Lab 4 : CS 387 Spring 2022 - MyASC@IITB

Course Registration Web Application
using Node.js, Postgres and React/Angular

Team Size : 2
Deadline : 2nd February, 2023, 11.59 PM

Overview
In this assignment you will create a backend server using node.js.
You will also create a web frontend, for which you can use the React framework, or the Angular framework.

Section A - Learning Basics of NodeJs, and React/Angular

Node.js Setup:

For Node.js development it is recommended to use Virtual Studio Code or Intellij Idea Ultimate. You don't have to use these, you can do it just as well with a text editor such as VIM or Emacs but these IDEs make life simpler.

Installation Instructions:

> sudo apt update

> sudo apt install nodejs

> nodejs -v (Confirm that the installation was successful)

> sudo apt install npm

> mkdir IITASC && cd IITASC // this is for the specific application you will build for this assignment.

> npm init

> npm -v // (Verify the installed version)

If successful, you will see the version number of npm installed

Read up on each of the below before starting on your assignment

1. **Introduction to node.js:** For an introduction to Node.js we recommend using w3schools.com/nodejs.
    a. But you may also use tutorialspoint.com/nodejs, with direct links to npm here, Hello World: here, and Callbacks: here
    b. A more detailed description may be found on the nodejs.de site
2. **Connecting to Postgres:** Node-postgres (pg) module to interact with postgres: from node-postgres.com
3. **Express framework:** For managing web pages, sessions etc, use the Express web application framework.
    a. Read the express tutorial from tutorialspoint
    b. Read up on Node.js Sessions using Express: Tutorial from section.io
4. **Frameworks for creating front ends**. There are many front end frameworks, here are a few of the popular ones. You may use either React or Angular, but we recommend React
    a. **React** with node.js: Read the React tutorial from reactjs.org/; if you want you can also read the react tutorial from w3schools or the tutorial from logrocket.
    b. **Angular** with Nodejs: tutorial from medium.com
    c. Embedded Java Script (EJS): tutorial from geeksforgeeks

Section B: Assignment

Inlab: Setup node.js, and React/Angular and try out the sample code from tutorials. There is no submission for inlab
Outlab: Assignment to be done as below in groups of 2

- We will use a slightly extended University Schema and data for this assignment.
    - We have provided you with a univ.ddl file on Moodle. This file creates additional relations and adds new data. This table consists of the registration dates corresponding to each semester and year. Use the current date to determine the running semester.
    - Add the following lines to your university schema ddl file to create a `web_user` table:

```
create table web_user
    (username          varchar(32),
     hashed_password     varchar(80),
     student_id       varchar(5),
     instructor_id          varchar(5),
```

```
    primary key (username),
    foreign key (student_id) references student (ID)
        on delete cascade,
    foreign key (instructor_id) references instructor (ID)
        on delete cascade
    );
```

    - You can use the small university data set for initial testing. If you want, you can use the large university dataset which is available here: https://www.db-book.com/university-lab-dir/sample_tables-dir/largeRelations/largeRelationsInsertFile.sql
- You must implement endpoints/interfaces, using node.js for the backend and React or Angular for the front-end as detailed below
- You must follow the submission instructions at the end of this document

Endpoints at the backend

You will need to create several endpoints as below. **Each endpoint will serve data in json form, to be rendered by the frontend using React of Angular.**

1. One backend endpoint per front end interface (details below)
2. Support endpoints for various services, including the following.
    a. List of all departments
    b. List of all courses in a department
    c. List of all courses
    d. Information about a student, including the courses taken by the student in different semesters (**Authorization**: accessible only to that student, and to faculty)
    e. Information about an instructor, including the courses taught by the instructor in the current and previous semesters (refer to the front-end interface for clarity)
    f. Information about a course, including prerequisites

Make sure to serve information using the endpoint only to logged in users, and where noted above, to authorized users.

Front End Interfaces

- **Note**: There are a few bonus points for aesthetics of the front-end interface so please make them look presentable

Interface 1 - Login Page
URL: /login/
For this section you may need to learn more about how passwords are securely stored in databases using salted hashes. here

1. Create a login page with the following credentials: student id and password as input, and authenticate the student credentials.
2. For valid student credentials, route to the student's homepage (/home/)
3. Handle the invalid credentials appropriately.
4. Use sessions here and in all other interfaces to ensure security

Note: The provided schema contains a relation that has the salt and hashed password. You have to use the sha256 hash function for hashing. You may use the postgres functionality given here. You need to append the salt to the password before hashing. We've provided the passwords for a fraction of the *students* in the univdb, which you may use for testing purposes. We'll evaluate the authentication features using the remaining half, so refrain from hardcoding.

