RentLedger — Technology Stack Document

Flutter-First, Trust-Critical Application

1. Product Context

RentLedger is a **system of record for rentals**, designed to create **verifiable, tamper-evident rental timelines** usable by tenants, landlords, brokers, and housing societies.

This is a **trust-first application**, where:

- Auditability > speed
- Consistency > cleverness
- Clarity > visual flair

The tech stack prioritizes:

- Long-term maintainability
- Legal defensibility (India-first)
- Global scalability
- Institutional confidence

2. High-Level Architecture

```
Flutter App (Mobile + Web)
        ↓
REST API (NestJS)
        ↓
PostgreSQL (System of Record)
        ↓
AWS S3 (Immutable Evidence Storage)
        ↓
BullMQ (Exports, OCR, Notifications)
```

Frontend is Flutter-first.
Backend exists to **serve trust**, not UI logic.

3. Frontend (Primary Focus)

3.1 Framework

Flutter (Dart) ✅

Why Flutter

- Single codebase: Android, iOS, Web
- Predictable rendering (important for legal UI)
- Strong long-term Google backing
- Excellent performance on low-end devices (India)

Flutter Version

- Stable channel only
- Lock version in repo (no auto upgrades)

3.2 Architecture Pattern

Clean Architecture (Mandatory)

```
presentation/
domain/
data/
core/
```

Principles

- UI has no business logic
- Domain is framework-agnostic
- Data layer owns API & persistence

This prevents **trust logic leaking into UI**.

3.3 State Management

Riverpod (recommended)
(Bloc acceptable if team prefers)

Why

- Compile-time safety
- Testability
- Explicit dependency graph

- No hidden magic

3.4 Routing & Navigation

- `go_router` or equivalent
- Role-aware navigation guards
- Explicit route permissions

No implicit routing.

4. UI / UX Stack

4.1 Design System

Material 3 — Heavily Customized

- Neutral color palette
- Institutional typography
- No playful components
- No animated transitions for trust actions

4.2 Typography

- **Inter** (Primary)
- System font fallback

Rules:

- Body: 15–16px
- Headings: weight, not size
- High contrast, high readability

4.3 Component Library

Custom-built Flutter components:

- Timeline list
- Event cards (immutable)

- Evidence viewer
- Export modal
- Role-aware dashboards

❌ No third-party UI kits

5. Backend (Support Layer)

Framework

- **NestJS (TypeScript)**

Purpose

- Enforce trust rules
- Manage timelines
- Generate certified exports
- Handle access control

The backend **owns integrity**, not the frontend.

6. Database Layer

Primary Database

PostgreSQL ✅

Why

- ACID compliance
- Strong relational modeling
- Append-only patterns
- Legal/audit friendly

Design Rules

- UUIDs for all entities
- Immutable event tables
- No soft deletes

- Full audit logs

7. Evidence & Media Storage

Object Storage

AWS S3

Configuration

- Versioning enabled
- Object immutability (where possible)
- Metadata preserved
- Pre-signed upload URLs

Frontend **never** talks directly to S3 without backend authorization.

8. Evidence Integrity Layer

Hashing

- **SHA-256**
- Client computes preview hash
- Backend computes authoritative hash
- Hash-chained per rental timeline

❌ No blockchain
❌ No marketing buzzwords

9. Certified Record Export System

Export Format

- Deterministic PDF bundles
- Timeline summary
- Event hashes
- Metadata

- Signature block

Legal Compliance

- Section 65B IT Act (India)
- Language: **“Certified record export”**

10. OCR & Document Processing

OCR Engine

- **Tesseract OCR (open-source)**

Usage

- Document indexing only
- No AI interpretation in MVP

Triggered via backend jobs.

11. Background Jobs & Queues

Queue System

- **BullMQ (Redis)**

Used For

- Evidence exports
- OCR processing
- Notifications
- Hash verification

Frontend polls job status via APIs.

12. Authentication & Authorization

Authentication

- JWT (short-lived)
- Refresh tokens

- Secure storage in Flutter

Authorization

- Role-Based Access Control (RBAC)
- Server-side enforced
- UI mirrors backend permissions

Roles:

- Tenant
- Landlord
- Broker
- Society Admin
- System Admin

13. Notifications

Channels

- Push: Firebase Cloud Messaging
- Email: SES / SendGrid
- SMS: India-compliant provider

Frontend only displays status — no logic.

14. Security & Compliance

Frontend Security

- HTTPS only
- Secure storage
- Screenshot blocking (where applicable)
- No client-side trust logic

Compliance

- India data residency awareness