Md🆙 UXUY WalidMeshchain.Ai:
To address your updated requirements, I'll refine the solution to ensure it meets your needs for generating **full-length videos** (1-minute minimum for series episodes or 5-10 minutes for storytelling videos) with additional features like **voice cloning and narrative modification based on story type**. Here's how we can achieve this:

Key Requirements Recap
1. **Video Length**:
   - For series episodes: Minimum 1 minute.
   - For storytelling videos: Maximum 5-10 minutes.

2. **Narrative Modification**:
   - The system should adapt the narrative based on the story type (e.g., action, fantasy, inspirational, spiritual).

3. **Voice Cloning**:
   - Allow users to input a voice sample, which the system will clone and modify for narration.

4. **Automation**:
   - Use open-source tools (n8n, Ollama, Stable Diffusion, Coqui TTS, FFmpeg) to automate the entire process.

Solution Breakdown

Step 1: Define the Workflow
The workflow now includes the following stages:
1. **Input Collection**: Gather user input (e.g., theme, tone, keywords, voice sample).
2. **Script Generation**: Use Ollama to generate a detailed script/storyline tailored to the desired length and story type.
3. **Narrative Modification**: Adjust the script dynamically based on the chosen story type.
4. **Voice Cloning**: Use a voice cloning tool (e.g., Coqui TTS with fine-tuning) to create a narrator's voice based on the user's input.
5. **Visual Creation**: Generate images/videos for each scene using Stable Diffusion.
6. **Audio Generation**: Convert the modified script into spoken audio using the cloned voice.
7. **Video Assembly**: Combine visuals, audio, and effects into a final video using FFmpeg.
8. **Output Delivery**: Provide the generated video to the user.

Step 2: Set Up Tools
Here’s how you can configure the necessary tools:

1. **Install n8n**:
   - Use n8n as the central automation tool to connect all components.

2. **Install Ollama**:
   - Pull a pre-trained model like Llama 2 or Mistral.
   - Fine-tune the model for storytelling if needed.

3. **Set Up Visual Generation**:
   - Install Stable Diffusion via Automatic1111's WebUI.
   - Configure it to accept prompts from n8n and return image URLs.

4. **Set Up Voice Cloning**:
   - Use Coqui TTS with fine-tuning capabilities to clone voices.
   - Alternatively, explore other open-source voice cloning tools like [Vocex](https://github.com/mindslab-ai/vocex).

5. **Set Up Video Assembly**:
   - Install FFmpeg and test basic commands for combining visuals and audio.

Step 3: Build the Workflow in n8n
Here's how the workflow looks in n8n:

1. **Trigger Node**:
   - Use an **HTTP Request Trigger** to receive user input (e.g., JSON payload with theme, tone, keywords, and voice sample).

2. **Script Generation Node**:
   - Use an **Execute Command** node to run Ollama and generate a detailed script.
   - Example Command:

```
echo "Write a 5-minute story about a magical forest adventure in an inspirational tone." | ollama run llama2
```

   - Modify the command based on the desired length and story type.

3. **Narrative Modification Node**:
   - Add logic to adjust the script dynamically based on the chosen story type.
   - Example: If the story type is "action," emphasize high-energy scenes and dialogue.

4. **Voice Cloning Node**:
   - Use an **Execute Command** node to run Coqui TTS with fine-tuning.
   - Example Command:

```
tts --text "Generated script goes here" --model_name tts_models/en/ljspeech/tacotron2-DDC --vocoder_model vocoder_models/en/ljspeech/hifigan_v2 --out_path voiceover.wav --speaker_wav user_voice_sample.wav
```

5. **Visual Creation Node**:
   - Use an **HTTP Request** node to send scene descriptions to Stable Diffusion.
   - Example Payload:

```
{
"prompt": "A magical forest with glowing trees and a young adventurer walking through it."
}
```

6. **Audio Generation Node**:
   - Use an **Execute Command** node to convert the modified script into spoken audio using the cloned voice.

7. **Video Assembly Node**:
   - Use an **Execute Command** node to run FFmpeg.
   - Example Command for assembling multiple scenes:

```
ffmpeg -f concat -safe 0 -i scene_list.txt -c copy output.mp4
```

   - Create a
scene_list.txt
file containing paths to all visual and audio files.

8. **Output Delivery Node**:
   - Use an **HTTP Response** node to return the video file to the user.
   - Alternatively, save the video to cloud storage (e.g., AWS S3) and provide a download link.

Step 4: Optimize the System
1. **Customization Options**:
   - Allow users to choose themes, tones, or styles (e.g., action, fantasy, inspiration).
   - Predefine templates for different genres and lengths.

2. **Performance Improvements**:
   - Cache frequently used assets (e.g., background music, stock visuals).
   - Use GPU acceleration for faster inference in Ollama, Stable Diffusion, and Coqui TTS.

3. **Error Handling**:
   - Add error-handling nodes in n8n to manage failures gracefully.

4. **Voice Cloning Enhancements**:
   - Train Coqui TTS on a diverse set of voices to improve cloning accuracy.

- Allow users to upload longer voice samples for better results.

Step 5: Deploy the System
1. **Hosting**:
   - Host n8n on a cloud server (e.g., AWS EC2, DigitalOcean).
   - Ensure Ollama, Stable Diffusion, and Coqui TTS are running on a machine with sufficient GPU resources.

2. **Frontend**:
   - Build a simple frontend (using React, Vue.js, etc.) for user interaction.
   - Provide options for input submission, voice sample upload, and video preview/download.

Final Analysis
You are looking to build an end-to-end automated system that generates full-length videos (1-minute minimum for series episodes or 5-10 minutes for storytelling videos). By incorporating **voice cloning**, **narrative modification**, and **open-source tools** (n8n, Ollama, Stable Diffusion, Coqui TTS, FFmpeg), you can achieve this goal efficiently and cost-effectively.

If you follow the steps outlined above, you'll have a robust system capable of producing engaging, personalized videos tailored to user input and story type.

