**✅ Tech Stack Summary (as per their preference)**

| **Layer** | **Technology** |
| --- | --- |
| Frontend | React.js (with Hooks), Bootstrap/Material UI (optional) |
| Backend | Node.js + Express.js |
| Database | MySQL |
| Hosting | Optional (Vercel for frontend, Render/Heroku for backend) |

**🛠️ Project Structure Overview**

pgsql

CopyEdit

task-manager/

├── backend/

│ ├── controllers/

│ ├── models/

│ ├── routes/

│ ├── config/

│ └── index.js

├── frontend/

│ ├── public/

│ └── src/

│ ├── components/

│ ├── App.js

│ └── index.js

├── README.md

└── .gitignore

**🪛 Step-by-Step Guide**

**🔁 PART 1 – Backend (Node.js + Express + MySQL)**

**✅ 1. Setup**

bash

CopyEdit

mkdir backend

cd backend

npm init -y

npm install express cors mysql2 sequelize dotenv

**✅ 2. Folder Structure**

bash

CopyEdit

backend/

├── config/db.js

├── models/task.model.js

├── controllers/task.controller.js

├── routes/task.routes.js

└── index.js

**✅ 3. DB Config – config/db.js**

js

CopyEdit

const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('task\_db', 'root', 'your\_password', {

host: 'localhost',

dialect: 'mysql',

});

module.exports = sequelize;

**✅ 4. Model – models/task.model.js**

js

CopyEdit

const { DataTypes } = require('sequelize');

const sequelize = require('../config/db');

const Task = sequelize.define('Task', {

title: { type: DataTypes.STRING, allowNull: false },

description: { type: DataTypes.TEXT },

status: { type: DataTypes.STRING, defaultValue: 'Pending' },

}, {

timestamps: true

});

module.exports = Task;

**✅ 5. Controller – controllers/task.controller.js**

js

CopyEdit

const Task = require('../models/task.model');

exports.getTasks = async (req, res) => {

const tasks = await Task.findAll();

res.status(200).json(tasks);

};

exports.createTask = async (req, res) => {

const { title, description } = req.body;

if (!title) return res.status(400).json({ error: "Title required" });

const task = await Task.create({ title, description });

res.status(201).json(task);

};

exports.updateTask = async (req, res) => {

const { id } = req.params;

const task = await Task.findByPk(id);

if (!task) return res.status(404).json({ error: "Task not found" });

await task.update(req.body);

res.status(200).json(task);

};

exports.deleteTask = async (req, res) => {

const { id } = req.params;

const task = await Task.findByPk(id);

if (!task) return res.status(404).json({ error: "Task not found" });

await task.destroy();

res.status(200).json({ message: "Deleted" });

};

**✅ 6. Routes – routes/task.routes.js**

js

CopyEdit

const express = require('express');

const router = express.Router();

const controller = require('../controllers/task.controller');

router.get('/tasks', controller.getTasks);

router.post('/tasks', controller.createTask);

router.put('/tasks/:id', controller.updateTask);

router.delete('/tasks/:id', controller.deleteTask);

module.exports = router;

**✅ 7. Main File – index.js**

js

CopyEdit

const express = require('express');

const cors = require('cors');

const sequelize = require('./config/db');

const taskRoutes = require('./routes/task.routes');

const app = express();

app.use(cors());

app.use(express.json());

app.use('/api', taskRoutes);

sequelize.sync().then(() => {

app.listen(5000, () => console.log('Backend running on port 5000'));

}).catch(err => console.log(err));

**🎨 PART 2 – Frontend (React.js)**

**✅ 1. Setup**

bash

CopyEdit

npx create-react-app frontend

cd frontend

npm install axios bootstrap

Add Bootstrap to index.js:

js

CopyEdit

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

**✅ 2. Basic Structure**

bash

CopyEdit

frontend/src/

├── components/

│ ├── TaskForm.js

│ ├── TaskList.js

├── App.js

└── index.js

**✅ 3. App.js**

jsx

CopyEdit

import React from 'react';

import TaskForm from './components/TaskForm';

import TaskList from './components/TaskList';

function App() {

return (

<div className="container mt-5">

<h2>Task Manager</h2>

<TaskForm />

<TaskList />

</div>

);

}

export default App;

**✅ 4. TaskForm.js**

jsx

CopyEdit

import React, { useState } from 'react';

import axios from 'axios';

function TaskForm() {

const [task, setTask] = useState({ title: '', description: '' });

const handleSubmit = async (e) => {

e.preventDefault();

await axios.post('http://localhost:5000/api/tasks', task);

window.location.reload();

};

return (

<form onSubmit={handleSubmit} className="mb-3">

<input className="form-control mb-2"

placeholder="Title"

value={task.title}

onChange={(e) => setTask({ ...task, title: e.target.value })}

/>

<textarea className="form-control mb-2"

placeholder="Description"

value={task.description}

onChange={(e) => setTask({ ...task, description: e.target.value })}

/>

<button className="btn btn-primary">Add Task</button>

</form>

);

}

export default TaskForm;

**✅ 5. TaskList.js**

jsx

CopyEdit

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

import axios from 'axios';

function TaskList() {

const [tasks, setTasks] = useState([]);

const fetchTasks = async () => {

const res = await axios.get('http://localhost:5000/api/tasks');

setTasks(res.data);

};

const deleteTask = async (id) => {

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

fetchTasks();

};

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

return (

<table className="table">

<thead><tr><th>Title</th><th>Description</th><th>Status</th><th>Action</th></tr></thead>

<tbody>

{tasks.map(task => (

<tr key={task.id}>

<td>{task.title}</td>

<td>{task.description}</td>

<td>{task.status}</td>

<td>

<button className="btn btn-danger btn-sm" onClick={() => deleteTask(task.id)}>Delete</button>

</td>

</tr>

))}

</tbody>

</table>

);

}

export default TaskList;

**🚀 Hosting (Optional Bonus)**

* **Frontend:** Deploy with Vercel
* **Backend:** Use Render or Railway
* **DB:** Use local MySQL or PlanetScale

**📦 GitHub Submission**

1. Push full project (frontend and backend) into a public repo:

arduino

CopyEdit

https://github.com/yourusername/task-manager

1. Add a README.md with:
   * Description
   * Tech stack
   * Setup instructions
   * Screenshots (optional)
2. Share the link via email