**✅ Prerequisites**

1. **Install Node.js**: [https://nodejs.org](https://nodejs.org/)
2. **Install MongoDB** (local or use [MongoDB Atlas](https://www.mongodb.com/cloud/atlas))
3. **Install VS Code**: [https://code.visualstudio.com](https://code.visualstudio.com/)

**🧩 Folder Structure**

crud-app/

├── backend/

└── frontend/

**🖥️ Step 1: Backend Setup (Node.js + Express + MongoDB)**

**1. Create backend folder and init project**

mkdir backend

cd backend

npm init -y

npm install express mongoose cors

**2. Create server.js**

// backend/server.js

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const app = express();

app.use(cors());

app.use(express.json());

// Connect to MongoDB

mongoose.connect('mongodb://localhost:27017/crudDB', {

useNewUrlParser: true,

useUnifiedTopology: true,

});

// Schema

const ItemSchema = new mongoose.Schema({

name: String,

description: String,

});

const Item = mongoose.model('Item', ItemSchema);

// Routes

// Create

app.post('/items', async (req, res) => {

const newItem = new Item(req.body);

await newItem.save();

res.send(newItem);

});

// Read

app.get('/items', async (req, res) => {

const items = await Item.find();

res.send(items);

});

// Update

app.put('/items/:id', async (req, res) => {

const updated = await Item.findByIdAndUpdate(req.params.id, req.body, { new: true });

res.send(updated);

});

// Delete

app.delete('/items/:id', async (req, res) => {

await Item.findByIdAndDelete(req.params.id);

res.send({ message: 'Deleted successfully' });

});

app.listen(5000, () => {

console.log('Server is running on port 5000');

});

**3. Run the backend**

node server.js

**🌐 Step 2: Frontend Setup (React)**

**1. Create frontend using React**

cd ..

npx create-react-app frontend

cd frontend

npm install axios

**2. Update App.js**

// frontend/src/App.js

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

import axios from 'axios';

import './App.css'; // Import the CSS

function App() {

  const [items, setItems] = useState([]);

  const [form, setForm] = useState({ name: '', description: '' });

  const [editingId, setEditingId] = useState(null);

  const fetchItems = async () => {

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

    setItems(res.data);

  };

  useEffect(() => {

    fetchItems();

  }, []);

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (editingId) {

      await axios.put(`http://localhost:5000/items/${editingId}`, form);

      setEditingId(null);

    } else {

      await axios.post('http://localhost:5000/items', form);

    }

    setForm({ name: '', description: '' });

    fetchItems();

  };

  const handleEdit = (item) => {

    setForm({ name: item.name, description: item.description });

    setEditingId(item.\_id);

  };

  const handleDelete = async (id) => {

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

    fetchItems();

  };

  return (

    <div className="container">

      <h1>📝 CRUD App</h1>

      <form onSubmit={handleSubmit} className="form">

        <input

          type="text"

          placeholder="Enter name"

          value={form.name}

          onChange={(e) => setForm({ ...form, name: e.target.value })}

          required

        />

        <input

          type="text"

          placeholder="Enter description"

          value={form.description}

          onChange={(e) => setForm({ ...form, description: e.target.value })}

          required

        />

        <button type="submit">{editingId ? 'Update' : 'Add'}</button>

      </form>

      <div className="item-list">

        {items.map((item) => (

          <div key={item.\_id} className="item-card">

            <h3>{item.name}</h3>

            <p>{item.description}</p>

            <div className="btn-group">

              <button className="edit" onClick={() => handleEdit(item)}>Edit</button>

              <button className="delete" onClick={() => handleDelete(item.\_id)}>Delete</button>

            </div>

          </div>

        ))}

      </div>

    </div>

  );

}

export default App;

**3. Update App.css (src/App.css) :**

/\* App.css \*/

body {

  margin: 0;

  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

  background: linear-gradient(135deg, #e3f2fd, #fffde7);

}

.container {

  max-width: 800px;

  margin: auto;

  padding: 20px;

  text-align: center;

}

h1 {

  color: #1976d2;

  margin-bottom: 20px;

}

.form {

  display: flex;

  flex-direction: column;

  gap: 12px;

  margin-bottom: 30px;

}

input {

  padding: 10px;

  border: 2px solid #90caf9;

  border-radius: 8px;

  font-size: 16px;

}

button {

  padding: 10px 16px;

  font-size: 16px;

  background-color: #64b5f6;

  color: white;

  border: none;

  border-radius: 8px;

  cursor: pointer;

  transition: background 0.3s;

}

button:hover {

  background-color: #42a5f5;

}

.item-list {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

  gap: 16px;

}

.item-card {

  background: white;

  border-radius: 12px;

  padding: 20px;

  box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);

}

.item-card h3 {

  margin: 0 0 10px;

  color: #424242;

}

.item-card p {

  margin: 0 0 12px;

  color: #616161;

}

.btn-group {

  display: flex;

  justify-content: space-between;

}

.edit {

  background-color: #4caf50;

}

.delete {

  background-color: #e53935;

}

.edit:hover {

  background-color: #388e3c;

}

.delete:hover {

  background-color: #c62828;

}

@media (max-width: 600px) {

  .form {

    flex-direction: column;

  }

  .btn-group {

    flex-direction: column;

    gap: 8px;

  }

}

**4. Run the frontend**

npm start

**🔗 Connecting MongoDB Atlas (Optional)**

Replace mongodb://localhost:27017/crudDB with your **MongoDB Atlas URI** in server.js.

**✅ Final Test**

* Backend: http://localhost:5000
* Frontend: http://localhost:3000

You can now insert, display, update, and delete items.

**ERROR FIX :-**

This error usually occurs when there's a **mismatch or corruption in your node\_modules**, especially with ajv and ajv-keywords, which are often used by tools like Webpack under the hood.

Let’s fix it step-by-step:

**✅ Step-by-Step Fix**

**1. Delete node\_modules and package-lock.json**

In your frontend folder:

rd /s /q node\_modules

del package-lock.json

Alternatively, you can do it from VS Code or File Explorer.

**2. Clear NPM Cache (optional but recommended)**

npm cache clean --force

**3. Reinstall Packages**

npm install

**4. Start the App Again**

npm start

**If That Doesn’t Work — Try Installing ajv Manually**

npm install ajv

If needed:

npm install ajv-keywords