Marketplace — Structure & Ops (Canonical Reference)

Version v1.0 • 2025-09-30

This document is the canonical reference for the Tenant–Landlord Marketplace repository structure, runtime operations,  
databases, admin roadmap, and operations practices. It aligns with the FRD (Single Source of Truth) and the Illustrated FRD.  
Sources: FRD\_Tenant\_Landlord\_Marketplace\_Complete\_v0.2.docx; FRD\_Illustrated\_vFinal.docx.

# 1. Repository Structure

Primary goals: predictability, Synology-friendly layout, clean separation of concerns, and ops-first design.  
Top-level layout:  
  
/ (repo root)  
 /public → Static assets & Admin UI (single-skin light+golden)  
 /images → Shared images (logos, banners)  
 /images/properties → Property & unit photos  
 /admin → Admin HTML + JS + CSS (theme: admin.css)  
 /myaccount → Tenant/Landlord self-service pages  
 /routes → Express route modules (REST under /api/\*)  
 /scripts → Ops, cron jobs, migrations, backups  
 /data → Databases & storage (dev/uat/prod subfolders or symlinks)  
 /logs → Rolling logs  
 /docs → FRD, Structure & Ops, ADRs, diagrams  
 .env → Environment config (never commit)  
 server.js → App entrypoint (mounts routes, static, health)  
  
Conventions:  
- Single admin CSS skin (admin.css) with “light + golden” tokens.  
- Cache-busting for static files via query tokens (?v=YYYYMMDDhhmm).  
- ADR files in /docs/adr (e.g., ADR-001\_API\_Conventions.md).  
- Use semantic versioning for releases and tag each deploy.

# 2. Environments & Ports

Environments: dev, uat (stg), prod — all independent, each with its own DB paths and logs.  
Suggested ports (configurable via .env):  
- API: 3101  
- Admin static: 3102 (optional; can be served by Express from /public)  
- Health: exposed under /api/health on the API port  
  
Runtime flags (.env examples):  
PORT=3101  
ADMIN\_STATIC\_PORT=3102  
NODE\_ENV=development|staging|production  
DB\_PATH=data/dev/marketplace.dev.db  
SESSION\_SECRET=change\_me  
CF\_TUNNEL=enabled|disabled  
PDF\_ENGINE=puppeteer

# 3. Database Strategy

Dev/UAT: SQLite (file-based, fast iteration). Prod: SQLite acceptable initially; plan migration path to Postgres.  
Overlay + History pattern (auditability and safe evolution):  
- Base tables (e.g., leases, invoices, payments) remain normalized.  
- Overlay tables hold mutable summary fields (e.g., current\_rent\_cents).  
- History tables track status and financial events (e.g., lease\_status\_history, ticket\_history, payment\_refunds).  
  
Migrations:  
- Store SQL migrations under /scripts/sql with a migration ledger (migration-ledger.md).  
- Use idempotent CREATE IF NOT EXISTS; add indexes.  
- Apply migrations in CI or via /scripts/migrate.sh (logs show applied versions).

# 4. Payments (P1 Canonical Ops)

Scope: Split payments, Auto-pay, Refunds, Automated Late Fees are mandatory in P1.  
Key operations:  
- Split Payments: one invoice ← many payment rows; maintain allocation per charge line.  
- Auto-Pay: scheduled job attempts charge on due date; retries; consent records.  
- Refunds: create refund rows linked to original payment; audit who/when/why; adjust balances.  
- Late Fees: nightly job applies rules (grace period, fixed/%, caps) and posts penalty entries.  
- Reconciliation: import bank statements, auto-match by ref/amount/date/payer; unresolved → Unassigned queue.  
- Receipts: email/SMS receipts on successful payments/refunds; PDF with signed URL.

# 5. Health, Monitoring, Backups

Health endpoint:  
GET /api/health → { ok, uptime, checks: { db: "ok"/"fail", storage: "ok"/"fail" }, versions, env }  
- DB check: open DB, simple SELECT 1  
- Storage check: read/write temp file under /data  
- Include overlay/history counts for quick diagnostics  
  
Logging & Monitoring:  
- Access logs (combined) and app logs in /logs with daily rotation (rotate\_logs.sh).  
- Structured JSON for key events (payments, refunds, late fees, auth changes).  
- Alerting: basic email for health failures; future webhook integration.  
  
Backups:  
- /scripts/backup\_sqlite.sh dumps SQLite DBs to timestamped files with retention policy.  
- Encrypt backup archives; verify restore quarterly (table counts & checksum).  
- Include /data/uploads (if any) and /docs releases in backup plan.

