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

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

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

<https://chatgpt.com/c/68fdf01c-b638-8320-98e4-b5649dfe25a8>

ChatGPT : <https://chatgpt.com/c/68fdf01c-b638-8320-98e4-b5649dfe25a8>

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

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

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

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

Web Service

node\_project-11

Node [Free](https://dashboard.render.com/web/srv-d3vh9v2li9vc73cr4ilg/plan) [Upgrade your instance](https://dashboard.render.com/web/srv-d3vh9v2li9vc73cr4ilg/plan)

ConnectManual Deploy

Service ID : srv-d3vh9v2li9vc73cr4ilg

[muhsinashoma / node\_project master](https://github.com/muhsinashoma/node_project/tree/master)

[https://node-project-11.onrender.com](https://node-project-11.onrender.com/)

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

dashboard link  
  
Go to your Render Dashboard

New web service

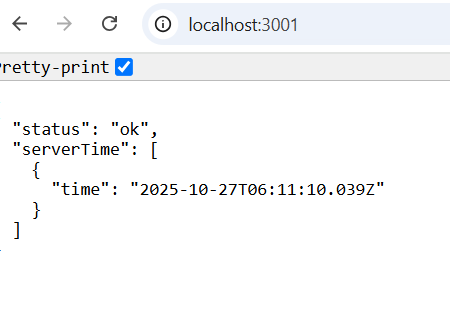
<https://dashboard.render.com/web/new>

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

Mysql database link for render

<https://dashboard.render.com/d/dpg-d3punnk9c44c73ccjbpg-a>

<http://localhost:3001/>



<http://localhost:3001/users/create>

<http://localhost:3001/users>

<http://localhost:3001/>

{

"status": "ok",

"serverTime": [

{

"time": "2025-10-27T06:11:10.039Z"

}

]

}

Perfect! 🎉

That means your **Node.js app is now successfully connected to the PostgreSQL database** on Render.

* ✅ Server is running
* ✅ Database connection works
* ✅ API responds with correct data

**Next Steps**

1. **Test more routes** in your app to ensure all database queries work.
2. **Push your code to GitHub**.
3. **Deploy to Render**:
   * Create a Web Service in Render.
   * Connect your GitHub repo.
   * Set the **Environment Variables** exactly like your local .env.
   * Use npm start as the start command.

After deployment, your live URL (e.g., https://school-project.onrender.com/) should show the same response.

If you want, I can provide a **full Render-ready setup** for your school management project with:

* database.ts
* .env template
* app.ts
* server.ts

So you can deploy **without any connection errors**.

Do you want me to do that?

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

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

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

**Environment Variables**

APP\_NAME : School Management System

APP\_ENV : production

PORT : 3001

BASE\_URL : <https://school-project.onrender.com>

DB\_CONNECTION : postgresql

DB\_HOST : dpg-d3punnk9c44c73ccjbpg-a.singapore-postgres.render.com

DB\_PORT : 5432

DB\_USER : mysql\_xdom\_user

DB\_PASSWORD=Ve6UBQ5psFmwRghDekXMFkILRwTXB902

DB\_NAME : mysql\_xdom

DB\_CHARSET : utf8

DEBUG : false

LOG\_LEVEL : info

TZ : UTC

JWT\_SECRET : your\_jwt\_secret\_key

SESSION\_SECRET : your\_session\_secret

CORS\_ORIGIN : \*

DB\_COLLATION : utf8\_general\_ci

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

<https://render.com/>

<https://dashboard.render.com/>

<https://dashboard.render.com/d/dpg-d3punnk9c44c73ccjbpg-a>

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

Ah — I see exactly what’s happening.

Your current .env is **still configured for MySQL on localhost**, not your Render PostgreSQL. That’s why your Node app is failing to connect when deployed — it’s trying to reach a database on 127.0.0.1:3306, which doesn’t exist in the Render environment.

