Full Stack Developer

React Developer :

1.Ans:

Select Component (Select.js):

```
import React, { useState, useEffect } from 'react';
const Select = ({
  options = [],
  multiple = false,
  asyncLoadOptions,
  onChange,
  value = multiple ? [] : '',
  placeholder = 'Select...',
  style = {},
  renderOption,
  dropdownPosition = 'bottom'
}) => {
  const [selected, setSelected] = useState(value);
  const [loadedOptions, setLoadedOptions] = useState(options);
  const [isOpen, setIsOpen] = useState(false);
  useEffect(() => {
    if (asyncLoadOptions) {
      asyncLoadOptions().then(data => setLoadedOptions(data));
    }
  }, [asyncLoadOptions]);

  const handleSelection = (option) => {
    if (multiple) {
      const newSelection = selected.includes(option)
        ? selected.filter(item => item !== option)
        : [...selected, option];
      setSelected(newSelection);
      onChange(newSelection);
    } else {
      setSelected(option);
      onChange(option);
```

```
      setIsOpen(false);
    }
  };

  const renderDropdown = () => (
    <ul style={{ position: 'absolute', [dropdownPosition]: '100%' }}>
      {loadedOptions.map((option, index) => (
        <li
          key={index}
          style={{ cursor: 'pointer' }}
          onClick={() => handleSelection(option)}
        >
          {renderOption ? renderOption(option) : option}
        </li>
      ))}
    </ul>
  );
  return (
    <div style={{ position: 'relative', ...style }}>
      <div
        style={{ cursor: 'pointer', ...style }}
        onClick={() => setIsOpen(!isOpen)}
      >
        {multiple ? selected.join(', ') : selected || placeholder}
      </div>
      {isOpen && renderDropdown()}
    </div>
  );
};
export default Select;
```

Example Usage in a Form (App.js):

```
import React from 'react';
import Select from './Select';
const App = () => {
```

```
  const asyncLoadOptions = async () => {
    // Simulate an API call
    return new Promise((resolve) => {
      setTimeout(() => resolve(['Option 1', 'Option 2', 'Option 3']), 1000);
    });
  };
  const handleSingleChange = (value) => {
    console.log('Selected:', value);
  };
  const handleMultipleChange = (values) => {
    console.log('Selected:', values);
  };
  return (
    <div>
      <h2>Single Select</h2>
      <Select
        asyncLoadOptions={asyncLoadOptions}
        onChange={handleSingleChange}
        style={{ border: '1px solid #ccc', padding: '10px', width: '200px' }}
      />
      <h2>Multiple Select</h2>
      <Select
        multiple
        asyncLoadOptions={asyncLoadOptions}
        onChange={handleMultipleChange}
        style={{ border: '1px solid #ccc', padding: '10px', width: '200px' }}
      />
    </div>
  );
};

export default App;
```

2.Ans:

Form Component (Form.js):

```
import React from 'react';
import { useFormik } from 'formik';
import * as Yup from 'yup';

const Form = ({ fields, onSubmit, initialValues = {}, validationSchema }) => {
  const formik = useFormik({
    initialValues,
    validationSchema,
    onSubmit,
  });

  const renderField = (field) => {
    const { name, label, component: Component, ...rest } = field;
    const error = formik.errors[name] && formik.touched[name];

    return (
      <div key={name} style={{ marginBottom: '20px' }}>
        <label style={{ display: 'block', marginBottom: '5px' }}>{label}</label>
        {Component ? (
          <Component
            name={name}
            value={formik.values[name]}
            onChange={formik.handleChange}
            onBlur={formik.handleBlur}
            {...rest}
          />
        ) : (
          <input
            name={name}
            value={formik.values[name]}
            onChange={formik.handleChange}
            onBlur={formik.handleBlur}
            {...rest}
```

```
        />
      )}
      {error && <div style={{ color: 'red' }}>{formik.errors[name]}</div>}
    </div>
  );
 };

 return (
  <form onSubmit={formik.handleSubmit}>
   {fields.map(renderField)}
   <button type="submit">Submit</button>
   <button type="button" onClick={formik.handleReset}>
    Reset
   </button>
  </form>
 );
};
export default Form;
```

