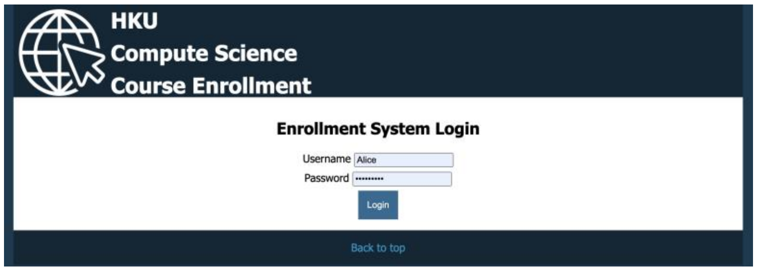
# COMP3322B Modern Technologies on World Wide Web

Lab 4: Node.js basics, AJAX

## Overview

In this lab exercise, we will implement a simple course enrollment system, in which the client side uses AJAX to send requests and the server side uses Express.js to respond. A user first logs in with a valid username, and then the page will show a course table. The user can then enroll in a course and observe the enrolment status update in the course table. Please refer to the following screenshots:



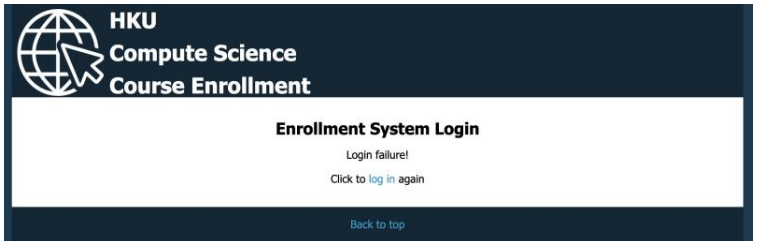
Enrollment System Login

Username Alice
Password ••••••••••

Login

Back to top

*Login page*



Enrollment System Login

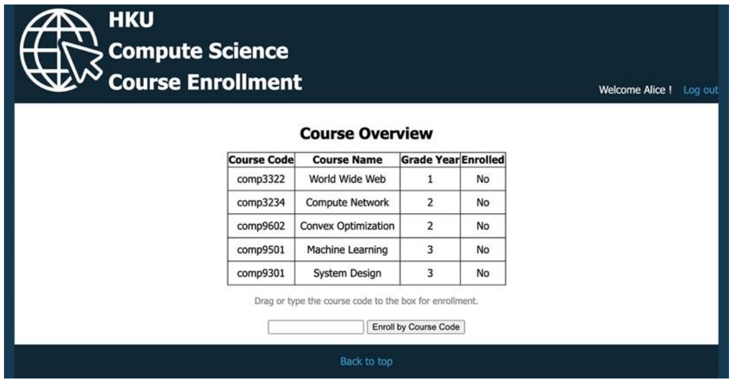
Login failure!

Click to log in again

Back to top

*Redirect to a login failure page if the username and password are invalid.*

***Figure 1. Login with session management.***



HKU
Compute Science
Course Enrollment

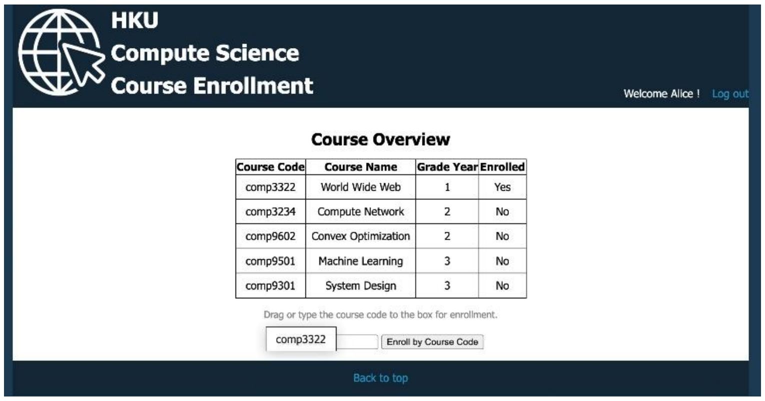
Welcome Alice ! Log out

Course Overview

| Course Code | Course Name | Grade Year | Enrolled |
|---|---|---|---|
| comp3322 | World Wide Web | 1 | No |
| comp3234 | Compute Network | 2 | No |
| comp9602 | Convex Optimization | 2 | No |
| comp9501 | Machine Learning | 3 | No |
| comp9301 | System Design | 3 | No |

Drag or type the course code to the box for enrollment.

