Property Management App – Project Design Document (PRD)

Owner/Requester: Gongjun Yan, Ph.D.

Version: 1.0

Date: 2026-01-13

1. Background & Vision

This project proposes a lightweight property management system optimized for small-to-mid sized landlords/property managers (10–200 units) who need fast rent follow-up workflows, proof-based maintenance tracking, and clean records for disputes and accounting.

The design is based on an existing real workflow captured in handwritten notes: rent collection follow-up (text/call/email), late fee notices, eviction prep paperwork, rent records per tenant/property, maintenance projects (bids + job site proof), and invoicing/payment.

2. Target Users & Scope

2.1 Target Users

- Primary: Landlord/property manager managing 10–200 rental units
- Secondary: Maintenance staff / contractors / vendors
- Optional: Tenants (limited portal for paying rent and receiving notices)

2.2 Non-goals (Out of Scope for MVP)

- Full double-entry accounting system (Buildium/AppFolio-level accounting)
- Full lease signing workflow and tenant screening (can be added later)
- Complex owner-investor reporting (portfolio/ownership splits)

3. Pain Points

3.1 Rent Collection Pain Points

- Late rent follow-up is repetitive and time-consuming
- Messages are spread across SMS, calls, email – no centralized timeline
- No consistent escalation process (Day 1 reminder vs Day 3 notice vs call log)
- Late fee calculations and notices are manual and error-prone

- Eviction paperwork preparation is stressful and not standardized

3.2 Maintenance / Projects Pain Points

- Hard to verify contractor work (no proof of time/location)
- Before/after evidence is inconsistent, leading to disputes
- Work logs and materials receipts are scattered
- Approval workflow for bids and invoices is informal and hard to audit

3.3 Records / Reporting Pain Points

- Need clear rent ledger per tenant/property for court and taxes
- Need monthly roll-up of income and delinquency
- Need exportable records (CSV/QuickBooks) without manual re-entry

4. Proposed Solution

Build a web-based property management platform with a mobile app shell (phone app) that focuses on four core workflows:

- Rent Chase Automation (communication + late fees + notices + logging)
- Rent Records (tenant/property ledger, payment history, statements)
- Maintenance Projects (bid → approve → work order → proof → invoice)
- Invoice & Payment (send invoice link, track payment status)

The key differentiator vs many existing apps is a 'proof-based workflow': GPS clock-in/out, address confirmation, timestamped photo gallery, and structured work logs – all tied to the project and invoice.

5. Competitive Comparison (High-Level)

Popular competitors (Buildium, AppFolio, DoorLoop, TenantCloud, Innago) provide broad all-in-one tools: rent collection, maintenance requests, portals, and accounting.

However, this project intentionally prioritizes speed and the most painful operational steps for a small team: rent escalation workflows, eviction prep templates, and maintenance proof.

6. Core Product Principles

- Workflow-first: one-click actions that match real landlord tasks
- Audit trail by default: every message, notice, and work proof is stored
- Mobile-native evidence capture: photos + GPS + timestamps must be easy
- Simple UI: optimized for a small team managing many units quickly
- Configurable policies: late fee rules, grace period, notice templates

7. Main Functions (Detailed Requirements)

7.1 Rent Collection Module

Goal: Minimize late rent and reduce manual effort via structured escalation and quick communication.

Key Features

- Tenant rent status dashboard (paid/unpaid/partial)
- One-click 'Rent Chase' sequences:
- Day 1: reminder text
- Day 3: late reminder / notice
- Call/voicemail log + follow-up text/email
- Final notice template
- Late fee policy rules per lease:
- Grace period (e.g., due 1st, late after 3rd)
- Late fee calculation (% or flat)
- Communication templates library (editable)
- Communication log: SMS/call/email stored in tenant timeline

7.2 Rent Records Module

Goal: Clean tenant/property ledger suitable for disputes, court filings, and accounting.

Key Features

- Tenant ledger: monthly rent charges, payments, fees, credits
- Property ledger: income + expenses view
- Payment tracking: method, date received, reference/check #
- Export: CSV; QuickBooks-ready format (future)
- Monthly statements (PDF) per tenant/property

7.3 Maintenance Projects Module

Goal: End maintenance disputes via proof-based workflows and structured work orders.