Example Usage with Form Component (App.js):

```
import React from 'react';
import * as Yup from 'yup';
import Form from './Form';

// Custom field components
const CustomInput = ({ name, value, onChange, onBlur, placeholder }) => (
 <input
  type="text"
  name={name}
  value={value}
  onChange={onChange}
  onBlur={onBlur}
  placeholder={placeholder}
  style={{ border: '1px solid #ccc', padding: '10px', width: '100%' }}
 />
```

```
);

const CustomSelect = ({ name, value, onChange, onBlur, options }) => (
  <select
    name={name}
    value={value}
    onChange={onChange}
    onBlur={onBlur}
    style={{ border: '1px solid #ccc', padding: '10px', width: '100%' }}
  >
    {options.map((option, index) => (
      <option key={index} value={option.value}>
        {option.label}
      </option>
    ))}
  </select>
);

const App = () => {
  const fields = [
    { name: 'firstName', label: 'First Name', component: CustomInput, placeholder: 'Enter your first
name' },
    { name: 'email', label: 'Email', component: CustomInput, type: 'email', placeholder: 'Enter your email'
},
    {
      name: 'age',
      label: 'Age',
      component: CustomInput,
      type: 'number',
      placeholder: 'Enter your age',
    },
    {
      name: 'gender',
      label: 'Gender',
      component: CustomSelect,
      options: [
        { value: '', label: 'Select Gender' },
```

```
      { value: 'male', label: 'Male' },
      { value: 'female', label: 'Female' },
    ],
  },
];

const initialValues = {
  firstName: '',
  email: '',
  age: '',
  gender: '',
};
const validationSchema = Yup.object({
  firstName: Yup.string().required('First Name is required'),
  email: Yup.string().email('Invalid email address').required('Email is required'),
  age: Yup.number().required('Age is required').min(0, 'Age must be a positive number'),
  gender: Yup.string().required('Gender is required'),
});
const handleSubmit = (values) => {
  console.log('Form Submitted:', values);
};
return (
  <div>
    <h2>Dynamic Form Example</h2>
    <Form
      fields={fields}
      initialValues={initialValues}
      validationSchema={validationSchema}
      onSubmit={handleSubmit}
    />
  </div>
);
};

export default App;
```

3.Ans:

Creating the Input Component:

```
import React from 'react';
import { useController } from 'react-hook-form';
const Input = ({
  name,
  control,
  type = 'text',
  label,
  placeholder,
  className = '',
  size = 'medium',
  shape = 'rounded',
  rules = {},
  ...props
}) => {
  const {
    field,
    fieldState: { error },
  } = useController({
    name,
    control,
    rules,
    defaultValue: '',
  });
  const inputClass = `input ${size} ${shape} ${className}`;
  return (
    <div className="input-wrapper">
      {label && <label htmlFor={name}>{label}</label>}
      <input
        {...field}
        type={type}
        placeholder={placeholder}
        className={inputClass}
        id={name}
```

```
        {...props}
      />
      {error ? (
        <p className="error-message">{error.message}</p>
      ) : (
        <p className="success-message">Looks good!</p>
      )}
    </div>
  );
};
export default Input;
```

Using the Input Component in a Form:

```
import React from 'react';
import { useForm, Controller } from 'react-hook-form';
import Input from './Input';

const MyForm = () => {
  const { control, handleSubmit } = useForm();

  const onSubmit = (data) => {
    console.log(data);
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <Input
        name="username"
        control={control}
        label="Username"
        placeholder="Enter your username"
        type="text"
        rules={{ required: 'Username is required' }}
        className="custom-input"
        size="large"
```

```
        shape="square"
      />
      <Input
        name="password"
        control={control}
        label="Password"
        placeholder="Enter your password"
        type="password"
        rules={{ required: 'Password is required' }}
        className="custom-input"
        size="large"
        shape="rounded"
      />
      <button type="submit">Submit</button>
    </form>
  );
};

export default MyForm;
```

