**Batch: A-3 Roll No.: 16010122104**

**Experiment / assignment / tutorial No. 6**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| Title: Implementation of Express.js |

**AIM:** Demonstrate the use of Express js functionalities.

**Problem Definition:**

Consider the basic concepts of Express.js, which are useful in the creation of an application.

Considering the following points, demonstrate the functionality of each with a simple script

**1) Scaffolding:**

          1. Demonstrate express scaffolding to fulfill the following requirements.

         Example: Consider Grocery Delivery Application and demonstrate the Scaffolding  
                Scaffold the application to create different routes such as.

* Sign up Page: (Root/ Homepage)

2) **Serving static files using Express.js:** With the help of Built in middleware, express. Static () to demonstrate the usage of serving static files in express.

          To demonstrate the above make a use of

·         Use of images where it should accept any type of image

·         Use of CSS and HTML files.

·         Make a Use json file of employee information, add file to the static folder, and show the response on the browser.

**Note:**

·         **Assume your own data whenever required to perform the operation.**

**Resources used:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO 4: Test the concepts and components of various front-end, back-end web app**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Shelly Powers Learning Node O’ Reilly 2 nd Edition, 2016.

**Pre Lab/ Prior Concepts:**

**Express.js**

Express.js is a minimal, flexible Node.js web application framework that provides a powerful set of features for web and mobile applications. Known for its simplicity and performance, Express streamlines the process of building robust APIs, single-page applications, and dynamic web applications. It allows developers to manage HTTP requests and responses easily, define various routes, and integrate middleware to handle tasks such as authentication and error handling. By using Express, developers can quickly set up server-side functionality and maintain scalable, structured applications.

**Scaffolding**

Scaffolding in web development refers to the automatic generation of a basic project structure, including the setup of folders, files, and boilerplate code. This helps developers to kickstart projects without manually creating standard setups. In the context of Express, tools like express-generator can create a predefined structure that includes essential folders like routes, views, and public for assets. Scaffolding speeds up development, ensures consistency, and provides a clean architecture to work with, allowing developers to focus more on building features than on repetitive setup tasks.

**Express Routing**

Express routing enables developers to define and handle various HTTP requests to different endpoints in a Node.js application. With routing, developers can specify how the server should respond to different URL paths and HTTP methods, such as GET, POST, PUT, and DELETE. By structuring routes efficiently, Express helps create organized, modularized endpoints for handling functionality like API responses, data retrieval, and user interactions. This routing approach is essential for creating scalable applications with clean, manageable code.

**Methodology:**

import React, { useState } from 'react'

import {assets} from '../assets/assets'

import axios from 'axios'

import { backendUrl } from '../App'

import { toast } from 'react-toastify'

const Add = ({token}) => {

  const [image1,setImage1] = useState(false)

  const [image2,setImage2] = useState(false)

  const [image3,setImage3] = useState(false)

  const [image4,setImage4] = useState(false)

   const [name, setName] = useState("");

   const [description, setDescription] = useState("");

   const [price, setPrice] = useState("");

   const [category, setCategory] = useState("Men");

   const [subCategory, setSubCategory] = useState("Topwear");

   const [bestseller, setBestseller] = useState(false);

   const [sizes, setSizes] = useState([]);

   const onSubmitHandler = async (e) => {

    e.preventDefault();

    try {

      const formData = new FormData()

      formData.append("name",name)

      formData.append("description",description)

      formData.append("price",price)

      formData.append("category",category)

      formData.append("subCategory",subCategory)

      formData.append("bestseller",bestseller)

      formData.append("sizes",JSON.stringify(sizes))

      image1 && formData.append("image1",image1)

      image2 && formData.append("image2",image2)

      image3 && formData.append("image3",image3)

      image4 && formData.append("image4",image4)

      const response = await axios.post(backendUrl + "/api/product/add",formData,{headers:{token}})

      if (response.data.success) {

        toast.success(response.data.message)

        setName('')

        setDescription('')

        setImage1(false)

        setImage2(false)

        setImage3(false)

        setImage4(false)

        setPrice('')

      } else {

        toast.error(response.data.message)

      }

    } catch (error) {

      console.log(error);

      toast.error(error.message)

    }

   }

  return (

    <form onSubmit={onSubmitHandler} className='flex flex-col w-full items-start gap-3'>

        <div>

          <p className='mb-2'>Upload Image</p>

          <div className='flex gap-2'>

            <label htmlFor="image1">

              <img className='w-20' src={!image1 ? assets.upload\_area : URL.createObjectURL(image1)} alt="" />

              <input onChange={(e)=>setImage1(e.target.files[0])} type="file" id="image1" hidden/>

