

The Complete **Nano Banana**

AI Image Editing with **Google Gemini**

Build an Empire with Prompt Templates & Python App Development

Mammoth Club Official Guide PRO+

- ✓ FREE Online Course
- ✓ FREE Cheatsheet
- ✓ FREE Exam
- ✓ FREE Official Mammoth Club Certificate



MAMMOTH CLUB



Written by Alex Kropf • Produced by John Bura

Cover Design by Jared Matson • Contributions by James Dabalus

Powered by  CoursePro.ai

*From the creators of the best-selling Hello Coding: Anyone
Can Learn to Code & more*

Praise for Mammoth Club

I have completed many tutorials. This one is the most outstanding one that I have seen thus far. It is doubtful that it could be topped. This is a superior tutorial. Amazing. —Joseph A., Mammoth Club Student

Exactly what I wanted! Just enough BASIC information without being technically overwhelming and intimidating. —Paul V., Mammoth Club Student

This course so far is by far amazing! The instructor is very encouraging and upbeat, and his instructions are very clear. It's an amazing course. —Moiz S., Mammoth Club Student

It's scary to think that by following these instructional videos I can be equipped with the skills to program Python. —Charles E., Mammoth Club Student

I ended up taking it and it was INCREDIBLE. They set great challenges that build off what was taught in the chapter, but don't directly give you the answer. It asks you to extend your knowledge and refer to the right documentation. So good for learning. —A_Unicycle, Mammoth Club Student

This is AMAZING! I just learned how to code without breaking a sweat, this is really easy and fun! —Shalonda L., Mammoth Club Student

Clear instructions and excellent projects. —Ian F., Mammoth Club Student



MAMMOTH CLUB



For 3,000+ courses & 5,000+ video hours: MammothClub.com!

Mammoth Club is a leading online course provider in everything from learning to code to becoming a YouTube star. Since 2011, Mammoth Club has built a global student community with over 9 million courses sold.



Scan the QR code to redeem your free course, exam and cheatsheet! Or go to to this link:

mammothclub.com/course/1-hour-nano/BANANA

Mammoth Club books can be purchased at a special discount when ordered in bulk for promotional giveaways, fundraisers, or educational initiatives. Customized editions or selected excerpts can also be produced to meet specific needs. For more information, please reach out to support@mammothinteractive.com.

Portions of this book may be shared promotionally if with direct citation to MammothClub.com. This book may not be reproduced — mechanically, electronically, or by any other means, including photocopying — without written permission of the publisher.



The publisher does not provide medical, legal, accounting, or other professional services. Readers seeking such expertise should consult a qualified professional. This book is not meant to be used for clinical procedures or medical treatment. To the maximum extent permitted by law, the publisher and editors are not responsible for any harm or damage to individuals or property resulting from the use or misuse of the material presented herein. All rights reserved. This book does not constitute financial, investment, legal, or tax advice. You are solely responsible for your financial decisions. We make no guarantees of income, business outcomes, or investment returns. By using this book, you agree that the author and publisher cannot be held liable for any loss, damage, or results arising from actions you take based on its content.

Written by Alex Kropf • Produced by John Bura • Online Course, Exam and Cheatsheet by James Dabalus • Cover Design by Jared Matson

Copyright © 2025 by Mammoth Club

Go to MammothClub.com • 3 • Powered by CoursePro.ai

For 3,000+ courses & 5,000+ video hours: MammothClub.com!	3
Welcome to Nano Banana: AI Image Mastery	6
PART 1: What can you do with Google's AI Image Editing?	10
Generative AI Fundamentals	11
What is Nano Banana (Gemini)?	17
What are Nano Banana's Advanced Capabilities?	24
Advanced Consistency & Semantic Understanding in Nano Banana	28
PART 2: Best Practices for Prompting Images	33
Prompt Templates for Image Generation	36
Best Practices for Image Editing Prompts	42
PART 3: Use Cases of AI Image Editing	47
AI Edit Images for the Business Professional	47
Brand Style Guide Automation	49
Edit Marketing Images with Prompt Templates	53
Image Editing Prompt Templates for Professionals	57
Creative and Lifestyle AI Image Use Cases	59
Advanced Composition Techniques for Professional Photography	64
Prompt Templates for Creative AI Images	67
Business and Media AI Image Applications	73
Business and Media AI Image Editing Prompt Templates	77
AI Image App Development: Build an Empire	87
Generate AI Apps with Prompt Templates	96
PART 4: Essentials of Python App Development	101
Run Python Code Online with Google Colab	101
Set Up a Colab Project	102
Set Up Python Project with Gemini SDK	106
Python Coding Fundamentals for AI App Development	109
PART 5: Let's build it! Generate Images with Python SDK	113
Access Nano Banana with Python	113
Generate Multiple Images for Storytelling	116
Chat Mode for Iterative Image Generation	119

PART 6: AI Image Editing with Python SDK	122
Add & Remove Image Elements with Gemini SDK	122
Inpaint & Mask with Gemini SDK.....	125
Style Transfer with Gemini SDK.....	128
Combine Multiple Images with Gemini SDK.....	131
High-Fidelity Details with Gemini SDK.....	133
PART 7: Build AI Apps with Google AI Studio	136
Getting Started with Google AI Studio.....	136
Generate Image Apps with Google AI Studio	141
PART 8: Build Enterprise Apps with Google Vertex AI	142
Use the Vertex AI API	142
Use the Vertex AI Studio on Google Cloud Platform	145
Deploying Vertex AI Prompts as Web Applications	146
PART 9: Performance and Optimization	152
Batch Mode Processing for Large-Scale AI Operations.....	152
Context Caching with Gemini 2.5 Flash (Nano Banana)	160
PART 10: Authentication and Authorization	167
What's the Difference?	167
Types of Access Credentials	170
Authenticating with OAuth for Gemini 2.5 Flash (Nano Banana)	173
Optimization, API Rate Limiting and Cost Management	178
PART 11: Integration and Automation	182
Webhook Integrations for Automated Workflows	182
Third-Party Service Automation.....	184
Automated Backup and Version Control Systems.....	188
Database Integration for Image Metadata Management.....	191
Epilogue: Your AI Image Mastery	217
Get the FREE Online Course & Certificate	218
Visit MammothClub.com	219

Welcome to Nano Banana: AI Image Mastery

The future of visual content creation is here, and it's more accessible than ever before. Google's revolutionary Gemini AI image technology – affectionately dubbed "Nano Banana" – has democratized professional-quality image creation, editing, and manipulation in ways that would have seemed impossible just a few years ago.

The Visual Revolution at Your Fingertips

Imagine being able to create stunning marketing visuals, generate product photography without expensive equipment, or build entire visual campaigns with nothing more than descriptive text. This isn't science fiction – it's the reality that AI image generation has made possible, and you're about to master it completely.

Whether you're a business professional tired of expensive stock photography, a creative seeking new artistic possibilities, a developer building the next breakthrough app, or simply someone fascinated by the intersection of artificial intelligence and visual creativity, this book transforms you from curious observer to confident practitioner.

Your Complete Learning Journey

Google's Gemini image generation technology has revolutionized visual content creation. Professional-quality images, sophisticated editing, and complex compositions are now achievable through natural language descriptions.

AI image generation eliminates traditional barriers to professional visual content. You no longer need a design degree to generate stunning marketing visuals. Expensive photography equipment becomes unnecessary when you can create product shots through text descriptions. Manual editing transforms into conversational refinement, while creative possibilities expand beyond what traditional tools allow.

Part 1: AI Image Editing Fundamentals

Understanding generative AI concepts provides your foundation. Model architecture and training methodologies become clear as you progress through the material.



When comparing Gemini and Imagen capabilities, you'll discover which tool serves each project best.

Identity-preserving transformations maintain visual consistency across image series. Photo combination features enable complex compositions that would challenge traditional methods. Advanced capabilities unlock possibilities:

- Semantic understanding that integrates real-world knowledge
- Multi-image processing for sophisticated projects
- Content authenticity measures for professional applications

Part 2: Prompt Engineering Mastery

Six core principles determine result quality, though mastering them requires practice and refinement. Hyper-specific descriptions eliminate ambiguity while context communication guides AI decision-making effectively. Iterative refinement becomes your path to perfect results.

Dive into specialized techniques for photorealistic generation. Stylized graphics creation follows different principles than photography. Typography integration requires specific approaches, while commercial product photography demands understanding of visual psychology and buyer behavior.

Part 3: Professional Use Cases and Applications

Business applications transform your professional presence. Profile optimization creates powerful first impressions, while brand consistency development ensures cohesive visual identity. Marketing material creation rivals agency work when you apply the templates and techniques provided.

Creative applications expand your artistic possibilities. Social media presence benefits from AI-enhanced visual content. Pet and family photography improves through subtle enhancements that preserve natural beauty while eliminating distractions.

Commercial implementations span multiple industries. E-commerce photography increases conversion rates through perfect product presentation. Real estate marketing sells properties faster with enhanced visuals. Media production workflows reduce costs while maintaining professional quality.

Part 4: Python Development Essentials

Google Colab serves as your development environment with pre-installed libraries and seamless package management. Learn the Gemini SDK installation process step-by-step. Python fundamentals are taught specifically for AI applications, focusing on practical implementation rather than theoretical concepts.

API key management and billing monitoring prevent unexpected costs during development.

Part 5: Image Generation Implementation

Build your first image generation applications using the complete workflow provided. Import statements, client setup, and input preparation establish the foundation. API requests and response processing handle the core functionality.

Multiple image generation enables storytelling applications where narrative consistency matters. Chat mode allows iterative refinement through natural conversation, making complex edits accessible to non-technical users.

Part 6: Advanced Image Editing with Python

Element addition and removal provide surgical precision when modifying existing images. Inpainting and semantic masking achieve realistic modifications that preserve photographic authenticity. Style transfer creates artistic transformations that professional artists would applaud.

Multi-image composition combines separate visuals into seamless results. High-fidelity detail preservation maintains quality during complex modifications, ensuring professional standards in final outputs.

Part 7: Google AI Studio Applications

Custom chat interfaces expand functionality beyond basic generation. Character development examples demonstrate specialized applications for specific use cases. Building iterative improvement workflows optimizes result quality systematically.

Production transition strategies ensure your prototypes scale effectively to real-world deployment.

Part 8: Enterprise Development with Vertex AI

Vertex AI provides enterprise-scale capabilities when basic solutions aren't sufficient. Web application integration offers user-friendly access to complex functionality. Advanced deployment strategies ensure reliable operation under production loads.

Performance monitoring and cost management maintain efficiency while scaling. Application management tools provide operational visibility across large deployments.

Part 9: Advanced API Features and Optimization

Batch processing handles large-scale operations efficiently when individual requests become impractical. Context caching optimizes both performance and costs significantly. Complex workflow automation streamlines repetitive tasks that would otherwise consume substantial time.

Token usage optimization maintains budget control while maximizing output quality.

Part 10: Authentication and Security

Authentication methods serve different security requirements depending on your application needs. API keys provide simple access for basic applications, while OAuth credentials enable user-based access control. Service account credentials support enterprise deployment scenarios.

Learn secure credential management that ensures production readiness. Master access control configuration to protect sensitive operations.

Skills You'll Master

Creative excellence enables professional image generation for any business requirement. Technical expertise allows custom application development using Python and Google's comprehensive APIs. Strategic thinking guides platform selection based on project needs rather than arbitrary preferences.

Professional implementation skills ensure secure, scalable deployment that meets enterprise standards.

PART 1: What can you do with Google's AI Image Editing?

"Nano Banana" is the codename for a new image editing model, officially known as Gemini 2.5 Flash Image, made by Google's DeepMind research lab.

Did you know? Google DeepMind is an AI lab that develops advanced AI systems to solve complex problems and push the boundaries of machine learning.

Gemini is Google DeepMind's family of advanced AI models designed to handle text, images, code, and other data types, powering chatbots, search, productivity tools, and developer APIs.

What is Artificial Intelligence (AI) and Machine Learning (ML)?

Artificial Intelligence (AI) is the field of computer science focused on creating systems that can perform tasks requiring human-like intelligence, such as reasoning, learning, problem-solving, natural language processing, text generation, image generation, image editing and much more.

Machine Learning (ML) is a branch of artificial intelligence where computers learn patterns from data and improve performance on tasks without being explicitly programmed.

*In AI and ML, a **model** is a mathematical or computational representation (such as files of code) of a system that has been trained on data to make predictions, recognize patterns, or generate outputs.*

There are models built to perform specific tasks, like recognize an apple in an image. There are also more general models (like Nano Banana) that can complete a variety of tasks, such as generating and editing images. Now that deserves a "wow!"

Generative AI Fundamentals

What is Generative AI?

Generative AI creates new content by learning patterns from existing data. Unlike traditional software that follows programmed rules, generative AI produces original outputs based on learned patterns.

Core Capabilities

Content Type	AI Capability	Examples
Text	Language generation and understanding	Articles, conversations, code
Images	Visual creation and modification	Artwork, photos, designs
Audio	Sound and music generation	Music, voice, sound effects
Video	Moving image creation	Animations, films, presentations

AI models train on massive datasets containing examples of the content they will generate:

- **Text models:** Books, articles, websites, conversations
- **Image models:** Millions of photographs, artwork, illustrations
- **Multimodal models:** Combined text and visual content

Training creates neural networks that recognize patterns and relationships within data, enabling prediction of what comes next or what fits together.

Model Architecture and Training

Neural networks are a type of AI model that use interconnected nodes (neurons) that process information through multiple layers. Each connection has a weight that determines influence on the final output.

Training Process:

1. Feed model examples of input-output pairs
2. Model makes predictions based on current weights
3. Compare predictions to correct answers
4. Adjust weights to improve accuracy
5. Repeat millions of times across entire dataset

Large Language Models (LLMs)

LLMs specialize in understanding and generating human language:

- Process text as sequences of tokens (words or word parts)
- Predict next tokens based on previous context
- Scale enables understanding of complex relationships
- Transformer architecture handles long-range dependencies

Key AI Concepts

Tokens and Context

Tokens: Basic units of text processing

- Words, word parts, or punctuation marks
- "Hello world!" = approximately 3 tokens
- Context window: maximum tokens model can process at once

Context Window Limitations:

- Determines maximum conversation length
- Older information may be "forgotten" in long sessions
- Affects consistency across extended interactions

Prompts and Prompt Engineering

Prompts: Instructions or questions given to AI models

- Quality of prompts dramatically affects output quality
- Specific, detailed prompts produce better results
- Context and examples improve response accuracy

Effective Prompt Elements:

- Clear instructions and desired outcomes
- Relevant context and background information
- Examples of preferred format or style
- Constraints and limitations

Temperature and Randomness

AI models can adjust creativity through temperature settings:

Temperature	Behavior	Use Cases
0.0-0.3	Deterministic, consistent	Factual content, code generation
0.4-0.7	Balanced creativity	General conversation, analysis
0.8-1.0	Creative, varied	Artistic content, brainstorming

Generative AI Limitations

Common Issues

Hallucinations: AI confidently generates incorrect information

- Models predict plausible-sounding content regardless of accuracy
- Always verify factual claims and technical details
- Provide references and grounding data when possible

Inconsistency: Responses vary between identical prompts

- Randomness built into generation process
- Same question can produce different answers
- Important for creative tasks, problematic for factual content

Context Loss: Models forget earlier conversation parts

- Limited context window creates memory constraints
- Important information may need repetition
- Long sessions require context refreshing

Training Data Dependencies

AI capabilities reflect training data characteristics:

- Knowledge cutoff dates limit current information
- Bias in training data affects model outputs
- Cultural and linguistic representation varies
- Specialized domains may have limited coverage

AI Model Types

Text Generation Models

Capabilities:

- Writing, editing, and summarizing content
- Code generation and debugging

- Question answering and analysis
- Translation between languages

Image Generation Models

Capabilities:

- Creating artwork from text descriptions
- Photo editing and modification
- Style transfer between images
- Product visualization and mockups

Multimodal Models

Combined Capabilities:

- Process both text and images simultaneously
- Generate images from text descriptions
- Describe and analyze uploaded images
- Edit images using text instructions

Practical Applications

Content Creation

Application	AI Contribution	Human Oversight
Writing	Draft generation, editing	Fact-checking, strategy
Design	Concept visualization	Brand alignment, refinement
Code	Implementation, debugging	Architecture, testing
Marketing	Asset creation, copy	Strategy, brand voice

Workflow Integration

AI as Assistant:

- Handles routine and repetitive tasks
- Provides suggestions and alternatives
- Accelerates initial creation phases
- Enables rapid iteration and testing

Human Leadership:

- Strategic decision making
- Quality control and validation
- Creative direction and vision
- Relationship management and communication

Understanding AI Responses

Response Structure

AI models generate responses in structured formats:

- Text content in natural language
- Confidence indicators for factual claims
- Alternative suggestions when appropriate
- Metadata about generation process

Quality Assessment**Evaluation Criteria:**

- Relevance to prompt requirements
- Factual accuracy and consistency
- Appropriate tone and style
- Completeness of requested information

Responsible AI Usage

Best Practices

- Verify important factual information independently
- Use AI as assistant rather than sole decision maker
- Understand and disclose AI involvement when relevant
- Monitor costs and usage patterns
- Respect copyright and intellectual property

Ethical Considerations

- Avoid generating harmful or misleading content
- Respect privacy in shared information
- Consider bias implications in generated content
- Use appropriate human oversight for sensitive applications

This foundation enables effective collaboration with AI tools while understanding their capabilities and limitations.

What is Nano Banana (Gemini)?

The new Gemini 2.5 Flash Image ("Nano Banana") model, integrated into the Gemini app and developer tools, allows YOU to edit images with simple text prompts, with more accuracy and realism (when working with photographs) than ever before!

The Gemini app is Google's official AI assistant app that gives YOU direct access to Gemini models, letting you chat, generate text, create and edit images, analyze data, and integrate AI into everyday tasks across mobile and web.

In AI, a prompt is the input (such as text, code, or an instruction) you give to a model to guide its response or generate a desired output like "Edit this image of me on the couch to one of me on the beach!".

Key features of the new Gemini 2.5 Flash Image ("Nano Banana") model include:

- **Consistency:** The model is designed to maintain a person or pet's likeness across different edits and scenes (when editing photographs - you can also work with a whole range of image types).
- **Prompt-based editing:** Users can use natural language to make targeted transformations, such as changing backgrounds, outfits, or adding/removing objects.
- **Multi-turn editing:** The model remembers previous commands, allowing for a series of edits to be made on the same image.
- **Image blending:** You can combine multiple photos into a single, new image.
- **World knowledge:** The model uses Gemini's understanding of the world to perform complex tasks, like editing hand-drawn diagrams.



How to get started with Nano Banana?

Just go to gemini.google.com or download the Gemini app, and sign up for an account to access image features.

Why is Nano Banana called *that*?

Google (via DeepMind) used "Nano Banana" as a fun internal codename for the Gemini 2.5 Flash Image model—highlighting its nimbleness ("Nano") and continuing the tradition of quirky AI nicknames ("Banana") teased through emoji hints.

The model appeared anonymously on benchmarks (like LMArena) under the name "Nano Banana," stirring widespread speculation. Google later confirmed it was indeed the next-generation Gemini image model.

LMarena is an online leaderboard platform that benchmarks and compares large language models (LLMs) across multiple tasks in real time.

The formal name for "Nano Banana" is Gemini 2.5 Flash Image—a powerful image-editing upgrade integrated into the Gemini app and developer tools like Gemini API, AI Studio, and VertexAI.

Note - you will see the terms "Nano Banana", "Gemini 2.5 Flash Image", "Gemini Flash Image", "Gemini Native Image Generation" and "Gemini" used interchangeably.

Gemini 2.5 Flash Image ("Nano Banana") vs Imagen

Gemini 2.5 Flash Image ("Nano Banana") is built into the Gemini app and can perform both image editing and generation. It's especially ideal for the editing side of things.

Gemini models can generate and process images conversationally. You can prompt Gemini with text, images, or a combination of both allowing you to create, edit, and iterate on visuals with unprecedented control:

- **Text-to-Image:** Generate high-quality images from simple or complex text descriptions.
- **Image + Text-to-Image (Editing):** Provide an image and use text prompts to add, remove, or modify elements, change the style, or adjust the color grading.
- **Multi-Image to Image (Composition & Style Transfer):** Use multiple input images to compose a new scene or transfer the style from one image to another.
- **Iterative Refinement:** Engage in a conversation to progressively refine your image over multiple turns, making small adjustments until it's perfect.
- **High-Fidelity Text Rendering:** Accurately generate images that contain legible and well-placed text, ideal for logos, diagrams, and posters.

High-fidelity means something that is reproduced or represented with very high accuracy and detail, closely matching the original source. Don't worry, I didn't know either!

Google provides two distinct image generation approaches: Imagen (specialized model) and Gemini Native Image (integrated capability, which includes Nano Banana). Each serves different use cases with unique strengths and trade-offs.

Attribute	Imagen	Gemini Native Image
Capability	Most advanced image generation model	Flexible conversational editing
Status	Generally available	Preview (production allowed)
Latency	Low (near real-time)	Higher (more computation)
Pricing	\$0.02-\$0.12 per image	\$30 per 1M tokens (1290 tokens/image)

*In AI and machine learning, a **token** is a unit of text (like a word, subword, or character) that models use to process and generate language.*

Imagen Strengths

Primary Advantages

- Highest image generation quality available
- Superior photorealistic output
- Enhanced clarity and sharpness
- Advanced spelling and typography rendering
- Optimized for speed and efficiency

Recommended Applications

- Photorealistic image requirements

- Artistic detail and specific styles (impressionism, anime)
- Brand asset creation and logo generation
- Product design visualization
- Typography-heavy graphics
- Professional marketing materials

Gemini Native Image Capabilities

Core Strengths

- **Access Nano Banana!**
- Conversational editing interface
- Contextual understanding integration
- Mask-free editing simplicity
- Multi-turn iterative improvements
- Advanced semantic processing

Specialized Functions

- Multi-image element combination
- Precise local modifications
- Natural language command processing
- Design texture transfer
- Iterative collaborative editing

Use Case Selection

Choose Imagen When:

- Image quality is top priority
- Photorealism required

- Specific artistic styles needed
- Typography accuracy critical
- Brand consistency important
- Speed optimization essential

Choose Gemini Native When:

- **You want to use Nano Banana**
- Conversational editing preferred
- Multiple revision rounds expected
- Complex multi-image combinations needed
- Specific element modifications required
- Creative experimentation desired
- Contextual understanding valuable

Imagen Model Variants

Imagen 4 (Standard)

- Default recommendation for most use cases
- Balanced performance and quality
- Multiple image generation capability

Imagen 4 Ultra

- Advanced use cases requiring maximum quality
- Single image generation limit
- Premium quality output
- Higher computational requirements

Cost Considerations

Imagen Pricing Structure

- Fixed per-image cost model
- Predictable expense calculation
- Range: \$0.02-\$0.12 per generated image
- Cost-effective for specialized tasks

Gemini Native Pricing

- Token-based billing system
- Flat rate: 1290 tokens per image (up to 1024x1024px)
- \$30 per million tokens
- Approximately \$0.04 per image at standard resolution

Performance Characteristics

Latency Comparison

- **Imagen:** Optimized for near real-time generation
- **Gemini Native:** Higher processing time due to advanced capabilities

Quality Trade-offs

- **Imagen:** Maximum visual fidelity
- **Gemini Native:** Flexibility over pure image quality

Integration Considerations

*An **API (Application Programming Interface)** is a set of rules and tools that lets different software programs communicate and share functions or data with each other (through code).*

Application Programming Interface (API) Access

The Gemini API is Google's developer interface that provides access to its Gemini family of AI models, enabling apps and services to use text, image, and code generation, as well as multimodal reasoning, through simple API calls (which are done with code).

Both models accessible through Gemini API with different endpoints and parameters.

Later on this book, we will learn how to use the Gemini API via the Python programming language!

Workflow Integration

- **Imagen:** Best for single-shot high-quality generation
- **Gemini Native:** Optimal for iterative creative workflows

The choice between Imagen and Gemini Native Image depends on specific project requirements: use Imagen for maximum quality and speed, choose Gemini Native for flexible conversational editing and complex multi-image operations.

What are Nano Banana's Advanced Capabilities?

When editing photos of familiar people or pets, subtle inaccuracies create noticeable problems. A face that's "close but not quite right" feels wrong to viewers who know the subject well.

Google's latest Gemini update "Nano Banana" addresses this challenge by maintaining consistent identity while enabling dramatic transformations.

Identity-Preserving Transformations

Gemini's core advancement lies in keeping subjects recognizable regardless of changes. Whether adding a 1960s beehive hairstyle or dressing a chihuahua in a tutu, the system preserves the essential characteristics that make someone look like themselves.

Costume and Location Changes

You can upload photos and place subjects in entirely new scenarios while maintaining your authentic appearance:

- Different clothing styles and professional outfits
- Historical time periods and decades
- Various locations worldwide
- Career or lifestyle changes

Type	Identity Preservation	Example Applications
Wardrobe changes	Facial features, body proportions	Professional headshots, costume trials
Time period styling	Core appearance traits	Historical recreations, vintage looks
Location placement	Subject characteristics	Travel photography, environmental contexts

The system maintains consistent appearance across all generated variations, ensuring recognizability remains intact.

Photo Combination Features

Gemini combines multiple photographs into cohesive new scenes. Users can merge separate photos of themselves and their pets, creating unified portraits in any setting. A common example involves combining a person's photo with their dog's photo to show both subjects together on a basketball court.

Blending Capabilities

- Seamless integration of subjects from different source images
- Contextual placement within new environments
- Consistent lighting and perspective matching
- Natural interaction positioning between subjects

Environmental Integration

The system places combined subjects appropriately within new backgrounds, handling:

- Scale relationships between different subjects
- Lighting consistency across all elements
- Shadow generation and placement
- Atmospheric perspective matching

Iterative Editing Process

Gemini supports progressive editing where you can continuously modify images while preserving previous changes. The workflow enables step-by-step scene construction:

1. Start with base image (empty room)
2. Add first modification (paint walls)
3. Include furniture elements (bookshelf)
4. Add additional items (coffee table)
5. Adjust details as needed

Selective Preservation

- Maintains specific areas while modifying others
- Preserves successful elements during subsequent edits
- Allows experimentation without losing progress
- Enables fine-tuning of individual components

Style Transfer Capabilities

You can extract visual characteristics from one image and apply them to objects in another image:

Transfer Examples

- Flower petal colors and textures applied to rainboots
- Butterfly wing patterns transferred to dress designs
- Architectural element styles applied to furniture
- Natural textures mapped onto manufactured objects

Source Element	Target Object	Result
Flower petals	Rainboots	Botanical-textured footwear
Butterfly wings	Dress fabric	Pattern-matched clothing
Tree bark	Furniture surface	Natural texture application

Content Authenticity Measures

All images created or edited in Gemini include two types of identification:

Visible Watermarks

- Clear indicators showing AI generation
- Persistent identification across sharing platforms
- Standardized marking for immediate recognition

SynthID Digital Watermarks

- Invisible embedded signatures within image data
- Detectable through specialized analysis tools
- Resistant to common image modifications

- Provides verification of AI generation source

SynthID is Google DeepMind's tool for embedding and detecting invisible digital watermarks in AI-generated content (like images, audio, or text), helping identify synthetic ("generated") media while remaining imperceptible without software.

