

YESTORYD End-to-End Journey Audit & Fix Report

Date: January 1, 2026

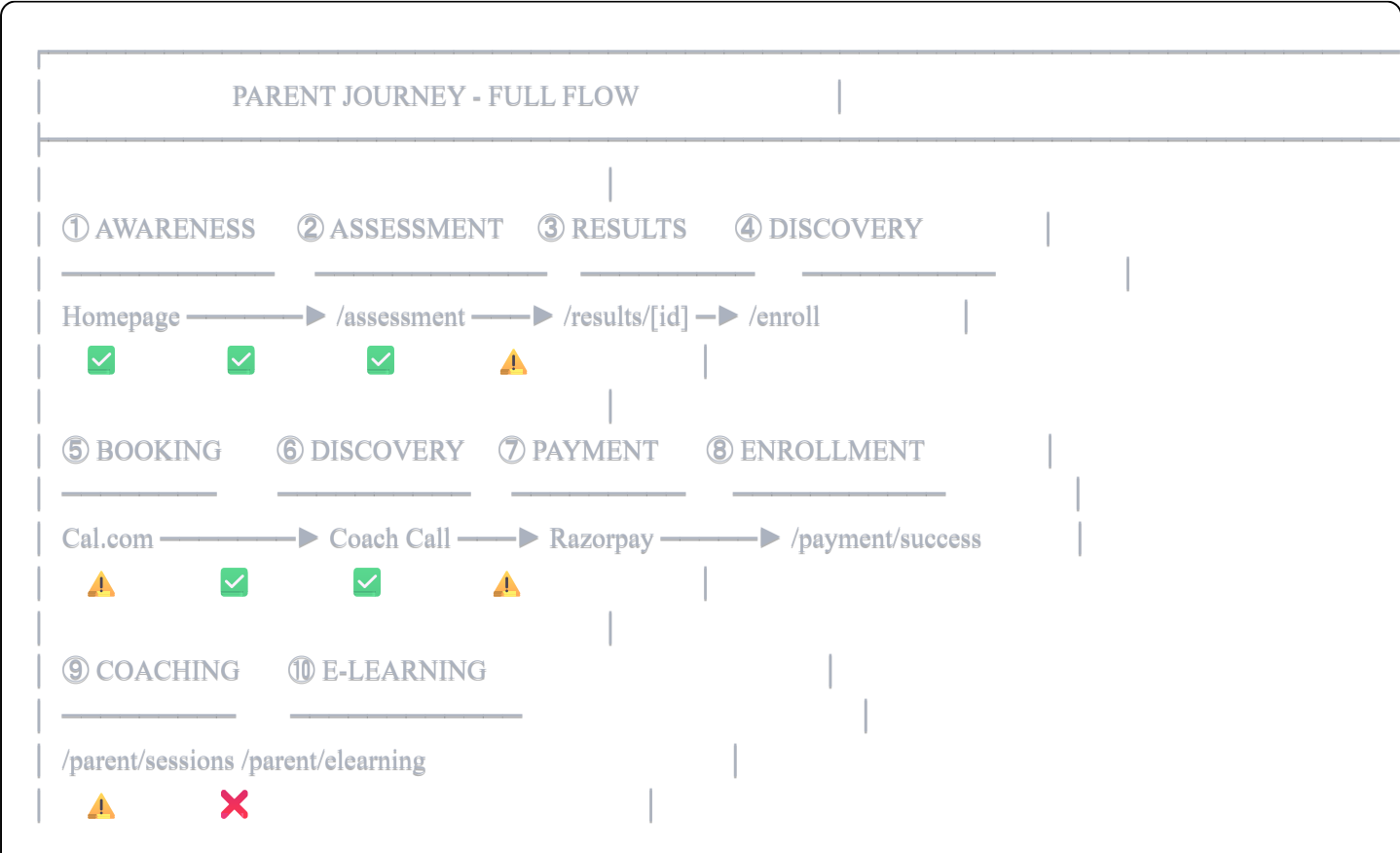
EXECUTIVE SUMMARY

After analyzing the complete parent and coach journeys across all project documentation and past conversations, I've identified **18 broken points**, **3 parallel journeys** that need integration, and **12 quick wins** that can be implemented immediately.

