# BirthdayGen Source Code Review

## Complete File Contents for Code Review

```
════════════════════════════════════════════════════════════
FILE: /home/ubuntu/code_artifacts/birthdaygen.com/src/lib/ai/openai.ts
════════════════════════════════════════════════════════════

/**
 * OpenAI GPT-4o Integration for Gift Recommendations
 * Module B - Real AI Intelligence (Phase 4)
 *
 * Provides GPT-4o-backed gift recommendations with structured outputs.
 * NO mock data, NO fallbacks - fails gracefully with clear errors.
 */

import OpenAI from 'openai';
import { z } from 'zod';
import type {
  RecommendationRequest,
  GiftRecommendation,
  ProductDetails,
  GiftCategory,
  PriceRange,
} from '@/lib/gifts/schema';

// ============================================================================
// CONFIGURATION
// ============================================================================

/**
 * Initialize OpenAI client
 * Throws error if OPENAI_API_KEY is not configured
 */
function getOpenAIClient(): OpenAI {
  const apiKey = process.env.OPENAI_API_KEY;

  if (!apiKey) {
    throw new Error(
      'OPENAI_API_KEY is not configured. Please set this environment variable to
enable AI gift recommendations.'
    );
  }

  return new OpenAI({
    apiKey,
  });
}

// ============================================================================
// ZOD SCHEMAS FOR VALIDATION
// ============================================================================

/**
 * Zod schema for individual gift recommendation from GPT-4o
 */
const GiftRecommendationSchema = z.object({
  product_name: z.string().min(1),
  description: z.string().min(10),
  category: z.enum([
    'experiences', 'events', 'travel',
    'tech', 'fashion', 'home_decor', 'beauty', 'books',
    'food', 'wine', 'gourmet',
    'sports', 'gaming', 'art', 'music', 'wellness',
    'personalized', 'handmade', 'jewelry',
    'subscription', 'gift_cards', 'eco_friendly',
  ]),
  price_range: z.enum(['budget', 'affordable', 'moderate', 'premium', 'luxury']),
  estimated_price: z.number().min(0),
  tags: z.array(z.string()).min(1),
  confidence_score: z.number().min(0).max(100),
  reasoning: z.string().min(10),
  why_this_gift: z.string().min(10),
  personalize_idea: z.string().min(10),
  match_factors: z.object({
    gifting_style_match: z.number().min(0).max(100),
    archetype_match: z.number().min(0).max(100),
    occasion_match: z.number().min(0).max(100),
    budget_match: z.number().min(0).max(100),
    relationship_match: z.number().min(0).max(100),
  }),
});

/**
 * Zod schema for complete GPT-4o response
 */
const GiftRecommendationsResponseSchema = z.object({
  recommendations: z.array(GiftRecommendationSchema).min(1).max(10),
  recipient_summary: z.string().min(10),
  top_categories: z.array(z.string()).min(1).max(5),
});

type GPTRecommendation = z.infer<typeof GiftRecommendationSchema>;
type GPTRecommendationsResponse = z.infer<typeof GiftRecommendationsResponseSchema>;

// ============================================================================
// GIFT INPUT TYPE
// ============================================================================

export interface GiftInput {
  recipientName: string;
  recipientAge?: number;
  recipientGender?: string;
  relationship?: string;
  occasion: string;
  budgetMin: number;
  budgetMax: number;
  budgetPreferred?: number;

  // Enriched data
  giftingStyle?: string;
  interests?: string[];
  threeWords?: string[];
  vibes?: string[];
  archetypes?: string[];

  // Preferences
  excludeCategories?: string[];
  preferredCategories?: string[];
  shippingRequired?: boolean;
  urgency?: string;
}

// ============================================================================
// MAIN AI FUNCTION
