Short answer: **yes**—you can code the whole thing in VS Code and ship to **Android + iOS**. You've got three good paths:

1) Fastest/Free-est now → PWA (Installable Web App)

- Keep your React + Vite app and make it a **Progressive Web App**.
- Users "Install to Home Screen" on Android/iOS (iOS 16.4+ supports web push).
- **What you get:** offline caching, home-screen icon, background sync (limited on iOS), **web push** (works on Android; on iOS only after install).
- **Great for MVP + deliverer app** (low friction, no store review).
- **Extras:** OneSignal (free) for **Web Push**, service worker via `vite-plugin-pwa`.

2) Share your React code → React Native + Expo

- Build truly native apps while coding in **VS Code** and **TypeScript**.
- Reuse your API client and business logic; rebuild screens with React Native UI.
- **Dev requirements:**
    - Android: Android Studio **only for SDK/emulator** (coding still in VS Code).
    - iOS: a Mac + Xcode to build locally **or** use **Expo EAS Cloud Build** (no Mac, but Apple dev account needed to publish).
- **Notifications:** `expo-notifications` → FCM (Android) + APNs (iOS) or OneSignal.
- **Auth:** Entra ID (B2C) via `expo-auth-session` (PKCE) or `msal-react-native`.
- **Media/Geo:** `expo-image-picker`, `expo-camera`, `expo-location`, `react-native-maps`.

3) Wrap your existing web app → Capacitor (Ionic)

- Keep your current React app; add **native wrappers** for Android/iOS.
- Access device APIs via Capacitor plugins (Camera, Geolocation, Push).
- **Dev requirements:** same as above (Android Studio/Xcode for building).
- **Good when** you want one codebase with minimal re-write.

What I recommend (for *free-friendly* + quickest path)

Phase 1 (this week): PWA

- Convert your buyer app + deliverer app to **PWA**.
- Add: web manifest, service worker (Vite plugin), **OneSignal Web Push**.
- Deploy on **Azure Static Web Apps (Free)**.
- ☑ No app-store fees; works on Android/iOS; great for pilots.

Phase 2 (when you want store presence): Expo

- Spin up a **React Native (Expo)** app for **buyer** and a second lightweight one for **deliverer**.
- Keep a **shared TypeScript package** for domain models + API client used by both web + mobile.

Monorepo sketch:

```
kidtokid/
  apps/
    web/            # React + Vite + PWA (Azure SWA Free)
    mobile/         # Expo React Native (buyer)
    deliverer/      # Expo React Native (deliverer)  OR keep as PWA
  packages/
    core/           # shared types, validation, api client (axios/fetch)
```

Store logistics (so you’re not surprised)

- **Android**: you can generate `apk/aab` for free; Play Console one-time $25.
- **iOS**: needs Apple Developer Program ($99/yr) to publish.
- You can **build and test** on devices for free; publishing is where costs appear.
- On **Windows**, you can't run the iOS Simulator; use a real iPhone (Expo Go) or **EAS Cloud Build**.

Feature mapping to mobile

- **Login (B2C)**:
  - PWA: MSAL.js (SPA)
  - Expo: `expo-auth-session` (Authorization Code + PKCE) with Entra B2C
- **Photos**:
  - PWA: `<input type="file">` + Blob Storage SAS upload
  - Expo/Capacitor: Camera plugin, compress then upload to Blob
- **Geolocation / distance search**:
  - PWA: `navigator.geolocation` (permission dependent)
  - Expo/Capacitor: Location plugin (more reliable)
- **Push notifications**:
  - PWA: Web Push via OneSignal (Android; iOS works when installed as PWA)
  - Expo/Capacitor: FCM (Android) + APNs (iOS) via OneSignal or native libs
- **Deliverer status updates (comment required)**: both PWA and RN handle this fine.

Concrete next steps (copy/paste plan)

1. **Make web app a PWA**

    - Add `vite-plugin-pwa`, web manifest (icons, name), and a basic service worker.
    - Integrate **OneSignal Web Push** (free) for delivery notifications.
2. **Ship to Azure Static Web Apps (Free)**
    - Connect GitHub → auto deploy on push.
3. **Prototype native**
    - `npx create-expo-app apps/mobile`
    - Add 2–3 key screens (Login, Listings, Basket) using your shared API client.
    - Test on Android device with Expo Go; on iPhone with Expo Go (no Mac needed).
4. **Decide on publishing**
    - Android: create Play Console when ready.
    - iOS: enroll Apple Developer when you want store listing (or stay PWA).

If you want, I'll generate:

- a ready PWA setup (manifest + service worker + OneSignal hooks), and/or
- an Expo starter wired to your current API + Entra B2C (auth flow + push).