Key Features

- Create work order / project per property
- Bid request workflow: invite vendors, compare bids, approve
- Work schedule + assignment (staff/vendor)
- Job site proof:
- GPS-based clock-in/out
- Geofence around property address (recommended)
- Photo gallery with timestamp + categories: before/during/after
- Work log fields: hours, materials, notes
- Completion approval checklist

7.4 Invoicing Module

Goal: Generate and send invoices quickly, collect payments, and store receipts.

Key Features

- Invoice builder: customer, property, project link
- Line items: labor/material; tax; attachments (receipts/photos)
- Send invoice via email/SMS with pay link
- Invoice status: draft/sent/paid/overdue
- Payment options: ACH/card (Stripe), manual payment entry

7.5 Eviction Prep Toolkit (Important Differentiator)

Goal: Reduce fear and complexity by providing standardized steps and documents. (Not legal advice).

- Templates for notices (late notice, pay-or-quit)
- Checklist timeline: what to do and when
- Auto-filled tenant/property information on documents
- Document export to PDF for printing/court filing

8. User Roles & Permissions

- Admin/Owner: full access, settings, policies, reports
- Staff: rent follow-up + maintenance, no financial policy changes
- Vendor/Contractor: access to assigned projects only; upload proof and invoices
- Tenant (optional): pay rent, view balance, submit maintenance requests

9. Data Model (Suggested)

- Properties (address, unit count, photos, metadata)
- Units (rent amount, lease terms)
- Tenants (contact info, lease docs)
- Rent ledger entries (charges/payments/fees/credits)
- Messages (SMS/email logs, template used, timestamp)
- Projects/Work orders (status, vendor, schedule)
- Proof items (GPS check-in/out, photos, timestamps, notes)
- Invoices (line items, attachments, payment status)

10. Implementation Suggestion: Web App + Phone App Shell

10.1 Why Web App First

- Fastest to build and iterate (admin workflows easier on desktop)
- Centralized backend and database
- Easier export/reporting and document generation (PDFs)

10.2 Phone App Shell

Use a mobile app shell for key mobile-only tasks: job site proof, photos, GPS, and quick tenant messaging.

- Recommended approach: React Native or Flutter app shell
- Primary mobile features in MVP:
- Project proof capture (GPS + photos + work log)
- Quick send rent reminder texts
- Notifications (late rent alerts, project updates)

10.3 Suggested Tech Stack

- Frontend web: React / Next.js (or Blazor for Microsoft-native UI)
- Backend API: ASP.NET Core Web API (.NET 9)
- Auth: ASP.NET Core Identity + JWT + RBAC (policy-based authorization)
- Database: SQL Server (or PostgreSQL with Npgsql)
- ORM & migrations: Entity Framework Core
- File storage: Azure Blob Storage (or AWS S3)
- Payments: Stripe (ACH + card)
- SMS/voice: Twilio
- Hosting: Azure App Service / Azure Container Apps (or Kubernetes)
- Observability: Application Insights + structured logging

10.4 Security & Compliance

- Encrypt sensitive tenant data at rest and in transit
- Role-based access control (RBAC)
- Audit logs for financial changes and document generation
- Backup and disaster recovery plan

11. MVP Plan & Timeline (Example 8–12 weeks)

- Phase 1 (Weeks 1–3): database + property/tenant setup + rent ledger
- Phase 2 (Weeks 4–6): rent chase automation + templates + communication log
- Phase 3 (Weeks 7–9): maintenance projects + proof capture + photo storage
- Phase 4 (Weeks 10–12): invoicing + payment link + exports + polish

12. Future Enhancements

- Tenant portal (maintenance requests, statement downloads)
- Vendor scorecards and preferred vendor list
- AI assistant: draft messages, summarize histories, detect delinquency risk
- Integration: Zillow/Avail import; QuickBooks syncing
- Bulk operations: send reminders to all late tenants, batch export

Appendix A: Key Screens (Wireframe List)

