Question 1

XYZ Enterprises wants to develop a web-based Employee Management System to handle employee data efficiently. They request your help to build the system using React and MongoDB as per the following specifications:

1. Create a MongoDB database named Employees to store details like EmployeeName, EmployeeID (unique for each employee), Designation, Department, and JoiningDate

2. In the React application, create three pages:
•Add Employee: This page allows users to input employee details and save them to the database.
•Search Employee: This page allows users to enter the EmployeeID and display the corresponding employee details (if found) or an appropriate error message in a textarea.
•Update Employee: This page allows users to update the designation of an employee. It should take EmployeeID as input and provide a form to update the Designation field in the database.

3. Create a homepage to navigate between the three pages mentioned above.

Specifications:

•Use React Router for navigation between the pages.
•Implement the REST API using Node.js and Express for adding, searching, and updating employee details

Solution :

App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Link, Routes } from 'react-router-dom';
import AddEmployee from './pages/AddEmployee';
import SearchEmployee from './pages/SearchEmployee';
import UpdateEmployee from './pages/UpdateEmployee';
```

```
import ViewEmployees from './pages/ViewEmployees';

function App() {
    return (
      <Router>
        <div>
          <h1>Employee Management System</h1>
          <nav>
            <ul>
              <li><Link to="/add">Add Employee</Link></li>
              <li><Link to="/search">Search Employee</Link></li>
              <li><Link to="/update">Update Employee</Link></li>
              <li><Link to="/view">View All Employees</Link></li> {/* New link */}
            </ul>
          </nav>
          <Routes>
            <Route path="/add" element={<AddEmployee />} />
            <Route path="/search" element={<SearchEmployee />} />
            <Route path="/update" element={<UpdateEmployee />} />
            <Route path="/view" element={<ViewEmployees />} /> {/* New route */}
          </Routes>
        </div>
      </Router>
    );
  }

export default App;
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bodyParser = require('body-parser');

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/Employees')
```

```
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.log(err));

// Create Employee Schema
const employeeSchema = new mongoose.Schema({
  EmployeeName: String,
  EmployeeID: { type: String, unique: true },
  Designation: String,
  Department: String,
  JoiningDate: Date
});

const Employee = mongoose.model('Employee', employeeSchema);

const app = express();
app.use(cors());
app.use(bodyParser.json());

// Add Employee
app.post('/add-employee', async (req, res) => {
  const employeeData = req.body;
  try {
    const employee = new Employee(employeeData);
    await employee.save();
    res.status(201).send("Employee added successfully");
  } catch (error) {
    res.status(400).send("Error adding employee");
  }
});

// Get all Employees
app.get('/employees', async (req, res) => {
    try {
      const employees = await Employee.find();
      res.status(200).json(employees);
    } catch (error) {
      res.status(400).send("Error fetching employee records");
```

```
    }
  });

// Search Employee by EmployeeID
app.get('/search-employee/:EmployeeID', async (req, res) => {
  try {
    const employee = await Employee.findOne({ EmployeeID:
req.params.EmployeeID });
    if (employee) {
      res.status(200).json(employee);
    } else {
      res.status(404).send("Employee not found");
    }
  } catch (error) {
    res.status(400).send("Error searching for employee");
  }
});

// Update Employee Designation
app.put('/update-employee/:EmployeeID', async (req, res) => {
  try {
    const { Designation } = req.body;
    const employee = await Employee.findOneAndUpdate({ EmployeeID:
req.params.EmployeeID }, { Designation }, { new: true });
    if (employee) {
      res.status(200).send("Designation updated successfully");
    } else {
      res.status(404).send("Employee not found");
    }
  } catch (error) {
    res.status(400).send("Error updating employee designation");
  }
});

// Start the server
const PORT = 3001;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

AddEmployee.js

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

function AddEmployee() {
  const [employeeData, setEmployeeData] = useState({
    EmployeeName: '',
    EmployeeID: '',
    Designation: '',
    Department: '',
    JoiningDate: ''
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setEmployeeData({ ...employeeData, [name]: value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post('http://localhost:3001/add-employee', employeeData);
      alert('Employee added successfully');
    } catch (error) {
      alert('Error adding employee');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" name="EmployeeName" placeholder="Employee Name"
onChange={handleChange} required />
      <input type="text" name="EmployeeID" placeholder="Employee ID"
onChange={handleChange} required />
```

```
      <input type="text" name="Designation" placeholder="Designation"
onChange={handleChange} required />
      <input type="text" name="Department" placeholder="Department"
onChange={handleChange} required />
      <input type="date" name="JoiningDate" onChange={handleChange} required
/>
      <button type="submit">Add Employee</button>
    </form>
  );
}

export default AddEmployee;
```

SearchEmployee.js

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

function SearchEmployee() {
  const [employeeID, setEmployeeID] = useState('');
  const [employee, setEmployee] = useState(null);
  const [error, setError] = useState('');

  const handleSearch = async () => {
    try {
      const response = await
axios.get(`http://localhost:3001/search-employee/${employeeID}`);
      setEmployee(response.data);
      setError('');
    } catch (error) {
      setEmployee(null);
      setError('Employee not found');
    }
  };

  return (
    <div>
```

```
      <input type="text" placeholder="Enter Employee ID" value={employeeID}
onChange={(e) => setEmployeeID(e.target.value)} />
      <button onClick={handleSearch}>Search Employee</button>
      {employee && (
        <textarea rows="4" value={JSON.stringify(employee, null, 2)} readOnly />
      )}
      {error && <p>{error}</p>}
    </div>
  );
}

export default SearchEmployee;
```

