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

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

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

dashboard link  
  
Go to your Render Dashboard

New web service

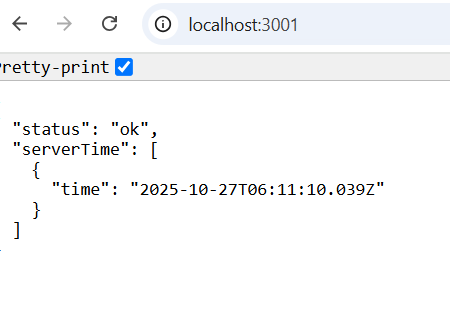
<https://dashboard.render.com/web/new>

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

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://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:

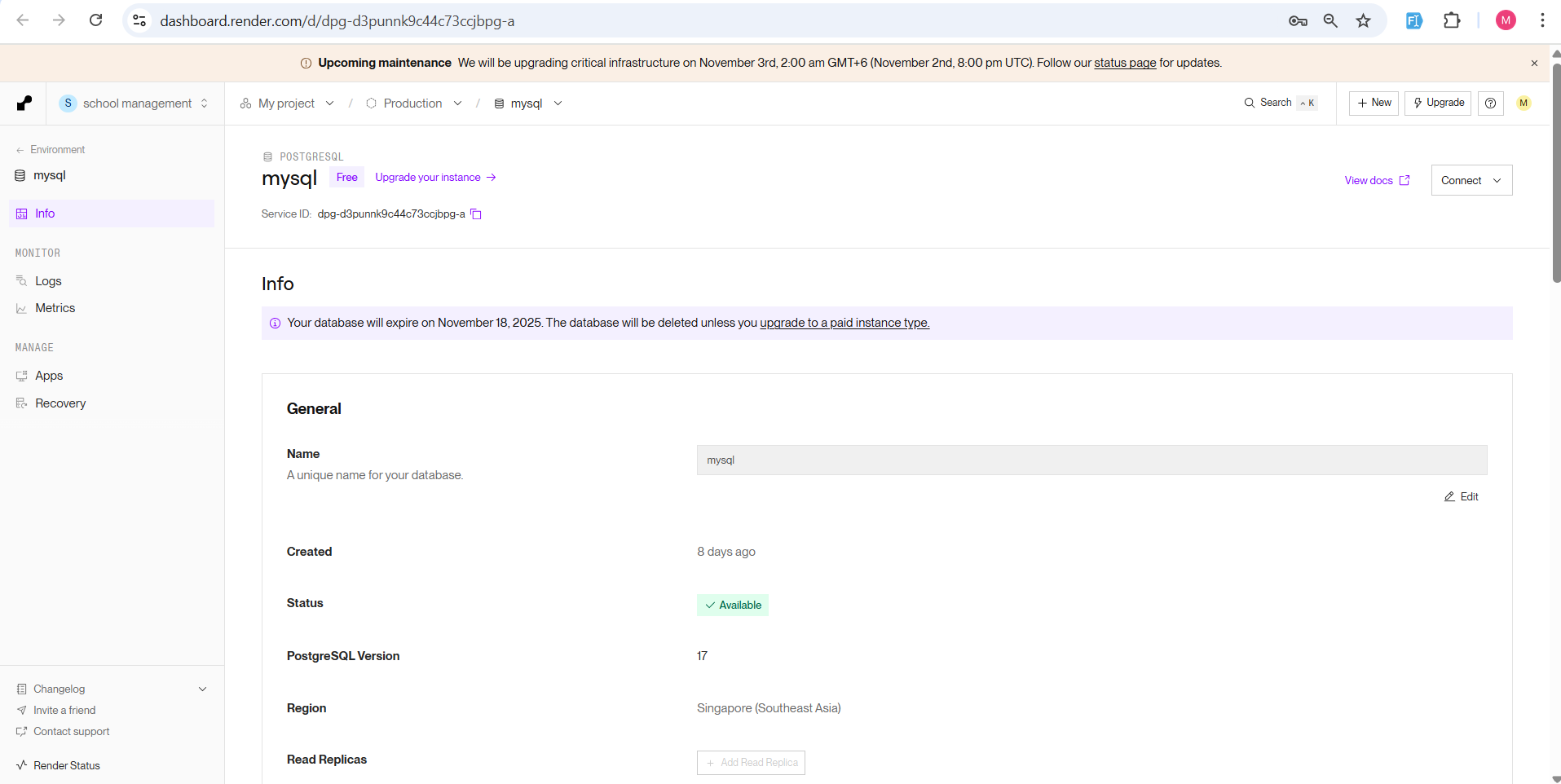
🚀 Server is running on port 3001

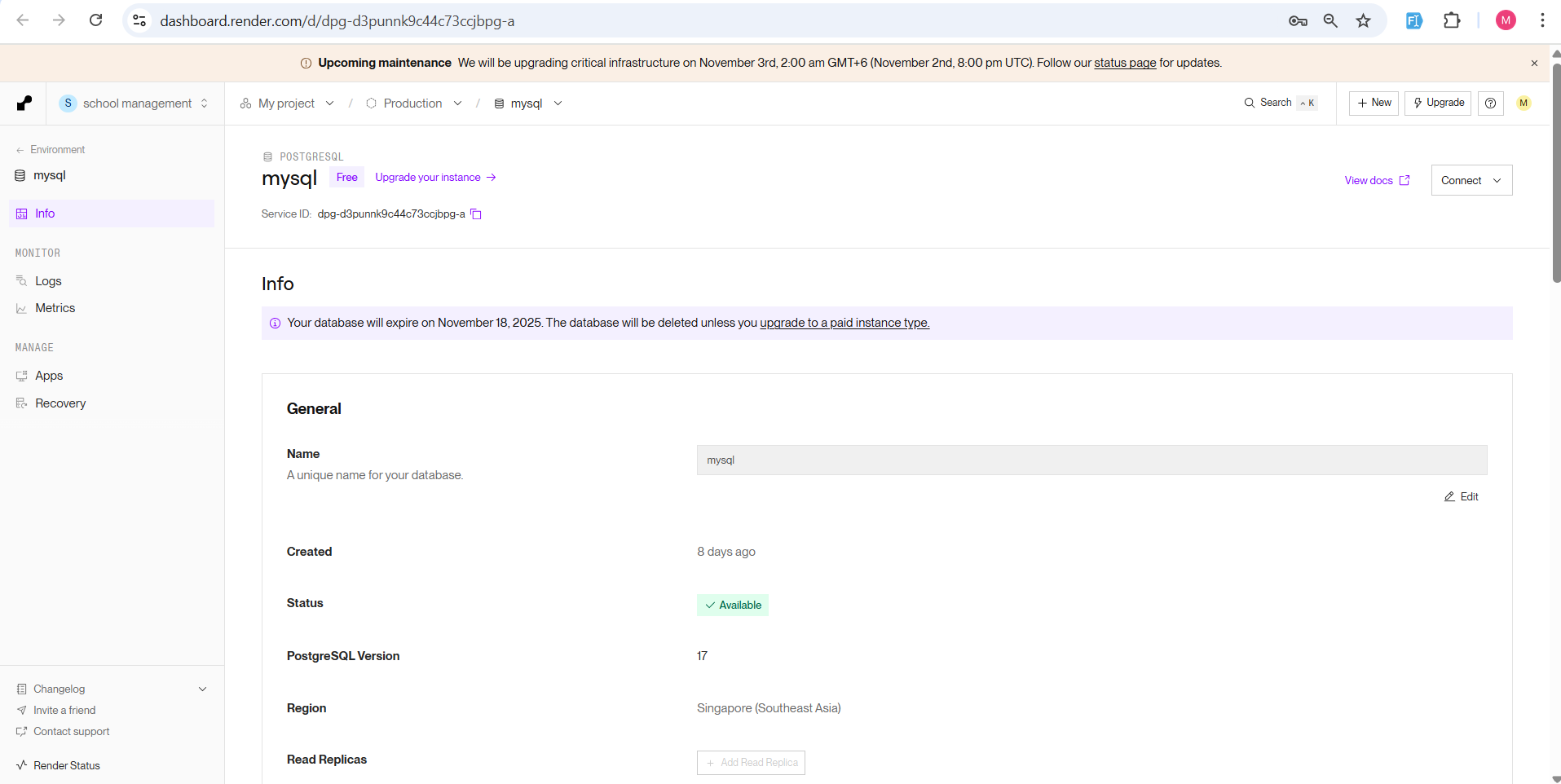
✅ PostgreSQL connected successfully

