Here's a **basic full-stack setup** using **React.js**, **Express.js**, and **MongoDB** to help you get started. We'll break it into parts:

---

# □ 1. Backend – Express + MongoDB

## a. Setup

Create a new backend folder and initialize a Node.js project:

```
mkdir backend
cd backend
npm init -y
```

## b. Install dependencies

```
npm install express mongoose cors
```

## c. Create basic Express server (`index.js`)

```
// 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());

// MongoDB connection
mongoose.connect('mongodb://localhost:27017/mydb', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log('MongoDB connected'))
.catch(err => console.error(err));

// Simple schema
const UserSchema = new mongoose.Schema({
  name: String,
  email: String,
});
const User = mongoose.model('User', UserSchema);

// Routes
app.get('/api/users', async (req, res) => {
  const users = await User.find();
  res.json(users);
});

app.post('/api/users', async (req, res) => {
```

```
  const user = new User(req.body);
  await user.save();
  res.json(user);
});

// Start server
app.listen(5000, () => {
  console.log('Server running on http://localhost:5000');
});
```

---

# □ 2. Frontend – React

## a. Setup React project

In a separate folder:

```
npx create-react-app frontend
cd frontend
npm install axios
```

## b. Create a basic form and display list (`App.js`)

```
// frontend/src/App.js
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function App() {
  const [users, setUsers] = useState([]);
  const [form, setForm] = useState({ name: '', email: '' });

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

  const fetchUsers = async () => {
    const res = await axios.get('http://localhost:5000/api/users');
    setUsers(res.data);
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    await axios.post('http://localhost:5000/api/users', form);
    setForm({ name: '', email: '' });
    fetchUsers();
  };

  return (
    <div>
      <h1>Users</h1>
      <form onSubmit={handleSubmit}>
        <input
          placeholder="Name"
```

```
          value={form.name}
          onChange={e => setForm({ ...form, name: e.target.value })}
        />
        <input
          placeholder="Email"
          value={form.email}
          onChange={e => setForm({ ...form, email: e.target.value })}
        />
        <button type="submit">Add User</button>
      </form>
      <ul>
        {users.map((user, idx) => (
          <li key={idx}>{user.name} ({user.email})</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

---

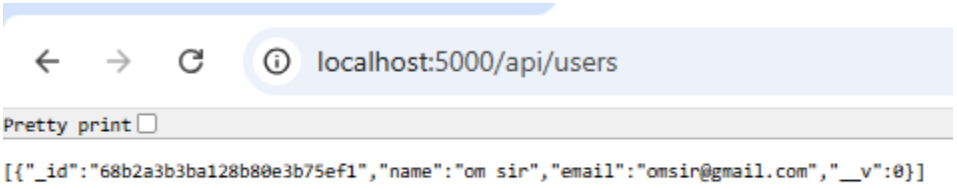
# □ Final Steps

- Run backend:

  ```
  node index.js
  ```

**For Output Open Browser:-**

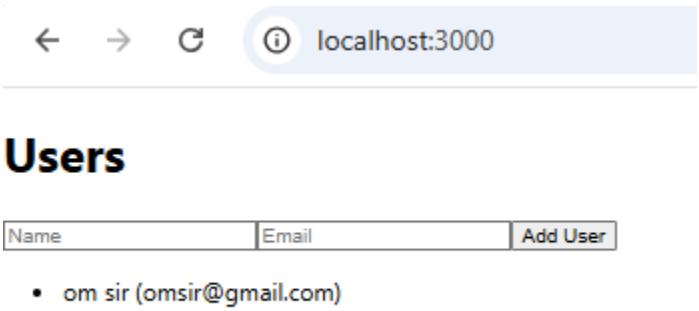
http://localhost:5000/api/users



- Run frontend:

```
npm start
```

```
output:-
```




Make sure MongoDB is running locally on port `27017`.

---

## □ Result

- A simple React frontend to **create and list users**
- A Node/Express backend connected to **MongoDB**
- Communication via REST API

Absolutely! Let's break down **each line of code** from both the **backend (Express + MongoDB)** and **frontend (React)** so you can fully understand what's happening.

---

## □ BACKEND: `index.js` (Express + MongoDB)

