COLLEGE CODE: 9111

COLLEGE :SRM Madurai College for Engineering and Technology

NAME : R.Karishma

DEPARTMENT : Information technology

STUDENT NM-ID :EEC121412BECD8580134A2ADB6F76377

ROLL NO : 23it18

DATE : 29/09/2025

Completed the project named as Phase_4_
TECHNOLOGY PROJECT NAME : Todo list app

SUBMITTED BY,

NAME :Karishma.R

MOBILE NO :7904410709

Additional Features

- Implement task categorization (e.g., Work, Personal)
- Add task priority levels and deadlines
- Enable user authentication with JWT for personalized task lists
- Add task reminders and notifications via email or push
- **Add recurring tasks** functionality to automatically generate tasks at set intervals (daily, weekly, monthly)

UI/UX Improvements

- Design a clean, responsive interface using React and CSS frameworks (e.g., Tailwind, Material-UI)
- Improve task management flow with drag-and-drop for task ordering
- Add dark mode for better accessibility and user preference
- Ensure mobile-friendly layout for smooth usage on all devices
- **Incorporate animated transitions** to make interactions like adding or deleting tasks visually engaging

API Enhancements

- Create secure RESTful endpoints for CRUD operations on tasks and users
- Implement pagination and filtering for large task lists
- Add proper error handling and validation with Express and Mongoose
- Document APIs using tools like Swagger for developer clarity
- **Add real-time updates via WebSockets** to push task changes instantly to connected clients

Performance & Security Checks

- Optimize database queries and indexing in MongoDB for faster retrieval
- Implement rate limiting and input sanitization to prevent attacks (e.g., XSS, SQL injection)
- Use HTTPS and secure headers for API communication
- Regularly update dependencies to patch known vulnerabilities
- **Implement caching mechanisms** like Redis to reduce database load and speed up responses

Testing of Enhancements

- Write unit tests for React components and Express routes using Jest and Supertest
- Perform integration tests to verify database and API interactions
- Conduct end-to-end testing with Cypress or Selenium to simulate user flows
- Automate tests in CI/CD pipeline to catch regressions early
- **Conduct usability testing sessions** to gather real user feedback and improve the user experience

Deployment (Netlify, Vercel, or Cloud Platform)

- Deploy frontend React app on Netlify or Vercel with continuous deployment from GitHub
- Host backend Express API and MongoDB on cloud platforms like Heroku, AWS, or MongoDB Atlas
- Set environment variables securely for production configuration
- Monitor application performance and logs for post-deployment issues
- **Set up automated database backups** and health checks to ensure data safety and reliability

```
<!-- index.html -->
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Todo List</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>

  </body>
</html>
//server.js

const express = require('express')
const mongoose = require('mongoose')
const cors = require('cors')
const TodoModel = require("./models/todoList")

var app = express();
app.use(cors());
app.use(express.json());

// Connect to your MongoDB database (replace with your database URL)
mongoose.connect("mongodb://127.0.0.1/todo");

// Check for database connection errors
mongoose.connection.on("error", (error) => {
    console.error("MongoDB connection error:", error);
});
```

```
// Get saved tasks from the database
app.get("/getTodoList", (req, res) => {
    TodoModel.find({})
        .then((todoList) => res.json(todoList))
        .catch((err) => res.json(err))
});

// Add new task to the database
app.post("/addTodoList", (req, res) => {
    TodoModel.create({
        task: req.body.task,
        status: req.body.status,
        deadline: req.body.deadline,
    })
        .then((todo) => res.json(todo))
        .catch((err) => res.json(err));
});

// Update task fields (including deadline)
app.post("/updateTodoList/:id", (req, res) => {
    const id = req.params.id;
    const updateData = {
        task: req.body.task,
        status: req.body.status,
        deadline: req.body.deadline,
    };
    TodoModel.findByIdAndUpdate(id, updateData)
        .then((todo) => res.json(todo))
        .catch((err) => res.json(err));
```

```
});

// Delete task from the database
app.delete("/deleteTodoList/:id", (req, res) => {
    const id = req.params.id;
    TodoModel.findByIdAndDelete({ _id: id })
        .then((todo) => res.json(todo))
        .catch((err) => res.json(err));
});

app.listen(3001, () => {
    console.log('Server running on 3001');
});
//todoList.js

const mongoose = require('mongoose');

const todoSchema = new mongoose.Schema({
    task: {
        type: String,
        required: true,
    },
    status: {
        type: String,
        required: true,
    },
    deadline: {
        type: Date,
    },
});
```

```
const todoList = mongoose.model("todo", todoSchema);

module.exports = todoList;
//App.js

import React from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import 'bootstrap/dist/css/bootstrap.min.css';
import Todo from './components/Todo';


function App() {
  const headStyle = {
    textAlign: "center",
  }
  return (
    <div>
      <h1 style={headStyle}>Todo List</h1>
      <BrowserRouter>
        <Routes>
          <Route path='/' element={<Todo/>}></Route>
        </Routes>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

