🤖 PlombiPro - The Complete Master Development Guide (Supabase Edition)

This master guide merges and corrects all project documentation (Parts 1-5) into a single, cohesive plan. It follows your explicit request to prioritize free-tier, scalable services, pivoting away from the original Google Cloud-based plan.

Core Technology Stack (Corrected):

- **Framework:** Flutter
- **Database & Auth:** Supabase (PostgreSQL)
- **Backend Functions: Supabase Edge Functions (TypeScript/Deno)**
- **Scheduled Jobs: Supabase Cron Jobs (pg_cron)**
- **Core APIs:** Supabase, Stripe
- **Free Tier APIs:** ocr.space (for OCR), Resend (for Emails)
- **Target:** iOS App Store + Google Play Store
- **Language:** French UI

🎨 PART 1: DETAILED LAYOUT SCHEMAS & UI ARCHITECTURE

(This section is unchanged from plombipro_part1_layouts.md and remains the "bible" for all UI development.)

Layout System & Responsive Design

Device Breakpoints (Flutter responsive)

Dart

```dart
class ResponsiveBreakpoints {
  static const double mobile = 0;     // 0 - 599px
  static const double tablet = 600;    // 600 - 1199px
  static const double desktop = 1200;  // 1200px+

  static bool isMobile(double width) => width < tablet;
  static bool isTablet(double width) => width >= tablet && width < desktop;
  static bool isDesktop(double width) => width >= desktop;
}
```

Base Spacing System (8dp grid)

Dart

```dart
class AppSpacing {
  static const double xs = 4.0;   // Extra small
  static const double sm = 8.0;   // Small
  static const double md = 16.0;  // Medium (base)
  static const double lg = 24.0;  // Large
  static const double xl = 32.0;  // Extra large
  static const double xxl = 48.0; // Extra extra large
}
```

📱 Core Layout Schemas

1. HomePage (Dashboard) Layout Tree

```
SafeArea
  ├─ Scaffold
  │   ├─ AppBar
  │   │   ├─ Title: "PlombiPro"
  │   │   ├─ Actions: [Settings, Notifications]
  │   │   └─ Bottom: SearchBar
  │   │
  │   ├─ Drawer: AppDrawer()
  │   │
  │   └─ Body: CustomScrollView(
  │       ├─ SliverAppBar (sticky stats)
  │       │   ├─ UserGreeting
  │       │   ├─ CompanyName
  │       │   └─ CurrentDate
  │       │
  │       └─ SliverList
  │           ├─ QuickStatsSection (4 cards in 2x2 grid)
  │           │   ├─ Card: "CA du mois: 12,450€"
  │           │   ├─ Card: "Factures impayées: 3 (2,100€)"
  │           │   ├─ Card: "Devis en attente: 2"
  │           │   └─ Card: "RDV aujourd'hui: 1"
  │           │
  │           ├─ Divider
  │           │
  │           ├─ RecentActivitySection
  │           │   ├─ SectionTitle: "Activité récente"
  │           │   │
  │           │   └─ HorizontalListView (scrollable)
  │           │       ├─ QuoteCard
  │           │       ├─ InvoiceCard
  │           │       ├─ ClientCard
  │           │       └─ QuoteCard
  │           │
  │           ├─ Divider
  │           │
  │           ├─ QuickActionsSection
  │           │   └─ Wrap (4 items, responsive)
```

```
│       │       ├─ ActionButton: "+ Nouveau devis"
│       │       ├─ ActionButton: "+ Nouvelle facture"
│       │       ├─ ActionButton: "📸 Scanner"
│       │       └─ ActionButton: "📞 Contacter"
│       │
│       └─ SizedBox(height: 32) // Bottom padding
│
└─ FloatingActionButton: "+"
```

2. QuotesListPage Layout Tree

```
SafeArea
├─ Scaffold
│   ├─ AppBar: "Mes Devis"
│   │   └─ Actions: [Search, Filter]
│   │
│   └─ Body: Column(
│       ├─ SearchAndFilterBar
│       │   ├─ TextField: Search by client/number
│       │   ├─ Chip: "Tous"
│       │   ├─ Chip: "Brouillon"
│       │   ├─ Chip: "Envoyés"
│       │   ├─ Chip: "Acceptés"
│       │   └─ Chip: "Rejetés"
│       │
│       ├─ Divider
│       │
│       └─ Expanded(
│           └─ ListView.builder (quotes list)
│               └─ QuoteCard
│                   ├─ Row
│                   │   ├─ Expanded (content)
│                   │   │   ├─ Row (quote# + status)
│                   │   │   ├─ SizedBox(4)
│                   │   │   ├─ Text: Client name
│                   │   │   └─ SizedBox(8)
```

```
│              │    ├─ Row (price + date)
│              │    │    ├─ Text: "1,250€"
│              │    │    └─ Text: "15 jan"
│              │    │
│              │    └─ PopupMenuButton
│              │         ├─ View
│              │         ├─ Edit
│              │         ├─ Delete
│              │         ├─ Send (email)
│              │         └─ Download PDF
│              │
│              └─ Divider
│
└─ FloatingActionButton: "➕ Nouveau devis"
```