```
// backend/index.js
```

```
const express = require('express');          // Import Express.js framework
const mongoose = require('mongoose');// Import Mongoose to interact with
MongoDB
const cors = require('cors');                // Import CORS to allow cross-
origin requests (like from React)

const app = express();                       // Create an Express application
app.use(cors());                             // Use CORS middleware so
frontend can access backend
app.use(express.json());                     // Parse JSON data from incoming
requests (like POST body)
```

---

## □ Connect to MongoDB

```
mongoose.connect('mongodb://localhost:27017/mydb', {
  useNewUrlParser: true,                     // Use new URL parser to remove
warning
  useUnifiedTopology: true,                  // Use new server discovery and
monitoring engine
})
.then(() => console.log('MongoDB connected'))// Log success
.catch(err => console.error(err));           // Log error if connection fails
```

- This connects to MongoDB on your local machine at `mongodb://localhost:27017`
- `mydb` is the database name.

---

## □ Create a Mongoose Schema & Model

```
const UserSchema = new mongoose.Schema({
  name: String,                              // User has a name (String)
  email: String,                             // User has an email (String)
});
const User = mongoose.model('User', UserSchema); // Create User model from
schema
```

- `User` is now a model you can use to read/write to the `users` collection in MongoDB.

---

## □ API Routes

### *1. GET users*

```
app.get('/api/users', async (req, res) => {
  const users = await User.find();           // Find all users from MongoDB
  res.json(users);                           // Return users as JSON
});
```

*2. POST user*

```
app.post('/api/users', async (req, res) => {
  const user = new User(req.body);          // Create a new user from request
body
  await user.save();                        // Save user to MongoDB
  res.json(user);                           // Return the saved user as JSON
});
```

---

## □ Start the server

```
app.listen(5000, () => {
  console.log('Server running on http://localhost:5000');
});
```

- This starts the server on port `5000`.

---

□ Backend Summary:

| Feature | Explanation |
|---|---|
| Express | Backend framework |
| Mongoose | MongoDB connector |
| CORS | Allow frontend to talk to backend |
| API Routes | Create and fetch users |
| MongoDB | Stores user data |

---

# □ FRONTEND: `App.js` (React + Axios)

```
import React, { useState, useEffect } from 'react'; // React imports for
state and side effects
import axios from 'axios';                          // Axios for HTTP
requests
```

---

## □ useState and useEffect hooks

```
const [users, setUsers] = useState([]);             // State to store list of
users
```

```
const [form, setForm] = useState({ name: '', email: '' }); // State to handle
form input
useEffect(() => {
  fetchUsers();                                    // Call fetchUsers when
component mounts
}, []);
```

---

## □ Fetch users from backend

```
const fetchUsers = async () => {
  const res = await axios.get('http://localhost:5000/api/users'); // Make GET
request
  setUsers(res.data);                          // Save returned users to state
};
```

---

## □ Submit new user

```
const handleSubmit = async (e) => {
  e.preventDefault();                          // Prevent default form
submission behavior (page reload)
  await axios.post('http://localhost:5000/api/users', form); // POST user to
backend
  setForm({ name: '', email: '' });            // Clear form inputs
  fetchUsers();                                // Reload the list of users
};
```

---

## □ Render UI

```
return (
  <div>
    <h1>Users</h1>

    <form onSubmit={handleSubmit}>
      <input
        placeholder="Name"
        value={form.name}
        onChange={e => setForm({ ...form, name: e.target.value })} // Update
name field
      />
      <input
        placeholder="Email"
        value={form.email}
        onChange={e => setForm({ ...form, email: e.target.value })} // Update
email field
      />
      <button type="submit">Add User</button> // Submit form
    </form>

