Perfect 👏 — that's a big milestone!
Here's a clear summary of everything we've built and configured so far, so you can safely **document and save your progress** 👇

✅ KidToKid Project — Progress Summary

🎯 Project Goal

A free-friendly marketplace where parents can buy, sell, or give away children's items (clothes, toys, gear) locally — with an optional deliverer workflow.

🧱 Architecture (current stage)

Layer	Stack	Hosting (now / later)
Frontend (Buyer/Admin)	React + Vite + Tailwind (PWA ready)	Local dev → Azure Static Web Apps (Free tier)
Backend (API)	Azure Functions (Python v2)	Local dev → Azure Functions (Consumption Plan)
Database	SQL Server Express (local)	Local dev → Azure SQL Database
Storage (images)	planned: Azure Blob	local for now
Auth (planned)	Microsoft Entra ID (B2C)	not yet implemented
Deliverer App	planned React PWA / mini app	not yet implemented

---

⚙️ □ Tools & Environment

- **VS Code** for development
- **Python 3.11** (Azure Functions Core Tools compatible)
- **Node.js + npm** for frontend
- **SSMS (SQL Server Management Studio)** for database
- **Postman** for API testing

📂 Folder Structure

```
KidToKid/
├── apps/
│   ├── web/          → React + Vite + Tailwind frontend
│   └── api/          → Azure Functions Python backend
├── db/               → SQL scripts (schema + seeds)
├── infra/            → environment files, settings
└── README.md         → documentation (to create next)
```

💻 Frontend

Framework: React + Vite + TypeScript + TailwindCSS + vite-plugin-pwa

✅ Set up successfully:

- Tailwind v4 manual config
- PWA manifest ready
- Home page lists sample listings
- API calls handled through `src/lib/api.ts`

Next planned: add navigation, basket, and checkout pages (already partially done).

⚙️ Backend (Azure Functions – Python v2)

Environment:

- `Python 3.11` (venv in `apps/api/.venv`)
- Dependencies: `azure-functions`, `pyodbc`
- `local.settings.json` holds DB connection and buyer ID.

✅ Routes currently working:

Route	Method	Description
`/api/ping`	GET	Health check
`/api/listings`	GET	Fetch active listings
`/api/basket`	GET	Get buyer basket
`/api/basket/{listingId}`	POST	Add item to basket

Route	Method	Description
`/api/basket/{listingId}`	DELETE	Remove item
`/api/orders/confirm`	POST	Confirm order + create record

🧩 Database (SQL Express)

Database: `KidToKid`

✅ Created tables:

- `dbo.listing` — listings catalog
- `dbo.listing_image` — image links (future use)
- `dbo.app_user` — simple buyer table (with one seeded buyer)
- `dbo.basket_item` — items in buyer basket
- `dbo.[order]` — order headers
- `dbo.order_item` — items per order

✅ Seed data added:

- 4 example listings (clothes, toys, stroller, boots)
- 1 developer buyer (`dev@kidtokid.local`)

🧪 Postman Testing

Tested successfully:

- `GET /api/listings` → returns active items
- `POST /api/basket/<listingId>` → adds item
- `GET /api/basket` → shows items in basket
- `DELETE /api/basket/<listingId>` → removes item
- `POST /api/orders/confirm` → creates order & clears basket

🚀 What's working end-to-end

- Backend fully functional with SQL Express
- Frontend connects to backend (`/api/listings`)
- Full basket + checkout flow operational
- All services running locally, free-tier ready

📜 Next Steps (Step 5)

1. **Add Favorites & Alerts (saved searches).**
2. **Add Deliverer module** (deliveries table + required comment + status updates).
3. **Add authentication (Microsoft Entra ID B2C)**.
4. **Upload images to Azure Blob Storage.**
5. **Deploy:**
    - Frontend → Azure Static Web Apps
    - Backend → Azure Functions (Consumption)
    - Database → Azure SQL
    - Storage → Azure Blob

Would you like me to create a clean **`README.md` file** summarizing this progress (ready for GitHub or backup)?