**PART 4: KEY FEATURES TO IMPLEMENT**

**FEATURE 1: Authentication & Authorization**

Create JWT-based authentication with role-based access control:

**Requirements:**

* Login page with email/password
* JWT tokens stored in httpOnly cookies
* Role-based route protection (board sees everything, owners see only their unit)
* Password reset functionality
* 2FA for board members (optional for MVP)

**Key files:**

* server/src/middleware/auth.ts - JWT verification
* server/src/routes/auth.ts - Login, logout, refresh token
* client/src/components/Login.tsx - Login UI
* client/src/hooks/useAuth.ts - Auth state management

**FEATURE 2: Board Dashboard**

Real-time financial overview with key metrics:

**Requirements:**

* Display total cash (operating + reserves + special assessment)
* Show accounts receivable and delinquency summary
* Alert system for critical issues (overdue financials, late vendors)
* Quick action buttons (approve payments, send notifications)
* Recent activity feed

**Key components:**

* client/src/pages/BoardDashboard.tsx
* client/src/components/dashboard/FinancialSummary.tsx
* client/src/components/dashboard/AlertsList.tsx
* client/src/components/dashboard/ActivityFeed.tsx

**Data to display:**

* Operating balance: $7,488.01
* Reserve balance: $20,385.87
* Special assessment: $253,903.33
* Total AR: $129,214.02
* Delinquent (>90 days): $70,584.35
* Number of delinquent units: 6

**FEATURE 3: Owner Portal**

Self-service portal for unit owners:

**Requirements:**

* View account balance in real-time
* See assessment breakdown (maintenance + reserve + special)
* Payment history with downloadable receipts
* Make payments via Stripe (credit card or ACH)
* View liens/violations on their unit
* Download monthly statements
* Auto-pay enrollment

**Key components:**

* client/src/pages/OwnerPortal.tsx
* client/src/components/owners/AccountSummary.tsx
* client/src/components/owners/PaymentForm.tsx
* client/src/components/owners/PaymentHistory.tsx
* client/src/components/owners/DocumentDownload.tsx

**Payment Integration:**

* Stripe Checkout for one-time payments
* Stripe Payment Intents for saved cards
* Webhook handling for payment confirmation
* Automatic receipt generation (PDF)

**FEATURE 4: Vendor Management**

Track all vendors and payment status:

**Requirements:**

* Vendor database with contact info and contracts
* Invoice submission workflow (management uploads, board approves)
* Payment approval queue with one-click approval
* Automatic late payment alerts
* Contract expiration reminders
* Payment history by vendor
* 1099 tracking

**Key components:**

* client/src/pages/VendorManagement.tsx
* client/src/components/vendors/VendorList.tsx
* client/src/components/vendors/InvoiceApproval.tsx
* client/src/components/vendors/PaymentQueue.tsx

**Workflow:**

1. Management uploads invoice (PDF) → Status: PENDING\_APPROVAL
2. Board receives email notification
3. Board reviews and approves/rejects
4. Approved invoices queued for payment
5. Payment processed (check # or ACH confirmation)
6. Status updated to PAID
7. Vendor notified via email

**FEATURE 5: Delinquency Tracking & Collections**

Automated collections workflow:

**Requirements:**

* Delinquency dashboard showing all late accounts
* Automatic escalation based on days late
* Email/SMS reminders at 30/60/90 days
* Demand letter generation (PDF)
* Lien tracking and documentation
* Payment plan management
* Attorney referral tracking

**Key components:**

* client/src/pages/Collections.tsx
* client/src/components/collections/DelinquentList.tsx
* client/src/components/collections/CollectionTimeline.tsx
* client/src/components/collections/LetterGenerator.tsx

**Automated Actions (use node-cron):**

javascript

*// server/src/services/collections.ts*

- Day 30: Send reminder email

- Day 60: Apply $25 late fee + send second notice

- Day 75: Generate demand letter

- Day 90: Send pre-lien notice

- Day 120: Flag for attorney referral

**FEATURE 6: Management Company Accountability**

Track management performance against contract SLAs:

**Requirements:**

* Performance scorecard with key metrics
* Financial report delivery tracking (due 25th of month)
* Response time tracking (board: 1.5 days, owners: 3 days)
* Vendor payment on-time rate
* Task assignment and completion tracking
* Contract violation alerts
* Automatic SLA breach notifications

**Key components:**

* client/src/pages/ManagementPerformance.tsx
* client/src/components/management/Scorecard.tsx
* client/src/components/management/TaskTracker.tsx
* client/src/components/management/ContractViolations.tsx

**Metrics to track:**

* Financial reports on-time: 14% (1 of 7) - FAILING
* Average response time: 8.3 days (target: 1.5) - FAILING
* Vendor payments on-time: 12% - FAILING
* Overall contract compliance: 23% - FAILING

