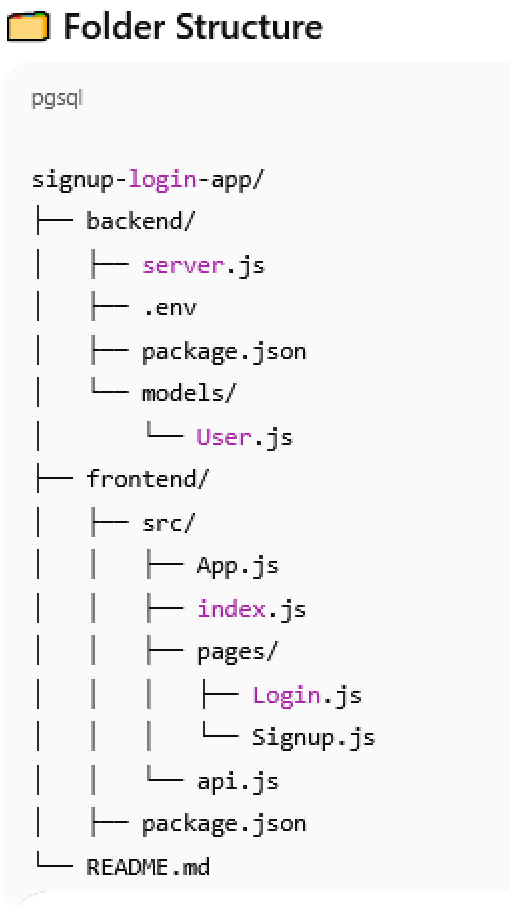
Let’s make a simple signup and login project using:

- **React** → frontend
- **Express.js** → backend middleware
- **MongoDB Atlas** → cloud database

Below is a **complete project structure and working code** you can copy-paste to run.



Folder Structure

pgsql

```
signup-login-app/
├── backend/
│   ├── server.js
│   ├── .env
│   ├── package.json
│   └── models/
│       └── User.js
├── frontend/
│   ├── src/
│   │   ├── App.js
│   │   ├── index.js
│   │   ├── pages/
│   │   │   ├── Login.js
│   │   │   └── Signup.js
│   │   └── api.js
│   ├── package.json
└── README.md
```

backend/package.json

```
{
  "name": "backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "mongoose": "^7.6.0"
  },
  "devDependencies": {
    "nodemon": "^3.0.2"
  }
}
```

.env

```
MONGO_URI=your_mongodb_atlas_connection_string
PORT=5000
JWT_SECRET=mysecretkey
```

backend/models/User.js

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  name: String,
  email: { type: String, unique: true },
  password: String
});

module.exports = mongoose.model('User', userSchema);
```

backend/server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');
const bcrypt = require('bcryptjs');
const User = require('./models/User');

dotenv.config();
const app = express();

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

mongoose.connect(process.env.MONGO_URI)
  .then(() => console.log("✅ MongoDB Connected"))
  .catch(err => console.log(err));

// Signup Route
app.post('/signup', async (req, res) => {
  const { name, email, password } = req.body;
  const userExists = await User.findOne({ email });
  if (userExists) return res.status(400).json({ msg: "User already exists" });

  const hashedPassword = await bcrypt.hash(password, 10);
  const user = new User({ name, email, password: hashedPassword });
  await user.save();
  res.json({ msg: "User registered successfully" });
});

// Login Route
app.post('/login', async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user) return res.status(400).json({ msg: "User not found" });

  const isMatch = await bcrypt.compare(password, user.password);
  if (!isMatch) return res.status(400).json({ msg: "Invalid credentials" });

  res.json({ msg: "Login successful", user });
});

app.listen(process.env.PORT, () => {
  console.log(`🚀 Server running on port ${process.env.PORT}`);
});
```

Step-by-Step: Create a Free MongoDB Atlas Cluster/Database

□ Step 1: Go to MongoDB Atlas

☞ Visit: https://www.mongodb.com/cloud/atlas

Step 2: Sign Up or Log In

You can:

- Sign up with **Google**, **GitHub**, or **email/password**.
  Once logged in, you'll be redirected to your **Atlas Dashboard**.