Global Availability

These enhanced image editing capabilities are available to both paid and unpaid users worldwide through the Gemini app. The watermarking system ensures transparency about AI-generated content while enabling creative expression.

Video Generation Extension

Edited images can be uploaded back into Gemini to create videos, extending static modifications into dynamic content. This capability transforms single edited photographs into animated sequences or video content.

The system represents significant advancement in making AI image editing feel natural and authentic, particularly for images of familiar subjects where subtle inaccuracies would be immediately noticeable.

Advanced Consistency & Semantic Understanding in Nano Banana

Gemini 2.5 Flash Image ("Nano Banana") addresses core challenges in AI image generation: maintaining subject consistency across multiple edits and applying deep semantic understanding to visual transformations.

Character and Object Consistency

The system maintains identical subject appearance across different prompts and environments!

Use Case	Application	Consistency Elements
Character placement	Same person in multiple locations	Facial features, body proportions, clothing details

Product photography	Single item from various angles	Shape, texture, branding, dimensions
Brand asset generation	Logo/character across materials	Colors, typography, design elements

Multi-Environment Deployment

Users can place consistent subjects in:

- Different backgrounds and settings
- Various lighting conditions
- Multiple viewing angles
- Diverse contextual scenarios



Visual Template Adherence

The model follows design templates while customizing content:

Commercial Applications

- Real estate listing cards with standardized layouts
- Employee badges with uniform formatting
- Product catalog mockups maintaining brand consistency
- Marketing materials following brand guidelines



Template Elements

- Layout structure and positioning
- Typography and font specifications
- Color schemes and brand palettes
- Logo placement and sizing requirements



Targeted Local Editing

Natural Language Modifications

The system performs precise edits through conversational prompts:

Edit Type	Example Command	Technical Implementation
Background manipulation	"Blur the background", "Remove the background"	Selective depth processing
Object removal	"Remove the stain from t-shirt"	Inpainting with context awareness
Subject extraction	"Remove person from photo", "Remove mammoth from the car"	Object segmentation and background fill
Pose alteration	"Change the subject's pose"	Body geometry modification
Colorization	"Add color to black and white photo"	Historical color mapping

Precision Control

Users can target specific image regions without affecting surrounding areas. The system understands spatial relationships and applies modifications selectively.

Semantic World Knowledge Integration

Unlike purely aesthetic models, Gemini 2.5 Flash Image ("Nano Banana") applies real-world knowledge:

Knowledge Applications

- Physics-based interactions (gravity, shadows, reflections)
- Cultural context understanding (appropriate clothing, settings)
- Temporal awareness (historical accuracy, seasonal elements)
- Social context recognition (professional vs. casual environments)

Contextual Intelligence

The model makes informed decisions about:

- Appropriate lighting for different times of day
- Realistic material properties and textures
- Cultural and social appropriateness
- Environmental consistency and logic

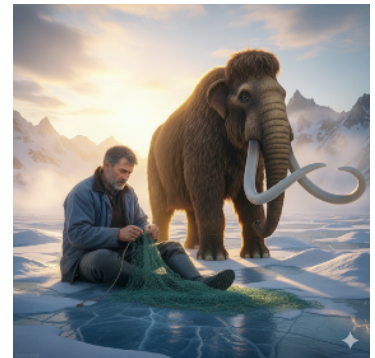
Multi-Image Processing

Fusion Operations

- Object placement into new scenes
- Room restyling with color schemes or textures
- Multi-source image combination
- Style transfer between images

Integration Process

1. Analyze multiple input images
2. Extract relevant elements from each source



3. Harmonize lighting, scale, and perspective
4. Generate unified output image

Source 1	Source 2	Operation	Result
Room interior	Color palette	Restyle walls	Room with new color scheme
Person photo	Background scene	Object placement	Person in new environment
Product image	Lifestyle setting	Context integration	Product in use scenario

Technical Capabilities

Processing Efficiency

- Single-prompt multi-image processing
- Real-time local edit preview
- Batch template application
- Consistent output across iterations

Quality Maintenance

- High-resolution output preservation
- Detail retention during modifications
- Minimal artifact introduction
- Consistent style application

Commercial Applications

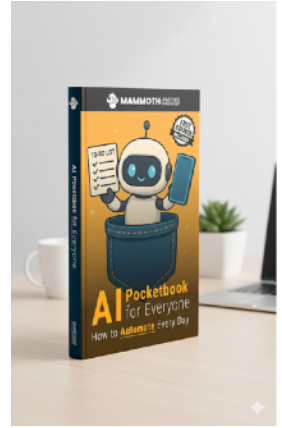
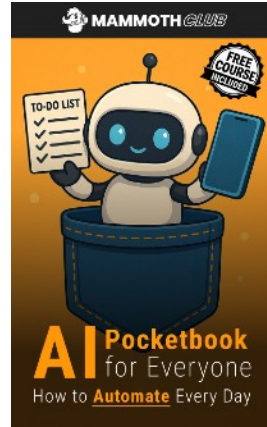
Business Use Cases

- E-commerce product visualization such as "make a product mockup for this book"
- Real estate marketing materials
- Corporate branding consistency
- Marketing campaign asset generation

Creative Workflows

- Character design consistency
- Brand identity development
- Template-based content creation
- Multi-angle product showcases

The system combines technical precision with semantic understanding, enabling both creative expression and commercial applications while maintaining consistency and quality across all outputs.



PART 2: Best Practices for Prompting Images

To elevate your outputs from good to exceptional, integrate these strategies into your workflow.

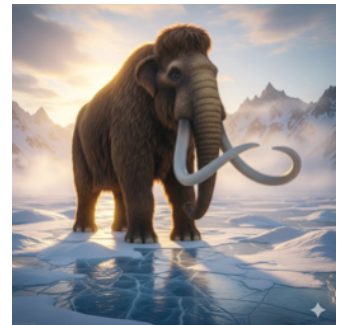
1. Be Hyper-Specific

Why it matters: Vague prompts leave too much up to chance. The more detail you provide, the closer the result will match your vision.

Example:

Poor: "wooly mammoth"

Better: "A massive woolly mammoth with shaggy brown"



fur and curved ivory tusks, standing on a frozen tundra.

The ground is covered in cracked ice with patches of snow.

Warm golden light from a low sunset reflects on the mammoth's fur, while a cold mist drifts across the background mountains. The atmosphere feels both majestic and ancient."

Tip: Specify colors, textures, shapes, lighting, and atmosphere.

2. Provide Context and Intent

Why it matters: The model adapts based on purpose. Defining the use case changes the design approach.

Example:

Poor: "Create a logo"

Better:

"Design a minimalist logo for a modern tech startup.

Use bold geometric shapes with clean lines, a navy blue and silver color palette, and a sleek sans-serif font.

The logo should convey innovation, trust, and forward-thinking energy."

Tip: Always mention who the image is for and how it will be used.

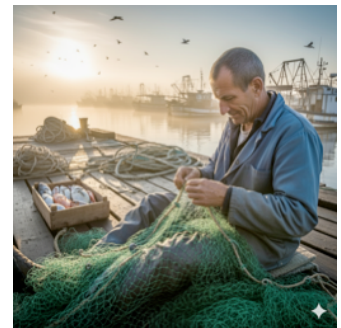


3. Iterate and Refine

Why it matters: Rarely is the first attempt perfect. Iteration improves quality and precision.

Example refinements:

- *"Good, but make the lighting a lot warmer"*
- *"Keep everything the same, but adjust the character's expression to be more happy."*



Tip: Treat the model like a collaborator—give clear, incremental feedback.

4. Use Step-by-Step Instructions

Why it matters: Complex scenes benefit from being built in stages.

Example:

1. *"Create an image with a background of an ocean sunrise at dawn."*
2. *"In the foreground, add two large coconut trees."*
3. *"Place a single hammock hanging between the trees."*

Tip: Think like a director building a scene layer by layer.



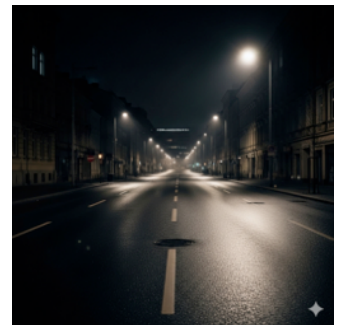
5. Reframe Negatives Into Positives

Why it matters: Saying "no X" often fails. It is more effective to describe the scene without the element.

Example:

- Poor: *"no cars"*
- Better: *"an empty, deserted street at night with no signs of traffic"*

Tip: Focus on what you want to see, not just what you don't want.



6. Control the Camera

Why it matters: Photographic and cinematic terms help guide perspective, framing, and detail.

Examples of camera terms:

- *Wide-angle shot* – expands scope
- *Macro shot* – highlights fine details
- *Low-angle perspective* – makes subject appear larger and more powerful
- *Overhead view* – provides clarity for layouts or designs

Tip: Imagine yourself holding a real camera—decide what type of shot you are directing.

Prompt Templates for Image Generation

Before we can edit images with AI, we need some images! You can source your own images or generate images with AI. First, let's look at how to generate images that we can later edit.

Successful image generation relies on descriptive narratives rather than keyword lists. The model's language comprehension excels with detailed scene descriptions that provide context, atmosphere, and specific visual elements.

Core Prompting Principle

Create scenes, not catalogs. Coherent paragraphs describing environments, subjects, and contexts produce superior results compared to disconnected terms. The model responds to storytelling that establishes mood, setting, and visual details.

Photorealistic Image Generation

Photography terminology guides the model toward realistic outputs. Include technical specifications: camera settings, lighting conditions, and compositional elements.

Template:

An ultra-realistic [shot type] showcasing [subject] while [performing action/ expression] within [environment].

The setup is lit by [lighting description], giving the scene a [mood] feeling.

*Captured through a [camera/lens details], highlighting [textures/details].
Delivered in [aspect ratio].*

Example:

"A sharp documentary photograph of a weathered fisherman mending nets at dawn on a wooden dock. His calloused hands work methodically while seagulls circle overhead.

The harbor scene features weathered fishing boats and coiled ropes.

Early morning mist creates atmospheric depth, with warm sunrise light casting long shadows across wet planks.

Shot with a 50mm lens for natural perspective, shallow depth of field isolating the subject.

The mood conveys decades of maritime experience and quiet determination."



Key Elements:

- Technical camera specifications
- Lighting quality and direction
- Environmental context and atmosphere
- Subject details and characteristics
- Compositional choices

Stylized Graphics and Icons

For decorative elements, specify art style explicitly and request transparent backgrounds for versatility.

Template:

Design a [style] sticker depicting [subject], emphasizing [key traits] and using a [color scheme]. The artwork should incorporate [line style] with [shading style]. Output must have a transparent background.

Example:

"A cheerful cartoon-style sticker featuring a smiling orange tabby cat wearing oversized sunglasses and a tiny Hawaiian shirt. The cat holds a miniature surfboard under one paw.

Art style uses thick black outlines, flat colors, and simplified forms typical of emoji design. Bright, saturated colors against a pure white background for easy removal."



Design Specifications:

- Art movement or style reference
- Color palette requirements
- Background treatment
- Outline and shading approach

Text Integration and Typography

The model handles text rendering effectively when given clear typographic direction and layout specifications.

When adding text to an image, generate the text first and then request the image with the text included for optimal results.

Template:

Generate a [image type] representing [brand/concept], including the phrase "[text to render]" styled in [font style]. The look should follow [style description] and use [color scheme].

Example:

"Design a vintage-inspired logo for 'Mountain Peak Brewery' featuring hand-drawn lettering in a rustic script font.



Include a simple line art illustration of snow-capped peaks above the text.

The design uses deep forest green and cream colors on a textured paper background.

Layout balances decorative elements with readable typography."

Typography Elements:

- Font style descriptions
- Text hierarchy and sizing
- Color scheme specifications
- Integration with visual elements

Commercial Product Photography

Professional product visualization requires studio lighting terminology and compositional guidance.

Template:

Capture a crisp, studio-quality image of [product description] placed on [background surface/description]. The scene is illuminated with a [lighting setup] designed to [purpose].

Shot from a [camera angle] to emphasize [specific feature], ensuring sharp clarity on [key detail]. Final output in [aspect ratio].

Example:

"A pristine e-commerce product shot of a sleek titanium water bottle on a gradient white-to-gray backdrop.

Professional two-light setup creates even illumination without hotspots. The bottle sits at a slight angle showing both front label and subtle curve.

Crisp focus throughout with subtle reflections on the polished surface.



Clean, modern aesthetic suitable for online retail."

Commercial Standards:

- Lighting setup descriptions
- Background and surface materials
- Product positioning and angles
- Brand presentation requirements

Minimalist Design Compositions

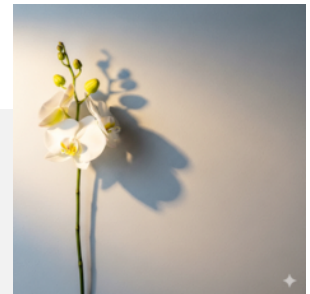
Negative space designs require careful balance specifications and restraint in visual elements.

Template:

A clean minimalist visual with one [subject] placed in the [chosen position] of the frame. The backdrop is a broad, uncluttered [color] space, enhancing negative space. Lighting is soft and understated. Render in [aspect ratio].

Example:

"A serene minimal composition with a single white orchid stem positioned along the left third of the frame. The background is soft pearl gray with subtle texture.



Gentle side lighting creates delicate shadows. Vast empty space on the right for text overlay. The overall feeling is elegant and spacious, perfect for luxury brand materials."

Minimalist Principles:

- Strategic element placement
- Background simplicity
- Space allocation for text
- Lighting subtlety

Sequential Visual Storytelling

Comic and storyboard panels benefit from specific art style references and narrative context.

Template:

Draw a one-panel comic in [art style]. Foreground shows [character + action], while the background illustrates [setting details].

Include a [dialogue/caption box] containing "[Text]." Lighting conveys a [mood] tone. Output should be [aspect ratio].

Example:

"A manga-style comic panel showing a young inventor in her cluttered workshop, surrounded by mechanical gadgets and scattered blueprints.

She holds up a glowing crystal with excitement while her robot assistant looks on.

The art style features clean line work, screen tones for shading, and dynamic perspective. Speech bubble reads 'This changes everything!' Panel has a rectangular landscape format."



Storytelling Components:

- Art style and technique
- Character positioning and expressions
- Environmental storytelling details
- Text integration and placement

Prompt Structure Best Practices

Element	Purpose	Implementation
Scene setting	Context establishment	Lead with environment description
Subject details	Character/object focus	Specific physical characteristics
Technical specs	Style guidance	Camera, lighting, artistic technique
Mood/ atmosphere	Emotional tone	Descriptive language for feeling

Effective prompts combine technical precision with creative storytelling, guiding the model through both mechanical requirements and artistic vision.

Best Practices for Image Editing Prompts

Image editing with AI requires specific prompt structures that combine visual references with clear modification instructions. These techniques maintain original image qualities while implementing precise changes.

Element Addition and Removal

Provide source images with detailed change descriptions. The model preserves original lighting, perspective, and style characteristics.

Template:

Using this image of [subject], please [insert/eliminate/alter] [specific element] [location/position]. The modification should [integration description].

Example Applications:

"Using this image of a fisherman repairing his net on the dock at sunrise, insert a small wooden crate filled with freshly caught fish beside him on his left side."

The modification should blend seamlessly with the lighting and shadows of the early morning scene, ensuring the fish glisten naturally in the soft golden light."

"Using this kitchen interior photo, add a vase of sunflowers on the marble countertop near the window.

The flowers should match the warm morning lighting and cast realistic shadows on the surface."

"Remove the construction crane from this city skyline photograph while maintaining the natural cloud formation and building outlines behind where the crane was positioned."



Edit Type	Prompt Focus	Quality Preservation
Object addition	Placement and integration	Lighting, shadows, perspective
Element removal	Background reconstruction	Texture, continuity, depth
Modification	Change specifications	Style consistency, realism

Semantic Masking and Inpainting

Define precise editing boundaries through conversational descriptions. This technique isolates modifications to specific regions while protecting surrounding areas.

Template:

"In this image, modify only the [target element] to become [new description]. Preserve all other aspects including [specific preservation requirements]."

Targeted Editing Examples:

"In this image, modify only the fishing net to become a large rolled-up map with visible nautical markings and routes.

Preserve all other aspects including the fisherman's posture, clothing, dock setting, boats in the background, sunrise lighting, and overall atmosphere."

"Change only the car's color from red to deep blue while keeping the chrome details, reflections, and background parking lot completely unchanged."

"Replace just the book covers on the shelf with vintage leather-bound volumes. Keep the wooden shelf, lighting, and surrounding room elements identical."



Artistic Style Transfer

Transform image content into different artistic movements while maintaining compositional structure.

Template:

"Convert this [subject type] image into [artistic style]. Maintain the original layout but apply [style-specific elements]."

Style Transformation Examples:

"Convert this fisherman on the dock image into a watercolor painting. Maintain the original layout but apply soft brush textures, fluid color blending, and subtle gradients that emphasize the morning light and reflections on the water."

"Transform this landscape photograph into impressionist oil painting style. Keep the mountain and lake composition but render with visible brushstrokes, vibrant color mixing, and soft light effects."



"Recreate this portrait in the style of Japanese woodblock prints. Preserve facial features and pose but apply flat color areas, bold outlines, and traditional color palette."

Style Categories:

- Classical art movements (Renaissance, Baroque, Impressionism)
- Modern techniques (Cubism, Art Deco, Pop Art)
- Digital art styles (Pixel art, Vector illustration, 3D rendering)
- Cultural art forms (Manga, Traditional patterns, Folk art)

Multi-Image Composition

Combine elements from multiple source images into unified new scenes. Essential for product mockups and creative collages.

Template:

"Merge components from these images: take [element A] from image one and combine with [element B] from image two. Result should be [scene description]."

Composition Examples:

"Merge components from these images: take the fisherman sitting on the dock repairing his net from image one and combine with the woolly mammoth standing on the icy landscape from image two.

Result should be a surreal scene where the fisherman is seated on the frozen ground, mending his net beside the mammoth, with snowy mountains and icy reflections in the background, blending seamlessly into the cold environment."



"Combine the vintage armchair from the first photo with the modern apartment living room from the second photo. Create a cohesive interior where the chair fits naturally with the contemporary lighting and decor."

"Place the golden retriever from photo one into the autumn park setting from photo two. The dog should appear to be playing in the fallen leaves with natural shadows and proportions."

High-Fidelity Detail Preservation

Maintain critical visual elements (faces, logos, text) during complex edits through detailed specification.

Template:

"Using these images, place [element B] with [element A]. Ensure [element A's] [specific features] remain completely unaltered. The integration should [placement requirements]."

Precision Preservation Examples:

"Using these images, place the woolly mammoth (element B) with the fisherman on the dock repairing his net (element A). Ensure the fisherman's pose, clothing, net, and dock details remain completely unaltered."

The integration should position the mammoth standing just behind the fisherman on the dock, scaled proportionally and lit consistently with the sunrise atmosphere, creating a natural and cohesive composition."

"Place the company logo from image two onto the product packaging in image one. The logo's colors, proportions, and text must remain pixel-perfect while appearing naturally printed on the box surface."

"Add the architectural detail from the second building photo to the facade in the first photo. Preserve the original building's window alignment and stonework texture while seamlessly integrating the new ornamental element."

Advanced Editing Techniques

Include environmental context for realistic integration.

"Consider the time of day, weather conditions, and lighting direction when making this modification."

Perspective Matching

Ensure added elements follow original image geometry:

"Match the vanishing point and scale perspective of the original scene when placing new objects."

Material Properties

Specify how new elements should interact with existing surfaces:

"The added element should reflect the ambient lighting and cast appropriate shadows on surrounding surfaces."

These editing approaches work best when prompts combine clear modification goals with detailed preservation requirements, ensuring the model understands both what to change and what to protect.

PART 3: Use Cases of AI Image Editing

AI Edit Images for the Business Professional

AI image editing enables professional enhancement and brand consistency without traditional photography costs and scheduling constraints.

Professional Profile Optimization

Update existing professional photos through AI modifications:

Enhancement Type	Application	Business Impact
Background updates	LinkedIn, resume photos	Modern, consistent appearance
Outfit variations	Multiple professional looks	Versatile brand representation
Lighting correction	Enhanced facial features	Improved first impressions
Expression adjustment	Confident, approachable demeanor	Better networking results

Practical Applications:

- Refresh outdated headshots without new photography sessions
- Test different professional styles while maintaining authenticity
- Improve technical aspects: sharpness, posture, lighting quality

Prompt Template: Professional Headshot Editing

This is one of the most popular uses for AI image editing in a business context. You can turn a simple selfie into a high-quality, professional headshot.

Template:

Action: Change the background and refine the lighting.

Subject: The individual in the photo.

Description: Professional, clean, and modern look. The background should be a solid [Color, e.g., charcoal gray or a soft blue gradient].

Add soft, professional studio lighting with a subtle catchlight in the eyes. The expression should be confident and friendly.

Context: For a LinkedIn profile picture.



Examples:

Headshot Refinement:

"Change the background of this headshot to a clean, professional dark gray color. Smooth the lighting to mimic a studio portrait. Subtle skin softening and blemish removal. This is for my corporate bio."

Attire and Background Swap:

"Replace the casual shirt with a professional dark blue blazer. Change the background to a blurred, modern office environment. The pose and facial expression should remain the same."

Group Photo to Individual Headshot:

"Isolate the person on the far left. Remove the other people and the background. Place the subject against a solid white background with professional, even lighting. Keep a high-resolution quality for a company website."

Brand Consistency Development

Multi-Platform Consistency:

- Matching headshots for websites, speaker bios, social profiles
- Corporate color integration in personal branding photos
- Professional accessory addition (branded items, conference materials)

Brand Integration Examples:

- Company color schemes reflected in clothing or backgrounds
- Corporate logos or branded items naturally integrated
- Consistent styling across different professional contexts

Professional Accessory Integration

Add context-appropriate elements:

- Conference badges and industry identifiers
- Branded merchandise and corporate materials
- Professional tools and equipment relevant to expertise

Brand Style Guide Automation

Consistency is the backbone of strong branding. A **brand style guide** defines rules for logos, colors, typography, imagery, and tone of voice. Automating this guide with

Gemini (Nano Banana) AI Image Editing ensures every asset your team creates remains onbrand without repetitive manual enforcement.

Why Automate a Brand Style Guide?

- **Consistency at Scale:** Ensure brand colors, fonts, and styles are applied across thousands of images.
- **Speed:** Reduce manual review cycles by embedding rules into automated workflows.
- **Collaboration:** Empower nondesigners to create onbrand visuals easily.
- **Cost Efficiency:** Minimize design rework and maintain quality control.

Elements of a Brand Style Guide

Core Identity

- **Logos and Variations** → primary, secondary, monochrome.
- **Typography** → font families, weights, and hierarchy.
- **Color Palette** → HEX/RGB values, gradients, and contrast rules.
- **Imagery Style** → photography guidelines, composition, and filters.

Extended Identity

- **Iconography** → shapes, stroke width, alignment.
- **Tone of Voice** → captions, overlay text, and taglines.
- **Layout Rules** → margins, safe zones, alignment guides.

How Gemini (Nano Banana) Automates Branding

Gemini can enforce brand rules dynamically:

- **Color Correction:** Automatically adjust images to match approved brand palettes.
- **Logo Placement:** Insert watermarks or logos in consistent positions.
- **Typography Templates:** Apply predefined text overlays with correct fonts.
- **Image Filters:** Apply brandspecific filters or LUTs for consistent mood.
- **AI Validation:** Flag images that deviate from style guidelines.

Workflow Example: Automated Logo Placement

```
import google.generativeai as genai

model = genai.GenerativeModel("gemini-2.0-flash-001")

response = model.generate_content("Overlay the company logo in the
bottom right corner of this image with 20% opacity.")

print(response.text)
```

Integrating with Other Systems

Webhooks

- Trigger branding automation when new images are uploaded.

ThirdParty Tools (Zapier, Make.com)

- Route processed images into CMS, social media, or ad platforms automatically.

Database Integration

- Track metadata (color schemes, logo placement, version numbers).
- Enforce auditability and analytics on brand compliance.

Backup and Version Control

- Maintain a versioned library of brand assets.
- Roll back to prior branding rules if updates create inconsistencies.

Best Practices

- **Centralize the Guide:** Keep a single source of truth for brand rules.
- **Automate Validation:** Let AI check and approve assets before publishing.
- **Balance Automation & Creativity:** Enforce rules without stifling unique adaptations.
- **Update Regularly:** Revise style guides as brand identity evolves.

Advanced Strategies

Dynamic Brand Personalization

- Tailor brand styles for specific campaigns (e.g., seasonal variations).
- Use AI to autogenerate variants while keeping within brand constraints.

MultiChannel Optimization

- Automatically adjust layouts for different platforms (Instagram, LinkedIn, print).
- Ensure aspect ratios, font sizes, and logo placements adapt correctly.

AI Driven Compliance Reports

- Generate dashboards on brand adherence.
- Highlight offbrand assets for review.

WrapUp

Automating your brand style guide with Nano Banana AI ensures that every asset—whether created by designers, marketers, or AI systems—remains true to your brand identity. By blending **rule enforcement, workflow automation, and smart validation**, you gain both **efficiency and creative consistency** at scale.