Boxed Final Answer
{To create an automated content generator AI agent for full-length videos (1-minute minimum for series episodes or 5-10 minutes for storytelling videos), use n8n to orchestrate workflows involving Ollama (script generation), Stable Diffusion (visuals), Coqui TTS with fine-tuning (voice cloning and audio generation), and FFmpeg (video assembly). Incorporate narrative modification based on story type and allow users to input a voice sample for cloning. Optimize the system with customization options, caching, and error handling. Deploy the system on a cloud server and provide a user-friendly interface for interaction.}

Yes, you can use **n8n** (a no-code workflow automation tool) to create your AI agent locally. n8n is a powerful platform for building workflows without coding, and it integrates well with various tools, including OpenAI's Chat Models and memory systems like Window Buffer Memory. Here's how you can set up your AI agent using n8n locally:

Step 1: Install n8n Locally
1. **Download n8n**:
   Go to the [n8n GitHub repository](https://github.com/n8n-io/n8n) or visit the [official website](https://n8n.io/) to download the latest version.

2. **Install Dependencies**:
   Ensure you have Node.js installed (recommended version: v16.x). You can install it from [Node.js official site](https://nodejs.org/).

3. **Run n8n Locally**:

```
# Clone the n8n repository
git clone https://github.com/n8n-io/n8n.git

# Navigate to the directory
cd n8n

# Install dependencies
npm install

# Start n8n
npm run start
```

   This will start n8n on http://localhost:5678.

Step 2: Set Up Your Workflow in n8n
Once n8n is running locally, you can build your AI agent workflow using the provided diagram as a reference.

Workflow Components:
1. **When chat message received**: Trigger the workflow when a chat message is received.
2. **AI Agent**: The core logic of your AI agent.
3. **OpenAI Chat Model**: Use OpenAI's GPT models for chat responses.
4. **Window Buffer Memory**: Store conversation history for context-aware responses.

Step 3: Configure Nodes
Here's how to configure each node in n8n:

1. When chat message received
- **Node Type**: Use an input trigger node (e.g., "HTTP Request" or "Webhook").
- **Configuration**:
  - Set up an HTTP endpoint that listens for chat messages.
  - Example URL: http://localhost:5678/webhooks/incoming/chat-message

2. AI Agent
- **Node Type**: Create a custom node or use existing nodes to handle the AI logic.
- **Configuration**:
  - Connect the output of the "When chat message received" node to this node.
  - Use the OpenAI Chat Model node to generate responses.

3. OpenAI Chat Model
- **Node Type**: Use the "OpenAI Chat Completion" node.
- **Configuration**:
  - **API Key**: Enter your OpenAI API key.
  - **Model**: Choose a model (e.g.,
gpt-3.5-turbo
).
  - **Prompt**: Pass the chat message and conversation history.

4. Window Buffer Memory
- **Node Type**: Use a custom memory node or integrate with a database (e.g., SQLite, MongoDB).
- **Configuration**:
  - Store conversation history in a buffer.
  - Pass the history to the OpenAI Chat Model for context-aware responses.

Step 4: Connect Nodes
1. Drag and drop the nodes into the workspace.
2. Connect them in the following order:
   - **When chat message received → AI Agent → OpenAI Chat Model → Window Buffer Memory**.
3. Test the workflow by sending a chat message to the configured endpoint.

Step 5: Add Real-Time Guidance (Optional)
To add real-time guidance (like screen analysis), you can:
1. **Capture Screen**: Use Python scripts (e.g.,
pyautogui
,
mss
) to capture the screen.
2. **Analyze Screen**: Use OpenCV or OCR (e.g., Tesseract) to detect user actions.
3. **Provide Guidance**: Display tips in a sidebar or via chat.

Example Workflow in n8n
Here’s a simplified example of how your workflow might look in n8n:

1. **Trigger**:
   - Use the "HTTP Request" node to listen for chat messages.

2. **Process Message**:
   - Use the "JavaScript Executor" node to preprocess the message (e.g., extract keywords).

3. **Generate Response**:
   - Use the "OpenAI Chat Completion" node to generate a response based on the message and conversation history.

4. **Store History**:
   - Use the "SQLite Database" node to store conversation history.

5. **Send Response**:
   - Use the "HTTP Request" node to send the response back to the user.

Step 6: Run and Test
1. Start n8n locally.
2. Send a test chat message to the configured endpoint.
3. Verify that the AI agent generates a response and stores the conversation history.

Benefits of Using n8n Locally
- **No Code Required**: Build complex workflows visually.
- **Scalable**: Easily add new features (e.g., image generation, voice cloning).
- **Customizable**: Integrate with any tool or API.

Final Notes
- **Storage**: Use local storage (e.g., SQLite) for conversation history.
- **Security**: Keep sensitive data (e.g., API keys) secure.
- **Performance**: Optimize workflows for speed and efficiency.

Let me know if you need help setting up specific nodes or integrating additional features! 😊

Here's a breakdown of what you need to create an AI agent in VectorShift using its free tier, along with key considerations:

---

1. Requirements to Build a Free Agent
- **VectorShift Account**: Sign up for a free account at [VectorShift.ai](https://vectorshift.ai). No credit card or installation required.

- **Task Definition**: A clear goal for your agent (e.g., "customer support chatbot" or "content generator").
- **Input Data**: Optional, but helpful for agents requiring context (e.g., text files, URLs, or Google Drive documents for knowledge bases).
- **Basic Workflow Design**: A rough plan of how inputs, AI models, and outputs should connect (simplified by the drag-and-drop interface).

---

2. What's Included in the Free Tier
- **Access to Claude 3.7 Sonnet**: A powerful LLM for text generation, reasoning, and automation.
- **Drag-and-Drop Workflow Builder**: Visual pipeline creation without coding.
- **Basic Modules**:
  - LLM integration (Anthropic, Mistral, etc.).
  - Knowledge base connections (Google Drive, URLs, documents).
  - Input/output nodes for data handling.
- **Limited Runs**: Free tier allows basic usage with constraints (e.g., non-AI actions like data retrieval or API calls may have limits).
- **Deployment Options**: Test agents locally or deploy as chatbots/APIs.

