0) Tóm tắt

SaaS **multi-tenant** cho **quản lý nhà trọ/chung cư mini** + **chợ nội bộ trong tòa nhà**. Mô hình vai trò: **Super Admin** (toàn hệ thống) → **Admin Tòa** (chủ tòa/ban quản lý) → **Thành viên/Cư dân** (người thuê/ở). Hỗ trợ thu phí, hợp đồng, công tơ điện/nước, thông báo, sự cố; kèm **Marketplace C2C** nội bộ để cư dân mua bán/trao đổi vật phẩm.

Triển khai theo hướng **MVP 6–8 tuần**, **monolith chuẩn hoá** + module hoá (có thể tách microservice sau). CI/CD, logging, giám sát đầy đủ.

---

1) Mục tiêu & Giá trị cốt lõi

- **Giảm 60–80% thời gian** quản lý của chủ tòa: hợp đồng, thu phí, công nợ, nhắc hạn.
- **Minh bạch**: hóa đơn, công tơ, lịch sử thanh toán.
- **Gắn kết cộng đồng**: chợ nội bộ, nhóm thông báo, sự kiện tòa nhà.
- **Mở rộng đa tòa/đa chủ** (multi-tenant): 1 hệ thống phục vụ nhiều tòa độc lập.

---

2) Vai trò & Quyền

Super Admin - Quản lý tenant (tòa), gói dịch vụ, thanh toán thuê bao, người dùng hệ thống, cấu hình tích hợp (ZaloPay/MoMo/VNPay), audit logs, moderation toàn hệ thống.

Admin Tòa (Chủ tòa/Quản lý) - Quản lý tòa nhà, căn hộ/phòng, cư dân, hợp đồng, công tơ, chỉ số, phí phát sinh, hoá đơn, nhắc nợ. - Ticket sự cố, thông báo (broadcast), lịch bảo trì. - Quản trị **chợ nội bộ** của tòa: duyệt tin, quy định, xử lý báo cáo.

Cư dân (Member) - Xem hợp đồng, hóa đơn, thanh toán online, gửi chỉ số công tơ, tạo ticket sự cố. - Đăng tin bán/cho tặng/trao đổi trong **chợ nội bộ**; chat, đặt hàng, đánh giá.

Khách/Visitor (tuỳ chọn) - Truy cập giới hạn, xem tin chợ (ẩn thông tin riêng tư), đăng ký thuê phòng.

---

3) Kiến trúc hệ thống

3.1 Kiến trúc tổng thể (theo stack bạn chọn)

- **Laravel 11 (PHP 8.3)**: core nghiệp vụ, REST API, quản trị web.
- **Node.js service (Realtime)**: Socket.IO cho chat, thông báo, trạng thái đơn hàng; subscribe sự kiện từ Redis Pub/Sub hoặc hàng đợi → broadcast tới client (web/mobile).
- **React Native app**: ứng dụng cư dân & kỹ thuật viên (Android/iOS). Giao tiếp qua REST + Socket.IO.
- **MySQL 8**: cơ sở dữ liệu chính (row-level tenant bằng `building_id`).
- **Redis 7**: cache, queue, pub/sub giữa Laravel ↔ Node realtime.

- **Storage**: MinIO/S3 cho ảnh hợp đồng, công tơ, ảnh sản phẩm.
- **Search**: MySQL FULLTEXT (InnoDB) giai đoạn đầu; có thể thêm MeiliSearch sau nếu cần.
- **Payments**: **Sepay** (webhook). Laravel tạo yêu cầu thanh toán/hoá đơn → đối soát qua webhook Sepay.

3.2 Dòng chảy sự kiện

- Laravel (Invoice/Orders) → **publish** `invoice.created`, `invoice.paid`, `chat.message.created` lên Redis.
- **Node Realtime** subscribe các kênh Redis → **Socket.IO** push tới các phòng theo `building_id`, `user_id`, `chat_id`.
- Sepay → **Webhook** vào Laravel `/payments/sepay/webhook` → xác thực chữ ký → cập nhật `payments` + `invoices` → publish `invoice.paid`.