**🔹 You need to update your .env for PostgreSQL on Render**

Replace the **database section** with your Render Postgres info like this:

# ==========================

# 🗄️ Database Settings (PostgreSQL)

# ==========================

DB\_CONNECTION=postgresql

DB\_HOST=dpg-d3punnk9c44c73ccjbpg-a.singapore-postgres.render.com

DB\_PORT=5432

DB\_USER=mysql\_xdom\_user

DB\_PASSWORD=Ve6UBQ5psFmwRghDekXMFkILRwTXB902

DB\_NAME=mysql\_xdom

DB\_CHARSET=utf8

DB\_COLLATION=utf8\_general\_ci

**🔹 Keep the rest of your .env as is**

APP\_NAME="School Management System"

APP\_ENV=production

PORT=3001

BASE\_URL=https://school-project.onrender.com

DEBUG=false

LOG\_LEVEL=info

TZ=UTC

JWT\_SECRET=your\_jwt\_secret\_key

SESSION\_SECRET=your\_session\_secret

CORS\_ORIGIN=\*

**🔹 Then rebuild and restart**

npm run build

npm start

✅ You should now see:

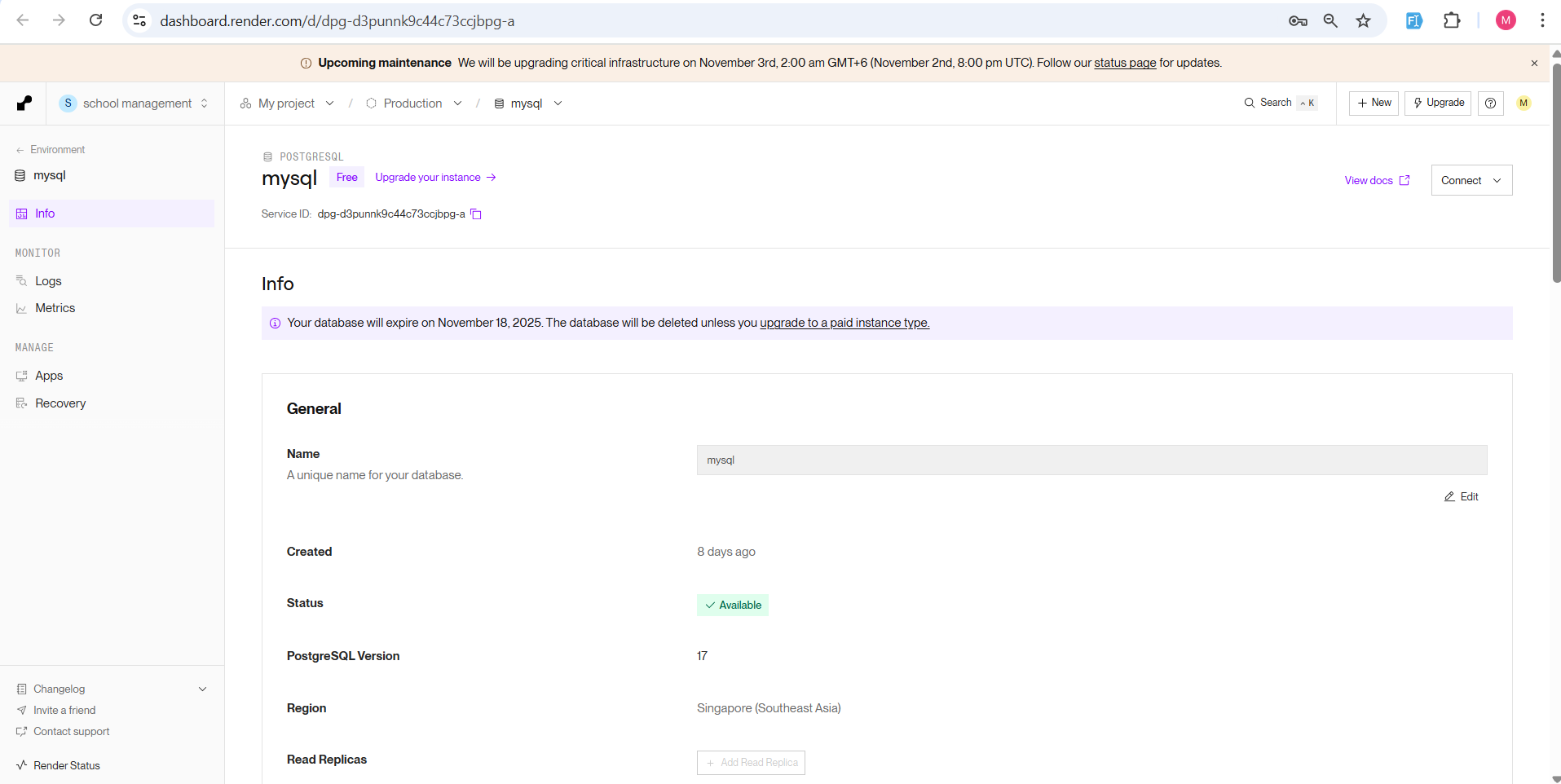
🚀 Server is running on port 3001

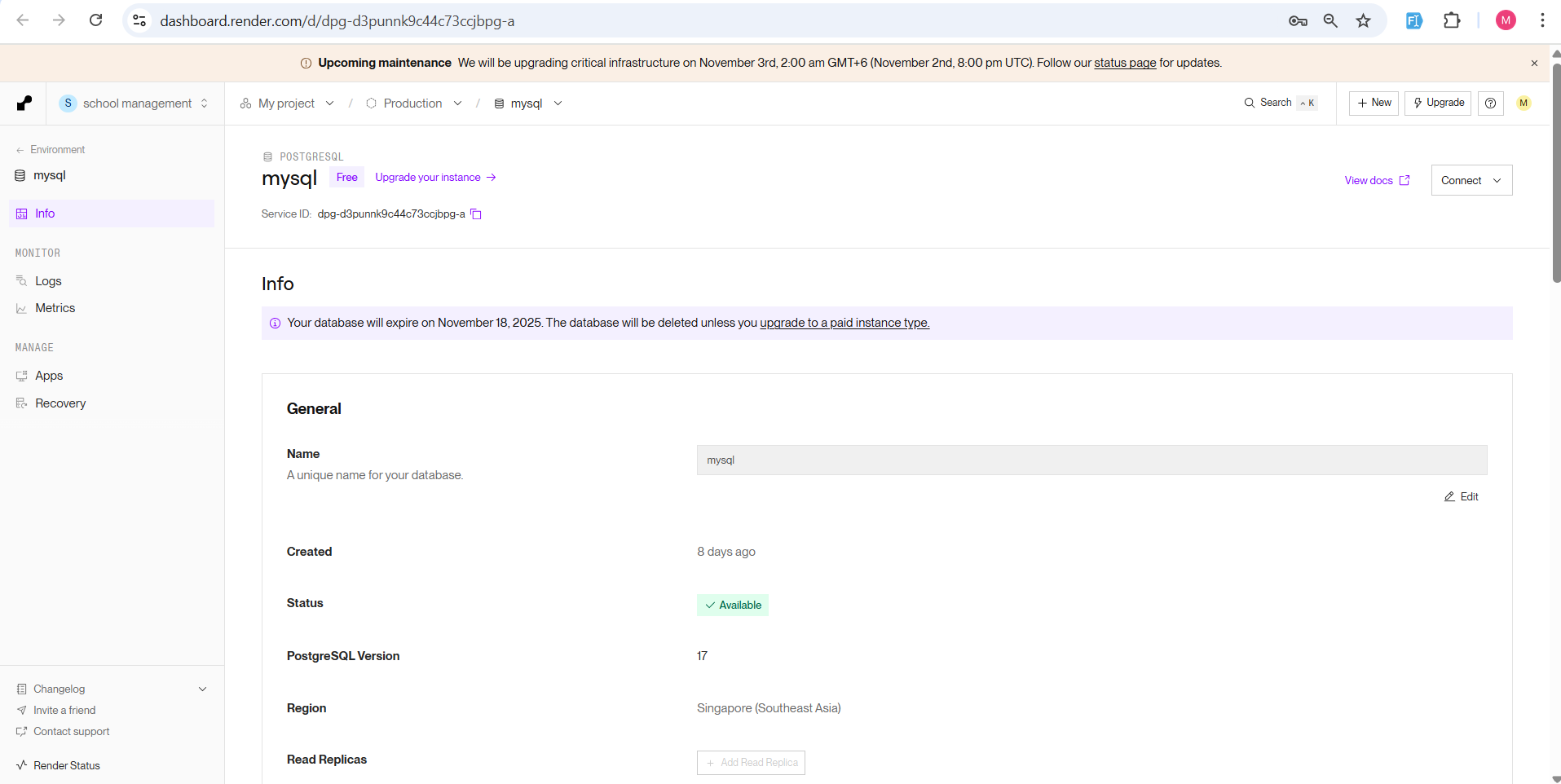
✅ PostgreSQL connected successfully

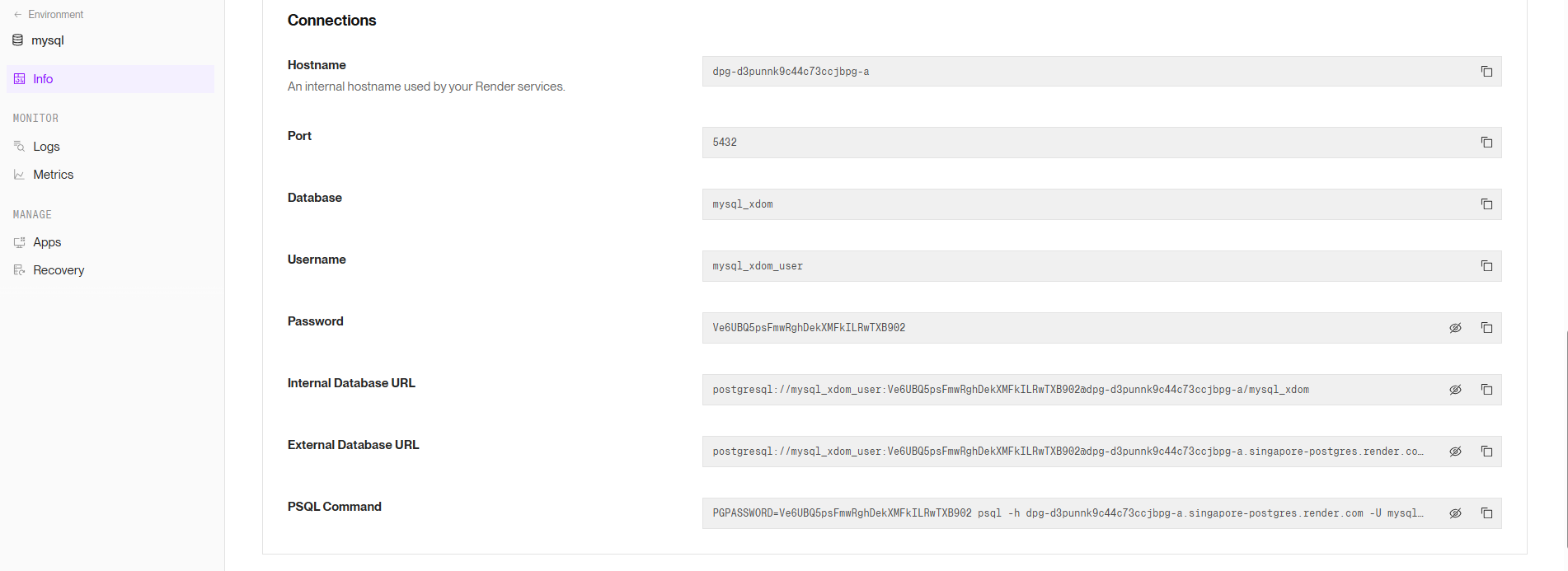
If you want, I can write a **ready-to-deploy .env + database.ts for Render** that will work out-of-the-box with your PostgreSQL credentials. This will eliminate all connection errors.

Do you want me to do that?

