CS628 Full-Stack Development – Web App

**HOS08A: MERN**

Created by Veerendra Jagatha on Aug 17, 2023

Revised by Sam Chung on Aug 26, 2023

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* Version numbers may not match the most current version at the time of writing. If given the option to choose between the stable release (long-term support) or the most recent, please select the **stable release** rather than the beta-testing version.
* There might be subtle discrepancies along with the steps. Please **use your best judgment** while going through this cookbook-style tutorial to complete each step.
* If you are not familiar with a terminal, command line, and bash scripts, check out this video: <https://youtu.be/Dp7uw9c6QH8>
* All the steps and concepts in this tutorial are from references, so if you encounter problems, please **try to read and compare the references to solve the problem**. If you still can't solve the problem, please contact your course TA.
* **Avoid copy-pasting code from the book or the GitHub repository**. Instead, type out the code yourself. Resort to copy-pasting only when you are stuck and find things not working as expected.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

#### **Readings and Examples:**

* Visit [CS 628 Repository for Examples](https://github.com/samchung0117/cs628-examples).
  + Select the related module.
  + Visit the README.md file.
  + Find examples for your practices.

**Learning Outcomes**

* Section 1: Accessing GitHub Codespaces.
* Section 2: Setting up the project.
* Section 3: Setting up the React Router
* Section 4: Creating the Components
* Section 5: Connecting the Front End to the Back End
* Section 6: Pushing your work to GitHub.

**Section 1: Accessing GitHub Codespaces**

Before you start the current code space, you need to execute your server. There are two ways. First, while you are working on HOS09, you can execute your backend under the HOS08 directory.

Otherwise, second, copy the backend code that we created in the “HOS08” to current HOS directory. You can use the steps mentioned in **CopyRepo.md** in the hos-examples to move your backend code to current repository.

Refer the steps from the [TA Center](https://cityuseattle.github.io/docs/git/github_codepsace/) to get started with this week’s module GitHub Codespace.

**Section 2: Setting up the project**

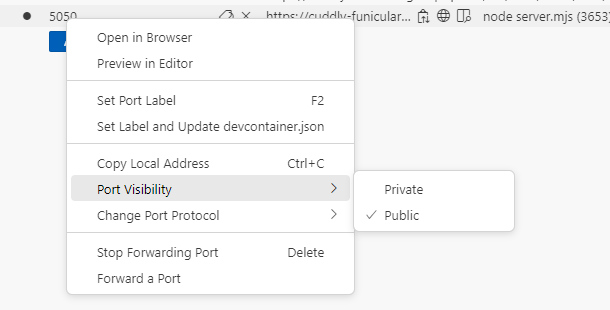
Before we begin creating the front end, make sure we run the backend server and test that our API is working.

Navigate to the backend directory that you practiced in HOS08. And start the server using the below commands.

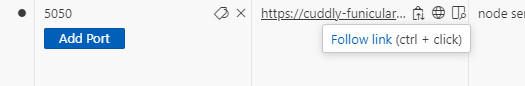
**>> cd backend/**

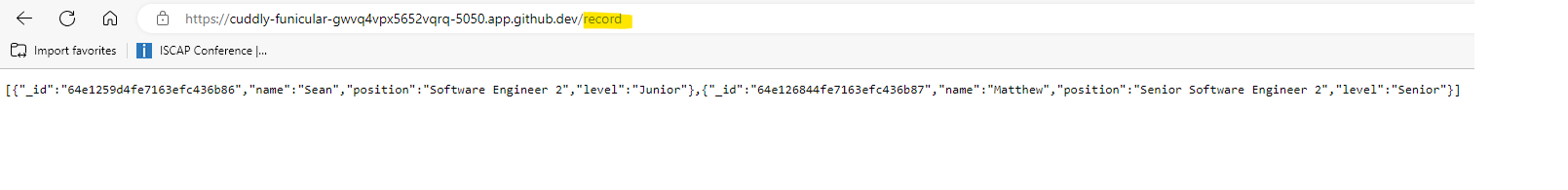
**>> node server.mjs**

Now you must make the port running your backend server public just like we did it in previous module.