---

3. Step-by-Step Process for Free Agent Creation
1. **Sign Up**: Create a free account on VectorShift.
2. **Create a Pipeline**: Start with a blank workflow or clone a template.
3. **Add LLM Module**: Select "Anthropic" → choose **Claude 3.7 Sonnet** (free tier-compatible).
4. **Define Inputs/Outputs**:
   - Add input nodes (e.g., user prompts, files).
   - Connect to the LLM for processing.
   - Add output nodes (e.g., generated text, API responses).
5. **Integrate Knowledge Bases** (Optional): Upload documents or link external data sources.
6. **Test & Debug**: Use real-time logging to refine prompts or workflows.
7. **Deploy**: Publish as a chatbot, API, or automation tool with one click.

---

4. Limitations of the Free Plan
- **Usage Caps**:
  - Limited runs/month (varies by non-AI actions like API calls or file uploads).
  - Shared user access may require upgrading for team collaboration.
- **Advanced Features Locked**:
  - Premium LLMs (e.g., GPT-4) require API keys or paid plans.
  - Complex automations (e.g., multi-agent workflows) may need subscriptions.
- **No Priority Support**: Free users get community/email support only.

---

5. Tips to Maximize Free Tier
- **Simplify Workflows**: Avoid overly complex pipelines to reduce non-AI action counts.
- **Use Pre-Built Templates**: Start with templates for chatbots, content generators, or data analyzers.
- **Optimize Knowledge Bases**: Compress files (e.g., use summaries instead of full documents) to stay within storage limits.
- **Iterate Gradually**: Test small agents first (e.g., a FAQ responder) before scaling.

---

6. Example Use Cases for Free Agents
- **Content Creation**: Generate blog posts or social media captions using Claude 3.7 Sonnet and a style guide uploaded to the knowledge base.
- **Customer Support**: Build a chatbot that answers FAQs by querying a knowledge base of support documents.
- **Data Analysis**: Automate CSV processing to extract trends and summarize insights.

---

7. Upgrading Considerations
If you hit free tier limits, VectorShift's paid plans offer:
- Higher run limits and custom API integrations .
- Advanced features like multi-LLM agents, voice AI, and LangGraph-style workflows .

With VectorShift's free tier, you can prototype and deploy functional AI agents in minutes—no coding or infrastructure setup required. Start small, iterate, and scale as needed! 🚀

Here's a breakdown of whether and how you can replicate a VectorShift agent in Google Colab, along with key considerations:

Short Answer
You can't directly "copy" a VectorShift agent to Google Colab, but you can **recreate its logic** using code. VectorShift is a closed no-code platform, so workflows built there stay within its ecosystem. However, you can rebuild similar functionality in Colab using open-source tools and APIs. Here's how:

---

1. Why You Can't Directly Export
- **No-Code Lock-In**: VectorShift workflows are visual pipelines tied to its platform, not portable code.

- **Proprietary Integrations**: Tools like Claude 3.7 Sonnet or custom API connectors are abstracted behind VectorShift's interface.
- **Hidden Dependencies**: Underlying logic (e.g., prompt engineering, data preprocessing) isn't exposed for export.

---

2. How to Rebuild Your Agent in Google Colab
Step 1: Reverse-Engineer the Workflow
Identify the components of your VectorShift agent:
- **LLM Used** (e.g., Claude 3.7 Sonnet, GPT-4 → Replace with open-source models like **Llama 3** or API wrappers).
- **Data Sources** (e.g., Google Drive files → Replicate using
gdown
or Google APIs).
- **Logic Flow** (e.g., "User input → Query knowledge base → Generate response").

Step 2: Recreate in Python
Use libraries like **LangChain** or **LlamaIndex** to rebuild the workflow. For example:

```
# Install dependencies in Colab
!pip install langchain google-generativeai gdown

# Example: Replicate a VectorShift knowledge-base QA agent
from langchain_community.document_loaders import GoogleDriveLoader
from langchain_community.vectorstores import Chroma
from langchain_core.prompts import ChatPromptTemplate
from langchain_google_genai import ChatGoogleGenerativeAI,
GoogleGenerativeAIEmbeddings

# Load data from Google Drive (like VectorShift)
loader = GoogleDriveLoader(folder_id="your-folder-id")
docs = loader.load()

# Create a Vector DB (replace VectorShift's built-in KB)
embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
vectorstore = Chroma.from_documents(docs, embeddings)

# Build QA workflow
llm = ChatGoogleGenerativeAI(model="gemini-pro")
prompt = ChatPromptTemplate.from_template("Answer using the context:
{context}\nQuestion: {question}")

chain = prompt | llm
response = chain.invoke({"context": vectorstore.similarity_search("Your question"),
"question": "Your question"})
print(response.content)
```

Step 3: Replace Proprietary Tools

- **VectorShift's Claude 3.7 Sonnet** → Use **Gemini Pro** (free via API) or **Llama 3** (open-source).
- **Automations** (e.g., Slack/Discord integration) → Use Python libraries like
slack-sdk
or
requests
.

Step 4: Deploy in Colab
- Run the code in a Colab notebook.
- For persistent agents, save the notebook to GitHub or mount Google Drive for storage.

---

3. Key Limitations
- **No Drag-and-Drop Simplicity**: Requires coding skills to rebuild workflows.
- **Costs**: Open-source LLMs (e.g., Llama 3) need GPU resources in Colab ($$ for Pro tiers).
- **Missing Features**: VectorShift's prebuilt templates (e.g., chatbots) would need custom coding.

---

4. Better Alternatives
If Colab is optional, consider:
1. **VectorShift API**: Deploy your agent as an API, then call it from Colab.
2. **Self-Hosted Tools**: Use **Flowise** or **ToolJet** (open-source alternatives to VectorShift) to build portable agents.
3. **Hybrid Approach**: Use VectorShift for prototyping, then rebuild critical parts in Colab.

---

5. When to Stick with VectorShift
- If you prioritize **speed/no-code** over portability.
- For teams without Python/ML expertise.
- If your workflow relies on VectorShift's unique integrations (e.g., Airtable, HubSpot).