3. QuoteFormPage Layout Tree (Complex)

```
SafeArea
  ├─ Scaffold
  │    ├─ AppBar: "Nouveau Devis" or "Éditer Devis"
  │    │    └─ Actions: [Save, More options]
  │    │
  │    └─ Body: Form(
  │         └─ ListView(
  │              children: [
  │                // ===== SECTION 1: CLIENT SELECTION =====
  │                SectionHeader("Client"),
  │                SearchableDropdown
  │                    ├─ TextField with autocomplete
  │                    └─ List of existing clients + "Créer nouveau"
  │                SizedBox(16)
  │
  │                // ===== SECTION 2: DATES =====
  │                SectionHeader("Dates"),
  │                Row(
  │                    ├─ Expanded: DatePicker("Date devis")
```

```
│                └── Expanded: DatePicker("Valide jusqu'au")
│            )
│            SizedBox(16)
│
│            // ===== SECTION 3: LINE ITEMS =====
│            SectionHeader("Lignes"),
│            LineItemsBuilder(
│                ├── LineItem 1
│                │   ├── Autocomplete product search
│                │   ├── Qty field
│                │   ├── Price field
│                │   ├── Discount field
│                │   ├── Total (calculated)
│                │   └── Delete button
│                │
│                ├── LineItem 2
│                ├── ...
│                │
│                └── "+ Ajouter ligne" button
│            )
│            SizedBox(16)
│
│            // ===== SECTION 4: CALCULATIONS =====
│            Card(
│                ├── Row: "Total HT" | "1,000€"
│                ├── Row: "TVA (20%)" | "200€"
│                ├── Divider
│                ├── Row: "Total TTC" | "1,200€" (bold)
│                ├── Divider
│                └── Row: "Acompte (20%)" | "240€"
│            )
│            SizedBox(16)
│
│            // ===== SECTION 5: OPTIONS =====
│            SectionHeader("Options"),
│            TextField: "Notes"
│            SizedBox(8)
│            Row(
│                ├── Checkbox: "Nécessite signature"
│                └── Checkbox: "Envoyer après création"
│            )
│            SizedBox(16)
│
```

```
│            // ===== SECTION 6: ACTIONS =====
│            Row(
│                ├─ Expanded: OutlinedButton("Annuler")
│                ├─ SizedBox(8)
│                └─ Expanded: ElevatedButton("Enregistrer")
│            )
│            SizedBox(8)
│            Row(
│                ├─ Expanded: OutlinedButton("Télécharger PDF")
│                ├─ SizedBox(8)
│                └─ Expanded: ElevatedButton("Envoyer par email")
│            )
│            SizedBox(32) // Bottom padding
│        ]
│      )
│   )
```

4. ScanInvoicePage Layout Tree

```
SafeArea
  ├─ Scaffold
  │    ├─ AppBar: "Scanner facture"
  │    │
  │    └─ Body: SingleChildScrollView(
  │         └─ Column(
  │              ├─ // ===== IMAGE CAPTURE SECTION =====
  │            SectionHeader("Étape 1: Capturer image"),
  │            Card(
  │                ├─ Center(
  │                │    ├─ IF image == null:
  │                │    │    ├─ Icon: Camera
  │                │    │    ├─ Text: "Prendre une photo"
  │                │    │    └─ Row buttons:
  │                │    │         ├─ Button: "📷 Caméra"
  │                │    │         └─ Button: "📁 Galerie"
  │                │    │
```

```
│           │    └─ IF image != null:
│           │        ├─ Image preview (400x300)
│           │        ├─ Button: "Retirer image"
│           │        └─ Button: "Modifier image"
│           └─)
│       )
│       SizedBox(24)
│
│       ├─ // ===== OCR PROCESSING =====
│       Center(
│           └─ ElevatedButton(
│              label: "Scan avec OCR",
│              onPressed: _procesOcr,
│              enabled: image != null
│           )
│       )
│
│       ├─ IF loading:
│       │    ├─ LinearProgressIndicator
│       │    ├─ Text: "Traitement OCR... 35%"
│       │    └─ Text: "Extraction des données..."
│       │
│       └─ IF result != null:
│           ├─ SectionHeader("Étape 2: Résultats OCR")
│           │
│           ├─ Card(
│           │    ├─ Row: "Fournisseur" | TextField (editable)
│           │    ├─ Row: "Montant" | TextField(€) (editable)
│           │    ├─ Row: "Confiance" | ProgressBar (color-coded)
│           │    │    // Green > 85%, Yellow 65-85%, Red < 65%
│           │    ├─ Divider
│           │    │
│           │    └─ Text: "Lignes détectées:"
│           │        └─ ListView
│           │            ├─ ListTile
│           │            │    ├─ "Robinet mélangeur"
│           │            │    ├─ "Qté: 1"
│           │            │    └─ "60€"
│           │            ├─ ListTile: ...
│           │            └─ ...
│           │)
│           │
│           ├─ SectionHeader("Vérification")
```

```
│           │
│           ├─ Row(
│           │    ├─ Icon: Check (green)
│           │    ├─ Expanded: Text: "Les données semblent correctes?"
│           │    └─ Row: [Non] [Oui]
│           │)
│           │
│           └─ IF verified:
│              ├─ SectionHeader("Étape 3: Créer facture")
│              │
│              ├─ Row(
│              │    ├─ Expanded: OutlinedButton("Annuler")
│              │    ├─ SizedBox(8)
│              │    └─ Expanded: ElevatedButton("Créer facture fournisseur")
│              │)
│              │
│              └─ Text: "Une facture sera créée avec les données extraites"
│
│       └─ SizedBox(32)
│     )
│   )
```

(Includes ClientsListPage and InvoiceFormPage layouts as described in the original file...)