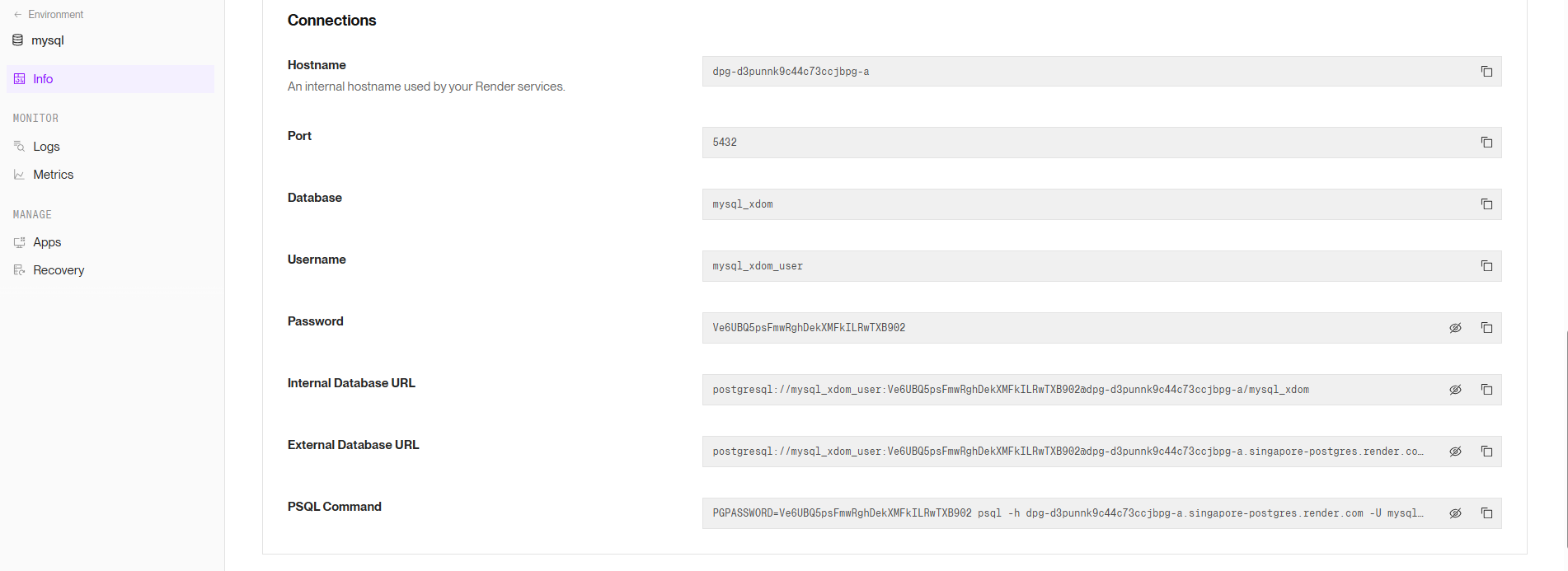
If you want, I can write a **ready-to-deploy .env + database.ts for Render** that will work out-of-the-box with your PostgreSQL credentials. This will eliminate all connection errors.

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.

**⚙️ 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