

SOCIAL YOLO

AI Architecture & Development Approach Report

1. Executive Summary

Social Yolo is an AI-powered creative generation platform designed for product-based businesses. The platform will allow users to upload product images and generate high-quality social media creatives based on their product, brand identity, selected visual style, platform, and marketing requirements.

For the initial version, the recommended approach is not to train custom AI models from scratch. Instead, Social Yolo should integrate existing AI models and APIs for specialized tasks such as image understanding, background removal, image enhancement, copy generation, and image generation.

The core of the system will be an AI orchestration layer within the backend. This layer will coordinate different AI services and combine their outputs to produce the final creative.

The initial architecture should focus on:

- Existing AI APIs/models instead of custom model training
- NestJS as the backend/orchestration layer
- React/Next.js for the frontend
- PostgreSQL for structured application and brand data
- Object storage for images and generated assets
- Image-generation APIs/models for creative generation
- LLMs for copy generation and creative planning
- Vision models for product/image analysis
- Background-removal and enhancement services for product preparation
- RAG and vector databases as an optional layer for brands with large amounts of knowledge
- Programmatic image composition for precise text, logos, CTA placement, and final export

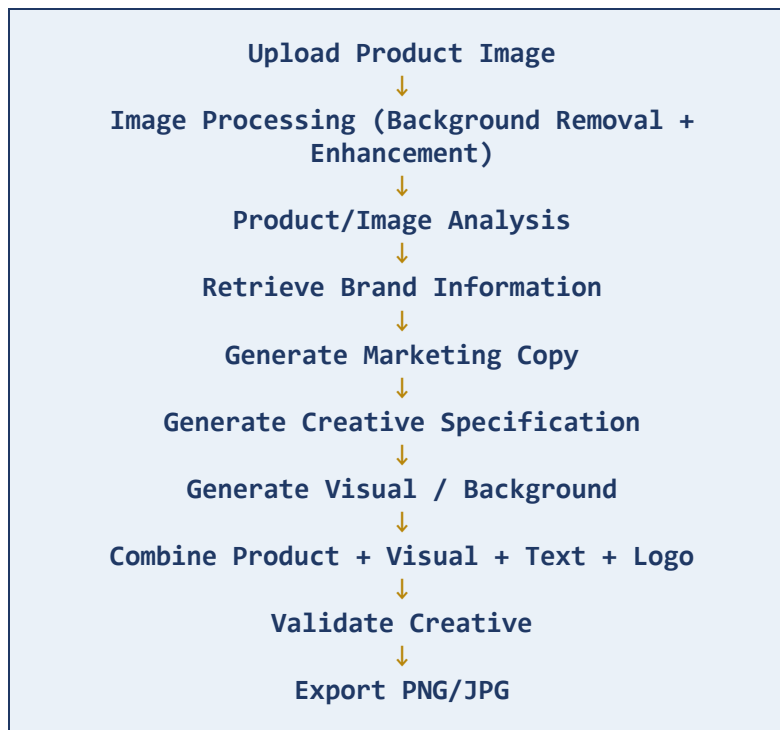
Video generation is intentionally excluded from the initial implementation.

2. Recommended Product Approach

The main idea is to treat Social Yolo as an AI-powered creative orchestration platform, rather than trying to build one large AI model that performs every task.

A typical generation flow will be:



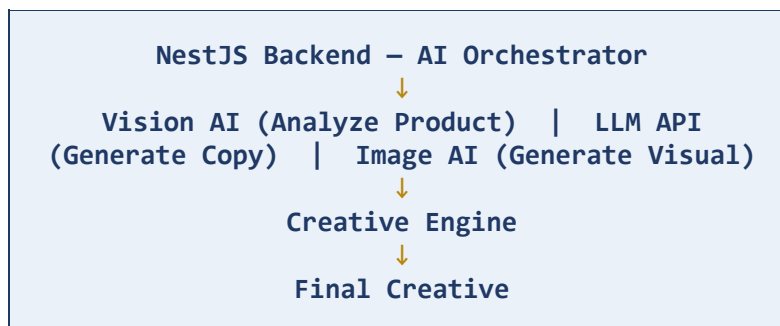


Different AI models/services will be used for different parts of this workflow.

