Product Requirements Document (PRD)

Product Name: The Tenant's Voice

Version: 2.1 (Post-Optimization)

Status: Live / Iterating

Owner: Product Lead

1. Problem Statement

The UK rental market suffers from a severe power and information asymmetry. When disputes arise—such as deposit theft or disrepair—tenants face opaque, intimidating legal processes. The "Frustrated Renter" is often tech-savvy but lacks legal expertise and cannot afford professional counsel. Existing solutions are either static, dense government guides (hard to parse) or expensive solicitors (hard to afford).

2. Product Vision

To empower UK tenants with instant, accurate, and actionable legal guidance, enabling them to resolve disputes formally without financial barrier.

3. User Persona

- **Name:** The Frustrated Renter
- **Attributes:** Tech-savvy, anxious about housing security, financially constrained, time-poor.
- **Core Need:** "I need to know my rights and exactly what email to write, right now, without paying a lawyer."

4. Functional Requirements

4.1. Core Experience: Recursive Confidence Loop

- **Logic:** The system must not output advice immediately upon receiving a user query. It must first assess its own "understanding confidence."
- **Threshold:** If confidence < 95% regarding the user's specific legal situation, the system triggers a **Clarification Mode**.
- **Behavior:** The model generates targeted follow-up questions (e.g., "Was the deposit protected within 30 days?" or "Do you live with your landlord?") to narrow down the scenario.
- **Termination:** Advice is generated only once the user provides sufficient context to meet the confidence threshold or after 3 clarification turns to prevent infinite loops.

4.2. Information Retrieval (RAG Pipeline)

- **Query Transformation:** All user queries must be rewritten by an LLM into "idealized search queries" to maximize keyword overlap with legal texts.

- **Hybrid Search:** Retrieval must utilize both semantic search (Vector) and keyword filtering (GIN index) to ensure coverage of specific legal terminology.
- **Freshness Filter:** The retrieval engine must prioritize documents based on a `priority_date` metadata field to prevent citation of superseded laws.

4.3. User Interface

Visual Cues:

- **Input Optimization:** The chat input must be an auto-resizing textarea that captures multi-line context. Submission shall be triggered by `Ctrl+Enter` (desktop) or a distinct button, preventing accidental partial submissions.
- **Load Performance:** The application must load instantly with zero bundle fatigue; no heavy frontend frameworks (React/Vue) shall be used.

4.4. User Interface: Framing & Safety

- **Positioning:** The UI must explicitly frame the tool as a **"Process Guide"** rather than a "Legal Advisor."
- **Visual Cues:**
    - Standard legal disclaimers must appear as static footnotes on every chat view.
    - "Advice" outputs must use soft language (e.g., "The standard process is..." rather than "You must...").

5. Non-Functional & Technical Requirements

5.1. Performance & Latency

- **End-to-End Latency:** The system must deliver a full response in under 4 seconds (Optimized from 40s+).
- **Database Search:** Vector query execution time must not exceed 50ms.

5.2. Privacy & Security

- **Stateless Architecture:** The server must not persist any chat logs or user data after the request is processed. All context must be passed ephemerally from the client.
- **Anonymity:** No user login, email collection, or cookies required for core functionality.

5.3. Reliability & Stability

- **Context Limit Handling:** The system must implement a "Dual-History" approach: submitting only `recentHistoryText` for embeddings (avoiding the 36k-byte crash limit) while passing `fullHistoryText` to the LLM for context.
- **Error Handling:** The frontend must gracefully handle non-JSON responses or timeouts with a fallback message rather than a crash.

6. Evaluation Criteria (Success Metrics)

- **Primary Metric (North Star): Completion Rate**. Percentage of sessions where the user receives a "Next Step" (e.g., drafts an email) rather than dropping off.
- **Latency Metric:** Average Time to First Token (TTFT) < 2 seconds; Total Response Time < 4 seconds.
- **Quality Metric: Hallucination Rate < 4%**. Verified via human-in-the-loop (HITL) testing against a "Golden Set" of known legal queries.
- **Operational Metric: Cost per Query**. Must remain < $0.01 to maintain the "Free Forever" promise.

6.1. Knowledge Base Maintenance

- **Ingestion Pipeline:** Updates to the vector store are triggered manually.
    - **Source Acquisition:** Manual retrieval of HTML/PDFs from official sources (Gov.uk, Shelter).
    - **Processing:** Local Python tooling handles HTML cleaning, chunking, embedding generation, and keyword extraction.
    - **Upload:** Cleaned vectors are pushed to the production database via the admin script.
- **Frequency:** Ad-hoc basis, triggered by major legislative updates (e.g., Renters Reform Bill passing).

Product Critique & Analysis

As a critical product lead, I have reviewed the above PRD and challenged its validity with the following 5 questions. This process ensures the document moves from "wishful thinking" to "engineering reality."

1. Is "Free & Anonymous" actually a sustainable product strategy?

- **Critique:** A product with zero revenue and costs that scale with usage (LLM tokens + Vector DB) is a financial time bomb. "Mission-driven" doesn't pay for Supabase overages.
- **Resolution in PRD:** We emphasized the "Architecture Choice" of Gemini Flash (low cost) and Supabase (integrated vector usage) to minimize burn. However, the PRD should explicitly list a "donation" or "grant funding" exploration in the roadmap to ensure server longevity without compromising the user promise.

2. Is "Stateless" a lazy excuse for poor UX?

- **Critique:** Refusing to store history means a user cannot switch devices, refresh the page, or reference a case they started yesterday. This friction might hurt the "Frustrated Renter" who needs to build a case over weeks.
- **Resolution in PRD:** We accepted this tradeoff explicitly for **Trust**. We added the client-side context requirement so the *current* session is smart. To fix the "weeks" issue without server storage, we could add a "Download Chat / Export to PDF" feature so the user owns their data.

3. Why Vanilla JS? Is this just developer ego?

- **Critique:** "No frameworks" sounds pure, but as the product grows (e.g., adding the P1 Council Finder), spaghetti code becomes a risk. React/Next.js offers better state management for complex multi-step forms.
- **Resolution in PRD:** The PRD validates this via the "Single Page, Single Function" scope. Since the app is strictly a chat interface, the overhead of React is unjustified. If scope expands to a "Dashboard," the stack will be re-evaluated.

4. Are we over-optimizing for latency at the cost of accuracy?

- **Critique:** The optimization log shows we removed a "Redundant AI Step" (Pre-processing) to save 27 seconds. That step likely added robustness or safety checks. Removing it relies heavily on the vector search being perfect.
- **Resolution in PRD:** The decision was validated by HITL (Human-In-The-Loop) testing which confirmed vector-only search matched hybrid quality. We formalized "Hallucination Rate < 4%" as a hard metric to prevent speed from degrading trust.

5. Legal Liability: What if the AI is wrong?

- **Critique:** Giving "legal guidance" is high-risk. If a tenant loses their deposit because the AI misquoted the Housing Act 2004, who is liable?
- **Resolution in PRD:** The output requirements include "Citations". We must also add a rigid **Disclaimer Requirement** to the UI (e.g., "This is information, not legal advice") and ensure the system favors "refer to Citizens Advice" over inventing answers when confidence is low.