---

4) Chức năng chi tiết (MVP)

4.1 Quản lý tòa nhà

- Tạo/sửa tòa, cấu hình phí (điện, nước, rác, dịch vụ, gửi xe), biểu giá bậc thang.
- Căn hộ/phòng: trạng thái (trống/đang thuê/đang sửa), diện tích, chỉ số công tơ.
- Cư dân/đồng thuê: thông tin, hợp đồng (file), lịch sử.

4.2 Hợp đồng & hoá đơn

- Tạo hợp đồng: kỳ hạn, đặt cọc, ngày chốt số, phương thức thanh toán.
- Tự động sinh hóa đơn hàng tháng: tiền phòng + điện/nước theo chỉ số + phí khác.
- Nhắc hạn qua push/email/Zalo OA; tra cứu lịch sử, công nợ.

4.3 Công tơ & chỉ số

- Ghi chỉ số điện/nước thủ công hoặc cư dân tự nhập; đối chiếu ảnh công tơ.
- Tính tiền theo bậc thang/đồng giá; phát hiện bất thường (AI đơn giản: outlier).

4.4 Sự cố/Bảo trì

- Ticket (cư dân tạo) → phân công kỹ thuật → chat + ảnh/video → cập nhật tiến độ.
- Lịch bảo trì định kỳ (thang máy, PCCC), nhắc lịch.

4.5 Thông báo & sự kiện

- Thông báo toàn tòa hoặc theo tầng/căn hộ; xác nhận đã đọc.
- Sự kiện (dọn vệ sinh, phun muỗi…), RSVP.

4.6 Chợ nội bộ (Marketplace C2C)

- Đăng tin (bán/cho tặng/trao đổi/thuê mượn), chỉ hiển thị trong **cùng tòa** (default), có tuỳ chọn mở sang tòa lân cận (opt-in của Admin tòa).

- Duyệt tin (moderation), báo cáo vi phạm.
- Chat giữa người mua – bán (real-time), đặt hàng, thương lượng.
- Thanh toán:
- MVP: **Thanh toán trực tiếp** khi giao hàng/nhận đồ (COD/transfer cá nhân) – hệ thống chỉ hỗ trợ chat và đặt lịch.
- Phase 2: ví tạm giữ (escrow nhẹ) qua cổng thanh toán; release khi hai bên confirm.

4.7 Báo cáo/Thống kê

- Doanh thu theo tòa, tỷ lệ thu phí đúng hạn, phòng trống, phân tích điện/nước.
- Logs hoạt động, nhật ký thay đổi dữ liệu (audit).