3. What is AI Orchestration?

AI orchestration means coordinating multiple AI models, APIs, databases, and application services to complete a larger task. Instead of asking one AI model to perform everything, the backend acts as an orchestrator.

For example:



The orchestrator determines which services are required, sends the appropriate input to each service, receives the output, and passes relevant information to the next stage.

This approach makes the system modular and allows individual AI components to be replaced or upgraded without redesigning the entire platform.

4. Why We Should Use Existing AI APIs/Models Initially

For V1, training our own AI models is not recommended. Existing models can already perform many of the required tasks at a high level of quality.

Potential technologies include:

Requirement	Possible Technology
Background Removal	Remove.bg, Photoroom, RMBG/open-source models
Image Enhancement	Specialized enhancement APIs/models, Replicate-hosted models, etc.
Product/Image Analysis	Vision-capable LLMs
Copy Generation	GPT, Claude, Gemini, or other LLMs
Image Generation	OpenAI image models, FLUX, Ideogram, Gemini image models, etc.
Embeddings	OpenAI, Cohere, Voyage, open-source embedding models
Vector Database	pgvector, Pinecone, Qdrant
Image Storage	Azure Blob Storage, S3, Cloudinary
Backend	NestJS
Frontend	React / Next.js
Database	PostgreSQL

The exact AI provider should be finalized after evaluating quality, cost, API availability, latency, rate limits, and licensing.

5. Advantages of Using Existing APIs/Models

5.1 Faster Development

We do not need to spend months collecting datasets, training models, tuning them, and maintaining ML infrastructure.

5.2 Better Initial Quality

Many existing image and language models are already highly capable and can provide strong results without custom training.

5.3 Lower Initial Engineering Complexity

The development team can focus on the actual Social Yolo product instead of building an AI model infrastructure from scratch.

5.4 Easy Experimentation

Different models can be tested and compared, for example Image Model A vs Image Model B vs Image Model C. The best model can then be selected based on quality and cost.

5.5 Easier MVP Development

The first version can be launched quickly, allowing us to collect real user feedback before making expensive AI infrastructure decisions.

6. Disadvantages of Using APIs

6.1 Recurring API Costs

Every AI generation or processing operation may incur a cost. A single creative could potentially involve multiple API calls: Background Removal + Image Enhancement + Vision Analysis + Copy Generation + Image Generation. If the number of users becomes large, these costs can become significant.

6.2 Vendor Dependency

The system may depend on external providers. Changes in the following can affect the product:

- Pricing
- API limits
- Model availability
- Model behavior
- Terms/licensing

6.3 Latency

Multiple sequential API calls can increase generation time.

6.4 Data Privacy Considerations

User images, brand assets, and other information may be sent to third-party providers. Provider policies, data handling, retention, and compliance requirements must therefore be reviewed before selecting services.

6.5 Less Control

With third-party models, we do not have complete control over model behavior and underlying infrastructure.

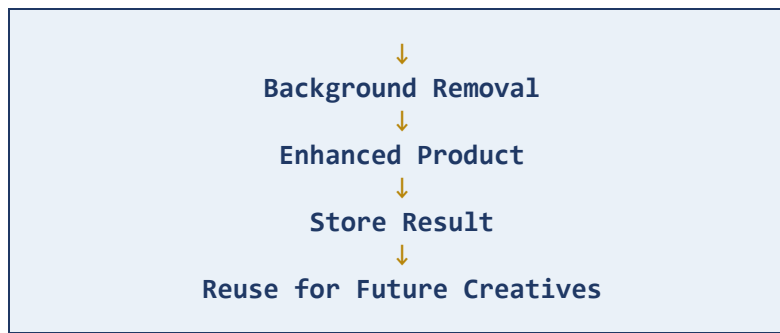
7. Cost Optimization Strategy

API usage does not necessarily mean that the product will be unprofitable. The system should be designed to minimize unnecessary AI calls.

7.1 Caching

If the same product image has already been processed, we should reuse the processed version rather than calling the image-processing API again.

