Name: Ranjitha

Employee id:12103

Q2) Bug Management System

Required Properties are: Bug Id, Bug Name, Bug Description, Bug identifiedby, BugresolvedBy.

Note:

- Define a web application for the given scenario using React as a front end.
- Configure JSON server and consider JSON file as your database.
- Perform all the CRUD operations using React HttpClient Service.
- Perform the validations for all the fields.
- Application must be the Responsive Page (Including Mobile View) and Single Page Application (SPA).
- UI must be a Rich UI and look & Feel.
- Maintain the Code Quality.

Source Code

Index.js:

Bugs.js:

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import BugsUpdate from "./BugsUpdate";
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.bundle.min";
const Bugs = () \Rightarrow \{
 const [Bugs, setBugs] = useState([]);
 const [newBug, setNewBug] = useState({
  Bugname: "",
  position: "",
  phone: "",
 });
 const [updatedBug, setUpdatedBug] = useState(null);
 useEffect(() => {
  fetchBugs();
 }, []);
 const fetchBugs = () => {
   .get("http://localhost:4000/employee")
   .then((response) => setBugs(response.data))
   .catch((error) => console.error("Error fetching data:", error));
 };
 const addBug = () => {
  axios
   .post("http://localhost:4000/employee", Bugs)
   .then(() => {
    fetchBugs();
     setNewBug({ BugName: "", position: "", phone: "" });
   .catch((error) => console.error("Error adding bugs:", error));
 };
 const updateBug = (id, updatedData) => {
  axios
   .put('http://localhost:4000/employee/${id}', updatedData)
   .then(() => fetchBugs())
   .catch((error) => console.error("Error updating bugs:", error));
 };
```

```
const deleteBug = (id) \Rightarrow \{
 axios
  .delete('http://localhost:4000/employee/${id}')
  .then(() => fetchBugs())
  .catch((error) => console.error("Error deleting bugs:", error));
};
const handleUpdateClick = (Bug) => {
 setUpdatedBug(Bug);
};
const handleUpdateSubmit = (updatedBug) => {
 updatedBug(updatedBug.id, updatedBug);
 setUpdatedBug(null);
};
return (
 <div className="container mt-4">
  <div className="container d-flex flex-column justify-content-center">
   <h2>Bug Management System</h2>
   <div className="mb-3 justify-content-center">
    <label className="col g-8 "> Bug Name:</label>
    <input
    className="form-control"
     type="text"
     value={newBug.BugName}
     onChange=\{(e) =>
       setNewBug({ ...newBug, BugName: e.target.value })
    />
   </div>
   <div className="mb-3">
    <label>Position:</label>
    <input
    className="form-control"
     type="text"
     value={newBug.position}
     onChange=\{(e) =>
       setNewBug({ ...newBug, position: e.target.value })
    />
   </div>
```

```
<div className="mb-3">
 <label>Phone:</label>
 <input
 className="form-control"
  type="text"
  value={newBug.phone}
  onChange=\{(e) =>
   setNewBug({ ...newBug, phone: e.target.value })
 />
</div>
<button className="btn btn-primary mr-2" onClick={addBug}>
 Add Bugs
</button>
</div>
<thead>
 <th>ID</th>
  Name
  Position
  phone
  Actions
 </thead>
\{Bugs.map((Bug) => (
  {Bug.id}
   {Bug.name}
   {Bug.position}
   {Bug.phone}
   <button
     className="btn btn-warning"
     onClick={() => handleUpdateClick(Bug)}
    >
     Update
    </button>
    <but
     className="btn btn-danger ms-3"
     onClick={() => deleteBug(Bug.id)}
```

