|  |  |  |
| --- | --- | --- |
|  | **BAHRIA UNIVERSITY, (Karachi Campus)**  *Department of Software Engineering*  **Final Lab Exam**  **Semester Fall 2023** |  |

**Course Title:**  Web Engineering **Course Code**: CSL-221

**Course Instructor:** Sir Adnan ur Rehman **Class**: BSE-5(A)

**Lab Instructor:** Engr. Ayesha Khan **Name: \_Muhammad Shoaib Akhter Qadri \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Max. Marks: 6** Marks **CLO :** CLO-3 **\_\_\_\_\_\_\_\_\_**

**Time:**  2 hours **Date:** 02-01-2024

**Note:**

* Plagiarism is not allowed, if found you will get zero marks.
* Try to submit the task on LMS in given Time.
* Include your name or enrollment no on footer.
* Your File name should be in the given format:
  + [Class Section] [Complete Name] DSAFINAL
  + **i.e., BSE3A Usman Ali DSAFINAL**

**QUESTION 01: (6 Marks)**

You are tasked with building a Task Management System for a small company. The system should allow employees to manage their tasks efficiently. Each task has a title, description, status (e.g., 'To Do', 'In Progress', 'Completed'), and a deadline. The system should support the basic CRUD operations for tasks.

Requirements:

Task Listing:

Display a list of tasks on the homepage, showing their titles, statuses, and deadlines.

**Include a feature to filter tasks based on their status.**

Task Details:

Clicking on a task in the list should lead to a detailed view showing the task's title, description, status, and deadline.

Task Creation:

Implement a form for creating new tasks.

Include fields for the task title, description, status, and deadline.

Validate the form to ensure that mandatory fields are filled, and the deadline is a future date.

Task Update:

Provide an option to edit existing tasks.

Update the task details, including the title, description, status, and deadline.

Validate the form during the update process.

Task Deletion:

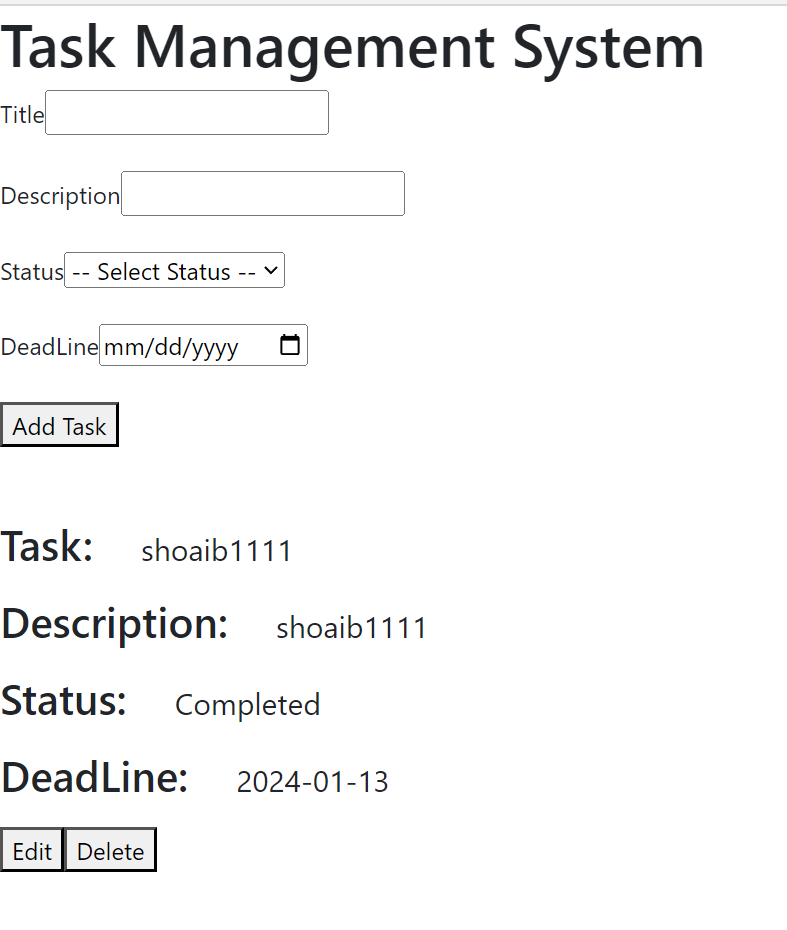
Allow users to delete a task from the system.

