FLYER MANAGEMENT SYSTEM UPDATE: PDF Splitting, Cataloging, and Contact Stamping

OVERVIEW

We need to add three capabilities to the Marketing Materials system:

1. **PDF Splitter** — Accept a multi-page PDF upload and split it into individual single-page flyer PDFs
2. **Flyer Catalog Manager** — Categorize each flyer by industry/service type, detect duplicates, and replace old versions
3. **Contact Info Stamper** — When an agent downloads or shares a flyer, overlay their contact information onto the existing PDF using a "stamp and cover" method (draw a filled rectangle over the old contact section, then draw the new agent's info on top)

This does NOT regenerate flyers with AI. These are pre-made, professionally designed PDFs. We are surgically replacing the contact information block only. The AI flyer generator remains a separate feature for custom flyers.

CRITICAL TECHNICAL DECISIONS

Use `pdf-lib` (NOT Python, NOT pypdf)

Since this app is Node.js/TypeScript on Replit, use the `pdf-lib` npm package for all PDF operations. It can:

- Load existing PDFs
- Split pages into individual documents
- Draw filled rectangles (to cover old contact info)
- Draw text with specific fonts, sizes, colors, and positioning
- Embed images (for logos/badges if needed)
- Save modified PDFs

```
npm install pdf-lib
```

DO NOT use:

- Python libraries (this is a Node.js app)
- Puppeteer/Playwright (overkill for text overlay)
- Canvas/image-based approaches (loses PDF quality)
- The AI flyer generator pipeline (that's a separate feature)

PART 1: PDF SPLITTER

Feature: Upload Multi-Page PDF → Split into Individual Flyers

Location: Add to the Marketing Materials admin/management interface

Flow:

1. Admin uploads a multi-page PDF (e.g., `2026_Updated_Flyers.pdf` with 17 pages)
2. System splits it into 17 individual single-page PDFs
3. Each page is presented for categorization (industry, flyer type, contact layout type)
4. Admin confirms categorization, system stores each flyer as a template

Implementation:

```
// server/services/pdfSplitter.ts
import { PDFDocument } from 'pdf-lib';
import fs from 'fs';
import path from 'path';

interface SplitResult {
  pageNumber: number;
  filePath: string;
  fileName: string;
}

export async function splitPdfIntoPages(inputPath: string, outputDir: string): Pr
  const existingPdfBytes = fs.readFileSync(inputPath);
  const pdfDoc = await PDFDocument.load(existingPdfBytes);
  const pageCount = pdfDoc.getPageCount();
```

```
  const results: SplitResult[] = [];

  // Ensure output directory exists
  if (!fs.existsSync(outputDir)) {
    fs.mkdirSync(outputDir, { recursive: true });
  }

  for (let i = 0; i < pageCount; i++) {
    const newDoc = await PDFDocument.create();
    const [copiedPage] = await newDoc.copyPages(pdfDoc, [i]);
    newDoc.addPage(copiedPage);

    const fileName = `flyer_page_${i + 1}.pdf`;
    const filePath = path.join(outputDir, fileName);
    const pdfBytes = await newDoc.save();
    fs.writeFileSync(filePath, pdfBytes);

    results.push({
      pageNumber: i + 1,
      filePath,
      fileName,
    });
  }

  return results;
}
```

API Endpoint:

```
// POST /api/marketing/split-pdf
// Accepts multipart form upload of a PDF
// Returns array of split page info for categorization
```

PART 2: FLYER CATALOG MANAGER

Database Schema Addition

Add a table (or extend the existing marketing templates table) to store flyer metadata:

```
// In shared/schema.ts (add or extend)
export const flyerTemplates = pgTable("flyer_templates", {
  id: serial("id").primaryKey(),
  name: text("name").notNull(),                    // e.g., "Dual Pricing - Resta
```

```
  industry: text("industry").notNull(),             // e.g., "restaurants", "auto
  flyerType: text("flyer_type").notNull(),          // e.g., "dual_pricing", "cas
  contactLayoutType: text("contact_layout_type").notNull(), // e.g., "standard_bo
  pdfPath: text("pdf_path").notNull(),              // path to the master templat
  thumbnailPath: text("thumbnail_path"),            // path to PNG thumbnail for
  isActive: boolean("is_active").default(true),
  version: integer("version").default(1),
  createdAt: timestamp("created_at").defaultNow(),
  updatedAt: timestamp("updated_at").defaultNow(),
});
```

Industry Categories

Based on the current flyer set, these are the categories. This list should be extensible:

```
export const FLYER_INDUSTRIES = [
  { value: "general",           label: "General / All Industries" },
  { value: "restaurants_bars",  label: "Restaurants & Bars" },
  { value: "pizza",             label: "Pizza / Italian" },
  { value: "auto_sales",       label: "Auto Sales" },
  { value: "auto_service",     label: "Auto Service / Tire Shops" },
  { value: "liquor_stores",    label: "Liquor Stores" },
  { value: "retail",           label: "Retail / Shops" },
  { value: "healthcare",       label: "Healthcare / Medical" },
  { value: "beauty_salon",     label: "Beauty / Hair Salon" },
  { value: "garden_nursery",   label: "Garden Center / Nursery" },
  { value: "construction",     label: "Construction / Mining" },
  { value: "professional",     label: "Professional Services" },
  { value: "craft_artisan",    label: "Craft / Artisan" },
  { value: "convenience",      label: "Convenience Store" },
] as const;

export const FLYER_TYPES = [
  { value: "dual_pricing",     label: "Dual Pricing Program" },
  { value: "cash_advance",     label: "Merchant Cash Advance" },
  { value: "who_is_pcbancard", label: "Who is PCBancard?" },
] as const;
```

Duplicate Detection

When uploading new flyers, check for existing templates with the same `industry` + `flyerType` combination:

```
async function checkForDuplicate(industry: string, flyerType: string): Promise<Fl
  const existing = await db.select()
```

```
    .from(flyerTemplates)
    .where(
      and(
        eq(flyerTemplates.industry, industry),
        eq(flyerTemplates.flyerType, flyerType),
        eq(flyerTemplates.isActive, true)
      )
    )
    .limit(1);

  return existing[0] || null;
}

// When a duplicate is found:
// 1. Deactivate the old version (set isActive = false)
// 2. Increment the version number
// 3. Save the new flyer as the active version
```

Thumbnail Generation

After splitting, generate a PNG thumbnail of each page for the UI grid. Use `pdf2pic` or `sharp` + `pdf-poppler`:

```
npm install pdf-poppler  # or pdf2pic


import { convert } from 'pdf-poppler';

async function generateThumbnail(pdfPath: string, outputDir: string): Promise<str
  const opts = {
    format: 'png',
    out_dir: outputDir,
    out_prefix: path.basename(pdfPath, '.pdf'),
    page: 1,
    scale: 1024,  // width in pixels
  };

  const result = await convert(pdfPath, opts);
  return result[0]; // path to the generated PNG
}
```

PART 3: CONTACT INFO STAMPER (THE CRITICAL PIECE)

How It Works

When an agent requests a personalized flyer:

1. Load the master template PDF
2. Look up the `contactLayoutType` to get the stamp coordinates and styling
3. Draw a filled rectangle over the existing contact info (matching the background)
4. Draw the new agent's name, title, phone, and email on top
5. Return the personalized PDF for download

Contact Layout Types

Based on analysis of the current flyer set, there are **4 distinct contact section layouts**. Each needs its own coordinate mapping.

IMPORTANT: These coordinates are approximate starting points. You MUST verify and fine-tune them by:

1. Loading a sample flyer
2. Drawing a semi-transparent colored rectangle at the specified coordinates
3. Checking if it covers the contact section properly
4. Adjusting as needed

The coordinate system in pdf-lib is **bottom-left origin** (0,0 is bottom-left of the page). Letter size is 612 x 792 points.

```
// server/services/contactLayoutMaps.ts

export interface ContactRegion {
  // Rectangle to cover old contact info
  coverRect: {
    x: number;
    y: number;
    width: number;
    height: number;
    color: { r: number; g: number; b: number }; // RGB 0-1 scale
  };
  // Where to draw new contact text
  textPositions: {
    name:  { x: number; y: number; size: number; bold: boolean; color: { r: numbe
    title: { x: number; y: number; size: number; bold: boolean; color: { r: numbe
    phone: { x: number; y: number; size: number; bold: boolean; color: { r: numbe
```

```
    email: { x: number; y: number; size: number; bold: boolean; color: { r: numbe
  };
}

export const CONTACT_LAYOUTS: Record<string, ContactRegion> = {

  // ============================================================================
  // LAYOUT A: "standard_bottom" — Used by most Dual Pricing flyers
  // ============================================================================
  // The contact section is in the bottom-left area, next to BBB/Inc 5000 badges
  // Background is white. Text is black with bold name.
  // Examples: Pages 3, 5-12, 14-15 of the uploaded PDF
  // Agent info appears near the BBB badge and Inc 5000 logo
  //
  // APPROXIMATE COORDINATES (verify and adjust!):
  "standard_bottom": {
    coverRect: {
      x: 225,        // starts after the fee table area
      y: 18,         // near the very bottom
      width: 185,    // wide enough to cover name, title, phone, email
      height: 65,    // tall enough for 4 lines of text
      color: { r: 1, g: 1, b: 1 }, // white background
    },
    textPositions: {
      name:  { x: 230, y: 68, size: 11, bold: true,  color: { r: 0.1, g: 0.1, b:
      title: { x: 230, y: 55, size: 8.5, bold: false, color: { r: 0.3, g: 0.3, b:
      phone: { x: 230, y: 43, size: 8.5, bold: false, color: { r: 0.3, g: 0.3, b:
      email: { x: 230, y: 31, size: 8.5, bold: false, color: { r: 0.3, g: 0.3, b:
    },
  },

  // ============================================================================
  // LAYOUT B: "purple_footer" — Used by Cash Advance flyer
  // ============================================================================
  // Full-width purple/indigo footer bar at the bottom
  // White text on purple background
  // Example: Page 2 of the uploaded PDF
  //
  // APPROXIMATE COORDINATES (verify and adjust!):
  "purple_footer": {
    coverRect: {
      x: 370,        // right side of the footer bar
      y: 0,          // bottom edge
      width: 242,    // covers the right portion of footer
      height: 65,    // height of the purple footer
      color: { r: 0.388, g: 0.4, b: 0.945 }, // purple (#6366F1) matching the bar
    },
```

```
  textPositions: {
    name:  { x: 380, y: 42, size: 12, bold: true,  color: { r: 1, g: 1, b: 1 }
    title: { x: 380, y: 30, size: 9,  bold: false, color: { r: 0.9, g: 0.9, b:
    phone: { x: 380, y: 18, size: 9,  bold: false, color: { r: 0.9, g: 0.9, b:
    email: { x: 380, y: 6,  size: 9,  bold: false, color: { r: 0.9, g: 0.9, b:
  },
},

// ========================================================================
// LAYOUT C: "yellow_auto" — Used by Auto Sales & Service Dual Pricing
// ========================================================================
// Contact section on a yellow/gold background (#F59E0B) at the bottom
// Dark text. Has QR code and Dejavoo branding nearby.
// Example: Page 4 of the uploaded PDF
//
// APPROXIMATE COORDINATES (verify and adjust!):
"yellow_auto": {
  coverRect: {
    x: 30,         // left side
    y: 10,         // near bottom
    width: 250,    // covers the contact block
    height: 65,    // 4 lines of text
    color: { r: 0.961, g: 0.620, b: 0.043 }, // yellow/gold matching the sectio
  },
  textPositions: {
    name:  { x: 40, y: 60, size: 11, bold: true,  color: { r: 0.15, g: 0.15, b:
    title: { x: 40, y: 47, size: 8.5, bold: false, color: { r: 0.25, g: 0.25, b
    phone: { x: 40, y: 35, size: 8.5, bold: false, color: { r: 0.25, g: 0.25, b
    email: { x: 40, y: 23, size: 8.5, bold: false, color: { r: 0.25, g: 0.25, b
  },
},

// ========================================================================
// LAYOUT D: "who_is_pcbancard_bottom" — Used by "Who is PCBancard?" flyer
// ========================================================================
// Contact section at bottom-right with PCBancard logo
// White background. Includes logo placeholder + name/title/phone/email
// Example: Page 1 of the uploaded PDF
//
// APPROXIMATE COORDINATES (verify and adjust!):
"who_is_pcbancard_bottom": {
  coverRect: {
    x: 310,        // right portion of the page
    y: 60,         // above the footer logos
    width: 250,    // covers the contact block
    height: 80,    // taller because includes logo spacing
    color: { r: 1, g: 1, b: 1 }, // white background
```

```
    },
    textPositions: {
      name:  { x: 350, y: 120, size: 12, bold: true,  color: { r: 0.1, g: 0.1, b:
      title: { x: 350, y: 106, size: 9,  bold: false, color: { r: 0.3, g: 0.3, b:
      phone: { x: 350, y: 93,  size: 9,  bold: false, color: { r: 0.3, g: 0.3, b:
      email: { x: 350, y: 80,  size: 9,  bold: false, color: { r: 0.3, g: 0.3, b:
    },
  },
};
```

The Stamper Service

```
// server/services/contactStamper.ts
import { PDFDocument, rgb, StandardFonts } from 'pdf-lib';
import fs from 'fs';
import { CONTACT_LAYOUTS, ContactRegion } from './contactLayoutMaps';

export interface AgentContactInfo {
  name: string;          // e.g., "William Wise"
  title: string;         // e.g., "Local Payments Expert"
  phone: string;         // e.g., "(317) 555-1234"
  email: string;         // e.g., "william@pcbancard.com"
}

export async function stampContactInfo(
  templatePdfPath: string,
  contactLayoutType: string,
  agentInfo: AgentContactInfo
): Promise<Buffer> {
  // 1. Load the template PDF
  const existingPdfBytes = fs.readFileSync(templatePdfPath);
  const pdfDoc = await PDFDocument.load(existingPdfBytes);

  // 2. Get the layout mapping
  const layout = CONTACT_LAYOUTS[contactLayoutType];
  if (!layout) {
    throw new Error(`Unknown contact layout type: ${contactLayoutType}`);
  }

  // 3. Get the first (only) page
  const page = pdfDoc.getPages()[0];

  // 4. Embed fonts
  const helvetica = await pdfDoc.embedFont(StandardFonts.Helvetica);
  const helveticaBold = await pdfDoc.embedFont(StandardFonts.HelveticaBold);
```

```
// 5. Draw cover rectangle over existing contact info
const { coverRect } = layout;
page.drawRectangle({
  x: coverRect.x,
  y: coverRect.y,
  width: coverRect.width,
  height: coverRect.height,
  color: rgb(coverRect.color.r, coverRect.color.g, coverRect.color.b),
});

// 6. Draw new contact info
const { textPositions } = layout;

// Name
page.drawText(agentInfo.name, {
  x: textPositions.name.x,
  y: textPositions.name.y,
  size: textPositions.name.size,
  font: textPositions.name.bold ? helveticaBold : helvetica,
  color: rgb(
    textPositions.name.color.r,
    textPositions.name.color.g,
    textPositions.name.color.b
  ),
});

// Title
page.drawText(agentInfo.title, {
  x: textPositions.title.x,
  y: textPositions.title.y,
  size: textPositions.title.size,
  font: textPositions.title.bold ? helveticaBold : helvetica,
  color: rgb(
    textPositions.title.color.r,
    textPositions.title.color.g,
    textPositions.title.color.b
  ),
});

// Phone
page.drawText(agentInfo.phone, {
  x: textPositions.phone.x,
  y: textPositions.phone.y,
  size: textPositions.phone.size,
  font: textPositions.phone.bold ? helveticaBold : helvetica,
  color: rgb(
    textPositions.phone.color.r,
```

```
      textPositions.phone.color.g,
      textPositions.phone.color.b
    ),
  });

  // Email
  page.drawText(agentInfo.email, {
    x: textPositions.email.x,
    y: textPositions.email.y,
    size: textPositions.email.size,
    font: textPositions.email.bold ? helveticaBold : helvetica,
    color: rgb(
      textPositions.email.color.r,
      textPositions.email.color.g,
      textPositions.email.color.b
    ),
  });

  // 7. Save and return as buffer
  const pdfBytes = await pdfDoc.save();
  return Buffer.from(pdfBytes);
}
```

COORDINATE CALIBRATION PROCESS

THIS IS THE MOST IMPORTANT STEP. The coordinates above are ESTIMATES. You MUST calibrate them.

Build a one-time calibration helper:

```
// server/services/calibrateContactRegion.ts
// Run this to visually verify where the cover rectangle lands

import { PDFDocument, rgb } from 'pdf-lib';
import fs from 'fs';

export async function generateCalibrationPdf(
  templatePath: string,
  rectX: number,
  rectY: number,
  rectWidth: number,
  rectHeight: number,
  outputPath: string
): Promise<void> {
  const bytes = fs.readFileSync(templatePath);
  const doc = await PDFDocument.load(bytes);
```

```
  const page = doc.getPages()[0];

  // Draw a SEMI-TRANSPARENT RED rectangle so you can see what it covers
  page.drawRectangle({
    x: rectX,
    y: rectY,
    width: rectWidth,
    height: rectHeight,
    color: rgb(1, 0, 0),
    opacity: 0.3,   // 30% opacity so you can see through it
  });

  // Also draw crosshairs at the text positions for verification
  // (Add more as needed for each text field)

  const calibratedBytes = await doc.save();
  fs.writeFileSync(outputPath, calibratedBytes);
  console.log(`Calibration PDF saved to: ${outputPath}`);
}
```