- Dashboard (late rent, unpaid balances, open projects)
- Tenant profile (timeline: rent + messages + documents)
- Rent chase action page (templates + send + schedule)
- Project page (bid, proof, invoice)
- Reports (monthly income/delinquency/expenses)

Appendix-- Database

```
-- 1. Create Database
CREATE DATABASE PropertyManagementDB;
GO
USE PropertyManagementDB;
GO

-- 2. Create Tables
CREATE TABLE Properties (
   PropertyID INT PRIMARY KEY IDENTITY(1,1),
   PropertyName NVARCHAR(100),
   Address NVARCHAR(255),
   UnitNumber NVARCHAR(20),
   MonthlyRent DECIMAL(18,2)
);

CREATE TABLE Tenants (
   TenantID INT PRIMARY KEY IDENTITY(1,1),
   FirstName NVARCHAR(50),
   LastName NVARCHAR(50),
   Email NVARCHAR(100),
   PhoneNumber NVARCHAR(20),
   PropertyID INT FOREIGN KEY REFERENCES Properties(PropertyID)
);

CREATE TABLE RentSchedules (
   ScheduleID INT PRIMARY KEY IDENTITY(1,1),
   TenantID INT FOREIGN KEY REFERENCES Tenants(TenantID),
   DueDate DATE,
   Status NVARCHAR(20) DEFAULT 'Unpaid', -- Unpaid, Paid, Late, Partial
   BaseRent DECIMAL(18,2),
   LateFeeAccrued DECIMAL(18,2) DEFAULT 0.00,
   ReminderCount INT DEFAULT 0 -- Tracks Day 1 text, Day 3 notice
);

CREATE TABLE RentPayments (
   PaymentID INT PRIMARY KEY IDENTITY(1,1),
   ScheduleID INT FOREIGN KEY REFERENCES RentSchedules(ScheduleID),
   PaymentDate DATETIME DEFAULT GETDATE(),
   AmountPaid DECIMAL(18,2),
   PaymentMethod NVARCHAR(50), -- ACH, Card
   TransactionRef NVARCHAR(100)
```

```
);

CREATE TABLE MaintenanceProjects (
    ProjectID INT PRIMARY KEY IDENTITY(1,1),
    PropertyID INT FOREIGN KEY REFERENCES Properties(PropertyID),
    ProjectTitle NVARCHAR(200),
    BidAmount DECIMAL(18,2),
    Status NVARCHAR(50), -- Bid, Approved, Work Order, Invoiced, Closed
    AssignedVendor NVARCHAR(100)
);

CREATE TABLE WorkLogs (
    LogID INT PRIMARY KEY IDENTITY(1,1),
    ProjectID INT FOREIGN KEY REFERENCES MaintenanceProjects(ProjectID),
    ClockInTime DATETIME,
    ClockOutTime DATETIME,
    GPSLocation NVARCHAR(100), -- Format: Lat, Long
    ProofPhotoURL NVARCHAR(MAX),
    MaterialsUsed NVARCHAR(MAX),
    VendorSignature NVARCHAR(MAX)
);

CREATE TABLE Invoices (
    InvoiceID INT PRIMARY KEY IDENTITY(1,1),
    ProjectID INT NULL FOREIGN KEY REFERENCES MaintenanceProjects(ProjectID),
    ScheduleID INT NULL FOREIGN KEY REFERENCES RentSchedules(ScheduleID),
    InvoiceDate DATETIME DEFAULT GETDATE(),
    TotalAmount DECIMAL(18,2),
    IsExported BIT DEFAULT 0 -- For QuickBooks tracking
);

GO

-- 3. Insert Dummy Data (10 Rows per table)

-- Properties
INSERT INTO Properties (PropertyName, Address, UnitNumber, MonthlyRent) VALUES
('Sunrise Apts', '123 Maple St', '1A', 1200), ('Sunrise Apts', '123 Maple St', '1B', 1250),
('Oak Estates', '456 Oak Ave', 'Unit 10', 2000), ('Oak Estates', '456 Oak Ave', 'Unit 11', 2100),