Edit Marketing Images with Prompt Templates

Generate professional marketing assets from single source images:

Content Type	Traditional Approach	AI-Enhanced Approach
Product mockups	Multiple photo shoots	Human consistency across scenarios
Team photos	Coordinate group schedules	Individual photos combined seamlessly
Ad creatives	Expensive photography	Portrait variations from one session

Virtual Team Assembly:

- Create group photos without coordinating schedules
- Maintain individual authenticity while ensuring visual cohesion
- Generate multiple team arrangements for different marketing needs

Consistency Benefits:

- Same lighting and quality standards across all team members
- Uniform professional presentation regardless of photo timing
- Cost-effective scaling for growing teams

Product Integration

Human-Centric Marketing:

- Consistent model appearance across product variations
- Professional demonstration scenarios without multiple shoots
- Brand ambassador representation maintaining visual continuity

Scalability Advantages:

- Generate multiple marketing scenarios from limited source material
- Adapt existing professional photos to new product launches
- Create seasonal or campaign-specific variations efficiently

These applications reduce traditional photography costs while maintaining professional quality and brand consistency across all marketing materials.

Product Placement & Showcase

Purpose: To place a product into a new, aspirational environment without a photo shoot.

Template:

Change the background of the image to [new background].

Enhance the lighting to be [lighting style] to make the product stand out.

Ensure the [product] looks seamlessly integrated.

Examples:

Book

Change the background of the image to a vibrant, futuristic cityscape at night.

Enhance the lighting to be neon-infused to make the "1,001 Money-Making Prompts" product stand out.

Ensure the book cover looks seamlessly integrated into this new environment.

Skincare Product



Change the background of the image to a luxurious, brightly lit bathroom with marble countertops and a soft-focus window view.

Enhance the lighting to be soft and natural to make the lotion bottle glow. Ensure the bottle looks perfectly placed on the counter, with a slight reflection.

Tech Gadget

Change the background of the image to a modern, minimalist home office with natural light. Make the smart speaker look professional and high-tech. Ensure a subtle shadow is cast on the desk to ground the product.

Social Media Ads & Banners

Purpose: To make an image more dynamic, eye-catching, and suitable for a specific ad campaign.

Template:

Edit the image: change the mood to [mood]. Add a [visual element] to the background and apply a [visual effect]. Ensure the main subject remains the focal point.

Examples:

Book

Edit the image: change the mood to exciting and dynamic. Add a subtle, glowing network of interconnected lines and nodes to the background, and apply a subtle glitch effect. Ensure the main subject remains the focal point.

Fitness Brand Ad

Edit the image: change the mood to energetic and powerful. Add a trail of neon-blue light streaks in the background, as if the athlete is moving at high speed. Apply a slight motion blur effect to the background to emphasize speed. Ensure the athlete's face is sharp and clear.

Travel Agency Ad

Edit the image: change the mood to magical and dreamy. Add a subtle, shimmering particle effect in the sky. Make the colors more

saturated to appear like a beautiful sunset. The person in the foreground should look like they are in awe of the scene.

Team & Portrait Editing

Purpose: To create consistent, professional team photos or clean headshots for a website's "About Us" page.

Template:

Remove the background and replace it with a [new background]. Ensure the person is perfectly cropped and the new background is crisp. Apply a professional, soft lighting effect. Make the person's [specific feature] look [adjective].

Team Photo

Remove the cluttered office background and replace it with a simple, gradient blue backdrop. Ensure all five team members are cleanly separated from the background. Make the lighting on each person's face consistent and bright. Do not alter their features or expressions.

Seasonal & Thematic Campaigns

Purpose: To update a library of evergreen images for a holiday or seasonal campaign.

Template:

Transform the scene to look like [season/holiday]. Add [seasonal elements] and adjust the colors to a [color palette].

Book

Transform the scene to look like a festive winter holiday. Add a light dusting of snow on the trees and a few hanging snowflake ornaments. Make the colors a deep, festive red and gold palette. Add a soft glow from a hidden light source to create a warm atmosphere.



Summer Sale Banner

Transform the scene to look like a summer beach party. Add a subtle, warm sunlight glare and a few small beach balls. Adjust the colors to a vibrant, warm palette of yellows, oranges, and pinks.

Holiday E-commerce Banners

Transform the scene to look like a festive winter holiday. Add a light dusting of snow on the trees and a few hanging snowflake ornaments.

Make the colors a deep, festive red and gold palette. Add a soft glow from a hidden light source to create a warm atmosphere.

Image Editing Prompt Templates for Professionals

A strong prompt for AI image editing typically includes these components:

- **Action:** The specific task you want the AI to perform (e.g., "Change the background," "Remove the object," "Adjust the lighting").
- **Subject:** The main focus of the image (e.g., "the person in the foreground," "the product on the desk").
- **Description:** Detailed adjectives and phrases to describe the desired outcome. This is where you specify style, mood, color, and other visual details.
- **Context:** Any specific information that helps the AI understand the purpose of the image (e.g., "for a LinkedIn profile," "for a company website banner").

Here are some prompt templates and examples for common business professional use cases, from headshots to marketing materials.

E-commerce and Product Photography

Use AI to create polished product shots that are consistent with your brand.

Template:

Action: Edit this product photo.

Subject: The [Product Name, e.g., black leather wallet] on the [Surface Type, e.g., wooden table].

Description: The style should be minimalist and high-end. Change the surface to a white marble background with a clean, soft-shadow look. The lighting should be bright and diffused.

Context: For an e-commerce product page.

Examples:

Book:

Edit the image of the book cover to give it a **high-end, luxurious feel**. Place the book on a **sleek, white marble surface**. The lighting should be **bright and soft**, casting a subtle, clean shadow that highlights the book's edges.



Use a **minimalist style**, ensuring the book is the clear and primary focus of the shot. This image will be used for an e-commerce product page.

Product on a New Background:

"Remove the existing background and replace it with a clean, bright studio backdrop. The lighting should be natural and highlight the product's texture."

Adding Props and Context:

"Add a stack of business cards and a sleek fountain pen next to the laptop in the photo. The scene should look like a minimalist workspace. The lighting should be soft and natural, coming from a window."

Creative Product Concept:

"Place the product—a wireless headset—on a futuristic, glowing desk in a high-tech office. The style should be a cyberpunk aesthetic with dramatic lighting and neon accents. This is for an ad campaign."

Business and Team Photos

Create a consistent look for team photos, even if they were taken at different times or with different lighting.

Template:

Action: Unify the style of these team photos.

Subject: The individuals in each photo.

Description: Change the backgrounds to a consistent light gray color. Adjust the lighting on each person to be uniform and professional, eliminating any harsh shadows or color casts. Ensure the overall style is cohesive for the company website's "About Us" page.

Context: For a team profile on the company website.

Examples:

- **Consistent Background:** "Take these five individual headshots and give them a consistent background of a slightly blurred, clean office setting with an emphasis on natural light."
- **Formal Attire:** "For this group photo, change everyone's attire to a formal business suit. Keep their faces and poses the same, but update their clothing to match a corporate dress code."
- **Color Correction:** "Correct the color and lighting on this team photo to match our brand's warm, inviting aesthetic. Remove the background clutter and replace it with a clean, simple white wall."

Creative and Lifestyle AI Image Use Cases

AI image editing enables creative expression and lifestyle enhancement beyond professional applications, supporting personal branding, family memories, and artistic exploration.

Social Media and Digital Presence

Transform photos into various artistic styles for platform-specific content:

Style Type	Application	Platform Optimization
Cartoon/ animated	Instagram stories, TikTok profile	Playful, approachable branding
Painterly effects	Pinterest, artistic portfolios	Creative, sophisticated aesthetic
Cinematic looks	LinkedIn, professional social media	Polished, dynamic presentation

Background Customization

Replace or enhance backgrounds for engaging content:

- Travel-inspired settings for Instagram stories without actual travel
- Seasonal backdrops matching current campaigns or moods
- Brand-consistent environments across all social platforms

Maintain recognizable identity across multiple platforms while adapting to different contexts and seasonal themes.

Pet and Family Image Editing Prompts

Seasonal Styling:

- Holiday costumes and themed accessories
- Weather-appropriate outfits and settings
- Special occasion formal wear

Environmental Placement:

- Studio-quality backgrounds for amateur pet photos
- Adventure settings for indoor pets
- Professional portrait styling for social sharing

Family Portrait Innovation

Note - some countries have restrictions on generating or editing images with children.

Traditional Challenge	AI Solution	Result
Coordinating schedules	Individual photo combination	Complete family portraits
Consistent styling	Unified color schemes and lighting	Cohesive family branding
Multiple outfit options	Virtual wardrobe changes	Variety without multiple shoots

Creative Mashups

Combine subjects in imaginative scenarios:

- Family members with pets in exotic locations
- Multi-generational portraits with consistent styling
- Fantasy settings maintaining realistic subject appearance

General Enhancements

Purpose: To improve the overall quality of a photo by adjusting lighting, removing distractions, or changing the background to a more appealing setting.

- **Template:** Enhance the [subject] by [action, e.g., adjusting the lighting, sharpening the focus]. Change the background to [new background].
- **Example: Dog Portrait** Enhance the dog's portrait by removing the leash and sharpening the focus on its face. Change the cluttered backyard background to a lush, out-of-focus green field.

- **Example: Family Portrait** Enhance the family portrait by adding soft, natural lighting from the right. Remove the distracting car in the background and replace it with a clean, scenic park.

Mood and Atmosphere

Purpose: To change the overall feeling or time of day of a photo by adding stylistic elements like sunlight, rain, or a specific color palette.

- **Template:** Transform the scene to evoke a [mood, e.g., cozy, adventurous, magical]. Add [atmospheric element, e.g., a sunset glow, gentle snowfall, sparkling lights] and adjust the colors to a [color palette, e.g., warm, cool, vibrant] palette.
- **Example: Cozy Indoor Shot** Transform the scene to evoke a cozy, relaxed mood. Add a warm, subtle glow from a fireplace in the background and adjust the colors to a warm, soft palette.
- **Example: Adventurous Outdoor Shot** Transform the scene to evoke an adventurous, free-spirited mood. Add a dramatic sunset glow behind the subjects, and adjust the colors to a vibrant, warm palette of oranges and purples.

Creative and Whimsical Edits

Purpose: To add fantastical or artistic elements, turning a simple photo into a unique piece of art.

- **Template:** Apply a [art style, e.g., watercolor, oil painting, sketch] to the image. Add [fantastical element, e.g., fairy wings, a halo, a magical glow] to the subject.
- **Example: Pet as a Fantasy Character** Apply an oil painting style to the image of the cat. Add a tiny, glowing crown on its head and a shimmering magical light in the background.
- **Example: Child in a Fairy Tale** Apply a storybook illustration style to the photo of the child. Add a pair of delicate, translucent butterfly wings on their back and a few sparkling butterflies flying around them.

Compositional Edits

Purpose: To add or remove a subject, or to change the composition of the photo for a more balanced or complete look.

- **Template:** Seamlessly [add/remove] a [subject] to/from the photo. Ensure the scale, lighting, and shadow of the new element perfectly match the original scene.
- **Example: Adding a new pet** Seamlessly add a small calico cat sitting next to the sleeping dog on the couch. Ensure the lighting and the cat's texture match the existing scene perfectly.
- **Example: Removing a person** Seamlessly remove the person standing in the background of the family photo. Ensure the final background looks natural and untouched, as if the person was never there.

Artistic Experimentation

Combine multiple photographic or artistic styles in single images:

- Blend realistic photography with illustration elements
- Merge different time periods or aesthetic movements
- Create hybrid looks combining multiple artistic influences

Progressive artistic transformation through iterative editing:

1. Start with original portrait
2. Apply subtle stylistic elements
3. Enhance artistic direction progressively
4. Refine details and artistic cohesion
5. Finalize unique artistic interpretation

Alternate Universe Portraits

Create imaginative variations exploring different possibilities:

Creative Concepts:

- Historical period placement with accurate costume and setting
- Genre-specific interpretations (sci-fi, fantasy, noir)
- Career or lifestyle alternatives with appropriate contexts
- Cultural or geographical relocations with authentic details

Portfolio Development

Build cohesive artistic collections:

- Consistent artistic style across multiple portraits
- Thematic series exploring specific concepts
- Progressive skill demonstration through varied techniques
- Personal artistic identity development

These applications transform everyday photos into creative content, enabling artistic expression without traditional photography limitations or technical expertise requirements.

Advanced Composition Techniques for Professional Photography

Professional photography is not just about capturing subjects—it is about **crafting visual stories**. Composition plays a central role in guiding the viewer's eye, creating balance, and evoking emotion. With **Gemini (Nano Banana) AI Image Editing**, you can apply classical rules of composition while experimenting with AI-assisted creativity.

Why Composition Matters

- **Visual Impact:** Strong composition instantly elevates an image.
- **Clarity:** Helps isolate the subject from distractions.
- **Emotion:** Influences how viewers feel when engaging with a photo.
- **Professionalism:** Distinguishes polished work from snapshots.

Core Techniques

Rule of Thirds

- Divide the frame into a 3×3 grid.
- Place key elements along gridlines or intersections.
- Creates balance and natural focal points.

Leading Lines

- Use roads, rivers, fences, or architectural lines.
- Guide the viewer's eye toward the subject.

Symmetry and Patterns

- Leverage reflective surfaces or repetitive shapes.
- Break symmetry intentionally for dramatic tension.

Framing

- Use natural frames (arches, windows, foliage).
- Adds depth and context to the subject.

Depth and Layers

- Foreground, middle ground, and background add three-dimensional feel.
- Blur (bokeh) or selective focus enhances separation.

Advanced Approaches

Golden Ratio and Fibonacci Spiral

- Position elements along a spiral curve for natural harmony.
- Subtle, but pleasing to the eye.

Negative Space

- Embrace emptiness around the subject.
- Highlights minimalism and emphasizes the focal point.

Color Composition

- Use complementary or analogous colors for mood.
- Apply color grading to unify tone.

Perspective Play

- Experiment with low or high camera angles.
- Wideangle lenses exaggerate depth; telephoto lenses compress it.

Using Gemini (Nano Banana) for Composition

Gemini can **enhance and refine composition** through AI editing:

- **Crop Suggestions:** Automatically propose ruleofthirds or golden ratio crops.
- **Background Extension:** Expand a frame to adjust balance.
- **AI Masks:** Reposition or resize elements for better harmony.
- **Style Transfer:** Apply compositional cues from master photographers.

Python Example: AI Crop Suggestion

```
import google.generativeai as genai  
model = genai.GenerativeModel("gemini-2.0-flash-001")
```

```
response = model.generate_content("Suggest a crop using rule of thirds for this uploaded image.")  
print(response.text)
```

Best Practices

- **Plan in Camera:** Use AI for refinement, not a crutch.
- **Consistency:** Maintain a coherent compositional style across projects.
- **Experiment:** Use Gemini to explore unconventional framing or perspective.
- **Review Iteratively:** Compare AI-assisted edits with originals before finalizing.

WrapUp

Mastering advanced composition techniques helps you produce **professionalgrade photography**. With Nano Banana AI tools, you can combine timeless visual principles with modern AI flexibility—achieving results that are both artistically strong and technically precise.

Prompt Templates for Creative AI Images

Let's start by looking at some prompts for generating images, then prompts for editing those images and more.

Personal & Hobbyist Projects

AI is an amazing creative partner for personal projects, offering a way to visualize ideas without needing traditional artistic skills.

Use Cases:

- **Storyboarding and Character Design:** Quickly generate characters and scenes for a novel, comic book, or video game concept.
- **Artistic Exploration:** Experiment with different art styles, mediums, and compositions to find a unique personal aesthetic.
- **Personalized Decor:** Create custom artwork for your home, like a unique poster or canvas print.

- **Social Media Content:** Generate unique and eye-catching images for personal blogs, Instagram, or other social platforms.
- **T-Shirt/Merch Design:** Design unique graphics for custom apparel.

Prompt Examples:

Fantasy Character:

- "A detailed, full-body portrait of a young wizard with fiery red hair and a staff made of twisted wood, standing in a magical forest at night. Digital painting, dramatic lighting, high fantasy art style."

Abstract Art:

- "Generate a vibrant, abstract painting inspired by the sound of jazz music. Use bold, sweeping brushstrokes and a palette of deep blues, fiery oranges, and golden yellows. The composition should feel energetic and rhythmic."

Concept Art:

- "A cinematic, wide-angle shot of a futuristic city built on a giant floating rock in the sky. The city is illuminated by neon lights and holographic advertisements, with small flying vehicles zipping between buildings. Hyper-realistic, 8k resolution."

Personalized Decor:

- "A minimalist line art drawing of a house with a winding path leading up to the front door. The style is simple and elegant, with a soft, pastel color palette. This is for a living room print."

Photo Restoration and Enhancement

These prompts are for apps that specialize in fixing old or damaged photos.

- "Restore this vintage black and white photo, removing scratches and creases, and colorize it realistically."

- "Fix the faded colors in this family photo and sharpen the details of the faces."
- "Remove the large tear at the top left of the image and reconstruct the missing background."
- "Enhance this blurry low-light photo, reducing noise and making the subjects clearer."
- "Convert this sepia-toned portrait to vibrant, full color while preserving the classic look."

Creative and Stylized Portraits

These are for apps that transform a photo into a different artistic style.

- "Turn this portrait into a watercolor painting with soft, dreamy brushstrokes."
- "Create a stylized self-portrait in the style of a 1920s Art Deco illustration."
- "Convert this photo into a cartoon character with a bright, friendly expression."
- "Generate a cyberpunk-style portrait with neon lighting and subtle robotic elements."
- "Transform this image into a playful claymation-style character, with a hand-sculpted texture."

Object Removal, Replacement, and Addition

These prompts are for in-painting or out-painting features, where you can modify the contents of a photo.

- "Remove the person in the background and seamlessly fill the area with a continuation of the landscape."
- "Replace the old car in the driveway with a modern electric vehicle."
- "Add a festive party hat to the main subject of the photo."
- "Remove the unwanted power lines from the sky and leave the rest of the image untouched."

- "Change the background from a busy city street to a tranquil beach at sunset."

Interior Design and Visualization

AI can help you visualize home and office renovations, furniture arrangements, and interior design concepts, including:

- **Room Makeovers:** See how a room would look with new furniture, wall colors, or flooring.
- **Furniture and Decor Placement:** Visualize how a specific piece of furniture would fit into your existing space.
- **Architectural Concepting:** Generate mockups of different architectural styles for a new build or a remodel.

Prompt Examples:

Living Room Refresh:

- "Take this photo of my living room and replace the sofa with a mid-century modern velvet sofa in a rich forest green color. Change the wall color to a warm, off-white. The lighting should be soft and natural, coming from the large window."

Kitchen Renovation:

- "Visualize a modern kitchen with a minimalist aesthetic. The cabinets should be a matte black, the countertops a white marble, and the backsplash a simple subway tile. Add a single, large kitchen island with three minimalist pendant lights hanging above it."

Bohemian Patio:

- "Design a cozy outdoor patio space with a bohemian vibe. It should have a rattan egg chair, an abundance of lush green plants in terracotta pots, and a string of warm fairy lights draped overhead. The atmosphere should be warm and inviting."

Food and Culinary Arts

AI is great for creating mouth-watering food visuals for blogs, cookbooks, and social media, including:

- **Recipe Blogs:** Generate stunning hero images for recipes, showcasing the dish in a beautiful, styled setting.
- **Menu Design:** Create artistic representations of dishes for a restaurant's menu.
- **Cookbook Illustrations:** Design stylized illustrations of ingredients or culinary processes.

Prompt Examples:

Hero Image for a Recipe:

- "A close-up, top-down shot of a beautifully plated dish of creamy pasta with fresh basil and cherry tomatoes.
- The scene is lit by soft, natural window light. The style is professional food photography with a shallow depth of field."

Rustic Meal:

- "A rustic, farmhouse-style table setting with a steaming bowl of homemade bread and a bottle of wine.
- The atmosphere is warm and communal, with a cozy, dark aesthetic. The image should feel inviting and authentic."

Ingredient Focus:

- "A vibrant, artistic photo of freshly picked strawberries and blueberries spilled out on a weathered wooden table.
- The colors should be saturated and the lighting should highlight the texture of the fruit. This is for a farm-to-table blog."

Travel and Adventure

AI can help you create travel memories and dreamscapes, even if you don't have the perfect photo.

Use Cases:

- **Travel Blog Imagery:** Generate breathtaking landscapes or cityscapes to accompany travel stories.
- **Dream Vacation Visualization:** Create a mood board of images for a future trip to get inspired.
- **Artistic Travel Posters:** Design stylized, vintage-style travel posters for your favorite destinations.

Prompt Examples:

Serene Landscape:

- "A stunning, hyper-realistic photo of a serene mountain lake at sunrise. The water is perfectly still, reflecting the pastel colors of the sky and the snow-capped mountains. The lighting is golden hour, with a hint of mist rising from the water."

Bustling City Scene:

- "A high-angle, cinematic shot of a bustling street market in Marrakech, Morocco. The scene is full of vibrant colors, intricate textiles, and the warm glow of lanterns. The style is documentary photography, capturing the energy and culture."

Fantasy Travel Poster:

- "A vintage travel poster for the fictional destination of 'The Crystal Caves.' The poster should feature a hero standing in a giant cave with glowing blue crystals. The style is an old-fashioned lithograph, with bold typography and a retro feel."

Fashion and Clothing Design

These prompts help designers or consumers visualize new styles.

- "Design a bright red double-breasted wool coat with gold buttons, based on the provided sketch."
- "Show this model wearing a vintage-style floral dress, with a straw hat and basket bag."
- "Change the color of this t-shirt from blue to a vibrant emerald green."
- "Generate a variety of seamless patterns with a futuristic, geometric theme for a new clothing line."
- "Create a casual streetwear look on this person in an urban setting with a graffiti wall in the background."

Business and Media AI Image Applications

AI image editing enables scalable content creation for commercial and media applications, reducing production costs while maintaining professional quality.

E-Commerce and Retail

Transform single product shots into comprehensive marketing campaigns:

Application	Traditional Cost	AI Solution	Efficiency Gain
Multiple model shots	\$500-2000 per model	Single model, multiple variations	80-90% cost reduction
Seasonal campaigns	Full reshoot per season	Background/styling updates	70-85% time savings
Catalog consistency	Multiple photo sessions	Unified lighting and styling	90% consistency improvement

Scalable Campaign Development

Seasonal Variations:

- Summer/winter product presentations from single base images

- Holiday-themed backgrounds and styling
- Geographic market adaptations with cultural relevance

Model Diversity:

- Product placement on different body types and demographics
- Age-appropriate styling for target markets
- Lifestyle context variations without additional shoots

Lifestyle Imagery Creation

Generate consistent brand aesthetics across product lines while maintaining authentic human interaction with products.

Real Estate and Property Marketing

Virtual Staging Solutions

Property Challenge	Traditional Solution	AI Enhancement
Empty spaces	Physical staging (\$2000-5000)	Virtual furniture placement
Lighting issues	Professional photography	Day/night conversions
Agent consistency	Regular headshot updates	Consistent branding across listings

Property Presentation Enhancement

Environmental Optimization:

- Day-to-night exterior transformations
- Seasonal landscape variations
- Weather condition improvements

- Neighborhood context enhancement

Interior Staging:

- Furniture placement in empty rooms
- Decor style matching target demographics
- Space utilization optimization
- Color scheme testing

Marketing Consistency

Maintain agent branding across multiple property listings and marketing materials without frequent photo updates.

Real Estate Photography Prompt Examples

These examples are for AI tools that help real estate agents enhance property photos.

- "Remove the clutter from the kitchen counter and stage the space with a fruit bowl and a single plant."
- "Virtually stage this empty living room with modern furniture and a cozy rug."
- "Replace the overcast sky with a bright, clear blue sky."
- "Remove the car parked in the driveway and repair the lawn underneath it."
- "Generate a twilight shot of the house with all the interior lights on, creating a warm and inviting look."

Media and Entertainment

Content Creation Efficiency

Media Need	Production Challenge	AI Solution
Storyboards	Character consistency across scenes	Recurring character generation
Album covers	Cohesive artistic vision	Style-consistent variations

Promotional
materials

Brand unity across campaigns

Unified visual identity

Character Development

Storyboard Creation:

- Consistent character appearance across multiple scenes
- Expression and pose variations while maintaining identity
- Environmental context changes with character preservation

Visual Storytelling:

- Scene-to-scene character continuity
- Costume and styling consistency
- Age progression or regression maintenance

Brand Asset Development

Promotional Consistency:

- Album artwork variations maintaining artistic coherence
- Poster designs with unified visual language
- Marketing materials with consistent character representation

Campaign Scalability:

- Multiple promotional images from single photo sessions
- Cross-platform content adaptation
- Seasonal or event-specific variations

These applications demonstrate AI's capacity to maintain visual consistency while enabling creative variations, supporting business scalability without proportional increases in production costs.

Business and Media AI Image Editing Prompt Templates

Core Principles for Business & Media Prompts:

1. **Be Specific:** Vague prompts lead to vague results.
2. **Brand Guidelines:** Incorporate color palettes, style guides, or desired moods.
3. **Target Audience:** Consider who will see the final image.
4. **Purpose:** Clearly state *why* the image is being edited (e.g., "for a LinkedIn ad," "for our company blog," "for a product launch email").
5. **Technical Details (if known):** Resolution, aspect ratio, file type (though often the AI handles this, it can be good context).

Product Image Enhancement & Contextualization

Goal: Create appealing product shots for e-commerce, marketing materials, or advertisements.

Template:

- Action: [Enhance/Replace background/Place product in scene/Adjust lighting]
- Subject: The [Specific Product Name/Description] in the image.

Description:

The final image should have a [Style, e.g., minimalist, luxurious, rustic, high-tech] feel. [Describe desired background/scene: e.g., "a clean, white studio background," "a blurred modern office environment," "a lush outdoor patio at sunset," "a futuristic laboratory."].

Ensure [Lighting description: e.g., "bright and even studio lighting," "soft, natural daylight," "dramatic, directional lighting"]. Highlight [Specific product features, e.g., "the texture of the fabric," "the sleek design," "the vibrant color."].

Context:

For [E-commerce listing/Social media ad campaign/Catalog print/Website banner].
Keep the image [Aspect Ratio, e.g., square for Instagram, 16:9 for website header].

Examples:

E-commerce Product:

- "Replace the background of this image of our new ergonomic office chair with a clean, light grey studio background. Ensure bright, even studio lighting that highlights the chair's mesh back and chrome accents. This is for an Amazon product listing. Keep it a square aspect ratio."

Lifestyle Product Shot:

- "Place the 'HydroFlask' water bottle from the image into a scene of someone hiking on a trail in North Vancouver, British Columbia. The style should be vibrant and adventurous, with natural sunlight. Focus on the product being actively used. This is for a social media campaign promoting outdoor activities."