UpdateEmployee.js

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

function UpdateEmployee() {
  const [employeeID, setEmployeeID] = useState('');
  const [designation, setDesignation] = useState('');

  const handleUpdate = async () => {
    try {
      await axios.put(`http://localhost:3001/update-employee/${employeeID}`, {
Designation: designation });
      alert('Designation updated successfully');
    } catch (error) {
      alert('Error updating designation');
    }
  };

  return (
    <div>
      <input type="text" placeholder="Enter Employee ID" value={employeeID}
onChange={(e) => setEmployeeID(e.target.value)} />
```

```
      <input type="text" placeholder="Enter New Designation" value={designation}
onChange={(e) => setDesignation(e.target.value)} />
      <button onClick={handleUpdate}>Update Designation</button>
    </div>
  );
}

export default UpdateEmployee;
```

ViewEmployee.js

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

function ViewEmployees() {
  const [employees, setEmployees] = useState([]);

  useEffect(() => {
    const fetchEmployees = async () => {
      try {
        const response = await axios.get('http://localhost:3001/employees');
        setEmployees(response.data);
      } catch (error) {
        alert('Error fetching employees');
      }
    };
    fetchEmployees();
  }, []);

  return (
    <div>
      <h2>Employee Records</h2>
      {employees.length > 0 ? (
        <table>
          <thead>
            <tr>
              <th>Employee Name</th>
```

```
            <th>Employee ID</th>
            <th>Designation</th>
            <th>Department</th>
            <th>Joining Date</th>
          </tr>
        </thead>
        <tbody>
          {employees.map((employee) => (
            <tr key={employee.EmployeeID}>
              <td>{employee.EmployeeName}</td>
              <td>{employee.EmployeeID}</td>
              <td>{employee.Designation}</td>
              <td>{employee.Department}</td>
              <td>{new Date(employee.JoiningDate).toLocaleDateString()}</td>
            </tr>
          ))}
        </tbody>
      </table>
    ) : (
      <p>No employee records found.</p>
    )}
  </div>
 );
}

export default ViewEmployees;
```

Question 2

LMN Co. is working on a Book Inventory Management System where they wish to store information about books in a MongoDB database and provide a React-based interface for easy management. As the development expert, you are tasked to build the application with the following features:

1. Create a MongoDB database named Books with fields BookTitle, ISBN, Author, Category, and Quantity.

2. Develop four pages using React:

•Add Book: This page should accept book information from the user and store it in the database.

•Search Book by ISBN: Allow the user to search for a book using the ISBN number.

•Delete Book: Allow the user to delete a book by entering its ISBN.

•Display All Books: Retrieve all book entries from the MongoDB database and display them in a textarea.

3. Create appropriate REST API routes in Express to handle the functionality of adding, searching, deleting, and retrieving books.

4. Create a homepage to navigate between the three pages mentioned above.

Specifications:

• Use React Hooks (useState and useEffect) to manage state and data fetching.

• Implement navigation using React Router.

Solution :

App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Link, Routes } from 'react-router-dom';
import AddBook from './pages/AddBook';
import SearchBook from './pages/SearchBook';
import DeleteBook from './pages/DeleteBook';
import ViewBooks from './pages/ViewBooks';

function App() {
  return (
    <Router>
      <div>
        <h1>Book Inventory Management</h1>
        <nav>
```

```
          <ul>
            <li><Link to="/add">Add Book</Link></li>
            <li><Link to="/search">Search Book by ISBN</Link></li>
            <li><Link to="/delete">Delete Book</Link></li>
            <li><Link to="/view">Display All Books</Link></li>
          </ul>
        </nav>
        <Routes>
          <Route path="/add" element={<AddBook />} />
          <Route path="/search" element={<SearchBook />} />
          <Route path="/delete" element={<DeleteBook />} />
          <Route path="/view" element={<ViewBooks />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bodyParser = require('body-parser');

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/Books')
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.log(err));

// Create Book Schema
const bookSchema = new mongoose.Schema({
  BookTitle: String,
  ISBN: { type: String, unique: true },
```

```
  Author: String,
  Category: String,
  Quantity: Number
});

const Book = mongoose.model('Book', bookSchema);

const app = express();
app.use(cors());
app.use(bodyParser.json());

// Add Book
app.post('/add-book', async (req, res) => {
  const bookData = req.body;
  try {
    const book = new Book(bookData);
    await book.save();
    res.status(201).send("Book added successfully");
  } catch (error) {
    res.status(400).send("Error adding book");
  }
});

// Search Book by ISBN
app.get('/search-book/:ISBN', async (req, res) => {
  try {
    const book = await Book.findOne({ ISBN: req.params.ISBN });
    if (book) {
      res.status(200).json(book);
    } else {
      res.status(404).send("Book not found");
    }
  } catch (error) {
    res.status(400).send("Error searching for book");
  }
});
```

```
// Delete Book by ISBN
app.delete('/delete-book/:ISBN', async (req, res) => {
  try {
    const book = await Book.findOneAndDelete({ ISBN: req.params.ISBN });
    if (book) {
      res.status(200).send("Book deleted successfully");
    } else {
      res.status(404).send("Book not found");
    }
  } catch (error) {
    res.status(400).send("Error deleting book");
  }
});

// Get all Books
app.get('/books', async (req, res) => {
  try {
    const books = await Book.find();
    res.status(200).json(books);
  } catch (error) {
    res.status(400).send("Error fetching books");
  }
});