            </label>

            <label htmlFor="image2">

              <img className='w-20' src={!image2 ? assets.upload\_area : URL.createObjectURL(image2)} alt="" />

              <input onChange={(e)=>setImage2(e.target.files[0])} type="file" id="image2" hidden/>

            </label>

            <label htmlFor="image3">

              <img className='w-20' src={!image3 ? assets.upload\_area : URL.createObjectURL(image3)} alt="" />

              <input onChange={(e)=>setImage3(e.target.files[0])} type="file" id="image3" hidden/>

            </label>

            <label htmlFor="image4">

              <img className='w-20' src={!image4 ? assets.upload\_area : URL.createObjectURL(image4)} alt="" />

              <input onChange={(e)=>setImage4(e.target.files[0])} type="file" id="image4" hidden/>

            </label>

          </div>

        </div>

        <div className='w-full'>

          <p className='mb-2'>Product name</p>

          <input onChange={(e)=>setName(e.target.value)} value={name} className='w-full max-w-[500px] px-3 py-2' type="text" placeholder='Type here' required/>

        </div>

        <div className='w-full'>

          <p className='mb-2'>Product description</p>

          <textarea onChange={(e)=>setDescription(e.target.value)} value={description} className='w-full max-w-[500px] px-3 py-2' type="text" placeholder='Write content here' required/>

        </div>

        <div className='flex flex-col sm:flex-row gap-2 w-full sm:gap-8'>

            <div>

              <p className='mb-2'>Collections</p>

              <select onChange={(e) => setCategory(e.target.value)} className='w-full px-3 py-2'>

                  <option value="Solid">Solid</option>

                  <option value="Simple">Simple</option>

                  <option value="Complex">Complex</option>

              </select>

            </div>

            <div>

              <p className='mb-2'>Tier</p>

              <select onChange={(e) => setSubCategory(e.target.value)} className='w-full px-3 py-2'>

                  <option value="One">One</option>

                  <option value="Two">Two</option>

                  <option value="Three">Three</option>

              </select>

            </div>

            <div>

              <p className='mb-2'>Product Price</p>

              <input onChange={(e) => setPrice(e.target.value)} value={price} className='w-full px-3 py-2 sm:w-[120px]' type="Number" placeholder='25' />

            </div>

        </div>

        <div>

          <p className='mb-2'>Product Sizes</p>

          <div className='flex gap-3'>

            <div onClick={()=>setSizes(prev => prev.includes("S") ? prev.filter( item => item !== "S") : [...prev,"S"])}>

              <p className={`${sizes.includes("S") ? "bg-pink-100" : "bg-slate-200" } px-3 py-1 cursor-pointer`}>S</p>

            </div>

            <div onClick={()=>setSizes(prev => prev.includes("M") ? prev.filter( item => item !== "M") : [...prev,"M"])}>

              <p className={`${sizes.includes("M") ? "bg-pink-100" : "bg-slate-200" } px-3 py-1 cursor-pointer`}>M</p>

            </div>

            <div onClick={()=>setSizes(prev => prev.includes("L") ? prev.filter( item => item !== "L") : [...prev,"L"])}>

              <p className={`${sizes.includes("L") ? "bg-pink-100" : "bg-slate-200" } px-3 py-1 cursor-pointer`}>L</p>

            </div>

            <div onClick={()=>setSizes(prev => prev.includes("XL") ? prev.filter( item => item !== "XL") : [...prev,"XL"])}>

              <p className={`${sizes.includes("XL") ? "bg-pink-100" : "bg-slate-200" } px-3 py-1 cursor-pointer`}>XL</p>

            </div>

          </div>

        </div>

        <div className='flex gap-2 mt-2'>

          <input onChange={() => setBestseller(prev => !prev)} checked={bestseller} type="checkbox" id='bestseller' />

          <label className='cursor-pointer' htmlFor="bestseller">Add to bestseller</label>

        </div>

        <button type="submit" className='w-28 py-3 mt-4 bg-black text-white'>ADD</button>

    </form>

  )

}

export default Add

**Implementation Details:**

 **Import Dependencies**

* Import React and useState for state management.
* Import assets from your project, Axios for API requests, and toast for notifications.

javascript

Copy code

import React, { useState } from 'react';

import { assets } from '../assets/assets';

import axios from 'axios';

import { backendUrl } from '../App';

import { toast } from 'react-toastify';

 **Define Component and State Variables**

* Define Add as a functional component that accepts token as a prop.
* Initialize state variables for product details (e.g., name, description, price, etc.), four image files (image1 to image4), and an array for selected sizes.