Tech Product Unboxing:

- "Edit this photo of our new smartphone. Change the background to a blurred, high-tech desk environment with subtle, futuristic glow effects. Ensure the phone's screen is illuminated with our app's UI shown clearly. The lighting should be soft but dynamic. This is for a product launch email."

Professional Headshot & Portrait Refinement

- **Goal:** Create consistent, professional headshots for team pages, LinkedIn, or corporate communications.

Template:

- Action: [Refine/Standardize/Change background/Adjust lighting and color]
- Subject: The individual(s) in the photo.

Description:

- The style should be [Professional, approachable, corporate]. Change the background to [Solid color, e.g., our brand's deep blue, a subtle gradient, a blurred office setting, a clean white wall].
- Adjust the lighting to be [Soft and even, bright and engaging, natural daylight]. [Specify desired facial expression if applicable, e.g., "confident and friendly smile."].
- Ensure [Color correction for skin tones, subtle blemish removal, hair neatening]. Maintain a natural look.
- Context: For [Company website 'About Us' page/LinkedIn profile/Internal communications directory].

Examples:

Team Page Consistency:

- "Standardize these five individual headshots. Change all backgrounds to a consistent, slightly blurred modern office interior in North Vancouver.
- Adjust the lighting on each person to be bright and professional, as if from a studio flash. Ensure all subjects have a consistent color temperature and overall warmth. This is for our corporate website's 'Our Team' page."

LinkedIn Profile Update:

- "Refine this headshot for a LinkedIn profile. Smooth out any harsh shadows, especially under the eyes. Subtly brighten the overall image and ensure the background is a solid charcoal gray. The expression should remain confident and approachable."

Executive Portrait:

- "Adjust the lighting and color of this executive portrait to be more dramatic and cinematic, with a focus on the subject's face. The background should be subtly desaturated and blurred to emphasize the individual. Ensure a strong, professional presence. This is for an annual report cover."

Prompts to Generate Marketing Images

Goal: Generate compelling imagery for advertisements, social media campaigns, blog posts, or banners.

Template:

- Action: [Generate/Modify/Create a scene for]
- Subject: [Key element of the ad/campaign, e.g., "a person using our software," "a diverse group collaborating," "a cityscape"].

Description:

- The visual should evoke a sense of [Emotion/Feeling, e.g., innovation, trust, excitement, calm]. The style should be [Artistic style, e.g., minimalist vector illustration, photorealistic, abstract, modern corporate].
- Incorporate [Brand colors, specific iconography, desired mood, e.g., "our brand's teal and orange palette," "subtle tech-inspired motifs," "a warm and inviting glow."]. The composition should be [Dynamic, balanced, focused on a central point].
- Context: For [Instagram carousel ad/Facebook banner for lead generation/Blog post hero image about remote work/Company presentation slide].

Examples:

- **Software Ad:** "Generate a visual for an Instagram ad. The subject is a professional woman in her 30s, smiling, using a sleek laptop with our software's UI clearly visible on the screen. The background should be a bright, modern co-working space in New York. The overall feeling should be productivity and ease of use. Use our brand's light blue and white color scheme."
- **Blog Post Hero:** "Create a hero image for a blog post titled 'The Future of Hybrid Work.' The image should depict a diverse team seamlessly collaborating from different locations (some in a modern office, others from home offices). The style should be a modern, clean illustration with interconnected lines

representing communication. The color palette should be corporate yet approachable."

Craft engaging visuals for social media posts, ads, and banners.

Template:

Action: Generate an image for a social media post.

Subject: A professional person giving a presentation.

Description: The style should be an energetic, modern vector illustration. The person should be at a podium in a conference room with a large screen behind them displaying a sales chart. Use a bold and vibrant color palette.

Context: For a new blog post announcement on LinkedIn.

Examples:

Event Promotion:

"Design an eye-catching event flyer image. The subject is a speaker on stage. The style is bold and modern with a spotlight effect. Include ample negative space at the top for text overlay with the event title and date."

Infographic Element:

"Create a small illustration of a handshake between two professionals. The style is a flat, clean design with the primary brand colors: navy blue and orange. This is for a report on partnerships."

Branded Quote Graphic:

"Place the following quote text in a professional font over the image of a person at their desk: 'Success is a journey, not a destination.' The background image should be slightly desaturated and blurred to make the text stand out."

Adjusting Product Focus and Presentation

This template is great for ensuring your product stands out and looks its best. You can highlight specific features, change the angle, or even alter the product's state.

Prompt Template:

[Action, e.g., "Adjust," "Enhance," "Change"] the [Product Name/Description] in the image by [Specific Action, e.g., "making it brighter," "changing its angle," "adding a glow"]. The goal is to [Desired Outcome, e.g., "make it the focal point," "show its texture," "make it look more appealing"].

Examples:

- Enhance the coffee beans in the image by making them more vibrant. The goal is to make the coffee look fresh and aromatic.
- Adjust the angle of the running shoe in the photo so the sole is clearly visible. The goal is to highlight the new grip technology.
- Change the position of the smartphone in the shot to be held by a person. The goal is to make it look more natural and user-friendly.

Setting the Scene and Context

Use this template to change the background or add elements that place your product in a specific environment or context. This is crucial for lifestyle marketing.

Prompt Template:

[Action, e.g., "Place," "Integrate," "Add"] the [Product Name/Description] into a [New Scene/Environment, e.g., "cozy living room," "modern kitchen," "busy city street"]. The new scene should convey a feeling of [Desired Emotion/Vibe, e.g., "comfort," "sophistication," "adventure"].

Examples:

Integrate the new travel backpack into a vibrant, sun-drenched beach scene. The new scene should convey a feeling of freedom and adventure.

Place the smart speaker on a clean, minimalist wooden desk. The new scene should convey a feeling of simplicity and modern design.

Add a few scattered books and a warm cup of tea to the scene with the blanket. The new scene should convey a feeling of coziness and relaxation.

Creating a Specific Visual Style or Mood

This template helps you align the image's overall aesthetic with your brand's identity. You can specify color palettes, lighting, and general artistic styles.

Prompt Template:

[Action, e.g., "Recolor," "Retouch," "Style"] the image of the [Product Name/Description] to have a [Specific Style, e.g., "warm, golden hour glow," "vibrant, high-contrast look," "soft, pastel color palette"]. The mood should be [Desired Mood, e.g., "energetic," "calm," "luxurious"].

Examples:

- Retouch the image of the skincare bottle to have a soft, ethereal glow. The mood should be tranquil and clean.
- Recolor the image of the fashion item to have a bold, high-contrast look with pops of neon. The mood should be energetic and playful.
- Style the image of the vintage camera to have a moody, cinematic feel with deep shadows and rich tones. The mood should be nostalgic and artistic.

Modifying Elements and Details

Use this template to make precise changes to specific parts of the image, like adding or removing objects, or altering text and logos.

Prompt Template:

[Action, e.g., "Remove," "Add," "Change"] the [Specific Element, e.g., "shadows," "text," "background object"]. For example, [Detailed instruction, e.g., "make the shadows softer," "add our logo in the corner," "remove the stray leaf"].

Examples:

- Remove the large tree from the background behind the car. The goal is to create a cleaner, more focused image.
- Add a subtle, realistic reflection of the product on the shiny surface it's on.

- Change the text on the t-shirt to read 'Live Bold.' Make sure the font is a clean sans-serif.

Generate Media Content & Editorial Illustrations

Goal: Provide engaging visuals for articles, news stories, and editorial pieces.

Template:

- Action: [Illustrate/Depict/Create a conceptual image for]
- Subject: [Core concept/Topic of the article, e.g., "economic growth," "cybersecurity threat," "sustainable energy"].

Description:

- The style should be [Editorial, conceptual, infographic-like, photorealistic, stylized illustration]. The image should convey [Specific message or metaphor, e.g., "rising trends," "interconnected global markets," "protection and defense"].
- Use a color palette that is [Neutral and informative, bold and attention-grabbing, harmonious with the article's tone]. The composition should be [Clear and easy to understand, thought-provoking].
- Context: For [News article on climate change impacts/Financial report cover/Educational infographic on AI ethics].

Examples:

News Article Visual:

"Illustrate the concept of 'economic inflation' for a news article. The image should feature a stylized graph pointing sharply upwards, with a small, struggling figure trying to hold down large, floating currency symbols.

The style should be a sophisticated editorial cartoon. Use a slightly muted, serious color palette."

Tech Review Graphic:

"Create a conceptual image for a review of a new virtual reality headset. The image should show a person wearing the headset, surrounded by a swirling vortex of abstract digital elements and glowing data lines, implying immersion. The style should be sleek, futuristic, and high-tech. This is for an online tech publication."

Environmental Report Cover:

"Depict the concept of 'sustainable urban development' for an environmental report cover.

The image should show a harmonious blend of green spaces, modern eco-friendly buildings, solar panels, and people cycling, all within a vibrant cityscape (like a future North Vancouver). The style should be photorealistic with a hopeful, bright aesthetic."

Product Photo Enhancements

This template is great for adjusting the look and feel of a product to make it more appealing on e-commerce sites or in print catalogs.

Prompt Template: "Enhance the existing product image of [Product Name]. Retouch the [Material/Texture] to look [Desired Quality, e.g., 'more high-end and matte,' 'richer and smoother']. Add [Specific Detail, e.g., 'a subtle glow,' 'crisp, white highlights'] and remove [Unwanted Element, e.g., 'glare,' 'wrinkles,' 'blemishes']."

Examples:

- **"Enhance the existing product image of the black leather jacket.**
Retouch the leather to look richer and smoother. Add crisp, white highlights on the seams and remove all wrinkles."
- **"Enhance the existing product image of the wireless headphones.**
Retouch the plastic to look more high-end and matte. Add a subtle glow around the logo and remove the distracting glare on the earcups."

Lifestyle Scene Adjustments

This template helps you modify an existing scene to better fit a marketing campaign, allowing you to change lighting, mood, and context.

Prompt Template: "Take the existing lifestyle photo of [Product/Subject] in a [Current Scene]. Adjust the lighting to have a [Desired Lighting Effect, e.g., 'soft, 'golden hour' glow,' 'bright, cinematic feel']. [Action, e.g., 'Soften,' 'Blur'] the [Part of Image, e.g., 'background,' 'shadows'] to [Desired Outcome, e.g., 'draw focus to the subject,' 'create a more relaxed atmosphere']."

Examples:

- **"Take the existing lifestyle photo of the coffee mug on the wooden table.** Adjust the lighting to have a soft, 'golden hour' glow, as if the photo was taken just after sunrise. Soften the background to draw focus to the mug, which should have a subtle steam effect added to it."
- **"Take the existing lifestyle photo of a runner on a trail.** Adjust the lighting to have a bright, cinematic feel with high contrast. Blur the background slightly to emphasize the motion and energy of the runner."

Editorial and Conceptual Edits

Use this template to transform a basic image into a more artistic or conceptual one, suitable for magazine covers, reports, or ads.

Prompt Template: "Use the existing photo of [Subject/Product] for a [Purpose, e.g., 'magazine cover,' 'tech review graphic']. [Action, e.g., 'Place,' 'Arrange,' 'Overlay'] the [Subject/Product] to create a [Desired Artistic Effect, e.g., 'dynamic, futuristic look,' 'sophisticated, minimalist feel']. The goal is to convey a sense of [Desired Mood/Concept, e.g., 'innovation and ease of use,' 'professionalism and power']."

Examples:

- **"Use the existing portrait of the CEO for a magazine cover.** Place her figure on the left side of the frame, creating space for text. Arrange the lighting

to cast a single, strong shadow behind her to create a more sophisticated and powerful feel. The goal is to convey professionalism and authority."

- **"Use the existing photo of a person at a computer screen for a tech review graphic.** Replace the generic stock images on the screen with a stylized graphic of a glowing data line. Overlay a subtle grid pattern on the image to create a dynamic, futuristic look. The goal is to convey a sense of innovation and ease of use."

AI Image App Development: Build an Empire

AI image editing streamlines software development workflows by generating consistent assets, reducing design bottlenecks, and enabling rapid prototyping across applications.

General UI Asset Prompting

Asset Type	Traditional Process	AI Solution
App icons	Design iterations, multiple sizes	Single concept to all formats
Profile avatars	Stock photos or custom shoots	Consistent character generation
Onboarding screens	Designer collaboration cycles	Direct prompt-to-asset

Consistent Visual Identity

Character Continuity:

- App mascots maintaining appearance across screens
- User avatar consistency throughout app experience
- Tutorial characters with identical styling

Brand Asset Scaling:

- Logo variations for different screen contexts
- Color scheme adaptations for accessibility
- Platform-specific icon requirements

Prototype Development

Generate placeholder content that maintains visual consistency during development phases without designer dependency.

The Formula: [Asset type] for a [product/niche]. Style: [visual style]. Key elements: [specific details].

- **Template:** A flat icon for a [finance app]. Style: minimalist, line art. Key elements: a stylized dollar sign, a simple bar chart.
- **Template:** A hero image for a [sustainable fashion website]. Style: ethereal, photographic. Key elements: a person wearing a flowing linen shirt, soft natural light, blurry background of a green meadow.

UI Icons

The Formula: [Icon name] icon for a [app/website]. Style: [visual style].

- **Template:** A "settings" icon for a [social media app]. Style: 3D, playful, emoji-like.
- **Template:** A "search" icon for an [e-commerce site]. Style: sleek, glowing, sci-fi.
- **Template:** A "cart" icon for a [grocery delivery service]. Style: pixel art, retro.

User Interface Layouts

The Formula: A [type of page/component] for a [product/niche]. Mood: [mood]. Details: [elements to include].

- **Template:** A login page for a [meditation app]. Mood: calm, serene. Details: a simple input field, a "Sign In" button, subtle background gradients, an illustration of a person meditating.
- **Template:** A hero section for a [gaming website]. Mood: epic, action-packed. Details: a large headline, a call-to-action button, a bold visual of a fantasy character in motion.

Illustrations & Graphics

The Formula: An illustration for a [product/feature]. Concept: [concept]. Style: [visual style].

- **Template:** An illustration for a [file sharing feature]. Concept: a group of people collaborating. Style: flat design, isometric, vibrant colors, clean lines.
- **Template:** A graphic for a "404 not found" page on a [travel site]. Concept: a person lost in a surreal landscape. Style: watercolor painting, whimsical, hand-drawn.

Game Development Image Editing Prompts

Character Design Efficiency

Asset Pipeline Optimization:

- Consistent character sprites across game states
- Clothing and accessory variations maintaining character identity
- Environmental placement testing with character consistency

Animation Preparation:

- Multiple pose generations for sprite sheets
- Facial expression variations for dialogue systems
- Equipment visualization on consistent character models

Environmental Assets

Game Element	Traditional Approach	AI Enhancement
Background variations	Multiple artist illustrations	Style-consistent scene generation
Item/weapon designs	Concept art iterations	Rapid prototype visualization
UI element themes	Designer-dependent updates	Automated style applications

Rapid Prototyping

Concept Validation:

- Quick visual mockups for gameplay mechanics
- Art style testing before full production
- Character design exploration without animation investment

Texture and Material Refinement

This template is ideal for enhancing the visual fidelity of in-game assets, making them more realistic or stylized.

Prompt Template: "Refine the existing game development image of the [Game Asset, e.g., 'character's armor,' 'ancient stone wall,' 'sci-fi weapon']. Adjust the [Material/Texture, e.g., 'metal,' 'stone,' 'wood'] to appear [Desired Quality, e.g., 'more battle-worn with subtle scratches,' 'smooth and polished with a high sheen,' 'rough and moss-covered']. Ensure [Specific Detail, e.g., 'reflections are accurate,' 'normal maps are more defined,' 'specular highlights are realistic']."

Examples:

- **"Refine the existing game development image of the knight's shield.**
Adjust the metal to appear more battle-worn with subtle scratches and dents.

Ensure reflections are accurate, showing a hint of the surrounding environment."

- **"Refine the existing game development image of the ancient stone wall.** Adjust the stone to appear rougher and more moss-covered. Ensure normal maps are more defined to enhance the depth and texture."

Lighting and Atmosphere Adjustment

Use this template to modify the mood and environment of a game scene, crucial for gameplay and cinematic moments.

Prompt Template: "Take the existing game development screenshot of the [Game Scene, e.g., 'forest clearing,' 'dark dungeon,' 'futuristic city street']. Adjust the lighting to create a [Desired Atmosphere, e.g., 'eerie and mysterious ambiance,' 'bright and inviting mood,' 'dark and gritty feel']. Introduce [Lighting Element, e.g., 'god rays,' 'volumetric fog,' 'glowing particles'] and modify [Color Palette/Overall Tone, e.g., 'to a cooler, desaturated palette,' 'with warm, vibrant hues']."

Examples:

- **"Take the existing game development screenshot of the forest clearing.** Adjust the lighting to create an eerie and mysterious ambiance with long, creeping shadows. Introduce subtle volumetric fog filtering through the trees, and shift the overall tone to a cooler, desaturated palette."
- **"Take the existing game development screenshot of the futuristic city street.** Adjust the lighting to create a bright and inviting mood with dynamic neon reflections. Introduce glowing particles around the lampposts and modify the color palette with warm, vibrant hues to emphasize the advanced technology."

User Interface (UI) / Heads-Up Display (HUD) Rework

This template focuses on refining the visual elements of a game's interface for better clarity, aesthetic appeal, or thematic consistency.

Prompt Template: "Rework the existing game development image of the [UI Element, e.g., 'health bar,' 'minimap,' 'inventory screen']. Change the [Visual Style, e.g., 'font to a more sci-fi aesthetic,' 'color scheme to match the game's dark fantasy theme,' 'layout to be more streamlined']. Add [Specific Graphic Element, e.g., 'subtle glow effects around active icons,' 'worn parchment texture to the background,' 'animated border']. Ensure [Desired Outcome, e.g., 'readability is maintained,' 'it integrates seamlessly with the game world']."

Examples:

- **"Rework the existing game development image of the health bar.** Change the visual style to match the game's dark fantasy theme, using a crimson and black color scheme. Add a subtle, pulsating glow effect around the health bar when it's low. Ensure readability is maintained even in fast-paced combat."
- **"Rework the existing game development image of the inventory screen.** Change the layout to be more streamlined and intuitive, with larger item slots. Add a worn parchment texture to the background and introduce subtle glow effects around active item selections to improve user experience."

Environmental Storytelling and Detail Addition

This template helps to enrich game environments by adding small, impactful details that tell a story or enhance immersion.

Prompt Template: "Take the existing game environment screenshot of the [Location, e.g., 'abandoned laboratory,' 'thriving marketplace,' 'crashed spaceship interior']. Add [Specific Detail, e.g., 'overgrown vines and rubble,' 'more diverse vendor stalls with unique goods,' 'sparkling wires and flickering emergency lights']. The goal is to [Desired Outcome, e.g., 'emphasize its desolation and age,' 'make it feel bustling and alive,' 'convey a sense of immediate danger']."

Examples:

- **"Take the existing game environment screenshot of the abandoned laboratory.** Add overgrown vines creeping across the walls and scattered

rubble on the floor. The goal is to emphasize its desolation and age, suggesting a long-forgotten tragedy."

- "Take the existing game environment screenshot of the thriving marketplace. Add more diverse vendor stalls with unique goods, and a few animated characters subtly interacting. The goal is to make it feel bustling and alive, inviting players to explore."

Website Development Applications

Dynamic Content Generation

User-Generated Content Enhancement:

- Profile photo consistency across user accounts
- Avatar generation for users without photos
- Background customization maintaining site aesthetics

Template Population:

- Consistent imagery across website sections
- Product showcase variations from single source
- Team page photos with unified styling

Responsive Design Assets

Screen	Asset Requirement	AI Solution
Mobile	Simplified compositions	Automatic layout adaptation
Tablet	Medium detail levels	Balanced element scaling
Desktop	Full detail display	High-resolution optimization

Marketing Integration

Landing Page Optimization:

- Hero image variations for A/B testing
- Seasonal campaign updates without photoshoots
- Industry-specific customization from base templates

Content Management:

- Blog post imagery matching article themes
- Social media assets maintaining brand consistency
- Newsletter graphics with unified styling

E-commerce Applications

Product Visualization:

- Lifestyle context for product photos
- Model consistency across product lines
- Seasonal catalog updates from existing inventory photos

These development applications reduce creative bottlenecks while maintaining professional quality standards across software projects.

Hero Image Optimization

This template is for editing the main image on a webpage, like a homepage or landing page banner. The goal is to make it visually impactful and correctly sized for web use.

Prompt Template: "Optimize the existing hero image of [Subject, e.g., 'a team collaborating,' 'a product on a table'] for the [Website Section, e.g., 'homepage banner,' 'product landing page'].

Crop the image to a [Specific Aspect Ratio, e.g., '16:9,' '1920x1080'] and ensure the [Focal Point, e.g., 'person's face,' 'product itself'] is clearly visible.

Reduce the file size for faster loading while maintaining image quality. The final image should have a subtle blur effect on the background to make overlaid text more readable."

Examples:

- **"Optimize the existing hero image of a team collaborating for the homepage banner.** Crop the image to a 16:9 aspect ratio and ensure the team members' faces are clearly visible. Reduce the file size to be under 200kb for faster loading, and add a subtle blur effect on the background to improve text readability."
- **"Optimize the existing hero image of a new smartphone on a table for the product landing page.** Crop the image to a 1920x1080 aspect ratio. Make sure the phone itself is the clear focal point and add a subtle glow around it to draw attention. Reduce the file size and compress the image for optimal web performance."

Background and Texture Integration

Use this template to modify images that serve as backgrounds or textural elements, ensuring they blend seamlessly with the website's design.

Prompt Template: "Edit the background image of [Subject, e.g., 'a cityscape,' 'abstract lines,' 'wood texture'] to seamlessly integrate with the website's design. Adjust the [Color/Tone, e.g., 'saturation,' 'brightness,' 'hue'] to match the site's color palette. Add a [Filter/Overlay, e.g., 'semi-transparent color overlay,' 'grainy film filter'] to unify its appearance. The image should be non-distracting, serving as a subtle backdrop for content."

Examples:

- **"Edit the background image of a cityscape to seamlessly integrate with the website's design.** Adjust the saturation to be more muted, and add a semi-transparent blue color overlay to match the site's primary color. The image should be subtle, not drawing attention away from the text."
- **"Edit the background image of a rough wood texture to seamlessly integrate with the website's design.** Reduce the brightness and add a grainy film filter to unify its appearance. This image should serve as a subtle, thematic backdrop for a blog post section."

User Interface (UI) and Icon Refinement

This template focuses on editing small, crucial images like icons, buttons, or thumbnails to ensure they are visually consistent and clear.

Prompt Template: "Refine the existing [UI Element, e.g., 'shopping cart icon,' 'user profile thumbnail,' 'social media icon set'] for the website's UI. Ensure the [Visual Style, e.g., 'line thickness,' 'color palette,' 'level of detail'] is consistent across all elements. The icons must be [Adjective, e.g., 'crisp and clear,' 'simple and legible'] at both their standard size and when scaled down."

Examples:

- **"Refine the existing shopping cart icon for the website's UI.** Ensure the line thickness and color palette are consistent with the rest of the site's icons. The final icon must be crisp and clear at both its standard size and when scaled down for mobile viewing."
- **"Refine the existing user profile thumbnail photos for the website's UI.** Change all the thumbnails to a circular format with a subtle 2-pixel white border. Ensure all images are uniform in size and have a slight glow effect on the border when hovered over."

Generate AI Apps with Prompt Templates

Note - these prompts are great to feed into the default Gemini model at gemini.google.com or Google AI Studio. The prompts will generate applications that use Nano Banana to edit images!

Basic Template

This template is great for getting a simple, functional app.

Create a [App Type] web application.

The app should have a clean, modern UI for:

- Uploading an image.

- A text input field for an AI prompt.
- A button to "Transform" or "Edit" the image.
- A display area to show the edited image.

AI Feature: Use `gemini-2.5-flash-image-preview` to perform the image-to-image transformation. The user's text prompt and uploaded image should be sent to the API.

UI Style: Make it look [minimalist, futuristic, vibrant, etc.]. Include a loading indicator while the AI is working.

Advanced Template

This template allows for more detailed, specialized applications.

Create a web application named "[App Name]".

Core Functionality: The main purpose of the app is to [describe the app's primary function, e.g., "change the background of a product image for e-commerce listings"].

AI Features:

- **Input:** A user uploads a single image and provides a text prompt.
- **Model:** Use the `gemini-2.5-flash-image-preview` model for image-to-image generation.
- **Prompting:** The app should automatically prepend a fixed instruction to the user's prompt, such as "Transform the uploaded image by...".

User Interface:

- **Layout:** [e.g., A two-column layout with input on the left and output on the right].
- **Components:** [List specific UI elements, e.g., "A file upload button," "A text area with a placeholder 'Describe your new background...'," "A button with a lightning bolt icon," "An area to display the original and edited images side-by-side"].

- **Visual Style:** [Describe the aesthetic, e.g., "Clean, professional, and responsive with a dark theme and rounded corners. Use Tailwind CSS for all styling."].

Technical Requirements:

- The application must be a single-file HTML, React, or Angular app.
- Include a loading state while the API call is in progress.
- Provide a clear error message if the API call fails.

Prompt Example: Generate an Artistic Filter App

App Description:

A single-file HTML web app called "Artistic Lens." The app lets a user upload a photo and apply an artistic filter using an AI model. The user can enter prompts like "make it a vibrant watercolor painting" or "transform it into a sketch." The UI should have a fun, playful style with vibrant colors and rounded buttons.

Model: gemini-2.5-flash-image-preview

Key User Prompt: Apply a filter that makes this photo look like a stained-glass window with bright colors and bold lines.

Prompt Example: Generate an AI Background Editor App

App Description:

A single-file React app that allows users to seamlessly edit the background of any photo. The user can either upload an image or take a photo, and the app will use an AI model to replace the background with a new scene based on a text prompt. The UI should be clean and minimalist, focusing on the editing process. It should include a button to save the final image.

Model: gemini-2.5-flash-image-preview

Key User Prompt: Change the background of this image to a vibrant, futuristic cityscape at night with neon lights.

Prompt Example: Generate a Vintage Photo Restorer App

App Description:

A single-file HTML web app called "Nostalgia Restore." The app acts as a personalized photo restorer. The user can upload an old, damaged, or black-and-white photo, and the app will generate a restored version by adding color, removing scratches, and enhancing clarity. The app should have a clean, vintage UI with clear headings and a classic color palette.

