engagement_output_v2...
Document

specificity_output_v2.txt
Document

i need to create PRD for new partner preference feature.

major change is we show profiles accorinf to partner prefernce like we boost profiles wiht preferenes higher or x:y ratio of profiles from partner preference and best_matches.

it is soft thing. on top of that there will be dealbraker toggle for each attribute, which we will completely remove if profiles from such outside taht atrributes, few like mariatl status, age. location will be free but other attributes deal breaker will be paid.

strict filter will be attacthd to patrner preference thing as hide me kind of feature. wher eit will be paid feature.

how shluld i structure this PRD. dont generate PRD. just tell me what all the things ishould include, things i shouldnt miss..etc. must have things. all deatiled

Thought for a couple of seconds ›

I'll outline a PRD structure tailored to your "soft preference boosting + dealbreaker toggles + paid strict filter" feature, and list the must-have sections, edge cases, metrics, and pitfalls so you don’t miss anything.

Thought for 54s ›

Here’s a **PRD structure + “don’t miss this” checklist** that fits your feature (soft preference boosting + X:Y mix with best_matches + per-attribute dealbreakers + strict/“hide me” paid filter). I’ll describe **what to include in each section**, the **key decisions you must nail**, and the **common gaps** teams forget.

---

1) One-page overview (keep this at the top)

Include (super crisp):

- **Problem statement** (what user pain / business problem you’re solving)
- **What’s changing** in one paragraph:
    - Feed will **rank/boost** profiles matching partner preferences (soft)
    - Feed will follow an **X:Y composition** between *preference-aligned* profiles and *best_matches*
    - **Dealbreaker toggles** per attribute (hard removal if on)
    - **Which dealbreakers are free vs paid**
    - **Strict filter / “hide me” behavior** as a paid feature
- **Success metrics** (primary + guardrails)
- **Scope** (what’s in / out)

Why this matters: stakeholders should understand the whole feature without reading the entire doc.

---

2) Context + insights (what we know today)

This section makes your PRD “credible” and helps justify trade-offs.

Include:

- **Current product behavior**: how feed ranking works today (best_matches logic, any existing filters)
- **User behavior baseline**:
    - % users who even set preferences, how specific they are, and strict filter usage
    - (Example: your analysis shows many user-edited preferences remain “open/any” for several attributes, and strict filter adoption differs a lot by

premium vs free.) specificity_output_v2

- **Outcome baseline**:
  - engagement differences when a shown profile is “in preference” vs “out of preference”
  - (Example: your engagement report shows acceptance rate tends to be higher for “in preference” interactions than “out”.) engagement_output_v2
- **User feedback / anecdotes**: top 5 complaints you’re addressing (e.g., “I keep seeing irrelevant profiles”; “dealbreakers aren’t respected”; “I want full control but not at the cost of zero supply”)

Don’t miss:

- Call out **why it must be “soft”** (avoid killing supply, avoid echo chamber, avoid empty feeds).

---

3) Goals, non-goals, principles

Goals (must be measurable)

Examples of goal types (pick yours):

- Improve **match quality perception** (e.g., ↓ “irrelevant profiles” feedback)
- Improve **conversion/engagement** (acceptance rate, conversation starts)
- Drive **monetization** via paid dealbreakers + strict filter

Non-goals (prevents scope creep)

- Not rebuilding best_matches model
- Not adding new preference attributes (unless explicitly in scope)
- Not changing onboarding flows (unless needed)

Product principles (VERY important for this feature)

- Keep feed **healthy**: don’t over-filter into emptiness
- Keep it **explainable**: user should understand why they see profiles
- Keep it **fair/safe**: avoid harmful bias or “discriminatory feel” surprises
- Preserve **exploration**: still show some high-quality outside-preference profiles

---

4) User stories + use cases

Write them by **persona + scenario**, not only “as a user I want…”.

Include at least:

- New user with **no preferences**
- User who sets a few preferences, expects **better feed**
- User who toggles **dealbreakers** and expects “hard remove”
- Paid user enabling strict filter (“hide me / strict filter visibility”)
- Low-supply user (small city / niche preferences) where the system must **fallback gracefully**

Don’t miss:

- “I set dealbreaker but still saw X” → trust breaker → needs strict correctness + messaging

---

5) Definitions & terminology (avoid ambiguity)

This feature is full of terms that can be interpreted differently. Lock them down.

Must define:

- **Partner preference**: attribute list, how ranges work, how “Any” works
- **Preference match score**: per-attribute match, weighted sum, etc.
- **Best_matches**: what signal/source, and how it’s ranked
- **Boosting** vs **filtering**:
  - Boosting = changes ordering/mixture
  - Filtering = removes from candidate set
- **Dealbreaker**: “hard exclusion if enabled”
- **Strict filter / hide me**: be explicit: *who is hidden from whom*?

Also define attribute categories:

- Range (age, height)
- Geo (location: city/state/country + radius)
- Categorical single (marital status)
- Multi-select lists (education, community, etc.)
- Missing/unknown values behavior

---

6) Proposed solution (functional requirements)

Split into 3 submodules so engineering/design can work in parallel.

A) Feed composition & ranking (soft preferences + X:Y mix)

Include:

- Candidate generation sources:
    - Best_matches candidate pool
    - Preference-matching candidate pool
- **X:Y ratio spec** (this is critical):
    - Default ratio (and why)
    - Is ratio enforced per page, per session, per day?
    - What happens when you can't satisfy the ratio (fallback rules)
- Boosting rules:
    - How much boost? (relative ranking vs absolute score bump)
    - Per-attribute weights
    - Do we boost "near matches" (e.g., age slightly outside) or only exact matches?
- Diversity controls:
    - Avoid showing only one segment repeatedly
    - Caps per attribute (optional)

Don't miss:

- "Soft" means you must document **minimum outside-preference exploration** rules, otherwise product becomes de-facto strict without users realizing.

B) Dealbreaker toggles (hard filters)

Include:

- Which attributes get dealbreaker toggles
- Exact semantics:
    - If dealbreaker ON → remove profiles outside range/category
    - What about profiles with missing info? (exclude or include?)
    - What about conflicting preferences? (multiple toggles)
- Free vs paid entitlements:
    - You mentioned **age + marital status** as free hard dealbreakers
    - **Location** is free but likely has special handling (radius / nearby)
    - Other dealbreakers are **paid** (document the list + SKU mapping)
- Paywall behavior:
    - If user tries to toggle paid dealbreaker:
        - show upsell, but do you allow "preview"?
        - what happens if they downgrade later?

Don't miss:

- “Hard remove” requires correctness in backend filtering; client-only filtering can cause bugs and trust loss.

C) Strict filter / “hide me” (paid)

This is the most ambiguous part, so your PRD must be extra explicit.

Include:

- Definition: is it…
  1. “Only show me profiles that match my dealbreakers” (strict discovery), OR
  2. “Hide my profile from people who don’t match my criteria” (visibility control), OR
  3. Both?
- Directionality:
  - If I enable strict filter, does it affect:
    - profiles I see?
    - profiles that see me?
    - both?
- Edge behavior:
  - If strict filter makes supply too low → what messaging do we show?
  - Does strict filter override X:Y soft mix, or still allow exploration?
- Monetization + eligibility:
  - who can access it (subscription tier)
  - is it bundled with other features?

Don’t miss:

- This feature can drastically affect marketplace liquidity. Your PRD must include **guardrail metrics** and **fallback rules**.

---

7) UX requirements (screens, states, copy guidance)

Even if you don’t include mockups, you must include **states**.

Include:

- Entry points:
  - where user sets preferences
  - where feed explains “based on your preferences”
- Dealbreaker toggles UI:

    - per attribute toggle
    - paid lock indicator + paywall flow
- Explanations:
    - “Soft preference” explanation (users often assume it’s strict)
    - “Why am I seeing this?” module (highly recommended for trust)
- Empty/low-supply states:
    - show “widen your filters” suggestions
    - show recommended defaults or quick widen actions
- Migration / defaulting:
    - existing users: what do they see on launch day? are dealbreakers OFF by default?

Don’t miss:

- Don’t silently change feed without telling users; it triggers “product feels broken” feedback.

---

8) Data & algorithm specs (what engineering/data science needs)

Include:

- Inputs:
    - preference fields schema
    - user attributes schema
    - missing values handling
- Scoring:
    - per attribute match function (binary / distance-based)
    - weights (even if “to be tuned”, specify ownership + tuning plan)
- Candidate selection:
    - pool sizes
    - dedupe rules between pools
- Ratio enforcement details:
    - enforce in ranker or post-ranker blending
- Performance constraints:
    - latency budget, caching strategy, pagination consistency

Don’t miss:

- Logging requirements for debugging (why a profile was shown, which pool it came from, which dealbreakers were applied).

---

9) Metrics (primary + guardrails + monetization)

Primary success metrics

- Feed-level: CTR, profile open rate, like/interest sent rate
- Match pipeline: acceptance rate, chat start rate, reply rate
- Quality perception: dismiss/skip, “not relevant” feedback, blocks

(Your baseline engagement report showing “in preference” tends to outperform “out” can be used to set targets and expected lift.) engagement_output_v2

Monetization metrics

- Paywall impressions → conversions (dealbreaker paid attributes)
- Strict filter attach rate
- Upgrade/downgrade churn impact

Guardrails (non-negotiable for marketplace)

- Supply: profiles per session, empty feed rate
- Diversity: segment distribution not collapsing
- Fairness/safety: disproportionate exposure drop for certain cohorts
- Retention: D1/D7/D30, complaint rates

Don’t miss:

- Guardrail for “over-filtering”: your specificity analysis suggests many users keep preferences open, so defaults and nudges matter; strict choices can suddenly reduce supply. specificity_output_v2

---

10) Experimentation & rollout plan

Include:

- A/B test plan:
  - test soft boosting first (no dealbreakers)
  - then introduce dealbreakers
  - then strict filter (highest risk)
- Ramp strategy:
  - internal → 1% → 10% → 50% → 100%
- Monitoring dashboard list
- Rollback plan

Don’t miss:

- Separate experiments for:
    - ratio (X:Y) tuning
    - boost strength
    - paywall placement + copy

---

11) Edge cases checklist (this is where PRDs usually fail)

Must cover:

- No preferences set → do we keep current best_matches fully?
- Preferences set but too strict → fallback behavior + messaging
- Paid dealbreaker toggled then subscription expires
- Conflicting toggles (e.g., age range too narrow + strict filter)
- Missing attributes in candidate profiles
- Fraud/spam profiles gaming preference boosts
- Regional supply differences (metro vs non-metro)
- Pagination consistency (user shouldn't see "random jumps" page to page)

---

12) Dependencies, risks, and open questions

Dependencies

- Ranking service / matching pipeline
- Billing/entitlements
- Analytics events
- Moderation/safety policies (if certain attributes are sensitive)

Risks (call out explicitly)

- Marketplace liquidity drop due to strict filters
- Perceived discrimination / PR risk depending on attributes
- Trust issues if dealbreaker doesn't behave strictly
- Revenue vs retention trade-off

Open questions (limit to the real ones)

Examples:

- Final paid attribute list for dealbreakers
- Strict filter semantics (see/seen-by)
- Default X:Y ratio and tuning ownership

Don't miss:

- Assign owners for unresolved decisions (PM/DS/Eng/Design).

What you should *not* forget (quick "PRD must-have" checklist)

If these are missing, you'll get stuck in execution:

- ✅ Clear definition of **soft vs hard** behavior
- ✅ Exact **X:Y ratio** enforcement rules + fallback
- ✅ Strict filter directionality (**I see** vs **people see me**)
- ✅ **Paid entitlements** mapping per attribute + downgrade behavior
- ✅ Low-supply and empty-state UX + "widen" suggestions
- ✅ Logging + explainability ("why shown")
- ✅ Guardrails for supply/diversity/fairness
- ✅ Experiment + rollback plan

If you want, paste your current PRD skeleton headings here and I'll **review it like a checklist audit** (what's missing, what's redundant, and what should move to an appendix).

Sources

i am having hard time on fixing problem statement and what are we solving as we are touching multiple things

Thought for 9s >

Totally normal to feel stuck here—when a feature touches ranking + filtering + paywalls, it *feels* like "multiple problems." The trick is to write **one umbrella problem** and then break it into **2–3 sub-problems** that map to each lever (soft boost, dealbreakers, strict filter).

Below is a practical way to do it + a few ready-to-use problem statement options (not a full PRD, just the problem framing).

A simple way to “fix” the problem statement (even when scope is broad)

Step 1: Pick the *one user outcome* you care about

Most likely for your case, it’s one of these:

- **Relevance:** “Show me people I’m more likely to match with”
- **Control + trust:** “If I say something is a dealbreaker, don’t show it”
- **Efficiency:** “Reduce time wasted swiping irrelevant profiles”
- **Discovery health:** “Do this without making the feed empty / too narrow”

Pick **one** as the umbrella. The others become supporting goals/constraints.

Step 2: Describe the *current failure* in one sentence

Use this formula:

Today, users ____ because ____, which leads to ____ (measurable impact).

Examples (fill with your real symptoms):

- “Today, users see too many profiles that don’t align with stated preferences, so they lose trust and disengage.”
- “Today, preferences aren’t reliably reflected in ranking, so the feed feels random and users waste time.”

Step 3: Add constraints (so it doesn’t become “just strict filters”)

Your constraints are important and actually unify the doc:

- Must **preserve supply** (avoid empty feed)
- Must maintain **exploration** (best_matches still matters)
- Must be **understandable** (users know what preferences do)
- Must support **monetization** without breaking core experience

So your umbrella statement becomes: improve relevance/control **while preserving healthy discovery**.

Step 4: Create 2–3 sub-problems (these explain why you need multiple changes)

These sub-problems should map cleanly:

1. **Relevance without killing supply** → solved by *soft boosting + X:Y blending with best_matches*

2. **Control users can trust** → solved by *dealbreaker toggles (hard remove)*
3. **Advanced control worth paying for** → solved by *paid dealbreakers + strict filter/hide-me behavior*

This makes it feel like one coherent initiative.

Problem statement templates you can use (choose one style)

Option 1: User-first (best when PRD is product-led)

Users invest time setting partner preferences, but the feed still surfaces many profiles that don't align—making discovery feel inefficient and lowering trust. At the same time, enforcing strict filters by default risks reducing supply and creating empty/low-quality feeds. We need a preference-driven discovery system that increases perceived relevance, offers trustworthy "dealbreaker" control, and preserves healthy exploration—while enabling premium tiers for users who want stricter control.

Why it works: ties together all 3 levers under one "preference-driven discovery system."

Option 2: Marketplace health + relevance (best when engineering/DS heavy)

Our current discovery ranking under-utilizes partner preference signals, causing suboptimal match quality and wasted impressions. However, moving to strict filtering broadly would reduce marketplace liquidity and increase empty-feed risk. We need a hybrid approach that blends preference-aligned profiles with best_matches, supports deterministic dealbreaker exclusions where promised, and introduces premium strict controls without degrading supply or fairness.

Why it works: makes "soft + hard + paid" logically necessary.

Option 3: Trust/control framing (best when users complain "it doesn't respect my preferences")

Users don't know what partner preferences actually do and can't rely on them to exclude unacceptable matches. This creates a trust gap: users either churn due to irrelevant profiles or over-constrain using strict filters and hit low supply. We need to make preferences meaningful by default (soft prioritization), provide explicit

dealbreaker guarantees for key attributes, and offer paid strict controls for users who want maximum certainty—without collapsing discovery variety.

Why it works: emphasizes explainability + “guarantees” (dealbreaker must be correct).

How to write “what are we solving” in a way that covers all parts

Use this 3-part structure:

1) Primary problem (1–2 sentences)

Focus on *one*:

- Relevance & efficiency OR trust/control OR marketplace health

2) Why now (1 sentence)

Examples:

- “Preference usage has increased but impact on feed is unclear.”
- “We see rising complaints about irrelevant profiles / mismatch.”
- “Premium demand exists for stricter control.”

3) What success looks like (bullet list)

- Users report “more relevant profiles”
- Higher acceptance / chat starts
- No big rise in empty feed / low supply
- Paid attach rate for advanced controls without hurting baseline retention

---

A quick sanity check: do your levers map to the problem?

If your problem statement is correct, each feature piece should answer a “how might we” question:

- **Soft boosting + X:Y blend** answers:
  “How might we increase relevance *without* losing supply/exploration?”
- **Dealbreakers** answer:
  “How might we give users guarantees for non-negotiables?”
- **Paid strict filter/hide-me** answers:
  “How might we monetize advanced control without forcing it on everyone?”

If any feature doesn’t map to one of these, it’s either extra scope or the problem statement is too vague.

If you're still stuck, use this "single sentence" trick

Write this exact sentence and fill blanks:

We are building preference-driven discovery so that users see more compatible profiles sooner, can enforce true dealbreakers when needed, and can opt into stricter (paid) controls—while keeping discovery supply and exploration healthy.

That single sentence can be your umbrella.

If you paste your current rough problem statement (even messy), I'll rewrite it into 2–3 strong variants in your tone and make sure each of your components clearly ladders up—without turning it into a full PRD.

thing is we collect partner preference currently, but we does not do anyhting we that data, that make user frustrated, they think app broken because of that, this is one of the reason we are doing it now

That's actually a *clean* core problem. You can anchor everything under one umbrella:

"We collect partner preferences but don't operationalize them, creating a trust gap."

Once you name it that way, the "multiple things" become *solutions to one trust + relevance gap*, not separate projects.

Here's how to write it in a tight, convincing way.

The right framing: "Trust gap" + "wasted user effort"

What you described has 3 sharp pains:

1. **Broken promise / expectation mismatch**

- Users spend time setting preferences.

- Feed doesn’t reflect them.
- Users conclude: *“app is broken / preferences don’t work.”*

2. **Wasted effort → frustration → churn risk**

- Any feature that asks for input but doesn’t use it feels deceptive.
- This is especially true in matchmaking/discovery.

3. **Support burden + negative word-of-mouth**

- People complain, leave low ratings, or stop engaging.

So your problem statement can be centered on:

- **Trust + perceived quality**, not just ranking.

A strong “Problem statement” template for your case

Use this structure:

Problem (what’s happening)

We currently collect partner preference data, but discovery does not use it in ranking or filtering.

Impact (why it matters)

This creates a trust gap: users expect preferences to influence what they see; when it doesn’t, they feel the app is broken and become frustrated, reducing engagement and retention.

Why now

We already have the data; using it is the fastest path to improving perceived relevance and restoring trust without needing new collection flows.

That’s enough. Then your “multiple things” become: the set of mechanisms to *deliver on that promise safely*.

“What are we solving” (1 paragraph that unifies everything)

You want a single umbrella statement like:

We are solving the gap between user-stated partner preferences and the profiles they are shown. Today, users provide preferences but don’t see any visible impact, which erodes trust and makes the experience feel broken. We need to make

preferences meaningfully influence discovery while protecting supply/variety (soft blending) and offering explicit control for non-negotiables (dealbreakers), with advanced strict controls monetized.

This makes:

- soft boosting = "visible impact without breaking supply"
- dealbreakers = "guarantee when users demand it"
- strict filter/visibility = "premium control layer"

All one story.

---

3 ready-to-use problem statement variants (pick the tone you like)

Variant A: Most direct (trust + broken promise)

Users set partner preferences but discovery does not reflect them, causing users to believe the app is broken and reducing trust in the product. We need to operationalize partner preferences in discovery so users consistently see a meaningful impact, with safeguards to maintain supply and exploration.

Variant B: More product-y (efficiency + relevance)

Discovery feels inefficient because users are shown many profiles that don't align with their stated partner preferences. Since we already collect preference data, we can improve perceived relevance quickly by using preferences in ranking and composition—while ensuring users can enforce true dealbreakers and keeping the feed healthy.

Variant C: Marketplace-aware (balances supply)

We collect preference data but underuse it, leading to low perceived relevance and trust. However, strict enforcement for everyone risks shrinking supply. We need a hybrid preference-driven discovery system: soft prioritization blended with best_matches, plus optional dealbreakers and paid strict controls.

The "promise" sentence (super useful in PRDs)

Add one line to clarify the contract with the user:

Current promise (implicit): “Tell us what you want; we’ll show more of it.”
Reality today: “We ask, but don’t apply it.”
New promise: “Preferences influence what you see by default; dealbreakers are enforced when enabled.”

This helps align UX copy, support expectations, and engineering correctness.

Don’t miss: Make the problem measurable (even if data isn’t perfect)

You don’t need perfect analytics to define measurable outcomes. Add metrics that capture *trust gap closing*.

Primary success signals (pick 2–3)

- ↑ “preference-aligned impressions share” (or “in-preference rate”)
- ↑ acceptance/like rate or conversation start rate
- ↓ skips/dismissals after profile open
- ↓ “irrelevant profiles” feedback / support tickets mentioning preferences
- ↑ retention among users who set preferences

Guardrails (because you’re changing feed logic)

- No spike in empty-feed or “low supply” sessions
- No meaningful drop in overall matches/conversations
- No disproportionate exposure collapse for certain segments

How to connect the “multiple things” to the single problem (one-liner mapping)

In your PRD, you can literally write:

- **Soft preferences (boost + X:Y blend)** → ensures preferences have a *visible* impact without killing supply
- **Dealbreakers** → closes the trust gap by providing *guarantees* for non-negotiables
- **Paid strict filter / hide-me** → advanced certainty/control for users who want it, without forcing strictness on everyone

That’s the narrative glue.

One caution: don’t overclaim what “preferences” mean

A common PRD pitfall: writing “preferences will be respected” (sounds like strict filtering) when you mean “soft boosting.”

So explicitly say:

- **Default = soft**
- **Dealbreaker toggles = strict for that attribute**
- **Strict filter = paid strict mode** (define directionality clearly)

This prevents “users still complain because they expected hard filtering.”

If you want, paste your draft 2–4 lines for the problem statement + why now, and I’ll tighten it into a final version that:

- is single-threaded (not scattered)
- doesn’t overpromise
- naturally justifies the boosting + dealbreaker + paid strict approach.

i have few doughts, things we cap 50 profiles per 12 hour for free users. so if we allow PP change the profiles will change, so may be they user keep on changing PP, so profiles shown will be higher than 50 profiles so how do we solve this

Thought for 28s ›

You’re right to worry: **if PP changes cause a “new” feed**, free users could game the 50/12h cap by repeatedly editing preferences. You solve this by deciding what the cap *actually counts* and how PP updates interact with the “eligible pool” inside a cap window.

Here are the **best-practice options** (with tradeoffs), plus what I’d recommend.

---

1) First lock the definition: what does “50 profiles” mean?

This is the key PRD decision.

Strong recommendation: define the cap as
"50 unique profiles revealed (impressed) per rolling 12 hours"
not "50 per preference configuration."

That means:

- If user changes PP 10 times, they still can't see >50 **new unique** profiles in that 12h window.
- They can reorder/reshuffle what they *haven't seen yet*, but they don't get extra inventory.

You'll need to specify what counts as "revealed":

- Count when a profile card is **actually shown** (e.g., enters viewport / user swipes to it), not when it exists in the backend list.
- Don't count duplicates (same profile shown again after PP change should not re-consume quota).

---

2) Implementation patterns to prevent PP abuse

Option A (most common): Quota is global, PP only changes ranking

Rule: Free user has 50 "new profile credits" per 12h. PP changes only affect:

- the **order** of remaining unseen profiles
- which unseen profiles are pulled next **within remaining credits**

What happens on PP change?

- Previously seen profiles remain "seen"
- You refresh scoring on candidates, but you only show as many **new unique** as credits remaining

✅ Prevents abuse
✅ Matches the cap promise
⚠️ User may expect "totally new people" after PP change; you must message it well ("Preferences updated; you'll see more relevant suggestions going forward")

UX copy idea:
"Preferences updated. You can view X more new profiles in this window. More unlock in Y hours."

Option B: Cooldown / rate-limit PP changes

Add a friction layer to prevent rapid cycling.

Examples:

- Allow PP edits anytime, but **apply to feed** only after a cooldown (e.g., 15–30 min)
- Or "You can update partner preferences **N times per 12 hours**" (e.g., 3)
- Or require "Save & Apply" confirmation that warns about limited daily viewing

✅ Stops rapid gaming / reduces system churn
✅ Helps performance + stability
⚠️ Can feel restrictive if user is genuinely trying to tune preferences

This works best as a **secondary guardrail**, not the main one.

Option C: Apply PP changes only at the next quota reset

Rule: PP edits are saved immediately, but discovery feed changes only when the 12h window resets.

✅ Super clean for abuse prevention
✅ Very simple to reason about
⚠️ Bad UX: "I changed preferences and nothing happened" (same trust issue you're trying to fix)

I'd avoid this unless your system can't support dynamic recalculation.

Option D: Freeze a "daily pool of 50" at window start

At the beginning of the 12h window, you pre-select 50 candidates (or a larger pool like 200) and then:

- PP changes only **re-rank** within that pool

✅ Prevents PP from "unlocking" new inventory
✅ Stable pagination and caching
⚠️ If PP changes drastically, pool may not contain enough matching profiles → user feels it still doesn't work

If you use this, freeze a *larger pool* than 50 (like 200–500) to allow meaningful re-ranking.

3) Best recommended approach (simple + fair + good UX)

Use a **hybrid**:

Core rule (non-negotiable)

Cap = 50 unique new profiles revealed per 12 hours (free).
This cap is **independent of PP changes**.

PP change behavior

- PP change triggers re-scoring and affects:
  - ranking of remaining unseen candidates
  - blending ratio (X:Y) *for the remaining credits*
- It does **not** reset credits or grant additional unseen profiles.

Guardrails

- Add a **PP apply cooldown** (e.g., 10–20 minutes) *only if needed* to stop spam + reduce backend thrash.
- Hard block obvious abuse patterns (e.g., 20 edits in 5 minutes).

UX

- Show **remaining new profiles** counter for free users (even if subtle).
- On cap hit: allow them to browse already seen, but lock new (or upsell).

---

4) Edge cases you must specify in PRD

These are the ones teams forget:

A) Does reopening the app show new profiles without consuming credits?

- If a profile was already revealed earlier in the window → **don't count again**.
- If the feed reshuffles due to PP change and surfaces old profiles again → still **no extra cost**.

B) When do we decrement the "new profile" count?

Pick one:

- On card impression (recommended)
- On swipe action
- On profile open

Be explicit or you'll get inconsistent counting across platforms.

C) What if strict dealbreakers reduce supply mid-window?

If user turns on strict dealbreakers and now there are only 12 matching profiles available:

- Show fewer, explain why
- Offer "widen criteria" nudges
- Do not secretly break dealbreaker rules to fill 50 (that kills trust)

D) Best_matches vs PP mix with cap

If user has 10 credits left, enforce your X:Y mix *within those 10* if possible, otherwise fallback rules apply.

E) Performance and caching

Frequent PP changes can be expensive. You'll want:

- caching of candidate pool
- re-scoring without full recompute where possible
- rate-limits if needed

---

5) Product decision: do you *want* to allow PP change to "swap" profiles already seen?

If your cap is "50 unique shown," then swapping would increase unique count → abuse vector. So typically:

- **Seen profiles stay seen** within the window.
- PP change affects **future** unseen only.

If you really want to allow swapping for UX, do it as a **paid perk** ("Refresh feed / more profiles") rather than letting free users bypass caps.

---

6) What to write in the PRD section (so this doesn't get missed)

Add a dedicated subsection:

"Interaction with Free User Impression Cap (50/12h)"

- Cap definition (unique reveals)
- Counting rules (impression vs swipe vs open)
- PP change behavior (re-rank only, no extra inventory)

- Abuse prevention (cooldown/rate limit)
- User messaging states (updated PP, cap nearing, cap hit)
- Logging events (pp_updated, impression_count, cap_hit_reason)

If you tell me **what currently counts toward the 50** in your app today (impression vs swipe vs open), I'll recommend the cleanest counting rule and the exact UX states to avoid "preferences don't work" complaints while still preventing cap circumvention.

another thiing is we have profile search, how will be this diffrent from search

I'll clarify the conceptual and functional difference between partner-preference-driven discovery and profile search, and give you concrete PRD bullets: user intent, ranking/filter semantics, caps, UX entry points, and edge cases.

Search and Partner Preferences (PP) *must* feel different because they solve **different intents** and have **different rules**. If you don't define that clearly in the PRD, users will expect PP to behave like strict search—and you'll get "app is broken" complaints again.

Here's how to frame it + the exact bullets to include.

---

1) Core difference: intent

Partner Preferences (Discovery)

- **Intent:** "Show me a *better default feed* over time."
- Users aren't trying to find one specific type instantly; they want **ongoing relevance**.
- PP should be **low effort**, mostly "set once, benefit continuously."

Profile Search

- **Intent:** "Let me *actively* find people with specific criteria right now."
- It's **explicit querying**: user is okay spending effort to narrow results.

Put this in PRD as:

PP = passive personalization, Search = active filtering/querying.

2) Semantics: soft ranking vs strict filtering

PP feed semantics

- Default PP behavior is **soft**:
  - **boost / blend** preference-aligned profiles with best_matches (X:Y)
  - still includes exploration
- Dealbreaker toggles = **strict** for those attributes (hard exclude)

Search semantics

- Search should be **strict filter by default**:
  - If you select filters, results must obey them
  - Ranking may still exist (e.g., activity, compatibility), but only *within* the filtered set

So the difference is:

- **Discovery answers:** "What should we show you next?"
- **Search answers:** "Show me the set that matches these constraints."

---

3) UX: entry points + expectation setting

PP (Discovery)

- Lives in "Preferences" / "Discovery settings"
- Copy should say:
  - "We'll prioritize profiles that match your preferences."
  - "Turn on dealbreakers to strictly exclude."

Search

- Lives in a dedicated "Search" screen
- Copy should say:
  - "Filter results by criteria"
  - "Results update instantly"

Key PRD point: **don't reuse identical UI controls** without labeling, or users will assume they work the same.

---

4) Caps & economy (important for your 50/12h constraint)

This is where you must draw a hard line.

PP (Discovery)

- Counts toward the **50 unique profiles / 12h** cap (free users)
- PP changes should **not** create new inventory beyond cap (as we discussed)

Search

You have two clean models (pick one and document):

Model A (recommended): Search is “preview-only” for free

- Free users can search but:
  - results are blurred/limited, or only show a small number
  - opening/liking requires credits/subscription
- This prevents search from being a cap bypass.

Model B: Search uses the same cap

- Any profile opened/revealed from search consumes from the same 50/12h pool.
- Simple and fair, but search feels “expensive” for free users.

Whatever you choose, state it explicitly or users will exploit search to see unlimited profiles.

---

5) Data source difference (how results are generated)

PP Discovery

- Candidate pool comes from:
  - best_matches model + exploration + freshness/diversity constraints
- PP influences:
  - ranking boost
  - blending ratio
  - dealbreaker exclusions

Search

- Candidate pool comes from:
  - all searchable profiles matching strict filters (plus safety/moderation constraints)
- Then rank by:
  - relevance to query, activity, distance, etc.

This is a good PRD line:

- **Discovery is curated/recommended. Search is retrieved.**

---

6) “Trust contract” (what users can rely on)

Define a contract for each:

PP contract

- “Your preferences will influence what you see.”
- “Only dealbreakers guarantee exclusion.”
- “We may still show some profiles outside preferences to keep discovery healthy.”

Search contract

- “Filters are strictly applied.”
- “If you filtered X out, you won’t see X.”

This prevents mismatched expectations.

---

7) Practical product rules to avoid confusion

Add these rules in PRD:

- **Don’t show “dealbreaker” toggles inside Search** (or if you do, they are just “strict filters,” not monetized dealbreakers—otherwise it becomes weird).
- **Keep Search filters and PP attributes aligned** where possible, but:
  - PP = preference/priority + optional strictness
  - Search = strict constraints
- If user sets a filter in Search, **don’t silently write it back into PP** (unless explicitly “Save as preference”).

---

8) Edge cases you should cover

- If a user can’t find enough results in Search, show:
  - “Widen your filters” suggestions
- If PP has strict dealbreakers enabled and Search shows profiles outside them:
  - that’s okay, because Search is independent—BUT you must be clear:
    - “Preferences apply to Discovery feed, not Search” (or unify them intentionally)

Pick one:

- **Independent:** PP affects Discovery only (simpler mentally)
- **Unified:** PP acts as default filters in Search (but then PP starts feeling like strict search, risky)

Most apps keep them **mostly independent** to avoid confusion.

A clean 2–3 sentence answer you can paste into PRD

Partner Preferences personalize the Discovery feed (recommended profiles) through soft prioritization and controlled exploration, with optional dealbreakers for strict exclusion. Profile Search is an active tool where user-selected filters are applied strictly to retrieve a result set. Search and Discovery share the same safety/moderation constraints, but differ in intent, ranking semantics, and how they interact with free-user viewing caps.

If you tell me how Search works today for free users (unlimited results? limited? counts towards 50?), I can suggest the cleanest policy so Search doesn’t become a loophole and your monetization stays consistent.

what if users put preference that doesnt make any sense, like a guy from KErala putting preferenece as venuzela. or divorced person put deal breaker as only never married. or a boy with age 23 looking for prefenrce 48.

Thought for 43s ›

Those aren’t “nonsense” preferences so much as **low-supply / low-reciprocity configurations**. The PRD should treat them as a *predictable system state* and define (a) what we allow, (b) what we warn about, and (c) what we do when it produces near-zero results—especially with **dealbreakers** and **paid strict/hide-me**.

Below is what you should include (and the recommended product rules), using your examples.

1) Separate “invalid input” vs “valid but low-supply”

A) Invalid input (hard validation — must block)

These are things you **must not allow** because they break logic / policy:

- Age < 18 anywhere (min age constraint)
- Impossible ranges (min > max)
- Unsupported locations (if you don’t actually have Venezuela as a supported search region, don’t let them select it)
- Corrupted values / missing mandatory fields

PRD must specify: exact validation rules + error states.

B) Valid but low-supply (don’t block; handle gracefully)

Examples you gave fall here:

- Kerala user preferring Venezuela ✅ valid (if your product supports it), but likely **low supply**
- Divorced user wants only never married ✅ valid, but may reduce pool
- Age 23 preferring age 48 ✅ valid, but may be **low reciprocity** (many 48yo may not prefer 23)

PRD must specify: how the system behaves when supply is low.

---

2) Add a “Supply & Reciprocity” layer (this is the real solution)

You need an explicit mechanism to detect “this will lead to few/no profiles.”

