Technical Specification: FREE Report Matching Logic

Date: 19 November 2025
Version: 2.3 — Aligned with Actual Questionnaire & Content
Status: ✅ CURRENT FOR MVP

---

1. Purpose

This technical specification details the step-by-step matching logic for the FREE care home shortlist report, using the real questionnaire structure and underlying scoring rules. Coverage spans all logic from questionnaire input to the generation of the three recommended homes (Safe Bet, Best Reputation, Smart Value), including edge cases and key downstream output fields.

---

2. Key Data Sources

- **CQC Public API**: Regulatory info, rating (Outstanding, Good, Requires Improvement, Inadequate), registration type
- **FSA Food Hygiene**: Basic score (mainly used for warnings, minor in shortlist logic)
- **Google Places**: Public rating (0.0–5.0), number of reviews
- **Distance Calculation**: Postcode-to-postcode (uses OSM or similar service)
- **Pricing**: From partners or scraped (current average weekly fee)
- **Basic home profile**: Name, location, contact, basic features (nursing, dementia, etc.)

---

3. Questionnaire Fields (MVP v2025-01-27)

The logic below is **driven by user answers** to the following questionnaire fields (key fields marked with *):

1. *Postcode (user_location)**
2. *Care type:** residential / nursing / dementia / respite / not_sure
3. ***Budget (weekly)*
4. **Distance willingness**: miles (default: up to 10 mi)
5. **Key requirements**: e.g., dementia care needed, step-free access
6. **Special medical conditions (optional)**

Derived fields:

- `max_distance`: User selection or default
- `specialists_required`: parsed from either their care type or freeform requirements

---

4. Candidate Pool Filtering

1. Step 1: Filter all homes that:

- Are within `max_distance` miles of user_postcode (geospatial query)
- Provide the selected `care_type`, or "not_sure" allows all
- **Weekly price** is less than or equal to budget + allowance (if budget is set; typically + £100–200 flexibility)
- (Optional) Special focus: If dementia or other condition required, only homes offering these are prioritized (but not hard-excluded at Free tier)

2. If <3 homes returned:

- Relax criteria in this order: care_type → price → distance

---

5. Scoring Logic for 3 Strategic Homes

Strategy 1: **Safe Bet**

- **Primary**: Highest CQC rating, then closest distance
- Scoring:
  - Outstanding: 25 pts, Good: 20, Requires Improvement: 10, Inadequate: 0
  - Bonus: If within 5mi: +5pts
  - Tiebreak: Lowest incident history (if data available)

Strategy 2: **Best Reputation**

- **Primary**: Highest Google rating (min 10–20 reviews for validity)
- Scoring:
  - Google 4.8–5.0: 20pts; 4.4–4.7: 15pts; 4.0–4.3: 10pts; <4.0: 0
  - Review count ≥20: +5pts
  - Tiebreak: Distance

Strategy 3: **Smart Value**

- **Primary**: Best quality-to-price
- Scoring:
  - (CQC_score + 2*Google_score) / weekly_price (normalized)
  - Penalize homes charging >10% higher than user budget unless quality is especially high
  - Prefer all-in prices, lowest extras if data present

---

6. Pseudocode: Home Scoring & Selection

Assume homes: list of CareHome, and user_inputs dict matching questionnaire

def shortlist_homes(homes, user_inputs):
# FILTER
candidates = [
h for h in homes
if within_distance(h, user_inputs["postcode"], user_inputs["max_distance"])
and (user_inputs["care_type"] == "not_sure" or user_inputs["care_type"] in h.care_types)
and h.weekly_price <= user_inputs["budget"] + 100
optionally: check for dementia/specialisms requirements if specified
]
if len(candidates) < 3: # Relax criteria as described

```
# SAFE BET
safe_candidates = sorted(
    candidates,
    key=lambda h: (
        cqc_to_weight(h.cqc_rating),
        -distance(h, user_inputs["postcode"])
    ),
    reverse=True
)
safe_bet = safe_candidates[0] if safe_candidates else None

# BEST REPUTATION
rep_candidates = sorted(
    [h for h in candidates if h.google_reviews >= 10],
    key=lambda h: (h.google_rating, h.google_reviews),
    reverse=True
)
best_rep = rep_candidates[0] if rep_candidates else safe_bet

# SMART VALUE
value_candidates = sorted(
    candidates,
    key=lambda h: (
        ((cqc_to_weight(h.cqc_rating) + (h.google_rating*4)) / h.weekly_price) if h.w
    ),
```

```
    reverse=True
)
smart_value = value_candidates[0] if value_candidates else safe_bet

# Return unique results preserving order
return list(dict.fromkeys([safe_bet, best_rep, smart_value]))
```

def cqc_to_weight(rating):
return {"Outstanding": 25, "Good": 20, "Requires Improvement": 10, "Inadequate": 0}.get(rating, 0)

---

7. Output Field Mapping (as delivered in report)

- careHome.name
- careHome.location, postcode
- careHome.strategy ("Safe Bet" / "Best Reputation" / "Smart Value")
- careHome.cqcRating
- careHome.distance
- careHome.weeklyPrice
- careHome.whyChosen
- careHome.keyStrengths
- careHome.contact.phone, email
- careHome.detailedProfile (optional unblock: specializations, performance, key features, considerations)

All fields must be completed for each of the 3 homes.

---

8. Edge Case Handling

- If no home provides the care type, use "not_sure" fallback (show warning to user)
- If <3 candidates, show what was possible, explain in report, and suggest increasing radius or raising budget
- Price missing: treat as inf (home not eligible unless all others lack price)
- Google reviews missing: deprioritize in Best Reputation, not disqualified

---

9. Telephone Enquiry Checklist and Next Steps

In addition to homes, FREE report must generate:

- Personalized printable checklist (10 questions), with autofilled names/contacts for each home
- Action plan cards (phone, visit, red flags, observation list)

---

10. Performance/Accuracy Considerations

- All filtering/selection logic must work near real-time (<3 sec SQL/filter, <10 sec render pre-caching).
- Mock data fallback: If API down/missing, inform user and display available homes as fallback (do not break the shortlist).

---

11. Example Data Structure (TypeScript)

```
interface CareHome {
id: string;
name: string;
location: string;
postcode: string;
strategy: 'safe_bet' | 'best_reputation' | 'smart_value';
cqcRating: string;
distance: string;
weeklyPrice: number;
whyChosen: string;
keyStrengths: string[];
contact: { phone: string; email: string; };
detailedProfile?: {
specializations: Array<{title: string, description: string}>;
keyFeatures: string[];
performance: {
staffRetention: string;
familySatisfaction: string;
incidentRate: string;
};
considerations: string[];
};
}
```

---

12. Future Extensions

- Extra filters for wheelchair/limited mobility, culture/language preferences, private vs LA funding paths
- Deeper integration of user text/voice input for conditions and wants

---

13. References

- [translate:Care Quality Commission] (CQC) API docs
- [translate:Google Places API] documentation
- [translate:Office for National Statistics] (postcode mapping)

End of document.