Delivery ETA System – Source of Truth

1. Overview

The Delivery ETA system is responsible for showing a dynamic “Delivery in X–Y mins” message based on:

- The customer’s location (lat/lng).
- The set of available branches with their coordinates.
- Global and per-branch configuration that defines:
    - Prep time (kitchen time).
    - Average delivery speed.
    - Service radius.

The main entry point is the backend endpoint:

`GET /api/delivery/estimate-for-location?latitude=<lat>&longitude=<lng>`

The mobile app **MUST NOT** implement its own distance or ETA logic. All decisions come from the backend.

2. Data Models

2.1 Branch Model

- **File**: `src/models/branch.js`
- **Collection**: `branches`

Relevant fields:

- `name` : String
- `location` :
    - `latitude` : Number
    - `longitude` : Number
- `address` : String
- `seller` : ObjectId('Seller')
- `deliveryPartners` : [ObjectId('DeliveryPartner')]
- `isActive` : Boolean
    - **Default**: `true`
    - Used to include/exclude branches from ETA selection.
- `serviceRadiusKm` : Number (optional)
    - Per-branch maximum delivery distance in kilometers.
    - If not set or <= 0, global default radius is used.
- `prepTimeMin` : Number (optional)
    - Per-branch prep time in minutes.
    - If not set or < 0, global default prep time is used.
- `averageSpeedKmph` : Number (optional)
    - Per-branch average delivery speed in km/h.
    - If not set or <= 0, global default speed is used.
- Timestamps (`createdAt` , `updatedAt`).

Admin page:

- AdminJS → Store Management → Branch
- Configured in: `src/config/setup.ts`
- Shows and edits: Lat/long, Is Active, Service Radius, Prep Time, Average Speed, etc.

Usage in ETA:

- Defines the actual branch locations and optional overrides.
- Only branches with:
  - `isActive !== false` and
  - non-null `location.latitude` and `location.longitude`
  - are considered for ETA.

2.2 DeliveryEtaConfig Model

- **File**: `src/models/deliveryEtaConfig.js`
- **Collection**: `deliveryetaconfigs`

Fields:

- `defaultPrepTimeMin` : Number
  - Required; >= 0
  - **Default**: 12
  - Global default prep time (minutes) when branch doesn't override.
- `defaultAverageSpeedKmph` : Number
  - Required; >= 1
  - **Default**: 25
  - Global default speed (km/h) when branch doesn't override.
- `defaultServiceRadiusKm` : Number
  - Required; >= 0.1
  - **Default**: 5
  - Global default service radius (km) when branch doesn't override.
- `isActive` : Boolean
  - Required; **default**: `true`
  - Indexed; only one config is meant to be active.
- `createdBy` : String
  - Required; admin identifier (email/name).
- Timestamps (`createdAt` , `updatedAt`).

Hooks and statics:

- `pre('save')` : If `this.isActive === true` , sets all other documents' `isActive` to `false` . Ensures there is at most one active config.
- `DeliveryEtaConfig.getActiveConfig()` : Returns the document where `isActive === true` , or `null` if none.

Admin page:

- AdminJS → Store Management → DeliveryEtaConfig (exact label may vary).
- Configured in: `src/config/setup.ts` .
- Allows: Viewing all ETA configs, Editing defaults, Selecting which config is active.

Usage in ETA:

- Supplies defaults when a branch does not set its own: `prepTimeMin` , `averageSpeedKmph` , `serviceRadiusKm` .

3. ETA API

3.1 Endpoint

- **Route**: `GET /api/delivery/estimate-for-location`
- **Parameters (query)**:
    - `latitude` – required; decimal string or number, convertible to float.
    - `longitude` – required; decimal string or number, convertible to float.
- **Auth**: Uses the same Fastify `verifyToken` hook as other order endpoints; expects a valid access token.
- **File**: `src/controllers/order/eta.js`
- **Registered in**: `src/routes/order.js`

3.2 Processing Steps

1. **Validate inputs**
    - If `latitude` or `longitude` is Missing, Empty string, or Non-numeric, then respond with `400` .
2. **Load branches and config**
    - **Fetch branches**:

      ```
      Branch.find({
        isActive: { $ne: false },
        'location.latitude': { $ne: null, $exists: true },
        'location.longitude': { $ne: null, $exists: true },
      }).select('name address location serviceRadiusKm prepTimeMin averageSpeedKmph seller');
      ```