// Start the server
const PORT = 3001
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

AddBook.js

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

function AddBook() {
  const [bookData, setBookData] = useState({
    BookTitle: '',
    ISBN: '',
```

```
    Author: '',
    Category: '',
    Quantity: 0
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setBookData({ ...bookData, [name]: value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post('http://localhost:3001/add-book', bookData);
      alert('Book added successfully');
    } catch (error) {
      alert('Error adding book');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" name="BookTitle" placeholder="Book Title"
onChange={handleChange} required />
      <input type="text" name="ISBN" placeholder="ISBN"
onChange={handleChange} required />
      <input type="text" name="Author" placeholder="Author"
onChange={handleChange} required />
      <input type="text" name="Category" placeholder="Category"
onChange={handleChange} required />
      <input type="number" name="Quantity" placeholder="Quantity"
onChange={handleChange} required />
      <button type="submit">Add Book</button>
    </form>
  );
}
```

```
export default AddBook;
```

SearchBook.js

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

function SearchBook() {
  const [ISBN, setISBN] = useState('');
  const [book, setBook] = useState(null);
  const [error, setError] = useState('');

  const handleSearch = async () => {
    try {
      const response = await
axios.get(`http://localhost:3001/search-book/${ISBN}`);
      setBook(response.data);
      setError('');
    } catch (error) {
      setBook(null);
      setError('Book not found');
    }
  };

  return (
    <div>
      <input type="text" placeholder="Enter ISBN" value={ISBN} onChange={(e) =>
setISBN(e.target.value)} />
      <button onClick={handleSearch}>Search Book</button>
      {book && (
        <textarea rows="4" value={JSON.stringify(book, null, 2)} readOnly />
      )}
      {error && <p>{error}</p>}
    </div>
  );
}
```

```
export default SearchBook;
```

DeleteBook.js

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

function DeleteBook() {
  const [ISBN, setISBN] = useState('');

  const handleDelete = async () => {
    try {
      await axios.delete(`http://localhost:3001/delete-book/${ISBN}`);
      alert('Book deleted successfully');
    } catch (error) {
      alert('Error deleting book');
    }
  };

  return (
    <div>
      <input type="text" placeholder="Enter ISBN" value={ISBN} onChange={(e) =>
setISBN(e.target.value)} />
      <button onClick={handleDelete}>Delete Book</button>
    </div>
  );
}

export default DeleteBook;
```

ViewBook.js

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

function ViewBooks() {
  const [books, setBooks] = useState([]);
```

```
  useEffect(() => {
    const fetchBooks = async () => {
      try {
        const response = await axios.get('http://localhost:3001/books');
        setBooks(response.data);
      } catch (error) {
        alert('Error fetching books');
      }
    };
    fetchBooks();
  }, []);

  return (
    <div>
      <h2>All Books</h2>
      {books.length > 0 ? (
        <textarea rows="20" value={JSON.stringify(books, null, 2)} readOnly />
      ) : (
        <p>No books found.</p>
      )}
    </div>
  );
}

export default ViewBooks;
```

Question 4

DEF Corporation wants to build a Student Enrollment System that will allow administrators to manage student data. As a full-stackdeveloper, you are required to set up a modular system with Webpack for bundling:

1. Set up Webpack to optimise the build process and enable Hot Module Replacement (HMR).

2. Create a MongoDB database named Students with fields StudentName, StudentID (unique), Course, Year, and Email.

3. Develop four React pages:

• Add Student: Allows users to input student details and store them in the database.

• Search Student by ID: Users can search for a student using StudentID and display their details if found.

• Update Student: Enable users to update the Course and Year of a student using their StudentID.

• Display All Students: Display all students' information from the database.

4. Modularize your code to separate the React components and Express routes.

5. Implement a REST API for handling CRUD operations and integrate React Router for navigation between the pages.

Specifications:

● Ensure all API routes and React components are modularized into separate files.
● UseReact Hooks for managing state and data fetching.
● Implement navigation using React Router and manage application state using React Hooks.

Solution :

App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Link, Routes } from 'react-router-dom';
import AddStudent from './pages/AddStudent.js';
import SearchStudent from './pages/SearchStudent';
import UpdateStudent from './pages/UpdateStudent';
import ViewStudents from './pages/ViewStudents';

function App() {
```

```
  return (
    <Router>
      <div>
        <h1>Student Enrollment System</h1>
        <nav>
          <ul>
            <li><Link to="/add">Add Student</Link></li>
            <li><Link to="/search">Search Student by ID</Link></li>
            <li><Link to="/update">Update Student</Link></li>
            <li><Link to="/view">View All Students</Link></li>
          </ul>
        </nav>
        <Routes>
          <Route path="/add" element={<AddStudent />} />
          <Route path="/search" element={<SearchStudent />} />
          <Route path="/update" element={<UpdateStudent />} />
          <Route path="/view" element={<ViewStudents />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bodyParser = require('body-parser');

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/Students')
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.log(err));
```

```
// Create Student Schema and Model
const studentSchema = new mongoose.Schema({
  StudentName: String,
  StudentID: { type: String, unique: true },
  Course: String,
  Year: Number,
  Email: String
});

const Student = mongoose.model('Student', studentSchema);

const app = express();
app.use(cors());
app.use(bodyParser.json());

// Add Student
app.post('/api/students/add', async (req, res) => {
  const studentData = req.body;
  try {
    const student = new Student(studentData);
    await student.save();
    res.status(201).send("Student added successfully");
  } catch (error) {
    res.status(400).send("Error adding student");
  }
});

