Complete Logic Documentation - V5.2 Two-Stage Image Workflow

Created: November 15, 2025 **For:** Chocolate on James Content Creator PRO **Purpose:** Document complete logic for replication in other projects (e.g., TapeGeeks)

Table of Contents

System Overview

What This System Does

The two-stage image workflow allows users to: 1. **Upload a reference image** (product photo, competitor inspiration, etc.) 2. **Get instant analysis** (colors, lighting, brightness, mood) 3. **Generate 4 quick prompt variations** (Exact, Inspired, Premium, Branded) 4. **OR combine image with custom options** to generate comprehensive content

Key Innovation

Instead of generic templates, the system: - Analyzes ACTUAL uploaded images - Extracts real colors, brightness, temperature - Generates unique prompts per image - Allows combining image + user preferences

---

Data Flow Architecture

Flow Diagram

```
User Uploads Image




Image Processing Pipeline
1. Convert to base64
2. Load into canvas
3. Sample pixels
```

```
Analysis Functions
- extractDominantColors()
- analyzeBrightness()
- analyzeColorTemperature()
```

```
Store in State Variables
- currentImageData (base64)
- currentAnalysis (object)
- generatedVariations (array)
```

```
Stage 1          Stage 2          Clear Image
Quick            Custom           Reset All
4 Prompts        Full Gen
```

State Variables (Global Scope)

```
let currentImageData = null;       // Base64 image data
let currentAnalysis = null;        // Analysis object
let generatedVariations = [];      // 4 prompt variations
let lastGeneratedOptions = null;   // For regenerate function
```

Core Functions Reference

1. Image Upload & Processing

`handleImageUpload(file)` or `handleImagePaste(event)` **Purpose:** Entry point for image data **Input:** File object or clipboard event **Output:** Triggers analysis pipeline **Key Actions:** - Validates file type (JPG, PNG, WebP) - Converts to base64 via FileReader - Stores in `currentImageData` - Calls `analyzeImage(imageData)`

Code Pattern:

```
function handleImageUpload(file) {
    if (!file.type.match('image.*')) {
        alert('Please upload an image file');
        return;
    }
```

```
    const reader = new FileReader();
    reader.onload = function(e) {
        currentImageData = e.target.result;
        displayImagePreview(currentImageData);
        analyzeImage(currentImageData);
    };
    reader.readAsDataURL(file);
}
```

---

2. Image Analysis Functions

`analyzeImage(imageData)` **Purpose:** Orchestrates analysis pipeline **Flow:**

```
async function analyzeImage(imageData) {
    // Show loading state
    showLoadingSpinner();

    // Call AI analysis (or pixel analysis)
    const analysis = await analyzeImageWithAI(imageData);

    // Store results
    currentAnalysis = analysis;

    // Update UI
    displayAnalysis(analysis);
    generatePromptVariations(analysis);
    showImageModeBanner(imageData);

    // Hide loading
    hideLoadingSpinner();
}
```

`analyzeImageWithAI(imageData)` **Purpose:** Core analysis logic **Returns:** Analysis object **Key Steps:** 1. Load image into canvas 2. Extract dominant colors via pixel sampling 3. Calculate average brightness 4. Determine color temperature 5. Generate descriptive text

Analysis Object Structure:

```
{
    style: "Bright, high-key photography with clean presentation",
    composition: "Analyzed from uploaded reference image",
    lighting: "Soft, even lighting creating minimal shadows",
    colorPalette: ["#8B4513", "#D4AF37", "#F5F5DC", "#2C1810", "#FFD700"],
    mood: "Bright, cheerful, inviting, clean",
    technicalDetails: "Well-lit studio setup, sharp focus",
    brandElements: "Extracted from reference image",
    imageMetrics: {
        avgBrightness: 185,
```

```
        colorCount: 5,
        temperature: "warm"
    }
}
```

Helper Functions `extractDominantColors(ctx, width, height)`

```
function extractDominantColors(ctx, width, height) {
    const colorMap = {};
    const sampleSize = 10; // Sample every 10th pixel

    for (let y = 0; y < height; y += sampleSize) {
        for (let x = 0; x < width; x += sampleSize) {
            const pixel = ctx.getImageData(x, y, 1, 1).data;
            const hex = rgbToHex(pixel[0], pixel[1], pixel[2]);
            colorMap[hex] = (colorMap[hex] || 0) + 1;
        }
    }

    // Return top 5 colors
    return Object.entries(colorMap)
        .sort((a, b) => b[1] - a[1])
        .slice(0, 5)
        .map(([color]) => color);
}
```