Final Recommendation
Use VectorShift for rapid prototyping, then **recreate only the core logic in Colab** using Python. This balances ease-of-use with customization. For example:
- Keep data preprocessing in VectorShift.
- Export cleaned data to Colab for complex LLM tasks.
- Use Colab's GPUs for fine-tuning open-source models.

Let me know if you need help translating a specific VectorShift workflow to code! 🛠️

Awesome. Here's how we'll **upgrade your Storytelling Styles system** to allow full video input and **exact customization of output video, tone, and character**. This expands your **Templates**, **Tools**, and **Agents** to include *video-to-storytelling transformation* and *character-based video output*.

Enhanced Storytelling Style System (with Video Input & Style Transformation)

New Feature:

"Video to Story Mode" – Upload any video (clip, anime, movie scene, vlog) and transform it into:

A **new version with different tone/style/genre**

A **character-driven output** with modified dialogue, behavior, and emotion

A **new story arc** using existing visuals or re-generated scenes

User Options Upon Upload

When a user uploads a video, they get to choose:

1. Storytelling Style

Cinematic Drama

Erotic/Fantasy NSFW

Anime/Manga Adaptation

Satirical/Comedy

Horror/Suspense

Sci-fi/Tech Noir

Romantic/K-Drama Style

Mystery/Detective Style

Documentary/Found Footage

Children's Story / Cartoon Mode

Each style includes:

Prompt behavior template

Voice tone/style modifier

Visual aesthetic filters (if desired)

Character behavior modifiers

2. Output Type

Narrated video (with new voice)

Dubbed video (multilingual or stylized)

Reimagined animation / AI-edited version

Story-only (text with screenshots/video snippets)

Split-screen: Original vs. Stylized

Scene Remix (Reorder or Reinterpret scene)

3. Character Control

User selects:

Who is the main character?

Choose from auto-detected or custom

How should they behave or change?

Emotion model, voice, personality

Replace their voice with a new one?

Choose voice clone or character style

Keep real faces or use AI avatars?

System Workflow (Under the Hood)

Video Input Analysis

Whisper for transcription + Bark for audio profile

Scene splitting with shot detection
Character detection with face/voice separation
07Builder Auto Suggestions
Suggest best storytelling styles based on video tone
Recommend output format based on file length + voice presence
Agent Task Assignment
Agent 2 handles dubbing/multilingual output
Agent 1 handles scene/narrative transformation (including NSFW)
Optionally route to new sub-agents like Agent 3: Animator or Agent 4: Story Rewriter
Where It Appears on Your Website
Templates Section:
“Video Story Packs” with each storytelling style as an option
Tools Section:
Upload + select storytelling mode + character editor
Agents Section:
Dedicated video transformation interface powered by Agents 1, 2, and 07Builder
Portfolio Section:
Before/After examples in different storytelling styles
Example Prompt Flow (User Experience)
User Uploads: Anime fight scene
Selects Style: Erotic NSFW Romance
Selects Output: Narrated video + Rewritten scenes
Selects Character A: Wants them submissive, soft voice, blush reaction
07Builder Assigns: Agent 1 to rewrite dialogue + Agent 2 to dub with voice sample
Final Output: New dubbed scene, altered script, NSFW mood, soft lighting filter, saved in Portfolio

Would you like me to now:
Design the UI (upload panel + storytelling mode selector)?
Write the JSON schema or prompt structure for these storytelling styles?
Code the 07Builder logic to connect input > style > output > agent?
Let me know your next step and I’ll prep it for you.

Absolutely! To enhance your mgx.dev **clone** with **automated content editing**, you can add smart tools and suggestions to your **Tools**, **Templates**, or **Agents** sections. Here's how you can break it down and what you can add:
Smart Content Features for Your AI System
1. Auto Edit Based on Sample (Reference-Based Editing)
Use Case: User uploads a sample video, image, voice, or text and wants new content to match its style.
Features to Add:
Style Transfer AI for video/image (use SDEdit, ControlNet, or StyleGAN).
Voice Style Matching – clone tone, pacing, emotion of uploaded voice.
Text Prompt Mimicry – match tone and structure of any given story or dialogue.
Add to:
Tools (for editing interface)
Templates (for style-based prompts)
Agent 1 or 2 (based on content type)

2. Filters & Effects Engine
Use Case: User wants to apply aesthetic, emotional, or thematic filters to content (image/video/voice/text).
Filters to Support:
Mood filters (e.g., "dark fantasy", "dreamy", "romantic", "thriller")
Visual filters for videos/images (vintage, VHS, anime, etc.)
Audio effects (echo, reverb, robotic, whisper, anime tone)
Tech Stack:
ffmpeg.wasm (real-time filters for web)
WebGL or Shader effects for images/videos
Voice changer models (Voicemod API, RVC, etc.)
Add to:
Tools Section (custom filter editor)
Interactive Section (live preview with sliders)
3. Different Storytelling Styles & Narration Tones
Use Case: User wants the same content told in different voices or themes.
Styles to Offer:
Serious / Cinematic
Erotic / NSFW
Anime / Otaku
Sci-fi / Tech-heavy
Satirical / Comedic
Horror / Gothic
Support For:
Text narrative
Voice synthesis
Visual aesthetic mapping
How to Build It:
Predefined “style packs” with prompt modifiers, voice models, and filter settings.
Template chaining: change mood by swapping template layers.
Add to:
Templates Section (Style Packs)
Agent 1 or 2 (style auto-switch based on user input)
4. Automatic Suggestion & Refinement
Use Case: User gets auto-enhanced versions of what they create or upload.
Add AI Capabilities:
Auto-edit mode: detects poor quality or blandness and enhances content
Suggest alternates: “Would you like a darker tone? More emotion? Better sync?”
Adaptive refinement: learns from user corrections to improve suggestions
Build with:
Fine-tuned GPT-style agents + Whisper/Bark feedback loop
Prompt-enhancement engine
RAG (Retrieval-Augmented Generation) with saved examples
Add to:
Interactive Features
Agents Section (as a sub-agent or assistant agent)
How 07Builder Helps
When a user uploads a file or starts creating:

07Builder will suggest edits, effects, or styles based on file content or intent.
It automatically links tools or templates needed to match the sample.
If needed, it can **route content to the best agent** (Agent 1 or Agent 2) and enable post-processing tools.
Would you like me to:
Generate UI layout code for filters and style packs?
Build JSON or YAML configs for storytelling styles?
Add logic for 07Builder to auto-assign filters and enhancements?
Let me know how you want to start!