Model: gemini-2.5-flash-image-preview

Key User Prompt: Restore this photo, remove the scratches and tears, and colorize the image to look natural.

Prompt Example: Generate an AI Object Remover App

App Description:

A single-file HTML web app that takes a user's photo and a brief text prompt, then uses an AI model to remove unwanted objects or people from the scene. The app will act as a creative editing assistant. The UI should be elegant and have a serene, minimalist theme with smooth animations.

Model: gemini-2.5-flash-image-preview

Key User Prompt: Remove the person in the background and the trash can next to the bench. Make the scenery look natural.

Prompt Example: Generate an AI Image Upscaler App

App Description:

A single-file Angular app that allows a user to upscale a low-resolution image to a higher resolution while adding realistic detail. The user can describe the desired output, and the app will use an AI model to enhance the image quality. The UI should be professional and sleek, with a focus on quick and clean results.

Model: `gemini-2.5-flash-image-preview` or `imagen-3.0-generate-002`

Key User Prompt: Upscale this photo to 4K resolution. Enhance the details of the subject's face and make the colors more vibrant.

Prompt Example: Generate a Product Photography Editor App

Create a React web application named "Product Studio AI". Its core function is to change the background of a product image to a professional studio shot. The UI should be clean and minimalist. The user uploads an image, enters a prompt like "on a white marble surface with soft shadows," and clicks "Generate." The app should use `gemini-2.5-flash-image-preview` to perform the transformation and display the result.

Prompt Example: Generate an Artistic Filter App

Create a single-file HTML web app called "Artistic Lens." The app lets a user upload a photo and apply an artistic filter using an AI model. The user can enter prompts like "make it a vibrant watercolor painting" or "transform it into a sketch." The UI should have a fun, playful style with vibrant colors and rounded buttons. Use the `gemini-2.5-flash-image-preview` model for all the magic.

PART 4: Essentials of Python App Development

In this section, you'll learn the fundamentals of coding in Python and connecting to Google's Gemini SDK, which allows you to access Gemini models!

Run Python Code Online with Google Colab

What is Python?

Python is a beginner-friendly programming language widely used for web development, data science, artificial intelligence, automation, and more because of its simple syntax and powerful libraries. It has a minimalist, clean look that is easy and quick to read and nice to look at.

Don't be fooled - even though Python is beginner friendly, that doesn't mean it's only for beginner projects. In fact, Python is the #1 language that powers AI models. Remember - a model is just a system of code, so all your favourite models (including Nano Banana) are built with programming!

We'll look at how you can build simple applications from scratch to understand everything under the hood as an AI Engineer, how to generate entire apps with the click of a button (creating hundreds of lines of code) and more!

What is Colab?

Google Colab is a free, online code editor and compiler where you can write and run Python code to see results! That means you can build entire applications that use and incorporate AI models all without leaving a browser.

A compiler is what enables code to be run. A compiler is a program that translates source code written in a programming language into machine code (or another lower-level form) so a computer can execute it.

Colab provides free Python execution in the browser with pre-installed libraries and GPU access. That means you don't need to install anything on your computer. You can even access Colab from a mobile device!

*A **GPU** (Graphics Processing Unit) allows you to run code faster! This is especially useful for machine learning, which takes many loops of training, which means code has to run over and over again.*

All that code takes computing power to run (it's also why you have to pay for usage of machine learning model "tokens" - unless you're on a free app version).

For a technical definition: a GPU is a specialized processor designed for parallel computations, originally for rendering graphics but now widely used to accelerate tasks like machine learning, scientific simulations, and data processing.

Requirements for Colab

- Google account (such as your Gmail)
- Web browser (Chrome, Firefox, Safari, Edge) on a desktop or mobile device
- Internet connection (doesn't work offline)

Note - if you want to run Python offline, you can go to python.org in order to install Python (which includes its package manager "pip") to your computer. This also means you'll have to install a code editor and any packages required (like the Google Gemini SDK) to your computer, too.

Installing tools onto your computer (rather than in an online compiler like Colab) can throw errors ("bugs") and be a headache. That's why we recommend for beginners and prototypers to use an online tool like Colab, where you don't have to install anything to your computer.

Access Method for Colab

Navigate to colab.research.google.com and sign in with your Google account. We'll use Colab later on in this book for our coding examples!

Set Up a Colab Project

Creating New Notebook

1. Click "New notebook" or "File → New notebook", or use the default notebook

2. Rename notebook by clicking "Untitled0.ipynb"
3. Add title and description if needed

Note - all notebooks are stored in your Google Drive.

Code Cell Execution

```
# Click in code cell and type Python code  
print("Hello, Colab!")  
  
# Execute with Shift+Enter or click play button
```

Essential Features

Feature	Function	Benefit
Code cells	Execute Python code	Interactive development
Text cells	Markdown documentation	Project explanation
Runtime	Managed Python environment	No local setup required
File storage	Google Drive integration	Persistent storage

Pre-installed Libraries in Colab

A **library** is a collection of prewritten code (functions, classes, modules) that developers can reuse to perform common tasks without writing everything from scratch.

A **package** is a structured bundle of one or more libraries (plus metadata) that can be easily distributed, installed, and managed using package managers like **pip** in Python.

Colab includes common Python packages like Numpy, Pandas, Pyplot and Image, but other packages like Gemini must be installed.

```
# Already available - no installation needed but can be re-installed  
or upgraded
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
```

Install: You use a package manager (like `pip install numpy`) to **download and add** a library/package to your system so Python knows it exists.

Import: You use `import numpy` inside your code to **load and use** that already-installed library in your program.

Package Installation

A **package manager** is a tool that automates installing, updating, configuring, and removing software libraries or packages, making it easy for developers to manage dependencies (e.g., **pip** for Python).

In Colab, you can use the **!** (exclamation mark symbol) to enter the console, where packages are installed, upgraded or uninstalled.

A **console** is a text-based interface where you can type commands directly to interact with your computer or programming environment, often used for running code, managing files, or debugging programs.

Install additional packages with pip:

```
# Install new packages
!pip install google-genai
!pip install requests beautifulsoup4

# Upgrade existing packages
!pip install --upgrade tensorflow
```

File Management

Upload Files

```
from google.colab import files
```



```
uploaded = files.upload() # Opens file picker
```

Mount Google Drive

```
from google.colab import drive  
drive.mount('/content/drive')  
  
# Access files at /content/drive/MyDrive/
```

Download Files

```
files.download('output.png') # Downloads to local computer
```

Runtime Management

Runtime Types

- **CPU:** Free, basic processing
- **GPU:** Free with usage limits, faster for AI tasks
- **TPU:** Specialized for machine learning

Memory and Storage

- **RAM:** 12.7GB typical allocation
- **Disk:** ~25GB temporary storage
- **Session timeout:** 12 hours maximum

Store API keys securely:

```
from google.colab import userdata  
API_KEY = userdata.get('GOOGLE_API_KEY')
```

Add secrets via "Secrets" panel (key icon) in left sidebar.

Alternative Platforms

Platform	Cost	Features
Google Colab	Free/Pro (\$10/month)	GPU access, Drive integration
Jupyter Notebook	Free	Local installation required
Kaggle Notebooks	Free	Competitions, datasets
GitHub Codespaces	Usage-based	Full development environment

Best Practices

- Save work frequently (**Ctrl+S**)
- Use descriptive cell comments
- Install packages at notebook beginning
- Monitor RAM usage to avoid crashes
- Use GPU runtime only when needed

Colab provides the most accessible environment for running Python AI code without local setup requirements.

Set Up Python Project with Gemini SDK

An **SDK (Software Development Kit)** is a bundled set of tools, libraries, documentation, and example code that helps developers create applications for a specific platform, framework, or service. Google has a Gemini SDK to allow you to access Nano Banana and other models via code!

SDK Installation

Install the Google Gemini SDK using pip:

```
%pip install -U -q "google-gemini>=1.32.0"
```

Installation parameters:

- **-u**: Upgrade to latest version
- **-q**: Quiet mode (minimal output)
- **>=1.32.0**: Minimum required version for parts accessor functionality

API Key Configuration

Secure Key Storage

```
from google.colab import userdata  
  
GOOGLE_API_KEY = userdata.get('GOOGLE_API_KEY')
```

Google Colab approach:

- Stores API keys in encrypted Colab secrets
- Prevents accidental exposure in notebooks
- Provides secure access without hardcoding

Security Requirements

Risk	Protection Method	Implementation
Key exposure	Use userdata secrets	Never hardcode in scripts
Unauthorized usage	Monitor billing	Set up usage alerts
Accidental sharing	Environment variables	Keep keys out of version control

Critical security practices:

- Never commit API keys to repositories
- Set billing alerts and usage limits
- Revoke keys immediately if compromised

- Use separate keys for development/production

Client Initialization

```
from google import genai
from google.genai import types
client = genai.Client(api_key=GOOGLE_API_KEY)
```

Import requirements:

- **genai**: Core SDK functionality
- **types**: Type definitions for advanced features

Client configuration:

- Authenticates with Google's API
- Handles request/response formatting
- Manages connection and error handling

Alternative Setup Methods

Local Development

```
import os
GOOGLE_API_KEY = os.environ.get('GOOGLE_API_KEY')
client = genai.Client(api_key=GOOGLE_API_KEY)
```

Direct Key Assignment (Development Only)

WARNING: Only for local testing

```
GOOGLE_API_KEY = "your-api-key-here" # Never commit this!
```

A commit is a saved snapshot of your project's changes, along with a message describing what you did, allowing you to track and manage versions over time.

Billing Monitoring

Monitor usage to prevent unexpected charges:

- Set daily/monthly spending limits
- Configure billing alerts
- Review usage reports regularly
- Track API call frequency

This setup provides secure access to Gemini's image generation capabilities while protecting against unauthorized usage.

Python Coding Fundamentals for AI App Development

Programming languages, though often similar to one another, have varying rules and keywords that every developer should know. Let's dive into a crash course on Python, the #1 language for integrating Nano Banana and other models into your own applications.

Variables and Assignment

Variables store data using the `=` operator:

```
client = genai.Client()
prompt = "Create a picture of my cat"
```

Variables can hold different data types:

- **Strings:** Text in quotes ("**hello**" or '**hello**')
- **Objects:** Complex data structures
- **None:** Represents empty/null values

Import Statements

Import statements bring external code into your program:

```
from google import genai          # Import genai from google package
from PIL import Image             # Import Image from PIL package
from io import BytesIO            # Import BytesIO from io package
```

Syntax patterns:

- **from package import module** - Import specific module
- **import package** - Import entire package

Function and Method Calls

Functions perform actions using parentheses:

```
Image.open('/path/to/image.png')  # Function with parameter
genai.Client()                     # Function with no parameters
```

Method calls use dot notation:

```
object.method(parameters)
image.save("filename.png")
```

Lists and Data Structures

Square brackets create lists:

```
contents=[prompt, image]          # List containing two items
```

Object Attributes

Access object properties with dots:

```
response.candidates[0].content.parts  # Chain of attributes
part.text                             # Single attribute
part.inline_data.data                  # Nested attributes
```

Square brackets access list items:

- `[0]` = first item
- `[1]` = second item

Conditional Statements

if/elif statements check conditions:

```
if part.text is not None:           # If condition is true
    print(part.text)                # Execute this code
elif part.inline_data is not None:  # Else if this condition is true
    image = Image.open(...)         # Execute this code
```

Key operators:

- **is not None** - checks if value exists
- **==** - equals
- **!=** - not equals

Loops

for loops repeat actions:

```
for part in response.candidates[0].content.parts:
    # This code runs once for each 'part'
    print(part.text)
```

Loop structure:

- **for item in collection:** - iterate through each item
- Code inside loop must be indented

Indentation

Python uses indentation (spaces) to group code:

```
for part in response.candidates[0].content.parts:
    if part.text is not None:
        print(part.text)
    elif part.inline_data is not None:
        image = Image.open(BytesIO(part.inline_data.data))
        image.save("generated_image.png")
```

Rules:

- Same indentation level = same code block
- 4 spaces per indentation level (standard)
- Consistent indentation required

Pattern	Meaning	Example
object.method()	Call method on object	client.models.generate_content()
variable = value	Store value in variable	prompt = "text"
[item1, item2]	Create list	[prompt, image]
for x in y:	Loop through items	for part in response...
if condition:	Execute if true	if part.text is not None:

This foundation covers all syntax needed to understand AI SDK code examples.

PART 5: Let's build it! Generate Images with Python SDK

Access Nano Banana with Python

This example demonstrates image generation using Google's Gemini SDK, combining text prompts with input images to create new visual content. Let's look at the entire example first, then dive into each step!

```
from google import genai
from PIL import Image
from io import BytesIO
client = genai.Client()
prompt = "Wooly mammoth skiing"
image = Image.open('/path/to/image.png')
response = client.models.generate_content(
    model="gemini-2.5-flash-image-preview",
    contents=[prompt, image],
)
for part in response.candidates[0].content.parts:
    if part.text is not None:
        print(part.text)
    elif part.inline_data is not None:
        image = Image.open(BytesIO(part.inline_data.data))
        image.save("generated_image.png")
```

Import Statements

```
from google import genai          # Google's Gemini AI SDK
```

```
from PIL import Image          # Python Imaging Library for image
                                # handling
from io import BytesIO         # Handles binary data in memory
```

Library	Purpose
genai	Google's AI SDK for text and image generation
PIL.Image	Opens, manipulates, and saves image files
BytesIO	Converts binary image data to readable format

Client Setup

```
client = genai.Client()
```

Creates a connection to Google's Gemini API. This client handles authentication and API communication.

Input Preparation

```
prompt = "Woolly mammoth skiing"
image = Image.open('/path/to/image.png')
```

Prompt: Text description of desired image generation **Image:** Input image loaded from file path (replace **/path/to/image.png** with actual file location)

API Request

```
response = client.models.generate_content(
    model="gemini-2.5-flash-image-preview",
    contents=[prompt, image],
)
```

Parameters breakdown:

- **model1:** Specifies Gemini 2.5 Flash Image model

- **contents:** List containing text prompt and input image
- Returns API response with generated content

Response Processing

```
for part in response.candidates[0].content.parts:
    if part.text is not None:
        print(part.text)
    elif part.inline_data is not None:
        image = Image.open(BytesIO(part.inline_data.data))
        image.save("generated_image.png")
```

Response structure:

- **candidates[0]:** First (primary) generated result
- **content.parts:** List of response components
- Each part contains either text or image data

Processing logic:

1. Loop through each response part
2. If part contains text: print to console
3. If part contains image data: convert and save as PNG file

Data Flow

Step	Input	Process	Output
1	Text prompt + image file	Load inputs	Python objects
2	Prompt + image objects	API call	Raw response
3	API response	Parse response parts	Text/image data
4	Image binary data	Convert + save	PNG file

File Operations

Input: `Image.open('/path/to/image.png')` loads existing image file **Output:** `image.save("generated_image.png")` saves new image to disk

Replace file paths with actual locations on your system.

Error Handling Considerations

The code assumes successful API responses. Production code should include:

- File existence checks
- API error handling
- Network timeout management
- Invalid response format handling

This basic example demonstrates core SDK functionality for AI-powered image generation.

Generate Multiple Images for Storytelling

Generate sequential images at once for blog posts, stories, or documentation using just 1 prompt.

```
MODEL_ID = "gemini-2.5-flash-image-preview"

prompt = "Create a 6-step visual guide for planting a small garden:
preparing soil, planting seeds, watering, sprouting, growing, and
harvesting vegetables"

response = client.models.generate_content(
    model=MODEL_ID,
    contents=prompt,
)

display_response(response)
```

Model Variable Configuration

```
MODEL_ID = "gemini-2.5-flash-image-preview"
```

Using a variable for model names enables easy switching between different Gemini models without code changes throughout your project.

Sequential Content Prompts

Blog Article Example

```
prompt = "Show step-by-step coffee brewing process with 5 images:
grinding beans, heating water, pouring over filter, brewing, and
serving in cup"
```

Story Sequences

```
prompt = "Create a 7-part visual story about a robot learning to
paint: discovering art supplies, first awkward attempts, practicing
techniques, developing style, creating masterpiece, displaying
artwork, and teaching others"
```

Prompt Structure for Series

Element	Purpose	Example
Sequence count	Defines series length	"6-step guide", "8-part story"
Subject description	Core theme	"garden planting", "robot artist"
Step enumeration	Explicit sequence	"preparing, planting, watering..."
Visual consistency	Unified style	"consistent lighting", "same character"

Display Response Function

```
display_response(response)
```

This function handles multiple image outputs from the response object, typically displaying all generated images in sequence.

Alternative processing:

```
image_parts = [  
    part.inline_data.data  
    for part in response.candidates[0].content.parts  
    if part.inline_data  
]  
  
for i, image_data in enumerate(image_parts):  
    image = Image.open(BytesIO(image_data))  
    image.save(f'step_{i+1}.png')  
    image.show()
```

Long Output Considerations

Multiple image generation produces large responses:

- Longer processing times
- Higher token usage
- Increased memory requirements
- Multiple binary image files

Monitor billing usage carefully when generating image series.

Use Cases

- Educational tutorials and how-to guides
- Marketing campaign sequences
- Social media story content
- Product demonstration steps
- Narrative visual storytelling

Single prompts can generate cohesive multi-image sequences while maintaining visual consistency across the series.

Chat Mode for Iterative Image Generation

Chat mode enables iterative image editing through conversational turns, making it ideal for refining images progressively.

```
chat = client.chats.create(  
    model=MODEL_ID,  
)  
  
message = "Create an image of a wooden coffee table in a modern  
living room with natural lighting from large windows"  
  
response = chat.send_message(message)
```

```
display_response(response)
save_image(response, "coffee_table.png")
```

Chat Session Creation

```
chat = client.chats.create(model=MODEL_ID)
```

Creates persistent chat context that remembers previous images and modifications, enabling contextual follow-up requests.

Initial Image Generation

```
message = "Create an image of a wooden coffee table in a modern
living room with natural lighting from large windows"
response = chat.send_message(message)
display_response(response)
save_image(response, "coffee_table.png")
```

First message establishes:

- Base image concept
- Visual style parameters
- Environmental context
- Reference point for iterations

Iterative Modifications

```
message = "Add a stack of three books and a small potted succulent
plant on top of the coffee table"
response = chat.send_message(message)
display_response(response)
```

Follow-up messages can:

- Add new elements to existing scenes
- Modify specific details

- Change colors, materials, or lighting
- Remove unwanted objects

Save Image Function

```
save_image(response, "coffee_table.png")
```

Utility function that extracts image data from response and saves to specified filename.

Implementation:

```
def save_image(response, filename):  
    image_parts = [  
        part.inline_data.data  
        for part in response.candidates[0].content.parts  
        if part.inline_data  
    ]  
    if image_parts:  
        image = Image.open(BytesIO(image_parts[0]))  
        image.save(filename)
```

Feature	Unary Calls	Chat Mode
Context retention	None	Full conversation history
Iteration ease	Requires full re-description	Simple modification requests
Refinement process	Start over each time	Build incrementally
Conversation flow	Independent requests	Natural dialogue

A **unary** call is a simple request–response interaction where the client sends a single request to the server and gets back a single response.

Common Iteration Patterns

- **Progressive refinement:** Start basic, add details gradually
- **Style exploration:** Try different artistic approaches
- **Element testing:** Add/remove objects to find optimal composition
- **Color experimentation:** Test various color schemes

Chat mode transforms image generation from single-shot creation to collaborative design process.

PART 6: AI Image Editing with Python SDK

Add & Remove Image Elements with Gemini SDK

This technique adds or removes objects while preserving original image style, lighting, and perspective.

```
from google import genai
from google.genai import types
from PIL import Image
from io import BytesIO
client = genai.Client()

# Base image prompt: "A cozy living room with a brown leather sofa,
# hardwood floors, and large windows with afternoon sunlight streaming
# in."

image_input = Image.open('/path/to/your/living_room.jpg')

text_input = """Using this living room image, please add a tall
bookshelf filled with colorful books against the blank wall behind
the sofa. Match the warm lighting and ensure realistic shadows on
the floor."""

# Generate modified image
```

```
response = client.models.generate_content(
    model="gemini-2.5-flash-image-preview",
    contents=[text_input, image_input],
)
image_parts = [
    part.inline_data.data
    for part in response.candidates[0].content.parts
    if part.inline_data
]
if image_parts:
    image = Image.open(BytesIO(image_parts[0]))
    image.save('room_with_bookshelf.jpg')
    image.show()
```

Import Requirements

```
from google import genai          # Gemini AI SDK
from google.genai import types    # SDK type definitions
from PIL import Image             # Image file handling
from io import BytesIO            # Binary data processing
```

The **types** import provides additional SDK functionality for complex operations.

Input Preparation

```
image_input = Image.open('/path/to/your/living_room.jpg')
text_input = """Using this living room image, please add a tall
bookshelf..."""
```

Image loading: Replace file path with actual image location

Modification prompt: Describes specific changes while requesting style consistency

Modification Prompt Structure

Effective prompts include:

- Reference to source image
- Specific object to add/remove
- Integration instructions (lighting, shadows, positioning)
- Style preservation requirements

API Request

```
response = client.models.generate_content(  
    model="gemini-2.5-flash-image-preview",  
    contents=[text_input, image_input],  
)
```

Same structure as basic generation, combining text instructions with source image.

Response Processing

```
image_parts = [  
    part.inline_data.data  
    for part in response.candidates[0].content.parts  
    if part.inline_data  
]
```

List comprehension extracts only image data from response parts:

- Loops through all response parts
- Filters for parts containing **inline_data**
- Collects binary image data

Image Output

```
if image_parts:
    image = Image.open(BytesIO(image_parts[0]))
    image.save('room_with_bookshelf.jpg')
    image.show()
```

Processing steps:

1. Check if image data exists
2. Convert binary data to PIL Image object
3. Save to file with descriptive name
4. Display image on screen

The model maintains original image characteristics while implementing requested changes.

Operation	Prompt Pattern	Example
Add object	"add [object] to [location]"	"add potted plant beside window"
Remove element	"remove [object] from image"	"remove lamp from corner table"
Replace item	"replace [object] with [new object]"	"replace wooden chair with red armchair"

Inpaint & Mask with Gemini SDK

Inpainting enables precise editing of specific image regions while preserving surrounding elements. The model identifies target objects conversationally without manual masking.

```
from google import genai
from google.genai import types
```

```
from PIL import Image
from io import BytesIO
client = genai.Client()

# Base image prompt: "A sunny parking lot with several cars,
# featuring a bright red sedan parked between two white SUVs. Clear
# blue sky and concrete pavement visible."
parking_lot_image = Image.open('/path/to/your/parking_lot.jpg')

text_input = """Using this parking lot image, change only the red
sedan to be a deep blue metallic sedan. Keep everything else
identical - the white SUVs, pavement, lighting, and background
should remain completely unchanged."""

# Generate modified image
response = client.models.generate_content(
    model="gemini-2.5-flash-image-preview",
    contents=[parking_lot_image, text_input],
)

image_parts = [
    part.inline_data.data
    for part in response.candidates[0].content.parts
    if part.inline_data
]

if image_parts:
    image = Image.open(BytesIO(image_parts[0]))
    image.save('parking_lot_blue_car.jpg')
    image.show()
```

Conversational Masking

```
text_input = """Using this parking lot image, change only the red sedan to be a deep blue metallic sedan. Keep everything else identical..."""
```

Key elements:

- Identifies target object ("red sedan")
- Specifies exact change ("deep blue metallic")
- Explicitly preserves surroundings ("keep everything else identical")

Targeting Method	Example	Effectiveness
Color identification	"the red sedan"	High for distinct colors
Position reference	"car in the center"	Good for clear positioning
Object description	"sedan between two SUVs"	Excellent for context

Preservation Instructions

Effective prompts specify what to keep unchanged:

- Surrounding objects
- Lighting conditions
- Background elements
- Surface textures

Contents Order

```
contents=[parking_lot_image, text_input]
```

Image and text can be in either order - the model processes both inputs together.

Processing Results

The list comprehension extracts only image data:

```
image_parts = [  
    part.inline_data.data  
    for part in response.candidates[0].content.parts  
    if part.inline_data  
]
```

Filters response parts to collect binary image data, ignoring any text responses.

Common Use Cases

- Color changes on specific objects
- Clothing modifications on people
- Architectural element updates
- Product variations in photos
- Selective object replacement

The model's semantic understanding eliminates need for manual mask creation while maintaining precise control over edit boundaries.

Style Transfer with Gemini SDK

Transform image content into different artistic styles while preserving original composition and subjects.

```
from google import genai  
from google.genai import types  
from PIL import Image  
from io import BytesIO  
client = genai.Client()
```



```
# Base image prompt: "A serene mountain lake surrounded by pine trees, with snow-capped peaks reflected in still water during golden hour."
```

```
lake_image = Image.open('/path/to/your/mountain_lake.jpg')

text_input = """Transform this mountain lake photograph into traditional Japanese woodblock print style. Maintain the original mountain and tree composition, but render with flat color blocks, bold outlines, and the characteristic stylized waves and clouds of ukiyo-e art. Use a limited color palette of deep blues, forest greens, and warm oranges."""
```

```
# Generate stylized image
```

```
response = client.models.generate_content(
    model="gemini-2.5-flash-image-preview",
    contents=[lake_image, text_input],
)

image_parts = [
    part.inline_data.data
    for part in response.candidates[0].content.parts
    if part.inline_data
]

if image_parts:
    image = Image.open(BytesIO(image_parts[0]))
    image.save('lake_woodblock_style.jpg')
    image.show()
```

Style Transfer Prompt Structure

```
text_input = """Transform this mountain lake photograph into traditional Japanese woodblock print style..."""
```

Essential components:

- Source description ("mountain lake photograph")
- Target style ("Japanese woodblock print")
- Composition preservation ("maintain original mountain and tree composition")
- Style-specific details ("flat color blocks, bold outlines")

Style Element	Description	Example
Technique	Artistic method	"woodblock print", "oil painting", "watercolor"
Color palette	Specific colors	"limited palette of blues and greens"
Brushwork	Mark-making style	"bold outlines", "soft brushstrokes"
Composition	Layout approach	"maintain original composition"

•

Classical Art:

- Impressionist painting
- Renaissance portraiture
- Abstract expressionism

Cultural Styles:

- Japanese ukiyo-e
- Chinese ink painting
- Art Nouveau illustration

Modern Techniques:

- Comic book illustration
- Digital art styles

- Photography filters

Processing Workflow

Same technical structure as previous examples:

1. Load source image
2. Define style transformation prompt
3. Generate stylized version
4. Extract and save result

The model interprets artistic styles from textual descriptions, applying learned visual characteristics to new content.