```
//todoList.js

const mongoose = require('mongoose');

const todoSchema = new mongoose.Schema({
    task: {
        type: String,
        required: true,
    },
    status: {
        type: String,
        required: true,
    },
    deadline: {
        type: Date,
    },
});


const todoList = mongoose.model("todo", todoSchema);

module.exports = todoList;
import axios from "axios";
import React from "react";
import { useEffect, useState } from "react";

function Todo() {
    const [todoList, setTodoList] = useState([]);
    const [editableId, setEditableId] = useState(null);
    const [editedTask, setEditedTask] = useState("");
```

```
const [editedStatus, setEditedStatus] = useState("");
const [newTask, setNewTask] = useState("");
const [newStatus, setNewStatus] = useState("");
const [newDeadline, setNewDeadline] = useState("");
const [editedDeadline, setEditedDeadline] = useState("");

// Fetch tasks from database
useEffect(() => {
    axios.get('http://127.0.0.1:3001/getTodoList')
        .then(result => {
            setTodoList(result.data)
        })
        .catch(err => console.log(err))
}, [])

// Function to toggle the editable state for a specific row
const toggleEditable = (id) => {
    const rowData = todoList.find((data) => data._id === id);
    if (rowData) {
        setEditableId(id);
        setEditedTask(rowData.task);
        setEditedStatus(rowData.status);
        setEditedDeadline(rowData.deadline || "");
    } else {
        setEditableId(null);
        setEditedTask("");
        setEditedStatus("");
        setEditedDeadline("");
    }
};
```

```
    // Function to add task to the database
    const addTask = (e) => {
        e.preventDefault();
        if (!newTask || !newStatus || !newDeadline) {
            alert("All fields must be filled out.");
            return;
        }

        axios.post('http://127.0.0.1:3001/addTodoList', { task: newTask, status: newStatus, deadline: newDeadline })
            .then(res => {
                console.log(res);
                window.location.reload();
            })
            .catch(err => console.log(err));
    }

    // Function to save edited data to the database
    const saveEditedTask = (id) => {
        const editedData = {
            task: editedTask,
            status: editedStatus,
            deadline: editedDeadline,
        };

        // If the fields are empty
        if (!editedTask || !editedStatus || !editedDeadline) {
            alert("All fields must be filled out.");
```

```
        return;
    }

    // Updating edited data to the database through updateById API
    axios.post('http://127.0.0.1:3001/updateTodoList' + id, editedData)
        .then(result => {
            console.log(result);
            setEditableId(null);
            setEditedTask("");
            setEditedStatus("");
            setEditedDeadline(""); // Clear the edited deadline
            window.location.reload();
        })
        .catch(err => console.log(err));
}


