Backend/index.js

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");

const app = express();

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

mongoose.connect("mongodb://127.0.0.1:27017/taskdb")
.then(() => console.log("MongoDB Connected"))
.catch((err) => console.log(err));

const TaskSchema = new mongoose.Schema({
  taskName: { type: String, required: true },
  description: { type: String, required: true },
  status: { type: String, default: "Pending" },
  createdDate: { type: Date, default: Date.now }
});

const TaskModel = mongoose.model("Task", TaskSchema);

app.post("/tasks", async (req, res) => {
  try {
    const newTask = await TaskModel.create({
      taskName: req.body.taskName,
      description: req.body.description
    });

    res.status(201).json(newTask);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

app.get("/tasks", async (req, res) => {
  try {
    const tasks = await TaskModel.find().sort({ createdDate: -1 });
    res.json(tasks);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

app.put("/tasks/:id", async (req, res) => {
  try {
    await TaskModel.findByIdAndUpdate(
      req.params.id,
      { status: req.body.status }
```

```
    );

    res.json({ message: "Task Updated" });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

app.delete("/tasks/:id", async (req, res) => {
  try {
    await TaskModel.findByIdAndDelete(req.params.id);
    res.json({ message: "Task Deleted" });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

app.listen(5000, () => {
  console.log("Server running on port 5000");
});
```

Frontend/App.jsx

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

function App() {

  const [taskName, setTaskName] = useState("");
  const [description, setDescription] = useState("");
  const [tasks, setTasks] = useState([]);
  const [loading, setLoading] = useState(false);

  const fetchTasks = async () => {
    try {
      setLoading(true);
      const res = await fetch("http://localhost:5000/tasks");
      const data = await res.json();
      setTasks(data);
      setLoading(false);
    } catch (error) {
      alert("Error fetching tasks");
      setLoading(false);
    }
  };

  useEffect(() => {
    fetchTasks();
  }, []);

  const addTask = async (e) => {
```

```
  e.preventDefault();

  if (!taskName || !description) return;

  try {
    await fetch("http://localhost:5000/tasks", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        taskName,
        description
      })
    });

    setTaskName("");
    setDescription("");
    fetchTasks();
    alert("Task Saved");
  } catch (error) {
    alert("Error saving task");
  }
};

const completeTask = async (id) => {
  try {
    await fetch(`http://localhost:5000/tasks/${id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ status: "Completed" })
    });

    fetchTasks();
  } catch (error) {
    alert("Error updating task");
  }
};

const deleteTask = async (id) => {

  if (!window.confirm("Delete this task?")) return;

  try {
    await fetch(`http://localhost:5000/tasks/${id}`, {
      method: "DELETE"
    });

    fetchTasks();
  } catch (error) {
    alert("Error deleting task");
  }
};
```

```
return (
  <div style={{ padding: "40px" }}>
    <h2>Task Manager</h2>

    <form onSubmit={addTask}>
      <input
        type="text"
        placeholder="Task Name"
        value={taskName}
        onChange={(e) => setTaskName(e.target.value)}
        required
      />

      <br /><br />

      <input
        type="text"
        placeholder="Description"
        value={description}
        onChange={(e) => setDescription(e.target.value)}
        required
      />

      <br /><br />

      <button type="submit">Add Task</button>
    </form>

    <br />

    {loading && <h3>Loading...</h3>}

    <table border="1" cellPadding="10">

      <thead>
        <tr>
          <th>Task Name</th>
          <th>Description</th>
          <th>Status</th>
          <th>Created Date</th>
          <th>Actions</th>
        </tr>
      </thead>

      <tbody>

        {tasks.map((t) => (
          <tr
            key={t._id}
            style={{
```

```
            textDecoration: t.status === "Completed" ? "line-through" : "none",
            backgroundColor: t.status === "Completed" ? "#d4ffd4" : "white"
          }}
        >

          <td>{t.taskName}</td>
          <td>{t.description}</td>
          <td>{t.status}</td>
          <td>{new Date(t.createdDate).toLocaleString()}</td>

          <td>

            {t.status !== "Completed" && (
              <button onClick={() => completeTask(t._id)}>
                Complete
              </button>
            )}

            <button
              onClick={() => deleteTask(t._id)}
              style={{ marginLeft: "10px" }}
            >
              Delete
            </button>

          </td>

        </tr>
      ))}

    </tbody>

  </table>

 </div>
 );
}

export default App;
```

CHANGES :

```
const TextModel = mongoose.model("Task", TextSchema);

await TaskModel.create({
```

```
const TaskModel = mongoose.model("Task", TaskSchema);
```

```
await TaskModel.create({
```

```
mongoose.connect("mongodb://127.0.0.1:27017/taskdb");
```

```
mongoose.connect("mongodb://127.0.0.1:27017/taskdb")
.then(() => console.log("MongoDB Connected"))
.catch((err) => console.log(err));
```

```
const TextSchema = new mongoose.Schema({
  taskName: String,
  description: String,
  status: String,
  createdDate: Date,
});
```

```
const TaskSchema = new mongoose.Schema({
  taskName: { type: String, required: true },
  description: { type: String, required: true },
  status: { type: String, default: "Pending" },
  createdDate: { type: Date, default: Date.now }
});
```

```
app.post("/tasks", async (req, res) => {
  await TaskModel.create(...)
  res.send("Saved");
});
```

```
try {
   ...
} catch(error) {
   res.status(500).json({ message: error.message });
}
```

```
res.send("Saved");
```

```
res.status(201).json(newTask);
```

```
const tasks = await TextModel.find();
```

```
const tasks = await TaskModel.find().sort({ createdDate: -1 });
```

---

const [loading, setLoading] = useState(false);

---

const res = await fetch(...)

 If backend fails → UI breaks silently

```
try {
  ...
} catch(error) {
  alert("Error fetching tasks");
}
```

---

if (!taskName || !description) return;

---

if (!window.confirm("Delete this task?")) return;

},();

}, []);