Original Product



7.2 Model Selection Based on Task Complexity

Not every operation requires the most expensive model. For example:

- Simple classification → cheaper model
- Basic copy generation → cheaper LLM
- Complex creative planning → stronger LLM
- Final image generation → specialized image model

7.3 Perform Simple Operations Internally

Tasks such as the following can be performed using our own application/server rather than calling an AI API:

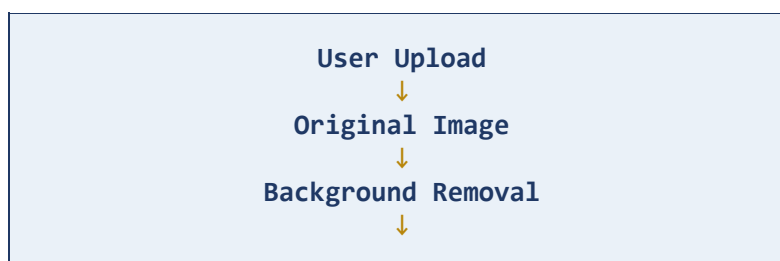
- Resizing
- Adding logos
- Placing text
- Compositing images
- Converting PNG/JPG
- Generating platform dimensions

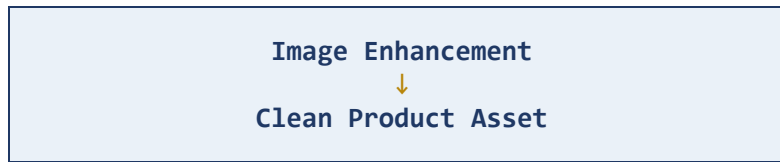
7.4 Credits-Based Pricing

The product can use a subscription + credits model. For example: 1 Creative Generation = X Credits, 1 Image Enhancement = Y Credits. This allows the business to control API costs while charging users based on their usage.

8. Product Image Processing

The first stage of the creative workflow should prepare the user's product image. The pipeline can be:





Possible technologies include background-removal APIs or open-source models such as RMBG. Image enhancement can similarly be handled through specialized APIs/models. The processed product image should be stored and reused for future generations.

9. Product/Image Understanding

Before generating the creative, the system should understand the uploaded product. A vision-capable AI model can analyze the image and produce structured information. For example:

```
{
  "category": "perfume",
  "dominant_colors": ["black", "gold"],
  "visual_style": "premium",
  "characteristics": [
    "glass bottle",
    "gold cap",
    "minimal label"
  ]
}
```

This information can then be passed to the creative-generation pipeline. This allows the system to generate designs based on the actual product rather than treating every uploaded image identically.

10. Brand Information

Social Yolo should maintain a structured brand profile, including:

- Brand Name
- Logo
- Brand Colors
- Typography Preferences
- Visual Style
- Tone of Voice
- Industry
- Target Audience
- Other Brand Preferences

Basic brand information can be stored directly in PostgreSQL (Brand → Name, Colors, Logo, Style, Tone, Target Audience). For this type of structured information, a normal relational database is sufficient and RAG is not required.

11. What Are Embeddings?

An embedding is a numerical representation of the meaning of information. For example, “Luxury perfume” can be converted into a numerical vector such as [0.21, -0.45, 0.78, 0.12, ...]. A similar phrase such as “Premium fragrance” would produce a mathematically similar vector.

This allows the system to search information based on meaning, rather than only exact keywords.

12. Why Embeddings May Be Needed

Embeddings become useful when Social Yolo stores a large amount of unstructured brand knowledge. For example, a company may upload:

- Brand Guidelines
- Website content
- Product Catalog
- Marketing Documents
- Previous Campaigns
- Customer Personas
- Product Descriptions
- FAQs

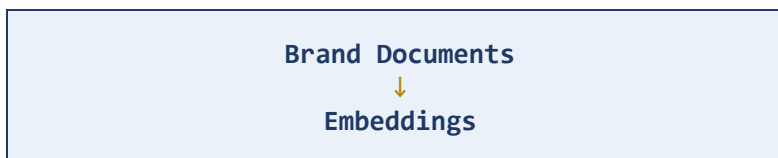
Instead of sending all this information to the LLM every time, the system can convert the documents into embeddings and store them in a vector database. Potential vector databases include:

- pgvector
- Pinecone
- Qdrant

13. RAG in Social Yolo

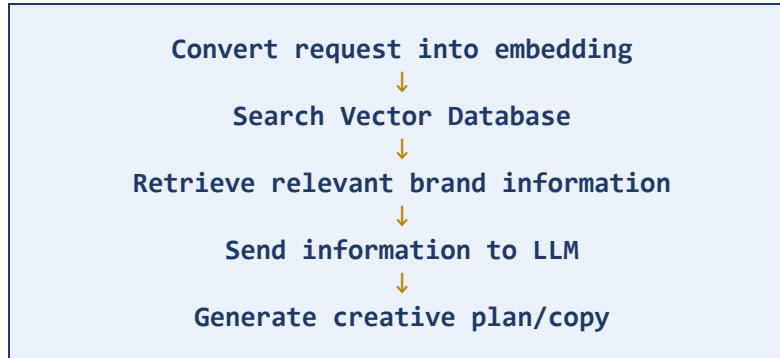
RAG stands for Retrieval-Augmented Generation. In simple terms: RAG retrieves relevant information from stored company knowledge and gives that information to the AI before generating the response/content.

Example:





Later, when a user makes a request such as “Create an Instagram post for our new perfume”:



The AI may retrieve:

- Brand Style: Luxury
- Colors: Black + Gold
- Tone: Elegant
- Target Audience: Professionals
- Design Preference: Minimal

The generated creative can then follow the company's established identity.

14. Does Every New Brand Need RAG?

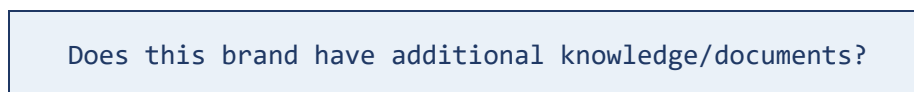
No. A new brand may have very little information. For example, a new user might simply provide: “This is a perfume from my company. Create an Instagram post.”

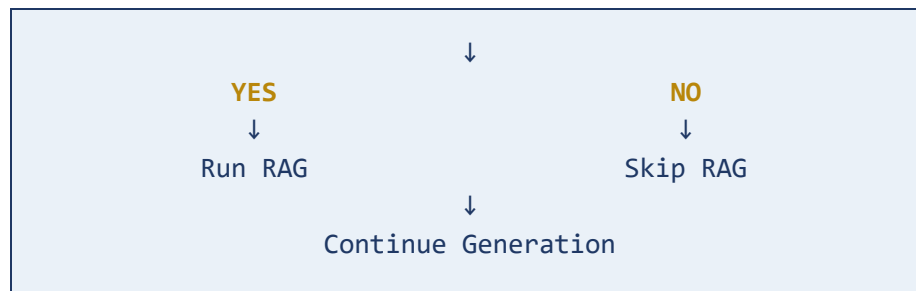
The system can use Product Image + User Request + Basic Brand Information + General AI Knowledge and generate a creative.

RAG is only useful when there is additional company knowledge worth retrieving. Therefore, the system should not automatically run RAG for every request.

15. How Does the System Decide Whether to Use RAG?

The AI orchestration layer can determine whether additional brand knowledge is required. A simple initial implementation can use backend rules. For example:





Examples:

- Background Removal — User: “Remove the background.” — RAG is unnecessary.
- Creative Generation — User: “Create a post according to our brand guidelines.” — RAG may be required.
- Simple New Brand — User: “This is my perfume. Make an Instagram post.” — RAG may not be necessary.

This makes RAG an optional tool inside the AI orchestration system, rather than a mandatory component of every generation.

16. RAG vs Normal Database

It is important to distinguish between structured data and unstructured knowledge.

Storage Type	Best Suited For
PostgreSQL	Best for structured information: Brand Name, Colors, Logo, Style, Tone, User, Product, Project
Vector Database / RAG	Useful for large unstructured information: Brand Guidelines, Website, PDFs, Marketing Documents, Previous Campaigns, Large Product Catalog

Therefore, Social Yolo can use both, with PostgreSQL handling structured data and the RAG system handling unstructured data, both feeding into the LLM.

