10 Pic Push - Implementation Guide

🚀 Quick Start

Prerequisites

1. n8n instance (self-hosted or cloud)
2. OpenAI API key with GPT-4 access
3. Firebase project with Firestore enabled
4. Image storage solution (Firebase Storage, UploadThing, etc.)

Step 1: Import Workflow

1. Copy the JSON workflow from the main artifact
2. In n8n, go to Workflows → Import
3. Paste the JSON and click Import

Step 2: Configure Credentials

OpenAI Configuration

1. Go to Credentials → New → OpenAI
2. Add your API key
3. Name it and save

Firebase Configuration

1. Go to Credentials → New → Firebase
2. Add your service account JSON
3. Set the database URL

Step 3: Update Node References

1. Vision Analyzer node: Update the API endpoint if using a different vision service
2. Firebase Writer: Confirm collection name matches your structure
3. Webhook: Note the webhook URL for your frontend

Step 4: Frontend Integration

javascript

```javascript
// Example frontend call
const generate10PicPush = async (userData) => {
  const response = await fetch('YOUR_N8N_WEBHOOK_URL/10-pic-push', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      userId: userData.id,
      images: userData.imageUrls, // Array of 2-10 image URLs
      quiz: {
        tone: userData.tone,
        vibes: userData.vibes, // Array of strings
        metaphor: userData.metaphor,
        audience: userData.audience,
        sampleCopy: userData.sampleCopy // Optional
      }
    })
  });

  return response.json();
};
```

Step 5: Testing

1. Use n8n's test webhook feature
2. Send a sample payload:

json

```json
{
  "userId": "test_user_123",
  "images": [
    "https://storage.example.com/image1.jpg",
    "https://storage.example.com/image2.jpg"
  ],
  "quiz": {
    "tone": "playful yet professional",
    "vibes": ["bold", "creative", "authentic"],
    "metaphor": "My brand is a neon-lit jazz club",
    "audience": "creative entrepreneurs",
    "sampleCopy": "Welcome to where creativity meets strategy..."
  }
}
```

🔧 Customization Options

Adjust Content Output

- Modify temperature in GPT nodes (0.7-0.9 range)
- Update max_tokens for longer/shorter content
- Change prompt instructions for different styles

Add Content Types

1. Duplicate an existing generator node
2. Update the prompt for new content type
3. Add to output formatter
4. Update Firebase schema

Enhance Vision Analysis

- Add multiple image analysis for better insights
- Implement style transfer detection
- Extract text from images for brand consistency

📊 Monitoring & Analytics

Track Key Metrics

- Webhook response times
- Content generation success rate
- Most selected vibes/tones
- Platform preference patterns

Firebase Queries

javascript

```javascript
// Get all results for a user
db.collection('10PicResults')
  .where('userId', '==', 'user_123')
  .orderBy('timestamp', 'desc')
  .limit(10)

// Analyze popular vibes
db.collection('10PicResults')
  .get()
  .then(snapshot => {
    const vibesCount = {};
    snapshot.forEach(doc => {
      doc.data().metadata.vibes.forEach(vibe => {
        vibesCount[vibe] = (vibesCount[vibe] || 0) + 1;
      });
    });
    console.log(vibesCount);
  });
```

🐛 Troubleshooting

Common Issues

1. **Timeout Errors**
   - Increase webhook timeout in n8n settings
   - Consider implementing queue system for high volume
2. **Vision API Failures**
   - Verify image URLs are publicly accessible
   - Check image size limits (typically 20MB)
   - Ensure proper image format (JPEG, PNG)
3. **Empty GPT Responses**

    - Check API quota/limits
    - Verify JSON parsing in prompts
    - Test with simplified inputs
4. **Firebase Write Errors**
    - Confirm credentials have write permissions
    - Check Firestore rules allow writes
    - Verify collection exists

🎯 Performance Optimization

Speed Improvements

1. Cache vision analysis results by image hash
2. Pre-warm GPT models with common patterns
3. Implement parallel processing where possible

Cost Optimization

1. Use GPT-3.5 for simpler tasks (hooks, formatting)
2. Batch similar requests
3. Implement result caching for common inputs

Scaling Considerations

- Add queue management for high volume
- Implement rate limiting
- Consider horizontal scaling of n8n instances
- Use Firebase Cloud Functions for post-processing

🔐 Security Best Practices

1. **Input Validation**
    - Verify image URLs before processing
    - Sanitize user text inputs
    - Limit file sizes and types
2. **API Security**
    - Use environment variables for keys
    - Implement request authentication

    - Add rate limiting to webhook

3. **Data Privacy**
    - Set Firebase retention policies
    - Anonymize data for analytics
    - Implement user data deletion endpoint

📈 Future Enhancements

1. **Multi-language Support**
    - Add language detection
    - Translate outputs
    - Culturally adapt content
2. **Advanced Analytics**
    - Track content performance
    - A/B test different prompts
    - User journey mapping
3. **Integration Expansions**
    - Direct posting to social platforms
    - Calendar integration for scheduling
    - Design tool integration for visuals