🚀 Auction Web

Next.js + Express + Node.js + PostgreSQL (PERN Stack - Monorepo)

Auction Web is a **fullstack monorepo** built with:

- **TypeScript**
- **Next.js** – Frontend
- **Node.js + Express** – Backend API
- **PostgreSQL** – Database
- **npm workspaces** – monorepo management

📁 Folder Structure

```
project-root/
|  # Next.js frontend (client + admin folder)
├── client/
│    ├── app/
│    ├── components/
│    ├── config/
│    ├── hooks/
│    ├── lib/
│    ├── public/
│    ├── services/
│    ├── shared/
│    ├── store/
│    ├── types/
│    ├── utils/
│    ├── next.config.ts
│    ├── postcss.config.mjs
│    ├── tsconfig.json
│    ├── .gitignore
│    ├── .env.local       # ENV frontend
│    └── package.json
│
├── admin/
│    ├── app/
│    ├── components/
│    ├── config/
│    ├── hooks/
│    ├── lib/
│    ├── public/
│    ├── services/
│    ├── store/
│    ├── types/
│    ├── utils/
│    ├── next.config.ts
│    ├── postcss.config.mjs
│    ├── tsconfig.json
│    ├── .gitignore
│    └── package.json
│
|  # Express backend
├── server/
```

```
│   ├── src/
│   │   ├── config/
│   │   ├── controllers/
│   │   ├── cron/
│   │   ├── factories/
│   │   ├── middlewares/
│   │   ├── routes/
│   │   ├── services/
│   │   ├── types/
│   │   ├── utils/
│   │   └── server.ts
│   ├── .env              # ENV backend
│   ├── nodemon.json
│   ├── .gitignore
│   ├── tsconfig.json
│   └── package.json
│
│  # Shared types & utils
├── shared/
│   ├── src/
│   │   ├── types/
│   │   ├── utils/
│   │   └── api/
│   ├── package.json
│   └── tsconfig.json
│
├── .gitignore
├── README.MD
├── package.json          # Root scripts (workspaces)
└── tsconfig.json
```

⚙️ Requirements

```
Node.js >= 18

npm >= 9

PostgreSQL >= 14

(Suggestion) pgAdmin 4
```

⚙️ Setup Instructions

1️⃣ Clone the repository

```
git clone https://github.com/Donavfulish/AutionWeb_FinalProject_PTUDW.git
cd AutionWeb_FinalProject_PTUDW
```

2️⃣ Install dependencies

At the root folder, run command:

```
npm install
```

Npm workspaces will install dependencies automatically for: `client`, `admin`, `server`, and `shared`.

🧱 Database Setup (PostgreSQL)

1️⃣: Use Neon Database (already setup)

Backend is already setup for Neon PostgreSQL.

```
DATABASE_URL="postgresql://neondb_owner:npg_im2UE6JSAIKP@ep-green-shape-a1pc3qjd-pooler.ap
-southeast-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require"
```

2️⃣: Create database from db.zip

Step 1: Unzip script in file db.zip

Step 2: Create new database using pgAdmin

1. Open pgAdmin
2. Right-click on Databases → Create → Database
3. Database name: auction_db
4. Owner: postgres

Step 3: Import data

1. Right-click database auction_db
2. Choose Restore
3. File: choose .sql file in db.zip
4. Format: Custom or tar
5. Restore

Step 4: Setup DATABASE_URL for backend

```
DATABASE_URL="postgresqlpostgresql://postgres:your_password@localhost:5432/auction_db"
```

⚙️ Environment Variables Setup

1️⃣ : server/.env

```
PORT=8080

# Using neon database or pgadmin follow the above instruction
DATABASE_URL=postgresql://neondb_owner:npg_im2UE6JSAIKP@ep-green-shape-a1pc3qjd-pooler.ap
-southeast-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require

ACCESS_TOKEN_SECRET=f02ef7ad90ba19273faa0385663268c100c3af2f4b8c8796f062e536db41048172e7e6ed
3e700393378a68d54db98ad53badf6646859024d7d304681c25e7b5c

R2_ACCOUNT_ID=cb5953b1e7c78dc509ddcff170b55b6e
R2_ACCESS_KEY_ID=afd28e938270629af69629789400de73
R2_SECRET_ACCESS_KEY=fd662d22b2c77522479df5c7cc24bc8880e04ebcc4a620786027a4401ec64afa
R2_BUCKET_NAME=ptudw-auction-images

SMTP_USER=flazerfa123@gmail.com

SMTP_PASS=eyfw qwju lswj qlfs

RECAPTCHA_SECRET_KEY=6LcZkjcsAAAAAMcv-XwCkN-EvZtnbcdBHrexrLcC
```

2️⃣ : client/.env.local

```
NEXT_PUBLIC_RECAPTCHA_SITE_KEY=6LcZkjcsAAAAAN7qnLO1BzlPcN2KdP2smMAemRPP
```

🚀 Development

Run both **client** and **server** concurrently from the root:

```
npm run dev
```

What happens

- 🖥️ Server runs at http://localhost:8080
- 🌐 Client runs at http://localhost:3000
- 🌐 Admin runs at http://localhost:3001

🔧 Scripts Reference

Command	Description
`npm run dev`	Run both client & server concurrently
`npm run dev:client`	Run only the Next.js app
`npm run dev:server`	Run only the Express backend
`npm run build --workspace client`	Build frontend
`npm run build --workspace server`	Build backend
`npm run build --workspace shared`	Build shared library