🎨 Component Styling Guide

Card Styling (Reusable)

Dart

```
class AppCard extends StatelessWidget {
  final Widget child;
  final EdgeInsets? padding;
```

```
  final double? elevation;
  final Function()? onTap;

  const AppCard({
    required this.child,
    this.padding,
    this.elevation = 1.0,
    this.onTap,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: elevation,
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
      child: InkWell(
        onTap: onTap,
        borderRadius: BorderRadius.circular(12),
        child: Padding(
          padding: padding ?? const EdgeInsets.all(16),
          child: child,
        ),
      ),
    );
  }
}
```

Section Header (Reusable)

Dart

```
class SectionHeader extends StatelessWidget {
  final String title;
  // ... (rest of the component code from file 1)
  const SectionHeader({
    required this.title,
```

```
    // ...
  });
  // ...
}
```

Empty State (Reusable)

Dart

```
class EmptyState extends StatelessWidget {
  final String title;
  // ... (rest of the component code from file 1)
  const EmptyState({
    required this.title,
    // ...
  });
  // ...
}
```

🎯 Interaction Patterns

Long Press Context Menu

Dart

```
onLongPress: () {
```

```
  showModalBottomSheet(
    context: context,
    builder: (context) => Column(
      mainAxisSize: MainAxisSize.min,
      children: [
        ListTile(
          leading: Icon(Icons.visibility),
          title: Text('Voir'),
          onTap: () => Navigator.pop(context),
        ),
        // ... (rest of the component code from file 1)
      ],
    ),
  );
}
```

Swipe to Delete

Dart

```
Dismissible(
  key: Key(quote.id),
  direction: DismissDirection.endToStart,
  // ... (rest of the component code from file 1)
  child: QuoteCard(quote: quote),
)
```

☁️ PART 2: SUPABASE-NATIVE BACKEND (Corrected & Cost-Effective)

*(This section **replaces** the original plombipro_part2_cloud_functions.md. All Python/GCP*

logic is replaced with TypeScript/Deno Supabase Edge Functions and free-tier APIs.)

Implementation Note

We are **discarding the Google Cloud Functions (Python) plan**. The backend logic will be implemented as **Supabase Edge Functions (TypeScript/Deno)**. This is more cost-effective (Supabase free tier includes 500,000 Edge Function invocations) and keeps your entire stack managed in one place.

All functions will be located in your supabase/functions/ directory.

☁️ Function 1: OCR Invoice Processing

- **Function Name:** ocr-process-invoice
- **Location:** supabase/functions/ocr-process-invoice/index.ts
- **Trigger:** HTTP POST
- **Cost-Effective API:** ocr.space (Free tier: 25,000 requests/month)
- **Logic:**
  1. Receives image_base64 from the Flutter app.
  2. Sends this data to the ocr.space API using fetch and your OCR_SPACE_API_KEY.
  3. Receives the JSON text response from ocr.space.
  4. Parses the raw text to find supplier, amount, and items (using Regex, similar to the original Python logic).
  5. Returns the structured JSON data to the Flutter app.
- **Example (index.ts):**

```TypeScript
import { serve } from 'https://deno.land/std@0.177.0/http/server.ts'

serve(async (req) => {
  const { image_base64 } = await req.json()
  const ocrSpaceApiKey = Deno.env.get('OCR_SPACE_API_KEY')

  // 1. Call ocr.space API
  const formData = new FormData()
  formData.append('base64Image', `${image_base64}`)
  formData.append('language', 'fre')
```

```
  const response = await fetch('https://api.ocr.space/parse/image', {
    method: 'POST',
    headers: { 'apikey': ocrSpaceApiKey },
    body: formData,
  })

  const ocrResult = await response.json()
  const rawText = ocrResult.ParsedResults[0].ParsedText

  // 2. Parse rawText (your regex logic here)
  const parsedData = _parseInvoiceData(rawText) // Implement this function

  return new Response(
    JSON.stringify({ success: true, result: parsedData }),
    { headers: { 'Content-Type': 'application/json' } },
  )
})

function _parseInvoiceData(text: string) {
  // ... (Implement regex parsing logic here) ...
  return { supplier: 'Fournisseur ABC', amount: 120.50, items: [] }
}
```

☁️ Function 2: Payment Intent (Stripe)