Implement a confirmation dialog before permanently removing a task.

Backend API:

Set up a Node.js server using Express to handle CRUD operations for tasks

**Simple UI:**



**Add Data:**

A screenshot of a computer

Description automatically generated

**After Adding Data, then data is show in Frontend:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Edited Data:**

A screenshot of a task management system

Description automatically generated

A screenshot of a computer

Description automatically generated

**Delete One Data:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**dbCollect.js**

const mongoose=require('mongoose');

const dbConnect=async ()=>{

try {

    const response=await mongoose.connect('mongodb+srv://shoaibakhter181422:todo@cluster0.x3d8fsb.mongodb.net/',{useNewUrlParser: true,

    useUnifiedTopology: true,})

    console.log("database is connected");

} catch (error) {

    console.log(error);

}

}

module.exports={dbConnect};

**Server.js:**

const express=require('express');

const mongoose=require('mongoose');

const bodyParser=require('body-parser');

const cors=require('cors');

const {dbConnect}=require('./dbConnect')

const {addTodo, getTodo, updateTodo, deleteTodo}=require('./route/todoRoute')

const app=express();

const PORT=process.env.PORT || 5000;

app.use(cors())

app.use(bodyParser.json());

dbConnect();

app.post('/add-todo',addTodo)

app.get('/get-todo',getTodo);

app.put('/update-todo/:id', updateTodo);

app.delete('/delete-todo/:id',deleteTodo);

app.listen(PORT, () => {

    console.log(`Server is running at http://localhost:${PORT}`);

}

)

**todoModel.js**

const mongoose=require('mongoose');

const todoSchema=mongoose.Schema({

    task:String,

    desc:String,

    status:String,

    deadline:String

})

const Todo=mongoose.model("TodoModel",todoSchema)

module.exports=Todo;

**todoRoute.js**

const Todo = require("../model/todoModel");

const addTodo=async (req,res)=>{

    try {

        const {task,desc,status,deadline}=req.body;

    const newTodo=new Todo({task,desc,status,deadline});

    await newTodo.save();

    res.json({'message':"data is uploaded"});

    } catch (error) {

        console.log(error);

    }

}

const getTodo=async (req,res)=>{

    try {

        const todos=await Todo.find();

        res.json(todos);

    } catch (error) {

        console.log("The data is not fetched "+ error);

    }

}

const updateTodo = async (req, res) => {

    try {

        const { id } = req.params;

        const { task,desc,status,deadline } = req.body;

        const updatedTodo = await Todo.findByIdAndUpdate(id, { task,desc,status,deadline }, { new: true });

        res.json(updatedTodo);

    } catch (error) {

        console.log("Error updating todo: ", error);

        res.status(500).json({ message: "Error updating todo" });

    }

}

const deleteTodo = async (req, res) => {

    try {

        const { id } = req.params;

        await Todo.findByIdAndDelete(id);

        res.json({ message: "Todo deleted successfully" });

    } catch (error) {

        console.log("Error deleting todo: ", error);

        res.status(500).json({ message: "Error deleting todo" });

    }

}

module.exports={addTodo,getTodo,updateTodo,deleteTodo}

**App.js**

import React, { useEffect, useState } from "react";

import axios from "axios";

import { format } from "date-fns";

import "./App.css";

import 'bootstrap/dist/css/bootstrap.min.css';

function App() {

  const [task, setTask] = useState();

  const [desc, setDesc] = useState();

  const [status, setStatus] = useState();

  const [deadline, setDeadline] = useState();

  const [formattedCurrentDate, setFormattedCurrentDate] = useState("");

  const [todos, setTodo] = useState([]);

  const [updatedTask, setUpdatedTask] = useState("");

  const [editMode, setEditMode] = useState(false);

  const [editId, setEditId] = useState(null);

  useEffect(() => {

    const fetchTodos = async () => {

      try {

        const response = await axios.get("http://localhost:5000/get-todo");

        setTodo(response.data);

      } catch (error) {

        console.log("Error Fetching todos: " + error);

      }

    };

    fetchTodos();

    const currentDate = new Date();

    const formattedDate = format(currentDate, "yyyy-MM-dd");

    setFormattedCurrentDate(formattedDate);