---

5) Thiết kế dữ liệu (MySQL 8 – bản rút gọn)

```
users(id BIGINT PK, name, phone UNIQUE, email UNIQUE NULL, password_hash,
status TINYINT, created_at, updated_at)
roles(id TINYINT PK, name)  -- super_admin, building_admin, resident
user_roles(user_id BIGINT, role_id TINYINT, building_id BIGINT NULL, PRIMARY
KEY(user_id, role_id, building_id))

buildings(id BIGINT PK, name, address, owner_user_id BIGINT, plan_id BIGINT
NULL, settings JSON, created_at, updated_at)
units(id BIGINT PK, building_id BIGINT, code, floor INT, area DECIMAL(6,2),
status ENUM('vacant','occupied','maintenance'))
leases(id BIGINT PK, unit_id BIGINT, building_id BIGINT, tenant_user_id
BIGINT, start_date DATE, end_date DATE NULL,
       deposit_amount DECIMAL(12,2), status ENUM('active','ended','overdue'),
contract_file_url VARCHAR(255))

meters(id BIGINT PK, building_id BIGINT, unit_id BIGINT, type
ENUM('electric','water'), number VARCHAR(64))
meter_readings(id BIGINT PK, meter_id BIGINT, reading DECIMAL(12,3),
photo_url VARCHAR(255), period_month TINYINT, period_year SMALLINT,
               read_at DATETIME, read_by_user_id BIGINT)

fees(id BIGINT PK, building_id BIGINT, name, calc_method
ENUM('fixed','per_m3','per_kwh','tiered'), price DECIMAL(12,3), meta JSON)

invoices(id BIGINT PK, building_id BIGINT, lease_id BIGINT, period_month
TINYINT, period_year SMALLINT, due_at DATETIME,
         total DECIMAL(12,2), status ENUM('draft','sent','paid','overdue'),
code VARCHAR(32) UNIQUE)
invoice_items(id BIGINT PK, invoice_id BIGINT, name, qty DECIMAL(12,3),
unit_price DECIMAL(12,2), amount DECIMAL(12,2), meta JSON)

tickets(id BIGINT PK, building_id BIGINT, unit_id BIGINT, created_by BIGINT,
assigned_to BIGINT NULL, category, content TEXT,
        status ENUM('open','in_progress','done'), priority
```

```
ENUM('low','med','high'))
announcements(id BIGINT PK, building_id BIGINT, title, content TEXT, scope
JSON, published_at DATETIME)

-- Marketplace
market_items(id BIGINT PK, building_id BIGINT, seller_user_id BIGINT, title,
description TEXT, price DECIMAL(12,2) NULL,
             type ENUM('sell','give','trade','lend'), qty INT, status
ENUM('pending','active','sold','hidden'), images JSON)
market_offers(id BIGINT PK, item_id BIGINT, buyer_user_id BIGINT, price_offer
DECIMAL(12,2) NULL,
              status ENUM('open','accepted','rejected','cancelled'))
market_orders(id BIGINT PK, item_id BIGINT, buyer_user_id BIGINT,
seller_user_id BIGINT, qty INT, amount DECIMAL(12,2),
              status ENUM('created','confirmed','completed','cancelled'))
chats(id BIGINT PK, building_id BIGINT, subject_type, subject_id BIGINT)
chat_messages(id BIGINT PK, chat_id BIGINT, sender_user_id BIGINT, message
TEXT, attachments JSON, created_at DATETIME)

payments(id BIGINT PK, building_id BIGINT, invoice_id BIGINT NULL, order_id
BIGINT NULL, provider ENUM('sepay'),
         txn_id VARCHAR(64) UNIQUE, amount DECIMAL(12,2), status
ENUM('pending','success','failed'), raw JSON,
         idempotency_key VARCHAR(128) UNIQUE NULL, paid_at DATETIME NULL)

audit_logs(id BIGINT PK, user_id BIGINT, building_id BIGINT, action, entity,
entity_id BIGINT, before JSON, after JSON, created_at DATETIME)
```

Ghi chú: - Dùng **JSON** MySQL 8 cho metadata. - Index gợi ý: `idx_units_building`, `idx_invoices_building_period`, FULLTEXT(`market_items`.`title`, `description`). - Mã hoá số tài khoản/CCCD ở tầng ứng dụng nếu cần (AES-256-GCM).

---

6) API/Module (ví dụ)

6.1 Auth & Tenant

- `POST /api/auth/login`
- `POST /api/auth/register` (Super Admin cấp tài khoản Admin Tòa)
- `GET /api/buildings` (Super Admin)
- `POST /api/buildings` (Super Admin)

6.2 Quản lý tòa & phòng

- `GET /api/buildings/{id}`
- `POST /api/buildings/{id}/units`
- `GET /api/buildings/{id}/units?status=occupied|vacant`

6.3 Hợp đồng & hoá đơn

- `POST /api/leases`
- `POST /api/invoices/generate?building_id=&period=YYYY-MM`
- `POST /api/invoices/{id}/send`
- `POST /api/invoices/{id}/pay` (tạo payment link → redirect → webhook)