- **Function Name:** create-payment-intent
- **Location:** supabase/functions/create-payment-intent/index.ts
- **Trigger:** HTTP POST
- **Logic:** Uses the official Stripe Node.js library to create a Payment Intent.
- **Example (index.ts):**

```
TypeScript
import { serve } from 'https://deno.land/std@0.177.0/http/server.ts'
import Stripe from 'https://esm.sh/stripe@11.1.0'

const stripe = new Stripe(Deno.env.get('STRIPE_SECRET_KEY')!, {
  apiVersion: '2022-11-15',
  httpClient: Stripe.createFetchHttpClient(),
})
```

```
serve(async (req) => {
  const { amount, currency = 'eur', invoice_id } = await req.json()

  const intent = await stripe.paymentIntents.create({
    amount: amount, // Must be in cents
    currency: currency,
    metadata: { invoice_id },
  })

  return new Response(
    JSON.stringify({ success: true, client_secret: intent.client_secret }),
    { headers: { 'Content-Type': 'application/json' } },
  )
})
```

☁️ Function 3: Refund Payment

- **Function Name:** refund-payment
- **Location:** supabase/functions/refund-payment/index.ts
- **Trigger:** HTTP POST
- **Logic:** Uses the Stripe library to process a refund.

☁️ Function 4: Send Email Notification

- **Function Name:** send-email
- **Location:** supabase/functions/send-email/index.ts
- **Trigger:** HTTP POST
- **Cost-Effective API:** Resend (Free tier: 3,000 emails/month)
- **Logic:** A single function that takes a template type (quote_sent, invoice_sent, etc.) and context data, builds the HTML, and sends it via the Resend API.
- **Example (index.ts):**

```
TypeScript
import { serve } from 'https://deno.land/std@0.177.0/http/server.ts'

const RESEND_API_KEY = Deno.env.get('RESEND_API_KEY')
```

```
serve(async (req) => {
  const { to, subject, template, context } = await req.json()

  const html_content = _build_email_html(template, context) // Implement this

  await fetch('https://api.resend.com/emails', {
    method: 'POST',
    headers: {
      'Authorization': `Bearer ${RESEND_API_KEY}`,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      from: 'facturation@plombipro.fr',
      to: to,
      subject: subject,
      html: html_content,
    }),
  })

  return new Response(JSON.stringify({ success: true }), {
    headers: { 'Content-Type': 'application/json' },
  })
})

function _build_email_html(template: string, context: any) {
  if (template === 'quote_sent') {
    return `<h2>Devis ${context.quote_number}</h2><p>...`
  }
  // ... (other templates) ...
  return `<p>Email</p>`
}
```

☁️ Function 5: Generate Factur-X XML (2026 Compliance)

- **Function Name:** generate-factur-x
- **Location:** supabase/functions/generate-factur-x/index.ts
- **Trigger:** HTTP POST
- **Logic:** Uses a Deno-compatible XML builder (e.g., xml-builder-js) to construct the EN16931 standard XML, then saves it to Supabase Storage. The logic is a direct

translation from the original Python file.

☁️ Function 6: Scheduled Payment Reminders

- **Implementation: Supabase Cron Jobs (pg_cron)**
- **Cost-Effective API:** Free, built into Supabase.
- **Logic:**
  1. **Create a PostgreSQL Function:** In the Supabase SQL Editor, create a public.handle_payment_reminders function.
  2. This function SELECTs all invoices that are due soon and not paid.
  3. For each invoice, it uses http_post (or pg_net) to call your send-email Edge Function with the payment_reminder template.
  4. **Schedule the Job:** In the Supabase dashboard (under "Database" -> "Cron Jobs"), schedule this SQL function to run daily. 0 9 * * * (9 AM UTC every day).
- **Example (SQL Function):**

```
SQL
CREATE OR REPLACE FUNCTION public.handle_payment_reminders()
RETURNS void AS $$
DECLARE
  invoice_record RECORD;
  client_email TEXT;
  -- Add other needed context variables
BEGIN
  FOR invoice_record IN
    SELECT * FROM public.invoices
    WHERE due_date <= (now() + interval '3 days')
      AND payment_status != 'paid'
      -- Add logic to avoid re-sending every day
  LOOP
    -- Get client email
    SELECT email INTO client_email FROM public.clients WHERE id = invoice_record.client_id;

    -- Call the send-email Edge Function
    PERFORM net.http_post(
      url := 'https://YOUR_PROJECT.supabase.co/functions/v1/send-email',
      headers := '{"Content-Type": "application/json", "Authorization": "Bearer
YOUR_SUPABASE_SERVICE_KEY"}'::jsonb,
      body := json_build_object(
        'to', client_email,
        'subject', 'Rappel: Facture ' || invoice_record.invoice_number,
```

```
      'template', 'payment_reminder',
      'context', json_build_object(
        'invoice_number', invoice_record.invoice_number,
        'amount', invoice_record.total_ttc
        -- ... other context fields
      )
    )::jsonb
  );
 END LOOP;
END;
$$ LANGUAGE plpgsql;

-- Schedule it (Run this once in the SQL editor)
SELECT cron.schedule('daily-reminders', '0 9 * * *', 'SELECT
public.handle_payment_reminders()');
```