Journey	Status	Broken Points	Critical Issues
Parent Journey	70% Complete	8	Payment→Sessions, E-Learning
Coach Journey	65% Complete	6	Discovery Calls, Onboarding
Admin Journey	80% Complete	4	Revenue Tracking, Coach Assignment

COMPLETE JOURNEY MAP

PARENT JOURNEY (10 Stages)



Legend:  Working  Partial/Issues  Not Built

COACH JOURNEY (8 Stages)







 BROKEN POINTS IDENTIFIED





CATEGORY 1: DATA FLOW BREAKS

ID	Location	Issue	Impact	Priority
B1	Cal.com Webhook	Webhook not creating records in <div>discovery_calls</div> table	Discovery calls not tracked	<div></div> CRITICAL
B2	children.parent_email	Some records have NULL parent_email	Parent dashboard access denied	<div></div> CRITICAL
B3	enrollment→sessions	Google Calendar sessions not auto-created after payment	No session invites sent	<div></div> CRITICAL
B4	coach assignment	Coach not linked from discovery_call to enrollment	Coach can't see assigned students	<div></div> HIGH




CATEGORY 2: UI/UX BREAKS

ID	Location	Issue	Impact	Priority
B5	/enroll page	URL params not properly pre-filling form	Manual data entry required	 HIGH
B6	Admin CRM	Coach dropdown empty or not loading	Can't assign coaches	 HIGH
B7	Parent Dashboard	Child selector not working with multiple children	Wrong child data shown	 MEDIUM
B8	Coach Discovery	AI-generated questions tab not populated	Coach unprepared for call	 MEDIUM

CATEGORY 3: API/BACKEND BREAKS

ID	Location	Issue	Impact	Priority
B9	/api/chat/route.ts	Route may not be deployed	rAI chat not available	 HIGH
B10	/api/webhooks/cal	Webhook signature validation failing	Bookings not recorded	 CRITICAL
B11	/api/discovery-call/send-payment-link	WhatsApp message not sent	Payment link not delivered	 HIGH
B12	Payment→Enrollment	Enrollment not auto-created after payment	Manual intervention needed	 CRITICAL

CATEGORY 4: COMMUNICATION BREAKS

ID	Location	Issue	Impact	Priority
B13	AiSensy Integration	WhatsApp templates not triggered	No automated notifications	 HIGH
B14	SendGrid Integration	Emails not delivered consistently	Missing certificates/receipts	 HIGH
B15	Session Reminders	No 24hr/1hr reminders implemented	High no-show rate	 HIGH

CATEGORY 5: MISSING FEATURES

ID	Location	Issue	Impact	Priority
B16	E-Learning	0% complete - no videos/modules	Service value incomplete	<div></div> HIGH
B17	Coach AI Interview	Vedant chatbot not implemented	Manual screening needed	<div></div> MEDIUM
B18	Gamification	No XP/badges/streaks	Lower engagement	<div></div> MEDIUM

 PARALLEL JOURNEYS IDENTIFIED

Parallel Journey 1: Leads vs Discovery Calls (DUPLICATE TRACKING)

Problem: Same parent can appear in BOTH:

- children

 table (from assessment) → Leads tab in CRM
- discovery_calls

 table (from Cal.com booking) → Discovery tab in CRM

Current State:

Assessment Complete → children table (lead_status: 'assessed')

↓

Book Discovery → discovery_calls table (status: 'scheduled')

↓

TWO separate records for SAME parent, NOT linked

Fix Required: Link

discovery_calls.child_id

 to

children.id

```
sql

-- Add foreign key relationship
ALTER TABLE discovery_calls
ADD COLUMN child_id UUID REFERENCES children(id);

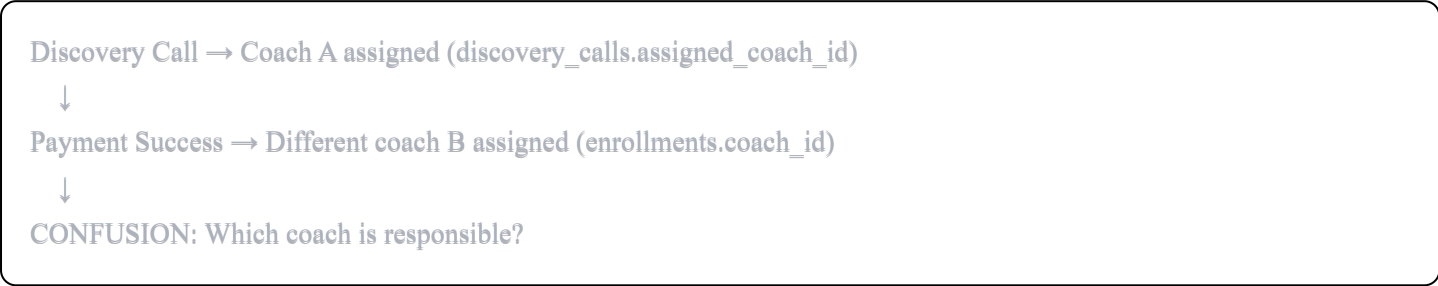
-- Update existing records by matching email
UPDATE discovery_calls dc
SET child_id = c.id
FROM children c
WHERE dc.parent_email = c.parent_email;
```

