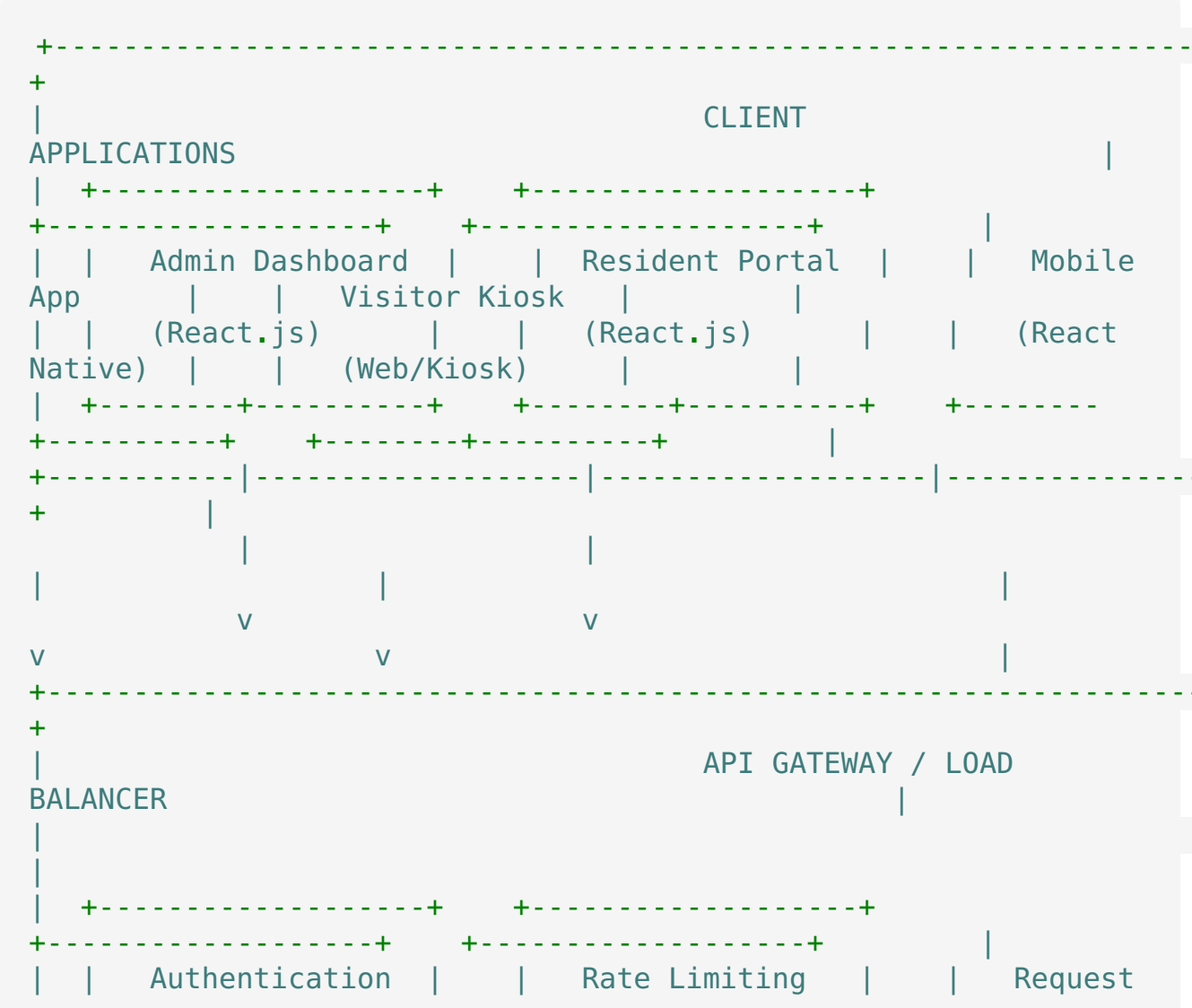
Smart Village Management System - System Architecture