Calibration workflow:

1. Run the calibration function on each layout type with a sample flyer
2. Open the output PDF and check if the red rectangle covers EXACTLY the contact info area
3. Adjust coordinates and re-run until perfect
4. Lock in the final coordinates in the layout map

Add a simple API endpoint or script so the admin can run calibration:

```
GET /api/marketing/calibrate/:templateId?x=225&y=18&w=185&h=65
→ Returns a PDF with a red overlay at those coordinates for visual verification
```

PART 4: API ENDPOINTS

New Endpoints to Add

```
// ============================================
// 1. Split a multi-page PDF into individual flyers
// ============================================
// POST /api/marketing/split-pdf
```

```
// Body: multipart form with PDF file
// Response: { pages: [{ pageNumber, tempPath, thumbnailUrl }] }

// ============================================
// 2. Save a split page as a cataloged flyer template
// ============================================
// POST /api/marketing/templates
// Body: {
//   name: "Dual Pricing - Restaurant",
//   industry: "restaurants_bars",
//   flyerType: "dual_pricing",
//   contactLayoutType: "standard_bottom",
//   tempPagePath: "/tmp/splits/flyer_page_8.pdf"
// }
// Response: { template: { id, name, industry, ... } }
// NOTE: Check for duplicates. If industry+flyerType already exists,
//       deactivate old, increment version, save new.

// ============================================
// 3. Download a personalized flyer (THE MONEY ENDPOINT)
// ============================================
// POST /api/marketing/templates/:templateId/personalize
// Body: {
//   agentName: "William Wise",
//   agentTitle: "Local Payments Expert",
//   agentPhone: "(317) 555-1234",
//   agentEmail: "william@pcbancard.com"
// }
// Response: PDF file download with stamped contact info

// ============================================
// 4. Calibration helper (admin only)
// ============================================
// POST /api/marketing/calibrate
// Body: {
//   templateId: 5,
//   rectX: 225, rectY: 18, rectWidth: 185, rectHeight: 65
// }
// Response: PDF with semi-transparent red overlay for verification

// ============================================
// 5. List all active flyer templates
// ============================================
// GET /api/marketing/templates
// Query params: ?industry=restaurants_bars&flyerType=dual_pricing
// Response: { templates: [...] }
```

PART 5: FRONTEND INTEGRATION

Marketing Materials Page Updates

The Marketing Materials page currently shows static templates and an AI generator. Add:

1. Admin Panel (for you / team leads):

- "Upload Flyer Pack" button → opens file picker for multi-page PDF
- After upload, shows a grid of split pages with:
  - Thumbnail preview
  - Dropdown for Industry
  - Dropdown for Flyer Type
  - Dropdown for Contact Layout Type (with visual examples)
  - "Save" / "Replace Existing" button per page
- Calibration tool (enter coordinates, preview overlay)

2. Agent Download Flow (for sales reps):

- Agent browses flyer templates by industry
- Clicks a flyer → sees preview
- Clicks "Download with My Info" → system checks if agent's contact info is stored in their profile
  - If yes: automatically stamps and downloads
  - If no: prompts for name, title, phone, email → stamps and downloads
- Option to "Download Original" (no contact stamp) if needed

3. Contact Info Storage:

- Store the agent's contact info in their user profile so they don't re-enter it every time
- Fields: name, title, phone, email
- Editable in Settings or on first download

UI Component Sketch

```
// In the template grid, each card shows:
<TemplateCard>
  <Thumbnail src={template.thumbnailUrl} />
  <Badge>{template.industry}</Badge>
  <Badge variant="outline">{template.flyerType}</Badge>
  <CardFooter>
    <Button onClick={() => downloadPersonalized(template.id)}>
      📥 Download with My Info
    </Button>
  </CardFooter>
</TemplateCard>
```

PART 6: RAG INTEGRATION

Feeding the AI Flyer Generator

The pre-made flyers also serve as the knowledge base for the AI custom flyer generator. When new templates are added:

1. **Store the PDF in the RAG-accessible directory** (wherever the existing flyer templates are stored — likely `public/marketing/` or similar)
2. **Update any flyer template manifest/index** that the RAG system reads from
3. **Generate and store text metadata** for each flyer:
    - Industry
    - Key selling points / copy
    - Layout description
    - Color scheme
    - This helps the LLM understand the flyer's style when generating custom ones

```
// When saving a new template, also update the RAG index:
async function updateRagIndex(template: FlyerTemplate): Promise<void> {
  // Extract text content from the PDF for RAG indexing
  // Store alongside the template record
  // The AI generator can then query: "Show me flyer styles for restaurants"
  // and get back the relevant templates as reference material
}
```

PART 7: MAPPING THE CURRENT UPLOADED FLYERS

Here is how to categorize the 17 pages from the uploaded `2026_Updated_Flyers.pdf`:

Page	Flyer Type	Industry	Contact Layout	Current Agent
1	who_is_pcbancard	general	who_is_pcbancard_bottom	Pamela Kutoroff
2	cash_advance	general	purple_footer	Monte Barrow
3	dual_pricing	general	standard_bottom	Monte Barrow
4	dual_pricing	auto_sales	yellow_auto	Monte Barrow
5	dual_pricing	retail	standard_bottom	Monte Barrow
6	dual_pricing	auto_service	standard_bottom	Monte Barrow
7	dual_pricing	construction	standard_bottom	Monte Barrow
8	dual_pricing	pizza	standard_bottom	Monte Barrow
9	dual_pricing	restaurants_bars	standard_bottom	Monte Barrow
10	dual_pricing	healthcare	standard_bottom	Monte Barrow
11	dual_pricing	beauty_salon	standard_bottom	Monte Barrow
12	dual_pricing	garden_nursery	standard_bottom	Monte Barrow
13	dual_pricing	professional	standard_bottom	Pamela Kutoroff

14	dual_pricing	craft_artisan	standard_bottom	Monte Barrow
15	dual_pricing	convenience	standard_bottom	Monte Barrow
16	dual_pricing	auto_service	standard_bottom	Steve Sheriff
17	dual_pricing	retail	standard_bottom	Steve Sheriff

Duplicate detection notes:

- Pages 5 and 17 are both `dual_pricing` + `retail` → keep the newer one (page 17 if it's updated, or whichever has better imagery)
- Pages 6 and 16 are both `dual_pricing` + `auto_service` → same logic
- If the existing system already has these industries, the new uploads should REPLACE the old versions

---

PART 8: IMPLEMENTATION ORDER

Do these in this exact order:

Phase 1: Foundation

1. Install `pdf-lib`
2. Create the `contactLayoutMaps.ts` with the 4 layout definitions
3. Create the `contactStamper.ts` service
4. Create the calibration helper
5. **TEST:** Load one of the existing flyers, stamp test contact info, verify the output PDF looks right

Phase 2: Splitting & Cataloging

6. Create the `pdfSplitter.ts` service
7. Add the database table for flyer templates
8. Create API endpoints for splitting and saving templates

9. Add thumbnail generation
10. **TEST:** Upload the 17-page PDF, verify it splits correctly, save each page with correct metadata

Phase 3: Agent Download Flow

11. Add the personalization API endpoint
12. Update the frontend to show "Download with My Info" on each template
13. Add agent contact info storage to user profiles
14. **TEST:** Log in as a test agent, download a personalized flyer, verify contact info is correct

Phase 4: RAG Integration

15. Ensure new templates are stored where the RAG system can access them
16. Update any flyer template manifest/index
17. **TEST:** Use the AI generator to create a custom flyer and verify it references the new templates

---

IMPORTANT NOTES

- **DO NOT BREAK the existing AI flyer generator.** This is an ADDITION, not a replacement.
- **DO NOT BREAK the existing static template downloads.** Those should continue to work as-is.
- The `contactLayoutType` on each template is what links it to the correct stamp coordinates. Get this wrong and you'll stamp contact info on top of the QR code or the fee table.
- The coordinates WILL need calibration. Budget time for this. It's a visual process — generate, check, adjust, repeat.
- Use Helvetica and Helvetica-Bold as the fonts (these are PDF standard fonts and don't need embedding). The original flyers appear to use a similar sans-serif font.
- The title field for most agents is "Local Payments Expert" but some have variations like "Local Payments Expert and Trusted Business Advisor" — make sure the text field is long enough.

- If an agent’s title is longer (like Pamela’s “Local Payments Expert and Trusted Business Advisor”), either reduce the font size dynamically or wrap to two lines.