  }, [todos]);

  const addTodo = async () => {

    if(task === ""){

      alert("Please provide Title")

    }

    else if(desc === ""){

      alert('Please provide Description')

    }

    else if(status === ""){

      alert('Please provide Status')

    }

    else if(deadline===""){

      alert('Please provide Deadline');

    }

    else{

    try {

      await axios.post("http://localhost:5000/add-todo", { task,desc,status,deadline });

      alert("The task is added");

      setTask("");

      setDesc("");

      setStatus("");

      setDeadline("");

    } catch (error) {

      alert(

        "The task is not added " +

          error +

          " The error message is " +

          error.message

      );

    }

  }

  };

  const enterEditMode = (id, task,desc,status,deadline) => {

    setEditId(id);

    setTask(task);

    setDesc(desc);

    setStatus(status);

    setDeadline(deadline);

    setEditMode(true);

  };

  const cancelEditMode = () => {

    setEditId(null);

    setTask("");

    setDesc("");

    setStatus("");

    setDeadline("");

    setEditMode(false);

  };

  const updateTodo = async (id) => {

    try {

      await axios.put(`http://localhost:5000/update-todo/${id}`, { task,desc,status,deadline });

      alert("Todo updated successfully");

      cancelEditMode();

    } catch (error) {

      alert("Error updating todo: " + error.message);

    }

  };

  const deleteTodo = async (id) => {

    try {

      await axios.delete(`http://localhost:5000/delete-todo/${id}`);

      alert("Todo deleted successfully");

    } catch (error) {

      alert("Error deleting todo: " + error.message);

    }

  };

  return (

    <>

      <h1>Task Management System</h1>

      {editMode ? (

        <>

          <label htmlFor="title">Title</label>

          <input type="text" value={task} onChange={(e) => setTask(e.target.value)}/> <br /><br />

          <label htmlFor="description">Description</label>

        <input type="text" id="description" required value={desc} onChange={(e) => setDesc(e.target.value)} /> <br /><br />

        <label htmlFor="status">Status</label>

        <select name="status" value={status} id="status" required onChange={(e) => setStatus(e.target.value)}>

        <option value="null">-- Select Status --</option>

          <option value="To Do">To Do</option>

          <option value="In Progress">In Progress</option>

          <option value="Completed">Completed</option>

        </select> <br /><br />

        <label htmlFor="deadline">DeadLine</label>

        <input type="date" id="deadline" min={formattedCurrentDate} required value={deadline} onChange={(e) => setDeadline(e.target.value)} /> <br /><br />

          <button onClick={() => updateTodo(editId)}>Update</button>

          <button onClick={cancelEditMode}>Cancel</button>

        </>

      ) : (

        <>

        <label htmlFor="title">Title</label>

        <input type="text" id="title" required value={task} onChange={(e) => setTask(e.target.value)}/> <br /><br />

        <label htmlFor="description">Description</label>

        <input type="text" id="description" required value={desc} onChange={(e) => setDesc(e.target.value)} /> <br /><br />

        <label htmlFor="status">Status</label>

        <select name="status" id="status" required onChange={(e) => setStatus(e.target.value)}>

          <option value="null">-- Select Status --</option>

          <option value="To Do">To Do</option>

          <option value="In Progress">In Progress</option>

          <option value="Completed">Completed</option>

        </select> <br /><br />

        <label htmlFor="deadline">DeadLine</label>

        <input type="date" id="deadline" min={formattedCurrentDate} required value={deadline} onChange={(e) => setDeadline(e.target.value)} /> <br /><br />

          <button onClick={addTodo}>Add Task</button>

        </>

      )}

      {todos.map((todo) => (

        <div key={todo.\_id}>

          <br /><br />

          <h3>Task:</h3>

          <ul>{todo.task}</ul> <br />

          <h3>Description:</h3>

          <ul>{todo.desc}</ul><br />

          <h3>Status:</h3>

          <ul>{todo.status}</ul><br />

          <h3>DeadLine:</h3>

          <ul>{todo.deadline}</ul><br />

          <button onClick={() => enterEditMode(todo.\_id, todo.task,todo.desc,todo.status,todo.deadline)}>

            Edit

          </button>

          <button onClick={() => deleteTodo(todo.\_id)}>Delete</button>

        </div>

      ))}

    </>

  );

}

export default App;