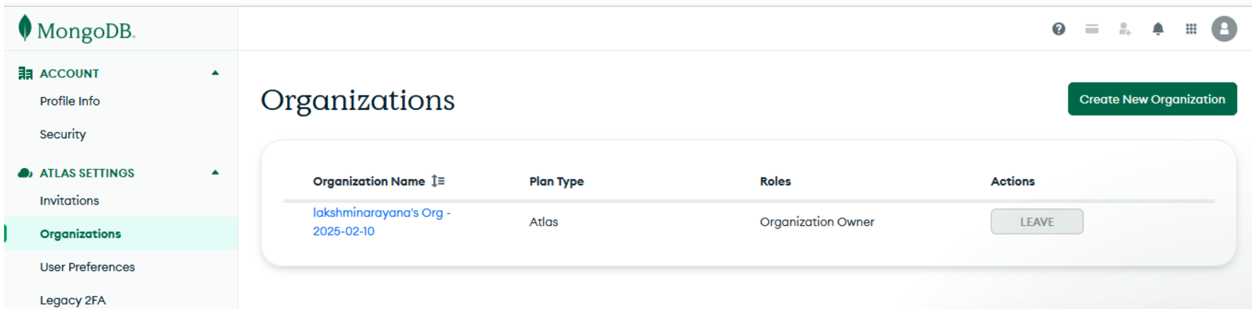
MongoDB Atlas Structure

Organization → contains multiple Projects → each Project contains Clusters

- **Organization** = a company-level or top-level container
- **Project** = where your actual **clusters (databases)** live

Step 2.1: Create an Organization

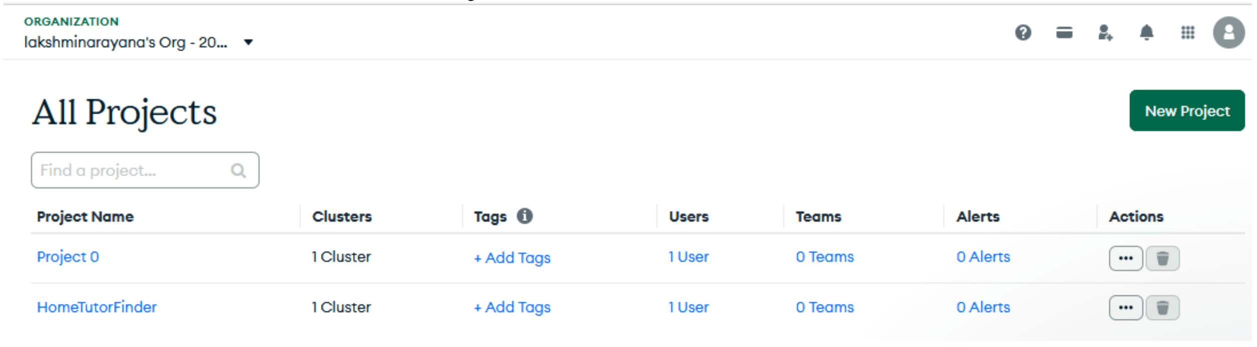
1. Click **"Create New Organization"**
2. Enter any name — e.g. `MyOrg` or `Personal`
3. Click **"Create Organization"**



Organizations

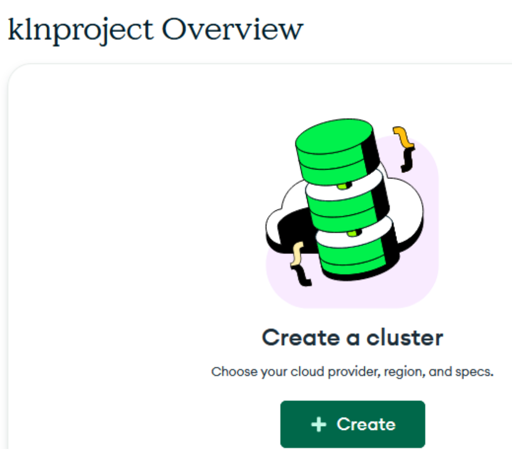
Organization Name	Plan Type	Roles	Actions
lakshminarayana's Org - 2025-02-10	Atlas	Organization Owner	LEAVE

Step 2.2: Create a New Project

1. Click **"New Project"**
2. Enter a **project name** (e.g., `SignupLoginApp`)
3. Click **"Next"** and then **"Create Project"**
4. 