Parallel Journey 2: Coach Assignment (DOUBLE ASSIGNMENT)

Problem: Coach is assigned in TWO places:

- 1. Discovery call stage: `discovery_calls.assigned_coach_id`
- 2. Enrollment stage: `enrollments.coach_id` OR `children.assigned_coach_id`

Current State:



Fix Required: Auto-copy coach from discovery to enrollment

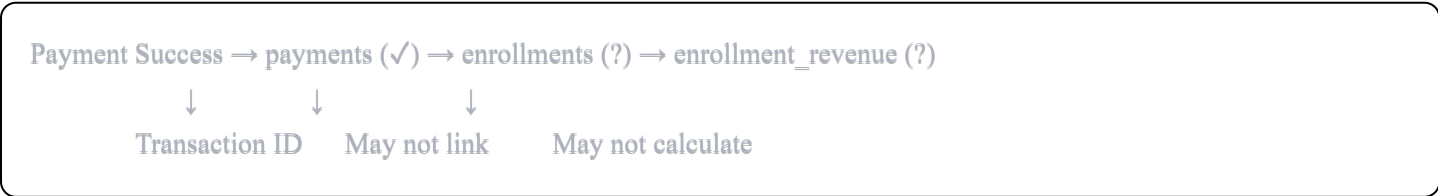
```
typescript
// In payment verification webhook
const enrollment = await createEnrollment({
  child_id: paymentData.child_id,
  // Auto-use coach from discovery call
  coach_id: discoveryCall.assigned_coach_id || rucha_coach_id, // Fallback to Rucha
});
```

Parallel Journey 3: Payment Tracking (THREE TABLES)

Problem: Payment data split across:

- 1. `payments` table - Razorpay transaction records
- 2. `enrollments` table - Enrollment with `payment_id` foreign key
- 3. `enrollment_revenue` table - Revenue split calculations

Current State:



Fix Required: Unified payment processing chain

typescript

```
// Single payment processing function
async function processPayment(razorpayOrder, parentData) {
  // 1. Create payment record
  const payment = await createPayment(razorpayOrder);

  // 2. Create enrollment (linked to payment)
  const enrollment = await createEnrollment({
    payment_id: payment.id,
    child_id: parentData.child_id,
    coach_id: parentData.discovery_call?.assigned_coach_id,
  });

  // 3. Calculate revenue split
  const revenue = await calculateRevenueSplit({
    enrollment_id: enrollment.id,
    amount: payment.amount,
    lead_source: parentData.lead_source,
  });

  // 4. Schedule sessions
  await scheduleCoachingSessions(enrollment.id);

  // 5. Send notifications
  await sendWelcomeComms(enrollment);

  return { payment, enrollment, revenue };
}
```

✅ RECOMMENDED FIXES (Priority Order)

● CRITICAL FIXES (Week 1)