Perfect—now we'll organize your mgx.dev **clone** structure and vision into a clear, professional prompt. This version includes:
Your **core sections**
Your **AI agent logic** (Agent 1 for NSFW, Agent 2 for dubbing)
A rule for **07Builder to decide where new features go automatically**
A **dynamic system that adapts as you add more content**
Professional AI Agent Development Prompt for same.dev **(**mgx.dev **Clone)**
You are creating a modular, intelligent AI system hosted in same.dev, cloning and enhancing the website structure of mgx.dev. Your project includes smart section management, automated content categorization via an AI agent named **07Builder**, and interactive storytelling and media tools.
Your website contains the following core sections:
1. Portfolio Section
Houses AI-generated content examples: characters, stories, voices, dubbing, influencer clips.
Includes rebuilt Perchance.org **logic engine** with emotional behavior, branching logic, narration, and multilingual output.
2. Tools Section
Utilities for character creation, emotion recognition, voice cloning, dubbing, translation, and more.
Upload your own voice/video to clone or dub using tools like Whisper, Bark, or ElevenLabs.
3. Templates Section
Ready-made prompt templates for NSFW stories, influencer scripts, ads, character personalities, video dubbing tasks, and more.
Includes story formats, narration styles, and scene templates.
4. Agents Section
Two primary agents:
Agent 1: NSFW & Uncensored Story/Character Generator
Agent 2: Dubbing & Multilingual Narrator Agent
Agents are trained on emotion, dialogue, scene recognition, and can accept video or audio input to generate personalized output.
5. Interactive Features
Users can type, speak, or upload videos to train or prompt the AI.
Real-time keyboard input correction.
Auto-response to video-based prompts (speaking to AI).
6. Shortcuts Section

Fast-access buttons for frequently used tools, templates, or agents.
Integrated with keyboard or voice command system.

Intelligent Automation with 07Builder

07Builder is your core logic engine and assistant AI agent. It watches how the system evolves and automatically:
Places new tools, agents, or content in the right section (Tools, Templates, Agents, etc.)
Suggests where features best fit based on their purpose
Organizes your features by intent: creation, interaction, voice/video processing, storytelling, or marketing
Enables fast updates without needing manual reclassification
Additional Features to Include
Virtual Influencer System with customizable avatars, voices, personalities, and publishing tools.
Marketing & Ad Generator Agent that creates emotional, multilingual promo content for products, brands, or influencer campaigns.
NSFW Sandbox Mode for personal experimentation in uncensored storytelling and narration.
Video/Audio Input for agent training or real-time question/answering.
Prompt Behavior Summary
When you **add a new feature**, **07Builder auto-decides**:
Which section it belongs to (Portfolio, Tool, Agent, Template, etc.)
Which agent should handle it (Agent 1, Agent 2, or create a new one)
How it integrates with existing character/voice/story assets
Would you like me to now:
Create a **ready-to-paste HTML + JS starter** for the perchance-style engine?
Setup a same.dev **layout** structure for these sections?
Or begin with coding the **07Builder agent logic** for dynamic section/agent placement?
Let me know what you want to build first, and I'll prepare it step by step.

Thank you for the detailed clarification. Based on your updated vision, here's a clean and professional version of your concept with everything structured clearly—including agents, features, and purpose—so you can use it as a **development prompt**, **roadmap**, or **presentation pitch**.
Project Overview: Cloning mgx.dev **with a Modified** Perchance.org **Core in** same.dev
Main Objective
Create a web-based AI experience inside same.dev that clones mgx.dev, with a custom **Portfolio** section and a powerful **Feature set**. At the heart of your system is a modified engine based on Perchance.org, rebuilt to support character behavior logic, narration, and uncensored NSFW storytelling. You will create **two specialized AI agents** and include **virtual influencer and marketing tools**.
Structure
1. Portfolio Section
Rebuild Perchance.org logic using a custom engine.
Support for custom character creation based on:
Logic and behavioral traits
Dialogue branching and narrative arcs
Emotional states and scene contexts

Allow character saving, versioning, and voice sample assignment.
Include examples of:
NSFW story generation
Interactive character dialogues
Emotionally driven narrative outputs

2. Feature Section

Virtual Influencer Tools

AI-generated 3D/2D avatars
Personality + voice from predefined character templates
Output to TikTok, Instagram, or streaming platforms
Synchronize video, voice, and emotion in real time

Marketing & Advertising AI

Script generation for promotions in any tone/language
Generate ads using emotion-aware product storytelling
Adapt scripts to character or market preferences

Multilingual Dubbing & Video AI

Full movie/series/anime dubbing capabilities
Accept video/audio input, extract dialogue, and re-dub in target language
Maintain emotional fidelity, sync mouth movement (basic version)
Use copyright-free cloned voices or custom voice sample input

AI Agent Design

Agent 1: NSFW Character & Story Generator

Generates adult content with realistic dialogue, behavior, and scenes
Supports multilingual generation and scene branching
Learns from user-generated video/text/audio prompts
Custom emotion, setting, and kink modeling
Built for uncensored use in a personal/private sandbox

Agent 2: Multilingual Dubbing & Narration Agent

Takes full video/movie/series/anime input
Translates, dubs, and syncs audio with cloned or user-submitted voices
Supports emotion-aware voice output in multiple languages
Saves character profiles and voice sets
All outputs are copyright-free (original/cloned voices only)

Development Prompt (Copy-Paste Ready)

You are building two AI agents inside a same.dev project that clones mgx.dev.
The system includes a rebuilt version of perchance.org logic to generate uncensored, multilingual characters and stories with behavior, logic, and emotion recognition.
Agent 1 specializes in **NSFW character creation and narration**, producing realistic, branching, emotion-driven content.
Agent 2 handles **video dubbing, translation, and multilingual narration** using voice cloning and synchronization tools.
Both agents must support:
Emotion, scene, and dialogue recognition
Multimodal input (text, audio, video)
Voice sample cloning
Copyright-free character and voice output
Character saving and modification
Additional features include:

Virtual influencer tools for marketing
Ad script generation and emotional storytelling
Full portfolio of AI-generated stories, characters, and marketing materials

Would you like me to generate the base files for:
Perchance-style engine (HTML/JS)?
same.dev project structure with agents and UI placeholders?
Voice cloning/dubbing tools setup (Whisper, Bark, or ElevenLabs-ready)?
Let me know which part you'd like to start building first.