// Search Student by ID
app.get('/api/students/search/:StudentID', async (req, res) => {
  try {
    const student = await Student.findOne({ StudentID: req.params.StudentID });
    if (student) {
      res.status(200).json(student);
    } else {
      res.status(404).send("Student not found");
    }
  } catch (error) {
```

```
    res.status(400).send("Error searching for student");
  }
});

// Update Student
app.put('/api/students/update/:StudentID', async (req, res) => {
  try {
    const { Course, Year } = req.body;
    const student = await Student.findOneAndUpdate(
      { StudentID: req.params.StudentID },
      { Course, Year },
      { new: true }
    );
    if (student) {
      res.status(200).send("Student updated successfully");
    } else {
      res.status(404).send("Student not found");
    }
  } catch (error) {
    res.status(400).send("Error updating student");
  }
});

// Get All Students
app.get('/api/students', async (req, res) => {
  try {
    const students = await Student.find();
    res.status(200).json(students);
  } catch (error) {
    res.status(400).send("Error fetching students");
  }
});

// Delete Student by ID
app.delete('/api/students/delete/:StudentID', async (req, res) => {
  try {
```

```
    const student = await Student.findOneAndDelete({ StudentID:
req.params.StudentID });
    if (student) {
      res.status(200).send("Student deleted successfully");
    } else {
      res.status(404).send("Student not found");
    }
  } catch (error) {
    res.status(400).send("Error deleting student");
  }
});

// Start the server
const PORT = 3001;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

AddStudent.js

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

function AddStudent() {
  const [studentData, setStudentData] = useState({
    StudentName: '',
    StudentID: '',
    Course: '',
    Year: '',
    Email: ''
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setStudentData({ ...studentData, [name]: value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
```

```
    try {
      await axios.post('http://localhost:3001/api/students/add', studentData);
      alert('Student added successfully');
    } catch (error) {
      alert('Error adding student');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" name="StudentName" placeholder="Student Name"
onChange={handleChange} required />
      <input type="text" name="StudentID" placeholder="Student ID"
onChange={handleChange} required />
      <input type="text" name="Course" placeholder="Course"
onChange={handleChange} required />
      <input type="number" name="Year" placeholder="Year"
onChange={handleChange} required />
      <input type="email" name="Email" placeholder="Email"
onChange={handleChange} required />
      <button type="submit">Add Student</button>
    </form>
  );
}

export default AddStudent;
```

SearchStudent.js

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

function SearchStudent() {
  const [studentID, setStudentID] = useState('');
  const [student, setStudent] = useState(null);
  const [error, setError] = useState('');
```

```
  const handleSearch = async () => {
    try {
      const response = await
axios.get(`http://localhost:3001/api/students/search/${studentID}`);
      setStudent(response.data);
      setError('');
    } catch (error) {
      setStudent(null);
      setError('Student not found');
    }
  };

  return (
    <div>
      <input
        type="text"
        placeholder="Enter Student ID"
        value={studentID}
        onChange={(e) => setStudentID(e.target.value)}
      />
      <button onClick={handleSearch}>Search Student</button>

      {student && (
        <div>
          <p><strong>Name:</strong> {student.StudentName}</p>
          <p><strong>Course:</strong> {student.Course}</p>
          <p><strong>Year:</strong> {student.Year}</p>
          <p><strong>Email:</strong> {student.Email}</p>
        </div>
      )}
      {error && <p>{error}</p>}
    </div>
  );
}

export default SearchStudent;
```

UpdateStudent.js

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

function UpdateStudent() {
  const [studentID, setStudentID] = useState('');
  const [course, setCourse] = useState('');
  const [year, setYear] = useState('');

  const handleUpdate = async () => {
    try {
      await axios.put(`http://localhost:3001/api/students/update/${studentID}`, {
Course: course, Year: year });
      alert('Student updated successfully');
    } catch (error) {
      alert('Error updating student');
    }
  };

  return (
    <div>
      <input
        type="text"
        placeholder="Enter Student ID"
        value={studentID}
        onChange={(e) => setStudentID(e.target.value)}
      />
      <input
        type="text"
        placeholder="Enter New Course"
        value={course}
        onChange={(e) => setCourse(e.target.value)}
      />
      <input
        type="number"
        placeholder="Enter New Year"
```

```
        value={year}
        onChange={(e) => setYear(e.target.value)}
      />
      <button onClick={handleUpdate}>Update Student</button>
    </div>
  );
}

export default UpdateStudent;
```

ViewStudent.js

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

function ViewStudents() {
  const [students, setStudents] = useState([]);

  useEffect(() => {
    const fetchStudents = async () => {
      try {
        const response = await axios.get('http://localhost:3001/api/students');
        setStudents(response.data);
      } catch (error) {
        alert('Error fetching students');
      }
    };
    fetchStudents();
  }, []);

  return (
    <div>
      <h2>All Students</h2>
      {students.length > 0 ? (
        <ul>
          {students.map((student) => (
            <li key={student.StudentID}>
```

```
            {student.StudentName} - {student.Course} ({student.Year}) -
{student.Email}
          </li>
        ))}
      </ul>
    ) : (
      <p>No students found.</p>
    )}
  </div>
 );
}

export default ViewStudents;
```

Question 5

GHI Enterprises is developing an Online Order Management System to handle customer orders efficiently. As the developer, you are tasked with building the following features:

1. MongoDB Database:
•Create a database named Orders with fields OrderID (unique), CustomerName, Product, Quantity, and OrderDate.

