DIGA4015A:
Interactive Media 4A | React Exercises Solutions

Note:

This is **one** method of completing the following exercises, not the only method. How you choose to answer will depend on how you interpret the question, but the logic should remain roughly the same.

Example 1:

Build an e-commerce application where each product has a unique ID, and the product details are displayed on a dynamic route.

Requirements:

1. Create a route `/products/:id` to display details of a specific product.
2. Use `useParams` to extract the product ID from the URL.
3. Fetch product details from an API or mock data based on the ID.

Hints:

1. Use `useParams` to access the product ID.
2. Use `useEffect` to fetch product details based on the ID.
3. Display the product details on the page.
4. You can use the mock product list, and product details, below.

Product List:
const products = [{ id: 1, name: 'Product 1', price: 100 }, { id: 2, name: 'Product 2', price: 200 }, { id: 3, name: 'Product 3', price: 300 },];

Product Details:
const products = [{ id: 1, name: 'Product 1', price: 100, description: 'This is product 1' }, { id: 2, name: 'Product 2', price: 200, description: 'This is product 2' }, { id: 3, name: 'Product 3', price: 300, description: 'This is product 3' },];

```
// src/App.js
import React from 'react';
import { BrowserRouter, Routes, Route, Link } from 'react-router-dom';
import ProductList from './components/ProductList';
```

```
import ProductDetails from './components/ProductDetails';



function App() {
  return (
    <BrowserRouter>
      <nav>
        <Link to="/">Home</Link>
      </nav>
      <Routes>
        <Route path="/" element={<ProductList />} />
        <Route path="/products/:id" element={<ProductDetails />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;



// src/components/ProductList.js
import React from 'react';
import { Link } from 'react-router-dom';

const products = [
  { id: 1, name: 'Product 1', price: 100 },
  { id: 2, name: 'Product 2', price: 200 },
  { id: 3, name: 'Product 3', price: 300 },
];

function ProductList() {
  return (
    <div>
      <h1>Product List</h1>
      <ul>
        {products.map((product) => (
          <li key={product.id}>
            <Link to={`/products/${product.id}`}>{product.name}</Link>
          </li>
```

```
        ))}
      </ul>
    </div>
  );
}

export default ProductList;



// src/components/ProductDetails.js
import React from 'react';
import { useParams } from 'react-router-dom';

const products = [
  { id: 1, name: 'Product 1', price: 100, description: 'This is product 1' },
  { id: 2, name: 'Product 2', price: 200, description: 'This is product 2' },
  { id: 3, name: 'Product 3', price: 300, description: 'This is product 3' },
];

function ProductDetails() {
  const { id } = useParams();
  const product = products.find((p) => p.id === parseInt(id));

  if (!product) {
    return <p>Product not found</p>;
  }

  return (
    <div>
      <h1>{product.name}</h1>
      <p>Price: ${product.price}</p>
      <p>Description: {product.description}</p>
    </div>
  );
}

export default ProductDetails;
```

Example 2:

Build a multi-step form (e.g., a registration form, Tinder signup form, etc.) where form data is shared across steps using Context.

Requirements:

1. Create a form with 3 steps: Personal Details, Address Details, and Confirmation.
2. Use Context to store and update form data as the user progresses through the steps.
3. Display a summary of the form data on the confirmation step.

Hinnts:

1. Create a `FormContext` to store form data and a function to update it.
2. Use `useContext` to access and update form data in each step.
3. Use `useNavigate` to navigate between steps.

```jsx
// src/context/FormContext.js
//Step One: Form Context
import { createContext, useState } from 'react';

const FormContext = createContext();

export const FormProvider = ({ children }) => {
  const [formData, setFormData] = useState({
    firstName: '',
    lastName: '',
    email: '',
    street: '',
    city: '',
    state: '',
    zip: '',
  });

  const updateFormData = (newData) => {
    setFormData((prevData) => ({ ...prevData, ...newData }));
  };

  return (
    <FormContext.Provider value={{ formData, updateFormData }}>
      {children}
    </FormContext.Provider>
  );
};

export default FormContext;




// src/components/PersonalDetails.js
//Step Two: Creating the Multi-Step Form Components
import React, { useContext, useState } from 'react';
import FormContext from '../context/FormContext';
import { useNavigate } from 'react-router-dom';

function PersonalDetails() {
```

```
  const { formData, updateFormData } = useContext(FormContext);
  const [firstName, setFirstName] = useState(formData.firstName);
  const [lastName, setLastName] = useState(formData.lastName);
  const [email, setEmail] = useState(formData.email);
  const navigate = useNavigate();

  const handleNext = () => {
    updateFormData({ firstName, lastName, email });
    navigate('/address');
  };

  return (
    <div>
      <h1>Personal Details</h1>
      <form>
        <input
          type="text"
          placeholder="First Name"
          value={firstName}
          onChange={(e) => setFirstName(e.target.value)}
        />
        <input
          type="text"
          placeholder="Last Name"
          value={lastName}
          onChange={(e) => setLastName(e.target.value)}
        />
        <input
          type="email"
          placeholder="Email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        />
        <button type="button" onClick={handleNext}>
          Next
        </button>
      </form>
    </div>
  );
}
```

```
export default PersonalDetails;




// src/components/AddressDetails.js
//Step Two: Creating the Multi-Step Form Components
import React, { useContext, useState } from 'react';
import FormContext from '../context/FormContext';
import { useNavigate } from 'react-router-dom';

function AddressDetails() {
  const { formData, updateFormData } = useContext(FormContext);
  const [street, setStreet] = useState(formData.street);
  const [city, setCity] = useState(formData.city);
  const [state, setState] = useState(formData.state);
  const [zip, setZip] = useState(formData.zip);
  const navigate = useNavigate();

  const handleNext = () => {
    updateFormData({ street, city, state, zip });
    navigate('/confirmation');
  };

  return (
    <div>
      <h1>Address Details</h1>
      <form>
        <input
          type="text"
          placeholder="Street"
          value={street}
          onChange={(e) => setStreet(e.target.value)}
        />
        <input
          type="text"
          placeholder="City"
          value={city}
          onChange={(e) => setCity(e.target.value)}
```

```
      />
      <input
        type="text"
        placeholder="State"
        value={state}
        onChange={(e) => setState(e.target.value)}
      />
      <input
        type="text"
        placeholder="ZIP Code"
        value={zip}
        onChange={(e) => setZip(e.target.value)}
      />
      <button type="button" onClick={handleNext}>
        Next
      </button>
    </form>
  </div>
 );
}

export default AddressDetails;


// src/components/Confirmation.js
//Step Two: Creating the Multi-Step Form Components
import React, { useContext } from 'react';
import FormContext from '../context/FormContext';
import { useNavigate } from 'react-router-dom';

function Confirmation() {
 const { formData } = useContext(FormContext);
 const navigate = useNavigate();

 const handleSubmit = () => {
  alert('Form submitted successfully!');
  navigate('/');
 };

 return (
```

```
    <div>
      <h1>Confirmation</h1>
      <div>
        <h2>Personal Details</h2>
        <p>First Name: {formData.firstName}</p>
        <p>Last Name: {formData.lastName}</p>
        <p>Email: {formData.email}</p>
      </div>
      <div>
        <h2>Address Details</h2>
        <p>Street: {formData.street}</p>
        <p>City: {formData.city}</p>
        <p>State: {formData.state}</p>
        <p>ZIP Code: {formData.zip}</p>
      </div>
      <button type="button" onClick={handleSubmit}>
        Submit
      </button>
    </div>
  );
}

export default Confirmation;




// src/App.js
//Step Three: Routing
import React from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import PersonalDetails from './components/PersonalDetails';
import AddressDetails from './components/AddressDetails';
import Confirmation from './components/Confirmation';
import { FormProvider } from './context/FormContext';

function App() {
  return (
    <FormProvider>
      <BrowserRouter>
```

```
      <Routes>
        <Route path="/" element={<PersonalDetails />} />
        <Route path="/address" element={<AddressDetails />} />
        <Route path="/confirmation" element={<Confirmation />} />
      </Routes>
    </BrowserRouter>
  </FormProvider>
 );
}

export default App;
```

Example 3:

Build an application that supports both theme (light/dark) and language (English/Spanish) toggling using Context.

Requirements:

1. Use Context to manage both theme and language settings.
2. Provide buttons to toggle the theme and language.
3. Update the UI text and styles based on the selected theme and language.

Hints:

1. Create a `SettingsContext` to store theme and language settings.

2. Use `useContext` to access and update the settings in the UI.
3. Use conditional rendering and CSS to apply the selected theme and language.

```
// src/context/SettingsContext.js
//Step One: Settings Context
import { createContext, useState } from 'react';

const SettingsContext = createContext();

export const SettingsProvider = ({ children }) => {
  const [theme, setTheme] = useState('light');
  const [language, setLanguage] = useState('english');

  const toggleTheme = () => {
    setTheme((prevTheme) => (prevTheme === 'light' ? 'dark' : 'light'));
  };

  const toggleLanguage = () => {
    setLanguage((prevLanguage) =>
      prevLanguage === 'english' ? 'spanish' : 'english'
    );
  };

  return (
    <SettingsContext.Provider value={{ theme, language, toggleTheme, toggleLanguage }}>
      {children}
    </SettingsContext.Provider>
  );
};

export default SettingsContext;



// src/App.js
//Step Two: Provider
import React, { useContext } from 'react';
```

```jsx
import { SettingsProvider } from './context/SettingsContext';
import Navbar from './components/Navbar';
import Content from './components/Content';

function App() {
  return (
    <SettingsProvider>
      <Navbar />
      <Content />
    </SettingsProvider>
  );
}

export default App;



// src/components/Navbar.js
//Step Three: Navbar Component
import React, { useContext } from 'react';
import SettingsContext from '../context/SettingsContext';

function Navbar() {
  const { theme, toggleTheme, language, toggleLanguage } = useContext(SettingsContext);

  return (
    <nav style={{ background: theme === 'light' ? '#fff' : '#333', color: theme === 'light' ? '#000' : '#fff' }}>
      <h1>{language === 'english' ? 'Navbar' : 'Barra de navegación'}</h1>
      <button onClick={toggleTheme}>
        {language === 'english' ? 'Toggle Theme' : 'Cambiar tema'}
      </button>
      <button onClick={toggleLanguage}>
        {language === 'english' ? 'Toggle Language' : 'Cambiar idioma'}
      </button>
    </nav>
  );
}

export default Navbar;
```

```jsx
// src/components/Content.js
//Step Four: Content Comonent
import React, { useContext } from 'react';
import SettingsContext from '../context/SettingsContext';

function Content() {
  const { theme, language } = useContext(SettingsContext);

  const englishContent = {
    title: 'Welcome to the App!',
    description: 'This is a simple app to demonstrate theme and language toggling.',
  };

  const spanishContent = {
    title: '¡Bienvenido a la aplicación!',
    description: 'Esta es una aplicación simple para demostrar el cambio de tema e idioma.',
  };

  const content = language === 'english' ? englishContent : spanishContent;

  return (
    <div style={{ background: theme === 'light' ? '#f5f5f5' : '#222', color: theme === 'light' ? '#000' : '#fff', padding:
'20px' }}>
      <h1>{content.title}</h1>
      <p>{content.description}</p>
    </div>
  );
}

export default Content;
```