*After login, the course page will be displayed, showing a course table and a course enrollment box. The navigation bar (in top right corner) will display a welcome message and a logout link (click and go back to the login page).*



HKU
Compute Science
Course Enrollment

Welcome Alice ! Log out

Course Overview

| Course Code | Course Name | Grade Year | Enrolled |
|---|---|---|---|
| comp3322 | World Wide Web | 1 | Yes |
| comp3234 | Compute Network | 2 | No |
| comp9602 | Convex Optimization | 2 | No |
| comp9501 | Machine Learning | 3 | No |
| comp9301 | System Design | 3 | No |

Drag or type the course code to the box for enrollment.

comp3322

*One can drag a course code from the table into the enrollment input box, or type the code into the box. Click the "Enroll by Course Code" button, and the respective course's "Enrolled" status in the course table will be updated upon successful enrollment with the server side.*

***Figure 2. Course enrollment***

Note: Due to compatibility of difference browsers, the same CSS code may lead to different display; we will check the lab in the Chrome browser. Therefore, you are recommended to use the Chrome browser to develop/test your page as well.

## Lab Exercise

## Part 1. Preparation

### Step 1. Download the code templates from Moodle

Download "**lab4_materials.zip**" from HKU Moodle, and extract it to a folder. In this folder, you will find 2 JavaScript files (“**index.js**”, “**app.js**”), 2 CSS files (“**main.css**” and “**basics.css**”), 3 HTML files (“**login.html**”, “**courses.html**”, **“fail.html**”), and 1 Image (“**logo.png**”). In this lab, we will ONLY edit “**courses.html**”, “**index.js**” and “**app.js**”, and keep other files unchanged. The HTML and JavaScript files can be edited using any code editors or IDEs.

### Step 2. Create an Express app skeleton

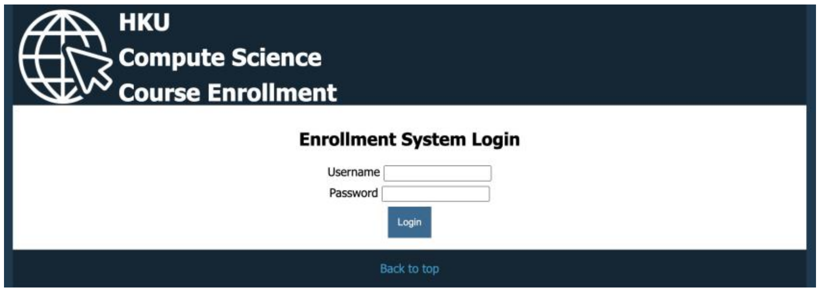
Follow step 1 to step 3 in the handout “setup_nodejs_runtime_and_example_1.pdf” to create an Express app. Move files extracted from "**lab4_materials.zip**" to the corresponding subdirectory: (1) move “**index.js**” to “public/javascripts/”; (2) overwrite the original “**app.js**” in the Express app directory with the “**app.js**” we provided; (3) move “**main.css**” and “**basic.css**” to “public/stylesheets/”; (4) move “**login.html**”, “**courses.html**”, **“fail.html**” to “public/”; (5) move “**logo.png**” to “public/images/”. If you check out contents of the HTML files, you will find that they are linking to the respective CSS, JavaScript and Image files.

## Part 2. Implement login validation.

### Step 3. Process HTTP GET request for http://127.0.0.1:8081/

Open “**app.js**”. Complete the callback function of route **app.get(‘/’, (req, res) =>{…})**: if the session variable loginName has been set, send “courses.html” to the client; otherwise, send “login.html” to the client.

After this step, run the Express app using “node app.js”, and visit http//127.0.0.1:8081/ to check the page out. The page should show up like the following:

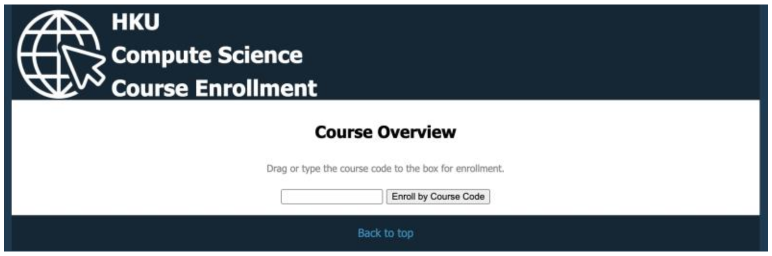


Note: Everytime after you have modified **app.js**, you need to re-launch the server program for testing.