6.4 Công tơ

- `POST /api/units/{id}/meter-readings` (resident hoặc admin)
- `GET /api/meter-readings?building_id=&period=`

6.5 Ticket/Thông báo

- `POST /api/tickets`
- `PATCH /api/tickets/{id}`
- `POST /api/announcements`

6.6 Marketplace

- `POST /api/market/items`
- `GET /api/market/items?building_id=&type=`
- `POST /api/market/items/{id}/offers`
- `POST /api/market/orders` (tạo đơn)
- `WS /realtime/chats/{subject}` (chat realtime)

---

7) Lựa chọn công nghệ

Backend chính: Laravel 11 (PHP 8.3), Sanctum/JWT, Horizon (queues), Policies/Gates.

Realtime service: **Node.js 20** + **Socket.IO**. Redis Pub/Sub làm bridge từ Laravel jobs/events sang Node để phát realtime.

Mobile: **React Native** (Expo), React Query, Zustand/Redux, Socket.IO client, Notifee/FCM push.

DB: **MySQL 8** (InnoDB).

Cache/Queue: Redis 7.

Storage: MinIO hoặc AWS S3.

Payments: **Sepay webhook** (đối soát chuyển khoản theo nội dung/Ref). Xác thực chữ ký, idempotency.

Logging/Monitoring: Telescope (dev), Sentry (FE/BE), Prometheus + Grafana, EFK/ELK cho logs.

CI/CD: GitHub Actions/GitLab CI → Docker build → push registry → deploy.

---

8) Bảo mật & Quy định

- **RBAC** + kiểm soát truy cập theo `building_id` ở mọi query/service.
- **Idempotency** cho thanh toán; xác minh webhook bằng chữ ký HMAC.
- **Chống XSS/CSRF/Rate limit** (throttling), validation chặt cho file upload.
- **Mã hoá** data nhạy cảm (số tài khoản, CCCD) ở at-rest (PG crypto/column-level encryption) + in-transit (TLS).
- **Backup** DB hàng ngày, lưu 7–30 ngày; DR plan.
- **Moderation** marketplace: danh sách cấm (vũ khí, thuốc men, động vật quý hiếm…), cơ chế report & tạm ẩn tin.

---

9) Quy trình DevOps & Triển khai

1) **Môi trường**: `dev` (Docker Compose), `staging`, `prod`. 2) **Docker Compose services**: - `nginx`, `php-fpm` (Laravel), `laravel-queue`, `scheduler`. - `node-realtime` (Socket.IO server), - `mysql`, `redis`, `minio` (tuỳ chọn), `meilisearch` (tuỳ chọn), `soketi` (nếu dùng Pusher protocol). 3) **Triển khai**: - IaC Terraform/Ansible (tuỳ hạ tầng) tạo máy chủ, security group, LB. - SSL: Let's Encrypt (Caddy/Traefik/Nginx). HTTP/2, HSTS. 4) **CI/CD**: lint+test → build images → push → run migrations (Laravel) → healthchecks → rolling/blue-green. 5) **Observability**: - Logs ứng dụng → Fluent Bit → OpenSearch/Elasticsearch → Dashboards. - Metrics: node exporter, php-fpm exporter, mysql exporter → Prometheus → Grafana. - Alerts: Slack/Telegram/Zalo.

---

10) Lộ trình sản phẩm (MVP → Mở rộng)

Sprint 1 (Tuần 1–2): Auth, RBAC, tenant guard; CRUD Building/Unit; FE dashboard; khởi tạo **Node realtime** + kết nối Redis.

Sprint 2 (Tuần 3–4): Lease + Invoice engine; nhập chỉ số; generate hóa đơn; RN app skeleton (Auth, Dashboard cư dân); Socket.IO nhận thông báo.

Sprint 3 (Tuần 5–6): Marketplace cơ bản: đăng tin + duyệt; chat realtime (web + RN); đặt đơn nội bộ (không escrow); push notification FCM.