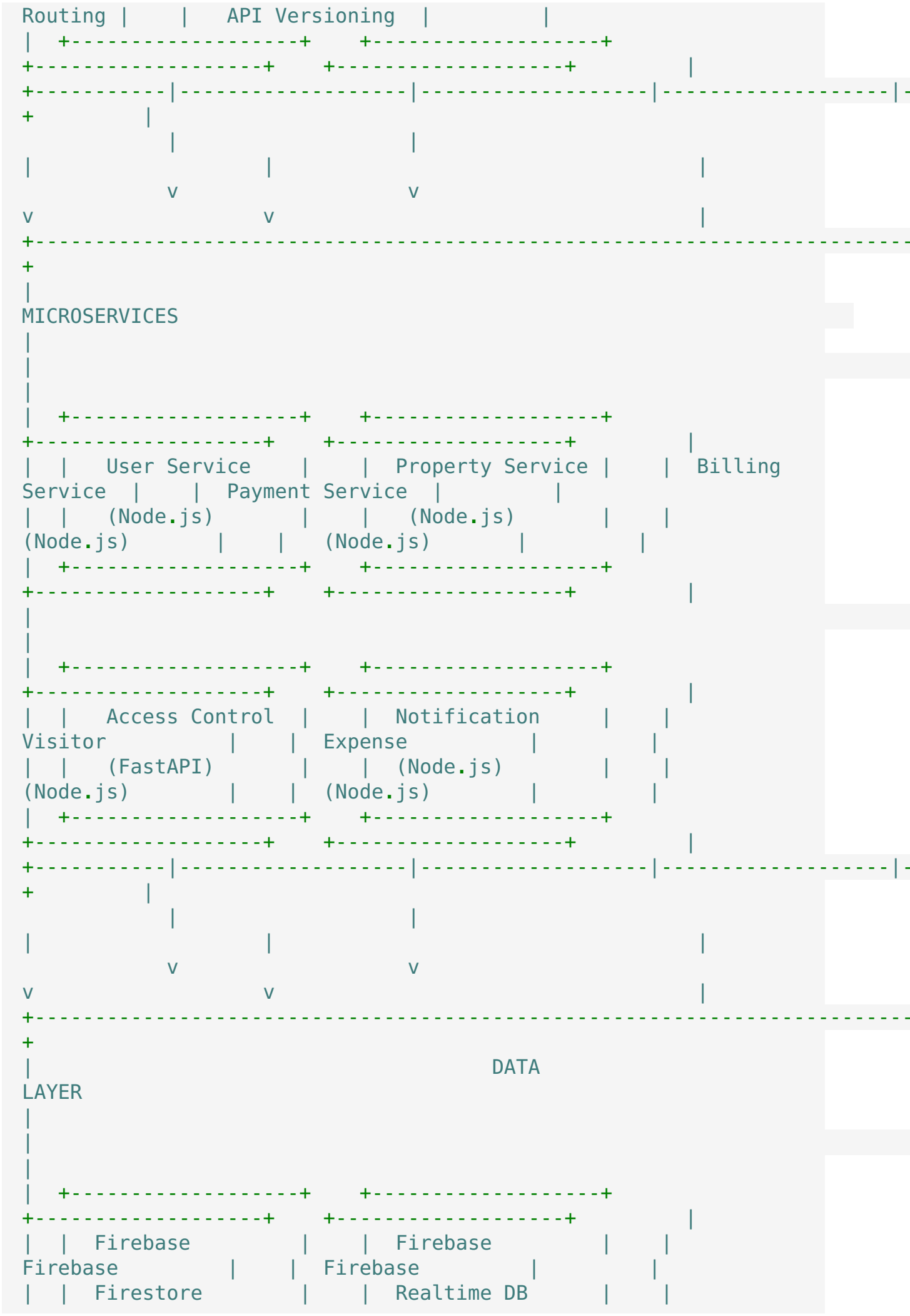
1. System Overview

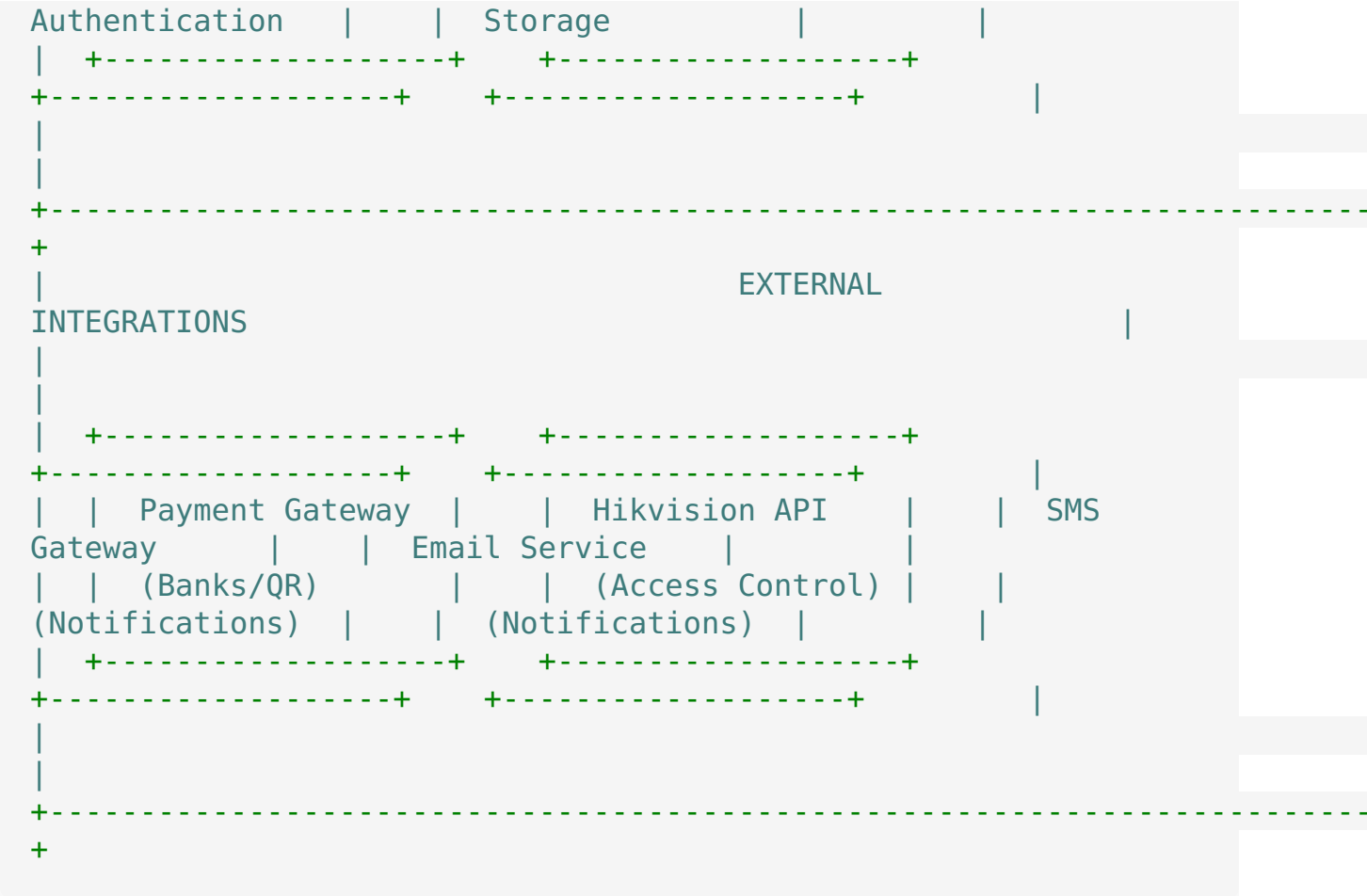
Smart Village Management System เป็นแพลตฟอร์มที่ออกแบบมาเพื่อจัดการหมู่บ้านอย่างครบวงจร โดยรวมฟีเจอร์สำคัญไว้ด้วยกัน ได้แก่ การจัดการผู้อยู่อาศัย การควบคุมการเข้าออก การจัดการการเงิน และการสื่อสารภายในหมู่บ้าน

ระบบถูกออกแบบให้มีความยืดหยุ่น ปลอดภัย และขยายขนาดได้ โดยใช้สถาปัตยกรรมแบบ Microservices และ Cloud-native ที่สามารถรองรับทั้งแบบ Multi-tenant (หลายหมู่บ้านในแพลตฟอร์มเดียว) และ Standalone (แยกเป็นแพลตฟอร์มเฉพาะสำหรับหมู่บ้านที่ต้องการซื้อแยก)