<https://dashboard.render.com/d/dpg-d3punnk9c44c73ccjbpg-a>



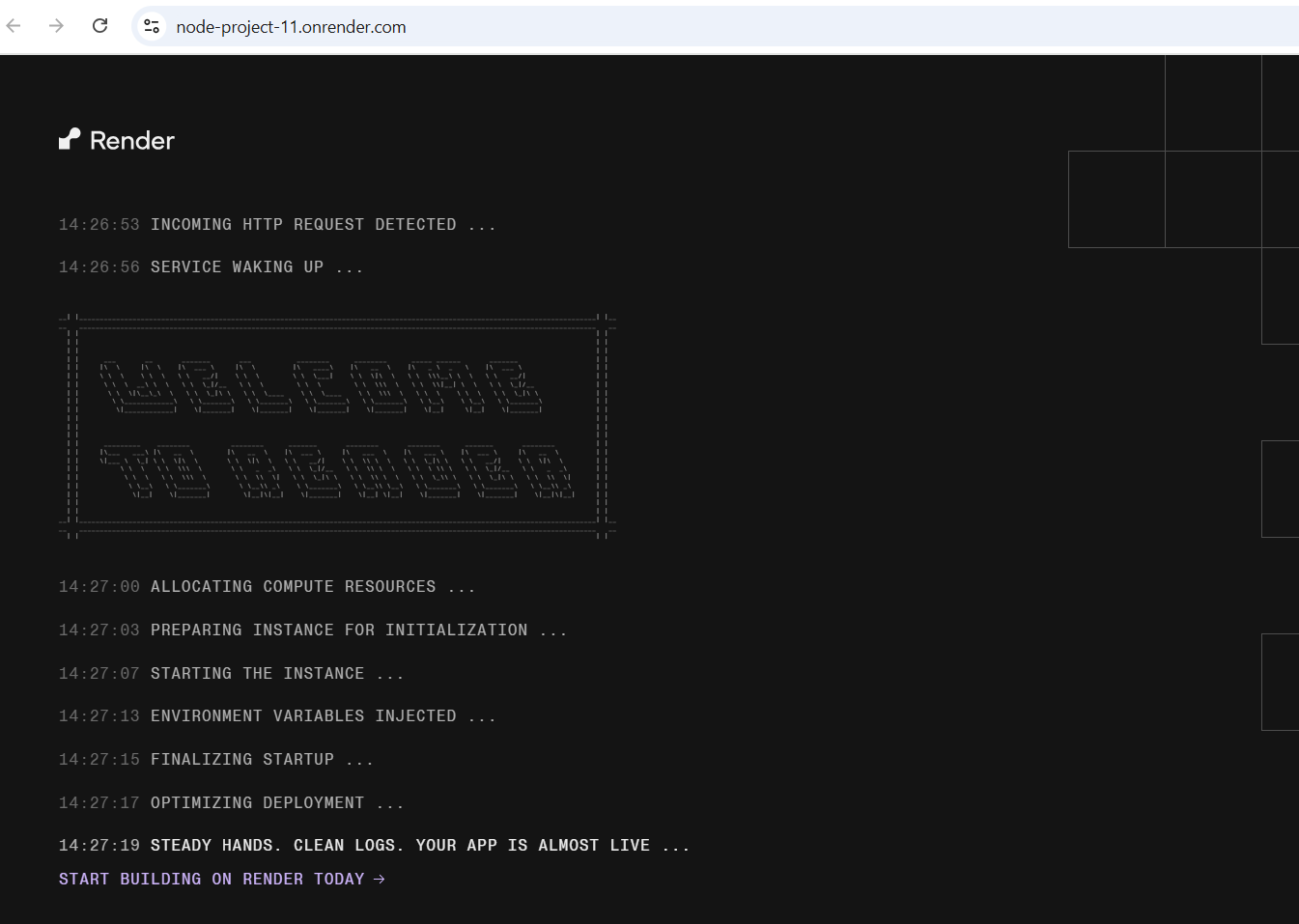




After deployment, your live URL (e.g., https://school-project.onrender.com/) should show the same response.

[**https://status.render.com/**](https://status.render.com/)

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



**⚙️ Quick recap**

* ✅ MySQL uses [rows] = await pool.query(...)
* ✅ PostgreSQL (pg) uses result.rows

**der structure (what it should look like)**

📁 node\_project/

├── app.ts

├── server.ts

├── database.ts ✅ (you’ll add this)