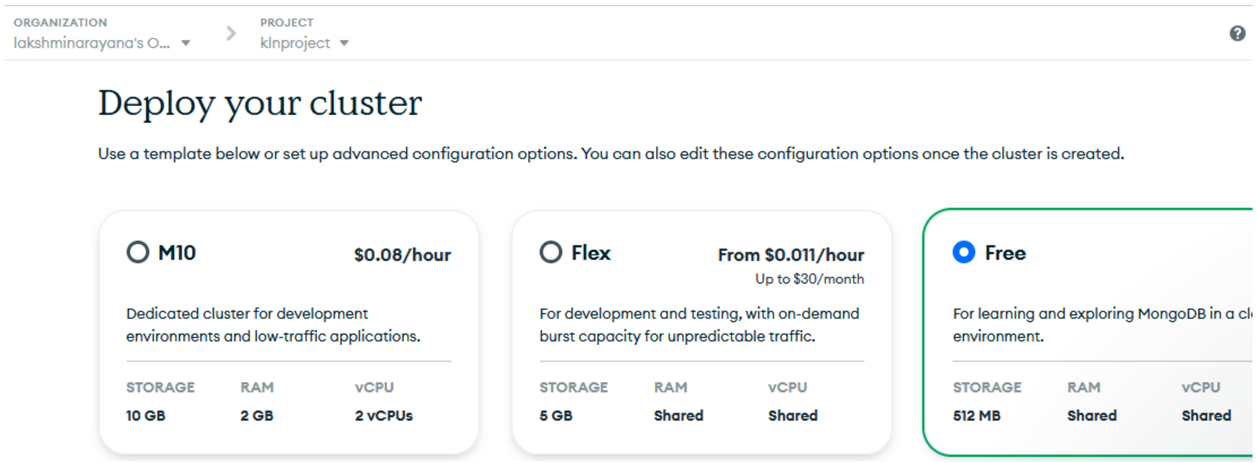
All Projects

Project Name	Clusters	Tags	Users	Teams	Alerts	Actions
Project 0	1 Cluster	+ Add Tags	1 User	0 Teams	0 Alerts	
HomeTutorFinder	1 Cluster	+ Add Tags	1 User	0 Teams	0 Alerts	



Step 4: Build a Cluster / Database

1. Click **"Create a Cluster / Build a Database"**
2. Choose **"Shared" (FREE)** — It's the **M0 cluster (Free Tier)**
   → It says "Shared clusters: Free forever".
3. Click **"Create"**


ORGANIZATION lakshminarayana's O... PROJECT klnproject

Deploy your cluster

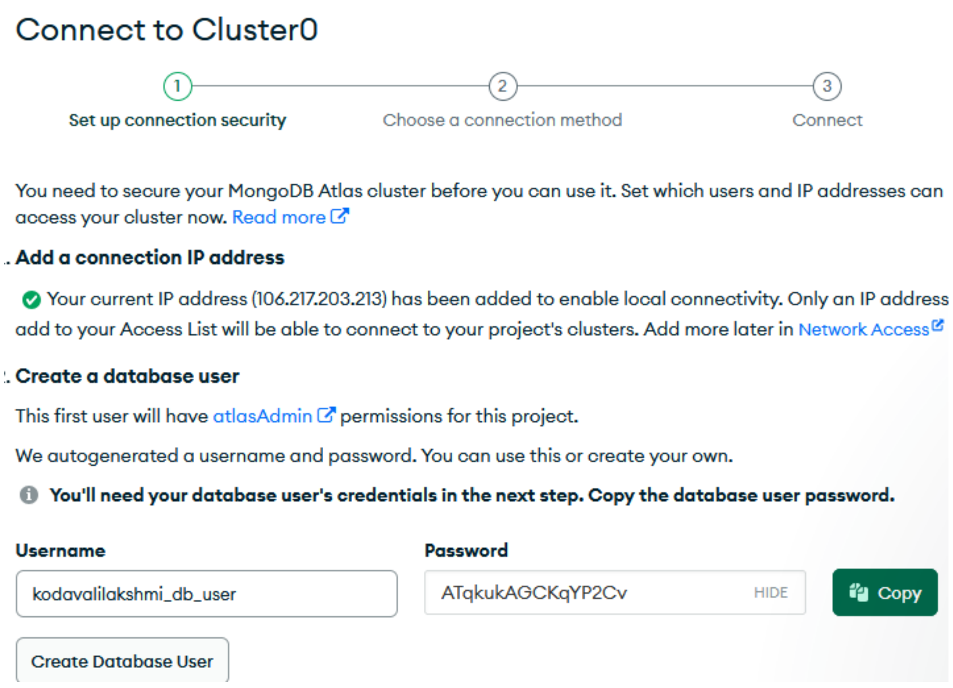
Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

M10	$0.08/hour		
Dedicated cluster for development environments and low-traffic applications.			
	STORAGE	RAM	vCPU
	10 GB	2 GB	2 vCPUs
Flex	From $0.011/hour Up to $30/month		
For development and testing, with on-demand burst capacity for unpredictable traffic.			
	STORAGE	RAM	vCPU
	5 GB	Shared	Shared
Free			
For learning and exploring MongoDB in a cl environment.			
	STORAGE	RAM	vCPU
	512 MB	Shared	Shared

⚙ Step 5: Configure Your Cluster

1. **Cloud Provider**: Choose any (usually **AWS**).
2. **Region**: Pick one near you (for India, e.g. `Mumbai (AWS ap-south-1)`).
3. Keep **M0 Sandbox (FREE)** selected.
4. Click **"Create Deployment"**

🕒 Wait 1–2 minutes while it sets up your free cluster.