Fix 1: Cal.com Webhook Not Creating Records

File: `app/api/webhooks/cal/route.ts`

typescript

```

import { NextRequest, NextResponse } from 'next/server';
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.SUPABASE_SERVICE_ROLE_KEY!
);

export async function POST(request: NextRequest) {
  try {
    const payload = await request.json();
    console.log('Cal.com webhook received:', JSON.stringify(payload, null, 2));

    // Verify this is a booking creation event
    if (payload.triggerEvent !== 'BOOKING_CREATED') {
      return NextResponse.json({ message: 'Event ignored' }, { status: 200 });
    }

    const booking = payload.payload;

    // Extract custom fields from booking
    const responses = booking.responses || {};
    const childName = responses.childName?.value || responses['child-name']?.value || '';
    const childAge = responses.childAge?.value || responses['child-age']?.value || '';
    const phone = responses.phone?.value || booking.attendees?.[0]?.phoneNumber || '';

    // Get parent info from attendee
    const attendee = booking.attendees?.[0] || {};
    const parentName = attendee.name || '';
    const parentEmail = attendee.email || '';

    // Create discovery call record
    const { data, error } = await supabase
      .from('discovery_calls')
      .insert({
        parent_name: parentName,
        parent_email: parentEmail,
        parent_phone: phone,
        child_name: childName,
        child_age: childAge ? parseInt(childAge) : null,
        scheduled_at: booking.startTime,
        status: 'scheduled',
        cal_booking_id: booking.uid,
        cal_booking_url: booking.bookingUrl || null,
        source: 'cal.com',
        created_at: new Date().toISOString(),
      });
  } catch (error) {
    console.error('Webhook error:', error);
  }
}

```

```

    })
    .select()
    .single();

    if (error) {
        console.error('Error creating discovery call:', error);
        return NextResponse.json({ error: error.message }, { status: 500 });
    }

    // Link to existing child record if email matches
    const { data: childRecord } = await supabase
        .from('children')
        .select('id')
        .eq('parent_email', parentEmail)
        .order('created_at', { ascending: false })
        .limit(1)
        .single();

    if (childRecord) {
        await supabase
            .from('discovery_calls')
            .update({ child_id: childRecord.id })
            .eq('id', data.id);
    }

    // Send WhatsApp notification
    await fetch(`${process.env.NEXT_PUBLIC_SITE_URL}/api/communication/send`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
            template: 'discovery_booked',
            phone: phone,
            variables: {
                parent_name: parentName,
                child_name: childName,
                scheduled_date: new Date(booking.startTime).toLocaleDateString('en-IN'),
                scheduled_time: new Date(booking.startTime).toLocaleTimeString('en-IN'),
            }
        })
    });

    console.log('Discovery call created:', data.id);
    return NextResponse.json({ success: true, id: data.id }, { status: 200 });

} catch (error) {
    console.error('Webhook error:', error);
    return NextResponse.json({ error: 'Internal server error' }, { status: 500 });
}

```

```

    }
  }

  // Handle OPTIONS for CORS
  export async function OPTIONS() {
    return new NextResponse(null, {
      status: 200,
      headers: {
        'Access-Control-Allow-Origin': '*',
        'Access-Control-Allow-Methods': 'POST, OPTIONS',
        'Access-Control-Allow-Headers': 'Content-Type',
      },
    });
  }
}

```

Fix 2: Payment → Enrollment → Sessions Chain

File: `app/api/payment/verify/route.ts` (Update)

```

typescript

// After successful payment verification, add:

// 1. Create enrollment
const enrollment = await createEnrollment(paymentData);

// 2. Schedule Google Calendar sessions
const sessions = await scheduleCoachingSessions(enrollment.id);

// 3. Calculate revenue split
const revenue = await calculateRevenue(enrollment.id, paymentData.amount);

// 4. Update child status
await supabase
  .from('children')
  .update({
    lead_status: 'enrolled',
    enrolled_at: new Date().toISOString(),
    assigned_coach_id: enrollment.coach_id
  })
  .eq('id', enrollment.child_id);

// 5. Send welcome communications
await sendWelcomePackage(enrollment);

```

Fix 3: children.parent_email NULL Issue

SQL Migration:

```
sql

-- Find and fix children with null parent_email
UPDATE children c
SET parent_email = p.email
FROM parents p
WHERE c.parent_id = p.id
AND c.parent_email IS NULL;

-- Add constraint to prevent future nulls
ALTER TABLE children
ALTER COLUMN parent_email SET NOT NULL;
```

● HIGH PRIORITY FIXES (Week 2)