├── package.json

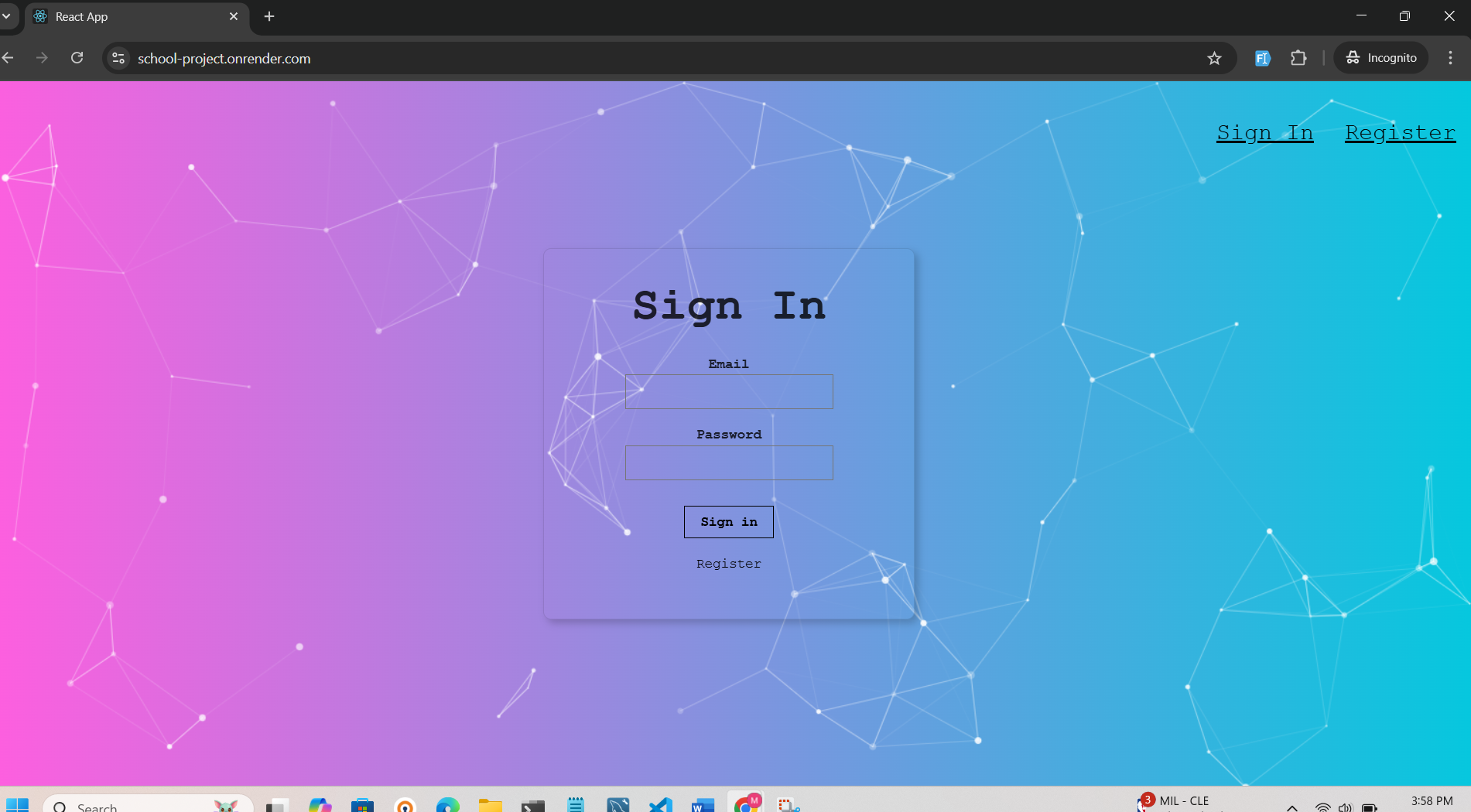
├── tsconfig.json

└── .env

**Google Chrome**

* **Windows/Linux:** Ctrl + Shift + N
* **Mac:** Command + Shift + N
* Or click the **three dots menu → New Incognito Windo**
* **Windows/Linux:** Ctrl + Shift + N(**New Incognito Windo**)

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



If you want, I can **show a trick to keep your free Render instance responsive** so your app loads instantly every time.

Do you want me to do that?

**On Free Cloud Platforms**

You can also deploy this script as a **tiny app on a free cloud service** like:

* Render (as a separate worker service)
* Railway
* Fly.io
* Replit (Node.js project)

It just needs to run continuously to ping your main app.

💡 **Important:** This script does not host your app—it only **pings your app** to keep the free Render instance awake.

Perfect 👏 — since you’ve already created the keep-alive-script folder in another location and saved your keepAlive.js, here’s exactly what to do next:

**✅ Step-by-step next actions:**

**1️⃣ Start the script**

In your terminal, run:

cd keep-alive-script

node keepAlive.js

You should see console logs like:

Pinged! Status code: 200

That means your script is successfully pinging  
👉 https://node-project-11.onrender.com every 4 minutes.

**2️⃣ Keep it running continuously**

If you close your terminal, the script will stop.  
To keep it alive **even when your computer sleeps or restarts**, you have a few options:

**🅐 Simple (while your PC is on)**

Just leave the terminal open — it will run as long as your PC is active.