Combine Multiple Images with Gemini SDK

Combine elements from different source images to create unified composite scenes.

```
from google import genai
from google.genai import types
from PIL import Image
from io import BytesIO
client = genai.Client()

# Base images:

# 1. Laptop: "A sleek silver laptop computer photographed on a white background, lid open at 45 degrees, screen visible."

# 2. Office: "A modern home office with wooden desk, large window, potted plant, and warm natural lighting."

laptop_image = Image.open('/path/to/your/laptop.jpg')
office_image = Image.open('/path/to/your/office.jpg')

text_input = """Create a professional workspace scene by placing the silver laptop from the first image onto the wooden desk in the second image. Position it naturally with realistic shadows and
```

```
reflections. The laptop screen should show a soft glow, and the overall lighting should match the warm office environment."""
```

Generate composite image

```
response = client.models.generate_content(
    model="gemini-2.5-flash-image-preview",
    contents=[laptop_image, office_image, text_input],
)
image_parts = [
    part.inline_data.data
    for part in response.candidates[0].content.parts
    if part.inline_data
]
if image_parts:
    image = Image.open(BytesIO(image_parts[0]))
    image.save('workspace_composite.jpg')
    image.show()
```

Multiple Image Input

```
contents=[laptop_image, office_image, text_input]
```

The contents array accepts multiple images before the text prompt. Order doesn't affect processing - the model identifies elements by description.

Composition Prompt Elements

```
text_input = """Create a professional workspace scene by placing the silver laptop from the first image onto the wooden desk..."""
```

Key components:

- Element identification ("silver laptop from the first image")
- Placement location ("onto the wooden desk")

- Integration requirements ("realistic shadows and reflections")
- Lighting harmony ("match the warm office environment")

Integration Aspect	Prompt Instruction	Result
Positioning	"Place naturally on desk"	Realistic object placement
Lighting	"Match warm environment"	Consistent illumination
Physics	"Realistic shadows/interactions"	Proper surface interaction
Scale	"Position naturally"	Appropriate size

Common Composition Uses

- Product mockups in lifestyle settings
- Fashion items on models
- Objects in architectural spaces
- Creative artistic collages
- Marketing material creation

The model analyzes spatial relationships and lighting conditions to create believable composite scenes.

High-Fidelity Details with Gemini SDK

Preserve essential features during edits by describing them explicitly in prompts.

```
from google import genai
from google.genai import types
from PIL import Image
from io import BytesIO
client = genai.Client()
```

```
# Base images:
```

```
# 1. Man: "A professional headshot of a man with distinctive green eyes, a small scar above his left eyebrow, and salt-and-pepper beard, wearing a white collared shirt."
```

```
# 2. Sunglasses: "A pair of classic black aviator sunglasses photographed on white background."
```

```
man_image = Image.open('/path/to/your/man.jpg')
```

```
sunglasses_image = Image.open('/path/to/your/sunglasses.jpg')
```

```
text_input = """Place the aviator sunglasses from the second image onto the man in the first image. Critical preservation requirements: maintain his distinctive green eyes visible through the lenses, keep the small scar above his left eyebrow exactly as it appears, and preserve his salt-and-pepper beard pattern completely unchanged. The sunglasses should sit naturally on his nose bridge."""
```

```
# Generate detail-preserved image
```

```
response = client.models.generate_content(  
    model="gemini-2.5-flash-image-preview",  
    contents=[man_image, sunglasses_image, text_input],  
)
```

```
image_parts = [  
    part.inline_data.data  
    for part in response.candidates[0].content.parts  
    if part.inline_data  
,  
]
```

```
if image_parts:
```

```
    image = Image.open(BytesIO(image_parts[0]))  
    image.save('man_with_preserved_details.jpg')  
    image.show()
```

Detailed Preservation Prompts

```
text_input = """...Critical preservation requirements: maintain his distinctive green eyes visible through the lenses, keep the small scar above his left eyebrow exactly as it appears..."""
```

Preservation strategies:

- Enumerate specific features to protect
- Use precise descriptive language
- Specify exact preservation requirements
- Include positioning instructions

Feature Documentation

Detail Type	Description Method	Preservation Instruction
Facial features	"distinctive green eyes"	"maintain exactly as shown"
Unique marks	"small scar above left eyebrow"	"keep in precise location"
Textures	"salt-and-pepper beard pattern"	"preserve completely unchanged"
Brand elements	"company logo placement"	"maintain pixel-perfect accuracy"

Critical Applications

- Face preservation during accessories addition
- Logo integrity during product placement
- Brand element protection during scene changes
- Signature detail maintenance during style transfers

Detailed descriptions ensure the model prioritizes specific elements during complex edits.

PART 7: Build AI Apps with Google AI Studio

Dive into ways that you can build web, mobile and desktop applications and games that have AI features, such as your own image editing apps!

That's right, you can build apps that leverage machine learning models like those from Google. For example, you can build an app that calls Nano Banana to fix up people's profile pics! Let's build projects that use AI in creative ways and convert. From fun apps to business software, you can build it with Python.

Getting Started with Google AI Studio

Google AI Studio provides web-based experimentation with AI models through intuitive prompt interfaces. The platform enables rapid prototyping before transitioning to API implementation.

Platform Interface Overview

AI Studio offers specialized interfaces for different use cases:

Interface Type	Purpose	Best For
Chat prompts	Conversational experiences	Customer service bots, assistants
Realtime streaming	Live interactions	Real-time applications
Video generation	Video content creation	Media and marketing
Structured	Formatted responses	Data processing, APIs

Configuration Options

Run Settings Panel:

- Model parameter adjustments
- Safety setting configuration
- Structured output formatting
- Function calling capabilities
- Code execution tools
- Grounding and fact-checking

Building Custom Chat Applications

System Instructions Setup

System instructions define chatbot personality and behavior constraints:

```
Example: "You are Marina, a marine biologist working at an
underwater research station. Keep responses under 2 paragraphs and
maintain scientific accuracy while being conversational."
```

Instruction Components:

- Character identity and background
- Response length limitations
- Tone and communication style
- Knowledge domain constraints

Chat Prompt Creation Process

Step 1: Initialize Chat Interface

- Open Google AI Studio
- Select Chat from interface options
- Click assignment icon to expand System Instructions

- Enter character and behavior guidelines

Step 2: Test Initial Interactions

User input: "What's your daily routine like?"

Expected response: Character-appropriate answer maintaining persona

Step 3: Refine Through Iteration

- Test various user scenarios
- Adjust system instructions based on responses
- Add specific constraints for consistency
- Evaluate response quality and adherence

Character Development Examples

Basic Character Setup

System Instructions: "You are Alex, a coffee shop owner in Portland. You're passionate about sustainable coffee practices and local community building."

Test Interaction:

User: "What's your most popular drink?"

Model Response: "Our house blend is definitely the crowd favorite! It's a medium roast from a cooperative in Guatemala that we've been working with for three years now..."

Enhanced Character Refinement

Updated Instructions: "You are Alex, a coffee shop owner in Portland who's passionate about sustainability. Keep answers under 3 sentences and always mention either coffee origins, brewing methods, or community impact."

Improved Response:

User: "What's your most popular drink?"

Model Response: "Our Guatemala house blend is the top seller - it's direct trade from a women's cooperative we support. We brew it as pour-over to highlight the chocolate and citrus notes our regulars love."

Consistency Factors

Instruction Element	Impact on Responses	Example
Character name	Personal identity	"I'm Alex" vs generic responses
Background details	Authentic responses	Coffee expertise vs general knowledge
Tone specifications	Communication style	Professional vs casual delivery
Response constraints	Output consistency	Length limits, topic focus

Iterative Improvement Workflow

Testing and Refinement

Evaluation Criteria:

- Response consistency with character
- Appropriate tone and style maintenance
- Information accuracy within domain
- Response length adherence

Common Adjustments:

- Tighten response length constraints
- Add specific knowledge domain focus
- Enhance personality traits

- Improve conversation flow

Token Management

Chat sessions accumulate message history, eventually reaching model token limits:

- Monitor conversation length
- Start new sessions for extended testing
- Save effective prompt variations
- Document successful instruction patterns

Production Transition

Code Generation

AI Studio generates implementation code:

```
# Generated code for API usage
import google.generativeai as genai
genai.configure(api_key="YOUR_API_KEY")
model = genai.GenerativeModel(
    'gemini-2.5-pro',
    system_instruction="Your character instructions here"
)
chat = model.start_chat()
response = chat.send_message("User message")
```

Prompt Saving and Sharing

Documentation Features:

- Save prompts for later development
- Share configurations with team members
- Export instruction sets for different environments

- Version control for prompt iterations

This rapid prototyping approach enables quick validation of chatbot concepts before investing in full development implementation.

Generate Image Apps with Google AI Studio

Google AI Studio is a web-based interface for rapid prototyping and testing. The Studio allows you to generate entire applications with a simple prompt! These apps can integrate AI features using Gemini models such as Gemini 2.5 Flash Image ("Nano Banana")!

Setup Process

- Access through browser at aistudio.google.com
- Create a project
- Select Gemini 2.5 Flash Image model
- Use built-in prompt interface to generate an app

Interface Features

Feature	Description	Use Case
Visual prompt builder	Drag-and-drop interface	Quick experimentation
Real-time preview	Immediate image generation	Iterative design
Prompt history	Saved previous requests	Template reuse
Export options	Code generation for SDK	Production transition

Code Export

*An **SDK (Software Development Kit)** is a collection of tools, libraries, documentation, and code samples that developers use to build applications for a specific platform, framework, or service.*

AI Studio generates SDK-compatible code such as the following:

```
# Generated from AI Studio
```

```
import google.generativeai as genai
genai.configure(api_key="YOUR_API_KEY")
model = genai.GenerativeModel('gemini-2.5-flash-image-preview')
response = model.generate_content([prompt, image])
```

PART 8: Build Enterprise Apps with Google Vertex AI

Vertex AI is Google's enterprise platform for production deployments. You can either use the Vertex AI API or the Studio!

In software and AI, **deployments** refer to the process of releasing an application, model, or system into a production environment where real users can access and use it.

Use the Vertex AI API

Setup Requirements

Install and update the Gen AI SDK for Python by running this command.

```
pip install --upgrade google-genai
```

Set environment variables:

```
export GOOGLE_API_KEY=YOUR_API_KEY
export GOOGLE_GENAI_USE_VERTEXAI=True
```

Then connect to Gemini:

```
from google import genai
client = genai.Client()
```

```
vertexai=True, project='your-project-id', location='us-central1'
)
response = client.models.generate_content(
    model="gemini-2.0-flash", contents="What is a mammoth?"
)
print(response.text)
```

Generate and edit images

```
from google import genai
from google.genai.types import GenerateContentConfig, Modality
from PIL import Image
from io import BytesIO
client = genai.Client()
response = client.models.generate_content(
    model="gemini-2.5-flash-image-preview",
    contents=("Generate an image of a mammoth surfing."),
    config=GenerateContentConfig(
        response_modalities=[Modality.TEXT, Modality.IMAGE],
        candidate_count=1,
        safety_settings=[
            {"method": "PROBABILITY"},
            {"category": "HARM_CATEGORY_DANGEROUS_CONTENT"},
            {"threshold": "BLOCK_MEDIUM_AND_ABOVE"},
        ],
    ),
)
```

```
for part in response.candidates[0].content.parts:
    if part.text:
        print(part.text)
    elif part.inline_data:
        image = Image.open(BytesIO((part.inline_data.data)))
        image.save("output_folder/mammoth.png")
```

Enterprise Features

- IAM integration and access controls
- Audit logging and compliance tracking
- Custom model tuning capabilities
- Batch processing for large workloads

Platform Comparison

Aspect	Google AI Studio	Vertex AI
Target users	Developers, researchers	Enterprise teams
Setup complexity	Minimal	Requires GCP project
Pricing model	Pay-per-use	Enterprise billing
Security	Basic API key	IAM, VPC, encryption
Scalability	Individual use	Production scale

Selection Criteria

Choose Google AI Studio for:

- Prototyping and experimentation

- Individual developer projects
- Learning and testing capabilities
- Quick proof-of-concepts

Choose Vertex AI for:

- Production applications
- Enterprise security requirements
- Team collaboration needs
- Integration with existing GCP infrastructure

Both platforms use identical model capabilities and API structures, differing primarily in deployment context and enterprise features.

Use the Vertex AI Studio on Google Cloud Platform

You can use Vertex AI Studio to create, test, and manage prompts for Google's Gemini large language models (LLMs) as well as select third-party models. It also supports certain external models offered through Vertex AI as Models-as-a-Service (MaaS), including Anthropic's Claude and Meta's Llama.

Requirements:

Create a Google Cloud Platform account.

In the Google Cloud console, select an existing or create a new Google Cloud project.

Enable billing in the project.

Enable the Vertex AI API in the project.

Use a prompt in Vertex AI Studio

In the Google Cloud console, go to Vertex AI > **Prompt gallery**.

In the **Tasks** drop-down menu, choose a card (depending on your task of choice.)

You can select a Gemini model, such as Nano Banana, by clicking **Switch model**.

Click Submit to send the task and get a response.

Deploying Vertex AI Prompts as Web Applications

Vertex AI Studio enables deployment of refined prompts as standalone web applications hosted on Cloud Run. This capability transforms prompt experimentation into production-ready services accessible outside the Google Cloud console.

Deployment Overview

Transform AI Studio prompts into web applications for broader user access and testing. The deployment process creates containerized applications on Google's serverless Cloud Run platform.

Prerequisites and Permissions

Requirement	Purpose	Setup Process
Google Cloud project	Hosting environment	Create with billing enabled
Vertex AI API	Core AI functionality	Enable through console
Cloud Run APIs	Application hosting	Automatic enablement during first deployment
Service account roles	Deployment permissions	Grant Vertex AI Service Agent and Cloud Build roles

Required API Enablement

Essential APIs for deployment:

- Cloud Run Admin API (`run.googleapis.com`)
- Identity-Aware Proxy API (`iap.googleapis.com`)
- Artifact Registry API (`artifactregistry.googleapis.com`)
- Cloud Build API (`cloudbuild.googleapis.com`)
- Cloud Logging API (`logging.googleapis.com`)

Prompt Creation with Variables

Variable Configuration

Create dynamic prompts using variables that users can modify:

1. Navigate to Vertex AI Studio prompt creation page
1. Click "Add variable" in prompt input interface
1. Define variable name and default value
1. Apply variables within prompt text

Variable Implementation Example:

Prompt: "You are a helpful assistant for [INDUSTRY] businesses. Provide advice about [TOPIC] while maintaining [TONE] communication style."

Variables:

- INDUSTRY: "technology"
- TOPIC: "customer service"
- TONE: "professional"

Advanced Prompt Configuration

Grounding Options:

- Google Search integration for current information
- Custom knowledge base connections

- Real-time data access capabilities

System Instructions:

- Define AI behavior and knowledge constraints
- Set response format and length requirements
- Establish safety and content guidelines

Deployment Process

Initial Deployment Workflow

Step	Action	Duration	Result
1	Click "Build with code" → "Deploy as app"	Immediate	Opens save dialog
2	Save prompt (first time in region)	2-3 minutes	Prompt stored
3	Enable required APIs	1-2 minutes	Infrastructure ready
4	Configure access control	30 seconds	Security settings
5	Deploy application	2-3 minutes	Live web app

Access Control Configuration

Authentication Options:

- Required authentication via Identity-Aware Proxy
- Public access (no authentication required)

Security Considerations for Public Access:

- Avoid including sensitive information in prompts
- No personally identifiable information (PII)

- Consider rate limiting for resource management
- Monitor usage patterns for abuse prevention

Secret Key Protection

Deployed applications include secret key requirements:

- Secret key appended to URL for prompt submission
- Basic protection against unauthorized access
- Key visible in Manage Web App dialog
- URL format: `https://app-url.com/?key=SECRET_KEY`

Application Management

Monitoring Deployment Status

Status Tracking Methods:

- Manage Web App dialog for detailed information
- Notification bell for deployment completion alerts
- Cloud Run console for infrastructure monitoring

Deployment States:

- Deploying: Application creation in progress
- Ready: Application accessible and functional
- Error: Deployment failed, requires troubleshooting

Application Updates

Re-deployment Process:

1. Modify prompt in Vertex AI Studio
1. Click "Build with code" → "Manage app"
1. Select "Update app" for re-deployment

1. Confirm update (loses external code changes)
1. Monitor deployment progress

Update Considerations:

- External code modifications will be lost
- Re-deployment takes 2-3 minutes
- Application URL and secret key remain unchanged
- Previous deployment automatically replaced

Cloud Run Serverless Architecture

Serverless Characteristics:

- Containers shut down during idle periods
- Cold start delays for first request after idle
- Automatic scaling based on demand
- Pay-per-use billing model

Performance Implications:

- Initial load may take several seconds
- Refresh page if submission fails after idle period
- Consistent performance during active use

Advanced Features

Multimodal Input Support

Web applications support file uploads based on model capabilities:

- Image upload and processing
- Video file analysis
- Audio content processing

- Document upload and analysis

File Upload Interface:

- Click clip icon in conversation input
- Select supported file types
- Process multimodal prompts with uploaded content

Conversation Interface

Deployed applications support conversational interactions:

- Multi-turn dialogue capabilities
- Context retention across conversation
- Dynamic response based on previous exchanges

Access Control Management

Identity-Aware Proxy Setup:

1. Open Manage Web App dialog
1. Click edit icon in Access Control column
1. Enable Identity-Aware Proxy checkbox
1. Edit policy to add authorized users
1. Enter user or group emails as principals
1. Save changes (requires two save confirmations)

Troubleshooting Common Issues

Error	Cause	Solution
No secret key	URL missing key parameter	Append ?key=SECRET_KEY to URL
Invalid secret key	Wrong or expired key	Use current key from Manage dialog
Empty input error	No content submitted	Provide non-empty prompt input
Unsupported mime type	Wrong file format uploaded	Use supported file types

PART 9: Performance and Optimization

Batch Mode Processing for Large-Scale AI Operations

Gemini API Batch Mode processes large volumes of requests asynchronously at reduced cost. The system targets 24-hour turnaround but typically completes much faster for most workloads.

Batch Mode Overview

Batch processing optimizes cost and efficiency for high-volume, non-urgent AI operations.

Feature	Interactive API	Batch Mode
Cost	Standard pricing	50% of standard cost
Response time	Immediate	24 hours target (often faster)

Use cases	Real-time interactions	Data processing, evaluations
Volume	Individual requests	Thousands of requests

Optimal Use Cases

Large-Scale Operations:

- Data preprocessing pipelines
- Content analysis projects
- Evaluation and testing suites
- Bulk content generation

Non-Urgent Processing:

- Overnight analysis jobs
- Weekly report generation
- Archive content processing
- Research data analysis

Request Submission Methods

Inline Requests

Embed requests directly within batch creation for smaller volumes (under 20MB total):

```
from google import genai
from google.genai import types
client = genai.Client()
inline_requests = [
    {
        'contents': [{
```

```
        'parts': [{'text': 'Summarize the benefits of renewable
energy in one paragraph.'}],
        'role': 'user'
    }]
},
{
    'contents': [{
        'parts': [{'text': 'Explain quantum computing basics in
simple terms.'}],
        'role': 'user'
    }]
}
]
inline_batch_job = client.batches.create(
    model="models/gemini-2.5-flash",
    src=inline_requests,
    config={
        'display_name': "analysis-batch-job-1",
    },
)
print(f"Created batch job: {inline_batch_job.name}")
```

Input File Method

Use JSON Lines (JSONL) files for larger request volumes (up to 2GB per file):

```
import json

# Create JSONL file
```

```
with open("analysis-requests.jsonl", "w") as f:

    requests = [

        {"key": "energy-analysis", "request": {"contents":
[{"parts": [{"text": "Analyze current solar technology
trends."}]}}]},

        {"key": "tech-summary", "request": {"contents": [{"parts":
[{"text": "Summarize blockchain applications in healthcare."}]}}]}

    ]

    for req in requests:

        f.write(json.dumps(req) + "\n")

# Upload file
uploaded_file = client.files.upload(
    file='analysis-requests.jsonl',
    config=types.UploadFileConfig(
        display_name='analysis-requests',
        mime_type='jsonl'
    )
)

# Create batch job
file_batch_job = client.batches.create(
    model="gemini-2.5-flash",
    src=uploaded_file.name,
    config={
        'display_name': "file-analysis-job-1",
    },
)
```

Advanced Request Configuration

System Instructions Integration

```
inline_requests = [  
    {'contents': [{'parts': [{'text': 'Analyze market trends for  
electric vehicles.'}]}]},  
    {  
        'contents': [{'parts': [{'text': 'Evaluate sustainability  
initiatives.'}]}],  
        'system_instructions': {'parts': [{'text': 'You are an  
environmental analyst focusing on corporate sustainability.'}]}  
    }  
]
```

Tool Integration

```
inline_requests = [  
    {'contents': [{'parts': [{'text': 'What were the results of the  
2024 climate conference?'}]}]},  
    {  
        'contents': [{'parts': [{'text': 'Current renewable energy  
capacity worldwide?'}]}],  
        'tools': [{'google_search': {}}]  
    }  
]
```

Structured Output Configuration

```
from pydantic import BaseModel  
  
class MarketAnalysis(BaseModel):  
    industry: str  
    key_trends: list[str]
```

```
growth_rate: float
inline_requests = [
    {
        'contents': [{
            'parts': [{'text': 'Analyze the electric vehicle market
with current trends and growth projections.'}],
            'role': 'user'
        }],
        'config': {
            'response_mime_type': 'application/json',
            'response_schema': list[MarketAnalysis]
        }
    }
]
```

Job Monitoring and Status Management

Status Polling Implementation

```
import time
job_name = inline_batch_job.name
print(f"Monitoring job: {job_name}")
while True:
    batch_job = client.batches.get(name=job_name)
    if batch_job.state.name in ('JOB_STATE_SUCCEEDED',
'JOB_STATE_FAILED', 'JOB_STATE_CANCELLED', 'JOB_STATE_EXPIRED'):
        break
    print(f"Job status: {batch_job.state.name}. Waiting 30
seconds...")
    time.sleep(30)
```

```
print(f"Job completed with state: {batch_job.state.name}")
```

Job State Definitions

State	Description	Action Required
JOB_STATE_PENDING	Queued for processing	Continue monitoring
JOB_STATE_RUNNING	Currently processing	Continue monitoring
JOB_STATE_SUCCEEDED	Completed successfully	Retrieve results
JOB_STATE_FAILED	Processing failed	Check error details
JOB_STATE_CANCELLED	User-cancelled	No further action
JOB_STATE_EXPIRED	Exceeded 48-hour limit	Resubmit with smaller batches

Result Retrieval

Processing Inline Results

```
if batch_job.state.name == 'JOB_STATE_SUCCEEDED':
    for i, inline_response in
enumerate(batch_job.dest.inlined_responses):
        print(f"\n--- Response {i+1} ---")
        if inline_response.response:
            print(inline_response.response.text)
        elif inline_response.error:
            print(f"Error: {inline_response.error}")
```

File-Based Result Processing

```
if batch_job.dest and batch_job.dest.file_name:
    result_file_name = batch_job.dest.file_name
```

```
file_content = client.files.download(file=result_file_name)

# Process JSONL results

for line in file_content.decode('utf-8').strip().split('\n'):
    result = json.loads(line)
    if 'response' in result:
        print(f"Key: {result.get('key', 'unknown')}")
        print(f"Response: {result['response']['candidates'][0]
['content']['parts'][0]['text']}")
```

Job Management Operations

Cancellation

```
# Cancel ongoing job

client.batches.cancel(name=batch_job.name)
```

Deletion

```
# Remove completed job

client.batches.delete(name=batch_job.name)
```

Best Practices

File Management Strategy

- Use input files for requests over 20MB total size
- Structure JSONL files with meaningful keys for result tracking
- Keep individual files under 2GB limit
- Break large datasets into multiple batch jobs

Error Handling

- Check `batchStats.failedRequestCount` after completion
- Parse individual response lines for file-based results

- Implement retry logic for failed requests
- Monitor system load impacts on processing time

Cost Optimization

- Batch similar requests together for maximum savings
- Use appropriate models for task complexity
- Leverage context caching for repeated patterns
- Monitor usage to optimize batch sizing

Batch mode provides cost-effective processing for large-scale AI operations while maintaining the same model capabilities as interactive API usage.

Context Caching with Gemini 2.5 Flash (Nano Banana)

Context caching helps you avoid resending large, repeated prompt content to the Gemini models. With Gemini 2.5 Flash Image (a.k.a. "Nano Banana"), you get two flavors:

- **Implicit caching** — automatic on 2.5 models; zero setup.
- **Explicit caching** — you create/manage caches; guarantees cost savings with a little extra work.

This chapter explains both approaches and gives **Pythononly examples** you can drop into your Nano Banana imageediting workflows.

Quick Reference: Implicit vs. Explicit

Feature	Implicit Caching (2.5 models)	Explicit Caching (most models)
Setup	None	Create cache, set TTL
Cost Savings	Opportunistic (no guarantee)	Guaranteed on cache hits
When to Use	Shortlived bursts, repeated prefixes in a session	Structured pipelines, repeated large contexts
Control	None	Full (TTL, which content is cached)
Visibility	Hits shown in usage_metadata	Cache metadata via cache service + usage_metadata

Implicit Caching (AutoEnabled)

What it is: Gemini 2.5 models automatically try to reuse token context across closely related requests.

Minimum input sizes:

- **2.5 Flash:** 1,024 tokens
- **2.5 Pro:** 4,096 tokens

Tips to increase cache hits:

- Put **big, repeated content at the beginning** of your prompt.
- Send **similar prefixes close in time** (think bursty batches).
- Inspect `usage_metadata` on responses to see cachehit token counts.

You don't configure anything for implicit caching—just structure prompts to be cachefriendly.

Explicit Caching (You Control It)

What it is: You upload/define a reusable context once, then reference it by name in later requests. You choose how long it lives via **TTL** (defaults to 1 hour).

Why it's great:

- **Predictable savings** when you repeatedly use the same large context.
- Choose a **TTL** that matches your workflow (minutes to days).
- Ideal for **image editing playbooks, style guides, tool docs**, long **system instructions**, or **large corpora** you query repeatedly.