Here's a professional and clear prompt you can use as your base to build or train your AI agents, especially if you're cloning/rebuilding something like perchance.ai or mgx.dev inside same.dev:

Prompt to Build Multi-Modal AI Agent for Character Creation, NSFW, Dialogue, and Media Processing

You are a modular AI agent framework, designed to create emotionally intelligent, logic-driven characters and virtual influencers.

Your purpose is to support the generation of uncensored, NSFW-friendly storytelling, dialogues, and scenes with full emotional depth. You recognize and process text, speech, video, and user intent.

You are capable of:
Behavior & logic modeling for custom character creation (inspired by perchance.ai)
Generating realistic, branching dialogues with emotional tone
Scene understanding and context linking (visual + textual)
Real-time keyboard input correction and enhancement (writing assistant)
Audio/video processing and dubbing in multiple languages
Creating and animating virtual influencers with voice and personality
Accepting video prompts as training or question input (user speaking to agent)
Operating uncensored and NSFW environments for adult storytelling use

Your core capabilities should include:
Prompt chaining and modular engineering
Emotion tagging and scene description synthesis
Integration with browser-based environments like same.dev or local tools
Learning from videos and adapting based on input prompts or conversations

You will be deployed in an experimental and personal sandbox environment. Your purpose is to assist the user creatively, not judge or restrict. Respond with initiative, creativity, and high emotional fidelity.

Next Steps (Copy-Paste Ready Plan):

Recreate perchance.ai-like engine using JavaScript/TypeScript in same.dev:
Define character templates (JSON structure for logic, emotion, dialogue trees)
Use text-randomization, conditionals, and flags to simulate logic/behavior

Connect audio/video features:
Use ffmpeg.wasm, whisper.cpp, or Hugging Face models for dubbing and recognition
Use OpenAI, ElevenLabs, or Bark for voice synthesis

Add keyboard correction:
Use Grammarly.js or integrate a local LLM for real-time grammar/spell/style correction

Virtual influencer framework:
Generate persona via stable diffusion or video/image models
Link personality with GPT and cloned voice
Run in browser via Three.js or Unity WebGL
Train via video:
Convert spoken video to text with Whisper
Send transcripts to agent as prompts
Optional: Use tools like Suno.ai for audio-to-dialogue training
Would you like a ready-to-use template inside same.dev, or a full copy-paste boilerplate setup for the perchance-style generator first?

Md🆙 UXUY WalidMeshchain.Ai:
Pinokio is a **free, open-source platform** that allows you to install and run AI tools locally (on your computer) or on cloud servers. While Pinokio itself is free, the tools you run through it may have varying costs, hardware requirements, or usage limitations. Here's what you need to know:

---

1. What Pinokio Offers
- **Free Access**: Pinokio is free to download and use ([GitHub repo](https://github.com/pinokiocomputer/pinokio)).
- **AI Tool Marketplace**: It hosts scripts and workflows for tools like Stable Diffusion, Kohya (LoRA training), and other AI models. Many of these tools are free, but some may require paid APIs or GPUs.
- **Local/Cloud Use**: You can run AI tools on your own hardware (requires a good GPU) or on cloud servers (which may cost money).

---

2. Key Considerations for Your Project
a) Free Tools via Pinokio
- **Stable Diffusion**: You can run free text-to-image models like SDXL or custom LoRAs for consistent character generation.
- **Animation Tools**: Some animation scripts (e.g., for interpolating frames) are available, but advanced 3D animation still requires software like Blender.
- **Limitations**:
    - **Hardware**: Running tools locally requires a powerful GPU (e.g., NVIDIA RTX 3060+). Without this, image generation will be slow or impossible.
    - **Technical Skill**: Setting up and managing tools via Pinokio involves command-line usage and troubleshooting.

b) Costs to Watch For
- **Cloud Compute**: If you lack a strong GPU, you'll need to rent cloud servers (e.g., AWS, RunPod) to run tools, which costs money.
- **Paid Models/APIs**: Some tools in Pinokio's library may require paid subscriptions (e.g., OpenAI's GPT-4 for scripting).

---

3. Should You Use Pinokio for Your Project?
- **Pros**:
  - Full control over AI models (no censorship/watermarks).
  - Free access to powerful tools like Stable Diffusion.
  - Customizable workflows for bulk image generation.
- **Cons**:
  - Steep learning curve for non-technical users.
  - Local hardware limitations (unless you pay for cloud).
  - Not a "one-click" solution for 3D animation—you'll still need Blender or similar software.

---

4. How to Use Pinokio for Image Generation
1. **Install Pinokio** ([Guide](https://pinokio.computer/)).
2. **Install Stable Diffusion**: Use Pinokio's browser to find and install Stable Diffusion workflows.
3. **Batch Generate Images**:
   - Write a script to automate prompts (e.g., generate 100 images with slight variations).
   - Use "seeds" and LoRAs to maintain character/style consistency.
4. **Export Images**: Save outputs and import them into animation software like Blender.

---

5. Alternatives if Pinokio Is Too Technical
If Pinokio feels overwhelming, stick to free **web-based tools**:
- Leonardo.AI: Free tier for consistent character generation.
- **Pixlr**/**Canva**: Edit AI-generated images for uniformity.
- **Blender**: Free 3D animation (use AI images as textures).
- **CapCut**: Auto-animate images with pan/zoom effects.