('River View', '789 River Rd', '302', 1500), ('River View', '789 River Rd', '303', 1550),
('The Lofts', '101 Urban Sq', 'A', 3000), ('The Lofts', '101 Urban Sq', 'B', 3200),
('Pine Ridge', '555 Pine Ln', '5', 900), ('Pine Ridge', '555 Pine Ln', '6', 950);
```

```
-- Tenants
INSERT INTO Tenants (FirstName, LastName, Email, PhoneNumber, PropertyID) VALUES
('John', 'Doe', 'john@example.com', '555-0101', 1), ('Jane', 'Smith', 'jane@example.com',
    '555-0102', 2),
('Mike', 'Jones', 'mike@example.com', '555-0103', 3), ('Sarah', 'Wilson',
    'sarah@example.com', '555-0104', 4),
('Alex', 'Brown', 'alex@example.com', '555-0105', 5), ('Chris', 'Davis', 'chris@example.com',
    '555-0106', 6),
('Pat', 'Taylor', 'pat@example.com', '555-0107', 7), ('Sam', 'Moore', 'sam@example.com',
    '555-0108', 8),
('Kelly', 'White', 'kelly@example.com', '555-0109', 9), ('Drew', 'Harris', 'drew@example.com',
    '555-0110', 10);

-- RentSchedules (The Monthly Rent Bills)
INSERT INTO RentSchedules (TenantID, DueDate, Status, BaseRent, ReminderCount)
    VALUES
(1, '2026-02-01', 'Paid', 1200, 0), (2, '2026-02-01', 'Unpaid', 1250, 1),
(3, '2026-02-01', 'Late', 2000, 3), (4, '2026-02-01', 'Unpaid', 2100, 0),
(5, '2026-02-01', 'Paid', 1500, 0), (6, '2026-02-01', 'Partial', 1550, 2),
(7, '2026-02-01', 'Paid', 3000, 0), (8, '2026-02-01', 'Unpaid', 3200, 0),
(9, '2026-02-01', 'Paid', 900, 0), (10, '2026-02-01', 'Unpaid', 950, 1);

-- RentPayments
INSERT INTO RentPayments (ScheduleID, AmountPaid, PaymentMethod) VALUES
(1, 1200, 'ACH'), (5, 1500, 'Card'), (7, 3000, 'ACH'), (9, 900, 'ACH'), (6, 500, 'Card');
-- Only 5 payments made to reflect unpaid statuses

-- MaintenanceProjects
INSERT INTO MaintenanceProjects (PropertyID, ProjectTitle, BidAmount, Status,
    AssignedVendor) VALUES
(1, 'Broken Faucet', 150, 'Closed', 'Fix-It Plumbing'), (2, 'Paint Bedroom', 400, 'Invoiced', 'Pro
    Painters'),
(3, 'AC Repair', 800, 'Work Order', 'CoolAir Inc'), (4, 'Roof Leak', 2500, 'Bid', 'TopRoofing'),
(5, 'Floor Buffing', 300, 'Approved', 'Janitor Pro'), (6, 'Door Lock Fix', 100, 'Closed',
    'SafeLocks'),
(7, 'Window Cleaning', 200, 'Invoiced', 'ClearView'), (8, 'Electrical Spark', 600, 'Work Order',
    'VoltGuys'),
(9, 'Gutter Clean', 150, 'Approved', 'YardHelp'), (10, 'New Carpet', 1200, 'Bid', 'CarpetWorld');

-- WorkLogs (Proof-based)
INSERT INTO WorkLogs (ProjectID, ClockInTime, ClockOutTime, GPSLocation,
    ProofPhotoURL) VALUES
(1, '2026-01-10 09:00', '2026-01-10 10:30', '34.05,-118.24', 'img01.jpg'),
```

(6, '2026-01-12 14:00', '2026-01-12 14:45', '34.06,-118.25', 'img02.jpg');
-- Logs only for 'Closed' projects

-- Invoices
INSERT INTO Invoices (ProjectID, TotalAmount, IsExported) VALUES
(1, 150, 1), (2, 400, 0), (6, 100, 1), (7, 200, 0);