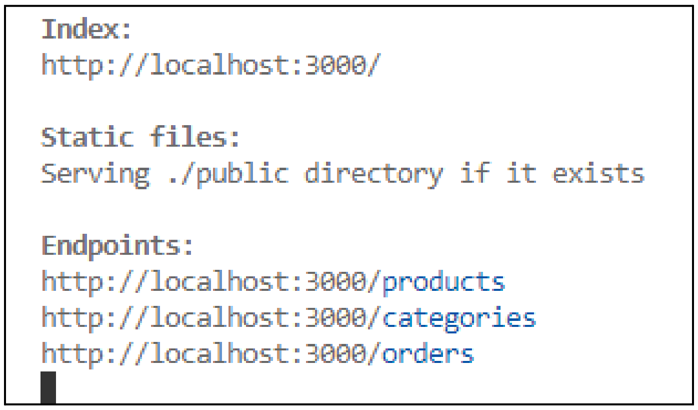
Advanced JS - Guided Project

Introduction

In this phase of our E-Commerce application development, we'll focus on enabling seamless interaction between the frontend and backend. You'll learn how to fetch and post data using a local JSON Server, which will simulate real-world backend operations. Additionally, we'll implement user authentication using **Local Storage** and **Session Storage**.

Problem Statement

Execute the index.html file and go through the entire application. Follow the instructions in the Readme file from the source code to launch a JSON server and complete the tasks below in the same order.



```
Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/products
http://localhost:3000/categories
http://localhost:3000/orders
```

1. Table Data

a. **Go through the structure of the data.json** file before we proceed further.

b. Use the URLs from json-server to **fetch data** and display it in the corresponding **tables on your web pages**.

c. Create a new JavaScript file inside the `js` folder and link it to the appropriate HTML file.

d. Use the **Fetch API** to retrieve data from the server and dynamically add it to the table's `<tbody>` section.

e. The Action Columns on the tables will have dummy Buttons like Edit, View, and Update Status as created in the previous versions of the project. Refer to the solution file if needed and replicate the same.

f. Repeat this process for **all tables** in the application to ensure each one is populated with server data.

2. Form Data

a. Go through the application and **identify the forms.**

b. When a form is submitted, **collect and validate** the input data to ensure it's complete and correct.

c. Use the **Fetch API** to **upload the data to the JSON Server** via a POST request.

d. **After successful submission, reset the form** and **redirect the user** to the corresponding table's view page.

   — ***Example***: *Submitting the **Add Product** form should navigate to the **View Products** page.*

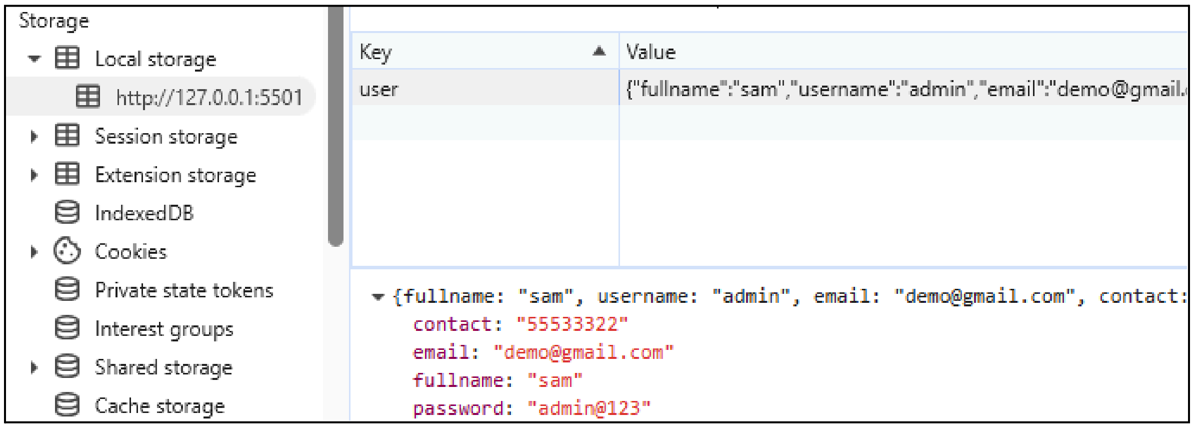
e. Ensure that the **newly added data appears** in the table immediately.

f. Repeat this process for **all form web pages** in the application to ensure the server is populated with form data.

3. Register

A user **must be registered** before accessing any page on the website.

a. Use the **Registration form** to collect user details and handle the **form submission** using JavaScript.
b. **Get input data** and **validate the password** upon submission.
c. If the password and confirm password are not the same, show an error message ("Password does not match. Re-enter the password"). If the passwords match, redirect to the sign-in page.
d. Store the **data** as an **object in Local Storage** for reference and authentication purposes as shown below.