    - **Fetch active ETA config**:

      ```
      const etaConfigRaw = await DeliveryEtaConfig.getActiveConfig();
      ```

        - If `etaConfigRaw` exists: Use it as `etaConfig` .
        - If `etaConfigRaw` is `null` :
            - Log a warning: "No active ETA configuration found; using fallback defaults".
            - Use hardcoded fallback defaults: `defaultPrepTimeMin = 12` , `defaultAverageSpeedKmph = 25` , `defaultServiceRadiusKm = 5` .
3. **Check branches**
    - If no branches returned: Respond with `404 NO_BRANCHES` .
4. **Filter branches by service radius**
    - For each branch:
        - Skip if lat/lng are invalid.
        - Compute distance using Haversine formula (Earth radius R = 6371 km).
        - `distanceKm = haversine(userLat, userLng, branchLat, branchLng)` .
        - Determine effective radius:

```
const radius = typeof branch.serviceRadiusKm === 'number' &&
branch.serviceRadiusKm > 0
  ? branch.serviceRadiusKm
  : etaConfig.defaultServiceRadiusKm;
```

    - If `distanceKm > radius`, this branch is ignored.
  - Collect all in-radius branches into `candidates`.

5. **Pick nearest branch**

   - If `candidates` is empty: Respond with `404 OUT_OF_COVERAGE`.
   - Sort `candidates` by `distanceKm` ascending.
   - Select `best = candidates[0]`.

6. **Determine effective prep time and speed**

   - **Prep time**:

```
const prepTimeMin = typeof best.branch.prepTimeMin === 'number' &&
best.branch.prepTimeMin >= 0
  ? best.branch.prepTimeMin
  : etaConfig.defaultPrepTimeMin;
```

   - **Average speed**:

```
const avgSpeed = typeof best.branch.averageSpeedKmph === 'number' &&
best.branch.averageSpeedKmph > 0
  ? best.branch.averageSpeedKmph
  : etaConfig.defaultAverageSpeedKmph;
```

7. **Compute ETA**

   - **Travel time**: `const travelTimeMin = (best.distanceKm / avgSpeed) * 60;`
   - **Total ETA (numeric)**: `const etaMinutes = Math.round(prepTimeMin + travelTimeMin);`
   - **Textual ETA (range)**:

```
const lower = Math.max(etaMinutes - 5, prepTimeMin);
const upper = etaMinutes + 5;
const etaText = (lower === upper) ? `${upper} mins` : `${lower}-${upper}
mins`;
```

8. **Return success response**

   - If everything above succeeds:

```
{
  "success": true,
  "etaMinutes": 18,
  "etaText": "15-20 mins",
  "branch": {
    "id": "BRANCH_ID",
    "name": "Test 2",
    "address": "Branch address…",
    "distanceKm": 2.35,
    "sellerId": "SELLER_ID"
  },
  "meta": {
    "prepTimeMin": 12,
    "averageSpeedKmph": 25,
    "serviceRadiusKm": 5
  }
}
```

9. **Unexpected errors**
   - If any unexpected error occurs: Log the error. Respond with `500 ETA_ERROR` .

4. Error Codes and When They Occur

All error responses are JSON with `success: false` and a `reason` .

4.1 400 – INVALID_LOCATION

- **When**: `latitude` or `longitude` is missing, empty, or not a valid number.
- **Response example**:

```
{
  "success": false,
  "reason": "INVALID_LOCATION",
  "message": "Latitude and longitude are required"
}
```

4.2 404 – NO_BRANCHES

- **When**: There are no active branches with valid lat/lng in the database.
- **Response**:

```
{
  "success": false,
  "reason": "NO_BRANCHES",
  "message": "No active branches available"
}
```

4.3 404 – OUT_OF_COVERAGE

- **When**: There are active branches, but all of them are:
  - Outside their effective radius (per-branch `serviceRadiusKm` or global default), or
  - Skipped due to invalid coordinates.
- **Response**:

```
{
  "success": false,
  "reason": "OUT_OF_COVERAGE",
  "message": "Delivery not available at your location"
}
```

4.4 500 – ETA_ERROR

- **When**: Any unexpected error occurs during processing (e.g., DB error, code bug).
- **Response**:

  ```
  {
    "success": false,
    "reason": "ETA_ERROR",
    "message": "Failed to estimate delivery time"
  }
  ```

- **Note**: The previous error `ETA_CONFIG_MISSING` is now handled internally via fallback defaults.

