Phase 2 Implementation Checklist

Revenue Cycle Management Build | FSC Portal

■ Week-by-Week Implementation Plan

Week 1: Database Foundation	Status
■ Run SQL scripts in Supabase (all 8 steps)	■
■ Verify all tables created successfully	■
■ Test views (dashboard_metrics, aging_report)	■
■ Set up status history trigger	■
■ Backfill existing sessions with status	■
■ Document database changes in team wiki	■

Week 2: Core UI Components	Status
■ Build StatusBadge component	■
■ Build StatusDropdown with validation	■
■ Build AlertBanner component	■
■ Build SessionCard component	■
■ Update existing Sessions page to show new status	■
■ Add ability to change status from Sessions page	■

Week 3: Revenue Dashboard	Status
■ Create /billing/revenue route	■
■ Build 4 stat cards (billed, outstanding, collected, avg days)	■
■ Build aging summary table	■
■ Build recent payments table	■
■ Add collections trend chart (Chart.js)	■
■ Add role-based access (admin, director only)	■

Week 4: Aging Report	Status
■ Create /billing/aging route	■
■ Add filters (age bracket, payer, provider, status)	■
■ Build aging table with color-coded rows	■
■ Add row actions dropdown	■
■ Implement bulk selection	■
■ Add export to CSV function	■

Week 5: Payment Entry	Status

■ Create /billing/payments route	■
■ Build payment entry form	■
■ Add session search & selection	■
■ Validate payment amount matches selected sessions	■
■ Auto-update session status to "paid" on save	■
■ Build recent payments table with edit/delete	■

Week 6: Claim Tracker & Session Detail	Status
■ Create /billing/claims route	■
■ Add search bar & status filter tabs	■
■ Build main claims table	■
■ Add pagination	■
■ Create /sessions/[id] detail page	■
■ Show full session info + timeline	■
■ Add quick actions (update status, add note)	■

Week 7: Work Queue & Alerts	Status
■ Create /work-queue route	■
■ Build alert banners (critical, warning, info)	■
■ Show assigned sessions grouped by urgency	■
■ Add quick action buttons	■
■ Set up daily cron job for aging alerts	■
■ Test alert generation logic	■

Week 8: Polish & Testing	Status
■ Update sidebar navigation with new routes	■
■ Add loading states to all async operations	■
■ Implement error handling & user feedback	■
■ Mobile responsive testing	■
■ Create user training documentation	■
■ Deploy to production	■
■ Train billing staff on new features	■

■ Critical Dependencies

NPM Packages Needed: • @supabase/supabase-js (already installed) • chart.js + svelte-chartjs (for revenue trend chart) • date-fns (for date formatting & calculations) **Supabase Features to Enable:** • Realtime (optional, for live status updates) • Cron Jobs (for automated aging alerts) • Email notifications (for alert emails) **External Integrations (Future):** • Consider API connections to major insurance portals • Automated claim status checking services • Payment processing integration

■ Testing Checklist

Test Area	What to Test	Pass
Status Transitions	Can only move to valid next statuses	■
Payment Validation	Amount must match selected sessions	■
Role Permissions	Billing sees only their work queue	■
Alert Generation	Alerts created for 30+ day old claims	■
Data Integrity	Status changes logged in history	■
CSV Import	New sessions get default "completed" status	■
Mobile UI	All pages usable on phone/tablet	■
Performance	Aging report loads <2 seconds	■

■ Quick Wins (Do First!)

These will give immediate value with minimal effort: 1. **Add billing_status to sessions table** (5 minutes) • Even just tracking "paid" vs "unpaid" is huge progress 2. **Create simple aging query** (10 minutes) • SELECT * FROM sessions WHERE date < NOW() - INTERVAL '30 days' AND billing_status != 'paid' 3. **Add status badges to existing Sessions page** (30 minutes) • Visual indicator helps staff immediately 4. **Build basic payment entry form** (2 hours) • Start recording what's actually been paid 5. **Dashboard metric cards** (2 hours) • Shows total outstanding, avg days to payment