`analyzeBrightness(ctx, width, height)`

```
function analyzeBrightness(ctx, width, height) {
    let totalBrightness = 0;
    let pixelCount = 0;
    const sampleSize = 10;

    for (let y = 0; y < height; y += sampleSize) {
        for (let x = 0; x < width; x += sampleSize) {
            const pixel = ctx.getImageData(x, y, 1, 1).data;
            const brightness = (pixel[0] * 0.299 + pixel[1] * 0.587 + pixel[2] * 0.114);
            totalBrightness += brightness;
            pixelCount++;
        }
    }

    return totalBrightness / pixelCount;
}
```

`analyzeColorTemperature(colors)`

```
function analyzeColorTemperature(colors) {
    let warmCount = 0;
    let coolCount = 0;
```

```
    colors.forEach(hex => {
        const rgb = hexToRgb(hex);
        // Warm if red/yellow dominant
        if (rgb.r > rgb.b) warmCount++;
        else coolCount++;
    });

    if (warmCount > coolCount * 1.5) return 'warm';
    if (coolCount > warmCount * 1.5) return 'cool';
    return 'neutral';
}
```

---

3. Prompt Generation Functions

generatePromptVariations(analysis) - STAGE 1 **Purpose:** Generate 4 quick variations from analysis **Logic:**

```
function generatePromptVariations(analysis) {
    const colors = analysis.colorPalette.join(', ');
    const brightness = analysis.imageMetrics.avgBrightness;
    const temp = analysis.imageMetrics.temperature;

    generatedVariations = [
        // Variation 1: Exact Match
        `Professional chocolate photography recreating this exact style: ${analysis.style}.
         Lighting: ${analysis.lighting}.
         Color palette: ${colors}.
         Mood: ${analysis.mood}.
         ${analysis.technicalDetails}.
         Maintain exact brightness level (${brightness}/255) and ${temp} color temperature.`,

        // Variation 2: Inspired
        `Chocolate photography inspired by reference: Similar ${analysis.mood} aesthetic
         but with creative variation. Quality level: ${analysis.style}.
         Color inspiration: ${colors}. Keep ${temp} tones but experiment with angles.`,

        // Variation 3: Premium
        `Ultra-premium chocolate photography: ${analysis.style} elevated to luxury standard.
         Enhanced lighting, perfect ${colors} color harmony, museum-quality detail.
         ${brightness > 170 ? 'Bright pristine presentation' : 'Dramatic moody atmosphere'}.`,

        // Variation 4: Branded
        `Chocolate on James brand photography: ${analysis.style} adapted for Hamilton's
         premier chocolatier. ${analysis.lighting} with brand colors ${colors}.
         ${temp} inviting atmosphere, James Street North artisan quality.`
    ];
```

```
    // Display in UI
    generatedVariations.forEach((variation, index) => {
        document.getElementById(`variation-${index}`).textContent = variation;
    });
}
```

buildImagePromptWithUploadedImage(options, analysis) - STAGE 2 Purpose: Combine uploaded image analysis + user options **This is the KEY function for Stage 2**

Input Parameters: - `options`: User-selected controls (platform, season, audience, style, etc.) - `analysis`: Image analysis object