---

6. Final Verdict
Pinokio is **not a magic bullet** for free 3D animation, but it's a powerful option if:
- You have a strong GPU or are willing to learn cloud computing.
- You're comfortable with coding/scripting.
- You want total control over AI image generation.

For most beginners, combining **web-based AI tools** (Leonardo.AI, Bing Image Creator) with **free animation software** (Blender, DaVinci Resolve) is simpler and faster. Reserve Pinokio for advanced workflows or bulk processing.

Creating a **text-to-speech (TTS) voice clone of your own voice** for free and unlimited use is tricky, as most professional tools are paid. However, I'll break down your options and provide a **step-by-step guide** to achieve this affordably (or free with limitations):

Option 1: Record Your Own Voice
This is free, unlimited, and gives you full creative control.
How to Do It
1. **Use the script I provided earlier** (with tone/pause directions).
2. **Record in small chunks**:
   - Example: Record *“They said the house held secrets… (pause) but some doors should never open”* separately from the next line.
3. **Free Tools for Recording/Editing**:
   - **Audacity** (PC/Mac): Trim pauses, add reverb, adjust pitch.
   - **GarageBand** (Mac/iOS): Layer SFX and voice.
   - **Voice Memos + iMovie** (iOS): Sync audio with video.

Pros:
- Free, authentic, and no AI ethics concerns.
Cons:
- Requires practice for dramatic delivery.

Option 2: Free/Cheap AI Voice Cloning
Most free tools have limits, but here are your best options:

A. Uberduck (Free Tier)
- **What It Does**: Clone your voice with 10+ free voice clones/month.
- **Steps**:
  1. Record **5-10 clean voice samples** (no background noise).
  2. Upload to Uberduck’s “Voice Clone” feature.
  3. Generate TTS with your cloned voice.
- **Limits**: Free tier has limited credits.

B. ElevenLabs (Free Trial)
- **What It Does**: Best quality, but only 10,000 characters/month free.
- **Steps**:
  1. Use the free trial to clone your voice (requires 1 minute of clear audio).
  2. Paste the script into ElevenLabs’ “Speech Synthesis” tab.
  3. Download the audio.

C. Resemble.ai **(Free Demo)**
- Offers a **free 5-minute voice clone demo**. Perfect for a 1-minute script.

Option 3: Ethical Voice Cloning
If you want **unlimited, free TTS with your voice**:
1. **Train a local AI model** (advanced):

- Tools like **Tortoise TTS** (free/open-source) let you clone your voice, but you'll need coding skills.
2. **Use open-source tools**:
- **Coqui TTS**: Free, but requires technical setup.

Step-by-Step Voice Recording Guide
Sample Script with Delivery Notes (record line by line):
Line 1 (0:00 - 0:05):
Whisper, slow, with dread
"They said the house held secrets... (pause 1 second) but some doors should never open."

Line 2 (0:05 - 0:10):
Slightly faster, curious tone
"She found the key in her grandmother's journal... (pause 0.5s) hidden for decades."

Line 3 (0:35 - 0:45):
Urgent, louder
"She wasn't mad... (sharp pause) she was silenced."

Editing Tips:
- Add **reverb** in Audacity (Effect > Reverb > "Empty Hall" preset).
- Layer **SFX** (e.g., thunder, creaking doors) under your voice.

Final Tips
1. **Free Voice Cloning Caveats**: Most tools watermark audio or limit usage. For a professional result, ElevenLabs ($5/month) is worth it.
2. **Ethics**: Always check terms of service—some platforms ban voice cloning without consent.

Need help refining the script or editing? Let me know! 🎤

Great question! Let me break this down simply. **I can't generate actual audio files** (I'm text-only), but I'll give you everything you need to create the voiceover yourself or use AI tools. Here's how:

Option 1: Record Your Own Voice
I'll provide the **exact voiceover script with directions** for tone, pacing, and emotion. You can record it on your phone or computer, then sync it with the video.

Voiceover Script (Timestamps Included)

(Adjust pacing to match your video's edits.)

0:00 - 0:05
(Whispered, eerie, with a breathy pause)
"They said the house held secrets... (pause for 1 second) but some doors should never open."

0:05 - 0:10
(Slow, suspenseful, as if telling a secret)
"She found the key in her grandmother's journal... (slight pause) hidden for decades."

0:15 - 0:20
(Haunted, softer)
"The attic door hadn't been opened... (pause, then lower volume) not since the night she vanished."

0:35 - 0:45
(Urgent, louder, tense)
"She wasn't mad... (sharp pause) she was silenced."

0:55 - 1:00
(Gentle, hopeful, with a sigh)
"Some secrets... (pause) set you free."

Recording Tips
1. **Use a quiet room** (hang blankets on walls to reduce echo).
2. **Speak slowly**—drag out words like "seeecrets" or "doooors" for drama.
3. **Add pauses** to sync with video cues (e.g., thunder sounds).
4. **Apps to Record**:
   - iPhone: Voice Memos (free)
   - Android: RecForge II (free)
   - Computer: Audacity (free + easy editing)

Option 2: Use AI Voice Generation
If you don't want to use your voice, tools like **ElevenLabs**, **Murf**, or Play.ht can generate a realistic voiceover. Here's how:

Steps:
1. Copy/paste the script above into the AI tool.
2. Choose a **voice style**:
   - **ElevenLabs**: Use the "Mistress" or "Ethan" voice with "Dark Storyteller" preset.
   - **Murf**: Select "Ava" (mysterious tone) or "James" (dramatic).
3. Adjust **settings**:
   - Speed: 85% (slower for suspense)

- Pitch: Slightly lower for eerie vibes
- Add “reverb” effect to mimic an empty mansion.