Sprint 4 (Tuần 7–8): **Sepay webhook** cho hóa đơn; trang thanh toán & đối soát; báo cáo cơ bản; cứng hoá bảo mật, rate limit; load test.

Post-MVP: Escrow nhẹ cho marketplace, đánh giá người bán/mua, IoT công tơ, lịch bảo trì nâng cao, mở portal kỹ thuật viên.

---

11) UX chính (Wireframe mô tả chữ)

- **Dashboard Admin Tòa**: Tổng quan doanh thu tháng, số phòng trống, hóa đơn quá hạn, ticket mở.

- **Quản lý Phòng**: bảng phòng theo tầng, badge trạng thái; click vào phòng → hợp đồng, công tơ, cư dân.
- **Hóa đơn**: list theo kỳ, filter trạng thái; modal chi tiết hóa đơn; nút "Gửi nhắc hạn".
- **Marketplace**: feed nội bộ, filter theo danh mục; form đăng tin (ảnh kéo thả), quy định tòa; chat bubble cố định.
- **Ticket**: Kanban (Open/In Progress/Done).

---

12) Kiểm thử & Chất lượng

- **Test levels**: Unit (Laravel Pest), Feature/E2E (Pest + Laravel Dusk/Playwright cho FE).
- **Performance**: k6/Locust test tạo 10–20k user, spike marketplace.
- **Security**: OWASP ASVS checklist, kiểm thử XSS/CSRF/IDOR, kiểm thử webhook giả mạo.
- **Data**: migration versioned, seed dataset demo, script anonymize khi clone prod → staging.

---

13) Ước lượng chi phí hạ tầng (tham khảo – 1 tòa đến 50 tòa)

- VPS 4 vCPU/8GB: ~30–60 USD/tháng (hoặc cloud tương đương).
- Postgres managed (tùy chọn) 2 vCPU/4GB: ~40–80 USD/tháng.
- S3/MinIO storage: 50–100 GB: ~5–10 USD/tháng.
- Pusher/Soketi: tự host gần như 0$, Pusher trả phí nếu dùng.
- Email (Amazon SES/Resend): theo số lượng.
- SMS/Zalo OA: theo nhà mạng.

---

14) Rủi ro & Giảm thiểu

- **Sai lệch chỉ số công tơ** → yêu cầu ảnh đối chứng + xác nhận 2 bước.
- **Tranh chấp giao dịch chợ** → rõ điều khoản, log chat & ảnh; giai đoạn 2 mới mở escrow.
- **Rò rỉ dữ liệu tenant** → RLS + policy chặt + pentest định kỳ.
- **Thanh toán lỗi** → idempotency key + retry queue + reconcile định kỳ.

---

15) Checklist bàn giao MVP

- [ ] RBAC + Guard theo tenant
- [ ] CRUD Building/Unit/Resident/Lease
- [ ] Invoice engine + công tơ + gửi hóa đơn
- [ ] Payment hoá đơn + webhook
- [ ] Ticket + Announcements
- [ ] Marketplace cơ bản + chat realtime
- [ ] Báo cáo cơ bản + export
- [ ] CI/CD + Monitoring + Backup

16) Mẫu đoạn code (gợi ý)

Laravel → Publish sự kiện sang Redis

```
// app/Events/InvoicePaid.php
class InvoicePaid implements ShouldBroadcastNow {
  use InteractsWithSockets, SerializesModels;
  public function __construct(public int $buildingId, public int $invoiceId,
public int $userId) {}
  public function broadcastOn() { return ['building:'.$this->buildingId]; }
  public function broadcastAs() { return 'invoice.paid'; }
}
```

> Nếu dùng Node làm realtime trung gian, thay vì Broadcasting Driver trực tiếp, có thể `Redis::publish('events', json_encode([...]))` và để Node subscribe.