🎯 PART 3: CUSTOM FLUTTER FUNCTIONS, APIs & LIBRARY CALLS

*(This section is based on plombipro_part3_custom_functions.md but is **corrected** to call the new Supabase Edge Functions.)*

1. Invoice Calculation Service (lib/services/invoice_calculator.dart)

*(This file is pure Dart logic and is **unchanged**. It is correct as-is.)*

Dart

```
import 'package:intl/intl.dart';

class InvoiceCalculator {
```

```
  /// Calculate line item total: qty × price × (1 - discount%)
  static double calculateLineTotal({
    required double quantity,
    required double unitPrice,
    int discountPercent = 0,
  }) {
    final discountFactor = 1 - (discountPercent / 100);
    return (quantity * unitPrice * discountFactor * 100).round() / 100;
  }

  // ... (All other calculation methods: calculateSubtotal, calculateVAT, etc.)
}

class LineItem {
  // ... (Model class unchanged)
}
```

2. PDF Generation Service (lib/services/pdf_generator.dart)

*(This file is pure Dart logic for client-side PDF generation and is **unchanged**. It is correct as-is.)*

Dart

```
import 'package:pdf/pdf.dart';
import 'package:pdf/widgets.dart' as pw;
// ...

class PdfGenerator {

  /// Generate quote PDF
  static Future<Uint8List> generateQuotePdf({
    required String quoteNumber,
    // ... (all other parameters)
  }) async {
```

```
    final pdf = pw.Document();

    pdf.addPage(
      pw.MultiPage(
        // ... (PDF layout logic unchanged) ...
      ),
    );

    return pdf.save();
  }
}
```

3. Supabase Service (lib/services/supabase_service.dart)

*(This file is for database CRUD operations and is **unchanged**. It is correct as-is.)*

Dart

```
import 'package:supabase_flutter/supabase_flutter.dart';

class SupabaseService {
  static final SupabaseClient _client = Supabase.instance.client;

  // ===== AUTHENTICATION =====
  static Future<AuthResponse> signUp({
    // ... (Auth logic unchanged) ...
  }) async {
    // ...
  }

  // ===== QUOTES CRUD =====
  static Future<List<Quote>> fetchQuotes() async {
    try {
      // ... (Database logic unchanged) ...
    } catch (e) {
      rethrow;
```

```
  }
 }

 // ... (All other CRUD functions: createQuote, fetchClients, etc. are correct)
}

// ... (Model classes: Quote, Client, Product are correct) ...
```

4. OCR Processing Service (lib/services/ocr_service.dart)

*(This file is **CORRECTED** to call the ocr-process-invoice Edge Function.)*

Dart

```
import 'package:image_picker/image_picker.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'dart:convert';

class OcrService {
 static final _supabase = Supabase.instance.client;

 static Future<OcrResult?> scanInvoice(XFile imageFile) async {
  try {
   // Read image bytes
   final bytes = await imageFile.readAsBytes();
   final base64Image = base64Encode(bytes);

   // ---
   // ❗ CORRECTION: Call the Supabase Edge Function, not a GCP URL
   // ---
   final response = await _supabase.functions.invoke(
    'ocr-process-invoice', // This is the new, correct function name
    body: {'image_base64': base64Image},
   );

   if (response.data['success'] == true) {
```

```
      // Assuming your Edge Function returns a 'result' object
      return OcrResult.fromJson(response.data['result']);
    } else {
      throw Exception(response.data['error'] ?? 'OCR failed');
    }
  } catch (e) {
    rethrow;
  }
 }
 // ... (saveOcrScan method and OcrResult class are unchanged)
}

class OcrResult {
 // ... (Model class unchanged)
}
```

5. Stripe Payment Service (lib/services/stripe_service.dart)

*(This file is **CORRECTED** to call the create-payment-intent Edge Function.)*

Dart

```
import 'package:stripe_flutter/stripe_flutter.dart';
import 'package:supabase_flutter/supabase_flutter.dart';

class StripePaymentService {
 static final _supabase = Supabase.instance.client;

 // ... (initializeStripe is unchanged) ...

 static Future<PaymentResult> processPayment({
  required double amount,
  required String invoiceId,
  required String clientEmail,
  String currency = 'eur',
 }) async {
```

```
    try {
      // ---
      // ❗ CORRECTION: Call the Supabase Edge Function
      // ---
      final response = await _supabase.functions.invoke(
        'create-payment-intent', // This is the new, correct function name
        body: {
          'amount': (amount * 100).toInt(), // Convert to cents
          'currency': currency,
          'invoice_id': invoiceId,
        },
      );

      if (response.data['success'] != true) {
        throw Exception(response.data['error'] ?? 'Payment intent creation failed');
      }

      final clientSecret = response.data['client_secret'] as String;

      // ... (Rest of the Stripe.instance.initPaymentSheet logic is unchanged) ...

      await Stripe.instance.presentPaymentSheet();

      return PaymentResult(
        success: true,
        message: 'Paiement réussi',
        transactionId: response.data['payment_intent_id'], // Get ID if returned
      );
    } on StripeException catch (e) {
      // ... (Error handling unchanged) ...
    } catch (e) {
      // ... (Error handling unchanged) ...
    }
  }

  static Future<bool> refundPayment(String paymentId) async {
    try {
      // ---
      // ❗ CORRECTION: Call the Supabase Edge Function
      // ---
      final response = await _supabase.functions.invoke(
        'refund-payment', // This is the new, correct function name
        body: {'payment_id': paymentId},
```

```
    );

    return response.data['success'] == true;
  } catch (e) {
    print('Refund error: $e');
    return false;
  }
 }
}

class PaymentResult {
 // ... (Model class unchanged)
}
```

