Final Project

Dec 21, 2018 (Fri)

Eunggu Yun

1. System Environment

1) OS: macOS High Sierra 10.13.6(17G4015)
2) Node.js: v10.13.0
3) MongoDB: v4.0.4 (distarch: x86_64)
4) Chrome: 71.0.3578.98 (Official, 64bit)

2. How to execute the web server

1) Install and configure MongoDB and Node.js
2) Run MongoDB

```
> mongod
```

3) Move to project root directory
4) Run Node.js

```
> node app.js
```

3. Modules

1) pug

   Pug is a template engine compatible with express. It compiles pug files into html files, which gives users dynamically changing web pages.

2) express

   Express is a web application framework for Node.js. It provides request/response handling methods for routing.

3) body-parser

   Body-parser is a body parsing middleware for Node.js. It parses incoming request bodies and makes req.body property

4) cookie-parser

   Cookie-parser is a cookie parsing middleware for Node.js. It parses cookie header and makes req.cookies with an object keyed by the cookie names

5) mongoose

   Mongoose is an ODM library for Node.js and MongoDB. It provides object modeling tool designed to work in an asynchronous environment.

4. Implementation Details

1) GLS (General)

① Outline

When a user requests a GLS page, the web server gets ID from the user's cookie. Then it searches the user database to determine whether the user is an administrator or a student. If the ID is not registered, it denies the access. If the ID is an administrator, it searches the course database and renders the GLS page with the course information. If the ID is a student, it searches the course database and renders the GLS page with the student information and the course information. The Pug template engine compares the student's course(class) information (course bag, registered) and the course information. Then it determines which information should be added to each page. When the GLS page is loaded on user's web browser, the JavaScript checks a hash in a URL and decides which GLS page should be shown.

② Home

This page uses #H as a hash tag (/gls, /gls#H). This page doesn't require reloading.
It only shows the image about GLS.

2) GLS (Admin)

① Add Course

This page uses #AC as a hash tag (/gls#AC). This page doesn't require reloading.

When the user presses an "Add Course" button, the JavaScript checks if all input fields are filled. If some input fields are not filled, it alerts to user and doesn't send the form. If all input fields are filled, it sends POST request to "/addcourse" with form data.

When the web server gets the request, it checks the course database to determine the course(class) ID duplicated or not. If the course(class) ID is duplicated, it sends "duplicate" response. If the course(class) ID is available, it adds the course to the course database and sends "success" response.

If the web server sends "duplicate" response, it alerts to user and doesn't make any change. If the web server sends "success" response, it moves URL to "Course List" page (/gls#CL) and reloads the page to reflect the changed data.

② Course List

This page uses #CL as a hash tag (/gls#CL). This page requires reloading (automatically reloaded).
It just shows the courses on the course database. The contents are rendered at the web server.

3) GLS (Student)

① Student Info

This page uses #SI as a hash tag (/gls#SI). This page requires reloading (automatically reloaded).
It just shows the student information. The contents are rendered at the web server.

② Course Info

This page uses #CI as a hash tag (/gls#CI). This page requires reloading (automatically reloaded).
It just shows the courses registered by the student. The contents are rendered at the web server.

③ Course Bag

This page uses #CB as a hash tag (/gls#CB). This page requires reloading (automatically reloaded).
It shows the courses on the course database and the courses saved in the students' course bag.
When the user presses a "Put in your bag" button, the JavaScript sends POST request to

"/coursebag/putinbag" with student ID and course(class) ID.

When the web server gets the request, it checks the student's course bag to determine the course is in the course bag or not. If the course is already in the course bag, it sends "failed" response. If the course is not in the course bag, it adds course(class) ID to the student's course bag and sends "success" response.

If the web server sends "failed" response, it alerts to user and doesn't make any change. If the web server sends "success" response, it reloads the page to reflect the changed data.

When the user presses a "Put out from your bag" button, the JavaScript sends POST request to "/coursebag/putoutbag" with student ID and course(class) ID.

When the web server gets the request, it checks the student's course bag to determine the course is in the course bag or not. If the course is not in the course bag, it sends "failed" response. If the course is already in the course bag, it removes course(class) ID from the student's course bag and sends "success" response.

If the web server sends "failed" response, it alerts to user and doesn't make any change. If the web server sends "success" response, it reloads the page to reflect the changed data.

4) Sugang

When a user requests a Sugang page, the web server gets ID from the user's cookie. Then it searches the user database to determine whether the user is an administrator or a student. If the ID is an administrator or not registered, it denies the access. If the ID is a student, it searches the course database and renders the GLS page with the student information and the course information. The Pug template engine compares the student's course(class) information (course bag, registered) and the course information. Then it determines which information should be added to the page.

It shows the student information, the courses registered by the student, and the courses saved in the student's course bag.

When the user presses a "Register" button, the JavaScript sends POST request to "/sugang/register" with student ID and course(class) ID.

When the web server gets the request, it checks the student's course register list to determine the student registered the course or not. If the student already registered the course, it sends "failed" response. If the student didn't register the course, it adds course(class) ID to the student's course register list, decreases student's available credit, and increases course's registered student count. Then it sends "success" response.

If the web server sends "failed" response, it alerts to user and doesn't make any change. If the web server sends "success" response, it reloads the page to reflect the changed data.

When the user presses a "Cancel" button, the JavaScript sends POST request to "/sugang/cancel" with student ID and course(class) ID.

When the web server gets the request, it checks the student's course register list to determine the student registered the course or not. If the student didn't register the course, it sends "failed" response. If the student already registered the course, it removes course(class) ID from the student's course register list, increases student's available credit, and decreases course's registered student count. Then it sends "success" response.

If the web server sends "failed" response, it alerts to user and doesn't make any change. If the web server sends "success" response, it reloads the page to reflect the changed data.