<https://node-project-11.onrender.com/users>

<https://chatgpt.com/c/68ff2b83-417c-8320-aa14-b435cf9667f1>

<https://chatgpt.com/c/68ff2b83-417c-8320-aa14-b435cf9667f1>

<https://school-project.onrender.com/>

<https://node-project-11.onrender.com/users>

<https://dashboard.render.com/web/srv-d3vh9v2li9vc73cr4ilg/deploys/dep-d3vh9vili9vc73cr4jfg>

<https://school-project.onrender.com/>

<https://dashboard.render.com/web/srv-d3vh9v2li9vc73cr4ilg/deploys/dep-d3vh9vili9vc73cr4jfg>

-----------------------------------------------------------------------------------------------------------------------------------------

✅ Perfect — that means your TypeScript compiler (tsc) **ran successfully without any errors** 🎉

That’s a **great sign** — it means your local build is clean, so now Render should also be able to deploy successfully.

Let’s continue ste

-----------------------------------------------------------------------------------------------------------------------------------------

Through this link https://node-project-11.onrender.com/users page is showing application loading

**ChatGPT said:**

Got it — when you open  
👉 [**https://node-project-11.onrender.com/users**](https://node-project-11.onrender.com/users),  
it just shows **“Application Loading…”** (and never loads the data).

That means your Node.js app is **running**, but the **frontend or API route** /users isn’t returning visible data.

Let’s diagnose this step by step 👇

**🧩 Step 1 — Check What /users Route Actually Does**

Open your project folder and look inside your Node app’s routes — for example, check:

src/routes/users.js

or

routes/userRoutes.js

Look for something like:

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

// example

const users = await User.findAll();

res.json(users);

});

If it looks like this — it should return JSON data, not a webpage.  
So when you open it in your browser, you should see something like:

[

{ "id": 1, "name": "John" },

{ "id": 2, "name": "Jane" }

]

👉 But if your /users route instead serves a React or HTML file (like index.html), it might be showing a “Loading…” message.

**🧠 Step 2 — Confirm the API Port & Route Binding**

Your log says:

🚀 Server is running on port 3001

✅ PostgreSQL connected successfully

That means your API runs on **port 3001**, but Render automatically routes it through HTTPS → port **443**.

So no problem there — just confirm your app actually defines the /users route.

If your **server.js** looks like this:

app.use('/users', userRoutes);

make sure the userRoutes file **exports a GET /** endpoint.

**🧪 Step 3 — Check the Logs on Render**

Go to your Render dashboard → node\_project-11 → **Logs**  
and look for anything like:

Error: Cannot GET /users

or

Database connection error

If it’s loading forever, it could mean the route hangs (e.g., waiting for a DB query that doesn’t resolve).

**🧰 Step 4 — Test the API manually**

Try this command from your local terminal (if you have curl installed):

curl https://node-project-11.onrender.com/users

If it returns JSON → good.  
If it returns HTML → you’re hitting a frontend, not the API.  
If it hangs → your backend route might be stuck.

**⚙️ Step 5 — (If you’re using a frontend)**

If you built a frontend (React, Angular, etc.) inside your project and your /users page just says “Loading…”, it means:

* The **frontend** is waiting for API data,
* But the **API endpoint** it’s trying to reach is not responding (maybe hardcoded to http://localhost:3001).

In that case, you must change your frontend API URL to:

https://node-project-11.onrender.com/api/users

(or whatever your actual deployed backend route is).

💬 Can you please share:

1. The **relevant part** of your server.js (where you define your /users route or import it),
2. And confirm whether you have a **frontend** (React/Vue/etc.) inside this same project?

Then I’ll tell you exactly what to fix so /users starts showing real data instead of “Loading…”.

✅ Excellent — that database.ts file is **correct and complete** for local development, but here’s an **important note** for Render:

Render **does not** provide separate DB\_USER, DB\_HOST, DB\_NAME, etc. — it provides a **single** environment variable called DATABASE\_URL.

So right now, your connection will **fail on Render** because those individual variables (DB\_USER, etc.) aren’t defined in its environment.

Let’s fix this cleanly so it works both **locally** and **on Render** 👇