Vendor Brief: Tenant Management System Upgrade

Date: June 10, 2025
Prepared For: Prospective Software Development Vendors

1. Introduction & Background

We are a real estate management entity seeking to upgrade our existing desktop-based tenant management software. Our goal is to transition from a legacy, on-premise application to a modern, cloud-based solution that offers enhanced functionality, improved usability, and better scalability. The current system is a Windows executable with a desktop interface, and we are looking for a comprehensive upgrade that addresses its limitations and introduces new features.

2. Project Goals & Objectives

The primary objective is to develop a cloud-based tenant management system that is efficient, user-friendly, and capable of supporting our growing portfolio. Key objectives include:

- Transitioning from a desktop application to a cloud-based, multi-user accessible platform.
- Improving data management, reporting, and payment handling processes.
- Enhancing communication with tenants through modern integration.
- Providing mobile accessibility for essential functions.
- Ensuring the system is lightweight and performs well even with large datasets.

3. Current System Overview

Our existing system is a desktop-based application with the following modules and functionalities:

- **Technology:** Desktop-based Windows application (needs to be upgraded to a modern, cloud-based solution).
- **Purpose:** Manages properties, tenants, transactions, billing, and reporting for a real estate portfolio.

Detailed Tab-by-Tab Description of the Current System:

1. **Home Tab:**
    - **Description:** This tab is intended to be a brief overview but currently fails at its task.
    - **Sections:**
        - **Properties:** Shows the number of properties in the system.
        - **Number of Tenants:** Shows the total number of tenants.

        - **Outstandings:** Shows total outstandings, but is not very useful as it doesn't indicate who the money is to come from.
        - **Current FY:** Described as "completely useless," not printed anywhere or used in any tables; might be used in the backend.
        - **Recent Payments:** The biggest section, displaying a table with columns for name, payment month, current rent, penalty, received amount, and entry creation date.
2. **Property Management Tab:**
    - **Dropdown Options:**
        - **View Properties:** A table that shows all available current properties.
        - **Add New Property:** Provides an option to add a new property.
3. **Tenant Management Tab:**
    - **Dropdown Options:**
        - **View Tenants:** A table of tenants and details of their basic rent, shop type, property tax, repair cess, miscellaneous, penalty, outstanding dues, total, and due date.
        - **Add New Tenants:** A form with fields corresponding to the details in the "View Tenants" table, used for adding new tenants.
4. **Transactions Tab:**
    - **Dropdown Options:**
        - **Update Tenant Factors:** Allows the user to increment existing hard-coded formulas for (basic rent, property tax, repair cess, misc, penalty, total) by a percentage.
        - **Make Payment Entry:** Used to add a payment entry for a tenant using a form.
        - **View Payment Entry:** Allows viewing payment entries made previously from the database.
        - **Debit Notes:** A feature previously added to issue any extra debit notes to the tenant.
5. **Billing Tab:**
    - **Only Item:**
        - **Generate Bills:** Used to generate bills by selecting a tenant and a specific period.
6. **Reports Tab:**
    - **Functionality:** Used to generate various reports.
7. **SMS Tab:**
    - **Functionality:** Used to send broadcast messages for due dates, outstanding amounts, penalties, etc. (Will be replaced by WhatsApp integration).
8. **Settings Tab:**
    - Standard settings configuration for the application.

Key Pain Points of the Current System:

- Desktop-based, lacking cloud accessibility and multi-user collaboration.
- "Home" dashboard is not effective for quick insights.
- Hardcoded formulas for rent/tax increments, requiring manual updates.
- Outstanding dues display is not useful, lacking detail on who owes.
- Limited reporting utilities within data tables.
- No mobile accessibility.
- Reliance on outdated SMS functionality instead of modern communication.
- Receipt formatting issues and lack of direct sharing options.
- Potential performance issues with large datasets.

4. Core Requirements for the Upgraded System

The new cloud-based system must incorporate the following features and enhancements:

4.1 Cloud-Based & Multi-User Collaboration:

- The system must be hosted online with real-time collaboration capabilities.
- Support a minimum of 2 simultaneous users.
- All edits must display "Last edited by [User]" for accountability and tracking, similar to Notion.

4.2 Dynamic Formula Management:

- Existing hardcoded formulas (e.g., for rent increment, property tax changes) must be made dynamic and editable by the admin via the UI.

4.3 User Roles & Property Classification:

- Ability to classify properties into categories: Shops, Godowns, and Mezzanine Floors.
- Each property class must have separate increment logic (e.g., 4% for mezzanine, 6% for first-floor shops).

4.4 Enhanced Payment Handling:

- A separate ledger view should display **only outstanding dues**.
- Debit Notes must be editable post-creation.

4.5 Table Enhancements:

- Addition of a count function and other basic utilities within data tables for improved reporting.

4.6 Mobile Accessibility (Responsive Web App):

- A mobile version (responsive web app is fine) should support only:
    - Viewing outstanding dues.
    - Updating payment entries.

4.7 Third-Party Integration (WhatsApp):

- Seamless WhatsApp integration to:
    - Send rent reminders.
    - Notify tenants about increment changes.
    - Share payment receipts.
- Message templates will be provided by us.

4.8 Receipts:

- Better receipt formatting is needed (a sample will be shared).
- Receipts should be downloadable and WhatsApp-shareable directly from the system.

4.9 Performance & Usability:

- The system must be lightweight and lag-free, even when handling large datasets.

5. Budget & Timeline

Our indicative budget for the initial development/upgrade phase is **INR 40,000 - 50,000 (approximately USD 480 - 600)**. This budget primarily covers the core development and implementation of the features outlined above.

Ongoing Tech Support: We require a separate, undisclosed budget for ongoing technical support and maintenance post-launch. Vendors should propose their preferred model for this (e.g., monthly retainer, per-incident, etc.).

We are looking for an efficient development timeline and would appreciate vendor proposals to include their estimated project duration.

6. Submission Requirements

Interested vendors are requested to submit a proposal that includes:

- An understanding of our requirements.
- A proposed technical solution and architecture.
- A detailed breakdown of costs aligning with the budget, indicating what is feasible within the specified range.
- A proposed project timeline with key milestones.
- Your approach to ongoing technical support and associated costs.
- Relevant case studies or client testimonials, especially for similar projects.
- Team composition and experience.