```
>
           Delete
          </button>
        ))}
     {updatedBug && (
     <BugsUpdate Bug={updateBug} onUpdate={handleUpdateSubmit} />
   )}
  </div>
);
};
export default Bugs;
BugsUpdate.js:
import React, { useState } from 'react'
const BugUpdate = ({ Bug, onUpdate }) => {
 const [updatedBug, setUpdatedBug] = useState(Bug)
 const handleInputChange = (e) => {
  const { name, value } = e.target
  setUpdatedBug((prevState) => ({ ...prevState, [name]: value }))
 }
 const handleUpdate = () => {
  onUpdate(updatedBug)
 return (
  <>
    <input
    type="text"
    name="name"
    value={updatedBug.name}
    onChange={handleInputChange}
   />
    <input
     type="text"
     name="position"
     value={updatedBug.position}
```

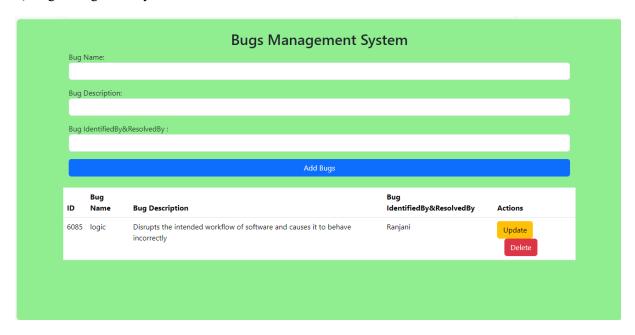
body{

}

background-color:lightgreen;

Output:

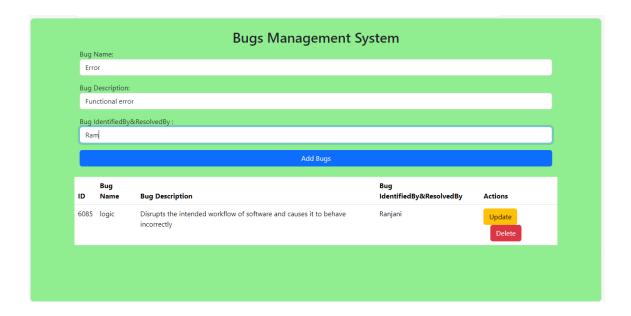
1) Bug management system



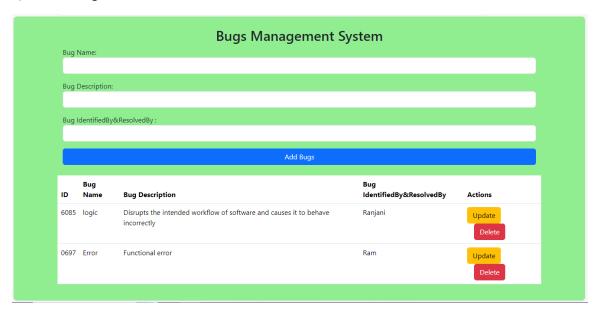
2)Server page



3)Adding Datas

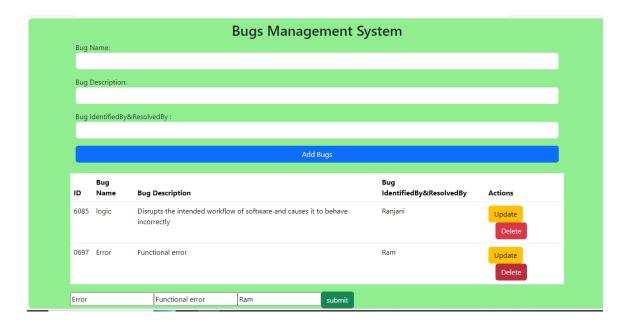


4)After adding the Data

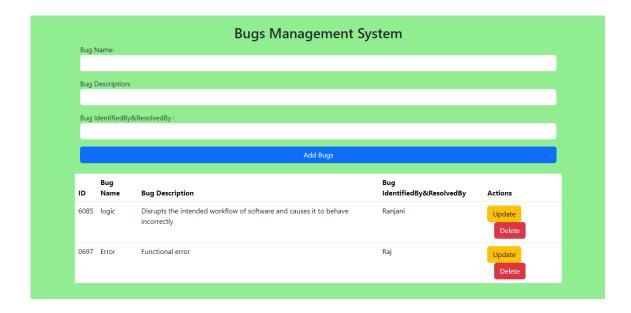


5)Server Data

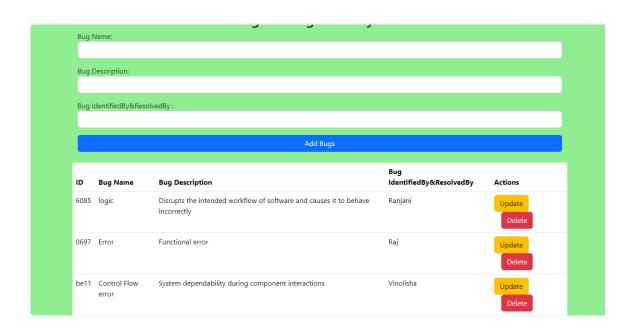
6)Updation



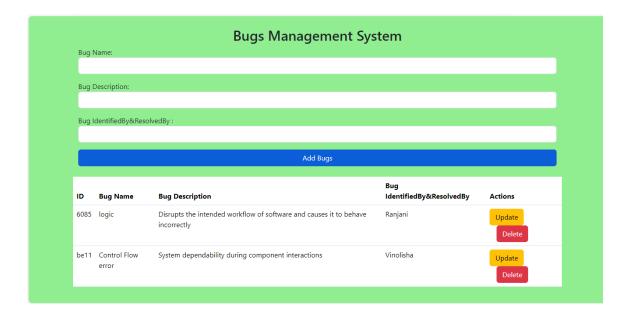
7)After updation



8)Before delete the data:



9)After delete the Data



10)Server page: