Part 1: Product Requirements Document (PRD)

1. Executive Summary

Curate is a decision-support intelligence layer for the physical dining experience. It solves the "paradox of choice" and "dietary anxiety" by transforming static, overwhelming restaurant menus into personalized, safe, and taste-optimized meal plans.

Problem Statement

- **Decision Paralysis: Diners spend 10–15 minutes decoding menus, often resulting in "safe" but boring choices.**
- **Dietary Risk: Users with allergies or strict diets (Vegan, Halal) experience anxiety, relying on server knowledge which varies in reliability.**
- **The "Missed" Dish: Users often realize too late they missed a signature dish that perfectly matched their tastes.**

Success Metrics (KPIs)

1. **Conversion Rate: % of recommendations accepted/ordered (validated via post-meal survey).**
2. **Safety Score: 0 reported allergic incidents (Critical).**
3. **Time-to-Decision: Reduction in time from app open to final decision compared to baseline.**
4. **NPS: User satisfaction score (1-10) post-meal.**

2. Scope & Roadmap

Feature Area	MVP (Launch)	V1 (Growth)	V2 (Scale)
Profile	Explicit data entry (Tags, Allergies, Budget).	Inferred preferences from history.	Biometric/Health app integration (optional).

Feature Area	MVP (Launch)	V1 (Growth)	V2 (Scale)
Ingestion	Restaurant Portal (Manual Entry + CSV).	OCR Menu Scanning for Restaurants.	POS Integration for real-time 86'ing (out of stock).
Recs	Rule-based filtering + Semantic Ranking.	"Why this?" LLM Explanations.	Group Mode (Consensus ordering).
Output	Single Course & Simple Bundles.	Multi-course dynamic pairing.	Wine/Drink pairing logic.
Business	Free.	Premium Subscription.	Dynamic Pricing/Yield Management.

3. User Journeys

Journey A: The Diner (In-Restaurant)

1. **Trigger: User sits at a table at "Bistro 42."**
2. **Context: User scans a QR code or selects "Bistro 42" from the geolocation list.**
3. **Input: User selects hunger level ("Snack" vs "Feast") and current mood ("Adventurous" vs "Comfort").**
4. **Processing:**
    - **System pulls the live menu.**
    - **Hard Filter: Removes all pork (Halal) and Peanuts (Allergy).**
    - **Soft Filter: Deprioritizes high-carb items (User preference).**
    - **Rank: Matches spicy flavor profile to user's "High Heat" tolerance.**
5. **Output: App displays 3 distinct "Bundles":**

    - ***The Chef's Tour (Appetizer + Main + Dessert)***
    - ***The Light & Spicy (Main + Drink)***
    - ***Value Pick (Main only)***
6. **Action: User shows screen to waiter or orders.**
7. **Feedback: Post-meal notification: "How was the Spicy Tuna?" (Rated 1-10).**

Journey B: The Restaurateur (Setup)

1. **Upload: Chef logs into the web portal.**
2. **Entry: Uploads menu items.**
3. **Tagging: System suggests tags via NLP; Chef confirms: Contains: Gluten, Spice: 3/5, Chef's Choice: Yes.**
4. **Promotion: Chef marks "Sea Bass" as a *Seasonal Special*.**
5. **Result: The Sea Bass is boosted in ranking, *only* for users who like fish and fit the budget.**

---

4. Functional Requirements

FR-01: User Profile Engine

- **Inputs: Strict constraints (Celiac, Nut-free, Vegan, Halal, Kosher), Dislikes (Cilantro, Blue Cheese), Spice Scale (0-5), Budget ($/$$/$$$).**
- **Validation: Mutually exclusive logic checks (cannot be "Vegan" and "Paleo" if conflicts arise).**

FR-02: Safety & Filtering (The "Red Line")

- **Requirement: Any dish containing a user's specific allergen must be hidden or clearly flagged as "UNSAFE" in red UI.**
- **Acceptance Criteria: A "Peanut Allergy" user profile *must never* see a Satay dish in a recommended bundle.**

FR-03: The Recommendation Engine

- **Logic: Must support "Bundling."**
    - ***Rule:* Bundle Price $\le$ User Budget.**

    - ***Rule:*** **Bundle Calories (approx) $\approx$ Portion preference.**

- **Explanations: User clicks "Why?" $\rightarrow$ System outputs: "Recommended because you love *Umami* flavors and rated *Miso Glazed Cod* highly last week."**

FR-04: Restaurant Portal

- **Promotions: "Smart Boost." Restaurants can boost items, but the AI weights user preference higher. If a user hates mushrooms, a promoted mushroom risotto is *not* shown.**

Part 2: Technical Design Document (TDD)

5. System Architecture

The system follows a **Event-Driven Microservices** architecture to separate the latency-sensitive recommendation engine from the data-heavy menu management system.