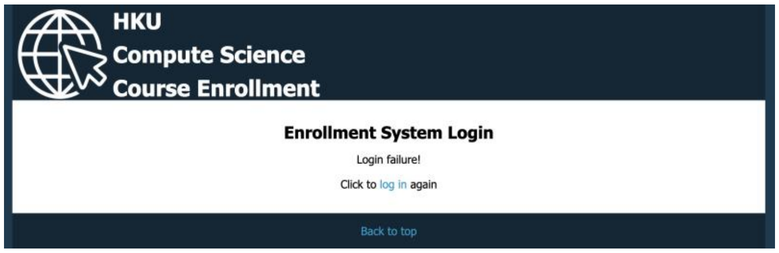
Step 4. Validate the user and respond.
In “**login.html**”, find the <section> element. We can see a <div> container with a <h2> heading which reads “Enrollment Sysetem Login”. Right under the heading, we have a **<form>** element with attributes: (1) **action** of “http://127.0.0.1:8081**/login_post**”, and (2) **method** of “**POST**”. Inside the form, we have created 3 <input> elements for username, password and submit.
(For reference: https://www.w3schools.com/tags/att_input_required.asp)

In “**app.js**”, complete the callback function in route **app.post('/login_post', express.urlencoded({ extended: true }), (req, res) => {…})**, that handles the HTTP POST request sent by the client when the login form is submitted. In the callback function, check whether the username and password carried in the request body are in array *user_name* and array *user_password*, respectively. If the username and password match the corresponding records in the arrays, store the username into the sessoin variable “**loginName**” and send “**courses.html**” back to the client; otherwise, send “**fail.html**” to the client.
(Refer to Example 7 of 8_NODEJS_I_COMP3322B_s2022.pdf)

After this step, when you have successfully logged in, you will see a page like:



If your login fails, you will see a login failure page as follows:



## Part 3. Display courses and implement course enrollment.

Step 5. Implement the navigation bar.
In “**couses.html**”, we can see that the function **init()** is called when the page is loaded.

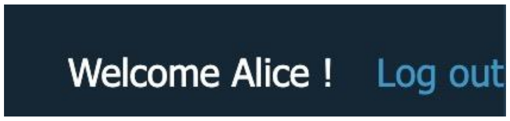
**5.1.** Open “**index.js**”. In the **init()** function, use **XMLHttpRequest** to send a **GET** request for “**courses/get_user**”. The server will process the request and respond with the user name in a string format (to be implement in **app.js** later). Upon successfully receiving server response (*readystate==4* and *status==200*), get the html element with id “**nav_guide**”, and set its

**innerHTML** value to "Welcome *username*!" (*username* should be replaced by the actual user name recevied from the server). Then create a <a> element and set its "href" attribute to "/logout" and innerHTML to "Log out". Append this hyperlink as a child to the **parent node** of id "**nav_guide**".

**5.2.** In "**app.js**", complete the callback function of route **app.get('/courses/get_user', (req, res) => {…})**, which sends the session variable **loginName**'s value to the client.

**5.3.** In "**app.js**", complete the callback function of route **app.get('/logout', (req, res) => {…})**, which resets the loginName session variable to *null* and redirects the client to '**/**'.

After this step, the navigation bar's contents can be displayed, as follows:



## Step 6. Show course information in a table.

**6.1.** In the client-side script "**index.js**", implement the **showAllCourses()** function to request from the server information of all coureses and display them in a course table, as follows: use **XMLHttpRequest** to send a HTTP **GET** request for "**courses/get_courses**"; upon receiving the server response, get the table element in "**courses.html**" with id "**course_table**" and set its **innerHTML** to the received *responseText* from the server (the server will render the course information in HTML representations and send it back to the client).