2. High-Level Architecture Diagram




Routing
API Versioning
MICROSERVICES
User Service
(Node.js)
Property Service
(Node.js)
Billing Service
(Node.js)
Payment Service
(Node.js)
Access Control
(FastAPI)
Notification
(Node.js)
Visitor Expense
(Node.js)
(Node.js)
DATA LAYER
Firebase Firestore
Firebase Realtime DB
Firebase
Firebase



3. Component Details

3.1 Client Applications

Admin Dashboard (React.js + Vercel)

- **ฟังก์ชัน**: จัดการผู้ใช้, บ้าน/ทรัพย์สิน, ใบแจ้งหนี้, การชำระเงิน, รายจ่าย, รายงาน
- **เทคโนโลยี**: React.js, Material-UI, Chart.js, Vercel Hosting
- **การเข้าถึง**: เฉพาะผู้ดูแลระบบและเจ้าหน้าที่นิติบุคคล

Resident Portal (React.js + Vercel)

- **ฟังก์ชัน**: ดูใบแจ้งหนี้, ชำระเงิน, แจ้งผู้มาเยือน, ดูประกาศ
- **เทคโนโลยี**: React.js, Tailwind CSS, Vercel Hosting
- **การเข้าถึง**: ผู้อยู่อาศัยในหมู่บ้าน

Mobile App (React Native)

- **ฟังก์ชัน**: ควบคุมไม้กั้น, ดูใบแจ้งหนี้, ชำระเงิน, แจ้งผู้มาเยือน, รับการแจ้งเตือน
- **เทคโนโลยี**: React Native, Firebase SDK
- **แพลตฟอร์ม**: iOS และ Android

Visitor Kiosk (Web Application)

- **ฟังก์ชัน**: ลงทะเบียนผู้มาเยือน, สแกน QR Code, แจ้งเจ้าบ้าน
- **เทคโนโลยี**: React.js, Progressive Web App
- **การติดตั้ง**: ป้อมยาม, ทางเข้าหมู่บ้าน

3.2 API Gateway / Load Balancer

- **ฟังก์ชัน**: จัดการการเข้าถึง API, การยืนยันตัวตน, การจำกัดอัตราการเรียกใช้, การจัดเส้นทาง
- **เทคโนโลยี**: Vercel Edge Functions, Firebase Cloud Functions
- **ความปลอดภัย**: JWT Authentication, HTTPS, CORS

3.3 Microservices

User Service

- **ฟังก์ชัน**: จัดการผู้ใช้, บทบาท, สิทธิ์การเข้าถึง
- **เทคโนโลยี**: Node.js, Express, Firebase Authentication
- **API Endpoints**: /users, /roles, /permissions

Property Service

- **ฟังก์ชัน**: จัดการบ้าน/ทรัพย์สิน, ผู้อยู่อาศัย, ความสัมพันธ์
- **เทคโนโลยี**: Node.js, Express, Firestore
- **API Endpoints**: /properties, /residents, /households

Billing Service

- **ฟังก์ชัน**: สร้างใบแจ้งหนี้, จัดการรายการ, คำนวณยอดรวม
- **เทคโนโลยี**: Node.js, Express, Firestore
- **API Endpoints**: /invoices, /invoice-items, /billing-cycles

Payment Service

- **ฟังก์ชัน**: บันทึกการชำระเงิน, ตรวจสอบการชำระเงิน, ออกใบเสร็จ
- **เทคโนโลยี**: Node.js, Express, Firestore
- **API Endpoints**: /payments, /receipts, /payment-verification

Access Control Service

- **ฟังก์ชัน**: ควบคุมไม้กั้น, บันทึกการเข้าออก, จัดการสิทธิ์การเข้าถึง
- **เทคโนโลยี**: FastAPI (Python), Hikvision ISAPI

- **API Endpoints**: /access, /gates, /access-logs

Notification Service

- **ฟังก์ชัน**: ส่งการแจ้งเตือน, จัดการประกาศ, การสื่อสาร
- **เทคโนโลยี**: Node.js, Firebase Cloud Messaging
- **API Endpoints**: /notifications, /announcements, /messages

Visitor Service

- **ฟังก์ชัน**: จัดการผู้มาเยือน, สร้าง QR Code, แจ้งเจ้าบ้าน
- **เทคโนโลยี**: Node.js, Express, Firestore
- **API Endpoints**: /visitors, /visitor-passes, /visitor-logs