2. React Pages:
● Add Order: Design a form with fields for CustomerName, Product, Quantity, and OrderDate.
● Search Order by OrderID: Create a search form with an input field for OrderID. Display the order details.
● Delete Order: Implement a form where users can enter the OrderID to delete an order.
● Update Order Quantity: Design a form where users can input an OrderID and the new quantity. Submit the updated quantity to the database.

3. Routing and State Management:
● Use React Router to navigate between the “Add Order”, “Search Order”, “Delete Order”, and “Update Order Quantity” pages.

● Manage form states using React Hooks (useState, useEffect) to control input fields and handle data submission and retrieval.

Specifications:

● Integrate React Hook Form or a similar form library for managing form data and validation.
● UseReact Hooks for managing state and data fetching.
● Implement navigation using React Router and manage application state using React Hooks.

Solution :

App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Link, Routes } from 'react-router-dom';
import AddOrder from './pages/AddOrder';
import SearchOrder from './pages/SearchOrder';
import DeleteOrder from './pages/DeleteOrder';
import UpdateOrder from './pages/UpdateOrder';

function App() {
  return (
    <Router>
      <div>
        <h1>Order Management System</h1>
        <nav>
          <ul>
            <li><Link to="/add">Add Order</Link></li>
            <li><Link to="/search">Search Order by ID</Link></li>
            <li><Link to="/delete">Delete Order</Link></li>
            <li><Link to="/update">Update Order Quantity</Link></li>
          </ul>
        </nav>
        <Routes>
```

```
          <Route path="/add" element={<AddOrder />} />
          <Route path="/search" element={<SearchOrder />} />
          <Route path="/delete" element={<DeleteOrder />} />
          <Route path="/update" element={<UpdateOrder />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bodyParser = require('body-parser');

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/Orders')
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.log(err));

// Create Order Schema and Model
const orderSchema = new mongoose.Schema({
  OrderID: { type: String, unique: true },
  CustomerName: String,
  Product: String,
  Quantity: Number,
  OrderDate: Date
});

const Order = mongoose.model('Order', orderSchema);

const app = express();
app.use(cors());
```

```
app.use(bodyParser.json());

// Add Order
app.post('/api/orders/add', async (req, res) => {
  const orderData = req.body;
  try {
    const order = new Order(orderData);
    await order.save();
    res.status(201).send("Order added successfully");
  } catch (error) {
    res.status(400).send("Error adding order");
  }
});

// Search Order by ID
app.get('/api/orders/search/:OrderID', async (req, res) => {
  try {
    const order = await Order.findOne({ OrderID: req.params.OrderID });
    if (order) {
      res.status(200).json(order);
    } else {
      res.status(404).send("Order not found");
    }
  } catch (error) {
    res.status(400).send("Error searching for order");
  }
});

// Delete Order by ID
app.delete('/api/orders/delete/:OrderID', async (req, res) => {
  try {
    const order = await Order.findOneAndDelete({ OrderID: req.params.OrderID });
    if (order) {
      res.status(200).send("Order deleted successfully");
    } else {
      res.status(404).send("Order not found");
    }
```

```
  } catch (error) {
    res.status(400).send("Error deleting order");
  }
});

// Update Order Quantity by OrderID
app.put('/api/orders/update/:OrderID', async (req, res) => {
  try {
    const { Quantity } = req.body;
    const order = await Order.findOneAndUpdate(
      { OrderID: req.params.OrderID },
      { Quantity },
      { new: true }
    );
    if (order) {
      res.status(200).send("Order updated successfully");
    } else {
      res.status(404).send("Order not found");
    }
  } catch (error) {
    res.status(400).send("Error updating order quantity");
  }
});

// Start the server
const PORT = 3001;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

AddOrder.js

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

function AddOrder() {
  const [orderData, setOrderData] = useState({
    OrderID: '',
    CustomerName: '',
```

```
  Product: '',
  Quantity: '',
  OrderDate: ''
});

const handleChange = (e) => {
  const { name, value } = e.target;
  setOrderData({
    ...orderData,
    [name]: value,
  });
};

const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    await axios.post('http://localhost:3001/api/orders/add', orderData);
    alert('Order added successfully');
  } catch (error) {
    alert('Error adding order');
  }
};

return (
  <form onSubmit={handleSubmit}>
    <input
      type="text"
      name="OrderID"
      placeholder="Order ID"
      value={orderData.OrderID}
      onChange={handleChange}
      required
    />

    <input
      type="text"
      name="CustomerName"
```

```
        placeholder="Customer Name"
        value={orderData.CustomerName}
        onChange={handleChange}
        required
      />

      <input
        type="text"
        name="Product"
        placeholder="Product"
        value={orderData.Product}
        onChange={handleChange}
        required
      />

      <input
        type="number"
        name="Quantity"
        placeholder="Quantity"
        value={orderData.Quantity}
        onChange={handleChange}
        required
      />

      <input
        type="date"
        name="OrderDate"
        value={orderData.OrderDate}
        onChange={handleChange}
        required
      />