17. Creative Generation Approach

One of the most important architectural decisions is that the image-generation model should not necessarily be responsible for creating the entire final advertisement. A better approach is a hybrid creative-generation system.

The LLM can act as a creative director and produce a structured creative specification. For example:

```
{
  "style": "luxury",
  "background": "dark marble",
  "accent_color": "#D4AF37",
  "headline": "Own the Night",
  "cta": "Shop Now",
```

```
"product_position": "center",  
"logo_position": "top-left",  
"platform": "instagram",  
"dimensions": "1080x1080"  
}
```

Then the application can use this specification to construct the final design.

18. AI vs Our Own Application

The responsibilities can be divided as follows.

AI should handle:

- Understanding the product
- Generating copy
- Suggesting creative concepts
- Generating visual backgrounds/scenes
- Understanding brand/style information
- Providing creative specifications

Our application should handle:

- User management
- Brand management
- Project management
- Storage
- Credits
- Database operations
- Image compositing
- Text placement
- Logo placement
- Resizing
- Export
- Generation history
- Caching

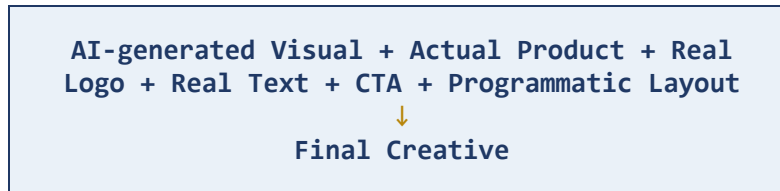
This division gives us better control and reduces unnecessary API usage.

19. Why Not Let the Image AI Generate Everything?

If we ask an image model to generate the entire advertisement, we may face problems such as:

- Incorrect text
- Incorrect logo
- Product changes
- Inconsistent branding
- Poor typography
- Incorrect positioning
- Lack of precise control

Therefore, a hybrid approach is preferable:



This gives us greater control over the final output.

20. Model Training / Fine-Tuning

Custom model training should not be part of the initial implementation.

V1

Use: Existing Models + APIs + Prompting + Our Application Logic

Later

If we identify a specific limitation, we can investigate:

- Fine-tuning
- LoRA
- Custom image models
- Custom style models
- Self-hosted open-source models

Training should only be considered when there is a clear business/quality reason. Training from scratch would require:

- Large datasets
- GPU infrastructure
- ML expertise
- Training time
- Evaluation
- Model maintenance

This would significantly increase complexity and cost.

21. Advantages of Not Training a Model Initially

Advantages

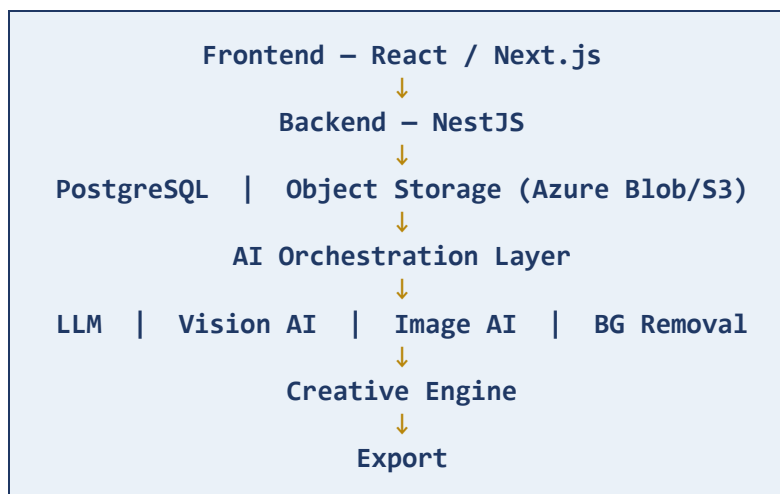
- Faster development
- Lower initial cost
- Less infrastructure
- Easier experimentation
- Access to state-of-the-art models
- Easier model replacement
- Faster MVP launch

Disadvantages

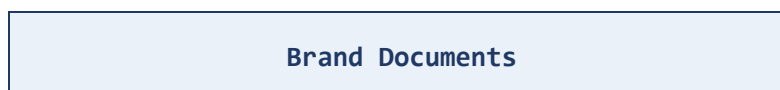
- API costs
- Third-party dependency
- Less control
- Potential provider changes
- Possible limitations in customization