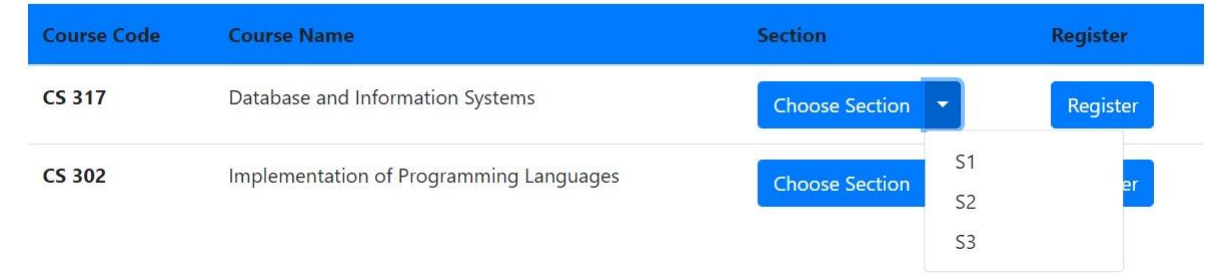
Interface 2 - Student Homepage
URL: /home/
Display the following information about the (logged in) student:

1. ID
2. Name
3. Department
4. Total Credits
5. List of courses registered in the previous semesters sorted in decreasing chronological order. Note that you should display a separate table for each semester (similar to external asc).
6. A separate list of courses registered this semester with an option to drop any particular course
7. (Optional) Any other relevant information

Interface 3 - Registration of Courses for logged in student
URL: /home/registration

Note: Unlike on ASC, we expect you to implement a functionality for registration of only one course at a time.

You need to implement a search box similar to asc with a search button. On typing a string and clicking search, you need to display all courses running in the current semester with a similar course id in the following way:



"Similar" here means that the search query needs to be a substring of the course id or course name. For each course display the available sections in a drop down. If you are using React you may find this component useful:
https://www.npmjs.com/package/react-search-autocomplete

On clicking the register button, you should check if the prerequisites for the course are satisfied. You also need to make sure that there are no slot clashes. On successful registration, the search box should be cleared and the relation for courses registered in this semester should be updated by adding the last registered course.

Note: On successful registration/dropping of a course, the list of courses registered this semester should be updated accordingly.

Interface 4 - Course information page (open to all logged in users)
URL: /course/:course_id/
For each course (being offered in the current semester), there should be a web page which displays the following information about the course:

1. Course id, course name, credits, venue
2. Course prerequisites (id and name)
    a. On clicking any of the pre-requisite courses, the student should be routed to the corresponding course's webpage.
3. Course Instructor(s)
    a. On clicking any of the course instructors, the student should be routed to the corresponding instructor's webpage.

The information about the courses running in the current semester should be organised as follows:

- There should be a page listing the departments which offer courses running in the semester.
  URL: /course/running/
- On clicking the link for a department, the list of running courses for that department must show up.
  URL: /course/running/:dept_id/
- On clicking the link for a running course in the department, the course information page for that course must be displayed, as described above.

Interface 5 - Instructor page (open to all logged in users)
URL: /instructor/:instructor_id/
For each instructor, there should be a web page which displays the following information about the instructor:

1. Name of the instructor
2. Instructor's department
3. List of courses (course id and name) being offered by the instructor in the current semester (sorted lexicographically by course id)
    a. Clicking on any of these courses should route the student to the course's information page
4. List of courses (course id and name) taught by the instructor (in the previous semesters) in decreasing chronological order

Submission instructions

Each team should submit one tarball named <rollnum1>-<rollnum2>-lab4.tar.gz. Consisting of the following:

1. Submit the frontend in directory named frontend
    a. For **ReactJs** : Need to submit the src folder of your reactjs project. It should contain the following files/folder.
        i. Assets : all static contents such as images, json files.
        ii. Components
        iii. App.js, App.css
        iv. Index.js, index.css
        v. Package.json, Package-lock.json
    b. For **Angular** : Need to submit the src folder of your angular project. It should contain the following files/folder :
        i. Services
        ii. Routes

iii. Utils : All static content and .ts/.js files should be provided.
iv. app.component.html, app.component.css, app.component.ts
v. App.module.ts, app-routing.module.ts
vi. main.ts

Note : You can group components and services into subfolders for each different component of your application. You can use global routing files as well as components specific routing files.

2. Submit the src folder of your **nodejs** backend **in a directory named backend**. It should contain the following files/folder.
    a. Controllers
    b. Models
    c. Services
    d. package.json
    e. config.txt

    Note : All backend config arguments like postgres username, password, host, port, database name, etc. should be present in the `config.txt` file and read from there, so that grading TAs can easily modify those values.

3. Readme.md file

    It will contain all the references you are using for your template and other static resources.