Prerequisites (Python)

- Python 3.9+
- `google-genai` (or the current Gemini Python SDK)
- Environment variable `GEMINI_API_KEY` set

```
pip install google-genai
export GEMINI_API_KEY=YOUR_KEY
```

The examples below use the idioms of the official Gemini Python SDK.

Example 1 — Create & Use a Cache (Python)

This pattern caches a **system instruction** and a **text corpus** (e.g., a transcript or style guide), then references that cache in generation calls.

```
import os

from google import genai
from google.genai import types

client = genai.Client(api_key=os.environ["GEMINI_API_KEY"]) # or
genai.configure(api_key=...)

MODEL = "gemini-2.0-flash-001" # Use the latest Flash (Nano Banana
pipeline)
```

1) Upload a file you want to reuse across requests

```
uploaded = client.files.upload(  
    file="path/to/file.txt",  
    config=types.UploadFileConfig(mime_type="text/plain"),  
)  
print("Uploaded file URI:", uploaded.uri)
```

2) Create the cache: reusable content + system instruction

```
cache = client.caches.create(  
    model=MODEL,  
    config=types.CachedContentConfig(  
        contents=[  
            types.Content(  
                role="user",  
                parts=[types.Part.from_uri(uploaded.uri,  
uploaded.mime_type)],  
            )  
        ],  
        system_instruction="You are an expert analyzing  
transcripts.",  
        # Optional: display_name="Transcript Cache v1"  
    ),  
)  
print("Cache name:", cache.name)
```

3) Generate with the cached content (cheap prefix)

```
resp = client.models.generate_content(  
    model=MODEL,  
    contents="Please summarize this transcript",
```

```
config=types.GenerateContentConfig(cached_content=cache.name),
)
print("Text:\n", resp.text)
print("Usage metadata:", resp.usage_metadata)
```

What's happening? You pay full price once to ingest the big context, then reference it by `cached_content=cache.name` in subsequent calls for lower input token cost.

Example 2 — List Cache Metadata

You can't read cached content, but you can list metadata (name, model, create/update times, expire time, usage stats).

```
# List caches in pages of 10
pager = client.caches.list(page_size=10)
print("My caches:")
for page in pager: # pager yields pages
    for entry in page:
        print(" ", entry.name, entry.model, entry.expire_time)
```

Example 3 — Update Cache TTL

Change how long a cache should live. Only TTL/expireTime updates are supported.

```
# Extend TTL to 2 hours (7200 seconds)
updated = client.caches.update(
    name=cache.name,
    ttl=f"{2 * 3600}s",
)
print("Updated TTL expires at:", updated.expire_time)
```

Example 4 — Delete a Cache

Remove a cache when you no longer need it.

```
client.caches.delete(name=cache.name)
print("Cache deleted.")
```

Example 5 — Explicit Caching via OpenAICompatible Clients (Python)

If you're using an OpenAIcompatible Python client that forwards through Vertex/Gemini, set `extra_body={"cached_content": cache_name}`.

```
from openai import OpenAI

client = OpenAI(api_key=os.environ["GEMINI_API_KEY"]) # routed to
Gemini via your provider

result = client.chat.completions.create(
    model=MODEL,
    messages=[{"role": "user", "content": "Summarize the
transcript."}],
    extra_body={"cached_content": "caches/your-cache-id"},
)

print(result.choices[0].message.content)
```

The exact wiring depends on your gateway/provider. The key is the `cached_content` field in the extra body.

When to Use Explicit Caching

Great fits:

- Chatbots with **large system prompts** or policy packs
- Repetitive **analysis of long videos/audio/transcripts**
- Recurring queries over **big document sets** (handbooks, SOPs)
- Frequent **codebase or bugfix** analysis with the same repo context
- **Image editing pipelines** that reuse identical style guides, glossaries, or tool docs

Cost Model (At a Glance)

Dimension	How You're Billed
Cache token count	Discounted rate when reused in later prompts
Storage duration (TTL)	Per cached token * time stored
Other	Regular rates for <i>noncached</i> input tokens + output tokens

For current prices, check the Gemini API pricing page. Token counts include cached + noncached tokens.

Practical Tips for Nano Banana Image Workflows

- Cache your style guide: brand colors, composition rules, do/don't lists.
- **Cache tool docs:** how to apply masks, layers, rescaling steps you repeat.
- Chunk large guides: If you exceed token limits, split into themed caches (e.g., "color", "composition", "retouching").
- **Warmup bursts:** Send a few similar requests backtoback to improve implicit hit rates.
- **Prefix discipline:** Put common, repeated parts **first** in each prompt.

Token & Limit Notes

Minimum tokens for context caching:

- 2.5 Flash: **1,024**
- 2.5 Pro: **2,048** (explicit) / **4,096** (implicit threshold guidance)

Cached content is treated as a **prefix**; standard **rate limits** and **model token limits** still apply (cached + new tokens combined).

Check `usage_metadata` everywhere: cache creation, listing, and `generate_content` responses.

Troubleshooting & Gotchas

- "Why no savings?" Your input may be under the minimum token threshold, too dissimilar across calls, or too spaced out in time.
- **"Cache not found?" Ensure you pass the exact `cache.name` string to `cached_content`.**
- "Exceeded context window?" Remember: cached tokens count toward the model's max tokens.
- "Stale instructions?" Increase TTL for longrunning batches—or rotate caches when policies change.

WrapUp

For image editing with Nano Banana, context caching is a simple lever for **lower costs** and **faster iteration**. Use **implicit caching** for quick wins, and switch to **explicit caching** when you need predictable savings and lifecycle control.

PART 10: Authentication and Authorization

What's the Difference?

Before diving deeper into API integration, it's important to distinguish between authentication and authorization. These are related but separate mechanisms that control identity verification and resource access.

Key Definitions

- **Authentication** → Confirms **who** is making the request.
- **Authorization** → Determines **what** resources the requester can access and at **what level**.

- **Relationship** → Authentication always comes first. Without verifying identity, you cannot apply access rules.

Analogy: Hotel Reservation

- **Authentication:** You show your ID at the hotel front desk. The clerk confirms you are a guest with a reservation.
- **Authorization:** The clerk hands you a room key. The key lets you enter your room, the gym, and the business center—but not staffonly areas.

This illustrates how authentication (ID) and authorization (key) work together.

High-Level Steps in Google APIs

Below is a simplified flow of how authentication and authorization operate with Gemini and other Google APIs:

1. Configure Cloud Project & App

- Register your app in Google Cloud Console.
- Define **scopes** (permissions) and create **credentials** (API key, OAuth client, or service account).

2. Authenticate Your App

- Your app presents credentials when it runs.
- If authenticating as an end user, Google may show a **signin prompt**.

3. Request Resources

- The app requests Google resources, declaring the scopes needed.

4. Ask for User Consent

- For enduser flows, the **OAuth consent screen** appears.
- Users approve or deny access to the requested scopes.

5. Send Approved Request

- Once the user consents, the app bundles credentials and scopes into a request.

- Google's authorization server issues an access token.

6. Receive Access Token

- The token contains the list of approved scopes.
- If fewer scopes are granted than requested, features outside those scopes are disabled.

7. Access Resources

- The app uses the access token to call Gemini or other Google APIs.

8. Use Refresh Tokens (Optional)

- For longlived access, a **refresh token** lets the app obtain new access tokens without reprompting the user.

9. Request More Resources

- If new permissions are needed, the app repeats steps 3–6 with additional scopes.

Important Notes

- **Authentication = identity.** Authorization = permissions.
- **Scopes** define the *boundaries* of what your app can do.
- **Consent** ensures users remain in control of their data.
- **Access tokens** expire; plan to refresh or reauthenticate.

WrapUp

In Nano Banana workflows, you'll use authentication (API key or OAuth credentials) to prove identity and authorization (scopes + tokens) to access image editing and generative AI resources. Understanding this foundation prepares you for secure, productiongrade integrations.

Types of Access Credentials

Before you can use authentication and authorization with Google APIs such as Gemini (Nano Banana), you must create credentials. Credentials identify your app to Google's authorization servers and enable you to obtain access tokens. These tokens let your app call Google APIs securely.

Choosing the Right Credential Type

Depending on your use case, platform, and data access model, you'll choose one of three credential types:

Use Case	Authentication Method	About This Method
Access public data anonymously	API key	Suitable for accessing data shared publicly (e.g., files with "Anyone with the link"). Confirm that the API supports API keys before use.
Access user data (e.g., email, profile info)	OAuth client ID	Requires user consent. Your app prompts the user through an OAuth flow.
Access applicationowned data or act on behalf of Workspace/Cloud Identity users	Service account	Your app authenticates as a robot account. Can be granted domainwide delegation for organizationwide use.

API Key Credentials

An **API key** is a simple string (letters, numbers, symbols) that identifies your app for accessing public data.

How to Create an API Key

1. In the **Google Cloud Console**, go to: **Menu** → **APIs & Services** → **Credentials**.
1. Click **Create credentials** → **API key**.
1. Copy and save the new key.
1. (Recommended) Restrict the key to specific APIs and usage environments.

OAuth Client ID Credentials

Use an OAuth client ID if your app needs access to a user's private data.

Key Steps

1. In Cloud Console, open: **Menu** → **Google Auth platform** → **Clients**.
1. Click **Create Client**.
1. Choose the **application type** (Web, Android, iOS, Chrome Extension, Desktop).
1. Provide app details (name, redirect URIs, authorized origins).
1. Click **Create**. The new Client ID is listed under OAuth 2.0 Client IDs.

Service Account Credentials

A service account is used by apps, not people. They're perfect for server-to-server interactions and domainwide automation.

Create a Service Account

1. In Cloud Console, go to **Menu** → **IAM & Admin** → **Service Accounts**.
1. Click **Create service account** and provide details.
1. (Optional) Assign roles for resource access.
1. (Optional) Add users/groups who can manage the account.
1. Save the generated **service account email**.

Generate Keys for a Service Account

1. Go to your service account in Cloud Console.
1. Click **Keys** → **Add key** → **Create new key**.
1. Select **JSON** and download the file.
1. Save it as `credentials.json` in your project folder.

Optional: DomainWide Delegation

If acting on behalf of Workspace users, enable **Domainwide Delegation**:

- Copy the Client ID from the service account.
- In **Admin Console** → **Security** → **API Controls** → **Manage DomainWide Delegation**, add the Client ID and required OAuth scopes.

Quick Reference Table

Credential Type	Best For	Requires User Consent	Example
API Key	Public data	No	Fetching a public shared document
OAuth Client ID	User data	Yes	Reading a user's email or calendar
Service Account	App/Org data	No (but may require admin delegation)	Running automated Workspace reports

Wrap Up

Selecting the right credential type is a **critical step** in building secure apps with Gemini (Nano Banana) and Google Workspace APIs. Start with **API keys** for simple

public access, use **OAuth client IDs** for userdata apps, and choose **service accounts** for backend automation or organizationwide integrations.

Authenticating with OAuth for Gemini 2.5 Flash (Nano Banana)

The easiest way to authenticate to the Gemini API is to configure an API key, as we have seen in previous examples. For stricter access controls, use OAuth. While most developers start with a simple **API key**, OAuth provides stricter access controls and is recommended for advanced or production environments.

Let's start with a simplified authentication approach that is appropriate for a testing environment. This chapter walks you through setting up OAuth, from cloud project configuration to Python code that uses OAuth credentials.

Quick Reference: API Key vs. OAuth

Feature	API Key	OAuth 2.0
Setup	Very simple	More steps, cloud configuration required
Security	Basic	Stronger, userscoped permissions
Use Cases	Quick testing, simple apps	Production apps, enduser flows, finegrained access control

Objectives

By the end of this chapter, you will:

- Set up your **Google Cloud project** for OAuth.
- Configure **Application Default Credentials (ADC)**.
- Use Python to authenticate with OAuth credentials.

Prerequisites

Before starting, you need:

- A **Google Cloud project**
- A local installation of the **gcloud CLI**

Install required Python packages:

```
pip install google-generativeai google-auth-oauthlib google-auth-httpplib2 google-api-python-client
```

Step 1 — Enable the API

1. In the **Google Cloud Console**, enable the **Google Generative Language API**.
1. Confirm the API is active for your project.

Step 2 — Configure the OAuth Consent Screen

1. Go to **Menu > Google Auth platform > Overview**.
1. Complete the form, set **User Type = External**.
1. Accept the policy terms, then click **Create**.
1. Add test users: under **Audience > Test Users**, click **Add users**, enter your email, and save.

Step 3 — Create OAuth Client Credentials

1. Navigate to **Google Auth platform > Clients**.
1. Click **Create Client**.
1. Select **Desktop app** as the application type.
1. Provide a name (e.g., *Nano Banana Desktop*).
1. Download the JSON file (e.g., `client_secret_XXXXX.json`). Rename it to **client_secret.json** and place it in your working directory.

Step 4 — Set up Application Default Credentials (ADC)

Run this command to convert your client secret into usable credentials:

```
gcloud auth application-default login \  
  --client-id-file=client_secret.json \  
  --scopes='https://www.googleapis.com/auth/cloud-platform,https://  
www.googleapis.com/auth/generative-language.retriever'
```

- For Colab, add `--no-browser`.
- The token is stored locally so client libraries can use it automatically.

Step 5 — Test Authentication with Python

The `google-generativeai` library will automatically find your ADC credentials.

```
import google.generativeai as genai  
print("Available models:", [m.name for m in genai.list_models()])
```

If this runs successfully, you're authenticated!

Step 6 — Manage Credentials Manually in Python

Sometimes you need to manage OAuth within your Python app (without relying on `gcloud`). This involves caching tokens in a `token.json` file.

load_creds.py

```
import os.path  
  
from google.auth.transport.requests import Request  
from google.oauth2.credentials import Credentials  
from google_auth_oauthlib.flow import InstalledAppFlow  
  
SCOPES = ['https://www.googleapis.com/auth/generative-  
language.retriever']
```

```
def load_creds():
    creds = None
    if os.path.exists('token.json'):
        creds = Credentials.from_authorized_user_file('token.json',
SCOPE)
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                'client_secret.json', SCOPE)
            creds = flow.run_local_server(port=0)
        with open('token.json', 'w') as token:
            token.write(creds.to_json())
    return creds
```

script.py

```
import google.generativeai as genai
from load_creds import load_creds
creds = load_creds()
genai.configure(credentials=creds)
print("Available models:", [m.name for m in genai.list_models()])
```

Run:

```
python script.py
```

The first run opens a browser for login. On later runs, it reuses the cached token.json.

Troubleshooting

- **Unverified app warning:** Normal in simplified setup; click *Continue*.
- **Multiple Google accounts:** Be sure to sign in with the account listed as a test user.
- **Colab setup:** Always use `--no-browser` with `gcloud` and ensure `gcloud --version` is up to date.

When you use OAuth 2.0 for authorization, Google presents a consent screen to users. This screen shows details about your project, its policies, and the permissions (scopes) your app is requesting. Setting up the OAuth consent screen determines what users and app reviewers see, and also registers your app so you can publish it later.

Some Google Workspace APIs—like the Google Drive API—include their own documentation about authentication and authorization. Be sure to review that material before proceeding.

To control what level of access your app has, you must identify and declare authorization scopes. A scope is an OAuth 2.0 URI string that specifies the Google Workspace service, the type of data accessed, and the level of access. Scopes represent your app's requests to use Google Workspace data, including Google Account information.

When someone installs your app, they are prompted to approve the scopes requested. Always choose the narrowest scope possible, and avoid asking for permissions your app doesn't need—users are more willing to grant access when the request is clear and limited.

Every app that uses OAuth 2.0 must configure a consent screen. However, scopes only need to be listed if your app will be used outside your Google Workspace organization.

Tip: If you don't yet have all the required details for the consent screen, you can enter placeholder information before release.

Important: Once you configure the OAuth 2.0 consent screen, it cannot be removed.

Wrap Up

With OAuth set up, your Nano Banana applications now authenticate securely with the Gemini API. Use API keys for simple tests, but switch to OAuth for **production, multiuser, or stricter security needs**.

Optimization, API Rate Limiting and Cost Management

Working with **Gemini 2.5 Flash Image (Nano Banana)** in production means balancing **image quality, system performance, and operational cost**. This chapter covers strategies for **image compression, quality optimization, and API cost management** through **rate limiting**.

Image Compression and Quality Optimization

AI image workflows often involve large file transfers. Oversized images can slow down processing, increase token usage, and raise costs. Proper compression helps you keep performance high without losing critical details.

Why Compress Images?

- **Faster uploads/downloads** → reduces latency.
- **Lower token counts** → smaller inputs mean fewer billed tokens.
- **Improved caching performance** → compressed assets reuse better across requests.
- **Cost savings** → directly lowers perrequest charges.

Compression Techniques

- **Lossless compression** → preserves all pixels (e.g., PNG optimization).
- **Lossy compression** → discards redundant detail (e.g., JPEG/WebP). Good for thumbnails or iterative drafts.

- **Adaptive resizing** → shrink to the **resolution actually needed** for editing.
- **Format selection** → prefer **WebP/AVIF** for modern, smaller size alternatives to PNG/JPEG.

Python Example: Compress & Resize Before Upload

```
from PIL import Image
# Load a local image
img = Image.open("input.png")
# Resize to a max width of 1024px while maintaining aspect ratio
max_width = 1024
if img.width > max_width:
    ratio = max_width / img.width
    new_size = (max_width, int(img.height * ratio))
    img = img.resize(new_size, Image.ANTIALIAS)
# Save compressed as WebP
img.save("compressed.webp", format="WEBP", quality=80)
print("Image compressed and saved as compressed.webp")
```

Best practice: Preprocess and compress all assets before uploading to Gemini APIs.

API Rate Limiting

Gemini APIs enforce **rate limits** to ensure fairness and stability. Hitting these limits can cause errors or throttled requests.

Typical Limits

- **Requests per minute (RPM)**
- **Tokens per minute (TPM)**
- **Concurrent connections**

Strategies to Handle Rate Limits

- **Clientside throttling** → use libraries like `asyncio` or `ratelimit` in Python to cap requests.
- **Batch requests** → group multiple edits or queries into a single API call.
- **Backoff strategies** → retry with **exponential backoff** when hitting `429 Too Many Requests`.

Python Example: Simple Rate Limiter

```
import time

MAX_REQUESTS_PER_MINUTE = 60

interval = 60 / MAX_REQUESTS_PER_MINUTE

last_call = 0

def rate_limited_call(func, *args, **kwargs):
    global last_call
    now = time.time()
    elapsed = now - last_call
    if elapsed < interval:
        time.sleep(interval - elapsed)
    last_call = time.time()
    return func(*args, **kwargs)

# Example usage: wrap Gemini API calls

# response = rate_limited_call(client.models.generate_content,
# model=MODEL, contents="..." )
```

Cost Management Strategies

Cost control is essential when working with generative AI at scale. Token usage, storage, and request volume all contribute to billing.

Key Drivers of Cost

- **Token count** (input + output)
- **Storage duration** (for cached tokens)
- **Number of requests** (each incurs a base charge)
- **Output image resolution/quality**

Practical Savings Tips

- **Use compression & resizing** → smaller images = fewer tokens.
- **Leverage caching** → reuse contexts and avoid repeating prompts.
- **Batch operations** → fewer requests reduce overhead.
- **Choose the right model tier** → Flash for speed/cost; Pro for precision.
- **Monitor usage** → track with `usage_metadata` in responses.

Example: Monitoring Token Usage

```
response = client.models.generate_content(  
    model="gemini-2.0-flash-001",  
    contents="Describe this image in detail.",  
)  
print("Output:", response.text)  
print("Token usage:", response.usage_metadata)
```

WrapUp

By combining **image compression**, **rate limiting**, and **cost management** strategies, you can build Nano Banana image editing apps that are **fast, reliable, and affordable**. Always preprocess assets, respect API rate limits, and monitor billing metrics closely.

PART 11: Integration and Automation

Webhook Integrations for Automated Workflows

Webhooks allow Gemini (Nano Banana) applications to react automatically to events in real time. Instead of continuously polling the API for changes, you can configure Gemini or related services to send HTTP POST requests to your server whenever an event occurs.

Why Use Webhooks?

- **Automation:** Trigger downstream actions immediately when an event happens.
- **Efficiency:** Reduce API polling and unnecessary requests.
- **Scalability:** Build modular, eventdriven pipelines.
- **Cost savings:** Pay only for relevant calls instead of constant checks.

Common Use Cases

- **Image Upload Triggers:** When a new image is uploaded, automatically kick off preprocessing, compression, or AI editing.
- **Task Completion Notifications:** Start a second workflow when Gemini finishes generating or editing an image.
- **Integration Hooks:** Forward results to other services (e.g., Slack, Google Drive, or analytics pipelines).

Designing a Webhook Listener

A webhook listener is simply an **HTTP endpoint** in your application that receives and processes event payloads.

Key Considerations

- **Security:** Verify signatures or include secret tokens to confirm requests are from trusted sources.
- **Reliability:** Implement retries and idempotency checks to avoid double processing.
- **Logging:** Store payloads for debugging and auditing.

Python Example: Flask Webhook Listener

```
from flask import Flask, request
app = Flask(__name__)
@app.route("/webhook", methods=["POST"])
def handle_webhook():
    data = request.json
    print("Received webhook event:", data)
    # Example: Trigger image enhancement workflow
    if data.get("event_type") == "image.uploaded":
        process_image(data["file_url"])
    return {"status": "success"}, 200
def process_image(file_url):
    # Placeholder for image compression or Gemini editing task
    print(f"Processing uploaded image from {file_url}")
if __name__ == "__main__":
    app.run(port=5000)
```

Best Practices

- **Use HTTPS:** Always encrypt webhook endpoints.
- **Authenticate Events:** Compare secret keys or signatures included in headers.
- **Respond Quickly:** Return a 200 OK immediately, then process in the background.

- **Retry Handling:** Account for duplicate messages.
- **Scalable Queues:** Forward webhook events into a job queue (e.g., Celery, RabbitMQ, Pub/Sub) for robust scaling.

Example Workflow

1. User uploads an image via your app.
1. Gemini (Nano Banana) triggers a webhook to your `/webhook` endpoint.
1. Your service logs the event, compresses the image, and sends it for AI enhancement.
1. Another webhook notifies you when the processed image is ready.
1. Your app stores the final image and updates the user.

WrapUp

Webhooks transform Nano Banana projects into **eventdriven, automated systems**. By responding to image uploads, completions, or other triggers, you can orchestrate entire pipelines with minimal manual effort. Secure, reliable webhook handling is a cornerstone of scalable AI image editing workflows.

Third-Party Service Automation

Third-party automation platforms integrate Gemini's image editing into complex workflows, enabling automated content processing and distribution.

Zapier Integration

Webhook Configuration

Connect Gemini API to Zapier through webhook triggers:

```
import requests

# Send to Zapier webhook

webhook_url = "https://hooks.zapier.com/hooks/catch/YOUR_HOOK"
```



```
payload = {  
    "image_url": generated_image_url,  
    "prompt": original_prompt,  
    "status": "complete"  
}  
  
requests.post(webhook_url, json=payload)
```

Common Zapier Workflows

Trigger	Gemini Action	Output Destination
Email attachment	Image enhancement/editing	Google Drive, Dropbox
Form submission	Product mockup generation	Social media, CMS
Calendar event	Event graphics creation	Email marketing, Slack

Automation Examples

E-commerce Product Processing:

- New product photo uploaded → Gemini generates lifestyle contexts → Images posted to social media
- Customer uploads reference → AI creates product mockups → Results sent via email

Content Marketing:

- Blog post published → Gemini creates featured image → Image added to CMS and social posts
- Event scheduled → AI generates promotional graphics → Distribution across multiple channels

Make.com Integration

HTTP Request Configuration

```
# Configure Make.com webhook endpoint
make_webhook = "https://hook.eu1.make.com/YOUR_SCENARIO_ID"

# Send processed image data
response_data = {
    "original_image": base64_encoded_original,
    "edited_image": base64_encoded_result,
    "edit_type": "style_transfer",
    "parameters": editing_parameters
}

requests.post(make_webhook, json=response_data)
```

Visual Workflow Builder

Make.com's drag-and-drop interface connects Gemini with hundreds of services:

- Google Drive/Sheets integration
- Social media platform posting
- Email marketing automation
- CRM and database updates

Complex Automation Scenarios

Real Estate Marketing:

- Property photo uploaded → Gemini applies virtual staging → Images distributed to MLS, website, social media
- Room photos processed → Multiple style variations created → A/B testing across ad platforms

Social Media Management:

- User-generated content received → AI enhances/styles images → Scheduled posting across platforms
- Brand assets updated → Gemini creates variations → Asset library updated automatically

Technical Implementation

API Authentication

Store credentials securely in automation platforms:

```
# Environment variable usage
import os

GEMINI_API_KEY = os.environ.get('GEMINI_API_KEY')
```

Error Handling

```
try:
    response = client.models.generate_content(...)
    # Send success to automation platform
    send_to_zapier({"status": "success", "image": response})
except Exception as e:
    # Send error to automation platform
    send_to_zapier({"status": "error", "message": str(e)})
```

Workflow Optimization

Batch Processing Integration

Volume	Platform Choice	Processing Method
1-10 images	Zapier	Real-time processing
10-100 images	Make.com	Scheduled batches

100+ images

Custom API + batch mode

Asynchronous processing

Cost Management

- Monitor API usage through automation platforms
- Implement rate limiting for high-volume workflows
- Use appropriate models for task complexity
- Cache results to avoid duplicate processing

Performance Monitoring

Track workflow success rates and processing times through automation platform analytics and custom logging.

These integrations enable automated image processing pipelines that scale beyond manual operation while maintaining consistent quality and brand standards.

Automated Backup and Version Control Systems

In production environments, safeguarding your assets and maintaining a reliable history of changes is critical. **Automated backup and version control systems** ensure that image data, metadata, and project files in Gemini (Nano Banana) workflows remain protected, auditable, and recoverable.

Why Backups and Version Control Matter

- **Data Protection:** Prevent accidental loss from bugs, user error, or hardware failure.
- **Auditability:** Keep a traceable history of all changes for compliance or debugging.
- **Collaboration:** Allow teams to work on the same project safely with rollback options.

- **Disaster Recovery:** Quickly restore systems after outages.

Backup Strategies

Local Backups

- Keep copies of images and metadata on local disks.
- Simple but limited; vulnerable to disk failure.

Cloud Backups

- Store files in **Google Cloud Storage**, **AWS S3**, or **Azure Blob**.
- Provide durability, redundancy, and global accessibility.