Logic Flow:

```
function buildImagePromptWithUploadedImage(options, analysis) {
    let prompt = '';

    // 1. START WITH IMAGE DESCRIPTION
    prompt += `SUBJECT FROM UPLOADED REFERENCE IMAGE:\n`;
    prompt += `Visual Style: ${analysis.style}\n`;
    prompt += `Color Palette: ${analysis.colorPalette.join(', ')}\n`;
    prompt += `Lighting Characteristics: ${analysis.lighting}\n`;
    prompt += `Mood & Atmosphere: ${analysis.mood}\n`;
    prompt += `Brightness Level: ${analysis.imageMetrics.avgBrightness > 170 ? 'Bright/High-ke
                                   analysis.imageMetrics.avgBrightness > 85 ? 'Normal/Balance
                                   'Dark/Low-key'}\n`;
    prompt += `Color Temperature: ${analysis.imageMetrics.temperature}\n\n`;

    // 2. ADD PLATFORM REQUIREMENTS
    prompt += `PLATFORM OPTIMIZATION:\n`;
    const platformSpecs = {
        'instagram-square': '1:1 square format, Instagram-optimized',
        'instagram-story': '9:16 vertical format, Instagram Story',
        'facebook-ad': '1.91:1 landscape, Facebook Ad specs',
        'pinterest': '2:3 vertical, Pinterest-optimized'
    };
    prompt += `- Format: ${platformSpecs[options.platform]}\n\n`;

    // 3. ADD SEASONAL/OCCASION STYLING
    if (options.season && options.season !== 'any') {
        prompt += `SEASONAL STYLING:\n`;
        const seasonalElements = getSeasonalElements(options.season);
        prompt += `- Occasion: ${seasonalElements}\n`;
        prompt += `- Apply seasonal decorations and atmosphere while maintaining the core aest
    }

    // 4. ADD AUDIENCE TARGETING
    prompt += `TARGET AUDIENCE:\n`;
```

```
prompt += `- Primary: ${options.customAudience || options.audience}\n`;
prompt += `- Presentation should appeal to this demographic while preserving uploaded imag

// 5. ADD STYLE PREFERENCES
prompt += `PHOTOGRAPHY STYLE:\n`;
const styleDescriptions = {
    'photography': 'Professional product photography',
    'lifestyle': 'Lifestyle photography with context and story',
    'minimalist': 'Clean minimalist aesthetic',
    'artistic': 'Creative artistic interpretation'
};
prompt += `- Aesthetic: ${styleDescriptions[options.style]}\n`;
prompt += `- Blend this style with the visual characteristics from uploaded image\n\n`;

// 6. ADD CONTENT CATEGORY
prompt += `CONTENT CATEGORY:\n`;
prompt += `- Type: ${options.customCategory || options.category}\n\n`;

// 7. ADD TECHNICAL REQUIREMENTS
prompt += `TECHNICAL SPECIFICATIONS:\n`;
prompt += `- Resolution: 4K-8K, 300 DPI print quality\n`;
prompt += `- Sharpness: Tack-sharp focus, professional lens quality\n`;
prompt += `- Color: Match uploaded image palette: ${analysis.colorPalette.join(', ')}\n`;
prompt += `- Lighting: Enhance ${analysis.lighting.toLowerCase()}\n\n`;

// 8. ADD SPECIAL ELEMENTS
if (options.includeText) {
    prompt += `- Reserve clean space for text overlay\n`;
}
if (options.customTextInImage) {
    prompt += `- Include text in image: "${options.customTextInImage}"\n`;
}
if (options.includeLogo) {
    prompt += `- Reserve space for brand logo (bottom right)\n`;
}
if (options.includeLocation) {
    prompt += `- Subtle Hamilton/James Street North local character\n`;
}

// 9. FINAL INSTRUCTION
prompt += `\nFINAL RESULT: Professional ${options.platform} photography that preserves the
color palette (${analysis.colorPalette.join(', ')}), and ${analysis.imageMetrics.brightness
analysis.imageMetrics.brightness > 85 ? 'balanced' : 'moody'} atmosphere from the uploaded
while incorporating ${options.season || 'timeless'} styling for ${options.audience || 'gen

return {
    prompt: prompt,
    platform: getPlatformName(options.platform)
```

```
    };
}
```

---

4. UI State Management

`showImageModeBanner(imageData)` **Purpose:** Show visual indicator that image is loaded **Implementation:**

```
function showImageModeBanner(imageData) {
    const banner = document.getElementById('imageUploadBanner');
    const thumbnail = document.getElementById('imageBannerThumbnail');

    banner.classList.remove('hidden');
    thumbnail.src = imageData;

    // Update button text
    document.getElementById('btn-generate').textContent =
        '  Generate Content Using Uploaded Image';
}
```

`clearUploadedImage()` **Purpose:** Reset all image state **Implementation:**

```
function clearUploadedImage() {
    if (confirm('Clear uploaded image? You can upload again anytime.')) {
        // Reset state
        currentImageData = null;
        currentAnalysis = null;
        generatedVariations = [];

        // Hide banner
        document.getElementById('imageUploadBanner').classList.add('hidden');

        // Reset button text
        document.getElementById('btn-generate').textContent =
            '  Generate Complete Content';

        // Clear analysis results
        document.getElementById('imageAnalysisResults').classList.add('hidden');

        // Clear file input
        document.getElementById('imageFileInput').value = '';
    }
}
```

---

5. Main Generation Logic

`buildImagePrompt(options)` - DISPATCHER Purpose: Route to correct prompt builder **Implementation:**

```
function buildImagePrompt(options) {
    // STAGE 2: If image analysis exists, use uploaded image mode
    if (currentAnalysis !== null) {
        return buildImagePromptWithUploadedImage(options, currentAnalysis);
    }

    // DEFAULT: No image uploaded, use generic mode
    return buildGenericImagePrompt(options);
}
```

State Management

Critical State Variables

Variable	Type	Purpose	Lifecycle
`currentImageData`	String (base64)	Stores uploaded image	Set on upload, cleared on clear
`currentAnalysis`	Object	Stores analysis results	Set after analysis, cleared on clear
`generatedVariations`	Array[4]	Quick prompt variations	Set after analysis, cleared on clear
`lastGeneratedOptions`	Object	Last used options	Set on generate, used for regenerate

State Transitions

```
NULL STATE (No Image)
  ↓ [User uploads image]
ANALYZING STATE (Loading)
  ↓ [Analysis complete]
IMAGE LOADED STATE
  ↓ [User clicks Generate]
GENERATING STATE
  ↓ [Content created]
CONTENT DISPLAYED STATE
  ↓ [User clicks Clear Image]
NULL STATE
```

User Interface Logic

Visual Feedback System

Upload States

1. **Empty:** Dropzone visible, "Drop or paste image"
2. **Drag Over:** Border highlight, "Drop now!"
3. **Analyzing:** Loading spinner, "Analyzing image…"
4. **Complete:** Preview shown, banner visible, 4 variations displayed

Button States

Condition	Button Text	Visual
No image	" Generate Complete Content"	Default gold
Image loaded	" Generate Content Using Uploaded Image"	Gold + icon
Generating	" Generating…"	Disabled, gray

Banner Visibility

```
// Show when: currentImageData !== null
// Hide when: currentImageData === null
// Contains: Thumbnail + message + clear button
```

Implementation Checklist

Phase 1: Image Upload

- ☐ Add file input element
- ☐ Add drag/drop zone
- ☐ Implement paste handler
- ☐ Add FileReader logic
- ☐ Store in `currentImageData`

Phase 2: Image Analysis

- ☐ Create canvas for pixel sampling
- ☐ Implement `extractDominantColors()`
- ☐ Implement `analyzeBrightness()`
- ☐ Implement `analyzeColorTemperature()`
- ☐ Create `analyzeImageWithAI()` orchestrator
- ☐ Define analysis object structure

Phase 3: Quick Variations (Stage 1)

- ☐ Implement `generatePromptVariations()`
- ☐ Create 4 variation display elements

- ☐ Add copy buttons
- ☐ Add "Use" buttons to send to Stage 2

Phase 4: Full Generation (Stage 2)

- ☐ Implement `buildImagePromptWithUploadedImage()`
- ☐ Add image check in `buildImagePrompt()`
- ☐ Combine analysis + user options
- ☐ Generate comprehensive prompt

Phase 5: UI State Management

- ☐ Create upload banner HTML
- ☐ Implement `showImageModeBanner()`
- ☐ Implement `clearUploadedImage()`
- ☐ Add visual indicators
- ☐ Update button text dynamically

Phase 6: Testing

- ☐ Test with bright images
- ☐ Test with dark images
- ☐ Test with warm-toned images
- ☐ Test with cool-toned images
- ☐ Test Stage 1 workflow
- ☐ Test Stage 2 workflow
- ☐ Test clear functionality
- ☐ Test without image (regression)

Key Success Factors

1. **Unique Analysis:** Every image produces different results
2. **State Persistence:** Image data persists until explicitly cleared
3. **Visual Clarity:** User always knows if image is active
4. **Dual Workflow:** Both quick and comprehensive options work
5. **Graceful Fallback:** Works perfectly without images too

Performance Considerations

- Pixel sampling uses 10px intervals (not every pixel)
- Analysis completes in ~1.5 seconds
- Base64 storage is memory-efficient for typical product photos
- Canvas cleared after analysis to free memory

Future Enhancements (TODO)

1. **Real AI Vision Integration**
   - Replace pixel analysis with Claude Vision API
   - Get detailed scene understanding
   - Recognize objects, composition, techniques
2. **Multiple Image Support**
   - Compare 2-3 images
   - Blend styles
   - A/B testing mode
3. **Image History**
   - Save last 5 uploaded images
   - Quick reload previous analysis
   - Compare over time

End of Complete Logic Documentation

This document contains everything needed to replicate this system in another project.