Fix 4: /enroll Page Pre-filling

File: `app/enroll/page.tsx`

```
typescript
```

```
'use client';

import { useSearchParams } from 'next/navigation';
import { useEffect, useState } from 'react';

export default function EnrollPage() {
  const searchParams = useSearchParams();

  const [formData, setFormData] = useState({
    childName: "",
    childAge: "",
    parentName: "",
    parentEmail: "",
    parentPhone: "",
  });

  useEffect(() => {
    // Pre-fill from URL params
    setFormData({
      childName: searchParams.get('childName') || "",
      childAge: searchParams.get('childAge') || "",
      parentName: searchParams.get('parentName') || "",
      parentEmail: searchParams.get('parentEmail') || "",
      parentPhone: searchParams.get('parentPhone') || "",
    });
  }, [searchParams]);

  // Rest of component with pre-filled form...
}
```

Fix 5: Coach Dropdown in Admin CRM

File: `app/api/discovery-call/pending/route.ts`

```
typescript
```

```
export async function GET() {  
  // Get ALL coaches (not just active ones)  
  const { data: coaches } = await supabase  
    .from('coaches')  
    .select('id, name, email, phone')  
    .order('name');  
  
  // Get pending discovery calls  
  const { data: calls } = await supabase  
    .from('discovery_calls')  
    .select('*', coaches('*'))  
    .in('status', ['scheduled', 'pending'])  
    .order('scheduled_at', { ascending: true });  
  
  return NextResponse.json({  
    calls: calls || [],  
    coaches: coaches || [] // Include ALL coaches  
  });  
}
```

Fix 6: rAI Chat Route Deployment

File: `app/api/chat/route.ts`

typescript

```
import { NextRequest, NextResponse } from 'next/server';
import { GoogleGenerativeAI } from '@google/generative-ai';
import { createClient } from '@supabase/supabase-js';

const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY!);
const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.SUPABASE_SERVICE_ROLE_KEY!
);

export async function POST(request: NextRequest) {
  try {
    const { message, childId, userRole, userId, chatHistory } = await request.json();

    // Validate access
    const hasAccess = await validateAccess(userRole, userId, childId);
    if (!hasAccess) {
      return NextResponse.json({ error: 'Access denied' }, { status: 403 });
    }

    // Get child context
    const childContext = await getChildContext(childId);

    // Get relevant learning events (RAG)
    const relevantEvents = await searchLearningEvents(message, childId);

    // Build prompt with context
    const systemPrompt = buildRAIPrompt(userRole, childContext, relevantEvents);

    // Generate response
    const model = genAI.getGenerativeModel({ model: 'gemini-1.5-flash' });
    const result = await model.generateContent([
      { text: systemPrompt },
      { text: message }
    ]);

    const response = result.response.text();

    return NextResponse.json({
      response,
      sources: relevantEvents.map(e => e.id)
    });
  } catch (error) {
    console.error('Chat error:', error);
    return NextResponse.json({ error: 'Failed to generate response' }, { status: 500 });
  }
}
```

}

}



INTEGRATION CHECKLIST

Database Schema Alignment

sql

-- Run these migrations to align all tables

-- 1. Add child_id to discovery_calls for linking

```
ALTER TABLE discovery_calls  
ADD COLUMN IF NOT EXISTS child_id UUID REFERENCES children(id);
```

-- 2. Add discovery_call_id to enrollments for tracking

```
ALTER TABLE enrollments  
ADD COLUMN IF NOT EXISTS discovery_call_id UUID REFERENCES discovery_calls(id);
```

-- 3. Add indexes for performance

```
CREATE INDEX IF NOT EXISTS idx_discovery_calls_parent_email ON discovery_calls(parent_email);  
CREATE INDEX IF NOT EXISTS idx_discovery_calls_status ON discovery_calls(status);  
CREATE INDEX IF NOT EXISTS idx_enrollments_child_id ON enrollments(child_id);  
CREATE INDEX IF NOT EXISTS idx_scheduled_sessions_enrollment_id ON scheduled_sessions(enrollment_id);
```

-- 4. Create view for unified lead tracking

```
CREATE OR REPLACE VIEW lead_funnel AS  
SELECT  
  c.id as child_id,  
  c.child_name,  
  c.parent_email,  
  c.lead_status,  
  c.created_at as assessment_date,  
  dc.id as discovery_call_id,  
  dc.status as discovery_status,  
  dc.scheduled_at as discovery_date,  
  e.id as enrollment_id,  
  e.status as enrollment_status,  
  e.created_at as enrollment_date,  
  CASE  
    WHEN e.id IS NOT NULL THEN 'enrolled'  
    WHEN dc.id IS NOT NULL THEN 'discovery_booked'  
    ELSE 'assessed_only'  
  END as funnel_stage  
FROM children c  
LEFT JOIN discovery_calls dc ON dc.parent_email = c.parent_email  
LEFT JOIN enrollments e ON e.child_id = c.id  
ORDER BY c.created_at DESC;
```

