**Step-by-Step: Create a SMART on FHIR Web App with Express**

**1. Set Up the Project Directory**

First, create your project folder with the following structure:

My folder is: D:\dDev\JavaScript\fhir-client-js\fhir-portal-js

smart-on-fhir-app/

│

├── public/ # Contains static HTML, CSS, JS files

│ ├── css/

│ │ └── styles.css

│ ├── js/

│ │ └── app.js # Main JavaScript file for app logic

│ └── index.html # Main HTML file

│

├── server.js # Express server setup

├── package.json # Project dependencies

└── callback.html # OAuth callback page

**2. Install Node.js and Express**

If you don’t have Node.js installed, follow the steps mentioned earlier to install it on your machine.

After installing Node.js, navigate to your project folder in your terminal and initialize the project:

npm init -y

Then, install the express package, which will be used to create the server for your web app:

npm install express //4.21.1

Also, install the SMART on FHIR client:

npm install fhirclient // 2.5.4

Your package.json file should now include both express and fhirclient as dependencies.

smart-on-fhir-app@1.0.0 D:\dDev\JavaScript\fhir-client-js\fhir-portal-js

+-- express@4.21.1

`-- fhirclient@2.5.4

**3. Create the Express Server**

Create the server.js file in your project root directory. This file will set up the Express server to serve your web app and handle the OAuth flow.

// server.js

const express = require('express');

const path = require('path');

const app = express();

// Serve static files from the "public" directory

app.use(express.static(path.join(\_\_dirname, 'public')));

// Route for the OAuth callback page

app.get('/callback', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'callback.html'));

});

// Start the server

const port = process.env.PORT || 3000;

app.listen(port, () => {

console.log(`Server running on http://localhost:${port}`);

});

**4. Create the HTML and Frontend Files**

In the public/ directory, create an index.html file. This will be the main interface where the user can log in using SMART on FHIR and view patient information.

<!-- public/index.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>SMART on FHIR Web App</title>

<link rel="stylesheet" href="css/styles.css">

</head>

<body>

<h1>SMART on FHIR Web App</h1>

<button id="login-button">Login with SMART on FHIR</button>

<div id="patient-info"></div>

<script src="js/app.js"></script>

</body>

</html>

Next, in public/js/app.js, implement the authentication and FHIR API calls using the SMART on FHIR JavaScript client.

// public/js/app.js

document.getElementById("login-button").addEventListener("click", function() {

FHIR.oauth2.authorize({

client\_id: "YOUR\_CLIENT\_ID",

scope: "launch/patient patient/\*.\* openid fhirUser",

iss: "https://r4.smarthealthit.org", // FHIR server URL

redirect\_uri: "http://localhost:3000/callback"

});

});

// After authentication, retrieve the patient data

FHIR.oauth2.ready().then(function(client) {

client.patient.read().then(function(patient) {

console.log(patient);

displayPatientInfo(patient);

});

});

// Function to display patient information in the UI

function displayPatientInfo(patient) {

const patientInfoDiv = document.getElementById("patient-info");

patientInfoDiv.innerHTML = `

<h2>Patient Information</h2>

<p>Name: ${patient.name[0].given.join(" ")} ${patient.name[0].family}</p>

<p>Gender: ${patient.gender}</p>

<p>Birth Date: ${patient.birthDate}</p>

`;

}

**5. Create the OAuth Callback Page**

The callback.html page will handle the OAuth callback response and redirect the user back to the main app page.

<!-- callback.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>SMART on FHIR Callback</title>

</head>

<body>

<h1>Loading...</h1>

<script src="https://unpkg.com/fhirclient@2.5.7/lib/fhir-client.js"></script>

<script>

FHIR.oauth2.ready(function(client) {

// After authentication, redirect to the main app

window.location.href = "/";

});

</script>

</body>

</html>

**6. Create a Basic CSS File**

You can create a simple CSS file to style your app (public/css/styles.css):

/\* public/css/styles.css \*/

body {

font-family: Arial, sans-serif;

margin: 20px;

}

h1 {

color: #2c3e50;

}

#patient-info {

margin-top: 20px;

background-color: #ecf0f1;

padding: 15px;

border-radius: 5px;

}

**7. Run the Application**

After you’ve created the files, you can now run the app locally.

Start the Express server by running the following command in the terminal:

node server.js

This will start the server, and you can access the app by navigating to http://localhost:3000 in your web browser.

**8. Test the App**

1. Open your browser and go to http://localhost:3000.
2. Click the "Login with SMART on FHIR" button.
3. The app will redirect you to the FHIR server login page. After authenticating, it will return you to the app with the patient information displayed.

**9. Configure the Client ID and FHIR Server**

Ensure you replace "YOUR\_CLIENT\_ID" with the actual client ID you registered with the FHIR server. You can register your app on the SMART Health IT Sandbox or use a similar FHIR server.

**Next Steps**

* **Add more features**: You can extend the app by adding features to fetch other FHIR resources, such as encounters, medications, or conditions.
* **Improve UI**: Style the app using CSS frameworks like Bootstrap or Material Design.
* **Deploy**: Consider deploying the app to a service like Heroku or Vercel for easier sharing.

This detailed guide should help you set up and structure a SMART on FHIR app using Express and the SMART JavaScript client. Let me know if you need further assistance!