// Delete task from database
const deleteTask = (id) => {
    axios.delete('http://127.0.0.1:3001/deleteTodoList' + id)
        .then(result => {
            console.log(result);
            window.location.reload();
        })
        .catch(err =>
            console.log(err)
        )
}

return (
```

```
        <div className="container mt-5">
            <div className="row">
                <div className="col-md-7">
                    <h2 className="text-center">Todo List</h2>
                    <div className="table-responsive">
                        <table className="table table-bordered">
                            <thead className="table-primary">
                                <tr>
                                    <th>Task</th>
                                    <th>Status</th>
                                    <th>Deadline</th>
                                    <th>Actions</th>
                                </tr>
                            </thead>
                            {Array.isArray(todoList) ? (
                                <tbody>
                                    {todoList.map((data) => (
                                        <tr key={data._id}>
                                            <td>
                                                {editableId === data._id ? (
                                                    <input
                                                        type="text"
                                                        className="form-control"
                                                        value={editedTask}
                                                        onChange={(e) =>
setEditedTask(e.target.value)}
                                                    />
                                                ) : (
                                                    data.task
                                                )}
```

```
                                    </td>
                                    <td>
                                        {editableId === data._id ? (
                                            <input
                                                type="text"
                                                className="form-control"
                                                value={editedStatus}
                                                onChange={(e) =>
setEditedStatus(e.target.value)}
                                            />
                                        ) : (
                                            data.status
                                        )}
                                    </td>
                                    <td>
                                        {editableId === data._id ? (
                                            <input
                                                type="datetime-local"
                                                className="form-control"
                                                value={editedDeadline}
                                                onChange={(e) =>
setEditedDeadline(e.target.value)}
                                            />
                                        ) : (
                                            data.deadline ? new
Date(data.deadline).toLocaleString() : ''
                                        )}
                                    </td>

                                    <td>
                                        {editableId === data._id ? (
```

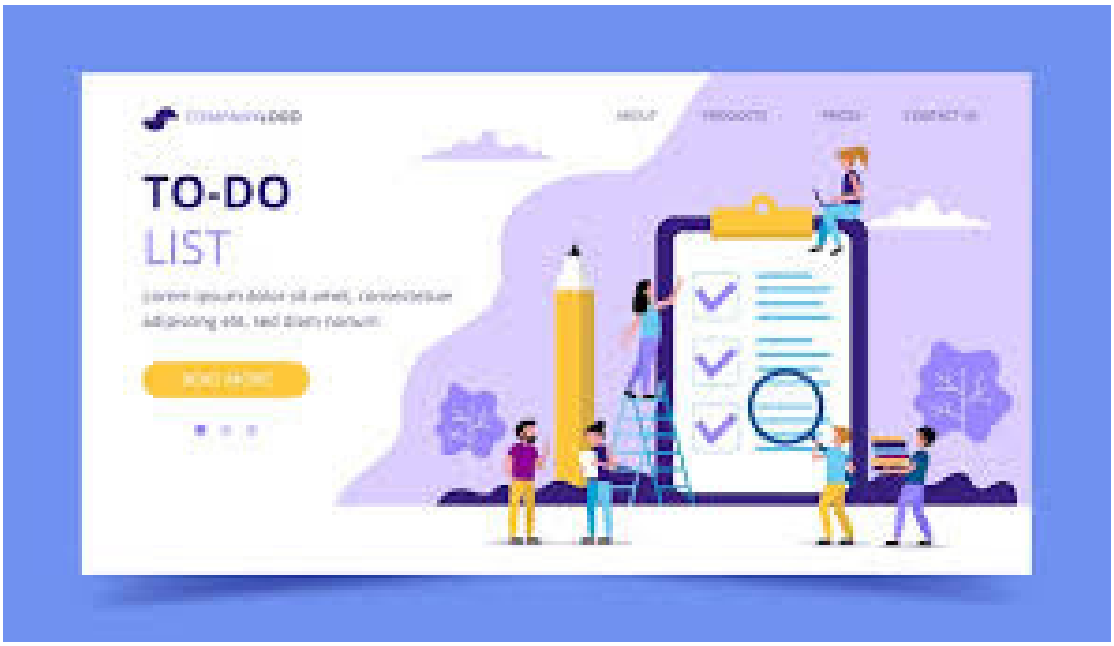
```
                                                    <button className="btn btn-
success btn-sm" onClick={() => saveEditedTask(data._id)}>
                                                        Save
                                                    </button>
                                                ) : (
                                                    <button className="btn btn-
primary btn-sm" onClick={() => toggleEditable(data._id)}>
                                                        Edit
                                                    </button>
                                                )}
                                                <button className="btn btn-
danger btn-sm ml-1" onClick={() => deleteTask(data._id)}>
                                                    Delete
                                                </button>
                                            </td>
                                        </tr>
                                    ))}
                                </tbody>
                            ) : (
                                <tbody>
                                    <tr>
                                        <td colSpan="4">Loading products...</td>
                                    </tr>
                                </tbody>
                            )}


