Perfect—clear scope. Here’s a tight, build-ready blueprint for v1.

Scope (MVP)

- **No chat.**
- **Login** (Entra External ID/B2C).
- **Browse & Search** listings (category, size, condition, distance).
- **Post Listing** (title, price/free, photos, condition, pickup location).
- **Basket**: add/remove items.
- **Confirm Purchase** (no online payments for v1).
- **Favorites & Alerts** (watchlist + saved searches).
- **Basic Moderation** (report listing/user; admin remove/ban).
- **Delivery flow**: buyer confirms → create delivery task → **notify deliverer app** → deliverer updates status + comment (required).

Azure-friendly Architecture (free/near-free)

- **Frontend (buyer/admin)**: React + Vite + Tailwind → **Azure Static Web Apps (Free)**
- **Deliverer PWA**: tiny React app in `/deliverer` → same SWA (separate route + role)
- **API**: **Azure Functions (Python)** HTTP + **Queue Trigger**
- **DB (dev)**: **SQL Server Express** (SSMS). Later: Azure SQL.
- **Storage**: **Azure Blob** for images; **Azure Queue Storage** to notify deliverers.
- **Auth**: **Entra External ID (B2C)** roles: `buyer`, `deliverer`, `admin`
- **Monitoring**: Application Insights

```
SWA (buyer/admin) ─┐           ┌─> Azure Blob (images)
SWA (deliverer) ───┼─> Functions API ──> SQL Express/Azure SQL
```

```
                   └─> Azure Queue ("deliveries") → Queue-trigger Function →
push to DB
```

Data Model (tables)

```
-- Users & roles
CREATE TABLE app_user (
  user_id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  email NVARCHAR(255) UNIQUE NOT NULL,
  display_name NVARCHAR(120),
  role NVARCHAR(20) NOT NULL CHECK (role IN ('buyer','deliverer','admin')),
  created_at DATETIME2 DEFAULT SYSUTCDATETIME(),
  is_banned BIT DEFAULT 0
);

-- Listings
CREATE TABLE listing (
  listing_id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  seller_id UNIQUEIDENTIFIER NOT NULL REFERENCES app_user(user_id),
  title NVARCHAR(200) NOT NULL,
  description NVARCHAR(MAX),
  category NVARCHAR(50),      -- clothes, toys, gear...
  size NVARCHAR(30),
  condition NVARCHAR(20),     -- new, like-new, good, fair
  price_cents INT NULL,       -- NULL or 0 for "free"
  is_free AS CASE WHEN price_cents IS NULL OR price_cents=0 THEN 1 ELSE 0 END
PERSISTED,
  latitude DECIMAL(9,6), longitude DECIMAL(9,6),
  city NVARCHAR(80), country NVARCHAR(80),
  is_active BIT DEFAULT 1,
  created_at DATETIME2 DEFAULT SYSUTCDATETIME()
);

CREATE TABLE listing_image (
  image_id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  listing_id UNIQUEIDENTIFIER NOT NULL REFERENCES listing(listing_id),
  blob_url NVARCHAR(500) NOT NULL,
  sort_order INT DEFAULT 0
);

-- Favorites & Alerts
CREATE TABLE favorite (
  user_id UNIQUEIDENTIFIER REFERENCES app_user(user_id),
  listing_id UNIQUEIDENTIFIER REFERENCES listing(listing_id),
  created_at DATETIME2 DEFAULT SYSUTCDATETIME(),
  PRIMARY KEY (user_id, listing_id)
);

CREATE TABLE saved_search (
  saved_search_id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  user_id UNIQUEIDENTIFIER REFERENCES app_user(user_id),
  query_json NVARCHAR(MAX) NOT NULL,  -- {category, size, condition, radiusKm,
geo}
  created_at DATETIME2 DEFAULT SYSUTCDATETIME(),
  is_active BIT DEFAULT 1
);

-- Basket + Orders
CREATE TABLE basket_item (
  user_id UNIQUEIDENTIFIER REFERENCES app_user(user_id),
  listing_id UNIQUEIDENTIFIER REFERENCES listing(listing_id),
  qty INT NOT NULL CHECK (qty=1),      -- one-off items
  PRIMARY KEY (user_id, listing_id)
);

CREATE TABLE [order] (
  order_id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  buyer_id UNIQUEIDENTIFIER NOT NULL REFERENCES app_user(user_id),
  total_cents INT NOT NULL DEFAULT 0,
  status NVARCHAR(20) NOT NULL CHECK (status IN
('created','confirmed','canceled')),
  created_at DATETIME2 DEFAULT SYSUTCDATETIME()
);

CREATE TABLE order_item (
  order_id UNIQUEIDENTIFIER REFERENCES [order](order_id),
  listing_id UNIQUEIDENTIFIER REFERENCES listing(listing_id),
  price_cents INT NOT NULL,
  PRIMARY KEY (order_id, listing_id)
);

-- Delivery
CREATE TABLE delivery (
  delivery_id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  order_id UNIQUEIDENTIFIER NOT NULL REFERENCES [order](order_id),
  deliverer_id UNIQUEIDENTIFIER NULL REFERENCES app_user(user_id),
  status NVARCHAR(20) NOT NULL
    CHECK (status IN
('queued','assigned','out_for_delivery','delivered','canceled','failed')),
  pickup_address NVARCHAR(255),
  dropoff_address NVARCHAR(255),
  notes NVARCHAR(400),
  created_at DATETIME2 DEFAULT SYSUTCDATETIME()
);

CREATE TABLE delivery_status_history (
  history_id BIGINT IDENTITY PRIMARY KEY,
  delivery_id UNIQUEIDENTIFIER REFERENCES delivery(delivery_id),
  status NVARCHAR(20),
  comment NVARCHAR(400) NOT NULL,  -- deliverer must comment
  changed_by UNIQUEIDENTIFIER NULL REFERENCES app_user(user_id),
  changed_at DATETIME2 DEFAULT SYSUTCDATETIME()
);

-- Reports / Moderation
CREATE TABLE report (
  report_id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  reporter_id UNIQUEIDENTIFIER REFERENCES app_user(user_id),
  target_type NVARCHAR(20) CHECK (target_type IN ('listing','user')),
  target_id UNIQUEIDENTIFIER NOT NULL,
  reason NVARCHAR(200),
  created_at DATETIME2 DEFAULT SYSUTCDATETIME(),
  resolved BIT DEFAULT 0,
  resolved_by UNIQUEIDENTIFIER NULL REFERENCES app_user(user_id),
  resolved_at DATETIME2 NULL
);
```