6. Email Service (lib/services/email_service.dart)

*(This file is **CORRECTED** to call the single send-email Edge Function.)*

Dart

```
import 'package:supabase_flutter/supabase_flutter.dart';
import 'dart:convert';

class EmailService {
 static final _supabase = Supabase.instance.client;

 // ---
 // ❗ CORRECTION: All methods now call the single 'send-email' function
 // ---

 static Future<bool> sendQuoteEmail({
  required String clientEmail,
  required String quoteNumber,
  required double amount,
  required DateTime validUntil,
  required String companyName,
```

```
  Uint8List? pdfBytes,
}) async {
  try {
    final pdfBase64 = pdfBytes != null ? base64Encode(pdfBytes) : null;

    final response = await _supabase.functions.invoke(
      'send-email', // This is the new, correct function name
      body: {
        'to': clientEmail,
        'subject': 'Devis $quoteNumber',
        'template': 'quote_sent', // Pass the template type
        'context': {
          'quote_number': quoteNumber,
          'amount': amount,
          'valid_until': validUntil.toString().split(' ')[0],
          'company_name': companyName,
        },
        'pdf_base64': pdfBase64,
        'filename': 'devis_$quoteNumber.pdf',
      },
    );

    return response.data['success'] == true;
  } catch (e) {
    print('Email error: $e');
    return false;
  }
}

static Future<bool> sendInvoiceEmail({
  // ... (parameters) ...
}) async {
  try {
    // ...
    final response = await _supabase.functions.invoke(
      'send-email', // This is the new, correct function name
      body: {
        'to': clientEmail,
        'subject': 'Facture $invoiceNumber',
        'template': 'invoice_sent',
        // ... (context and other fields) ...
      },
    );
```

```
    return response.data['success'] == true;
  } catch (e) {
    // ...
  }
 }

 static Future<bool> sendPaymentReminder({
  // ... (parameters) ...
 }) async {
  try {
   final response = await _supabase.functions.invoke(
    'send-email', // This is the new, correct function name
    body: {
     'to': clientEmail,
     'subject': 'Rappel: Facture $invoiceNumber en attente',
     'template': 'payment_reminder',
     // ... (context) ...
    },
   );
   return response.data['success'] == true;
  } catch (e) {
   // ...
  }
 }
}
```

7. Troubleshooting Guide

(This section is from plombipro_part3_custom_functions.md and remains relevant.)

Issue 1: Supabase Connection Timeout

Dart

```
// ✅ CORRECT: With timeout and retry
Future<List<Quote>> fetchQuotesWithRetry() async {
 int retries = 3;
```

```
  // ... (Retry logic unchanged) ...
}
```

Issue 2: PDF Generation Out of Memory

Dart

```
// ✅ CORRECT: Stream large PDFs
Future<File> generateQuotePdfToFile(Quote quote) async {
  // ... (Logic unchanged) ...
}
```

Issue 3: OCR Low Confidence

Dart

```
// Log OCR confidence for debugging
if (ocrResult.confidence < 0.65) {
  // ... (Logic unchanged) ...
}
```

(...and so on for all troubleshooting scenarios.)

🤖 PART 4: iOS/ANDROID DEPLOYMENT & AI FLUTTERFLOW PROMPTS

*(This section is from plombipro_part4_deployment_prompts.md and is **unchanged**. The iOS/Android deployment process and the UI-focused AI prompts are not affected by the backend technology change.)*

📱 Complete iOS Deployment Guide

Step 1: Prerequisites (15 minutes)

Bash

```
# Install/update Xcode command line tools
xcode-select --install
# Check Flutter doctor
flutter doctor -v
```

(...all iOS deployment steps are unchanged.)

🤖 Complete Android Deployment Guide

Step 1: Prerequisites

Bash

```
# Ensure Android SDK 28+ installed
flutter doctor --android
```

(...all Android deployment steps are unchanged.)

🤖 Exact FlutterFlow AI Prompts

*(These are for generating UI and are **unchanged**.)*

Prompt 1: HomePage/Dashboard

```
Create a professional Flutter dashboard page with Material Design 3:
LAYOUT:
- SafeArea with Scaffold
- AppBar (title: "PlombiPro", color: #1976D2)
... (prompt unchanged) ...
```

Prompt 2: QuotesListPage

```
Create a professional Quotes list page:
STRUCTURE:
- SafeArea with Scaffold
- AppBar: "Mes Devis" with search icon
... (prompt unchanged) ...
```

Prompt 3: QuoteFormPage (Complex Multi-Section Form)

Create advanced Quote form with Material Design 3:
STRUCTURE:
- Scaffold with AppBar: "Nouveau Devis" or "Éditer Devis"
... (prompt unchanged) ...