    <ul>
      {users.map((user, idx) => (
```

```
        <li key={idx}>{user.name} ({user.email})</li> // Display each user
      ))}
    </ul>
  </div>
);
```

---

 Frontend Summary:

| Feature | Explanation |
|---|---|
| `useState` | Store form input and users |
| `useEffect` | Load users when component starts |
| Axios | Send GET/POST requests to backend |
| Form | Input for name and email |
| Render list | Show users from the backend |

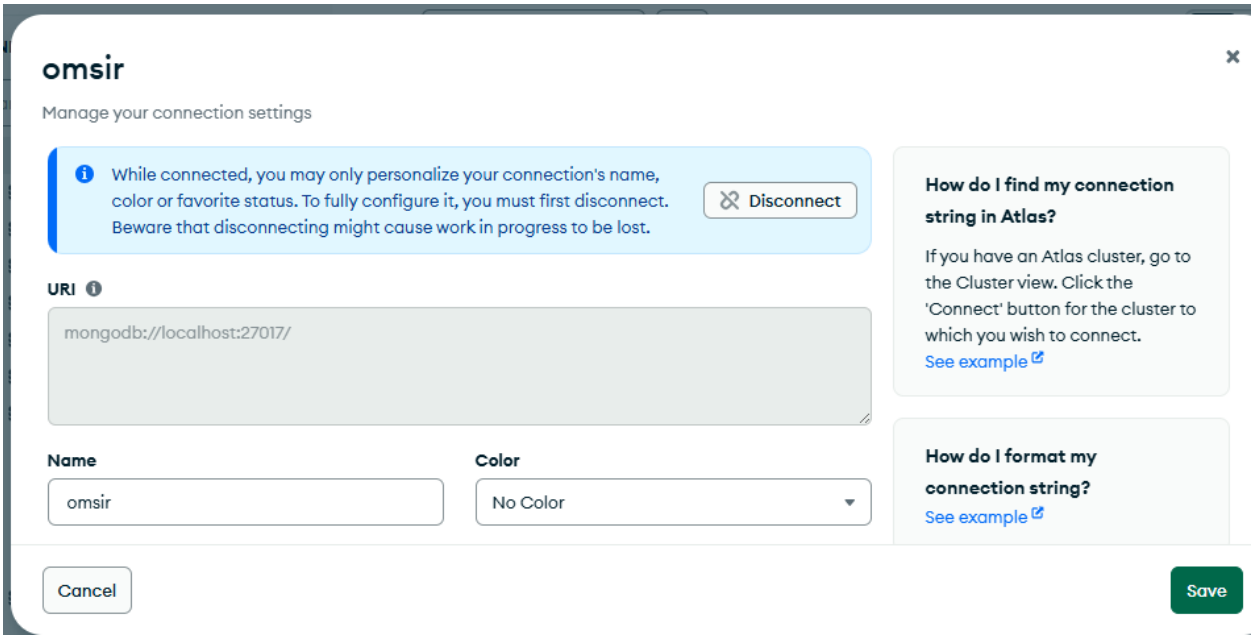
---

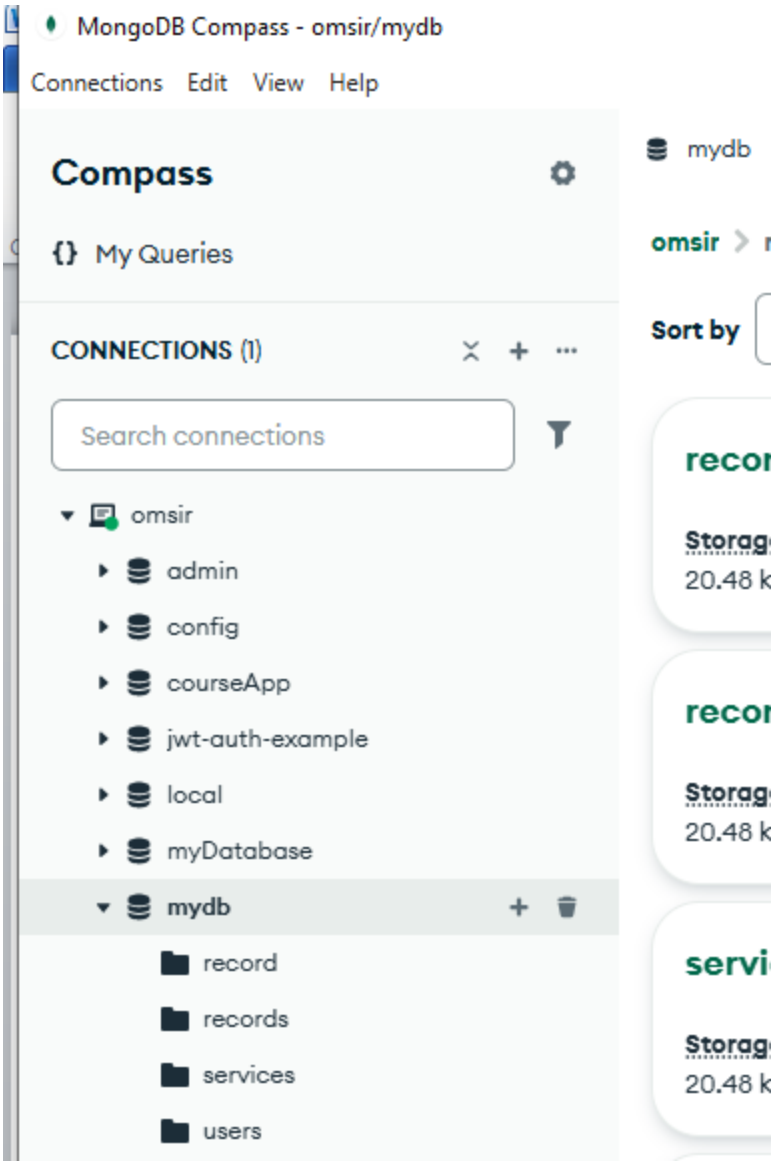