javascript

Copy code

const Add = ({ token }) => {

const [image1, setImage1] = useState(false);

const [image2, setImage2] = useState(false);

const [image3, setImage3] = useState(false);

const [image4, setImage4] = useState(false);

const [name, setName] = useState("");

const [description, setDescription] = useState("");

const [price, setPrice] = useState("");

const [category, setCategory] = useState("Men");

const [subCategory, setSubCategory] = useState("Topwear");

const [bestseller, setBestseller] = useState(false);

const [sizes, setSizes] = useState([]);

 **Define the onSubmitHandler Function**

* The onSubmitHandler is an asynchronous function that prevents form submission, creates a FormData object, and appends product details and images.
* It then sends the form data to the backend using an Axios POST request with the token as a header for authentication.
* If successful, it displays a success toast and clears the form fields; otherwise, it displays an error message.

javascript

Copy code

const onSubmitHandler = async (e) => {

e.preventDefault();

try {

const formData = new FormData();

formData.append("name", name);

formData.append("description", description);

formData.append("price", price);

formData.append("category", category);

formData.append("subCategory", subCategory);

formData.append("bestseller", bestseller);

formData.append("sizes", JSON.stringify(sizes));

image1 && formData.append("image1", image1);

image2 && formData.append("image2", image2);

image3 && formData.append("image3", image3);

image4 && formData.append("image4", image4);

const response = await axios.post(`${backendUrl}/api/product/add`, formData, { headers: { token } });

if (response.data.success) {

toast.success(response.data.message);

setName('');

setDescription('');

setImage1(false);

setImage2(false);

setImage3(false);

setImage4(false);

setPrice('');

} else {

toast.error(response.data.message);

}

} catch (error) {

console.log(error);

toast.error(error.message);

}

};

 **Render the Form**

* The form includes fields for uploading images, entering product details (name, description, price, category, etc.), selecting sizes, and a checkbox to mark the product as a bestseller.
* Each input field updates its corresponding state variable on change.

javascript

Copy code

return (

<form onSubmit={onSubmitHandler} className='flex flex-col w-full items-start gap-3'>

{/\* Image Upload \*/}

<div>

<p>Upload Image</p>

<div className='flex gap-2'>

{/\* Image Inputs \*/}

<label htmlFor="image1">

<img src={!image1 ? assets.upload\_area : URL.createObjectURL(image1)} alt="" />

<input onChange={(e) => setImage1(e.target.files[0])} type="file" id="image1" hidden />

</label>

{/\* Repeat for image2, image3, and image4 \*/}

</div>

</div>

{/\* Product Name \*/}

<input onChange={(e) => setName(e.target.value)} value={name} placeholder='Type here' required />

{/\* Product Description \*/}

<textarea onChange={(e) => setDescription(e.target.value)} value={description} placeholder='Write content here' required />

{/\* Category and Subcategory \*/}

<select onChange={(e) => setCategory(e.target.value)}>

<option value="Solid">Solid</option>

{/\* Add other options \*/}

</select>

<select onChange={(e) => setSubCategory(e.target.value)}>

<option value="One">One</option>

{/\* Add other options \*/}

</select>

{/\* Sizes \*/}

<div onClick={() => setSizes(prev => prev.includes("S") ? prev.filter(size => size !== "S") : [...prev, "S"])}>

<p className={`${sizes.includes("S") ? "bg-pink-100" : "bg-slate-200"}`}>S</p>

</div>

{/\* Repeat for other sizes \*/}

{/\* Bestseller Checkbox \*/}

<input onChange={() => setBestseller(!bestseller)} checked={bestseller} type="checkbox" id='bestseller' />

<label htmlFor="bestseller">Add to bestseller</label>

{/\* Submit Button \*/}

<button type="submit" className='w-28 py-3 mt-4 bg-black text-white'>ADD</button>

</form>

);

 **Export the Component**

* Export Add as the default export for use in other parts of your application.

javascript

Copy code

export default Add;

**Steps for execution:**

 **Setup Dependencies**  
Ensure axios and react-toastify are installed in your project. Use npm install axios react-toastify.

 **Backend API Endpoint**  
Verify that the backend API (backendUrl + "/api/product/add") is set up to accept multipart/form-data and process product information and images.

 **Rendering the Component**  
Import Add in the desired file (e.g., App.js) and pass the token as a prop.

javascript

Copy code

import Add from './components/Add';

function App() {

const token = "YOUR\_AUTH\_TOKEN";

return <Add token={token} />;

}

 **Run the Application**  
Start your application with npm start. The Add form should now render, allowing product information and images to be uploaded.

 **Testing**  
Submit the form with sample data to test functionality and verify that product information is sent correctly to the backend

**Conclusion: We implemented express.js functionalities in our project.**