Prompt 4: ScanInvoicePage (OCR)

Create OCR Invoice scanning page:
STRUCTURE:
- SafeArea with Scaffold
- AppBar: "Scanner une facture"
... (prompt unchanged) ...

Prompt 5: Custom AppDrawer (Sidebar)

Create Material Design 3 sidebar menu/drawer:
HEADER (top):
- UserAccountsDrawerHeader with:
... (prompt unchanged) ...

(...all other content from file 4, including deployment checklists, remains valid.)

📋 PART 5: FINAL SUMMARY, ENV SETUP & MASTER CHECKLIST

*(This section is from plombipro_part5_final_summary.md and is **CORRECTED** to reflect the new Supabase-native stack.)*

🎯 Project Summary (Corrected)

Aspect	Details
App Name	PlombiPro
Target Market	French Plumbers (Plombiers)
Primary Tech	Flutter + Supabase
Backend	**Supabase Edge Functions (TypeScript) + PostgreSQL**
MVP Pages	20 core pages
Database	12 tables + 6 storage buckets
APIs	Supabase, Stripe, **ocr.space**, **Resend**
Deployment	iOS App Store + Google Play
Development Time	10 weeks (1 dev + 1 AI)
Monthly Cost	**$0 (on free tiers) - scales with usage**

🔐 Environment Variables Setup (Corrected)

1. Flutter App (.env file)

Create lib/.env (add to .gitignore):

Code snippet

```
# SUPABASE
SUPABASE_URL=https://YOUR_PROJECT.supabase.co
SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

# STRIPE
STRIPE_PUBLISHABLE_KEY=pk_live_YOUR_KEY

# APP CONFIG
APP_NAME=PlombiPro
APP_VERSION=1.0.0
ENVIRONMENT=production
```

*(**Note:** Secret keys like STRIPE_SECRET_KEY should NOT be in your Flutter app. They live only in your Supabase Edge Function environment variables.)*

2. Supabase Edge Function Environment Variables

Set these in your Supabase Project Dashboard under Settings -> Edge Functions or in your supabase/.env file for local development.

Code snippet

```
# Supabase CLI will use this for local dev
SUPABASE_URL=https://YOUR_PROJECT.supabase.co
SUPABASE_ANON_KEY=YOUR_ANON_KEY
SUPABASE_SERVICE_ROLE_KEY=YOUR_SERVICE_ROLE_KEY
```

```
# API Keys for Functions
STRIPE_SECRET_KEY=sk_live_YOUR_KEY
RESEND_API_KEY=re_YOUR_KEY
OCR_SPACE_API_KEY=YOUR_KEY
```

3. Load in Dart (main.dart)

Dart

```dart
// main.dart
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await dotenv.load(fileName: 'lib/.env');

  await Supabase.initialize(
    url: dotenv.env['SUPABASE_URL'] ?? '',
    anonKey: dotenv.env['SUPABASE_ANON_KEY'] ?? '',
  );

  Stripe.publishableKey = dotenv.env['STRIPE_PUBLISHABLE_KEY'] ?? '';

  runApp(const MyApp());
}
```

📋 Complete Deployment Master Checklist (Corrected)

PHASE 1: Local Development Setup (Week 1-2)

Environment:

- [ ] Flutter SDK 3.22.0+ installed
- [ ] Xcode 14.3+ installed
- [ ] Android SDK 28+ installed
- [ ] Git repository initialized
- [ ] **Supabase CLI installed**
- [ ] **supabase init and supabase link run**

Database:

- [ ] Supabase project created
- [ ] All 12 tables created with SQL (from supabase_schema.sql)
- [ ] Row Level Security (RLS) policies applied
- [ ] 6 storage buckets created (avatars, logos, documents, etc.)
- [ ] Bucket policies configured (from supabase_bucket_policies.sql)
- [ ] Database backups enabled

Backend Services (Corrected):

- [ ] **supabase/functions directory created**
- [ ] **All 6 Edge Functions created (TS/Deno)**
- [ ] **Resend** account with API key (for emails)
- [ ] **ocr.space** account with API key (for OCR)
- [ ] Stripe account (test mode initially)
- [ ] All environment variables set in Supabase Dashboard
- [ ] **pg_cron extension enabled in Supabase**

Flutter Project:

- [ ] pubspec.yaml dependencies all added
- [ ] Supabase Flutter package configured in main.dart
- [ ] Router configured with all 20 pages
- [ ] Theme Material 3 configured
- [ ] Localization (intl) for French

PHASE 2: Core Features Development (Week 3-6)

(This phase is frontend-heavy and remains unchanged.)

