# Pocket Pallet – Design & Build Reference

## Purpose / End State

Pocket Pallet empowers drinkers to make informed decisions when buying natural wine.

It gives them confidence that what they open will align with their taste — or that any risk they take is deliberate and rewarding.

## Users & Environment

Pocket Pallet is made for people who love natural, small-producer wine and drink it regularly.

They’re curious and discerning but not experts. Today, they rely on Vivino, friends, or whoever’s behind the counter. The app replaces that mix of guesswork and social dependence with something more personal, data-backed, and quietly expert.

## Behavioral Change

Instead of making uninformed decisions, users move through the world with greater confidence in the wines they choose.

They pick bottles knowing why they’ll enjoy them — and when they experiment, they do it on purpose.

## Tensions

Pocket Pallet balances the precision of data with the intuition of taste. Its design favors momentum over verbosity, trust over transparency, and consistency over endless choice.

| Axis | Balance |

| :--- | :--- |

| Accuracy ↔ Speed | Both must be high. A wrong recommendation is worse than a slow one. |

| Simplicity ↔ Transparency | The system should feel clear without needing to explain itself. Users trust the outcome, not the algorithm. |

| Automation ↔ Human Curation | Machine learning handles the heavy lifting; human taste defines the edges. |

| Consistency ↔ Flexibility | A stable experience builds confidence, but subtle variation keeps discovery alive. |

## Design Principles

Pocket Pallet should feel like a calm, knowledgeable companion — confident but never performative. The design must scale seamlessly across iOS and Android, preserving warmth, speed, and clarity in both ecosystems.

1. Confident, never pushy. Feels like a trusted friend recommending a bottle, not an app selling one.

2. Positive by design. Language and visuals celebrate taste; there’s no “wrong” preference.

3. Frictionless motion. Transitions and micro-animations feel organic, not ornamental — quick swipes, one-tap saves, clean modals.

4. Familiar, not generic. Native gestures and UI conventions of each OS are honored; the brand’s personality comes from tone and texture, not novelty.

5. Human in the loop. Curated notes, personal insights, and small human touches appear throughout to keep the experience grounded.

6. Quiet confidence. Minimal visual noise; typography and color carry emotion, not decoration.

## System Vision

Pocket Pallet isn’t a database or a recommender. It’s a living palate that grows alongside you.

It lives on your phone, goes to dinner with you, helps you shop, remembers what you liked last week, and quietly learns your taste over time.

The system should feel less like “logging” and more like a conversation — brief, contextual, and responsive wherever the user happens to be.

Whether someone’s scanning a bottle in a store, browsing a list at a restaurant, or planning a delivery from home, Pocket Pallet offers just the right amount of guidance — never interrupting, always available.

### Core Interaction Loop

1. Observe — The user captures or selects a wine (photo, label scan, or search).

2. Interpret — The system identifies the wine, checks provenance, and predicts taste alignment.

3. Reflect — The user records how they felt about it — quick, emotional, natural language.

4. Adapt — Pocket Pallet refines its understanding of their palate and future suggestions.

### Design Implication

The app must work in motion — at the table, in a store, on the go.

It needs short loops, offline tolerance, fast camera recognition, and gentle cues for when to re-engage (like recommending a similar bottle next week).

## Core Modules

Pocket Pallet is built around four living systems that learn, guide, and respond to the drinker. Each module feeds the others — taste, context, and curation form a single loop.

| Module | Role | Description |

| :------ | :---- | :----------- |

| Palate Engine | Learns | Builds a taste fingerprint from user ratings, notes, and behavior. Balances structured data with emotional input (“bright,” “funky,” “feels like summer”). |

| Journal | Reflects | Quick-entry memory layer for photo, note, vibe — improves Palate Engine and archives experiences. |

| Discovery | Inspires | Suggests bottles, producers, or regions aligned with taste or mood. Context-aware and situational. |

| Companion View | Guides | Lightweight in-store/dinner mode — label recognition, scan-to-match, and “buy again.” |

| Curation Layer | Grounds | Human-managed database blending public data and curated entries. |

| Social Loop | Connects | Optional sharing of favorites or discoveries with friends — “who drinks like me.” |

\*\*System Behavior:\*\*

Each module runs semi-autonomously but contributes to a shared palate profile on the backend.

The Palate Engine is the spine — every touchpoint updates it.

The Journal and Companion View provide data density; Discovery and Social provide delight.

---

# Appendix – Pocket Pallet Build Specification

## System Architecture

- Frontend: Vite + React + TanStack Query + Tailwind

- Backend: FastAPI + SQLAlchemy + Alembic

- Database: PostgreSQL

- Storage: S3 / Supabase

- Async Jobs: Celery / RQ

- Auth: Clerk / JWT

- Infra: Railway (backend + DB) + Vercel (frontend)

## Data Model (v1)

| Table | Key Fields |

| :----- | :---------- |

| wines | id, producer, cuvée, vintage, region\_id, grape\_blend, price, notes |

| regions | id, name, country |

| sources | id, name, type (csv, api, manual) |

| imports | id, source\_id, file\_name, created\_at |

| import\_rows | id, import\_id, raw\_json, status, error\_msg |

| duplicates | id, wine\_a, wine\_b, confidence\_score |

## Milestones

| Phase | Deliverable | ETA |

| :----- | :----------- | :--- |

| MVP | CSV upload → parsed → DB insert | Weeks 1–3 |

| Phase 2 | Review Queue UI + merge endpoints | Weeks 4–5 |

| Phase 3 | Version history + rollback | Weeks 6–7 |

| Phase 4 | Source priority + freshness scoring | Weeks 8–9 |

| Phase 5 | Analytics dashboard + search | Weeks 10–11 |