# 6. Security & Privacy Operations

- Role-based access middleware (requireAuth, requireRole).  
- Short-lived session cookies (HttpOnly, Secure, SameSite=Lax/Strict).  
- Server-side rendering for sensitive views; minimize browser-readable business logic.  
- Secrets in .env only; never commit. Provide /scripts/env\_example.sh.  
- Audit trail: who/when/what/before/after for admin & finance edits.  
- Document access via signed URLs with expiry; watermark sensitive PDFs.  
- Data retention and tenant erasure workflow (subject to law and business policy).

# 7. Deploy & Runtime Operations (Synology)

Start/Stop/Update scripts:  
- scripts/start.sh → loads .env, runs node server.js via nohup or DSM Task Scheduler  
- scripts/stop.sh → finds PID and terminates gracefully  
- scripts/update.sh → git pull, install, migrate, restart; creates a pre-update backup  
- scripts/status.sh → prints ports, PIDs, /api/health, DB paths, overlay counts  
  
Cloudflare Tunnel:  
- One tunnel per environment; hostname per env (api-uat, api).  
- Health dashboard page includes CF edge check + /api/health.  
  
Release Management:  
- Tag every deploy (e.g., marketplace-v1.0.0); keep CHANGELOG.md.  
- Canary on UAT before Prod; rollback via previous tag & DB restore script.

# 8. Admin UI & Static Delivery

- Single CSS skin (admin.css) with light+golden theme; consistent .btn classes and status chips.  
- Partials for Dashboard, Properties, Tenants, Financials, Utilities, Maintenance, Reports, Settings.  
- Pagination, filter bars in one row; modals for Add/Edit/View; table actions aligned to the right.  
- Cache-bust static assets with ?v= timestamps; same-origin fetch only.

# 9. API Conventions & Versioning

ADR-001 conventions (summary):  
- REST under /api; plural resources; consistent { success, data, error } envelope.  
- Pagination: ?page=&limit= returns { total, page, limit, data }.  
- Validation errors use error.code and error.details[].  
- Versioning: default /api/v1; additive changes only; breaking changes require /v2 with migration notes.

# 10. Admin Roadmap (Ops-Facing)

Sprint 0 – Repo & Runtime: repo skeleton, start/stop/update, health, backups, admin theme scaffold.  
Sprint 1 – Properties & Units: CRUD + images + listings basics, audit hooks.  
Sprint 2 – Tenants & Leases: onboarding, tenant CRUD, lease lifecycle + history.  
Sprint 3 – Invoices & Payments (P1): auto-invoice, split payments, auto-pay, refunds, late fees, receipts.  
Sprint 4 – Utilities & Maintenance: utilities bulk upload & validate; maintenance tickets + history.  
Sprint 5 – Financials & Reports: statements, arrears, income vs expenses, exports.  
Sprint 6 – Notifications & Messaging: queue + templates, email baseline, SMS optional.  
Sprint 7 – Reconciliation & Unassigned: bank import, auto-match, exceptions workflow.  
Sprint 8 – Admin Billing: platform fees for landlords/managers; billing and audit.  
Sprint 9 – Mobile Readiness: API hardening, signed PDF links, role-adaptive endpoints.  
Sprint 10 – Security Hardening: CSRF, headers, rate limits, audit deepening.  
Sprint 11 – UAT → Prod: staging readiness checklist, DR drills, go-live runbook.  
Sprint 12 – Postgres Path: design & pilot migration plan, performance profiling.

# 11. Checklists

Go-Live Readiness (excerpt):  
- [ ] /api/health green; CF edge OK  
- [ ] Backups working; restore test completed  
- [ ] Admin RBAC verified; audit trails populated  
- [ ] Payments P1: split, auto-pay, refunds, late fees running & logged  
- [ ] PDF receipts render with signed links  
- [ ] Incident contacts & runbooks published  
  
Staging (uat) Promotion:  
- [ ] Tag built; CHANGELOG updated  
- [ ] DB migrations applied; counts verified  
- [ ] Canary tests pass (happy paths)  
- [ ] Rollback plan validated

# 12. Appendices

Appendix A – ADR Index  
 • ADR-001 API Conventions  
 • ADR-002 Auth & Session  
 • ADR-003 Overlay + History  
 • ADR-004 Payments P1 (Split/Auto/Refunds/Late Fees)  
Appendix B – Scripts (examples)  
 • scripts/backup\_sqlite.sh, scripts/rotate\_logs.sh, scripts/migrate.sh  
Appendix C – Health Page Spec  
 • /status shows API health, CF edge check, DB paths, overlay counts, version