4.Ans:

Creating the Filter Component:

```
import React, { useState } from 'react';
const Filter = ({ filters, onFilterChange }) => {
  const [filterValues, setFilterValues] = useState({});
  const handleChange = (name, value) => {
    const newFilters = { ...filterValues, [name]: value };
    setFilterValues(newFilters);
    onFilterChange(newFilters);
  };
  return (
    <div className="filter-container">
      {filters.map((filter) => {
        switch (filter.type) {
          case 'text':
```

```
    return (
      <input
        key={filter.name}
        type="text"
        placeholder={filter.label}
        value={filterValues[filter.name] || ''}
        onChange={(e) => handleChange(filter.name, e.target.value)}
      />
    );
  case 'select':
    return (
      <select
        key={filter.name}
        value={filterValues[filter.name] || ''}
        onChange={(e) => handleChange(filter.name, e.target.value)}
      >
        <option value="">{filter.label}</option>
        {filter.options.map((option) => (
          <option key={option.value} value={option.value}>
            {option.label}
          </option>
        ))}
      </select>
    );
  case 'date':
    return (
      <input
        key={filter.name}
        type="date"
        value={filterValues[filter.name] || ''}
        onChange={(e) => handleChange(filter.name, e.target.value)}
      />
    );
  default:
    return null;
}
```

```
      ))}
    </div>
  );
};
export default Filter;
```

Using the Filter Component in a Data-Driven Application:

```
import React, { useState, useEffect } from 'react';
import Filter from './Filter';
const ExampleApp = () => {
  const [items, setItems] = useState([]);
  const [filteredItems, setFilteredItems] = useState([]);
  const filters = [
    { name: 'search', type: 'text', label: 'Search...' },
    { name: 'category', type: 'select', label: 'Category', options: [
        { value: 'all', label: 'All' },
        { value: 'category1', label: 'Category 1' },
        { value: 'category2', label: 'Category 2' },
      ]
    },
    { name: 'startDate', type: 'date', label: 'Start Date' },
    { name: 'endDate', type: 'date', label: 'End Date' }
  ];

  useEffect(() => {
    // Fetch the data from an API or other source
    const fetchData = async () => {
      const data = await fetch('/api/items').then((res) => res.json());
      setItems(data);
      setFilteredItems(data);
    };
    fetchData();
  }, []);

  const handleFilterChange = (newFilters) => {
```

```
    let filtered = items;
    if (newFilters.search) {
      filtered = filtered.filter(item =>
        item.name.toLowerCase().includes(newFilters.search.toLowerCase())
      );
    }
    if (newFilters.category && newFilters.category !== 'all') {
      filtered = filtered.filter(item => item.category === newFilters.category);
    }
    if (newFilters.startDate) {
      filtered = filtered.filter(item => new Date(item.date) >= new Date(newFilters.startDate));
    }
    if (newFilters.endDate) {
      filtered = filtered.filter(item => new Date(item.date) <= new Date(newFilters.endDate));
    }
    setFilteredItems(filtered);
  };
  return (
    <div>
      <h1>Filterable List</h1>
      <Filter filters={filters} onFilterChange={handleFilterChange} />
      <ul>
        {filteredItems.map(item => (
          <li key={item.id}>{item.name}</li>
        ))}
      </ul>
    </div>
  );
};
export default ExampleApp;
```

5.Ans:

Creating the Modal Component:

```
import React, { useEffect, useRef } from 'react';
import ReactDOM from 'react-dom';
```

```
import './Modal.css';
const Modal = ({ isOpen, onClose, children, size = 'medium', position = 'center', backdrop = true }) => {
  const modalRef = useRef();
  useEffect(() => {
    if (isOpen) {
      // Lock scroll on the body
      document.body.style.overflow = 'hidden';
      // Focus on the modal for accessibility
      modalRef.current.focus();
    } else {
      document.body.style.overflow = 'unset';
    }
    return () => {
      document.body.style.overflow = 'unset';
    };
  }, [isOpen]);

  if (!isOpen) return null;
  const handleKeyDown = (e) => {
    if (e.key === 'Escape') {
      onClose();
    }
  };
  const modalClasses = `modal ${size} ${position}`;
  return ReactDOM.createPortal(
    <div className={`modal-backdrop ${backdrop ? 'visible' : ''}`} onClick={onClose}>
      <div
        className={modalClasses}
        ref={modalRef}
        tabIndex="-1"
        role="dialog"
        aria-modal="true"
        onKeyDown={handleKeyDown}
        onClick={(e) => e.stopPropagation()}
      >
        <button className="modal-close" onClick={onClose} aria-label="Close Modal">&times;</button>
```

```
        {children}
      </div>
    </div>,
    document.body
  );
};
export default Modal;
```

Example Usage Scenarios:

a. Confirmation Dialog:

```
import React, { useState } from 'react';
import Modal from './Modal';

const ConfirmationDialog = () => {
  const [isOpen, setIsOpen] = useState(false);

  const handleOpen = () => setIsOpen(true);
  const handleClose = () => setIsOpen(false);

  return (
    <div>
      <button onClick={handleOpen}>Open Confirmation Dialog</button>
      <Modal isOpen={isOpen} onClose={handleClose} size="small">
        <h2>Are you sure?</h2>
        <p>This action cannot be undone.</p>
        <button onClick={handleClose}>Cancel</button>
        <button onClick={() => { handleClose(); alert('Action confirmed!'); }}>Confirm</button>
      </Modal>
    </div>
  );
};

export default ConfirmationDialog;
```

b. Form Submission Modal:

```
import React, { useState } from 'react';
import Modal from './Modal';
const FormSubmissionModal = () => {
  const [isOpen, setIsOpen] = useState(false);

  const handleOpen = () => setIsOpen(true);
  const handleClose = () => setIsOpen(false);
  const handleSubmit = (e) => {
    e.preventDefault();
    alert('Form Submitted!');
    handleClose();
  };

  return (
    <div>
      <button onClick={handleOpen}>Open Form Modal</button>
      <Modal isOpen={isOpen} onClose={handleClose} size="medium">
        <h2>Submit Your Details</h2>
        <form onSubmit={handleSubmit}>
          <input type="text" placeholder="Enter your name" required />
```

```
            <input type="email" placeholder="Enter your email" required />
            <button type="submit">Submit</button>
          </form>
        </Modal>
      </div>
    );
  };

  export default FormSubmissionModal;
```

6.Ans:

Creating the Tabs Component:

```
import React, { useState } from 'react';
import './Tabs.css';

const Tabs = ({ tabs }) => {
  const [activeTab, setActiveTab] = useState(0);
  const handleKeyDown = (e, index) => {
    if (e.key === 'ArrowRight') {
      setActiveTab((prev) => (prev + 1) % tabs.length);
    } else if (e.key === 'ArrowLeft') {
      setActiveTab((prev) => (prev - 1 + tabs.length) % tabs.length);
    } else if (e.key === 'Enter') {
      setActiveTab(index);
    }
  };
  return (
    <div className="tabs">
      <div className="tab-list" role="tablist">
        {tabs.map((tab, index) => (
          <button
            key={index}
            role="tab"
            aria-selected={activeTab === index}
            aria-controls={`panel-${index}`}
            id={`tab-${index}`}
            tabIndex={activeTab === index ? 0 : -1}
            className={`tab ${activeTab === index ? 'active' : ''}`}
            onClick={() => setActiveTab(index)}
```

```
            onKeyDown={(e) => handleKeyDown(e, index)}
          >
            {tab.label}
          </button>
        ))}
      </div>
      <div
        id={`panel-${activeTab}`}
        role="tabpanel"
        aria-labelledby={`tab-${activeTab}`}
        className="tab-panel"
      >
        {tabs[activeTab].content}
      </div>
    </div>
  );
};

export default Tabs;
```