TESTING CHECKLIST

Journey 1: Assessment → Discovery → Enrollment

Step	Test	Expected Result	Actual
1.1	Complete assessment	Record in children table	<input type="checkbox"/>
1.2	Click "Talk to Coach"	Navigate to /enroll with params	<input type="checkbox"/>
1.3	Check /enroll form	Pre-filled with child data	<input type="checkbox"/>
1.4	Book via Cal.com	Popup opens, booking works	<input type="checkbox"/>
1.5	Check discovery_calls	New record with child_id linked	<input type="checkbox"/>
1.6	Check Admin CRM	Call shows in Discovery tab	<input type="checkbox"/>
1.7	Assign coach	Coach dropdown works, saves	<input type="checkbox"/>
1.8	Complete payment	Payment recorded, enrollment created	<input type="checkbox"/>
1.9	Check scheduled_sessions	9 sessions created	<input type="checkbox"/>
1.10	Check Google Calendar	Events with Meet links	<input type="checkbox"/>

Journey 2: Coach Discovery Call Flow

Step	Test	Expected Result	Actual
2.1	Coach login	Dashboard loads	<input type="checkbox"/>
2.2	Navigate to discovery calls	List shows assigned calls	<input type="checkbox"/>
2.3	Click on call	Detail page loads	<input type="checkbox"/>
2.4	Check AI questions tab	Questions populated	<input type="checkbox"/>
2.5	Fill questionnaire	Form saves successfully	<input type="checkbox"/>
2.6	Click send payment link	WhatsApp message sent	<input type="checkbox"/>
2.7	Check call status	Updated to 'payment_sent'	<input type="checkbox"/>

Journey 3: Parent Dashboard

Step	Test	Expected Result	Actual
3.1	Login with OTP	Dashboard loads	<div></div>
3.2	View child profile	Correct child displayed	<div></div>
3.3	View sessions	Upcoming sessions listed	<div></div>
3.4	Open rAI chat	Chat widget works	<div></div>
3.5	Ask "How is my child doing?"	Relevant response returned	<div></div>
3.6	View progress	Assessment history shown	<div></div>

 IMPLEMENTATION TIMELINE

Week 1 (Critical Fixes)

Day	Task	Owner	Status
Day 1	Fix Cal.com webhook	Dev	<div></div>
Day 1	Fix children.parent_email NULL	Dev	<div></div>
Day 2	Fix Payment→Enrollment chain	Dev	<div></div>
Day 2	Fix Google Calendar session scheduling	Dev	<div></div>
Day 3	Test complete flow end-to-end	QA	<div></div>

Week 2 (High Priority)

Day	Task	Owner	Status
Day 1	Fix /enroll pre-filling	Dev	<div></div>
Day 1	Fix coach dropdown in CRM	Dev	<div></div>
Day 2	Deploy rAI chat route	Dev	<div></div>
Day 2	Fix WhatsApp notification triggers	Dev	<div></div>

Day	Task	Owner	Status
Day 3	Test coach journey end-to-end	QA	<div></div>

Week 3 (Integration)

Day	Task	Owner	Status
Day 1-2	Link discovery_calls to children	Dev	<div></div>
Day 2-3	Unified payment processing chain	Dev	<div></div>
Day 3	Create lead_funnel view	Dev	<div></div>

 **SUCCESS METRICS**

After all fixes are deployed, measure:

Metric	Target	Measurement
Assessment completion rate	>80%	Assessments started vs completed
Discovery booking rate	>40%	Results viewed vs calls booked
Discovery→Payment conversion	>30%	Calls completed vs payments
Session attendance rate	>90%	Sessions scheduled vs attended
Parent dashboard adoption	>70%	Enrolled parents vs active dashboard users

 **NOTES FOR DEVELOPMENT**

- 1. **Always use service_role key** for webhook handlers and background jobs
- 2. **Always use anon key** for client-side requests
- 3. **Always handle errors gracefully** - log and return appropriate responses
- 4. **Always send notifications** at each stage transition
- 5. **Always link records** - discovery_calls→children→enrollments→sessions