**FEATURE 7: Document Management**

Centralized document repository:

**Requirements:**

* Upload documents with category tags
* Version control (track changes)
* Expiration date tracking (insurance, contracts)
* Access control (board-only, all-owners, public)
* Search by category, date, keyword
* Automatic expiration alerts (30/60/90 days before)

**Key components:**

* client/src/pages/Documents.tsx
* client/src/components/documents/DocumentUpload.tsx
* client/src/components/documents/DocumentList.tsx
* client/src/components/documents/DocumentViewer.tsx

**Categories:**

* Insurance policies
* Vendor contracts
* Meeting minutes
* Financial statements
* Governing documents (declaration, bylaws, rules)
* Inspection reports
* Violation notices

**FEATURE 8: Reporting**

Generate financial and operational reports:

**Requirements:**

* Monthly financial statement (balance sheet + income statement)
* Budget vs. actual comparison
* Accounts receivable aging report
* Delinquency summary
* Vendor payment history
* Reserve fund status
* Annual owner statement
* Board meeting package

**Key components:**

* client/src/pages/Reports.tsx
* client/src/components/reports/ReportGenerator.tsx
* client/src/components/reports/FinancialStatement.tsx
* client/src/components/reports/ARAgingReport.tsx

**Export formats:**

* PDF (for official distribution)
* Excel (for board analysis)
* CSV (for accounting software import)

**PART 5: UI/UX REQUIREMENTS**

**Design System:**

Use shadcn/ui components for consistency:

* Install: npx shadcn-ui@latest init
* Components to use: Button, Card, Table, Dialog, Dropdown, Badge, Alert, Tabs

**Color Palette:**

css

:root {

--primary: #0066CC; */\* Trust blue \*/*

--success: #00CC66; */\* Green - paid/compliant \*/*

--warning: #FFCC00; */\* Yellow - pending \*/*

--danger: #CC0000; */\* Red - overdue \*/*

--neutral: #666666; */\* Gray - inactive \*/*

--background: #F5F5F5; */\* Light gray \*/*

--card: #FFFFFF; */\* White \*/*

}

**Typography:**

* Headings: Inter (bold)
* Body: Inter (regular)
* Monospace (numbers): JetBrains Mono

**Responsive Design:**

* Mobile-first approach
* Breakpoints: 640px (sm), 768px (md), 1024px (lg), 1280px (xl)
* All tables scrollable on mobile
* Collapsible sidebar navigation

**Key UI Components:**

**1. Dashboard Card:**

tsx

<Card className="p-6">

<div className="flex justify-between items-start">

<div>

<p className="text-sm text-gray-600">Operating Balance</p>

<h3 className="text-3xl font-bold mt-2">$7,488.01</h3>

</div>

<Badge variant={status === 'healthy' ? 'success' : 'danger'}>

{status}

</Badge>

</div>

</Card>

**2. Delinquency Status Badge:**

tsx

const getStatusColor = (days: number) => {

if (days === 0) return 'bg-green-500';

if (days < 60) return 'bg-yellow-500';

if (days < 90) return 'bg-orange-500';

return 'bg-red-500';

};

**3. Data Table with Sorting:** Use TanStack Table (React Table v8) for all data tables:

* Sortable columns
* Pagination
* Search/filter
* Export to CSV

**PART 6: API ENDPOINTS**

**Authentication:**

* POST /api/auth/login - Login with email/password
* POST /api/auth/logout - Logout (clear cookie)
* POST /api/auth/refresh - Refresh JWT token
* POST /api/auth/forgot-password - Send reset link
* POST /api/auth/reset-password - Reset password with token

**Dashboard:**

* GET /api/dashboard/board - Board overview data
* GET /api/dashboard/owner/:unitId - Owner portal data
* GET /api/dashboard/alerts - Critical alerts

**Units & Owners:**

* GET /api/units - List all units (board only)
* GET /api/units/:id - Get unit details
* GET /api/units/:id/account - Get owner account balance
* GET /api/units/:id/transactions - Payment history
* PATCH /api/units/:id - Update unit info (board only)

**Payments:**

* POST /api/payments/intent - Create Stripe payment intent
* POST /api/payments/confirm - Confirm payment
* GET /api/payments/:id/receipt - Download receipt PDF
* POST /api/payments/autopay - Enroll in autopay
* DELETE /api/payments/autopay - Cancel autopay

**Vendors:**

* GET /api/vendors - List all vendors
* POST /api/vendors - Add new vendor (board only)
* PATCH /api/vendors/:id - Update vendor
* DELETE /api/vendors/:id - Delete vendor