                        </table>
                    </div>
                </div>
                <div className="col-md-5">
```

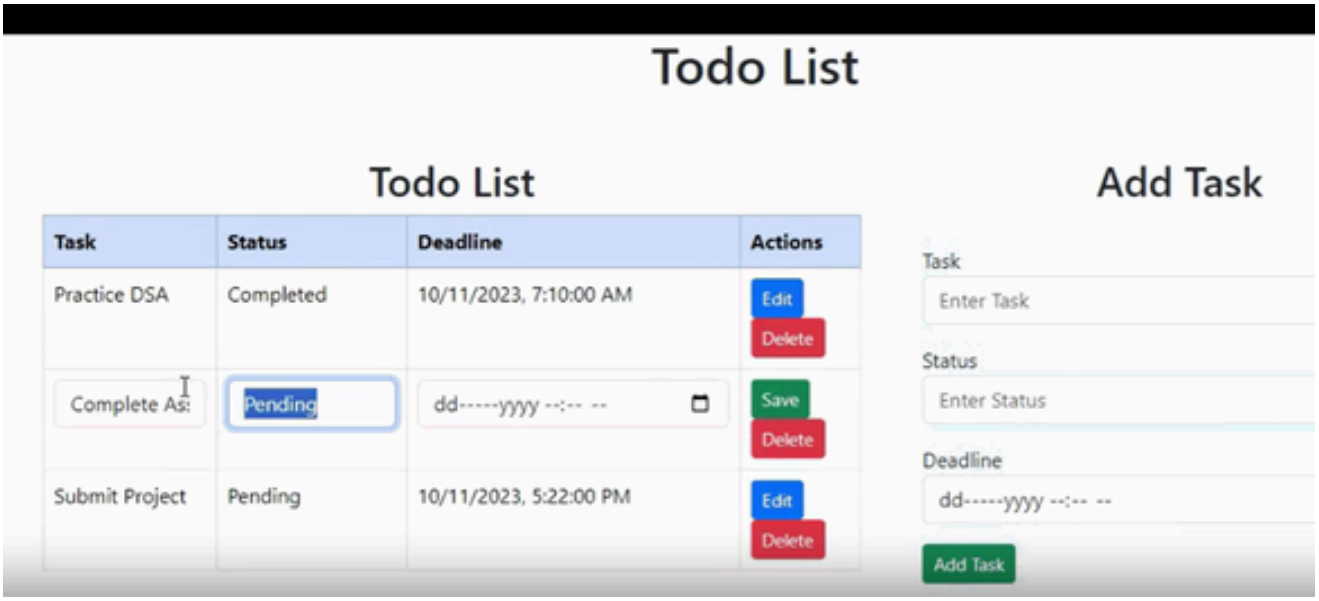
```
                    <h2 className="text-center">Add Task</h2>
                    <form className="bg-light p-4">
                        <div className="mb-3">
                            <label>Task</label>
                            <input
                                className="form-control"
                                type="text"
                                placeholder="Enter Task"
                                onChange={(e) => setNewTask(e.target.value)}
                            />
                        </div>
                        <div className="mb-3">
                            <label>Status</label>
                            <input
                                className="form-control"