**🅑 Use pm2 (recommended)**

Install **pm2** globally once:

npm install -g pm2

Then start your script with:

pm2 start keepAlive.js --name "render-keepalive"

Now it will:

* Run in the background.
* Restart automatically if it crashes or reboots.
* You can check logs anytime:
* pm2 logs render-keepalive

If you reboot your PC and want it to start automatically:

pm2 startup

pm2 save

**3️ Confirm your Render app is staying awake**

* Wait 10–15 minutes and visit your Render URL again:  
  👉 <https://node-project-11.onrender.com>
* If it loads instantly (not delayed by ~50 s), your keep-alive pings are working 🎉

Do you want me to show you how to make this **run automatically on startup** (so you never need to start it manually again)?

**You’re using Node locally, so let’s use PM2 autostart**

PM2 can manage background Node.js processes and restart them after reboot.

**✅ Step 1: Make sure PM2 is installed**

If you haven’t already:

npm install -g pm2

Then confirm:

pm2 --version

**✅ Step 2: Start your script with PM2**

Go to your script folder and start it:

cd keep-alive-script

pm2 start keepAlive.js --name "render-keepalive"

Check it’s running:

pm2 list

You should see something like:

┌─────┬─────────────────────┬─────────┬──────┬────────┬───────────┐

│ id │ name │ status │ cpu │ mem │ uptime │

├─────┼─────────────────────┼─────────┼──────┼────────┼───────────┤

│ 0 │ render-keepalive │ online │ 0.0% │ 10.0mb │ 2m │

└─────┴─────────────────────┴─────────┴──────┴────────┴───────────┘

**✅ Step 3: Enable auto-start on boot**

Now tell PM2 to remember this process and restart it automatically after reboot:

pm2 startup

It will show you a command like:

[PM2] To setup the Startup Script, copy/paste the following command:

sudo env PATH=$PATH:/usr/bin pm2 startup systemd -u <your-username> --hp /home/<your-username>

👉 Copy and paste that full line into your terminal and press Enter.

Then save the current PM2 process list:

pm2 save

Now your keep-alive script will:

* Start automatically when your PC boots.
* Keep pinging Render every 4 minutes forever. 🚀

**✅ Step 4: Optional management commands**

* **Check logs:**
* pm2 logs render-keepalive
* **Stop the script:**
* pm2 stop render-keepalive
* **Restart manually (if needed):**
* pm2 restart render-keepalive

Would you like me to give you the same setup guide for **Windows startup (Task Scheduler)** too — in case you ever want to run it there instead of PM2?

**What just happened**

* PM2 successfully started and **daemonized** (it now runs quietly in the background).
* Your script keepAlive.js is **online** and running every 4 minutes.
* It’s actively pinging your Render app (https://node-project-11.onrender.com) to keep it awake.
* You can close this terminal — it will **keep running** even after you do.

**Useful PM2 commands:**

| **Command** | **Purpose** |
| --- | --- |
| pm2 list | Show all running apps |
| pm2 logs render-keepalive | View real-time ping logs |
| pm2 stop render-keepalive | Stop the script |
| pm2 restart render-keepalive | Restart it manually |
| pm2 delete render-keepalive | Remove it completely |
|  |  |

**This is my local PC**

render-keepalive

PS E:\keep-alive-script> pm2 start keepAlive.js --name "render-keepalive"

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

\_\_/\\\\\\\\\\\\\\_\_\_\_/\\\\\_\_\_\_\_\_\_\_\_\_\_\_/\\\\\_\_\_\_/\\\\\\\\\\_\_\_\_\_

\_\/\\\/////////\\\\_\/\\\\\\\_\_\_\_\_\_\_\_/\\\\\\\_\_/\\\///////\\\\_\_\_

\_\/\\\\_\_\_\_\_\_\_\/\\\\_\/\\\//\\\\_\_\_\_/\\\//\\\\_\///\_\_\_\_\_\_\//\\\\_\_

\_\/\\\\\\\\\\\\\/\_\_\/\\\\///\\\/\\\/\_\/\\\\_\_\_\_\_\_\_\_\_\_\_/\\\/\_\_\_

\_\/\\\/////////\_\_\_\_\/\\\\_\_\///\\\/\_\_\_\/\\\\_\_\_\_\_\_\_\_/\\\//\_\_\_\_\_

\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\/\\\\_\_\_\_\///\_\_\_\_\_\/\\\\_\_\_\_\_/\\\//\_\_\_\_\_\_\_\_

\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\/\\\\_\_\_/\\\/\_\_\_\_\_\_\_\_\_\_\_

\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\/\\\\_\_/\\\\\\\\\\\\\\\\_

\_\///\_\_\_\_\_\_\_\_\_\_\_\_\_\_\///\_\_\_\_\_\_\_\_\_\_\_\_\_\_\///\_\_\///////////////\_\_

Runtime Edition

PM2 is a Production Process Manager for Node.js applications

with a built-in Load Balancer.