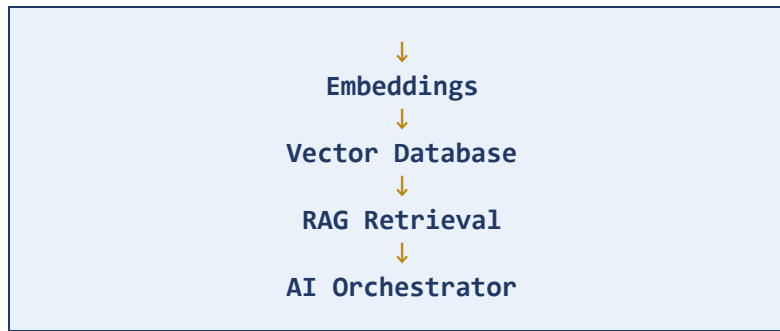
22. Recommended Technology Architecture

The initial technology stack can be:



RAG can be added as an optional component:





23. Recommended V1 Development Strategy

The development should be incremental.

Phase 1 — Basic Product

- Authentication
- Brand creation
- Product upload
- Project management
- Image storage
- Basic brand profile
- Basic creative generation

Phase 2 — Image Processing

- Background removal
- Image enhancement
- Product analysis

Phase 3 — AI Creative Generation

- Copy generation
- Style selection
- Platform selection
- Creative specification generation
- Image generation
- Product compositing

Phase 4 — Brand Intelligence

- Brand documents
- Website ingestion
- Embeddings
- Vector database

- RAG

Phase 5 — Optimization

- API costs
- Generation quality
- Latency
- Caching
- Model selection
- Failure/retry handling

Phase 6 — Advanced AI (only after evaluating the above)

- Fine-tuning
- LoRA/custom models
- Self-hosted models
- Custom image-generation infrastructure

24. Overall Pros and Cons of the Proposed Approach

Pros

- Faster MVP development — existing AI models eliminate the need to train models from scratch
- Lower initial development complexity — the team can focus on product functionality rather than ML infrastructure
- Flexible — individual AI providers can be replaced
- Scalable architecture — the orchestration layer can support additional AI capabilities later
- Better control — separating AI-generated visuals from programmatic text/logo/layout composition gives greater control
- Cost optimization opportunities — caching and internal processing can reduce API usage
- Gradual introduction of RAG — RAG can be introduced only when the amount of brand knowledge justifies it

Cons

- API costs — frequent image and AI generation can become expensive at scale
- Third-party dependency — the system depends on external AI providers
- API latency — multiple AI calls may increase generation time
- Inconsistent AI output — image models may produce different results for similar requests
- Product preservation challenge — AI image generation may alter the user's actual product, so product compositing/reference-image techniques need to be evaluated carefully
- Brand consistency challenge — maintaining consistent visual identity across many generations can be difficult

- Privacy considerations — user images and company information may be processed by third-party AI providers
- Provider changes — pricing, limits, models, and policies can change over time

25. Final Recommended Approach

For Social Yolo V1, the recommended approach is:

Do not train our own AI models initially. Build a modular AI orchestration layer that combines existing AI APIs/models with our own application logic.

The system should use:

- PostgreSQL for structured user, brand, product, and project information
- Object storage for original, processed, and generated images
- Vision AI for understanding uploaded products
- LLMs for copy generation and creative planning
- Image-generation models/APIs for visual generation
- Background-removal/enhancement services for product preparation
- Our own creative engine for precise text, logo, CTA, composition, resizing, and export
- Embeddings + Vector DB + RAG only when brands have large amounts of unstructured information that needs semantic retrieval

The high-level philosophy should be:



The goal is not to build a new AI model. The goal is to build a strong product around existing AI capabilities.

As the product grows, we can monitor the cost and quality of each component. If a particular AI service becomes too expensive or does not provide sufficient quality, that component can later be replaced with an open-source/self-hosted model or potentially a fine-tuned model.

This approach allows Social Yolo to move quickly toward an MVP while keeping the architecture flexible enough for future AI customization and optimization.