Expense Service

- **ฟังก์ชัน**: บันทึกรายจ่าย, จัดการประเภทรายจ่าย, รายงาน
- **เทคโนโลยี**: Node.js, Express, Firestore
- **API Endpoints**: /expenses, /expense-categories, /expense-reports

3.4 Data Layer

Firebase Firestore

- **ฟังก์ชัน**: จัดเก็บข้อมูลหลัก (ผู้ใช้, บ้าน, ใบแจ้งหนี้, การชำระเงิน)
- **คุณสมบัติ**: NoSQL, Document-based, ขยายขนาดได้อัตโนมัติ
- **การใช้งาน**: ข้อมูลที่มีโครงสร้างและความสัมพันธ์

Firebase Realtime Database

- **ฟังก์ชัน**: จัดเก็บข้อมูลที่ต้องการ real-time updates
- **คุณสมบัติ**: Real-time synchronization, offline support
- **การใช้งาน**: สถานะการเข้าออก, การแจ้งเตือน, สถานะไม้กั้น

Firebase Authentication

- **ฟังก์ชัน**: จัดการการยืนยันตัวตน, สิทธิ์การเข้าถึง
- **คุณสมบัติ**: หลายวิธีการยืนยันตัวตน, ความปลอดภัยสูง
- **การใช้งาน**: การเข้าสู่ระบบของผู้ใช้ทุกประเภท

Firebase Storage

- **ฟังก์ชัน**: จัดเก็บไฟล์และรูปภาพ
- **คุณสมบัติ**: ขยายขนาดได้, ความปลอดภัยสูง, การควบคุมการเข้าถึง

- **การใช้งาน**: สลิปการโอนเงิน, รูปภาพผู้อยู่อาศัย, เอกสาร

3.5 External Integrations

Payment Gateway

- **ฟังก์ชัน**: เชื่อมต่อกับระบบธนาคารและ QR Payment
- **เทคโนโลยี**: RESTful API, Webhooks
- **การใช้งาน**: การชำระเงินออนไลน์

Hikvision API (ISAPI)

- **ฟังก์ชัน**: ควบคุมไม้กั้นและระบบรักษาความปลอดภัย
- **เทคโนโลยี**: ISAPI, RESTful API
- **การใช้งาน**: การควบคุมการเข้าออก, กล้องวงจรปิด

SMS Gateway

- **ฟังก์ชัน**: ส่ง SMS แจ้งเตือน
- **เทคโนโลยี**: RESTful API
- **การใช้งาน**: แจ้งเตือนใบแจ้งหนี้, การชำระเงิน, ผู้มาเยือน

Email Service

- **ฟังก์ชัน**: ส่งอีเมลแจ้งเตือนและรายงาน
- **เทคโนโลยี**: SMTP, API
- **การใช้งาน**: ส่งใบแจ้งหนี้, ใบเสร็จ, รายงาน

4. Data Flow

4.1 การสร้างและชำระใบแจ้งหนี้

1. Billing Service สร้างใบแจ้งหนี้อัตโนมัติหรือตามที่ Admin กำหนด
2. ข้อมูลใบแจ้งหนี้ถูกบันทึกใน Firestore
3. Notification Service ส่งการแจ้งเตือนไปยังผู้อยู่อาศัย
4. ผู้อยู่อาศัยดูใบแจ้งหนี้ผ่าน Mobile App หรือ Resident Portal
5. ผู้อยู่อาศัยชำระเงินผ่าน Payment Gateway
6. Payment Service บันทึกการชำระเงินและออกใบเสร็จ
7. Notification Service แจ้งเตือนการชำระเงินสำเร็จ

4.2 การควบคุมการเข้าออก

1. ผู้อยู่อาศัยใช้ Mobile App เพื่อเปิดไม้กั้น
2. Mobile App ส่งคำขอไปยัง Access Control Service
3. Access Control Service ตรวจสอบสิทธิ์และส่งคำสั่งไปยัง Hikvision API
4. Hikvision API ควบคุมไม้กั้นให้เปิด
5. Access Control Service บันทึกการเข้าออกใน Firestore
6. ข้อมูลสถานะไม้กั้นถูกอัปเดตแบบ real-time ใน Realtime Database