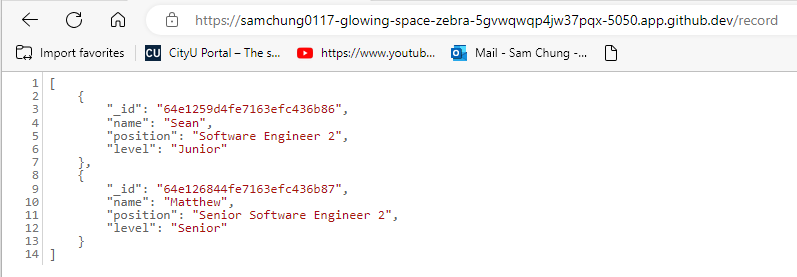
****

Open the local address in the new tab and make sure the API is working,

****

****

**Another example,**



Now open a **new** terminal and let us start creating the front-end client to consume this API.

Create a directory with name “mern” under “HOS09” and inside it, let’s create a react app with name “hos09-client” using the following commands.

**>>mkdir mern**

**>>cd mern**

**>>npx create-react-app hos09-client**

Next, we'll navigate to the 'hos09-client' directory in the terminal and proceed to install two extra dependencies essential for our project. Execute the following commands to accomplish this.

**>>cd hos09-client**

**>>npm install react-router-dom**

**>>npm install bootstrap**

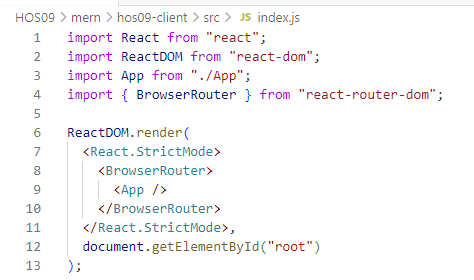
Bootstrap enables rapid deployment of templates and components for your new web app, sparing you from starting from scratch, while 'react-router-dom' installs React router components tailored for web applications.

****

**Section 3: Setting up the React Router**

We will use the “BrowserRouter” which we learned in previous modules to ensure our UI stays synchronized with the URL, enabling smooth transitions when moving between components.

Replace the code in the “index.js” with the following,

****

**Section 4: Creating the Components**

Now we need to create components for our application. Use the following commands to create four components. Assume you are under the “mern/hos09-client” directory.

1. create.js
2. edit.js
3. recordList.js
4. navbar.js

**>> mkdir src/components**

**>> cd src/components/**

**>> touch create.js edit.js navbar.js recordList.js**

Update these 4 components with the code provided in the “cs628-examples”

**create.js:** The provided code will function as a creation component for our records, allowing users to generate new entries. Through this component, a create command will be sent to our server.

**edit.js**: edit component will act as an editing component for our records, adopting a layout like the create component, and ultimately sending an update command to our server.

**recordList.js**: The provided code will function as a viewing component for our records, retrieving all database entries through a GET method.

**navbar.js**: Within the 'navbar.js' component, we'll have a navigation bar that link us to the necessary components.

After creating the components, replace the code in your App.js with the following,



**Section 5: Connecting the Front End to the Back End**

We've successfully created components and established a connection between our React app and the backend using the 'fetch' function, which simplifies handling HTTP requests. This approach is applied in 'create.js', 'edit.js', and 'recordList.js' to manage their respective requests.

In 'create.js', we enhance the 'onSubmit(e)' block with code that triggers a POST request to the create URL, adding new records to the database. Similarly, in 'edit.js', we insert code within the 'onSubmit(e)' block and beneath the constructor to make updates. Lastly, in 'recordList.js', which fetches records from the database, we utilize 'fetch's' GET method to retrieve these records.