Connect to Cluster0

1 Set up connection security — 2 Choose a connection method — 3 Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. Read more

. Add a connection IP address

Your current IP address (106.217.203.213) has been added to enable local connectivity. Only an IP address add to your Access List will be able to connect to your project's clusters. Add more later in Network Access

. Create a database user

This first user will have atlasAdmin permissions for this project.

We autogenerated a username and password. You can use this or create your own.

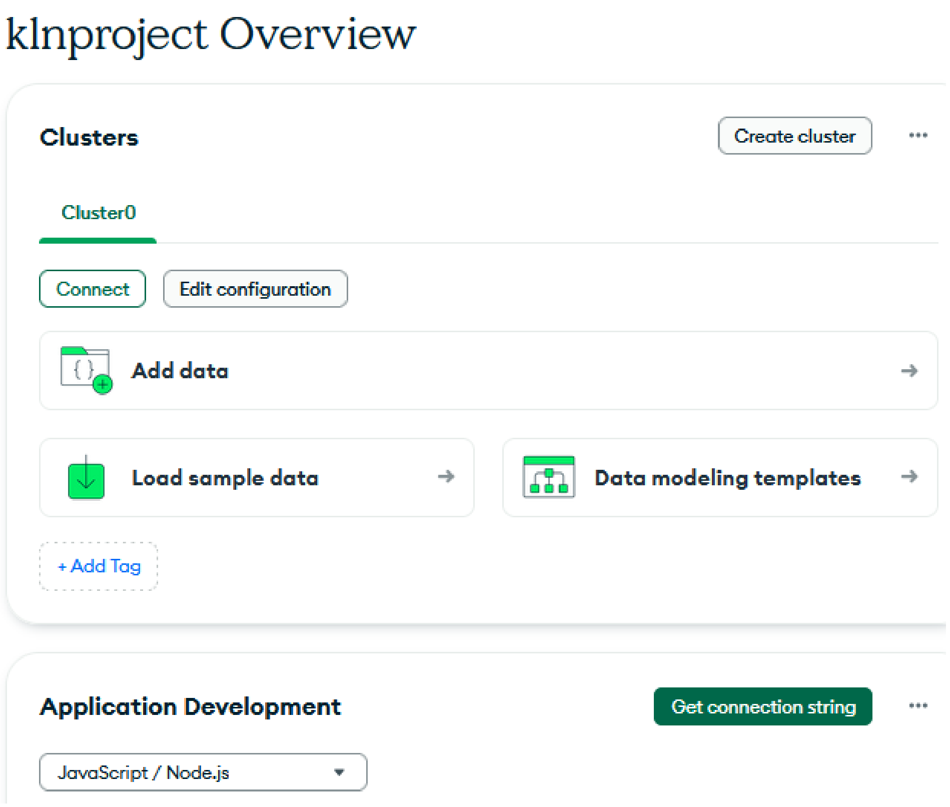
You'll need your database user's credentials in the next step. Copy the database user password.

Username: kodavalilakshmi_db_user

Password: ATqkukAGCKqYP2Cv HIDE Copy

Create Database User

our current IP address (106.217.203.213)
UN: kodavalilakshmi_db_user (kln)
PW: ATqkukAGCKqYP2Cv (saixxxdd)