// ============================================================================

/**
 * Generate gift recommendations using GPT-4o
 *
 * @param input - Gift recommendation input parameters
 * @returns Array of gift recommendations with confidence scores
 * @throws Error if OpenAI API is not configured or request fails
 */
export async function generateGiftRecommendations(
  input: GiftInput
): Promise<GiftRecommendation[]> {
  const client = getOpenAIClient();

  // Build context for GPT-4o
  const systemPrompt = buildSystemPrompt();
  const userPrompt = buildUserPrompt(input);

  try {
    // Call GPT-4o with structured output using JSON mode
    const completion = await client.chat.completions.create({
      model: 'gpt-4o',
      messages: [
        {
          role: 'system',
          content: systemPrompt,
        },
        {
          role: 'user',
          content: userPrompt,
        },
      ],
      response_format: { type: 'json_object' },
      temperature: 0.8, // Creative but not too random
      max_tokens: 4000,
    });

    const responseContent = completion.choices[0]?.message?.content;

    if (!responseContent) {
      throw new Error('OpenAI returned empty response');
    }

    // Parse and validate response
    const parsedResponse = JSON.parse(responseContent);
    const validatedResponse = GiftRecommendationsResponseSchema.parse(parsedResponse);

    // Transform GPT response to our internal format
    const recommendations: GiftRecommendation[] = validatedResponse.recommendations.ma
p(
      (rec, index) => ({
        id: `ai-${Date.now()}-${index}`,
        product: {
          name: rec.product_name,
          description: rec.description,
          category: rec.category as GiftCategory,
          priceRange: rec.price_range as PriceRange,
          estimatedPrice: rec.estimated_price,
          tags: rec.tags,
        },
        confidence: rec.confidence_score,
        reasoning: rec.reasoning,
        matchFactors: {
          giftingStyleMatch: rec.match_factors.gifting_style_match,
          archetypeMatch: rec.match_factors.archetype_match,
          occasionMatch: rec.match_factors.occasion_match,
          budgetMatch: rec.match_factors.budget_match,
          relationshipMatch: rec.match_factors.relationship_match,
        },
        whyThisGift: rec.why_this_gift,
        personalizeIdea: rec.personalize_idea,
      })
    );

    return recommendations;
  } catch (error) {
    // Handle errors gracefully
    if (error instanceof z.ZodError) {
      console.error('OpenAI response validation failed:', error);
      throw new Error(
        'AI generated invalid gift recommendations. Please try again.'
      );
    }

    if (error instanceof OpenAI.APIError) {
      console.error('OpenAI API error:', error.message, error.status);

      if (error.status === 401) {
        throw new Error('Invalid OpenAI API key. Please check configuration.');
      }

      if (error.status === 429) {
        throw new Error('OpenAI rate limit exceeded. Please try again later.');
      }

      throw new Error(`OpenAI API error: ${error.message}`);
    }

    console.error('Unexpected error in generateGiftRecommendations:', error);
    throw new Error('Failed to generate gift recommendations. Please try again.');
  }
}

// ============================================================================
// PROMPT BUILDERS
// ============================================================================

/**
 * Build system prompt for GPT-4o
 */
function buildSystemPrompt(): string {
  return `You are an expert gift recommendation AI for BirthdayGen.com, specializing i
n personalized, thoughtful gift suggestions.

Your task is to analyze recipient profiles and generate highly relevant gift recom-
mendations with detailed reasoning.