4.3 การจัดการผู้มาเยือน

1. ผู้อยู่อาศัยสร้าง Visitor Pass ผ่าน Mobile App
2. Visitor Service สร้าง QR Code และบันทึกข้อมูลใน Firestore
3. ผู้มาเยือนแสดง QR Code ที่ป้อมยาม
4. Visitor Kiosk สแกน QR Code และส่งข้อมูลไปยัง Visitor Service
5. Visitor Service ตรวจสอบความถูกต้องและบันทึกการเข้า
6. Notification Service แจ้งเตือนผู้อยู่อาศัยเมื่อผู้มาเยือนมาถึง

5. Security Architecture

5.1 Authentication & Authorization

- Firebase Authentication สำหรับการยืนยันตัวตน
- JWT (JSON Web Tokens) สำหรับการอนุญาตการเข้าถึง API
- Role-Based Access Control (RBAC) สำหรับการควบคุมสิทธิ์
- Multi-factor Authentication สำหรับบัญชีที่มีความเสี่ยงสูง

5.2 Data Security

- การเข้ารหัสข้อมูลในการส่ง (HTTPS)
- การเข้ารหัสข้อมูลที่จัดเก็บ (Encryption at Rest)
- การแยกข้อมูลระหว่างหมู่บ้าน (Multi-tenancy Isolation)
- การสำรองข้อมูลอัตโนมัติ (Automated Backups)

5.3 API Security

- Rate Limiting เพื่อป้องกัน DDoS
- Input Validation เพื่อป้องกัน Injection Attacks
- CORS (Cross-Origin Resource Sharing) เพื่อควบคุมการเข้าถึงจากโดเมนอื่น
- API Keys และ Secret Management

6. Deployment Architecture

6.1 Frontend Deployment

- Vercel สำหรับ Admin Dashboard และ Resident Portal
- App Store และ Google Play สำหรับ Mobile App
- Progressive Web App สำหรับ Visitor Kiosk

6.2 Backend Deployment

- Vercel Functions สำหรับ API Endpoints
- Firebase Cloud Functions สำหรับ Event-driven Functions
- GitHub Actions สำหรับ CI/CD Pipeline

6.3 Monitoring & Logging

- Firebase Performance Monitoring
- Firebase Crashlytics
- Centralized Logging System
- Real-time Alerts และ Notifications

7. Scalability & Performance

7.1 Horizontal Scaling

- Stateless Microservices ที่สามารถขยายได้ตามต้องการ
- Load Balancing สำหรับการกระจายโหลด
- Auto-scaling ตามปริมาณการใช้งาน

7.2 Performance Optimization

- CDN สำหรับ Static Assets
- Caching Strategies
- Database Indexing
- Lazy Loading Components

7.3 Multi-tenancy Support

- Tenant Isolation ในระดับข้อมูล
- Shared Infrastructure สำหรับประสิทธิภาพต้นทุน
- Tenant-specific Configurations

8. Disaster Recovery & Business Continuity

8.1 Backup Strategy

- Automated Daily Backups
- Point-in-time Recovery
- Geo-redundant Storage

8.2 Failover Mechanism

- Multi-region Deployment
- Automatic Failover
- Service Health Monitoring

9. Development & Testing Environment

9.1 Development Environment

- Local Development Setup
- Development Firebase Project
- Mock External Services

9.2 Testing Environment

- Staging Environment
- Integration Testing
- End-to-end Testing
- Performance Testing

9.3 Production Environment

- Production Firebase Project
- Production External Service Integrations
- Monitoring และ Alerting

10. Conclusion

สถาปัตยกรรมระบบนี้ได้รับการออกแบบให้มีความยืดหยุ่น ปลอดภัย และขยายขนาดได้ โดยใช้เทคโนโลยีที่ทันสมัยและเป็นที่ยอมรับในอุตสาหกรรม การใช้ Vercel และ Firebase ที่คุณคุ้นเคยอยู่แล้วช่วยให้การพัฒนาและการบำรุงรักษาเป็นไปอย่างมีประสิทธิภาพ

สถาปัตยกรรมแบบ Microservices ช่วยให้ทีมพัฒนาสามารถทำงานได้อย่างอิสระและปรับปรุงแต่ละส่วนได้โดยไม่กระทบกับส่วนอื่น ในขณะที่การใช้ Cloud-native Technologies ช่วยให้ระบบมีความยืดหยุ่นและขยายขนาดได้ตามความต้องการของธุรกิจ