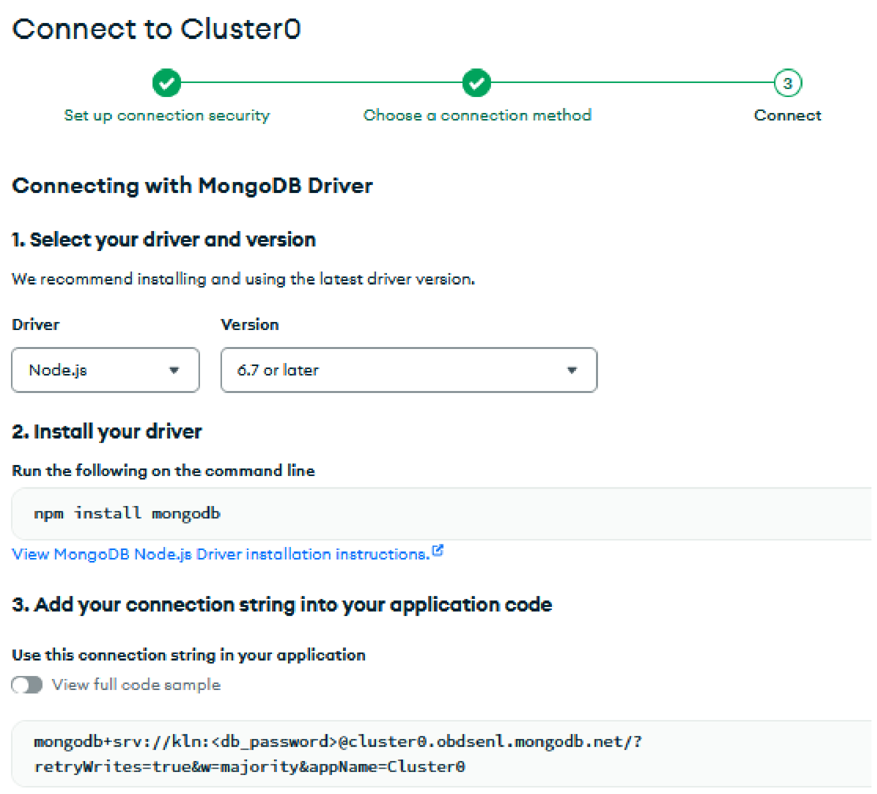


Connection Method: You may choose :"Drivers" (Other options Atlos / Shell /…)

Step 6: Create a Database User

1. In the "Database Access" step:
   - Click **"Add New Database User"**
   - Choose a username (e.g. `myuser`)
   - Set a **strong password** (e.g. `MyPass1234`)
   - Keep "**Read and write to any database**" selected.
   - Click **"Add User"**

📌 Save this username and password — you'll need them later.


Connect to Cluster0

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver: Node.js

Version: 6.7 or later

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

View MongoDB Node.js Driver installation instructions.

3. Add your connection string into your application code

Use this connection string in your application

View full code sample

```
mongodb+srv://kln:<db_password>@cluster0.obdsenl.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0
```

Connction string:

```
mongodb+srv://kln:<db_password>@cluster0.obdsenl.mongodb.net/?retryWrites=tru
e&w=majority&appName=Cluster0
```

🌍 Step 7: Allow Network Access

1. Go to **Network Access** (left sidebar)
2. Click **"Add IP Address"**
3. Choose:
   - **Allow access from anywhere** → `0.0.0.0/0`
   - Or restrict to your current IP
4. Click **"Confirm"**

🔗 Step 8: Get Connection String (URI)

1. Go to **Database** → **Connect** → **Connect your application**
2. Copy the connection string — it looks like:

mongodb+srv://<username>:<password>@cluster0.abcd123.mongodb.net/?retryWrites=true&w=majority

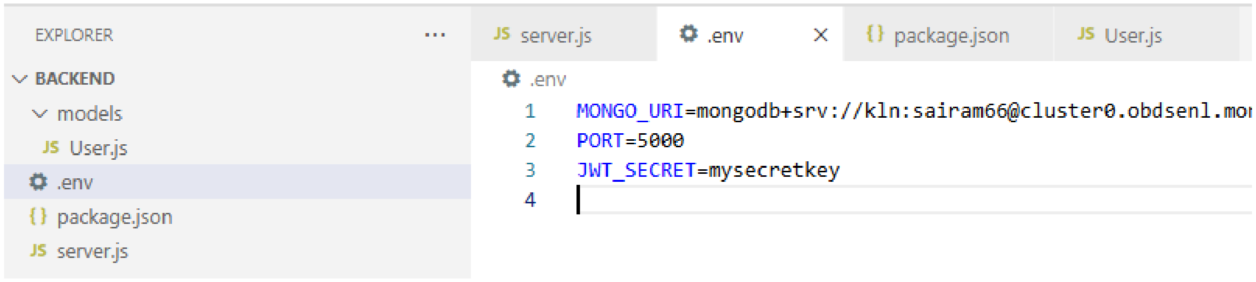
Replace `<username>` and `<password>` with your credentials.

