StoreFlow — A1.4 Shipping Engine & Fulfilment Workflows (Part 1)

1. Shipping Engine – Conceptual Overview

The StoreFlow shipping engine is responsible for transforming:

• Cart contents

• Destination address (country, state, postcode)

• Store shipping configuration

into:

• A list of valid shipping options (name, price, meta)

• A selected shipping method and cost stored on the order

Conceptual properties:

• Deterministic: same inputs $\rightarrow$ same outputs.

• Predictable: merchants can reason about final prices.

• Pluggable: internal logic is structured so that carrier integrations can be added later.

• Fast: optimized to serve in real time at checkout, with caching where appropriate.

2. Inputs & Outputs of the Shipping Engine

Inputs:

• store_id

• merchant_id (implicit via store)

• cart_items:

- product_id

- quantity

• destination:

- country (ISO-3166)

- state (optional)

- postcode (integer or numeric string)

• optional: cart subtotal (if precalculated)

Derived Inputs:

• total_weight_grams = Σ(product.weight_grams × quantity)

• cart_total_cents = Σ(line_total_cents)

• active shipping_zones, shipping_methods, shipping_rates for store

Outputs:

• A list of candidate shipping options:

- shipping_method_id

- label (e.g., 'Standard', 'Express')

- description (optional)

- price_cents

- estimated_delivery_text (optional)

• An error condition if no method matches (e.g., “Delivery not available to your area”).

3. Zone Resolution Algorithm

Zone resolution is the first step in the engine.

Steps:

1. Normalize destination:

- Ensure country is uppercase ISO-3166.

- Strip spaces from postcode and parse to integer where possible.

2. Query zones:

```
SELECT * FROM shipping_zones
WHERE store_id = ?
AND country = ?
AND postcode_from <= ?
AND postcode_to >= ?;
```

3. If state is present on both zone and destination:

- Require state equality.

4. Result:

- Zero zones: no delivery possible.

- One or more zones: engine will attempt to match rates via methods.

Multiple matches:

• It is allowed, but recommended to keep zones non-overlapping.

• If overlapping exists, all candidate zones are evaluated; if more than one rate matches, the engine chooses the cheapest valid option per method or per overall configuration rule.

4. Shipping Method Filtering

After resolving zones, shipping methods are filtered:

Steps:

1. Retrieve all active methods for the store:

SELECT * FROM shipping_methods

WHERE store_id = ?

AND is_active = 1;

2. For each active method:

- Attempt to locate shipping_rates rows that match:

- A zone from the resolved set

- The weight range (if set)

- The cart total range (if set)

3. A method is considered "available" if:

- At least one shipping_rates row matches for the requested destination and cart.

Method Types:

• flat: uses base_price_cents, ignores weight/price ranges used only for gating.

• weight: uses base + price_per_kg scaling with total_weight_grams.

• price: typically used with thresholds; may have multiple tiers.

• formula: reserved for future complex carriers; in v1 behaves similar to weight/price but may store additional metadata.

5. Rate Matching & Cost Calculation

For each candidate method and zone, rate matching:

Core SQL filter:

• min_weight_grams IS NULL OR min_weight_grams <= total_weight_grams

• max_weight_grams IS NULL OR max_weight_grams >= total_weight_grams

• min_cart_total_cents IS NULL OR min_cart_total_cents <= cart_total_cents

• max_cart_total_cents IS NULL OR max_cart_total_cents >= cart_total_cents

If multiple rows match:

• Choose the most specific row, preferring:

- narrowest weight range

- then narrowest cart total range

- or explicitly, the row with highest min_weight_grams and/or min_cart_total_cents that still matches.

Cost formula:

• base_price = base_price_cents

• if price_per_kg_cents not NULL:

variable_component = price_per_kg_cents * ceil(total_weight_grams / 1000)

else:

variable_component = 0

• total_shipping_cost = base_price + variable_component

The engine returns an option like:

{

"method_id": 12,

"label": "Standard",

"price_cents": 1299,

"meta": {

"zone_id": 3,

"calc": {

```
"base_cents": 999,

"per_kg_cents": 300,

"billable_weight_kg": 1

}

}

}
```

6. Error Handling & Fallback Strategies

There are multiple points where shipping calculation can fail:

1. No zones match:

- Return explicit failure: `DELIVERY_NOT_AVAILABLE_FOR_POSTCODE`.

- UI should prompt user to verify postcode or switch to pickup.

2. Zones match but no methods:

- Merchant misconfiguration: surface generic message to customer.

- Optionally log a warning to merchant/admin.

3. Methods exist but no rate rows:

- Treat as configuration error.

- Auto-hide that method from the options for this request.

- Continue calculating with other methods.

4. Calculation errors (e.g., invalid data):

- Catch exceptions.

- Return generic error: “We couldn’t calculate shipping for this order.”

- Log details for support.

Fallback:

• If shipping fails:

- If pickup enabled, suggest pickup as alternative.

- If no fulfilment is possible, block checkout and instruct to contact store.