      <button type="submit">Add Order</button>
    </form>
  );
}
```

```
export default AddOrder;
```

SearchOrder.js

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

function SearchOrder() {
  const [orderID, setOrderID] = useState('');
  const [order, setOrder] = useState(null);
  const [error, setError] = useState('');

  const handleSearch = async () => {
    try {
      const response = await
axios.get(`http://localhost:3001/api/orders/search/${orderID}`);
      setOrder(response.data);
      setError('');
    } catch (error) {
      setOrder(null);
      setError('Order not found');
    }
  };

  return (
    <div>
      <input
        type="text"
        placeholder="Enter Order ID"
        value={orderID}
        onChange={(e) => setOrderID(e.target.value)}
      />
      <button onClick={handleSearch}>Search Order</button>

      {order && (
        <div>
          <p><strong>Customer:</strong> {order.CustomerName}</p>
```

```
          <p><strong>Product:</strong> {order.Product}</p>
          <p><strong>Quantity:</strong> {order.Quantity}</p>
          <p><strong>Order Date:</strong> {new
Date(order.OrderDate).toLocaleDateString()}</p>
        </div>
      )}
      {error && <p>{error}</p>}
    </div>
  );
}

export default SearchOrder;
```

UpdateOrder.js

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

function UpdateOrder() {
  const [orderID, setOrderID] = useState('');
  const [quantity, setQuantity] = useState('');

  const handleUpdate = async (e) => {
    e.preventDefault();
    try {
      await axios.put(`http://localhost:3001/api/orders/update/${orderID}`, {
Quantity: quantity });
      alert('Order updated successfully');
    } catch (error) {
      alert('Error updating order');
    }
  };

  return (
    <form onSubmit={handleUpdate}>
      <input
        type="text"
```

```
        placeholder="Enter Order ID"
        value={orderID}
        onChange={(e) => setOrderID(e.target.value)}
        required
      />

      <input
        type="number"
        placeholder="Enter New Quantity"
        value={quantity}
        onChange={(e) => setQuantity(e.target.value)}
        required
      />

      <button type="submit">Update Quantity</button>
    </form>
  );
}

export default UpdateOrder;
```

DeleteOrder.js

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

function DeleteOrder() {
  const [orderID, setOrderID] = useState('');

  const handleDelete = async () => {
    try {
      await axios.delete(`http://localhost:3001/api/orders/delete/${orderID}`);
      alert('Order deleted successfully');
    } catch (error) {
      alert('Error deleting order');
    }
  };
```

```
  return (
    <div>
      <input
        type="text"
        placeholder="Enter Order ID"
        value={orderID}
        onChange={(e) => setOrderID(e.target.value)}
      />
      <button onClick={handleDelete}>Delete Order</button>
    </div>
  );
}

export default DeleteOrder;
```

Question 6

JKL Services wants to build a Customer Support Ticketing System that will allow customers to submit support requests, track them, and update their status. As the full-stack developer, you are tasked with developing the following features:

1. MongoDB Database:
•Create a database named Tickets with fields TicketID (unique), CustomerName, IssueDescription, Status (open/closed), and CreatedDate.

2. React Pages:
- Submit Ticket: Design a form with fields for CustomerName, IssueDescription, and Status (open by default). On successful submission, store the ticket details in the MongoDB database.
- View Ticket by TicketID: Create a form where users can input TicketID to search for a ticket. Display the ticket details in a read-only format if found.
- Update Ticket Status: Design a form to update the Status of a ticket by entering its TicketID.

3. Routing and State Management:

•Use React Router to navigate between the “Submit Ticket”, “View Ticket”, and “Update Ticket Status” pages.
•Manage form states with React Hooks (useState, useEffect) to handle user input, form validation, and data fetching.

Specifications:

● Integrate React Hook Form or a similar form library for managing form data and validation.
● UseReact Hooks for managing state and data fetching.
● Implement navigation using React Router and manage application state using React Hooks.

Solution :

App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Link, Routes } from 'react-router-dom';
import SubmitTicket from './pages/SubmitTicket.js';
import ViewTicket from './pages/ViewTicket';
import UpdateTicket from './pages/UpdateTicket';

function App() {
  return (
    <Router>
      <div>
        <h1>Customer Support Ticketing System</h1>
        <nav>
          <ul>
            <li><Link to="/submit">Submit Ticket</Link></li>
            <li><Link to="/view">View Ticket by ID</Link></li>
            <li><Link to="/update">Update Ticket Status</Link></li>
          </ul>
        </nav>
        <Routes>
```

```
          <Route path="/submit" element={<SubmitTicket />} />
          <Route path="/view" element={<ViewTicket />} />
          <Route path="/update" element={<UpdateTicket />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bodyParser = require('body-parser');

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/Tickets')
//mongodb://localhost:27017/<any name>
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.log(err));

// Create Ticket Schema and Model
const ticketSchema = new mongoose.Schema({
  TicketID: { type: String, unique: true },
  CustomerName: String,
  IssueDescription: String,
  Status: { type: String, default: 'open' },
  CreatedDate: { type: Date, default: Date.now }
});

const Ticket = mongoose.model('Ticket', ticketSchema);

const app = express();
```

```
//Middleware
app.use(cors());
app.use(bodyParser.json());

// Submit Ticket
app.post('/api/tickets/submit', async (req, res) => {
  const ticketData = req.body;
  try {
    const ticket = new Ticket(ticketData);
    await ticket.save();
    res.status(201).send("Ticket submitted successfully");
  } catch (error) {
    res.status(400).send("Error submitting ticket");
  }
});

// View Ticket by TicketID
app.get('/api/tickets/view/:TicketID', async (req, res) => {
  try {
    const ticket = await Ticket.findOne({ TicketID: req.params.TicketID });
    if (ticket) {
      res.status(200).json(ticket);
    } else {
      res.status(404).send("Ticket not found");
    }
  } catch (error) {
    res.status(400).send("Error retrieving ticket");
  }
});