Step 9: Add URI in `.env` File

In your backend project folder, open `.env` and paste:

```
MONGO_URI=mongodb+srv://myuser:MyPass1234@cluster0.abcd123.mongodb.net/mydatabase
PORT=5000
```

(You can replace `mydatabase` with any database name you like — e.g. `signupdb`)



```
MONGO_URI=mongodb+srv://kln:sairam66@cluster0.obdsenl.mo
PORT=5000
JWT_SECRET=mysecretkey
```

Step 10: Test the Connection

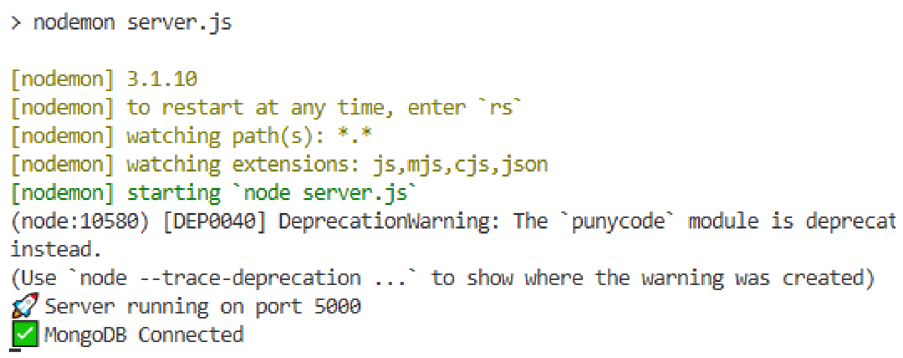
Run the Project

1□ Run Backend

```
cd backend
npm install
npm run dev
```

If everything is correct, you'll see:

✓ MongoDB Connected
🚀 Server running on port 5000



```
> nodemon server.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
(node:10580) [DEP0040] DeprecationWarning: The `punycode` module is deprecat
instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
🚀 Server running on port 5000
✅ MongoDB Connected
```

Backend is ready, Now lets go for froend part.

Frontend (React)

Create React App

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

frontend/src/api.js

```
import axios from 'axios';

const api = axios.create({
  baseURL: "http://localhost:5000",
});

export default api;
```

frontend/src/App.js

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
import Signup from './pages/Signup';
import Login from './pages/Login';

function App() {
  return (
    <Router>
      <nav style={{ margin: "20px" }}>
        <Link to="/signup" style={{ marginRight: "10px" }}>Signup</Link>
        <Link to="/login">Login</Link>
      </nav>
      <Routes>
        <Route path="/signup" element={<Signup />} />
        <Route path="/login" element={<Login />} />
      </Routes>
    </Router>
  );
}

export default App;
```

frontend/src/pages/Signup.js

```
import React, { useState } from 'react';
import api from '../api';

function Signup() {
  const [form, setForm] = useState({ name: '', email: '', password: '' });

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await api.post('/signup', form);
      alert(res.data.msg);
    } catch (err) {
      alert(err.response.data.msg);
    }
  };

  return (
    <div style={{ margin: "50px" }}>
      <h2>Signup</h2>
      <form onSubmit={handleSubmit}>
        <input placeholder="Name" onChange={e => setForm({ ...form, name: e.target.value })} /><br />
        <input placeholder="Email" onChange={e => setForm({ ...form, email: e.target.value })} /><br />
        <input type="password" placeholder="Password" onChange={e => setForm({ ...form, password:
e.target.value })} /><br />
        <button type="submit">Signup</button>
      </form>
    </div>
  );
}