                                type="text"
                                placeholder="Enter Status"
                                onChange={(e) => setNewStatus(e.target.value)}
                            />
                        </div>
                        <div className="mb-3">
                            <label>Deadline</label>
                            <input
                                className="form-control"
                                type="datetime-local"
                                onChange={(e) => setNewDeadline(e.target.value)}
                            />
                        </div>
                        <button onClick={addTask} className="btn btn-success
btn-sm">
```

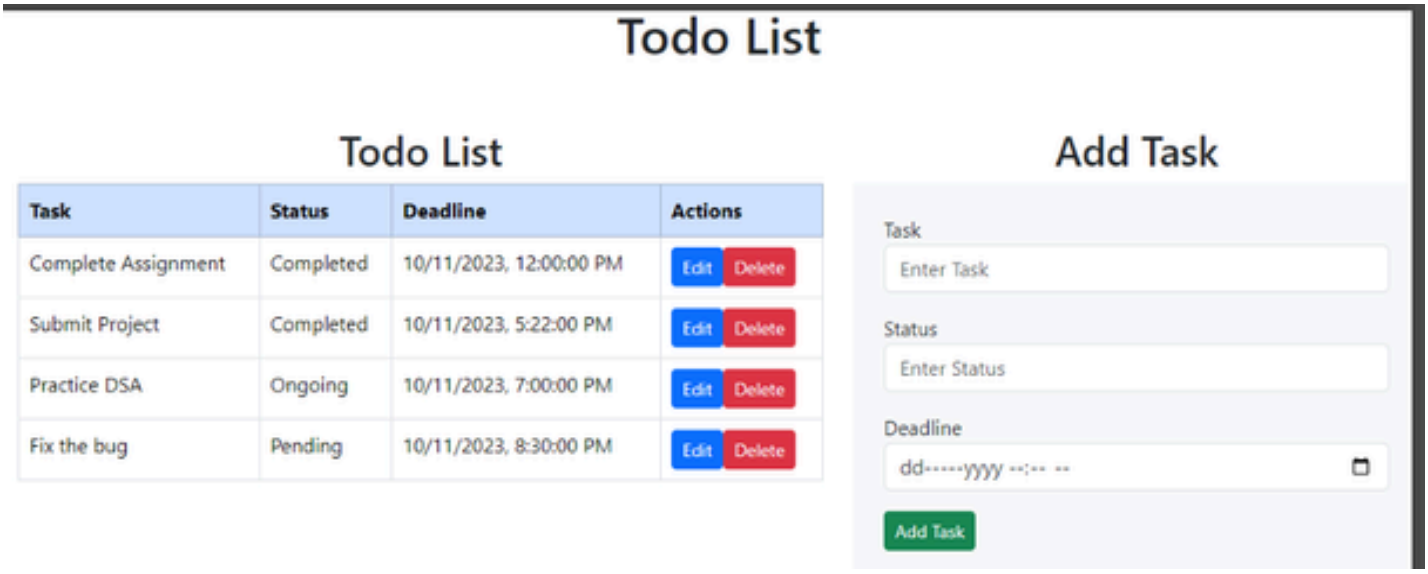
```
                            Add Task
                        </button>
                    </form>
                </div>