// Update Ticket Status by TicketID
app.put('/api/tickets/update/:TicketID', async (req, res) => {
  try {
    const { Status } = req.body;
    const ticket = await Ticket.findOneAndUpdate(
      { TicketID: req.params.TicketID },
      { Status },
```

```
      { new: true }
    );
    if (ticket) {
      res.status(200).send("Ticket status updated successfully");
    } else {
      res.status(404).send("Ticket not found");
    }
  } catch (error) {
    res.status(400).send("Error updating ticket status");
  }
});

// Start the server
const PORT = 3001;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

SubmitTicket.js

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

function SubmitTicket() {
  const [ticketData, setTicketData] = useState({
    TicketID: '',
    CustomerName: '',
    IssueDescription: '',
    Status: 'open' // Default status is 'open'
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setTicketData({
      ...ticketData,
      [name]: value,
    });
  };
```

```
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    await axios.post('http://localhost:3001/api/tickets/submit', ticketData);
    alert('Ticket submitted successfully');
  } catch (error) {
    alert('Error submitting ticket');
  }
};

return (
  <form onSubmit={handleSubmit}>
    <input
      type="text"
      name="TicketID"
      placeholder="Ticket ID"
      value={ticketData.TicketID}
      onChange={handleChange}
      required
    />

    <input
      type="text"
      name="CustomerName"
      placeholder="Customer Name"
      value={ticketData.CustomerName}
      onChange={handleChange}
      required
    />

    <textarea
      name="IssueDescription"
      placeholder="Issue Description"
      value={ticketData.IssueDescription}
      onChange={handleChange}
      required
    ></textarea>
```

```
      <button type="submit">Submit Ticket</button>
    </form>
  );
}

export default SubmitTicket;
```

UpdateTicket.js

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

function UpdateTicket() {
  const [ticketID, setTicketID] = useState('');
  const [status, setStatus] = useState('');

  const handleUpdate = async (e) => {
    e.preventDefault();
    try {
      await axios.put(`http://localhost:3001/api/tickets/update/${ticketID}`, { Status:
status });
      alert('Ticket status updated successfully');
    } catch (error) {
      alert('Error updating ticket status');
    }
  };

  return (
    <form onSubmit={handleUpdate}>
      <input
        type="text"
        placeholder="Enter Ticket ID"
        value={ticketID}
        onChange={(e) => setTicketID(e.target.value)}
        required
      />
```

```
      <select
        value={status}
        onChange={(e) => setStatus(e.target.value)}
        required
      >
        <option value="">Select Status</option>
        <option value="open">Open</option>
        <option value="closed">Closed</option>
      </select>

      <button type="submit">Update Status</button>
    </form>
  );
}

export default UpdateTicket;
```

ViewTicket.js

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

function ViewTicket() {
  const [ticketID, setTicketID] = useState('');
  const [ticket, setTicket] = useState(null);
  const [error, setError] = useState('');

  const handleSearch = async () => {
    try {
      const response = await
axios.get(`http://localhost:3001/api/tickets/view/${ticketID}`);
      setTicket(response.data);
      setError('');
    } catch (error) {
      setTicket(null);
      setError('Ticket not found');
```

```
    }
  };

  return (
    <div>
      <input
        type="text"
        placeholder="Enter Ticket ID"
        value={ticketID}
        onChange={(e) => setTicketID(e.target.value)}
      />
      <button onClick={handleSearch}>View Ticket</button>

      {ticket && (
        <div>
          <p><strong>Customer Name:</strong> {ticket.CustomerName}</p>
          <p><strong>Issue Description:</strong> {ticket.IssueDescription}</p>
          <p><strong>Status:</strong> {ticket.Status}</p>
          <p><strong>Created Date:</strong> {new
Date(ticket.CreatedDate).toLocaleDateString()}</p>
        </div>
      )}
      {error && <p>{error}</p>}
    </div>
  );
}

export default ViewTicket;
```

Extra Question

App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes, Link } from 'react-router-dom';
import { Container, Navbar, Nav } from 'react-bootstrap';
import AddPatient from './pages/AddPatient';
import SearchPatient from './pages/SearchPatient';
import DeletePatient from './pages/DeletePatient';

function App() {
  return (
    <Router>
      <Navbar bg="light" expand="lg">
        <Container>
          <Navbar.Brand href="/">Patient Record System</Navbar.Brand>
          <Nav className="me-auto">
            <Nav.Link as={Link} to="/add">Add Patient</Nav.Link>
            <Nav.Link as={Link} to="/search">Search Patient</Nav.Link>
            <Nav.Link as={Link} to="/delete">Delete Patient</Nav.Link>
          </Nav>
        </Container>
      </Navbar>
      <Container>
        <Routes>
          <Route path="/add" element={<AddPatient />} />
          <Route path="/search" element={<SearchPatient />} />
          <Route path="/delete" element={<DeletePatient />} />
        </Routes>
      </Container>
    </Router>
  );
}

export default App;
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bodyParser = require('body-parser');

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/Patients')
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.log(err));

// Create Patient Schema and Model
const patientSchema = new mongoose.Schema({
  PatientID: { type: String, unique: true },
  Name: String,
  Diagnosis: String,
  Doctor: String,
  AdmissionDate: Date
});

const Patient = mongoose.model('Patient', patientSchema);

const app = express();
app.use(cors());
app.use(bodyParser.json());

// Add Patient
app.post('/api/patients/add', async (req, res) => {
  const patientData = req.body;
  try {
    const patient = new Patient(patientData);
    await patient.save();
    res.status(201).send("Patient added successfully");
  } catch (error) {
    res.status(400).send("Error adding patient");
  }
```

```
});

// Search Patient by PatientID
app.get('/api/patients/search/:PatientID', async (req, res) => {
  try {
    const patient = await Patient.findOne({ PatientID: req.params.PatientID });
    if (patient) {
      res.status(200).json(patient);
    } else {
      res.status(404).send("Patient not found");
    }
  } catch (error) {
    res.status(400).send("Error searching for patient");
  }
});

// Delete Patient by PatientID
app.delete('/api/patients/delete/:PatientID', async (req, res) => {
  try {
    const patient = await Patient.findOneAndDelete({ PatientID:
req.params.PatientID });
    if (patient) {
      res.status(200).send("Patient record deleted successfully");
    } else {
      res.status(404).send("Patient not found");
    }
  } catch (error) {
    res.status(400).send("Error deleting patient record");
  }
});

// Start the server
const PORT = 3001;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

AddPatient.js