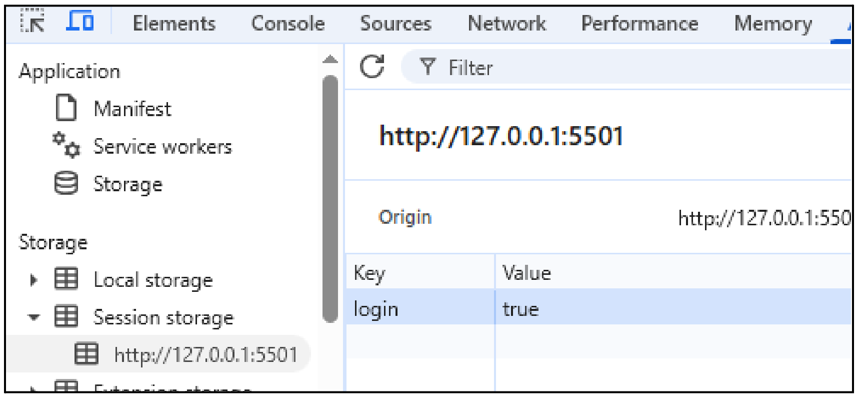


Key	Value
user	{"fullname":"sam","username":"admin","email":"demo@gmail.

```
▾ {fullname: "sam", username: "admin", email: "demo@gmail.com", contact:
    contact: "55533322"
    email: "demo@gmail.com"
    fullname: "sam"
    password: "admin@123"
```

4. Sign In

a. Use the **Sign In form** to **capture the username and password** upon submission.
b. **Retrieve the stored user data** from **Local Storage** and **verify the credentials** against it.
c. If the credentials are valid, **set the login status to true** and **store it in Session Storage**. Also, **redirect the user to the home page** of the application.
d. If the credentials are not valid, **show a message on the page to register before Sign in.**
e. Ensure that access to the home page is **restricted** unless the user has successfully **signed in** and the login status is verified.



http://127.0.0.1:5501

Origin http://127.0.0.1:550

Key	Value
login	true

5. Validate Sign In

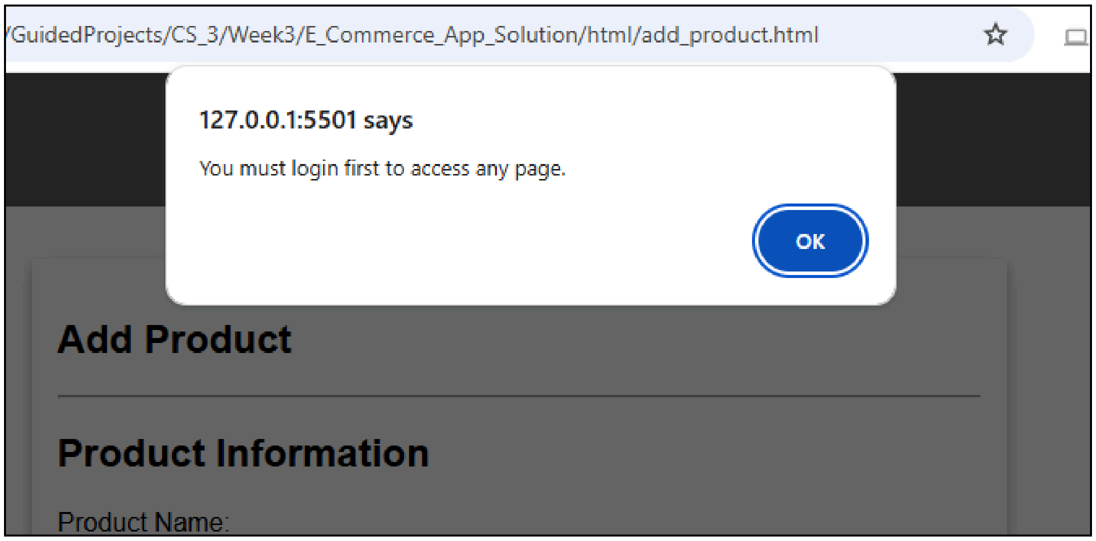
a. On loading **any web page**, first **check the log-in status** from **Session Storage**.

b. If the login status is **false or missing**, display an **alert** prompting the user to sign in, and **redirect** them to the **Sign In** page.

c. If the login status is true in session storage, allow access to the page and **display the username in the header**.

d. Apply this validation logic to **all HTML pages** in the application **except** `sign-in.html` and `register.html`.

e. Add an event listener to the **Sign Out** button on every page header, when clicked, **removes the login status from Session Storage** and **redirects the user to the Sign In page**.


/GuidedProjects/CS_3/Week3/E_Commerce_App_Solution/html/add_product.html

127.0.0.1:5501 says

You must login first to access any page.

!!! Happy Coding !!!