Core Components

1. **Client (Mobile):** Flutter (iOS/Android). Local caching of active restaurant menus for offline resilience.
2. **API Gateway (GraphQL):** Aggregates requests for User Profile and Menu Data.
3. **Menu Service (Restaurant Platform):** Postgres DB. Handles CRUD for dishes, pricing, and availability.
4. **Profile Service:** Manages dietary DNA and taste vectors.
5. **The "Sommelier" (AI Engine):** Python/FastAPI service containing the Logic & ML pipeline.

---

6. AI & ML System Design

This is the core differentiator. We avoid generic LLM calls to prevent hallucinations. We use a **Retrieval Augmented Generation (RAG) + Constraint Satisfaction Problem (CSP)** approach.

6.1 Data Representation (Embeddings)

- **Dish Vectors:** Every dish is converted into a vector embedding using a fine-tuned food-specific model (e.g., BERT-Food).
    - *Text:* "Spicy Tuna Roll with jalapeno and unagi sauce."
    - *Tags:* [Spicy, Fish, Sushi, Gluten]
- **User Vectors:** User history and preferences are aggregated into a "Taste Vector" in the same latent space.

6.2 The Pipeline (Runtime Flow)

Step 1: The Hard Guardrails (SQL/Filter)

- *Input:* Full Restaurant Menu (100 items).

- *Process:* Filter WHERE allergen NOT IN user.allergens AND diet_type = user.diet.
- *Output:* Safe Menu (60 items).

Step 2: Semantic Ranking (Vector Search)

- *Process:* Cosine Similarity between User_Taste_Vector and Dish_Vectors (subset of Safe Menu).
- *Adjustment:* Apply weights for "Chef Specials" (Business Logic).
- *Output:* Ranked List of Candidate Dishes.

Step 3: Bundle Optimization (Combinatorics/CSP)

- *Algorithm:* Knapsack Problem variant.
- *Objective:* Maximize Total Taste Score.
- *Constraint:* Sum(Price) $\le$ User Budget.
- *Constraint:* Category Diversity (1 App, 1 Main, 0-1 Side).

Step 4: Explanation Generation (LLM)

- *Model:* GPT-4o-mini or Llama-3 (Low latency).
- *Prompt Strategy:*

"You are a culinary expert. Explain why [Dish Name] fits [User Preferences]. Do not invent ingredients. Use the provided ingredient list: [Ingredients]. Keep it under 20 words."

6.3 Feedback Loop (Reinforcement Learning)

- **Explicit:** User rates meal 1-10.
- **Implicit:** Dwell time on explanation.
- **Update:** If user rates "Spicy Tuna" a 10, the User Vector is updated to move closer to the "Spicy" and "Raw Fish" centroids.

---

7. Data Models (Conceptual)

User_Profile

JSON

```
{
  "user_id": "uuid",
  "constraints": ["vegan", "gluten_free"],
  "allergens_strict": ["peanuts", "shellfish"],
  "spice_tolerance": 4, // 0-5
  "budget_setting": 2, // 1: Cheap, 2: Moderate, 3: Expensive
  "taste_embedding": [0.02, -0.45, ... ] // Vector
}
```

8. Non-Functional Requirements

- **Latency:** Recommendation generation must complete in **< 1.5 seconds**. Pre-calculate embeddings upon menu upload, not at runtime.
- **Availability:** 99.9%. If AI fails, fallback to "Most Popular" safe items (Hard filters still apply).
- **Security:** PII (User diet data) is sensitive health data. Encrypted at rest (AES-256) and in transit.
- **Observability:** Log every "Block" event. If a user sees a dish that violates their allergy profile, this is a P0 incident.

---

9. Risk Analysis

Risk	Impact	Mitigation
Allergy Hallucination	Severe (Health)	Do not use LLMs for allergy filtering. Use deterministic code/tagging only. Mandatory "Ask Staff" disclaimer on UI.
Menu Staleness	Medium (User Trust)	Timestamp menus. If menu > 48 hours old, warn user "Menu may be outdated."
Cold Start	High (UX)	Onboarding quiz to initialize User Vector. "Generic Popular" fallback for new restaurants.

---

10. MVP Build Plan

Phase 1: Foundation (Weeks 1-4)

- Setup DB (Postgres + Pinecone).
- Build Restaurant Portal (Menu Upload).
- Build User Profile Onboarding Screens.

Phase 2: The Engine (Weeks 5-8)

- Implement "Hard Guardrail" logic (Filter).
- Implement Vector Embedding pipeline (OpenAI Embeddings or HuggingFace).
- Develop Bundling Algorithm.

Phase 3: Mobile Client & Alpha (Weeks 9-12)

- Flutter App implementation.
- Integration with 5 Pilot Restaurants.
- Internal testing for Allergy Safety (Red Teaming).

Phase 4: Launch (Week 13)

- Release to App Store.
- Launch with "Scan QR" functionality at pilot locations.