Q&A Generation JSON Parsing Error Fix

Problem Summary

The Q&A generation system was experiencing JSON parsing errors when processing PDF files. The error manifested as:

```
JSON parse error for AI_ASSISTANT_QUICK_START.pdf: SyntaxError: Expected ',' or ']'
after array element in JSON at position 4352
Attempted to parse: {
  "qas": [
    {
      "question": "What is the main purpose of PDF/A-3 as mentioned in the document?",
      "answer": "The primary function of a digital asset management system (DAMS)
within an or...
```

The response was being truncated mid-sentence, causing the JSON parser to fail.

Root Cause

The issue was caused by **insufficient token limits** in the Ollama API call:

1. **Line 445**: `num_predict: 1000` - This parameter limited responses to 1000 tokens
2. When generating Q&As with detailed answers, responses were cut off before completion
3. Truncated responses resulted in invalid JSON that couldn't be parsed

Changes Made

1. Increased Token Limit (Line 445)

Before:

```
num_predict: 1000, // Limit response length
```

After:

```
num_predict: 2048, // FIXED: Increased from 1000 to prevent truncation
```

Rationale: Doubled the token limit to allow for complete Q&A generation with detailed answers. 2048 tokens provides sufficient space for 2-3 Q&A pairs with comprehensive answers.

2. Enhanced Truncation Detection (Lines 475-491)

Added early detection of truncated responses before attempting to parse:

```
// FIXED: Check if response was truncated by Ollama
const responseText = data.response;
const wasTruncated = data.done === false ||
                     !responseText.trim().endsWith('}') ||
                     responseText.includes('...');

if (wasTruncated) {
  console.warn(`Response appears truncated for ${fileName}${chunkLabel}`);
  console.warn(`Response length: ${responseText.length} chars`);
  console.warn(`Response end: ...${responseText.substring(Math.max(0, response-
Text.length - 100))}`);

  if (retries < MAX_RETRIES) {
    console.log(`Retrying with increased token limit (attempt ${retries + 1}/${MAX_RET
RIES})...`);
    retries++;
    continue;
  }
}
```

Benefits:
- Detects truncation before JSON parsing
- Provides detailed logging for debugging
- Triggers automatic retry with same parameters
- Prevents wasted parsing attempts on incomplete responses

3. Improved JSON Validation (Lines 532-559)

Enhanced the `parseGeneratedQAs` function with better validation:

```
// FIXED: Validate JSON completeness before parsing
const hasOpeningBrace = cleanedResponse.includes('{');
const hasClosingBrace = cleanedResponse.includes('}');

if (!hasOpeningBrace || !hasClosingBrace) {
  console.error(`Incomplete JSON response for ${sourceFile} - missing braces`);
  console.error(`Response preview: ${response.substring(0, 300)}...`);
  console.error(`Response end: ...${response.substring(Math.max(0, response.length - 1
00))}`);
  return [];
}

// Additional validation - check for truncation indicators
if (jsonString.includes('...') || !jsonString.trim().endsWith('}')) {
  console.error(`Response appears truncated for ${sourceFile}`);
  console.error(`JSON end: ...${jsonString.substring(Math.max(0, jsonString.length - 1
50))}`);
  return [];
}
```

Benefits:
- Validates JSON structure before parsing
- Checks for truncation indicators
- Provides detailed error context showing both start and end of response
- Fails fast on obviously incomplete responses

4. Enhanced Error Logging (Lines 565-578)

Improved error messages to show exact parse failure location:

```
// FIXED: Log the exact position where parsing failed
if (parseError instanceof SyntaxError) {
  const match = parseError.message.match(/position (\d+)/);
  if (match) {
    const position = parseInt(match[1]);
    const contextStart = Math.max(0, position - 100);
    const contextEnd = Math.min(jsonString.length, position + 100);
    console.error(`Parse error context: ...${jsonString.substring(contextStart, con-
textEnd)}...`);
  }
}
```

Benefits:
- Shows exact context around parse error
- Makes debugging much easier
- Helps identify patterns in failures

5. Improved Retry Logic (Lines 502-511)

Enhanced retry behavior with better feedback:

```
if (retries < MAX_RETRIES) {
  console.log(`Retrying (attempt ${retries + 1}/${MAX_RETRIES})...`);
  retries++;
  await new Promise(resolve => setTimeout(resolve, 1000)); // Wait 1s before retry
  continue;
}

console.error(`Failed to generate valid Q&As after ${MAX_RETRIES} retries`);
break;
```

Benefits:
- Adds 1-second delay between retries to avoid overwhelming Ollama
- Provides clear retry attempt logging
- Explicit failure message after exhausting retries

Testing

A test script has been created at `scripts/test-qa-fix.ts` to verify the fix:

```
npx tsx scripts/test-qa-fix.ts
```

This script:
- Tests Q&A generation on the problematic PDF file
- Monitors job progress
- Reports success/failure with detailed status

Expected Behavior After Fix

1. **Complete Responses**: Ollama responses will have sufficient token budget to complete Q&A generation
2. **Early Detection**: Truncated responses are detected before parsing attempts
3. **Automatic Retry**: System automatically retries on truncation
4. **Better Logging**: Detailed error messages help diagnose any remaining issues
5. **Graceful Degradation**: System handles failures without crashing

Configuration Parameters

Current settings in `qa-generator.ts`:

- `num_predict`: 2048 tokens (increased from 1000)
- `MAX_RETRIES`: 2 attempts
- `QA_GENERATION_TIMEOUT`: 600 seconds (10 minutes)
- `temperature`: 0.3 (for consistent JSON output)
- `format`: 'json' (enforces JSON response format)

Future Improvements

Consider these enhancements if issues persist:

1. **Dynamic Token Adjustment**: Increase `num_predict` on retry attempts
2. **Streaming with Accumulation**: Use streaming mode with proper buffer accumulation
3. **Response Validation**: Add JSON schema validation before parsing
4. **Model Selection**: Test with different models that may handle JSON better
5. **Chunking Optimization**: Adjust `CHUNK_SIZE` based on model performance

Related Files

- `src/lib/services/qa-generator.ts` - Main Q&A generation service
- `scripts/test-qa-fix.ts` - Test script for verification
- `src/app/api/ai/qa-generate/route.ts` - API endpoint
- `ai-assistant/services/ollamaService.ts` - Ollama service wrapper