```
import React, { useState } from 'react';
import { Form, Button, Alert } from 'react-bootstrap';
import axios from 'axios';

function AddPatient() {
  const [patientData, setPatientData] = useState({
    PatientID: '',
    Name: '',
    Diagnosis: '',
    Doctor: '',
    AdmissionDate: ''
  });
  const [message, setMessage] = useState('');

  const handleChange = (e) => {
    const { name, value } = e.target;
    setPatientData({ ...patientData, [name]: value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post('http://localhost:3001/api/patients/add', patientData);
      setMessage('Patient added successfully');
    } catch (error) {
      setMessage('Error adding patient');
    }
  };

  return (
    <div>
      {message && <Alert variant="success">{message}</Alert>}
      <Form onSubmit={handleSubmit}>
        <Form.Group>
          <Form.Label>Patient ID</Form.Label>
          <Form.Control type="text" name="PatientID" value={patientData.PatientID}
onChange={handleChange} required />
```

```
        </Form.Group>

        <Form.Group>
          <Form.Label>Name</Form.Label>
          <Form.Control type="text" name="Name" value={patientData.Name}
onChange={handleChange} required />
        </Form.Group>

        <Form.Group>
          <Form.Label>Diagnosis</Form.Label>
          <Form.Control type="text" name="Diagnosis"
value={patientData.Diagnosis} onChange={handleChange} required />
        </Form.Group>

        <Form.Group>
          <Form.Label>Doctor</Form.Label>
          <Form.Control type="text" name="Doctor" value={patientData.Doctor}
onChange={handleChange} required />
        </Form.Group>

        <Form.Group>
          <Form.Label>Admission Date</Form.Label>
          <Form.Control type="date" name="AdmissionDate"
value={patientData.AdmissionDate} onChange={handleChange} required />
        </Form.Group>

        <Button variant="primary" type="submit">Add Patient</Button>
      </Form>
    </div>
  );
}

export default AddPatient;
```

SearchPatient.js

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

```
import { Form, Button, Alert } from 'react-bootstrap';
import axios from 'axios';

function SearchPatient() {
  const [patientID, setPatientID] = useState('');
  const [patient, setPatient] = useState(null);
  const [message, setMessage] = useState('');

  const handleSearch = async () => {
    try {
      const response = await
axios.get(`http://localhost:3001/api/patients/search/${patientID}`);
      setPatient(response.data);
      setMessage('');
    } catch (error) {
      setMessage('Patient not found');
      setPatient(null);
    }
  };

  return (
    <div>
      {message && <Alert variant="danger">{message}</Alert>}
      <Form>
        <Form.Group>
          <Form.Label>Patient ID</Form.Label>
          <Form.Control type="text" value={patientID} onChange={(e) =>
setPatientID(e.target.value)} />
        </Form.Group>
        <Button variant="primary" onClick={handleSearch}>Search</Button>
      </Form>
      {patient && (
        <div>
          <h3>Patient Details</h3>
          <p><strong>Name:</strong> {patient.Name}</p>
          <p><strong>Diagnosis:</strong> {patient.Diagnosis}</p>
          <p><strong>Doctor:</strong> {patient.Doctor}</p>
```

```
          <p><strong>Admission Date:</strong> {new
Date(patient.AdmissionDate).toLocaleDateString()}</p>
        </div>
      )}
    </div>
  );
}

export default SearchPatient;
```

DeletePatient.js

```
import React, { useState } from 'react';
import { Form, Button, Alert } from 'react-bootstrap';
import axios from 'axios';

function DeletePatient() {
  const [patientID, setPatientID] = useState('');
  const [message, setMessage] = useState('');

  const handleDelete = async () => {
    try {
      await axios.delete(`http://localhost:3001/api/patients/delete/${patientID}`);
      setMessage('Patient record deleted successfully');
    } catch (error) {
      setMessage('Error deleting patient');
    }
  };

  return (
    <div>
      {message && <Alert variant={message.includes('successfully') ? 'success' :
'danger'}>{message}</Alert>}
      <Form>
        <Form.Group>
          <Form.Label>Patient ID</Form.Label>
```

```
          <Form.Control type="text" value={patientID} onChange={(e) =>
setPatientID(e.target.value)} />
        </Form.Group>
        <Button variant="danger" onClick={handleDelete}>Delete</Button>
      </Form>
    </div>
  );
}

export default DeletePatient;
```