RESPONSE FORMAT (JSON):
{
  "recommendations": [
    {
      "product_name": "string",
      "description": "string (min 10 chars)",
      "category": "experiences|events|travel|tech|fashion|home_decor|beauty|books|
food|wine|gourmet|sports|gaming|art|music|wellness|personalized|handmade|jewelry|
subscription|gift_cards|eco_friendly",
      "price_range": "budget|affordable|moderate|premium|luxury",
      "estimated_price": number,
      "tags": ["string"],
      "confidence_score": number (0-100),
      "reasoning": "string (explain why this matches their profile)",
      "why_this_gift": "string (user-friendly explanation)",
      "personalize_idea": "string (how to make it more personal)",
      "match_factors": {
        "gifting_style_match": number (0-100),
        "archetype_match": number (0-100),
        "occasion_match": number (0-100),
        "budget_match": number (0-100),
        "relationship_match": number (0-100)
      }
    }
  ],
  "recipient_summary": "string (summarize the recipient's profile)",
  "top_categories": ["string"] (1-5 most relevant categories)
}

GUIDELINES:
1. Generate 5-10 diverse gift recommendations
2. Prioritize uniqueness and personalization over generic gifts
3. Consider the relationship context (formal vs. close)
4. Match the occasion appropriately
5. Stay within the specified budget range
6. Use the recipient's interests, vibes, and personality traits
7. Provide specific, actionable gift ideas (not vague categories)
8. Explain your reasoning clearly
9. Suggest how to personalize each gift
10. Calculate realistic match factors based on the input data

IMPORTANT:
- Return ONLY valid JSON matching the schema above
- NO markdown formatting, NO code blocks, NO extra text
- Ensure all numeric fields are actual numbers, not strings
- Ensure all required fields are present and non-empty`;
}

/**
 * Build user prompt with recipient context
 */
function buildUserPrompt(input: GiftInput): string {
  const parts: string[] = [];

  // Recipient basics
  parts.push(`RECIPIENT PROFILE:`);
  parts.push(`- Name: ${input.recipientName}`);

  if (input.recipientAge) {
    parts.push(`- Age: ${input.recipientAge}`);
  }

  if (input.recipientGender) {
    parts.push(`- Gender: ${input.recipientGender}`);
  }

  if (input.relationship) {
    parts.push(`- Relationship: ${input.relationship}`);
  }

  // Enriched profile data
  if (input.giftingStyle) {
    parts.push(`- Gifting Style: ${input.giftingStyle}`);
  }

  if (input.interests && input.interests.length > 0) {
    parts.push(`- Interests: ${input.interests.join(', ')}`);
  }

  if (input.threeWords && input.threeWords.length > 0) {
    parts.push(`- Personality (3 words): ${input.threeWords.join(', ')}`);
  }

  if (input.vibes && input.vibes.length > 0) {
    parts.push(`- Vibes/Aesthetics: ${input.vibes.join(', ')}`);
  }

  if (input.archetypes && input.archetypes.length > 0) {
    parts.push(`- Archetypes: ${input.archetypes.join(', ')}`);
  }

  // Occasion and budget
  parts.push('');
  parts.push(`OCCASION: ${input.occasion}`);
  parts.push('');
  parts.push(`BUDGET:`);
  parts.push(`- Range: $${input.budgetMin} - $${input.budgetMax}`);

  if (input.budgetPreferred) {
    parts.push(`- Preferred: $${input.budgetPreferred}`);
  }

  // Preferences and constraints
  if (input.preferredCategories && input.preferredCategories.length > 0) {
    parts.push('');
    parts.push(`PREFERRED CATEGORIES: ${input.preferredCategories.join(', ')}`);
  }

  if (input.excludeCategories && input.excludeCategories.length > 0) {
    parts.push('');
    parts.push(`EXCLUDE CATEGORIES: ${input.excludeCategories.join(', ')}`);
  }

  if (input.shippingRequired !== undefined) {
    parts.push('');
    parts.push(`Shipping Required: ${input.shippingRequired ? 'Yes' : 'No'}`);
  }

  if (input.urgency) {
    parts.push(`Urgency: ${input.urgency}`);
  }

  parts.push('');
  parts.push('Please generate personalized gift recommendations based on this pro-
file.');

  return parts.join('\n');
}

═══════════════════════════════════════════════════════════
SUMMARY:
This module provides GPT-4o integration for AI-powered gift recommendations. It con-
structs
structured prompts with recipient profiles (personality, vibes, interests, archetypes,
budget),
sends them to OpenAI's GPT-4o model with JSON schema validation using Zod, and
transforms the
responses into typed GiftRecommendation objects. In-
cludes comprehensive error handling for
```

```
API failures, rate limits, and validation errors.