            </div>
        </div>
    )
}
export default Todo;
//index.js - Serves as the entry point for your React application

import React from 'react';
import ReactDOM from 'react-dom';
import App from './App'; // Your main application component

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
//server.js

const express = require('express')
const mongoose = require('mongoose')
const cors = require('cors')
const TodoModel = require("./models/todoList")

var app = express();
```

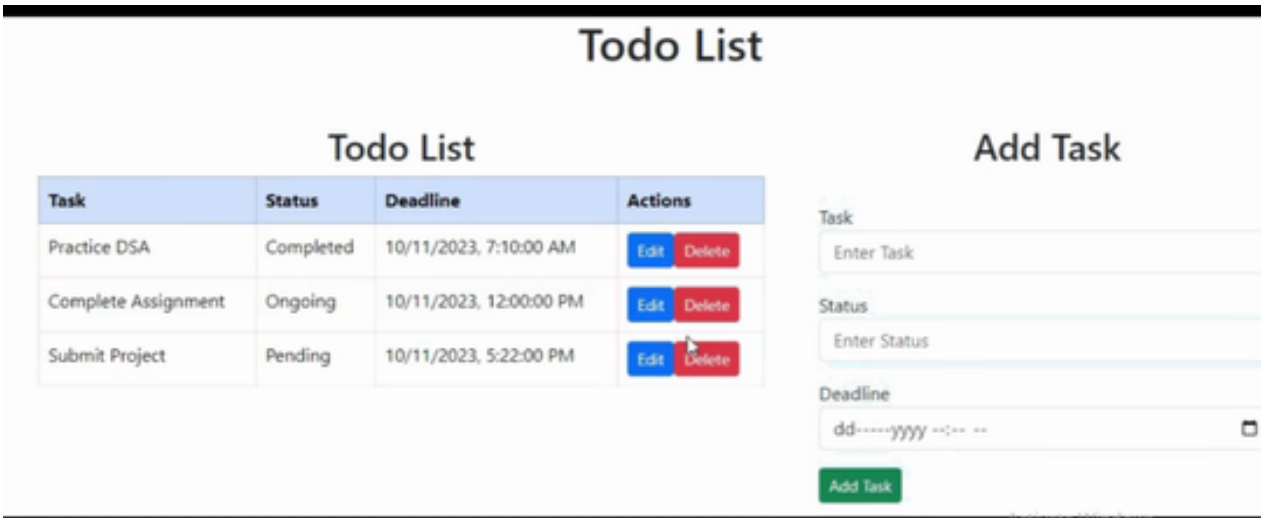
```
app.use(cors());
app.use(express.json());

// Connect to your MongoDB database (replace with your database URL)
mongoose.connect("mongodb://127.0.0.1/todo");

// Check for database connection errors
mongoose.connection.on("error", (error) => {
    console.error("MongoDB connection error:", error);
});

// Get saved tasks from the database
app.get("/getTodoList", (req, res) => {
    TodoModel.find({})
        .then((todoList) => res.json(todoList))
        .catch((err) => res.json(err))
});

// Add new task to the database
app.post("/addTodoList", (req, res) => {
    TodoModel.create({
        task: req.body.task,
        status: req.body.status,
        deadline: req.body.deadline,
    })
        .then((todo) => res.json(todo))
        .catch((err) => res.json(err));
});

// Update task fields (including deadline)
```

```
app.post("/updateTodoList/:id", (req, res) => {
    const id = req.params.id;
    const updateData = {
        task: req.body.task,
        status: req.body.status,
        deadline: req.body.deadline,
    };
    TodoModel.findByIdAndUpdate(id, updateData)
        .then((todo) => res.json(todo))
        .catch((err) => res.json(err));
});

// Delete task from the database
app.delete("/deleteTodoList/:id", (req, res) => {
    const id = req.params.id;
    TodoModel.findByIdAndDelete({ _id: id })
        .then((todo) => res.json(todo))
        .catch((err) => res.json(err));
});

app.listen(3001, () => {
    console.log('Server running on 3001');
});
```


TO-DO
LIST


Todo List
Todo List
Task
Status
Deadline
Actions
Practice DSA
Completed
10/11/2023, 7:10:00 AM
Edit
Delete
Complete As
Pending
dd-----yyyy --:-- --
Save
Delete
Submit Project
Pending
10/11/2023, 5:22:00 PM
Edit
Delete
Add Task
Task
Enter Task
Status
Enter Status
Deadline
dd-----yyyy --:-- --
Add Task


Todo List
Todo List
Task
Status
Deadline
Actions
Complete Assignment
Completed
10/11/2023, 12:00:00 PM
Edit
Delete
Submit Project
Completed
10/11/2023, 5:22:00 PM
Edit
Delete
Practice DSA
Ongoing
10/11/2023, 7:00:00 PM
Edit
Delete
Fix the bug
Pending
10/11/2023, 8:30:00 PM
Edit
Delete
Add Task
Task
Enter Task
Status
Enter Status
Deadline
dd-----yyyy --:-- --
Add Task



LINK : https://github.com/karishma0509-r/project.git