export default Signup;
```

frontend/src/pages/Login.js

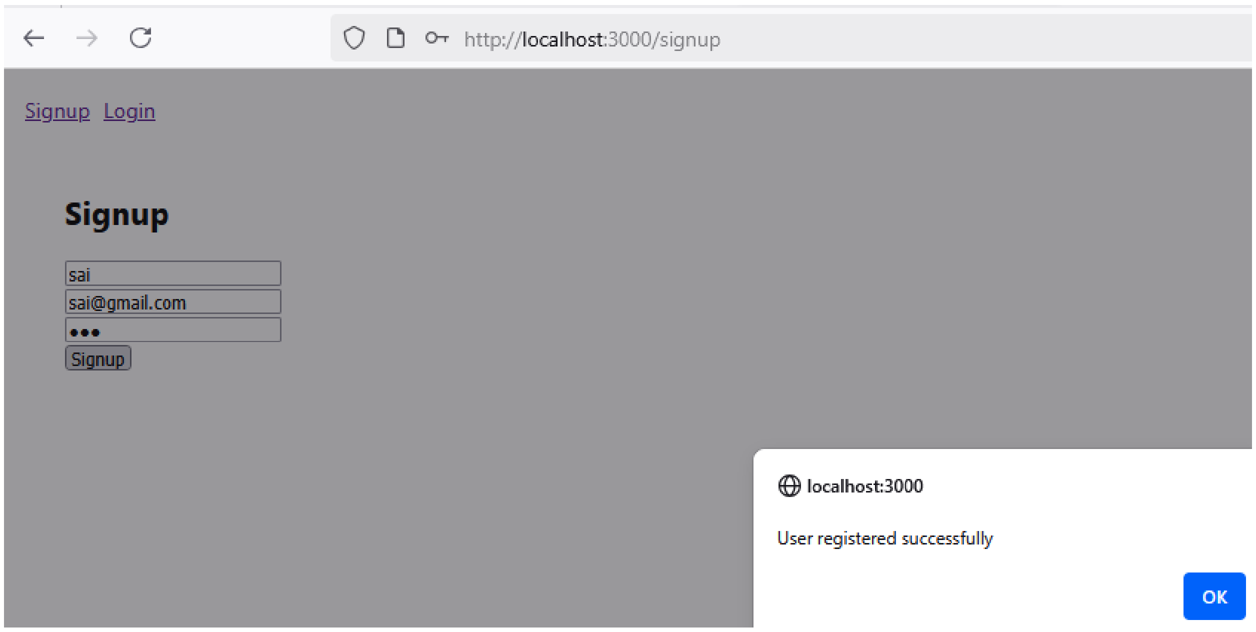
```
import React, { useState } from 'react';
import api from '../api';

function Login() {
  const [form, setForm] = useState({ email: '', password: '' });

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const res = await api.post('/login', form);
      alert(res.data.msg);
    } catch (err) {
      alert(err.response.data.msg);
    }
  };

  return (
    <div style={{ margin: "50px" }}>
      <h2>Login</h2>
      <form onSubmit={handleSubmit}>
        <input placeholder="Email" onChange={e => setForm({ ...form, email: e.target.value })} /><br />
        <input type="password" placeholder="Password" onChange={e => setForm({ ...form, password:
e.target.value })} /><br />
        <button type="submit">Login</button>
      </form>
    </div>
  );
}