Example Usage with Dynamic Tabs and Nested Tabs:

```
import React from 'react';
import Tabs from './Tabs';
const NestedTabs = () => {
  const nestedTabs = [
    { label: 'Nested Tab 1', content: <p>Content of Nested Tab 1</p> },
    { label: 'Nested Tab 2', content: <p>Content of Nested Tab 2</p> },
  ];
  return <Tabs tabs={nestedTabs} />;
};
const ExampleApp = () => {
  const mainTabs = [
    {
      label: 'Tab 1',
      content: <p>Content of Tab 1</p>,
```

```
    },
    {
      label: 'Tab 2',
      content: (
        <div>
          <h3>Tab 2 with Nested Tabs</h3>
          <NestedTabs />
        </div>
      ),
    },
    {
      label: 'Tab 3',
      content: <p>Content of Tab 3</p>,
    },
  ];

  return (
    <div>
      <h1>Example with Tabs Component</h1>
      <Tabs tabs={mainTabs} />
    </div>
  );
};

export default ExampleApp;
```

7.Ans:

Creating the DataTable Component:

```
import React, { useState, useMemo } from 'react';
import './DataTable.css';

const DataTable = ({ columns, data, pageSize = 10 }) => {
  const [currentPage, setCurrentPage] = useState(1);
  const [sortConfig, setSortConfig] = useState(null);
  const [filterText, setFilterText] = useState('');
```

```
const sortedData = useMemo(() => {
  let sortableData = [...data];
  if (sortConfig !== null) {
    sortableData.sort((a, b) => {
      if (a[sortConfig.key] < b[sortConfig.key]) return sortConfig.direction === 'ascending' ? -1 : 1;
      if (a[sortConfig.key] > b[sortConfig.key]) return sortConfig.direction === 'ascending' ? 1 : -1;
      return 0;
    });
  }
  return sortableData;
}, [data, sortConfig]);

const filteredData = useMemo(() => {
  return sortedData.filter(row =>
    columns.some(column =>
      row[column.accessor].toString().toLowerCase().includes(filterText.toLowerCase())
    )
  );
}, [sortedData, filterText, columns]);

const paginatedData = useMemo(() => {
  const startIndex = (currentPage - 1) * pageSize;
  return filteredData.slice(startIndex, startIndex + pageSize);
}, [filteredData, currentPage, pageSize]);

const requestSort = (key) => {
  let direction = 'ascending';
  if (sortConfig && sortConfig.key === key && sortConfig.direction === 'ascending') {
    direction = 'descending';
  }
  setSortConfig({ key, direction });
};
const totalPages = Math.ceil(filteredData.length / pageSize);
return (
  <div>
```

```
      <input
        type="text"
        placeholder="Filter..."
        value={filterText}
        onChange={(e) => setFilterText(e.target.value)}
        className="data-table-filter"
      />
      <table className="data-table">
        <thead>
          <tr>
            {columns.map((column) => (
              <th key={column.accessor} onClick={() => requestSort(column.accessor)}>
                {column.label}
                {sortConfig?.key === column.accessor ? (sortConfig.direction === 'ascending' ? ' □' : ' □') : null}
              </th>
            ))}
          </tr>
        </thead>
        <tbody>
          {paginatedData.map((row, index) => (
            <tr key={index}>
              {columns.map((column) => (
                <td key={column.accessor}>
                  {column.render ? column.render(row[column.accessor], row) : row[column.accessor]}
                </td>
              ))}
            </tr>
          ))}
        </tbody>
      </table>
      <div className="pagination">
        <button onClick={() => setCurrentPage((prev) => Math.max(prev - 1, 1))} disabled={currentPage ===
1}>
          Previous
        </button>
        <span>Page {currentPage} of {totalPages}</span>
```

```
      <button onClick={() => setCurrentPage((prev) => Math.min(prev + 1, totalPages))}
disabled={currentPage === totalPages}>
        Next
      </button>
    </div>
  </div>
 );
};
export default DataTable;
```