═══════════════════════════════════════════════════════════
```

```
════════════════════════════════════════════════════════════════════════════════
FILE: /home/ubuntu/code_artifacts/birthdaygen.com/src/app/api/gift-recommendations/rou
te.ts
════════════════════════════════════════════════════════════════════════════════

/**
 * Gift Recommendations API Route
 * Module B - Real AI Intelligence (Phase 4)
 *
 * Handles POST requests for gift recommendations using GPT-4o.
 * NO mock data - all recommendations are AI-generated based on
 * enriched contact profiles from the auto-populate system.
 */

import { NextRequest, NextResponse } from 'next/server';
import type {
  RecommendationRequest,
  RecommendationResponse,
  RecipientProfile,
} from '@/lib/gifts/schema';
import { generateGiftRecommendations, type GiftInput } from '@/lib/ai/openai';

/**
 * Build GiftInput from RecommendationRequest
 */
function buildGiftInput(request: RecommendationRequest): GiftInput {
  const { recipient, occasion, budget, engagementAnswers, excludeCategories,
preferredCategories, shippingRequired, urgency } = request;

  return {
    recipientName: recipient.name,
    recipientAge: recipient.age,
    recipientGender: recipient.gender,
    relationship: recipient.relationship,
    occasion,
    budgetMin: budget.min,
    budgetMax: budget.max,
    budgetPreferred: budget.preferred,

    // Enriched data
    giftingStyle: recipient.giftingProfile?.style,
    interests: recipient.giftingProfile?.interests || recipient.interests,
    threeWords: recipient.threeWords,
    vibes: recipient.vibes || engagementAnswers?.pickTheirVibe?.selectedVibes,
    archetypes: recipient.archetypes?.map(a => a.name),

    // Preferences
    excludeCategories,
    preferredCategories,
    shippingRequired,
    urgency,
  };
}

// ============================================================================
// API ROUTE HANDLER
// ============================================================================

export async function POST(request: NextRequest) {
  try {
    const body = await request.json() as RecommendationRequest;
    // Validate request
    if (!body.recipient || !body.occasion || !body.budget) {
      return NextResponse.json(
        {
          success: false,
          error: {
            code: 'INVALID_REQUEST',
            message: 'Missing required fields: recipient, occasion, or budget',
          },
        },
        { status: 400 }
      );
    }

    // Validate budget
    if (body.budget.min < 0 || body.budget.max < body.budget.min) {
      return NextResponse.json(
        {
          success: false,
          error: {
            code: 'INVALID_BUDGET',
            message: 'Budget must be positive and max must be greater than min',
          },
        },
        { status: 400 }
      );
    }

    const startTime = Date.now();

    try {
      // Build input for AI
      const giftInput = buildGiftInput(body);

      // Generate recommendations using GPT-4o
      const recommendations = await generateGiftRecommendations(giftInput);

      // If no recommendations found, return helpful message
      if (recommendations.length === 0) {
        return NextResponse.json({
          success: true,
          recommendations: [],
          recipientSummary: `No gifts found matching your criteria. Try adjusting your
budget ($${body.budget.min}-$${body.budget.max}) or preferences.`,
          totalMatches: 0,
          processingTime: Date.now() - startTime,
          topCategories: [],
          budgetUtilization: { min: 0, max: 0, average: 0 },
          warnings: ['No matching gifts found. Consider adjusting your criteria.'],
        } as RecommendationResponse);
      }

      // Calculate response metadata
      const topCategories = [...new Set(recommendations.map(r =>
r.product.category))].slice(0, 5);
      const prices = recommendations.map(r => r.product.estimatedPrice || 0);
      const budgetUtilization = {
        min: Math.min(...prices),
