Smart Village Management System - MVP Progress Report

Project Overview

The Smart Village Management System is a comprehensive platform designed to streamline and enhance the management of residential communities. The system integrates access control, financial management, and resident communication into a unified platform, improving efficiency and transparency for village administrators and enhancing the living experience for residents.

Current Progress

We have successfully completed the following phases of the project:

1. **Project Kickoff and Team Setup**
2. Defined project scope and objectives
3. Established team roles and responsibilities

4. Set up communication channels and project management tools

5. **Development Environment Preparation**

6. Created GitHub repository: smart-village-management
7. Set up project structure and foundational files

8. Configured development tools and dependencies

9. **System Architecture and Database Design**

10. Designed comprehensive database schema with multi-tenant support
11. Created entity relationship diagrams
12. Defined data models and relationships

13. Implemented security and access control mechanisms

14. **MVP Core Features Development**

15. Implemented backend API endpoints for all core modules:
    - User management and authentication

    - Property management
    - Invoice and payment processing
    - Access control system
    - Expense tracking
    - Visitor management
16. Developed comprehensive validation and security measures

17. Implemented multi-tenant data isolation

18. **Validation and Testing**

19. Created comprehensive test suite for all API endpoints
20. Implemented unit tests for core functionality
21. Validated multi-tenant data isolation
22. Verified security measures and access controls

Technical Implementation Details

Backend Architecture

The backend is built using FastAPI, a modern, high-performance web framework for building APIs with Python. Key components include:

- **API Framework**: FastAPI with Pydantic for data validation
- **Database**: SQLAlchemy ORM with PostgreSQL
- **Authentication**: JWT-based authentication with role-based access control
- **Security**: Password hashing, input validation, and request verification

Database Schema

The database schema supports both multi-tenant and standalone deployment models with the following key entities:

- Villages (for multi-tenant support)
- Users (admin, staff, resident roles)
- Properties
- Invoices
- Payments
- Access Logs
- Expenses and Categories
- Visitors

API Endpoints

The following API endpoints have been implemented:

1. **Authentication**
2. Login and token generation
3. Password management
4. User profile access
5. **User Management**
6. CRUD operations for users
7. Role-based access control
8. **Property Management**
9. CRUD operations for properties
10. Owner and resident assignment
11. **Invoice Management**
12. CRUD operations for invoices
13. Automatic monthly invoice generation
14. Invoice status tracking
15. **Payment Processing**
16. Payment recording and verification
17. Payment slip upload
18. Partial payment handling
19. Overpayment credit system
20. **Access Control**
21. Gate access via mobile app
22. Access log recording and retrieval
23. Access statistics
24. **Expense Management**
25. Expense categorization and tracking
26. Receipt upload and management

27. Expense reporting

28. **Visitor Management**

29. Visitor registration
30. Entry code generation and verification
31. Visitor access logging

Validation Results

The comprehensive test suite validates all core functionality:

- **Authentication Tests**: Verify login, token generation, and authorization
- **User Management Tests**: Validate CRUD operations and permission controls
- **Property Tests**: Confirm property management functionality
- **Invoice Tests**: Verify invoice creation, updating, and status management
- **Payment Tests**: Validate payment processing, verification, and invoice updates
- **Access Control Tests**: Confirm gate access and logging functionality
- **Expense Tests**: Verify expense tracking and categorization
- **Visitor Tests**: Validate visitor management and entry code verification

All tests pass successfully, confirming that the MVP functionality meets the requirements.

Next Steps

The following steps are recommended to complete the project:

1. **Frontend Development**
2. Develop admin dashboard using React.js
3. Implement responsive design for mobile and desktop

4. Create intuitive user interfaces for all core features

5. **Mobile App Development**

6. Develop resident mobile app using React Native
7. Implement gate control functionality

8. Create invoice viewing and payment features

9. **Integration and Deployment**

10. Set up CI/CD pipeline

11. Configure production environment

12. Deploy backend API to cloud infrastructure

13. **User Acceptance Testing**

14. Conduct UAT with stakeholders
15. Gather feedback and implement improvements

16. Validate real-world scenarios

17. **Documentation and Training**

18. Create user manuals and documentation
19. Provide training for administrators and staff
20. Develop onboarding materials for residents

Conclusion

The Smart Village Management System MVP backend has been successfully developed with all core features implemented and tested. The system is designed to be flexible, secure, and scalable, supporting both multi-tenant and standalone deployment models.

The project is now ready to proceed to frontend and mobile app development phases, followed by integration, deployment, and user acceptance testing.