Delivery Status (required comment on each change)

```
queued → assigned → out_for_delivery → delivered | failed | canceled
```

API (Azure Functions) — key endpoints

```
GET  /api/me                                      # whoami (role, email)
GET  /api/listings?category=&size=&cond=&lat=&lng=&radiusKm=
POST /api/listings                                # create (uploads pre-signed to
Blob)
GET  /api/listings/{id}
POST /api/favorites/{listingId}
DELETE /api/favorites/{listingId}
GET  /api/favorites

GET  /api/basket                                  # list items
POST /api/basket/{listingId}                      # add
DELETE /api/basket/{listingId}                    # remove

POST /api/orders/confirm                          # create order from basket
GET  /api/orders/{orderId}

-- Delivery + queue
GET  /api/deliveries?mine=1                       # deliverer view
POST /api/deliveries/{id}/status                  # body: {status, comment}
(comment required)

POST /api/reports                                  # moderation
POST /api/admin/listings/{id}/toggle               # activate/deactivate
```

Queue message (notify deliverers)

```
{
  "delivery_id": "GUID",
  "order_id": "GUID",
  "pickup_address": "string",
  "dropoff_address": "string",
  "items": [{"listing_id":"GUID","title":"Shoes 24-36m"}]
}
```

- **On `POST /api/orders/confirm`**:
  1. create order + order_items, mark listings reserved/inactive
  2. create `delivery` with status `queued`
  3. enqueue message to **Azure Queue** `deliveries`
  4. queue-trigger function fans out notifications (email/push) to deliverers

---

Frontend structure

```
/frontend
  src/
    pages/
      Home.tsx              # browse/search
      Listing.tsx
      PostListing.tsx
      Basket.tsx
      Checkout.tsx
      Favorites.tsx
      AdminModeration.tsx
    lib/
      api.ts                # fetch helpers (with MSAL token)
      auth.ts               # MSAL B2C
      storage.ts            # get upload URL, put to Blob
/deliverer
  src/
    pages/
      Dashboard.tsx         # list assigned/queued nearby
      DeliveryDetails.tsx   # update status + comment (required)
```

Required env (samples)

```
# API
SQL_CONN_STR=Server=localhost\SQLEXPRESS;Database=KidToKid;Trusted_Connection=
True;Encrypt=False;
BLOB_CONN_STR=DefaultEndpointsProtocol=...   # or Azurite for dev
QUEUE_CONN_STR=DefaultEndpointsProtocol=...
DELIVERIES_QUEUE_NAME=deliveries

# Auth
```

```
ENTRA_TENANT_ID=...
ENTRA_CLIENT_ID=...
ENTRA_CLIENT_SECRET=...
AUDIENCE=api://kidtokid
```

Core flows

Confirm Purchase

1. Validate basket availability (listings active).
2. Create order + items → set listings inactive.
3. Create delivery (`queued`) with addresses.
4. Enqueue delivery message.
5. Return order + delivery IDs.

Deliverer App

1. `GET /api/deliveries?mine=1` (role=deliverer).
2. Open delivery → update status with required `comment`.
3. History row written on each change.

Moderation

- Users report → admin sees queue → toggle listing active/inactive; mark report resolved.

Next steps (day-1 checklist)

1. Create DB in SQL Express → run the schema above.
2. Scaffold Azure Functions (Python): implement `listings`, `basket`, `orders/confirm`, `deliveries`.
3. Create Storage Account (or Azurite) + Queue `deliveries`.
4. Frontend pages (browse, post, basket, checkout).

5. Deliverer mini-app (Dashboard + Details).
6. Wire B2C auth + roles.
7. Set up SWA (Free) + CI/CD from GitHub.

If you want, I can generate:

- a **seed SQL** (sample users/listings),
- **Function stubs** for the endpoints, and
- a minimal **React page** for buyer + deliverer to start clicking.