- [ ] All pages (HomePage, QuotesListPage, etc.) created
- [ ] All custom components (QuoteCard, StatusBadge, etc.) created
- [ ] All Dart services (InvoiceCalculator, PdfGenerator, etc.) implemented

PHASE 3: Advanced Features (Week 7-8) (Corrected)

OCR Implementation:

- [ ] **ocr-process-invoice Edge Function** tested
- [ ] OcrService in Flutter successfully calls the function
- [ ] ScanInvoicePage fully functional
- [ ] Confidence scoring working
- [ ] Manual override UI implemented
- [ ] OCR tested on 20+ real invoices

Payments Integration:

- [ ] Stripe test mode validated
- [ ] **create-payment-intent Edge Function** tested
- [ ] PaymentSheet Flutter integration complete
- [ ] **refund-payment Edge Function** tested
- [ ] Test payment flow: €0.50 charge + refund

Electronic Invoicing (2026):

- [ ] **generate-factur-x Edge Function** complete
- [ ] Factur-X XML validation (EN16931 standard)
- [ ] XML storage in Supabase verified

PHASE 4: Notifications & Scheduling (Week 9) (Corrected)

Email Notifications:

- [ ] **send-email Edge Function** tested
- [ ] Quote sent, Invoice sent, and Reminder templates working
- [ ] PDF attachment in emails working (via base64)
- [ ] Resend domain authentication configured

Scheduled Tasks:

- [ ] **pg_cron job** created for payment reminders
- [ ] **PostgreSQL function handle_payment_reminders** deployed and tested
- [ ] Payment reminder emails sent on schedule
- [ ] Overdue invoice detection logic in SQL working

Push Notifications:

- [ ] Firebase Cloud Messaging configured (remains a valid choice)
- [ ] Push notification permission request UI
- [ ] Test notification sent and received

PHASE 5: Quality Assurance (Week 10 - Day 1-3)

(This phase remains unchanged and is critical.)

- [ ] Functional Testing (Create quote -> invoice -> email, etc.)
- [ ] Performance Testing (App launch < 2s, etc.)
- [ ] Security Testing (RLS policies, input validation)
- [ ] Responsive Design Testing (Mobile, Tablet)
- [ ] Error Handling (Network errors, timeouts, etc.)

PHASE 6: iOS Release (Week 10 - Day 4-5)

(This phase remains unchanged.)

- [ ] Xcode Setup (Signing, provisioning)
- [ ] App Store Preparation (Listing, screenshots, privacy policy)
- [ ] TestFlight build and testing
- [ ] Submission

PHASE 7: Android Release (Week 10 - Day 6-7)

(This phase remains unchanged.)

- [ ] Play Store Preparation (Signing, listing, content rating)
- [ ] Build appbundle
- [ ] Internal/Beta testing
- [ ] Move to Production

🚀 QUICK START COMMAND SEQUENCE (Corrected)

Bash

```
# 1. Clone and setup
git clone YOUR_REPO
cd plombipro
flutter pub get

# 2. Set up Supabase
supabase init
supabase login
supabase link --project-ref YOUR_PROJECT_REF
supabase db push # Push your local schema
supabase functions deploy # Deploy all Edge Functions

# 3. Set environment
# (Ensure .env file is populated)
export SUPABASE_URL=https://YOUR_PROJECT.supabase.co
export SUPABASE_ANON_KEY=YOUR_KEY

# 4. Run dev
flutter run

# 5. Build iOS
cd ios && pod install && cd ..
flutter build ios --release

# 6. Build Android
flutter build appbundle --release

# 7. Submit to app stores
# iOS: Open ios/Runner.xcworkspace → Archive → Distribute
# Android: Upload build/app/outputs/bundle/release/app-release.aab to Play Store
```

💡 NEXT FEATURES (Version 2.0+)

(This section remains unchanged.)

1. Multi-user teams with role-based access
2. Advanced scheduling & resource planning
3. Full offline mode with real-time sync
   ... (etc.)

✅ FINAL SIGN-OFF CHECKLIST (Corrected)

Before going live:

- [ ] All 20 pages functional
- [ ] All calculations verified
- [ ] All integrations tested (**Stripe**, **ocr.space**, **Resend**)
- [ ] **All 6 Supabase Edge Functions tested and deployed**
- [ ] **pg_cron job running correctly**
- [ ] Performance meets targets
- [ ] Security audit passed (RLS!)
- [ ] iOS submitted to App Store
- [ ] Android submitted to Play Store
- [ ] Support infrastructure ready
- [ ] Monitoring dashboards active
- [ ] Backup strategy in place
- [ ] Legal (privacy policy, TOS) reviewed
- [ ] Launch date set & communicated

🎉 YOU'RE READY FOR LAUNCH!

Summary (Corrected):

- ✅ 20 core pages built
- ✅ 12 database tables with RLS
- ✅ **6 Supabase Edge Functions deployed**
- ✅ **Supabase Cron Job for reminders active**
- ✅ Stripe payments integrated

- ✅ OCR scanning working (via ocr.space)
- ✅ Electronic invoicing 2026-compliant
- ✅ Email notifications active (via Resend)
- ✅ iOS & Android builds ready
- ✅ Production infrastructure ready (and on the free tier)

LET'S GO LIVE! 🚀