**Invoices:**

* GET /api/invoices - List all invoices
* GET /api/invoices/pending - Pending approval queue
* POST /api/invoices - Upload new invoice (management)
* PATCH /api/invoices/:id/approve - Approve invoice (board)
* PATCH /api/invoices/:id/reject - Reject invoice (board)
* PATCH /api/invoices/:id/pay - Mark as paid (management)

**Collections:**

* GET /api/collections/delinquent - List delinquent accounts
* POST /api/collections/letters - Generate collection letter
* PATCH /api/collections/:unitId/lien - Update lien status
* GET /api/collections/timeline/:unitId - Collection activity

**Documents:**

* GET /api/documents - List documents (filtered by access level)
* POST /api/documents - Upload document
* GET /api/documents/:id - Download document
* DELETE /api/documents/:id - Delete document (board only)

**Reports:**

* GET /api/reports/financial?month=2025-05 - Financial statement
* GET /api/reports/ar-aging - AR aging report
* GET /api/reports/delinquency - Delinquency summary
* GET /api/reports/vendor-payments - Vendor payment history

**Management Performance:**

* GET /api/management/scorecard - Performance metrics
* GET /api/management/violations - Contract violations
* GET /api/management/tasks - Assigned tasks
* PATCH /api/management/tasks/:id - Update task status

**PART 7: IMPLEMENTATION STEPS**

Follow this order for building:

**PHASE 1: Foundation (Week 1)**

1. Set up project structure (client + server)
2. Install dependencies
3. Create Prisma schema
4. Run database migrations
5. Create seed data
6. Set up environment variables
7. Implement basic authentication

**PHASE 2: Core Features (Week 2-3)** 8. Build board dashboard 9. Build owner portal 10. Implement payment processing (Stripe) 11. Create vendor management 12. Build document upload/storage

**PHASE 3: Advanced Features (Week 4)** 13. Implement delinquency tracking 14. Build management performance tracker 15. Create reporting system 16. Add email notifications

**PHASE 4: Polish (Week 5)** 17. Responsive design testing 18. Error handling and validation 19. Performance optimization 20. Security audit 21. User testing and bug fixes

**PART 8: ENVIRONMENT VARIABLES**

Create .env file in server directory:

env

# Database

DATABASE\_URL="postgresql://user:password@localhost:5432/heritage\_condo"

# JWT

JWT\_SECRET="your-super-secret-jwt-key-change-this"

JWT\_REFRESH\_SECRET="your-refresh-secret-key-change-this"

# Stripe

STRIPE\_SECRET\_KEY="sk\_test\_..."

STRIPE\_PUBLISHABLE\_KEY="pk\_test\_..."

STRIPE\_WEBHOOK\_SECRET="whsec\_..."

# Email (for notifications)

SMTP\_HOST="smtp.gmail.com"

SMTP\_PORT=587

SMTP\_USER="heritage.condo@gmail.com"

SMTP\_PASS="your-app-password"

# Application

NODE\_ENV="development"

PORT=3001

CLIENT\_URL="http://localhost:5173"

# Optional: File storage (for documents)

AWS\_S3\_BUCKET="heritage-condo-docs"

AWS\_ACCESS\_KEY\_ID="..."

AWS\_SECRET\_ACCESS\_KEY="..."

**PART 9: CRITICAL SECURITY REQUIREMENTS**

**Must implement:**

1. **Password Requirements:**
   * Minimum 12 characters
   * At least 1 uppercase, 1 lowercase, 1 number, 1 special char
   * Bcrypt hashing with salt rounds = 12
2. **Rate Limiting:**
   * Login attempts: 5 per 15 minutes per IP
   * API calls: 100 per minute per user
   * Payment attempts: 3 per hour per user
3. **Input Validation:**
   * Use Zod for all API input validation
   * Sanitize file uploads (only PDF, JPG, PNG)
   * Maximum file size: 10MB
4. **SQL Injection Protection:**
   * Prisma ORM handles this automatically
   * Never use raw SQL queries
5. **XSS Protection:**
   * Sanitize all user inputs
   * Use React's built-in XSS protection
   * Set proper CSP headers
6. **CSRF Protection:**
   * Use httpOnly cookies for JWT
   * Implement CSRF tokens for state-changing operations
7. **HTTPS Only:**
   * Force HTTPS in production
   * Set Secure flag on cookies

**PART 10: TESTING REQUIREMENTS**

**Unit Tests:**

* Test all API endpoints
* Test authentication logic
* Test payment processing
* Test delinquency calculations

**Integration Tests:**

* Test full payment workflow
* Test invoice approval workflow
* Test collection escalation

**E2E Tests (Playwright):**

* Test user login flow
* Test owner making payment
* Test board approving invoice