Example Usage with Custom Columns and Expandable Rows:

```
import React, { useState } from 'react';
import DataTable from './DataTable';

const ExampleApp = () => {
 const [expandedRow, setExpandedRow] = useState(null);

 const columns = [
  { label: 'ID', accessor: 'id' },
  { label: 'Name', accessor: 'name' },
  { label: 'Age', accessor: 'age' },
  {
   label: 'Actions',
   accessor: 'actions',
   render: (value, row) => (
    <button onClick={() => setExpandedRow(expandedRow === row.id ? null : row.id)}>
     {expandedRow === row.id ? 'Collapse' : 'Expand'}
    </button>
   ),
  },
 ];

 const data = [
  { id: 1, name: 'John Doe', age: 28, details: 'John is a software engineer.' },
  { id: 2, name: 'Jane Smith', age: 34, details: 'Jane is a project manager.' },
```

```
    { id: 3, name: 'Mike Johnson', age: 45, details: 'Mike is a designer.' },
    // Add more data here...
  ];

  return (
    <div>
      <h1>DataTable Example</h1>
      <DataTable
        columns={columns}
        data={data}
        pageSize={5}
      />
      {expandedRow && (
        <div className="expanded-content">
          <h2>Details for ID {expandedRow}</h2>
          <p>{data.find(row => row.id === expandedRow)?.details}</p>
        </div>
      )}
    </div>
  );
};

export default ExampleApp;
```

8.Ans:

Creating the Notification Component:

```
import React, { useState, useEffect, useContext, createContext } from 'react';
import './Notification.css';
const NotificationContext = createContext();
export const useNotification = () => {
  return useContext(NotificationContext);
};

const NotificationProvider = ({ children }) => {
  const [notifications, setNotifications] = useState([]);
```

```
  const addNotification = (message, type = 'info', duration = 3000) => {
    const id = Math.random().toString(36).substr(2, 9);
    setNotifications([...notifications, { id, message, type, duration }]);
  };
  const removeNotification = (id) => {
    setNotifications(notifications.filter(notification => notification.id !== id));
  };

  useEffect(() => {
    notifications.forEach(notification => {
      if (notification.duration) {
        const timer = setTimeout(() => removeNotification(notification.id), notification.duration);
        return () => clearTimeout(timer);
      }
    });
  }, [notifications]);

  return (
    <NotificationContext.Provider value={addNotification}>
      <div className="notification-container">
        {notifications.map(notification => (
          <div key={notification.id} className={`notification ${notification.type}`}>
            {notification.message}
            <button onClick={() => removeNotification(notification.id)}>&times;</button>
          </div>
        ))}
      </div>
      {children}
    </NotificationContext.Provider>
  );
};

export default NotificationProvider;
```

Example Usage:

```
import React from 'react';
import NotificationProvider, { useNotification } from './NotificationProvider';

const ExampleApp = () => {
  const addNotification = useNotification();

  const handleAction = (type) => {
    addNotification(`This is a ${type} notification!`, type);
  };

  return (
    <div>
      <h1>Notification Example</h1>
      <button onClick={() => handleAction('success')}>Show Success</button>
      <button onClick={() => handleAction('error')}>Show Error</button>
      <button onClick={() => handleAction('info')}>Show Info</button>
      <button onClick={() => handleAction('warning')}>Show Warning</button>
    </div>
  );
};

const App = () => (
  <NotificationProvider>
    <ExampleApp />
  </NotificationProvider>
);

export default App;
```

9.Ans:

Creating the Wizard Component:

```
import React, { useState } from 'react';

const Step = ({ children }) => <div>{children}</div>;
```

```
const Wizard = ({ children, initialData = {} }) => {
  const [currentStep, setCurrentStep] = useState(0);
  const [formData, setFormData] = useState(initialData);

  const nextStep = () => setCurrentStep(prev => Math.min(prev + 1, children.length - 1));
  const prevStep = () => setCurrentStep(prev => Math.max(prev - 1, 0));

  const updateData = (newData) => {
    setFormData({ ...formData, ...newData });
  };
  const isLastStep = currentStep === children.length - 1;
  return (
    <div>
      <div className="stepper">
        {React.Children.map(children, (child, index) => (
          <div className={`step ${index === currentStep ? 'active' : ''}`}>
            Step {index + 1}
          </div>
        ))}
      </div>
      {React.cloneElement(children[currentStep], { updateData, formData })}
      <div className="navigation">
        <button onClick={prevStep} disabled={currentStep === 0}>Back</button>
        <button onClick={nextStep}>{isLastStep ? 'Finish' : 'Next'}</button>
      </div>
    </div>
  );
};

// Example Usage

const Step1 = ({ updateData, formData }) => (
  <Step>
    <h2>Step 1: Personal Info</h2>
    <input
      type="text"
```

```
      placeholder="First Name"
      value={formData.firstName || ''}
      onChange={(e) => updateData({ firstName: e.target.value })}
    />
  </Step>
);

const Step2 = ({ updateData, formData }) => (
  <Step>
    <h2>Step 2: Contact Info</h2>
    <input
      type="email"
      placeholder="Email"
      value={formData.email || ''}
      onChange={(e) => updateData({ email: e.target.value })}
    />
  </Step>
);

const Step3 = ({ updateData, formData }) => (
  <Step>
    <h2>Step 3: Review & Submit</h2>
    <p><strong>First Name:</strong> {formData.firstName}</p>
    <p><strong>Email:</strong> {formData.email}</p>
    <button onClick={() => alert('Form Submitted!')}>Submit</button>
  </Step>
);

const ExampleApp = () => (
  <Wizard>
    <Step1 />
    <Step2 />
    <Step3 />
  </Wizard>
);
export default ExampleApp;
```

10.Ans:

Creating the Carousel Component:

```
import React, { useState, useEffect } from 'react';
import './Carousel.css';

const Carousel = ({ children, autoSlide = true, slideInterval = 3000 }) => {
  const [currentSlide, setCurrentSlide] = useState(0);

  useEffect(() => {
    if (autoSlide) {
      const timer = setInterval(() => {
        setCurrentSlide((prev) => (prev + 1) % children.length);
      }, slideInterval);
      return () => clearInterval(timer);
    }
  }, [autoSlide, slideInterval, children.length]);

  const nextSlide = () => {
    setCurrentSlide((prev) => (prev + 1) % children.length);
  };

  const prevSlide = () => {
    setCurrentSlide((prev) => (prev - 1 + children.length) % children.length);
  };

  return (
    <div className="carousel">
      <div className="carousel-slides" style={{ transform: `translateX(-${currentSlide * 100}%)` }}>
        {children.map((child, index) => (
          <div className="carousel-slide" key={index}>
            {child}
          </div>
        ))}
      </div>
```

```
      <button className="carousel-control prev" onClick={prevSlide}>‹</button>
      <button className="carousel-control next" onClick={nextSlide}>›</button>
      <div className="carousel-dots">
        {children.map((_, index) => (
          <button
            key={index}
            className={`carousel-dot ${index === currentSlide ? 'active' : ''}`}
            onClick={() => setCurrentSlide(index)}
          />
        ))}
      </div>
    </div>
  );
};

export default Carousel;
```

Example Usage:

```
import React from 'react';
import Carousel from './Carousel';

const ExampleApp = () => {
  return (
    <div>
      <h1>Simple Carousel Example</h1>
      <Carousel autoSlide={true} slideInterval={2000}>
        <div style={{ backgroundColor: 'lightcoral', height: '200px' }}>
          <h2>Slide 1</h2>
          <p>This is the first slide.</p>
        </div>
        <div style={{ backgroundColor: 'lightblue', height: '200px' }}>
          <h2>Slide 2</h2>
          <p>This is the second slide.</p>
        </div>
        <div style={{ backgroundColor: 'lightgreen', height: '200px' }}>
          <h2>Slide 3</h2>
```

```
          <p>This is the third slide.</p>
        </div>
      </Carousel>
    </div>
  );
};

export default ExampleApp;
```