        max: Math.max(...prices),
        average: Math.round(prices.reduce((a, b) => a + b, 0) / prices.length),
      };

      // Generate recipient summary
      let recipientSummary = `Based on `;
      if (body.recipient.threeWords && body.recipient.threeWords.length > 0) {
        recipientSummary += `your description of ${body.recipient.name || 'them'} as
${body.recipient.threeWords.join(', ')}`;
      } else if (body.recipient.giftingProfile) {
        recipientSummary += `their ${body.recipient.giftingProfile.style} gifting
style`;
      } else {
        recipientSummary += `the occasion and your preferences`;
      }
      recipientSummary += `, here are ${recommendations.length} AI-powered personal-
ized gift recommendations.`;

      const response: RecommendationResponse = {
        success: true,
        recommendations,
        recipientSummary,
        totalMatches: recommendations.length,
        processingTime: Date.now() - startTime,
        topCategories,
        budgetUtilization,
      };

      return NextResponse.json(response);

    } catch (aiError) {
      // Handle AI-specific errors
      console.error('AI gift generation error:', aiError);

      const errorMessage = aiError instanceof Error ? aiError.message : 'Failed to
generate gift recommendations';

      return NextResponse.json(
        {
          success: false,
          error: {
            code: 'AI_ERROR',
            message: errorMessage,
          },
        } as RecommendationResponse,
        { status: 500 }
      );
    }

  } catch (error) {
    console.error('Gift recommendations error:', error);

    return NextResponse.json(
      {
        success: false,
        error: {
          code: 'INTERNAL_ERROR',
          message: 'An error occurred while processing your request',
        },
      } as RecommendationResponse,
      { status: 500 }
    );
  }
}

════════════════════════════════════════════════════════════
SUMMARY:
This Next.js API route handler processes POST requests for gift recommendations. It
```

```
validates
incoming requests (recipient, occasion, budget), transforms them into GiftInput
format, calls
the OpenAI integration to generate recommendations, and returns structured JSON re-
sponses with
metadata (processing time, budget utilization, top categories). Includes comprehensive
error
handling for validation failures and AI errors.
```

```
════════════════════════════════════════════════════════════════════════════════
FILE: /home/ubuntu/code_artifacts/birthdaygen.com/src/lib/gifts/engagement-processor.t
s
════════════════════════════════════════════════════════════════════════════════

/**
 * Engagement Game Processor
 * Phase 3 - BirthdayGen.com (AI Gift Recommendation Foundation)
 *
 * Processes engagement game answers into structured data for recommendations.
 * Maps game data to recommendation parameters and enhances recipient profiles.
 */

import type {
  EngagementGameAnswers,
  RecipientProfile,
  ThreeWordsGameAnswer,
  PickTheirVibeGameAnswer,
} from '@/lib/gifts/schema';
import { VIBE_OPTIONS, GiftCategory } from '@/lib/gifts/schema';
import type { EnrichedContact, GiftingStyle } from '@/lib/autopopulate/types';

// ============================================================================
// ENGAGEMENT DATA PROCESSING
// ============================================================================

/**
 * Process ThreeWordsGame answer to extract insights
 */
export function processThreeWordsAnswer(answer: ThreeWordsGameAnswer): {
  inferredGiftingStyle: GiftingStyle | null;
  preferredCategories: GiftCategory[];
  personalityInsights: string[];
} {
  const { extractedTraits } = answer;

  // Infer gifting style from personality traits
  let inferredGiftingStyle: GiftingStyle | null = null;

  if (extractedTraits.personality.includes('creative') || extrac-
tedTraits.personality.includes('artistic')) {
    inferredGiftingStyle = 'creative' as GiftingStyle;
  } else if (extractedTraits.personality.includes('adventurous') || extractedTraits.pe
