FRD – Tenant–Landlord Marketplace (Complete, Single Source of Truth)

Current Version: v0.2 • 2025-09-26

This document is the single source of truth for the Tenant–Landlord Marketplace functional requirements.  
It applies the “WattSun Reuse Pack” patterns (runtime, admin CSS, ADR-001, auth/session, overlay+history,  
image structure, ops, Cloudflare/health) while keeping the project fully independent from WattSun  
(separate repository, databases, ports, and roadmap).

# Document Control

|  |  |
| --- | --- |
| Document Name | FRD\_Tenant\_Landlord\_Marketplace\_Complete.docx |
| Version | v0.2 (Enriched) |
| Date | 2025-09-26 |
| Status | Authoritative – Single Source of Truth |
| Sources merged | Seed FRD v0.1, FRD\_Tenant\_Landlord\_Marketplace\_Complete.md, FRD\_Illustrated\_vFinal (annexed) |

# 1. Independence & Reuse Pack

This is a new project that only reuses conventions from WattSun (not code or data).  
- Separate repository & CI/CD  
- Separate dev/uat/prod databases & backups  
- Separate runtime ports (suggested: 3101 API, 3102 Admin static)  
- Separate Cloudflare tunnel & status checks  
- Separate roadmap & documentation  
  
The “WattSun Reuse Pack” is applied by default:  
- Runtime folder layout on Synology; /public, /routes, /scripts, /data, /logs  
- Single-skin light+golden admin.css; card layout; single-row filters; status chips; consistent .btn  
- ADR‑001 API conventions ({ success, data, error }, pagination, validation, consistent errors)  
- Auth/session with roles & CSRF; requireAuth/requireRole; hashed passwords  
- Overlay + history tables for auditability  
- Image/static organization with cache-busting  
- Ops scripts, health endpoint, Cloudflare ingress

# 2. Data Security & Privacy (Paramount)

This marketplace handles extensive private data (IDs, leases, payment references, balances, contact info).  
Security and privacy are first-class requirements:  
  
Non-functional controls (mandatory from P1):  
• Least-privilege, role-based access (Admin, Landlord, Tenant; optional Support Agent)  
• Server-side rendering for sensitive pages; minimize browser-readable business logic  
• No secrets or PII in client-side storage; short-lived session tokens; secure cookies  
• Strong password hashing; rate limiting on auth; basic anomaly monitoring  
• Field-level validation & output encoding to prevent injection/XSS  
• Audit trail for admin & finance actions; immutable history tables  
• Data retention & archival policy; right-to-erase workflow for tenants where lawful  
• Encrypted transport (HTTPS via Cloudflare Tunnel); at-rest encryption for DB backups  
• Document access controls (signed URLs with expiry for PDFs/images)

# 3. Actors & Roles

- Tenant – browse listings, apply, sign leases, pay, statements, maintenance tickets.  
- Landlord – manage properties/units/tenants, listings, approvals, leases, invoices & payments.  
- Property Manager – manages portfolios for multiple landlords.  
- Admin (Platform Owner) – global settings, subscriptions/fees, billing & audit.  
- Team Members – delegated roles (Viewer, Editor, Admin).  
- System – automation engine for invoicing, reminders, late fees, allocations, notifications.

## 4. Functional Requirements (Merged & Enriched)

This section consolidates the Markdown FRD and Illustrated vFinal.  
  
4.1 Pre‑Login & Onboarding  
- FR-0001 Verification on first login (email/SMS)  
- FR-0002 Role survey  
- FR-0003 Wizard: add property → units → tenants (skippable; progress tracked)  
  
4.2 Dashboard  
- Quick actions (reminders, record payment, add utilities); KPIs and expected vs received chart  
  
4.3 Property & Unit Management  
- Properties top-level; units; one active tenant per unit  
- Recurring bills with rent; penalties fixed/% rent/% balance; payment instructions in invoices  
  
4.4 Tenant Management  
- Required: name, ≥1 phone, National ID; optional email, deposits, notes, docs, move dates  
- Up to 10 phones; payer names; archive tenant (vacates unit); customizable account number  
- Actions: Add Invoice, Record Payment, Reminders/Messages, Statement, Transfer, Archive  
  
4.5 Applications & Screening (P1 Basic)  
- Public listings & Apply; application flow; landlord notes; admin override  
  
4.6 Leases  
- Draft→Active→Terminated/Expired; docs; status history; overlay for current rent/deposit  
  
4.7 Invoices & Payments (P1)  
Core: auto rent invoices (1st, no duplicates); manual/bulk; combine; payments & unmatched; balances & history  
P1 Enhancements (MANDATORY IN P1):  
- Split payments across one invoice  
- Auto‑Pay with consent and retries  
- Refunds (partial/full) with audit entries  
- Automated late fees (grace, fixed/percent, caps, waiver logic)  
- Allocation rules (FIFO with overrides)  
- Reconciliation (bank import & auto‑match, Unassigned bucket)  
- Receipts via email/SMS; printable PDFs  
  
4.8 Utilities  
- Per‑property readings; bulk upload; validation; prompt to invoice & notify  
  
4.9 Maintenance  
- Issues per property/unit; link to expenses; statuses; photos  
  
4.10 Financials & Expenses  
- Invoices, payments, expenses; income vs expenses report; monthly statements  
  
4.11 Settings  
- Company profile, Mpesa; auto‑invoice; auto‑reminders; alerts; SMS balance; team roles; billing; audit trail  
  
4.12 Reports  
- Tenant/property statements; arrears; expenses; MoM/YoY; export PDF/Excel  
  
4.13 Admin Platform Fees & Billing  
- Plans, listing/promotions, SMS bundles; charge landlords/managers; revenue reports; full audit

# 5. Cross-Platform & Mobile (Tenant/Landlord Apps)

Web P1: design mobile-first & responsive.  
Future: native/hybrid apps with role-adaptive features.  
- Tenant app: invoices, auto-pay, receipts, statements, maintenance with photos, notifications.  
- Landlord app: KPIs, approve applications, record payments, refunds (guarded), maintenance approvals, reminders.  
APIs must be versioned; no breaking changes for mobile clients.

# 6. Technology & Environments (Synology-friendly)

Backend: Node.js (Express), SQLite for Dev/UAT; path to Postgres.  
Frontend: Admin single-skin theme; minimize client-side business logic for sensitive flows.  
Background: cron/Task Scheduler; optional queues later.  
Hosting: Synology DS224+; ports 3101/3102; Cloudflare Tunnel.  
PDF: Server-rendered (Puppeteer/wkhtmltopdf) with signed, expiring links.

# 7. Non-Functional Requirements

Performance: p95 < 500ms LAN; pagination.  
Security: per Section 2; secure headers; CSRF where applicable.  
Reliability: daily backups; health checks; integrity verifications; rollback/roll-forward scripts.  
Observability: structured logs; audit trails for admin actions (who/when/what/before/after).

# 8. API Health & Ops

/api/health → { ok, uptime, checks: { db, storage } }.  
Scripts: backup\_sqlite.sh, rotate\_logs.sh; migration ledger; environment probes.

# Annexes

Annex A – ADR‑001 API Contract Conventions (to be expanded).  
Annex B – FRD\_Illustrated\_vFinal (Illustrations & UI walkthroughs). Kept as annex; figures referenced in Section 4.  
Annex C – Roadmap Outline (to be refined after FRD approval).