Conceptual design and flow chart

"Property Management System API - Design Document"

1. Introduction

Purpose

This document provides an architectural overview of the Property Management System API, outlining its backend components, database design, system functionalities, and security considerations.

Scope

The system is designed exclusively for property owners to efficiently manage their properties and tenants. It allows owners to:

- Register, log in, and log out
- Create, update, and delete properties
- Add and remove tenants
- Calculate rent distribution

The API ensures secure authentication, data validation, and restricted access so that owners can manage only their own properties and tenants.

Features Included:

- Authentication (register, login, logout) using Laravel Sanctum
- Property management (CRUD operations)
- Tenant management
- Rent distribution logic

2. System Overview

Components

- **Backend**: Laravel (Manages business logic, API endpoints, and database interactions**)**
- **Database:** MySQL (Stores user, property, and tenant data)

3. Architectural Strategy

Design Principles

- **Modularity**: backend is loosely coupled to allow independent scaling (solid).

Conceptual design and flow chart

- **Scalability:** Supports horizontal and vertical scaling to handle more users and properties.
- **Security:** Implements authentication, data validation, and secure API endpoints.

4. Technology Stack

- **Framework:** Laravel 10^
- **Authentication:** Laravel Sanctum
- **Database:** MySQL
- **ORM:** Eloquent
- **Environment Management:** .env files

Tools

- **Version Control:** Git
- **Package Managers:** Composer (PHP)

5. Database Design

Schema

Users Table

- owner credentials and details (Laravel default user table)
- One-to-Many relationship with properties

Properties Table

- property details, including the owner ID (foreign key from users table)
- One-to-Many relationship with tenants

Tenants Table

- tenant details, linked to a property ID (foreign key)
- Belongs to One property

Conceptual design and flow chart

6. API Design

Authentication APIs

Method	Endpoint	Description
POST	/api/register	Registers a new owner
POST	/api/login	Authenticates an owner
POST	/api/logout	Logs out an owner

Property APIs

Method	Endpoint	Description
GET	/api/properties	Retrieve all properties (only for the authenticated owner)
POST	/api/properties	Create a new property
GET	/api/properties/{id}	Retrieve a single property with its tenants
PUT	/api/properties/{id}	Update property details
DELETE	/api/properties/{id}	Soft delete a property

Tenant APIs

Method	Endpoint	Description
POST	/api/tenants	Assign a tenant to a property
GET	/api/tenants	Retrieve all tenants (only for the authenticated owner)
DELETE	/api/tenants/{id}	Remove a tenant from a property

Rent Distribution API

Conceptual design and flow chart

Method	Endpoint	Description
GET	/api/properties/{id}/rent-distribution	Calculate and return rent share in tenants

Bonus Features (Optional)

Method	Endpoint	Description
GET	/api/tenants/{id}/rent	Get a selected tenant’s monthly rent

7. Security Considerations

- **Authentication:** Uses Laravel Sanctum with Bearer tokens
- **Authorization:** Owners can only access their properties and tenants
- **Data Validation:** Implemented strict validation rules to prevent SQL injection and invalid data entry

8. Assumptions & Challenges

Assumptions (no new assumptions scope based)

- Late fees are not implemented (assumed to be 0).
- Scope based implementation only (each property has a single owner).
- Agreement percentages do not need to sum to 100%; rent is distributed based on defined percentages. Or divided equally using count
- Testing is performed using a separate. env.testing database.

9. Challenges and Solutions

Challenge	Solution
• **Restricting access to only the owner's properties and tenants**	• **Used whereHas('property', fn($q) => $q->where('owner_id', auth()->id()))** • **(auth id-based filter)**
• **Handling rent distribution dynamically (sample data <100% or mixed)**	• **Implemented logic to split rent by agreement percentage or equally (skip others)**

Conceptual design and flow chart

Challenge	Solution
• **Validation** • **Status and success responses** • **Test cases** • **Managing soft deletes for properties and tenants**	• **Unique, validation handled** • **Handle in separated class and injected** • **Some manual changes in logics** • **Used SoftDeletes in Eloquent models**

10. Testing & Validation

- Unit & Feature Tests: Implemented using Feature and Unit with Refresh Database
- Edge Cases Tested:
    - No tenants in a property
    - Single tenant pays full rent
    - tenants with and without agreements rent
    - crud / auth / some relationship (1 to n)

11. Backend

Run Laravel on port 8000:

http://127.0.0.1:8000

deployed to the server you can access via this endpoint

https://api.property.arunyadhav.live/api

API Collection: Attached in document folder

12. References

- Laravel documentation
- Chat gpt
- You tube