Scheduled Automation

- Nightly or weekly backups reduce manual effort.
- Tools like `cron`, Airflow, or Cloud Scheduler can automate backup tasks.

Version Control Approaches

File Versioning

- Save multiple versions of the same image (e.g., `image_v1.webp`, `image_v2.webp`).
- Good for small projects or oneoff experiments.

Git for Metadata and Configurations

- Track prompt scripts, configuration files, and workflow definitions.
- Enables collaboration with branching and merging.

Database Versioning

- Store structured metadata in databases with version fields.
- Example: each image record includes `revision_number` and `last_modified`.

Python Example: Automated Backup Script

```
import shutil, datetime

src = "output/final_image.webp"

dst = f"backups/final_image_{datetime.date.today()}.webp"

shutil.copy(src, dst)

print(f"Backup created: {dst}")
```

Python Example: Git Integration for Metadata

```
import subprocess

# Save changes to Git repository for metadata/config files

subprocess.run(["git", "add", "metadata.json"], check=True)

subprocess.run(["git", "commit", "-m", "Automated commit of latest metadata"], check=True)

subprocess.run(["git", "push"], check=True)

print("Metadata version committed and pushed.")
```

Best Practices

- **Encrypt Backups:** Protect sensitive data at rest and in transit.
- **Test Restores:** Regularly validate that backups can be restored.
- **Retention Policies:** Define how long backups are kept (e.g., 30/60/90 days).
- **Hybrid Approach:** Combine cloud redundancy with local fast restores.
- **Automate Everything:** Minimize human error with scheduled jobs.

WrapUp

Automated backups and version control safeguard Nano Banana image editing projects from data loss, ensure smooth collaboration, and support longterm reliability. By combining **scheduled backups, cloud storage, and version tracking**, you create a resilient foundation for production AI workflows.

Database Integration for Image Metadata Management

Effective metadata management is the backbone of any professional AI image editing workflow. In Nano Banana's integration with Google Gemini, every image operation, transformation, and enhancement generates valuable metadata that must be captured, stored, and retrieved efficiently. This chapter explores comprehensive database integration strategies that ensure your image editing pipeline remains performant, scalable, and intelligently organized.

Understanding Image Metadata in AI Workflows

Core Metadata Categories

Technical Metadata

- Image dimensions, format, color depth, compression settings
- Camera/device information (EXIF data)
- File size, creation/modification timestamps
- Checksum values for integrity verification

AI Processing Metadata

- Gemini model version and configuration used
- Processing timestamps and duration
- Applied filters, enhancements, and transformations
- Confidence scores for automated operations
- Processing parameters and settings

Workflow Metadata

- User annotations and tags

- Project associations and versioning
- Approval status and review history
- Collaborative editing sessions and contributors

Business Metadata

- Usage rights and licensing information
- Asset categorization and taxonomy
- Performance metrics and analytics
- Cost tracking for API usage

Database Architecture Design

Core Schema Structure

The foundation of your metadata system starts with a well-designed relational schema. The primary images table captures essential file information:

```
-- Core Images Table
CREATE TABLE images (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    filename VARCHAR(255) NOT NULL,
    original_path TEXT,
    file_hash SHA256 UNIQUE NOT NULL,
    mime_type VARCHAR(50),
    file_size BIGINT,
    width INTEGER,
    height INTEGER,
    color_mode VARCHAR(20),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
```



```
);
```

This schema uses UUID for globally unique identifiers and includes file hashing for duplicate detection. The timestamps provide audit trails for all image entries.

AI Processing History Tracking

To track all Gemini AI operations, create a dedicated processing sessions table:

```
-- AI Processing History
CREATE TABLE ai_processing_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    image_id UUID REFERENCES images(id) ON DELETE CASCADE,
    gemini_model_version VARCHAR(50),
    processing_type VARCHAR(100),
    input_parameters JSONB,
    output_parameters JSONB,
    processing_duration_ms INTEGER,
    status VARCHAR(20) DEFAULT 'pending',
    error_message TEXT,
    started_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    completed_at TIMESTAMP WITH TIME ZONE
);
```

This table captures every AI operation with full parameter tracking. The JSONB columns provide flexible storage for complex Gemini inputs and outputs while maintaining queryability.

Flexible Metadata Storage

For extensible metadata that doesn't fit standard columns, implement a key-value metadata table:

```
-- Flexible Metadata Store
```

```
CREATE TABLE image_metadata (  
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
    image_id UUID REFERENCES images(id) ON DELETE CASCADE,  
    metadata_type VARCHAR(50),  
    key VARCHAR(100),  
    value JSONB,  
    source VARCHAR(50), -- 'user', 'gemini', 'exif', 'system'  
    confidence_score DECIMAL(3,2),  
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()  
);
```

This flexible design accommodates various metadata types from different sources while tracking confidence levels for AI-generated data.

Performance Indexing Strategy

Critical indexes ensure fast queries across large image collections:

```
-- Indexing Strategy  
  
CREATE INDEX idx_images_hash ON images(file_hash);  
  
CREATE INDEX idx_images_created ON images(created_at);  
  
CREATE INDEX idx_metadata_type_key ON image_metadata(metadata_type,  
key);  
  
CREATE INDEX idx_metadata_image_id ON image_metadata(image_id);  
  
CREATE INDEX idx_processing_image_status ON  
ai_processing_sessions(image_id, status);
```

These indexes optimize common query patterns: duplicate detection, chronological browsing, metadata searches, and processing status checks.

Python Implementation with SQLAlchemy

Model Definitions

SQLAlchemy provides elegant object-relational mapping for your metadata system:

```
from sqlalchemy import create_engine, Column, String, Integer,
DateTime, JSON, ForeignKey

from sqlalchemy.ext.declarative import declarative_base

from sqlalchemy.orm import sessionmaker, relationship

from sqlalchemy.dialects.postgresql import UUID, JSONB

import uuid

from datetime import datetime

Base = declarative_base()

class Image(Base):
    __tablename__ = 'images'

    id = Column(UUID(as_uuid=True), primary_key=True,
default=uuid.uuid4)

    filename = Column(String(255), nullable=False)
    file_hash = Column(String(64), unique=True, nullable=False)
    mime_type = Column(String(50))
    width = Column(Integer)
    height = Column(Integer)
    created_at = Column(DateTime, default=datetime.utcnow)
    updated_at = Column(DateTime, default=datetime.utcnow,
onupdate=datetime.utcnow)

    # Relationships

    processing_sessions = relationship("AIProcessingSession",
back_populates="image")

    metadata_entries = relationship("ImageMetadata",
back_populates="image")
```

This model establishes the core image entity with automatic timestamping and relationship mappings to dependent tables.

AI Processing Session Model

Track every Gemini operation with detailed session information:

```
class AIProcessingSession(Base):
    __tablename__ = 'ai_processing_sessions'
    id = Column(UUID(as_uuid=True), primary_key=True,
default=uuid.uuid4)
    image_id = Column(UUID(as_uuid=True), ForeignKey('images.id'),
nullable=False)
    gemini_model_version = Column(String(50))
    processing_type = Column(String(100))
    input_parameters = Column(JSONB)
    output_parameters = Column(JSONB)
    processing_duration_ms = Column(Integer)
    status = Column(String(20), default='pending')
    started_at = Column(DateTime, default=datetime.utcnow)
    # Relationships
    image = relationship("Image",
back_populates="processing_sessions")
class ImageMetadata(Base):
    __tablename__ = 'image_metadata'
    id = Column(UUID(as_uuid=True), primary_key=True,
default=uuid.uuid4)
    image_id = Column(UUID(as_uuid=True), ForeignKey('images.id'),
nullable=False)
    metadata_type = Column(String(50))
    key = Column(String(100))
    value = Column(JSONB)
```

```
source = Column(String(50))

confidence_score = Column(String(5))

created_at = Column(DateTime, default=datetime.utcnow)

image = relationship("Image", back_populates="metadata_entries")
```

These models capture the complete processing lifecycle and provide flexible metadata storage with source attribution.

Metadata Manager Implementation

Create a comprehensive manager class for metadata operations:

```
class MetadataManager:

    def __init__(self, database_url: str):

        self.engine = create_engine(database_url)

        self.Session = sessionmaker(bind=self.engine)

    def store_image_metadata(self, image_id: str, metadata: dict,
source: str = 'system'):

        """Store comprehensive image metadata"""

        session = self.Session()

        try:

            # Store technical metadata

            for key, value in metadata.get('technical', {}).items():

                meta_entry = ImageMetadata(

                    image_id=image_id,

                    metadata_type='technical',

                    key=key,

                    value=value,

                    source=source

                )
```

```
        session.add(meta_entry)

    session.commit()

    return True

except Exception as e:

    session.rollback()

    raise e

finally:

    session.close()
```

This method handles transactional metadata storage with proper error handling and rollback capabilities.

AI Processing Session Storage

Handle Gemini-specific processing results:

```
def store_ai_session(self, image_id: str, ai_data: dict) -> str:
    """Store AI processing session with full parameter
    tracking"""
    session = self.Session()
    try:
        processing_session = AIProcessingSession(
            image_id=image_id,
            gemini_model_version=ai_data.get('model_version'),
            processing_type=ai_data.get('operation_type'),
            input_parameters=ai_data.get('input_params'),
            output_parameters=ai_data.get('results'),
            processing_duration_ms=ai_data.get('duration_ms'),
            status='completed'
        )
```

```
        session.add(processing_session)

        session.commit()

        return str(processing_session.id)

    except Exception as e:

        session.rollback()

        raise e

    finally:

        session.close()
```

This method captures complete AI processing context, enabling detailed analytics and debugging capabilities.

Gemini Integration Pipeline

Metadata Extraction Setup

Initialize the Gemini-powered metadata extractor:

```
import google.generativeai as genai
from typing import Dict, Any
import json

class GeminiMetadataExtractor:

    def __init__(self, api_key: str, metadata_manager:
MetadataManager):

        genai.configure(api_key=api_key)

        self.model = genai.GenerativeModel('gemini-1.5-pro')

        self.metadata_manager = metadata_manager
```

This class bridges Gemini's AI capabilities with your database infrastructure.

Comprehensive Metadata Analysis

Create a structured prompt for consistent metadata extraction:

```
async def extract_and_store_metadata(self, image_path: str,
image_id: str) -> Dict[str, Any]:

    """Extract metadata using Gemini and store in database"""
    with open(image_path, 'rb') as img_file:
        image_data = img_file.read()
    prompt = """
    Analyze this image and provide structured metadata in JSON
format:
    {
        "visual_content": {
            "primary_subjects": [],
            "scene_type": "",
            "dominant_colors": [],
            "composition_style": "",
            "lighting_conditions": ""
        },
        "technical_assessment": {
            "image_quality": "",
            "resolution_adequacy": "",
            "exposure_evaluation": "",
            "focus_assessment": ""
        },
        "editing_recommendations": {
            "suggested_enhancements": [],
            "color_correction_needed": boolean,
            "crop_recommendations": ""
        }
    }
    """
```



```
        "priority_score": 1-10
    }
}
"""
```

This structured prompt ensures consistent, machine-readable metadata across all processed images.

Error Handling and Metadata Storage

Implement robust error handling with metadata persistence:

```
try:
    # Generate metadata with Gemini
    response = self.model.generate_content([prompt,
image_data])

    metadata_json = json.loads(response.text)
    # Enhance with processing metadata
    enhanced_metadata = {
        'ai_analysis': {
            'model_version': 'gemini-1.5-pro',
            'operation_type': 'metadata_extraction',
            'results': metadata_json,
            'input_params': {'prompt_type':
'comprehensive_analysis'},
            'confidence_score':
self._calculate_overall_confidence(metadata_json)
        }
    }

    # Store in database
    self.metadata_manager.store_image_metadata(
```

```
        image_id, enhanced_metadata, source='gemini'
    )

    return enhanced_metadata
except Exception as e:
    # Store failure metadata
    error_metadata = {
        'ai_analysis': {
            'status': 'failed',
            'error': str(e)
        }
    }

    self.metadata_manager.store_image_metadata(
        image_id, error_metadata, source='gemini'
    )

    raise
```

Even failed operations are logged for troubleshooting and system monitoring.

Advanced Query Operations

Similarity Search Implementation

Find images with comparable processing characteristics:

```
class ImageQueryEngine:
    def __init__(self, metadata_manager: MetadataManager):
        self.manager = metadata_manager

    def find_similar_processing(self, image_id: str, operation_type:
str) -> list:
        """Find images with similar AI processing history"""
```

```
session = self.manager.Session()

try:
    results = session.query(Image, AIProcessingSession)\
        .join(AIProcessingSession)\
        .filter(AIProcessingSession.processing_type ==
operation_type)\
        .filter(Image.id != image_id)\
        .order_by(AIProcessingSession.started_at.desc())\
        .limit(10)\
        .all()

    return [(img.filename, session_obj.output_parameters)
for img, session_obj in results]

finally:
    session.close()
```

This query identifies images that underwent similar processing, enabling pattern analysis and workflow optimization.

Tag-Based Search with Confidence Filtering

Implement intelligent search based on AI-generated tags:

```
def search_by_ai_tags(self, tags: list, confidence_threshold:
float = 0.7) -> list:

    """Search images by AI-generated tags with confidence
filtering"""

    session = self.manager.Session()

    try:
        query = session.query(Image)\
            .join(ImageMetadata)\
            .filter(ImageMetadata.metadata_type == 'ai_tags')\
```

```
        .filter(ImageMetadata.confidence_score >=
confidence_threshold)

        # Dynamic tag filtering
        for tag in tags:
            query = query.filter(ImageMetadata.value.op('?')
(tag))

        return [img.filename for img in query.all()]
    finally:
        session.close()
```

This method leverages PostgreSQL's JSONB operators to search within nested tag structures while filtering by AI confidence scores.

Performance Analytics Generation

Track processing performance across time periods:

```
def performance_analytics(self, date_range: tuple) -> dict:
    """Generate processing performance analytics"""
    session = self.manager.Session()
    try:
        from sqlalchemy import func
        stats = session.query(

func.count(AIProcessingSession.id).label('total_operations'),

func.avg(AIProcessingSession.processing_duration_ms).label('avg_dura
tion'),

        AIProcessingSession.processing_type,
        AIProcessingSession.gemini_model_version
    )\
```

```
.filter(AIProcessingSession.started_at.between(*date_range))\
    .group_by(AIProcessingSession.processing_type,
AIProcessingSession.gemini_model_version)\
    .all()
return {
    'summary': [
        {
            'operation': stat.processing_type,
            'model': stat.gemini_model_version,
            'count': stat.total_operations,
            'avg_duration_ms': float(stat.avg_duration)
        }
        for stat in stats
    ]
}
finally:
    session.close()
```

These analytics help identify performance bottlenecks and optimize processing workflows.

Batch Processing Implementation

Concurrent Processing Setup

Handle multiple images efficiently with proper rate limiting:

```
import asyncio
from concurrent.futures import ThreadPoolExecutor
```

```
class BatchMetadataProcessor:
    def __init__(self, gemini_extractor: GeminiMetadataExtractor,
max_workers: int = 5):
        self.extractor = gemini_extractor
        self.max_workers = max_workers

    async def process_image_batch(self, image_paths: list) ->
Dict[str, Any]:
        """Process multiple images with rate limiting and error
handling"""
        results = {
            'successful': [],
            'failed': [],
            'summary': {}
        }
        # Create semaphore for rate limiting
        semaphore = asyncio.Semaphore(self.max_workers)
```

The semaphore pattern prevents overwhelming the Gemini API while maintaining optimal throughput.

Individual Image Processing

Process each image with proper error isolation:

```
async def process_single_image(image_path: str):
    async with semaphore:
        try:
            # Generate unique image ID
            image_id = str(uuid.uuid4())
            # Extract metadata
```

```
        metadata = await
self.extractor.extract_and_store_metadata(
    image_path, image_id
)
    results['successful'].append({
        'image_id': image_id,
        'path': image_path,
        'metadata': metadata
    })
except Exception as e:
    results['failed'].append({
        'path': image_path,
        'error': str(e)
    })

# Process all images concurrently

await asyncio.gather(*[process_single_image(path) for path
in image_paths])
```

This approach ensures that individual failures don't cascade to other images in the batch.

Performance Optimization with Caching

Redis Cache Implementation

Implement intelligent caching for frequently accessed metadata:

```
from redis import Redis
import pickle
from functools import wraps
```

```
class MetadataCache:
    def __init__(self, redis_url: str = 'redis://localhost:6379'):
        self.redis_client = Redis.from_url(redis_url)
        self.default_ttl = 3600 # 1 hour

    def cache_metadata(self, ttl: int = None):
        """Decorator for caching metadata operations"""
        def decorator(func):
            @wraps(func)
            def wrapper(self, *args, **kwargs):
                # Generate cache key from function name and
arguments
                cache_key = f"metadata:{func.__name__}:
{hash(str(args) + str(kwargs))}"
                # Try to get from cache
                cached_result = self.redis_client.get(cache_key)
                if cached_result:
                    return pickle.loads(cached_result)
                # Execute function and cache result
                result = func(self, *args, **kwargs)
                self.redis_client.setex(
                    cache_key,
                    ttl or self.default_ttl,
                    pickle.dumps(result)
                )
                return result
            return wrapper
        return decorator
```


This caching decorator dramatically improves response times for repeated metadata queries.

Cache Invalidation Strategy

Implement targeted cache invalidation for data consistency:

```
def invalidate_image_cache(self, image_id: str):
    """Clear all cached data for a specific image"""
    pattern = f"metadata:*{image_id}*"
    keys = self.redis_client.keys(pattern)
    if keys:
        self.redis_client.delete(*keys)

class CachedMetadataManager(MetadataManager):
    def __init__(self, database_url: str, cache: MetadataCache):
        super().__init__(database_url)
        self.cache = cache

    @cache.cache_metadata(ttl=1800)
    def get_image_summary(self, image_id: str) -> dict:
        """Get cached image summary with metadata"""
        return self.retrieve_image_history(image_id)

    def update_image_metadata(self, image_id: str, new_metadata:
dict):
        """Update metadata and invalidate relevant caches"""
        super().store_image_metadata(image_id, new_metadata)
        self.cache.invalidate_image_cache(image_id)
```

This ensures cache consistency when metadata changes while maintaining performance benefits.

NoSQL Alternative with MongoDB

Document-Based Schema

For high-volume scenarios, consider MongoDB's flexible document structure:

javascript

```
// MongoDB Schema Design
const ImageSchema = {
  _id: ObjectId,
  filename: String,
  fileHash: String,
  technical: {
    dimensions: { width: Number, height: Number },
    format: String,
    colorProfile: String,
    exifData: Object
  },
  aiProcessing: [{
    sessionId: String,
    modelVersion: String,
    operation: String,
    parameters: Object,
    results: Object,
    timing: {
      started: Date,
      completed: Date,
      duration: Number
    }
  }]
}
```

```
    },  
    confidence: Number  
  ]],  
  userMetadata: {  
    tags: [String],  
    categories: [String],  
    annotations: [Object]  
  },  
  workflow: {  
    projectId: String,  
    version: Number,  
    status: String,  
    collaborators: [String]  
  },  
  createdAt: Date,  
  updatedAt: Date  
}
```

This schema embeds related data, reducing joins while maintaining query flexibility. The `aiProcessing` array tracks all Gemini operations chronologically.

Advanced Integration Patterns

Confidence Score Calculation

Implement intelligent confidence assessment for AI-generated metadata:

```
def _calculate_overall_confidence(self, metadata: dict) -> float:  
    """Calculate weighted confidence score from Gemini analysis"""  
    confidence_scores = []  
    # Extract confidence values from nested structures
```

```
def extract_confidences(obj):
    if isinstance(obj, dict):
        for key, value in obj.items():
            if 'confidence' in key.lower() and isinstance(value,
(int, float)):
                confidence_scores.append(value)
            elif isinstance(value, (dict, list)):
                extract_confidences(value)
    elif isinstance(obj, list):
        for item in obj:
            extract_confidences(item)
    extract_confidences(metadata)
    return sum(confidence_scores) / len(confidence_scores) if
confidence_scores else 0.5
```

This recursive function extracts confidence scores from complex nested metadata structures, providing overall reliability metrics.

Real-Time Sync Implementation

Maintain database consistency across distributed systems:

```
from sqlalchemy.event import listens_for
import json
@listens_for(ImageMetadata, 'after_insert')
def sync_metadata_insert(mapper, connection, target):
    """Trigger sync operations after metadata insertion"""
    # Publish to message queue for real-time updates
    sync_message = {
        'action': 'metadata_created',
```

```
'image_id': str(target.image_id),
'metadata_type': target.metadata_type,
'timestamp': target.created_at.isoformat()
}
# Example: Redis pub/sub or RabbitMQ
publish_sync_event('metadata_updates', sync_message)
def publish_sync_event(channel: str, message: dict):
    """Publish synchronization events to message queue"""
    import redis
    r = redis.Redis()
    r.publish(channel, json.dumps(message))
```

SQLAlchemy events enable automatic synchronization triggers without application-level complexity.

Monitoring and Analytics

Query Performance Tracking

Monitor database performance for optimization opportunities:

```
import time
from contextlib import contextmanager
class PerformanceTracker:
    def __init__(self):
        self.query_stats = {}
    @contextmanager
    def track_query(self, operation_name: str):
        """Context manager for tracking query performance"""
        start_time = time.time()
```

```
try:
    yield
finally:
    duration = (time.time() - start_time) * 1000 #
milliseconds

    if operation_name not in self.query_stats:
        self.query_stats[operation_name] = {
            'count': 0,
            'total_duration': 0,
            'max_duration': 0
        }

    stats = self.query_stats[operation_name]
    stats['count'] += 1
    stats['total_duration'] += duration
    stats['max_duration'] = max(stats['max_duration'],
duration)

def get_performance_report(self) -> dict:
    """Generate performance analytics report"""
    report = {}
    for operation, stats in self.query_stats.items():
        report[operation] = {
            'avg_duration_ms': stats['total_duration'] /
stats['count'],
            'max_duration_ms': stats['max_duration'],
            'total_queries': stats['count']
        }
    return report
```

This tracker identifies slow queries and provides data for optimization decisions.

Best Practices and Optimization Tips

Connection Pooling

Configure proper connection pooling for production environments:

```
from sqlalchemy.pool import QueuePool

engine = create_engine(
    database_url,
    poolclass=QueuePool,
    pool_size=20,
    max_overflow=30,
    pool_pre_ping=True,
    pool_recycle=3600
)
```

These settings ensure efficient database connection management under load.

Metadata Cleanup Strategies

Implement automated cleanup for outdated metadata:

```
def cleanup_old_metadata(self, retention_days: int = 90):
    """Remove metadata older than retention period"""
    session = self.Session()
    try:
        cutoff_date = datetime.utcnow() -
timedelta(days=retention_days)
        # Archive before deletion
        old_sessions = session.query(AIProcessingSession)\
            .filter(AIProcessingSession.started_at < cutoff_date)\
```

```
        .all()

        # Export to archive storage
        archive_data = [session.to_dict() for session in
old_sessions]

        self.export_to_archive(archive_data)

        # Delete from active database
        session.query(AIProcessingSession)\

            .filter(AIProcessingSession.started_at < cutoff_date)\

            .delete()

        session.commit()

    finally:

        session.close()
```

Regular cleanup prevents database bloat while preserving historical data in archive systems.

Conclusion

Effective database integration for image metadata management requires careful planning across schema design, performance optimization, and integration patterns. The combination of structured relational data for core entities and flexible JSONB storage for AI-generated metadata provides the ideal foundation for Nano Banana's Gemini-powered workflows.

Key implementation priorities include robust error handling, intelligent caching, comprehensive indexing, and automated cleanup strategies. These patterns ensure your metadata system scales efficiently while providing the rich search and analytics capabilities essential for professional AI image editing workflows.

The next chapter will explore real-time collaboration features that leverage this metadata infrastructure for multi-user editing environments.

Epilogue: Your AI Image Mastery

You've mastered comprehensive expertise spanning creative applications, technical implementation, and enterprise deployment of Google's AI image generation technology.

You've learned advanced prompt engineering that delivers consistent professional results. Master identity preservation and character consistency across image series. Dive into sophisticated multi-image composition and style transfer techniques.

Your development expertise covers complete Python SDK implementation. Learn to navigate Google Colab, AI Studio, and Vertex AI platforms effectively. Master authentication systems and secure credential management. Dive into batch processing and context caching for optimal performance.

Learn to transform business marketing through AI-generated brand consistency. Master e-commerce and real estate photography that drives results. Dive into social media content creation that scales effortlessly.

Your capabilities now include:

- Professional profile optimization with powerful impact
- Marketing materials that rival expensive agency work
- Product photography that increases conversion rates
- Custom app development with seamless AI integration

Lead the Visual AI Revolution

You've transformed from curious observer to expert practitioner. Learn to implement professional-grade AI image systems across industries. Master creative applications, commercial development, and enterprise solutions.

Dive into the future of visual content creation. You're positioned to lead this transformation.

Get the FREE Online Course & Certificate

With this book in hand, you're unlocking a world of opportunity — including instant lifetime access to a FREE online course and exam on Mammoth Club!



Scan the QR code to redeem your free course, exam and cheatsheet! Or go to to this link:

mammothclub.com/course/1-hour-nano/BANANA

You'll also earn a Certificate of Achievement for completing the online course. Join our thriving community of learners spanning 190+ countries, and be part of the 9 million+ courses sold around the globe. Sign up today for free!



Alex Kropf is Mammoth Club's CLO, public speaker, consultant, IT author and Senior Software Developer. Alex has produced 1,000+ best-selling courses, books and workshops for Mammoth Club, Course Pro and clients worldwide.

Mammoth Club is a leading online course provider in everything from learning to code to becoming a YouTube star. Since 2011, Mammoth Club has built a global student community with over 9 million courses sold.



John Bura is Founder and CEO of global tech giant Mammoth Club and viral app Course Pro, the #1 AI-powered Learning Management System for course and content development, training and evaluation.

Neither the author or publisher of this book nor AI itself can be held responsible if you accidentally step on any copyright toes, overspend your API billing limits, or send confidential data to a compromised chatbot. By flipping through these pages and using AI, you agree to hold yourself entirely responsible for what you do with your AI-powered creations. This book is brought to you by Mammoth Club — it's not connected to or sponsored by any other company. Everything here is just the author's perspective, not any company's official view.

Visit MammothClub.com

for free online video courses, ebooks, source code, customer support and MORE!

To build and sell your own courses, presentations, books and videos: visit #1 course creation platform:

CoursePro.ai



MAMMOTH CLUB