rsonality.includes('outdoorsy')) {
    inferredGiftingStyle = 'experiential' as GiftingStyle;
  } else if (extractedTraits.personality.includes('tech') || extractedTraits.personal-
ity.includes('intellectual')) {
    inferredGiftingStyle = 'tech_savvy' as GiftingStyle;
  } else if (extractedTraits.personality.includes('sophisticated') || extrac-
tedTraits.aesthetic.includes('luxurious')) {
    inferredGiftingStyle = 'luxurious' as GiftingStyle;
  } else if (extractedTraits.personality.includes('practical') || extractedTraits.per-
sonality.includes('pragmatic')) {
    inferredGiftingStyle = 'practical' as GiftingStyle;
  } else if (extractedTraits.personality.includes('thoughtful') || extractedTraits.per
sonality.includes('caring')) {
    inferredGiftingStyle = 'sentimental' as GiftingStyle;
  } else if (extractedTraits.aesthetic.includes('natural') || extractedTraits.person-
ality.includes('spiritual')) {
    inferredGiftingStyle = 'eco_conscious' as GiftingStyle;
  }
  // Map traits to preferred gift categories
  const preferredCategories: GiftCategory[] = [];

  // Personality-based categories
  if (extractedTraits.personality.includes('creative')) {
    preferredCategories.push(GiftCategory.ART, GiftCategory.HANDMADE);
  }
  if (extractedTraits.personality.includes('adventurous')) {
    preferredCategories.push(GiftCategory.EXPERIENCES, GiftCategory.TRAVEL, GiftCat-
egory.SPORTS);
  }
  if (extractedTraits.personality.includes('tech')) {
    preferredCategories.push(GiftCategory.TECH, GiftCategory.GAMING);
  }
  if (extractedTraits.personality.includes('sophisticated')) {
    preferredCategories.push(GiftCategory.FASHION, GiftCategory.BEAUTY, GiftCategory.W
INE);
  }
  if (extractedTraits.personality.includes('outdoorsy')) {
    preferredCategories.push(GiftCategory.SPORTS, GiftCategory.WELLNESS, GiftCat-
egory.EXPERIENCES);
  }
  if (extractedTraits.personality.includes('spiritual')) {
    preferredCategories.push(GiftCategory.WELLNESS, GiftCategory.BOOKS);
  }

  // Aesthetic-based categories
  if (extractedTraits.aesthetic.includes('luxurious') || extractedTraits.aesthetic.in-
cludes('glam')) {
    preferredCategories.push(GiftCategory.FASHION, GiftCategory.JEWELRY, GiftCat-
egory.BEAUTY);
  }
  if (extractedTraits.aesthetic.includes('minimalist')) {
    preferredCategories.push(GiftCategory.HOME_DECOR, GiftCategory.TECH);
  }
  if (extractedTraits.aesthetic.includes('vintage')) {
    preferredCategories.push(GiftCategory.BOOKS, GiftCategory.MUSIC, GiftCategory.HOME
_DECOR);
  }
  if (extractedTraits.aesthetic.includes('bohemian')) {
    preferredCategories.push(GiftCategory.HANDMADE, GiftCategory.ART, GiftCategory.JEW
ELRY);
  }
  if (extractedTraits.aesthetic.includes('natural')) {
    preferredCategories.push(GiftCategory.ECO_FRIENDLY, GiftCategory.WELLNESS);
  }

  // Tone-based insights (for messaging)
  const personalityInsights: string[] = [];

  if (extractedTraits.tone.includes('playful')) {
    personalityInsights.push('Has a fun, lighthearted personality');
  }
  if (extractedTraits.tone.includes('sophisticated')) {
    personalityInsights.push('Appreciates refined, elegant things');
  }
  if (extractedTraits.tone.includes('warm')) {
    personalityInsights.push('Values emotional connection and thoughtfulness');
  }
  if (extractedTraits.tone.includes('edgy')) {
    personalityInsights.push('Likes bold, unconventional choices');
  }
  if (extractedTraits.tone.includes('calm')) {
    personalityInsights.push('Prefers peaceful, serene experiences');
  }

  // Remove duplicates from categories
  const uniqueCategories = Array.from(new Set(preferredCategories));

  return {
    inferredGiftingStyle,
    preferredCategories: uniqueCategories,
    personalityInsights,
  };
}

/**
 * Process PickTheirVibeGame answer to extract category preferences
 */
export function processPickTheirVibeAnswer(answer: PickTheirVibeGameAnswer): {
  preferredCategories: GiftCategory[];
  aestheticTags: string[];