5. Override Rules (Branch vs DeliveryEtaConfig)

5.1 Service Radius

For each branch:

- If `branch.serviceRadiusKm` is A number **AND** > 0 → use this as radius.
- Else: Use `etaConfig.defaultServiceRadiusKm` .

5.2 Prep Time

- If `branch.prepTimeMin` is A number **AND** >= 0 → use this as prep time.
- Else: Use `etaConfig.defaultPrepTimeMin` .

5.3 Average Speed

- If `branch.averageSpeedKmph` is A number **AND** > 0 → use this as speed.
- Else: Use `etaConfig.defaultAverageSpeedKmph` .

5.4 Impact

- **Global DeliveryEtaConfig** provides base defaults for all branches.
- **Individual branches** can override any of: `serviceRadiusKm` , `prepTimeMin` , `averageSpeedKmph` .
- ETA always uses branch-specific values when present, falling back to global values when not.

6. Admin Pages and Their Roles

6.1 Branch (Store Management → Branch)

- **Who uses it**: Operations / Admin.
- **What it controls**:
  - Whether a branch participates in ETA (`isActive`).
  - Where it is on the map (`location.latitude` , `location.longitude`).
  - How far it can deliver (`serviceRadiusKm`).
  - How fast it is (`prepTimeMin` , `averageSpeedKmph`).
- **Impact**:
  - Controls which branches can serve a given location.

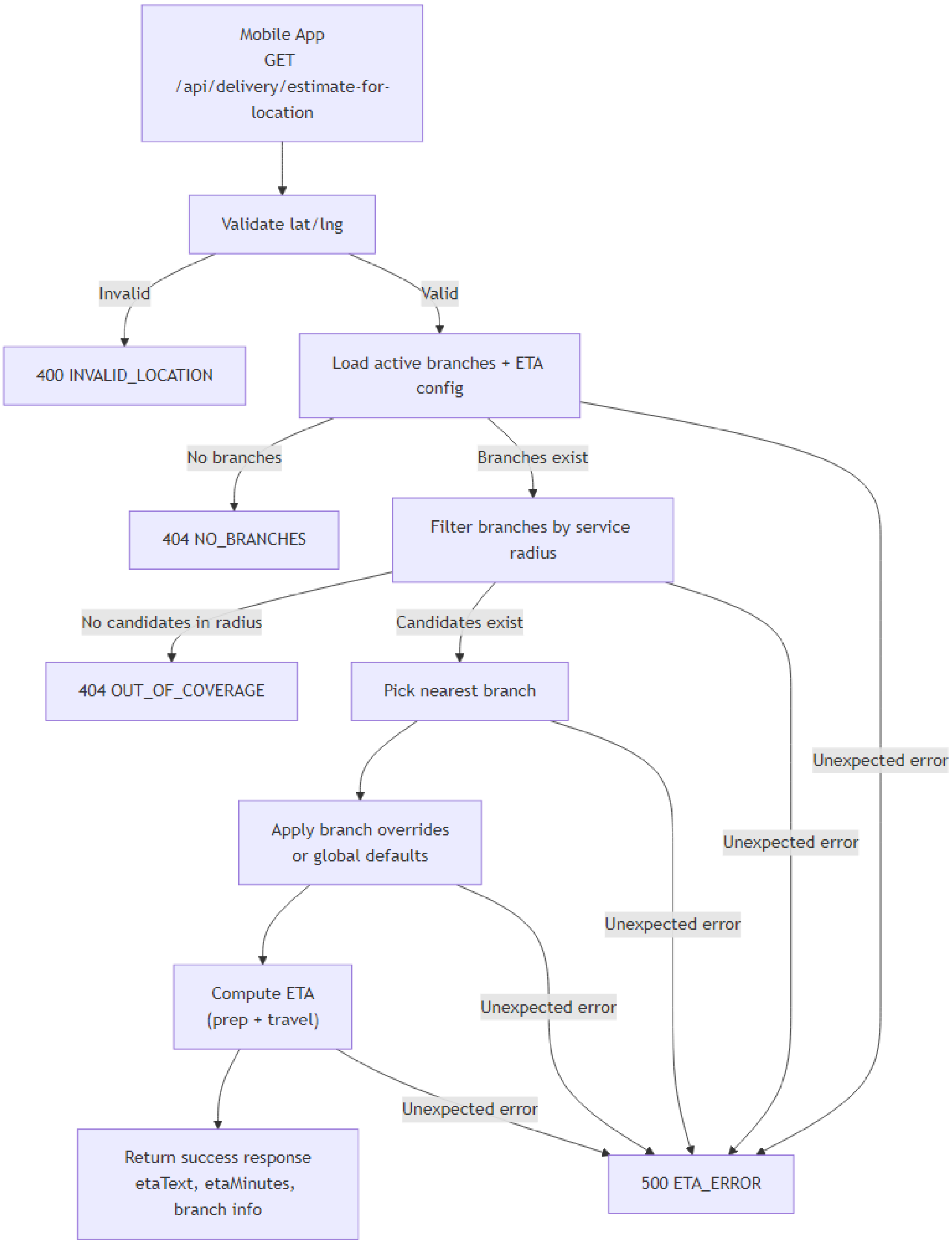
    - Controls per-branch override of ETA dynamics.