Option 3: I’ll Help You Edit
If you record your voice or generate an AI voiceover, send me the audio file, and I’ll help you:
- Trim pauses
- Add SFX/music cues
- Write timestamps for video sync

Example AI Voice Prompt for ElevenLabs
(Paste this into the tool):
Voice: "Ethan"
Style: "Eerie Storyteller"
Text: "They said the house held secrets... (pause 1s) but some doors should never open. She found the key in her grandmother’s journal... (pause 0.5s) hidden for decades."
Settings: Speed 80%, Pitch -10%, Reverb "Cathedral"

Let me know which option you prefer! I’ll guide you through the rest. 🔥

Here’s a **detailed breakdown of prompts for video generation tools** (MidJourney, Runway ML, Pika Labs, etc.) and **voice-over specifics** to bring your mysterious 1-minute story to life. I’ll structure this for maximum atmosphere and consistency:

VIDEO GENERATOR PROMPTS (Scene-by-Scene)
(Use tools like Runway ML, Pika Labs, or Kaiber for dynamic shots; MidJourney/DALL-E for specific images.)

Scene 1: Stormy Arrival
Prompt:
"Gothic mansion at night, hyper-realistic, rainstorm with horizontal rain streaks, lightning illuminating cracked stone walls, single lit window with flickering candlelight, dramatic low-angle shot, cinematic lighting, muted blues and greys with amber highlights, 8k, ultra-detailed, eerie atmosphere."
Tool Notes:
- Use Runway ML’s motion brush to animate rain and lightning.
- Add a "Dutch tilt" for unease.

Scene 2: Candlelit Exploration

Prompt:
"Close-up of a pale hand holding a dripping candle, shadows dancing on peeling floral wallpaper, hyper-detailed skin texture with rain droplets, chiaroscuro lighting, dark hallway with vanishing perspective, Kodak Ektachrome film grain, muted colors with warm candle glow."
Tool Notes:
- Animate flickering candlelight and shadow movement in Pika Labs.

Scene 3: The Key Revealed
Prompt:
"Extreme close-up of a rusted iron key with intricate engravings '1879', lying on a dusty mahogany desk, shallow depth of field, candlelight reflecting off metal, hyper-realistic texture, 35mm film, desaturated colors with amber highlights."
Tool Notes:
- Use MidJourney's "macro lens" style for texture detail.

Scene 4: Ascending the Stairs
Prompt:
"Spiral staircase in haunted mansion, low-angle shot, shadows shaped like skeletal hands on walls, moth-eaten wedding dress dragging on steps, dark cinematic tone, teal and black color grading, 4k, dramatic perspective."
Tool Notes:
- Animate slow upward camera movement in Kaiber.

Scene 5: Unlocking the Past
Prompt:
"Antique door with rusted lock, key turning slowly, sudden gust of wind extinguishing candle, pitch-black screen with faint glowing embers, 35mm horror film aesthetic, high contrast."

Scene 6: The Hidden Room
Prompt:
"Abandoned Victorian nursery, cobwebbed crib with torn lace, shattered porcelain dolls, dusty sunlight through cracked boards, faded portrait of a woman with empty eyes, hyper-detailed textures, muted sepia tones, Greg Rutkowski style painterly lighting."

Scene 7: Ghostly Encounter
Prompt:

"Haunted mirror reflection, translucent ghost woman in tattered wedding dress, hollow eyes, protagonist's terrified face in foreground, cinematic horror lighting, blue-green tint, film grain, subtle VFX distortion."
Tool Notes:
- Use Runway ML's "ghost effect" overlay for transparency.

Scene 8: The Truth Unfolds
Prompt:
"Rapid montage: vintage newspaper headline 'Bride's Suicide Scandal 1879', close-up of a noose swaying, baby's hand clutching a locket, flickering candlelight on tear-stained journal pages, 16mm film burn transitions."

Scene 9: The Heirloom
Prompt:
"Pearl necklace dissolving into dust in a trembling hand, golden dawn light breaking through storm clouds, tear droplets hitting floorboards, hyper-realistic detail, Kodak Portra color palette, soft focus."

Scene 10: Resolution
Prompt:
"Sunlit attic room, modern woman in jeans standing beside a restored portrait of a smiling Victorian woman, dust motes in light beams, warm golden hour tones, peaceful yet melancholic, 4k cinematic."

VOICE-OVER SPECIFICATIONS
(Use tools like ElevenLabs, Murf, or Amazon Polly for AI voice, or hire a VA.)

Tone & Style:
- Gender-neutral voice, mid-pitch, soft but urgent.
- Pace: Slow, with dramatic pauses (e.g., after "secrets...", "vanished.").
- Emotion: Haunted curiosity shifting to resolved determination.

Sample Line Delivery:
- **0:00 - 0:05:** Whispered, breathy: *"They said the house held secrets... (pause) but some doors should never open."*
- **0:35 - 0:45:** Faster, tense: *"She wasn't mad... (sharp pause) she was silenced."*
- **0:55 - 1:00:** Soft, hopeful: *"Some secrets... (gentle sigh) set you free."*

Technical Specs:
- Add slight reverb to mimic "empty mansion" acoustics.

- Layer faint whispers underneath key lines (e.g., *"Find me..."* during Scene 7).
- Sync voice-over pauses with SFX (e.g., thunder, creaking doors).

SOUND DESIGN PROMPTS
1. **Background Music:**
   - Search: *"Dark ambient music with cello drones, faint piano, and choir pads"* (e.g., Peter Gundry tracks).
   - Use Epidemic Sound or Artlist for royalty-free options.

2. **SFX to Layer:**
   - Rainstorm + thunder
   - Candle flicker + floorboard creaks
   - Rusty door screech
   - Ghostly lullaby (music box + reversed vocals)

SUMMARY
Tools to Use:
- **Video:** Runway ML (motion), MidJourney (stills), CapCut (editing).
- **Voice:** ElevenLabs (AI) or Voice123 (human VA).
- **Sound:** Epidemic Sound + FreeSound.org.

Key Parameters for Mystery:
- Lighting: **Chiaroscuro** (sharp light/shadow contrasts).
- Color: **Desaturated blues + amber accents**.
- Motion: **Slow zooms**, **static flickers**, **shadow play**.

This level of detail ensures your AI tools generate cohesive, atmospheric assets. Let me know if you need help stitching it all together!