**6.2.** In the server-side script "**app.js**", complete the callback function of route **app.get('/courses/get_courses', (req, res) => {…})**, by creating the HTML contents for displaying the course information in a table in the "response" string and sending "response" back to the client. We have provided information of 5 courses in 4 arrays: *course_code*, *course_name*, *course_year*, *course_enrollment* (entries at the same index in the 4 arrays provide information of the same course). In the callback function, we have provided the code to create the header row with 4 **<th>** elements. You should create a row for each course and use 4 **<td>** elements to display the course's information (as shown in Fig. 2). Make the **<td>** elements containing course codes **draggable** by adding additional attributes to each **<td>** element: **id** (set the course code as the id value), **draggable** ("true"), and **ondragstart** ("**drag(event)**"). The **drag(event)** function has been given in "**index.js**".
(For reference of drag-and-drop, see page 24 of *7_HTML_withScript_COMP3322B_s2022.pdf*).

After finishing this step, you are able to display a course table with 5 courses, and drag the course code (we will implement drop later), as shown in Figure. 2.

Step 7. Course enrollment.

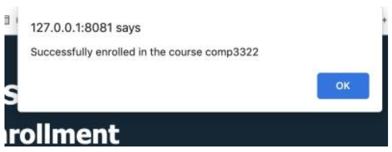
**7.1.** In "**courses.html**", we can find a **button** element "Enroll by Course Code" with an **onclick** event handled by **enrollCourse()** (to be implemented in "**index.js**"). Right above the button, there is a **<input>** element with **type="text"**. To allow dropping the dragged course code into this input box, add two attributes **ondragover="allowDrop(event)"** and **ondrop="drop(event)"** in the <input> tag. The two event handler functions are implemented in "**index.js**".

**7.2.** In "**index.js**", complete the **drop()** function by setting the value of the text input element to the **id** of the dragged element (the **id** has been set into the drag data store using the dataTransfer interface in the **drag()** function).

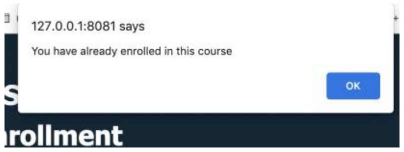
**7.3.** In "**index.js**", complete the function **enrollCourse()**: obtain the course code from the value of the "enroll_course" input element in "**courses.html**", and send a HTTP **POST** request for **"courses/enroll"**. Set "**application/x-www-form-urlencoded**" as a request header and put the course code in the request body in the format of "code=xx" (replace xx by the course code). Upon successfully receiving server response (a message string in *responseText*), **alert** this message and call **showAllCourses()** to refresh the course table display.

**7.4.** In "**app.js**", complete the callback function of route **app.post('/courses/enroll', express.urlencoded({ extended: true}), (req, res) => {…})**. First get the course code from the request body and find the index of this course in the *course_code* array. Then send a response string according to different cases: (1) if the course exists and has not been enrolled in before, send "*Successfully enrolled in the course xxx*" (xxx should be replaced by the course code) and set this course's enrollment status in the *course_enrollment* array to "**Yes**"; (2) if the course exists but has been enrolled in before, send "*You have already enrolled in this course"*; (3) otherwise, send "*The course code doesn't exist*".

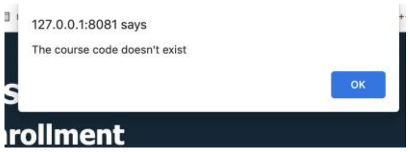
After this step, you can try dragging course code into the course code input box and clicking "Enroll by Course Code". You are able to see alert messages in your browser in different cases, and the "Enrolled" status in the course table will be updated accordingly.


127.0.0.1:8081 says

Successfully enrolled in the course comp3322


127.0.0.1:8081 says

You have already enrolled in this course


127.0.0.1:8081 says

The course code doesn't exist

Congratulations! Now you have finished Lab 4. You should test the pages and the final results should look similar to the screenshots at the beginning of this document.

## Submission:

Please finish this lab exercise before 23:59 Wednesday March 9, 2022. Please compress the entire app folder (i.e., the folder in which you create the express app) into a .zip file and submit it on Moodle.

(1) Login Moodle.
(2) Find “Labs” section and click “Lab 4”.
(3) Click “Add submission”, browse your .zip file and save it. Done.
(4) You will receive an automatic confirmation email, if the submission was successful.
(5) You can “Edit submission” to your already submitted file, but ONLY before the deadline.