# □ Example User Flow

1. User opens React app → `GET /api/users` is called → users are shown.
2. User fills name/email → submits form → `POST /api/users` is called.
3. Backend saves to MongoDB.
4. React app refreshes the list automatically.

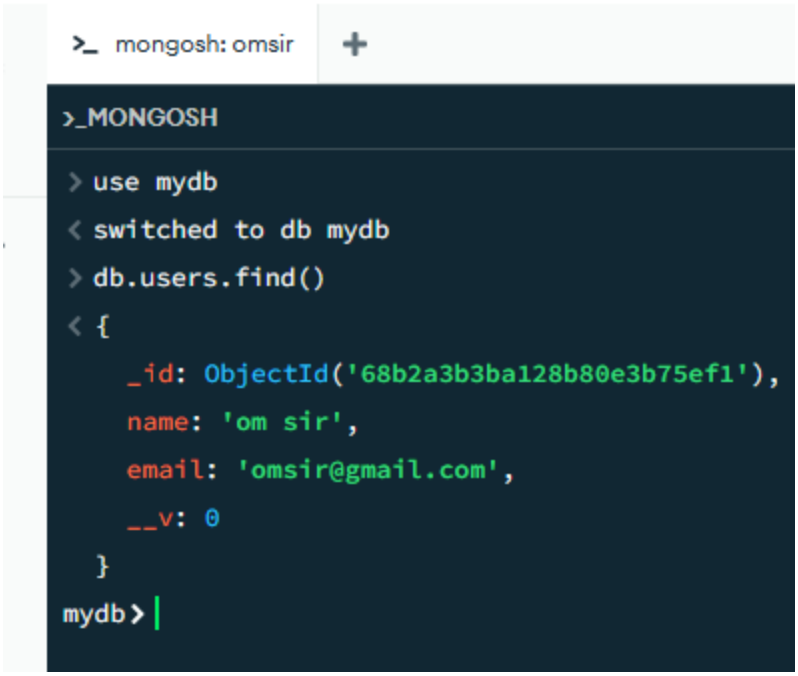
MongoDB :-

Connection settings:-






And Display data using mongoDB shell :-


mongosh: omsir
>_MONGOSH
> use mydb
< switched to db mydb
> db.users.find()
< {
_id: ObjectId('68b2a3b3ba128b80e3b75ef1'),
name: 'om sir',
email: 'omsir@gmail.com',
__v: 0
}
mydb>