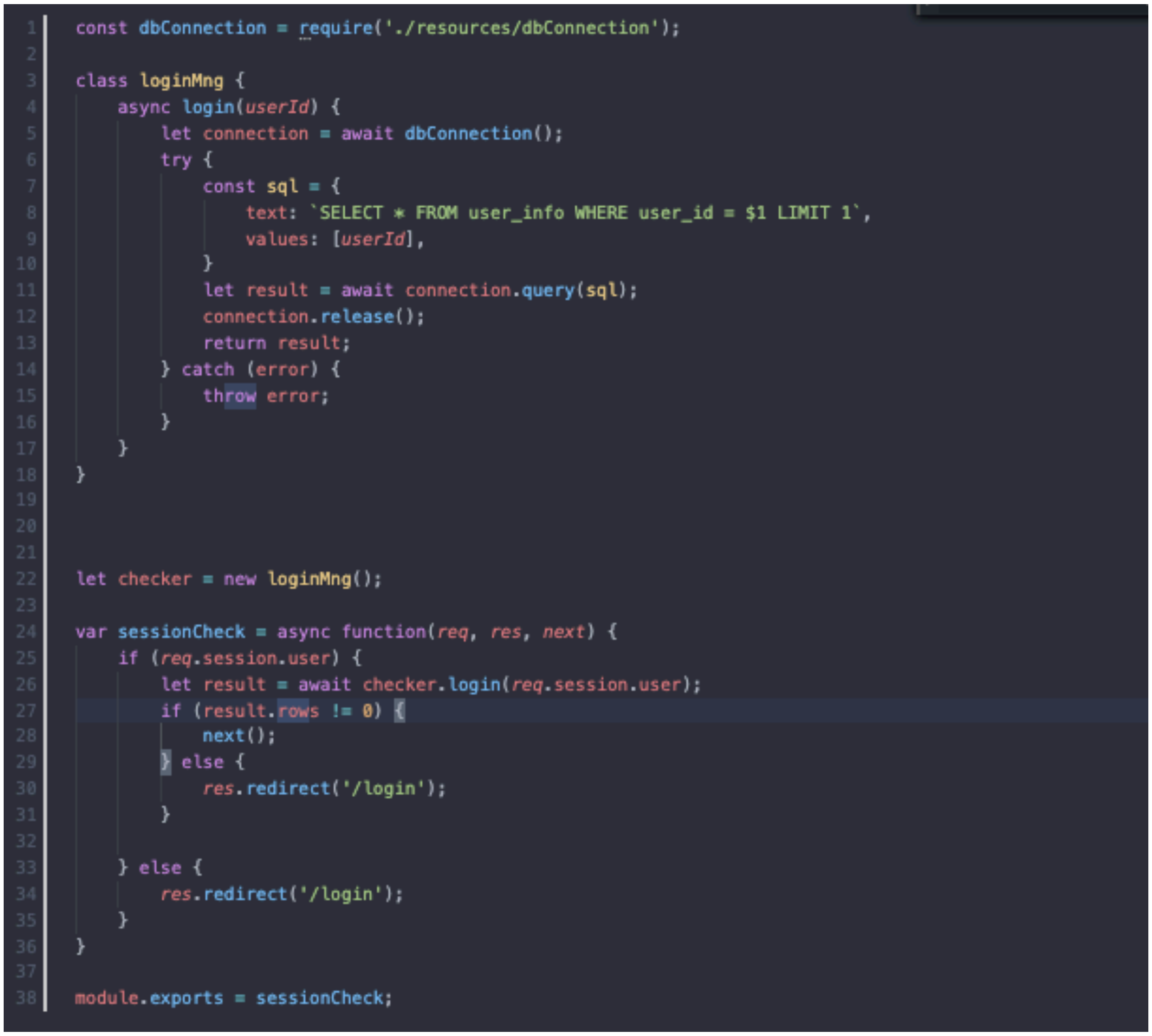
The table would be updated by every time after typing any key.

In this page, the admin can do other actions like update information and delete an account.

In addition, admin user can search movie as well as the same as general user.

User management

In this web service, all users should be stored in the database, and login information will be stored in the web-browser once a user successfully login to this service. In this point our group uses “express-session” module which will interact with session data storage. The session will only keep user id information, and this id will be used every time for recognizing whether the user has permission to access to search page and admin page. The all registered user will be able to pass the “setUser.js” which is stored in the top directory. This file will just refer the data from the database, especially “user_info” table. In addition, admin user which has a role as “admin” status can pass the “setAdminUser.js” which is stored in the top directory as well. This is the system that our webservice routing the different page to the different type of user.



```javascript
const dbConnection = require('./resources/dbConnection');

class loginMng {
    async login(userId) {
        let connection = await dbConnection();
        try {
            const sql = {
                text: `SELECT * FROM user_info WHERE user_id = $1 LIMIT 1`,
                values: [userId],
            }
            let result = await connection.query(sql);
            connection.release();
            return result;
        } catch (error) {
            throw error;
        }
    }
}



let checker = new loginMng();

var sessionCheck = async function(req, res, next) {
    if (req.session.user) {
        let result = await checker.login(req.session.user);
        if (result.rows != 0) {
            next();
        } else {
            res.redirect('/login');
        }

    } else {
        res.redirect('/login');
    }
}

module.exports = sessionCheck;
```

This is “setUser.js” which contains both model and verifying function.