6.2 DeliveryEtaConfig (Store Management → DeliveryEtaConfig)

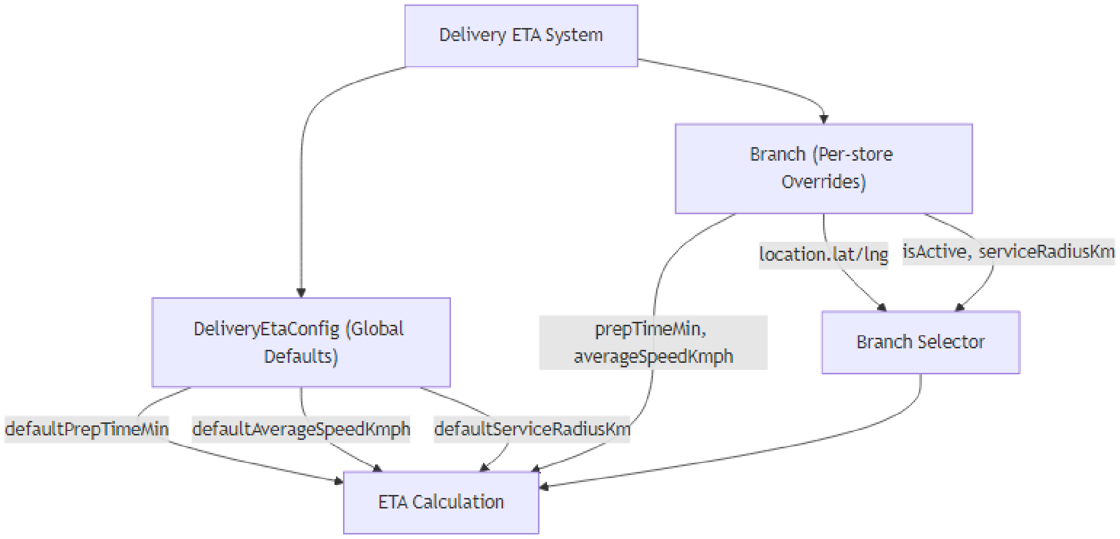
- **Who uses it**: Operations / Admin, higher-level.
- **What it controls**:
    - Default prep time for all branches.
    - Default speed for all branches.
    - Default service radius.
- **Impact**:
    - Acts as the fallback configuration for branches without overrides.
    - Only one active config at a time.

7. Mermaid Diagrams

7.1 High-Level Flow



7.2 Configuration Hierarchy



8. Summary

- **DeliveryEtaConfig** defines global defaults: prep time, speed, radius.
- **Branch** defines actual locations and optional overrides.
- **GET /api/delivery/estimate-for-location**:
    1. Validates location.
    2. Finds all active branches with coordinates.
    3. Filters by effective radius (branch override → global).
    4. Picks the nearest branch.
    5. Computes ETA using branch overrides or global defaults.
    6. Returns a single, simple response for the mobile app.
- Errors are clearly defined with reason codes (`INVALID_LOCATION` , `NO_BRANCHES` , `OUT_OF_COVERAGE` , `ETA_ERROR`).