Start and Daemonize any application:

$ pm2 start app.js

Load Balance 4 instances of api.js:

$ pm2 start api.js -i 4

Monitor in production:

$ pm2 monitor

Make pm2 auto-boot at server restart:

$ pm2 startup

To go further checkout:

http://pm2.io/

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

[PM2] Spawning PM2 daemon with pm2\_home=C:\Users\Muhsina\.pm2

[PM2] PM2 Successfully daemonized

[PM2] Starting E:\keep-alive-script\keepAlive.js in fork\_mode (1 instance)

[PM2] Done.

┌────┬────────────────────┬──────────┬──────┬───────────┬──────────┬──────────┐

│ id │ name │ mode │ ↺ │ status │ cpu │ memory │

├────┼────────────────────┼──────────┼──────┼───────────┼──────────┼──────────┤

│ 0 │ render-keepalive │ fork │ 0 │ online │ 0% │ 45.5mb │

└────┴────────────────────┴──────────┴──────┴───────────┴──────────┴──────────┘

**PS E:\keep-alive-script> pm2 save**

[PM2] Saving current process list...

[PM2] Successfully saved in C:\Users\Muhsina\.pm2\dump.pm2

**PS E:\keep-alive-script> pm2 list**

┌────┬────────────────────┬──────────┬──────┬───────────┬──────────┬──────────┐

│ id │ name │ mode │ ↺ │ status │ cpu │ memory │

├────┼────────────────────┼──────────┼──────┼───────────┼──────────┼──────────┤

│ 0 │ render-keepalive │ fork │ 0 │ online │ 0% │ 49.7mb │

└────┴────────────────────┴──────────┴──────┴───────────┴──────────┴──────────┘

PS E:\keep-alive-script>

**✅ What you can do instead (Windows setup)**

**Option 1 — Easiest: Use PM2 normally and restart it when needed**

You can safely **skip the pm2 startup step**.

Just keep your script running with:

pm2 start keepAlive.js --name keep-alive

Then save your PM2 process list:

pm2 save

To see it running:

pm2 list

If you restart your PC, just open PowerShell and run:

pm2 resurrect

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

Perfect 🎯

This log confirms your **main Render deployment (node\_project-11)** is working **exactly as intended** ✅

Here’s the breakdown 👇

**✅ Your Active Deployment**

**🔹 Service name:** node\_project-11  
**🔹 URL:** <https://node-project-11.onrender.com>  
**🔹 Status:** ✅ Build successful and deployed  
**🔹 Database:** PostgreSQL connected successfully  
**🔹 Port:** 3001  
**🔹 Commit:** b0dc544 (from master branch)  
**🔹 Build command:** npm install && npm run build  
**🔹 Start command:** npm start → runs node dist/server.js

**🟢 What This Means**

* Your **Node.js backend** is **live and running** on Render.
* The app successfully connected to your **PostgreSQL database**.
* Render correctly detected your running server (port 3001).
* You can now access it anytime using:  
  👉 **https://node-project-11.onrender.com**

**⚙️ Next Steps**

**1️⃣ Keep this as your main project**

This one (node\_project-11) is your **production-ready** and stable version.  
You can safely delete all other node\_project-\* duplicates (like node\_project-1, -2, -4, etc.) from Render.

**2️⃣ Test it live**

Try opening your API or frontend endpoint in your browser:

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

or

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

You should see a JSON response or homepage — confirming your app is live.

**3️⃣ Keep-alive script**

You already set up your keepAlive.js script locally with PM2.  
As long as that script keeps running, your Render app will be pinged every 4 minutes and **stay awake**, avoiding the “50-second delay” caused by Render’s free plan auto-sleep.

**4️⃣ Optional: Auto-start PM2 (if you want)**

If you want your PC to start the ping script automatically on boot:

* Create a Windows **Task Scheduler** job
* Command to run:
* pm2 resurrect

This brings back your previous PM2 processes (like render-keepalive).

Would you like me to show you **exactly how to set up Task Scheduler** so your keep-alive script starts automatically whenever your PC boots?

**You said:**

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…”.