A) Supply estimator (must-have)

For any PP change (and especially when turning on a dealbreaker / strict mode), compute:

- **Estimated candidates available** (within your marketplace inventory + moderation rules)
- Optionally: **Estimated mutual candidates** (people who might plausibly accept you, based on their preferences if you use that)

Then show a simple indicator:

- **High / Medium / Low** availability (don’t show exact numbers unless you want to)

Why it matters: it prevents "app is broken" feeling because you're explaining the outcome before it happens.

B) Reciprocity awareness (optional but powerful)

For cases like 23 → 48, supply might exist but *reciprocity* might be low.

- If you already have "best_matches," it probably uses mutual signals; PRD should clarify whether PP boosting:
  - boosts **only among mutually compatible profiles**
  - or boosts everything (and accepts low conversion)

I'd recommend: **keep mutual compatibility as a guardrail**, or at least down-rank obviously non-reciprocal profiles.

---

3) Define behavior by mode (soft PP vs dealbreakers vs strict filter)

This is the most important part for your PRD.

Mode 1: Soft PP (default)

Rule: PP should never be able to "break" discovery.

- If PP is too narrow, you still blend in best_matches (your X:Y ratio), so user sees something.
- Show lightweight messaging:
  - "We're prioritizing your preferences and also showing strong matches to keep results flowing."

PRD must include fallback rules:

- Minimum exploration floor (e.g., at least Y% best_matches)
- What happens if preference-matching pool < needed? (fill from best_matches)

Mode 2: Dealbreaker toggles (hard exclusions)

Here you *can* hit empty states.

Rule: When dealbreaker ON → hard exclude outside attribute.

PRD must define:

- What about missing values? (exclude by default is safest for "dealbreaker guarantee," but reduces supply; choose intentionally)
- What happens if results drop to near zero?

    - Show “No profiles match your dealbreakers right now”
    - Provide one-tap actions: “Turn off dealbreaker” / “Widen range”

Critical: never silently violate a dealbreaker to fill the feed. That destroys trust.

Mode 3: Paid strict filter / hide-me

This is highest risk because it can collapse both:

- what the user sees
- and/or who can see the user (depending on how you implement “hide me”)

Recommended rule: require a minimum estimated supply before allowing strict mode.

- Example policy: “Strict filter requires at least N estimated matches/week (or month).”
- If below threshold: allow enabling but warn strongly OR block with explanation.

PRD must clearly specify:

- Is strict filter affecting “profiles I see,” “profiles that see me,” or both?
- What warning is shown if it will hide them from most people?
- What the system does when strict mode produces empty feed (don’t auto-relax unless user explicitly agrees).

---

4) UX nudges for “unlikely” preferences (don’t judge; guide)

You should not enforce moral logic like “divorced can’t prefer never married.” That’s a user choice. But you *should* nudge based on outcomes.

Trigger nudges when:

- Estimated availability becomes “Low”
- A dealbreaker toggle drops availability sharply
- Strict filter hides user from >X% of the audience

Nudge examples (copy direction)

- **Kerala → Venezuela:** “We currently have limited profiles in Venezuela. Consider adding nearby locations to see more people.”
- **Divorced → never married dealbreaker:** “This may significantly reduce results. You can keep it, or broaden to ‘Never married + Divorced’.”
- **Age 23 → 48:** “This preference may reduce matches. Consider widening your age range to increase results.”

Make it about **supply**, not about whether it “makes sense.”

5) System protections (to prevent abuse + confusion)

A) Prevent “I’ll game the cap by changing PP”

You already raised this; tie it here too:

- Free cap counts **unique profiles revealed** in 12h
- PP changes re-rank remaining unseen, but don’t unlock more than 50 new profiles

B) Preference change rate limit (optional)

To reduce thrash + gaming:

- apply a small cooldown for repeated edits (not for normal use)
- or only recompute feed after “Save preferences”

C) Explainability

Add a “Why am I seeing this?” (even minimal) so when someone sees “outside PP,” they don’t assume broken behavior:

- “Shown because it’s a strong match”
- “Shown to keep your results flowing”

---

6) What to explicitly add to your PRD (section checklist)

Create a subsection called **“Invalid / Low-supply Preference Handling”** with:

1. **Validation rules (hard blocks)**
2. **Supply estimation logic**
    - when computed
    - what thresholds define low/medium/high
3. **Fallback behavior**
    - soft PP blending fallback
    - dealbreaker empty-state handling
    - strict filter enablement constraints
4. **UX states + copy**
    - low supply warning
    - empty feed state
    - strict filter warning
5. **Metrics / guardrails**
    - empty feed rate after PP changes

   - % users hitting low supply
   - dealbreaker enable → supply drop distribution
   - strict filter enable rate vs churn/engagement
6. **Logging**
   - preference_saved
   - supply_bucket_shown
   - dealbreaker_enabled
   - empty_feed_reason

My recommended stance in one line (good for alignment)

Allow any “valid” preference, but make the system supply-aware: warn early, never break dealbreaker guarantees, and gate strict mode if it would reliably lead to zero results or extreme hiding.

If you tell me one thing—**does your app actually have meaningful inventory outside India (e.g., Venezuela)?**—I can suggest whether to treat “country” as a real selectable preference or as a “low-supply warning + nearby suggestion” case by default.