  vibeDescriptions: string[];
} {
  const { selectedVibes } = answer;

  const preferredCategories: GiftCategory[] = [];
  const aestheticTags: string[] = [];
  const vibeDescriptions: string[] = [];

  selectedVibes.forEach((vibeId) => {
    const vibe = VIBE_OPTIONS.find((v) => v.id === vibeId);
    if (vibe) {
      // Add associated categories
      vibe.associatedCategories.forEach((cat) => preferredCategories.push(cat));

      // Add aesthetic tags
      vibe.aestheticTags.forEach((tag) => aestheticTags.push(tag));

      // Add vibe description
      vibeDescriptions.push(vibe.description);
    }
  });

  // Remove duplicates
  const uniqueCategories = Array.from(new Set(preferredCategories));
  const uniqueAestheticTags = Array.from(new Set(aestheticTags));

  return {
    preferredCategories: uniqueCategories,
    aestheticTags: uniqueAestheticTags,
    vibeDescriptions,
  };
}

// ============================================================================
// RECIPIENT PROFILE ENHANCEMENT
// ============================================================================

/**
 * Enhance recipient profile with engagement game data
 * Combines auto-populate enriched contact data with engagement game insights
 */
export function enhanceRecipientProfile(
  baseProfile: Partial<RecipientProfile>,
  engagementAnswers: EngagementGameAnswers,
  enrichedContact?: EnrichedContact | null
): RecipientProfile {
  // Start with base profile
  const enhanced: RecipientProfile = {
    name: baseProfile.name || 'Recipient',
    relationship: baseProfile.relationship,
    giftingProfile: baseProfile.giftingProfile,
    archetypes: baseProfile.archetypes,
    interests: baseProfile.interests || [],
  };

  // Add enriched contact data if available
  if (enrichedContact) {
    enhanced.relationship = enrichedContact.inferredRelationship?.type;
    enhanced.giftingProfile = enrichedContact.giftingProfile;
    enhanced.archetypes = enrichedContact.archetypes;
    enhanced.enrichmentConfidence = enrichedContact.enrichmentMetadata?.confidence?.ov
erall;
  }

  // Process ThreeWordsGame data
  if (engagementAnswers.threeWords) {
    enhanced.threeWords = engagementAnswers.threeWords.words;

    const processed = processThreeWordsAnswer(engagementAnswers.threeWords);

    // Override gifting style if inferred from game (game data is more recent/accur-
ate)
    if (processed.inferredGiftingStyle) {
      enhanced.giftingProfile = {
        ...enhanced.giftingProfile,
        style: processed.inferredGiftingStyle,
      } as any;
    }

    // Add inferred interests from personality traits
    const personalityTraits = engagementAnswers.threeWords.extractedTraits.personal-
ity;
    enhanced.interests = Array.from(new Set([...(enhanced.interests || []), ...person-
alityTraits]));
  }

  // Process PickTheirVibeGame data
  if (engagementAnswers.pickTheirVibe) {
    enhanced.vibes = engagementAnswers.pickTheirVibe.selectedVibes;

    const processed = processPickTheirVibeAnswer(engagementAnswers.pickTheirVibe);

    // Add aesthetic tags as interests
    enhanced.interests = Array.from(new Set([...(enhanced.interests || []), ...pro-
cessed.aestheticTags]));
  }

  return enhanced;
}

// ============================================================================
// RECOMMENDATION REQUEST BUILDER
// ============================================================================

/**
 * Build a complete recommendation request from engagement data
 */
```

```
export function buildRecommendationRequest(
  recipientProfile: RecipientProfile,
  engagementAnswers: EngagementGameAnswers,
  options: {
    occasion: string;
    budgetMin: number;
    budgetMax: number;
    budgetPreferred?: number;
    urgency?: 'low' | 'medium' | 'high';
  }
): {
  recipient: RecipientProfile;
  occasion: string;
  budget: { min: number; max: number; preferred?: number };
  engagementAnswers: EngagementGameAnswers;
  preferredCategories: GiftCategory[];
  urgency?: 'low' | 'medium' | 'high';