Laravel – Webhook Sepay (xác thực & idempotency)

```
// routes/api.php
Route::post('/payments/sepay/webhook', [SepayWebhookController::class,
'handle']);

// app/Http/Controllers/Webhooks/SepayWebhookController.php
public function handle(Request $r) {
    $payload = $r->getContent();
    $sig = $r->header('X-Sepay-Signature');
    if(!$this->verifySignature($payload, $sig,
config('services.sepay.secret'))){
        abort(401, 'Invalid signature');
    }
    $data = $r->json()->all();
    $eventId = $data['event_id'] ?? ($data['txn_id'] ?? Str::uuid()-
>toString());
    if(!Cache::add('sepay:webhook:'.$eventId, 1, 3600)) {
        return response()->json(['ok'=>true,'dup'=>true]);
    }

    DB::transaction(function() use ($data) {
        $payment = Payment::updateOrCreate(
            ['txn_id' => $data['txn_id']],
            [
              'provider' => 'sepay',
              'amount' => $data['amount'],
              'status' => $data['status'] === 'SUCCESS' ? 'success' :
'failed',
              'raw' => $data,
```

```
                'paid_at' => now()
            ]
        );
        if($payment->invoice_id && $payment->status === 'success'){
            Invoice::where('id',$payment->invoice_id)-
>update(['status'=>'paid']);
            // publish event cho Node realtime
            Redis::publish('events', json_encode([
              'type'=>'invoice.paid',
              'building_id'=>$payment->building_id,
              'invoice_id'=>$payment->invoice_id,
              'user_ids'=>[$payment->payer_user_id ?? null]
            ]));
        }
    });
    return response()->json(['ok'=>true]);
}
```

Node.js Realtime (Socket.IO + Redis)

```
// server.js
const { createClient } = require('redis');
const { Server } = require('socket.io');
const http = require('http');

const server = http.createServer((req, res) => { res.writeHead(200);
res.end('OK'); });
const io = new Server(server, { cors: { origin: '*' } });

io.on('connection', (socket) => {
  // client gửi building_id để join phòng
  socket.on('join', ({ buildingId, userId }) => {
    if (buildingId) socket.join(`building:${buildingId}`);
    if (userId) socket.join(`user:${userId}`);
  });
});

(async () => {
  const sub = createClient({ url: process.env.REDIS_URL });
  await sub.connect();
  await sub.subscribe('events', (message) => {
    const evt = JSON.parse(message);
    if (evt.type === 'invoice.paid') {
      io.to(`building:${evt.building_id}`).emit('invoice.paid', evt);
      (evt.user_ids||[]).forEach(uid => io.to(`user:${uid}
`).emit('invoice.paid', evt));
    }
    if (evt.type === 'chat.message.created') {
      io.to(`chat:${evt.chat_id}`).emit('chat.message.created', evt);
    }
```

```
  });
  server.listen(process.env.PORT || 3001);
})();
```

React Native – kết nối Socket.IO

```
// useSocket.ts
import { io } from 'socket.io-client';
import { useEffect, useRef } from 'react';

export function useSocket({ buildingId, userId }: { buildingId?: number;
userId?: number; }) {
  const socketRef = useRef<any>();
  useEffect(() => {
    const socket = io(process.env.EXPO_PUBLIC_RT_URL || 'https://
rt.example.com', { transports: ['websocket'] });
    socket.emit('join', { buildingId, userId });
    socketRef.current = socket;
    return () => socket.disconnect();
  }, [buildingId, userId]);
  return socketRef;
}
```

---

17) Bước tiếp theo

1) Khởi tạo **Docker Compose** với `mysql`, `redis`, `php-fpm`, `nginx`, `node-realtime`, `minio`. 2) Sinh **migrations MySQL** theo schema trên + seed dữ liệu mẫu. 3) Tạo **service Sepay** (config, verify signature, webhook route) + tài liệu đối soát. 4) Tạo RN app (Expo) với module Auth, Invoice list/detail, Ticket, Marketplace feed + chat, tích hợp Socket.IO.