export default Login;
```

Run Frontend

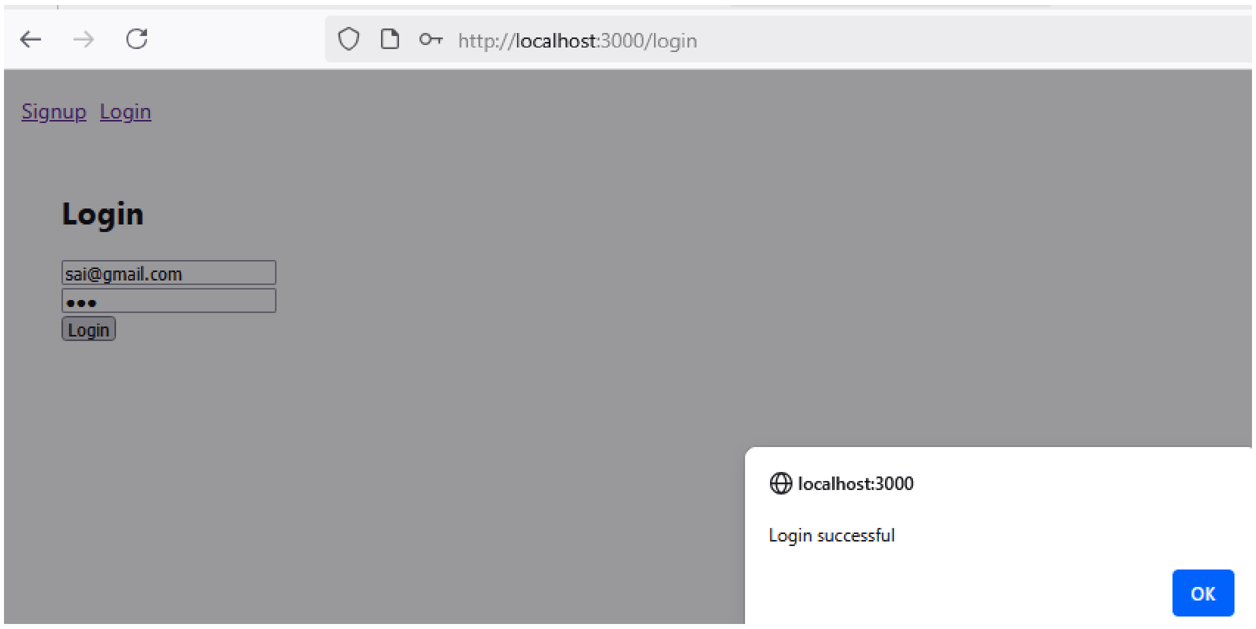
```
cd frontend
npm start
```

Output

- Visit `http://localhost:3000/signup` → create account
- Visit `http://localhost:3000/login` → log in



User registered successfully

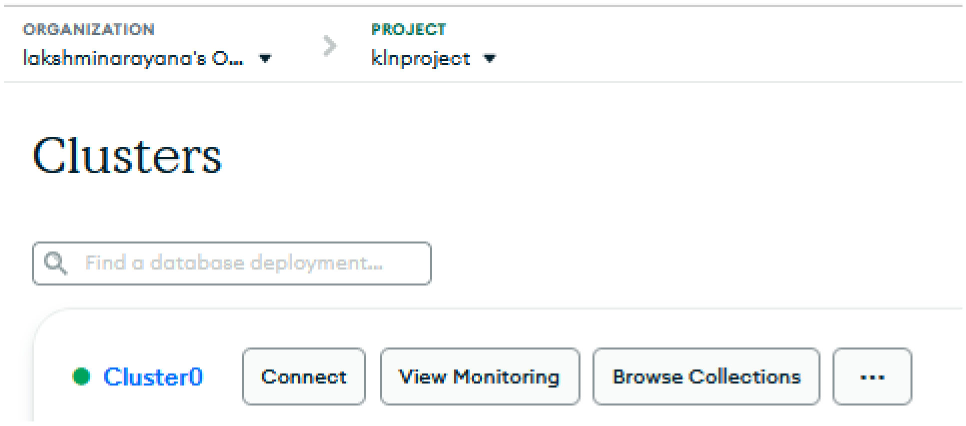


Login

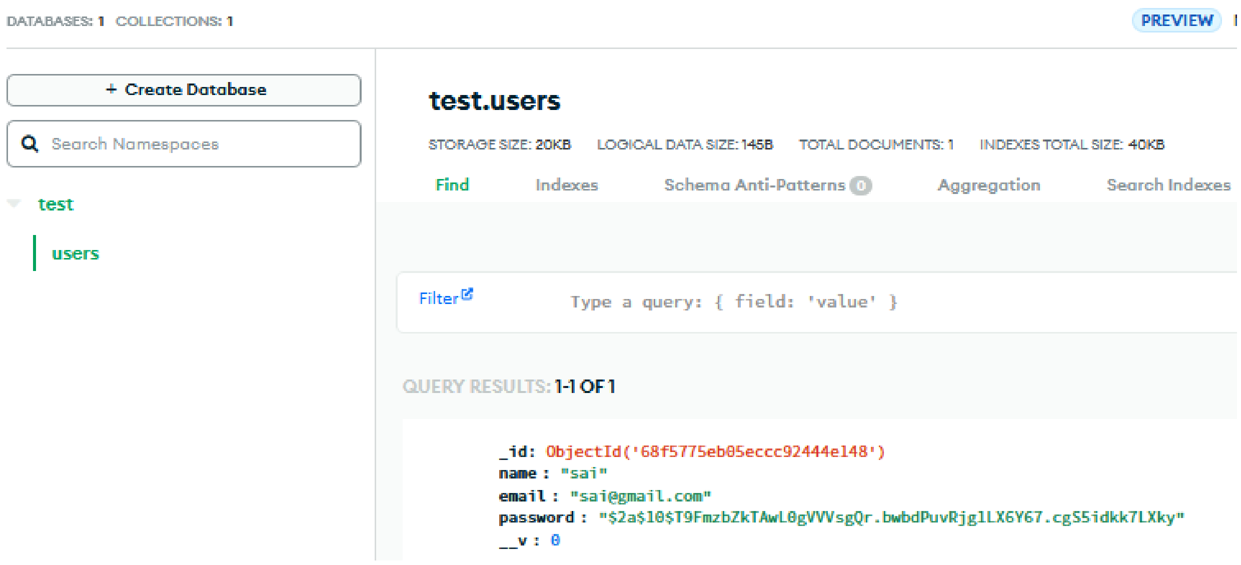
```
sai@gmail.com
•••
Login
```

```
localhost:3000

Login successful

OK
```

Click on “Browse Collections” to verify the user details in the database.



User details as follows:



```
_id: ObjectId('68f5775eb05eccc92444e148')
name : "sai"
email : "sai@gmail.com"
password : "$2a$10$T9FmzbZkTAwL0gVVVsgQr.bwbdPuvRjglLX6Y67.cgS5idkk7LXky"
__v : 0
```