} {
  // Collect all preferred categories from both games
  let preferredCategories: GiftCategory[] = [];

  if (engagementAnswers.threeWords) {
    const processed = processThreeWordsAnswer(engagementAnswers.threeWords);
    preferredCategories.push(...processed.preferredCategories);
  }

  if (engagementAnswers.pickTheirVibe) {
    preferredCategories.push(...engagementAnswers.pickTheirVibe.categoryPreferences);
  }

  // Remove duplicates and keep top 5 categories
  preferredCategories = Array.from(new Set(preferredCategories)).slice(0, 5);

  return {
    recipient: recipientProfile,
    occasion: options.occasion as any,
    budget: {
      min: options.budgetMin,
      max: options.budgetMax,
      preferred: options.budgetPreferred,
    },
    engagementAnswers,
    preferredCategories,
    urgency: options.urgency,
  };
}

// ============================================================================
// INSIGHT GENERATION
// ============================================================================

/**
 * Generate human-readable insights from engagement data
 */
export function generateEngagementInsights(
  engagementAnswers: EngagementGameAnswers
): {
  summary: string;
  keyTraits: string[];
  giftingHints: string[];
} {
  const keyTraits: string[] = [];
  const giftingHints: string[] = [];
  // Process ThreeWordsGame insights
  if (engagementAnswers.threeWords) {
    const { extractedTraits } = engagementAnswers.threeWords;

    // Add personality traits
    keyTraits.push(...extractedTraits.personality.slice(0, 3));
    keyTraits.push(...extractedTraits.tone.slice(0, 2));

    // Generate gifting hints from traits
    const processed = processThreeWordsAnswer(engagementAnswers.threeWords);
    processed.personalityInsights.forEach((insight) => giftingHints.push(insight));
  }

  // Process PickTheirVibeGame insights
  if (engagementAnswers.pickTheirVibe) {
    const { selectedVibes } = engagementAnswers.pickTheirVibe;

    selectedVibes.forEach((vibeId) => {
      const vibe = VIBE_OPTIONS.find((v) => v.id === vibeId);
      if (vibe) {
        keyTraits.push(vibe.label.toLowerCase());
        giftingHints.push(`Appreciates ${vibe.description.toLowerCase()}`);
      }
    });
  }

  // Generate summary
  let summary = 'Based on your inputs, ';

  if (engagementAnswers.threeWords) {
    summary += `they are ${engagementAnswers.threeWords.words.join(', ')}`;
  }

  if (engagementAnswers.pickTheirVibe && engage-
mentAnswers.pickTheirVibe.selectedVibes.length > 0) {
    const vibeLabels = engagementAnswers.pickTheirVibe.selectedVibes
      .map((vibeId) => VIBE_OPTIONS.find((v) => v.id === vibeId)?.label.toLowerCase())
      .filter(Boolean);

    if (engagementAnswers.threeWords) {
      summary += ` with ${vibeLabels.join(' and ')} vibes`;
    } else {
      summary += `they have ${vibeLabels.join(' and ')} vibes`;
    }
  }

  summary += '. We\'ll find gifts that match their unique personality!';

  // Remove duplicates from key traits
  const uniqueKeyTraits = Array.from(new Set(keyTraits));

  return {
    summary,
    keyTraits: uniqueKeyTraits,
    giftingHints,
  };
}

---
SUMMARY:
This module processes engagement game data (ThreeWords and PickTheirVibe games) to ex-
tract
```

```
personality insights, infer gifting styles, and map traits to gift categories. It en-
hances
recipient profiles by combining auto-populated contact data with game answers, builds
recommendation requests with preferred categories, and generates human-readable sum-
maries
and gifting hints for the AI recommendation engine.
```

---

## Code Review Ready ✓

All three source files have been extracted and displayed in full. These files show the complete AI gift recommendation pipeline:

1. **openai.ts** - GPT-4o integration with structured JSON responses
2. **route.ts** - Next.js API endpoint handling recommendation requests
3. **engagement-processor.ts** - Game data processing and profile enhancement

Ready for your code review assessment.