If you observe all the fetch requests will be missing the API URL in your code under all components. You will see something like this in create.js for example,



Update this with your backend API URL to have something like this,



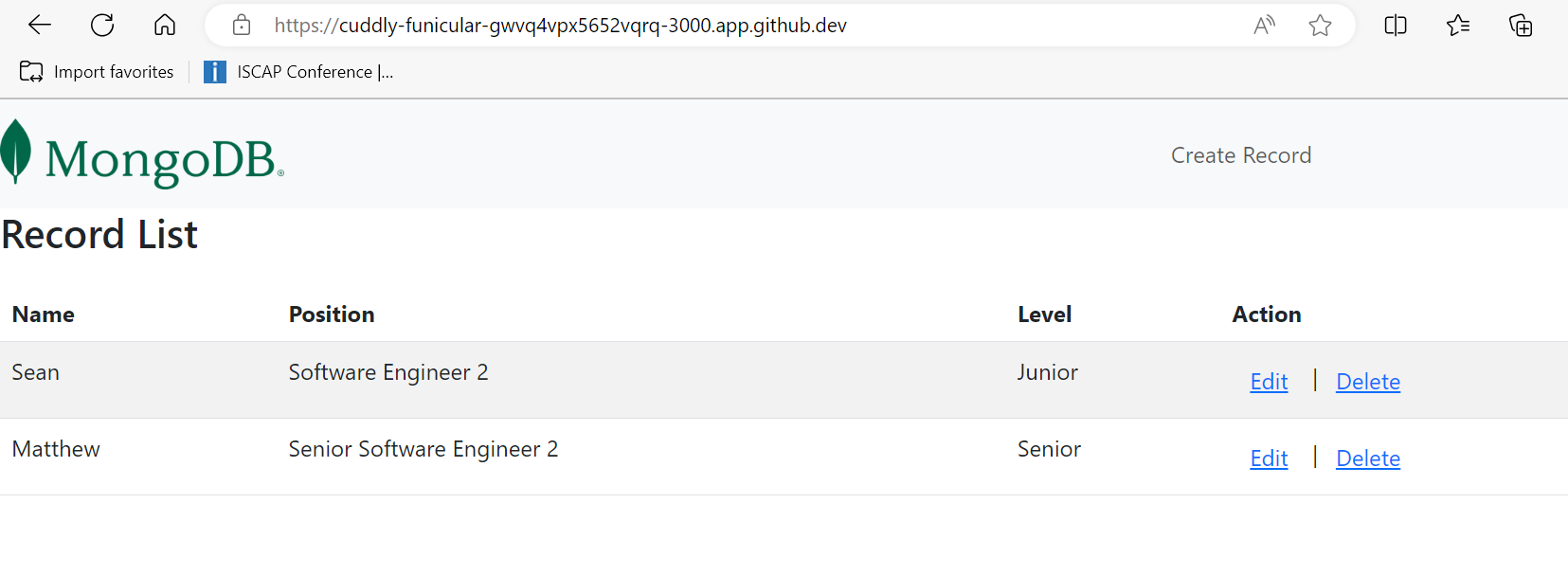
You must do this for all fetch request in all other components. Once all the components are updated with the API URL, we need to navigate to “hos09-client” directory in the terminal and start the application using the command below,

**>>npm start**

**Note:** Make sure your backend API server is running in another terminal. Otherwise in the backend directory we can start it using the below command.

**>> node server.mjs**

After refreshing the development browser window, you'll observe the frontend client application has successfully linked with the backend server, retrieving records from the MongoDB Atlas Database.



We are now able to generate new records and perform updates or deletions through the frontend, initiating the corresponding backend API calls.

**A screenshot of a computer

Description automatically generated**

**Section 6: Pushing your work to GitHub**

* 1. Go to Source Control on your GitHub codespace and observe the pending changes.

Graphical user interface, text, application

Description automatically generated

* 1. Type the Message for your changes in the Message box on the top. For example,” **Submission for Module09 – Your Name**”
  2. Click on the dropdown beside the commit button and select **Commit & Push** to update the changes to your repository main branch.
  3. Select **Yes** when prompted.

Graphical user interface, application

Description automatically generated